PERBANDINGAN METODE SUPPORT VECTOR MACHINE DAN LOGISTIC REGRESSION PADA ANALISIS SENTIMEN ULASA APLIKASI MAXIM DI GOOGLE PLAY STORE

Oleh

MARDIANSYAH

T3121022

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana

PROGRAM SARJANA
TEKNIK INFORMATIKA
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2025

PERSETUJUAN SKRIPSI

PERBANDINGAN METODE SUPPORT VECTOR MACHINE DAN LOGISTIC REGRESSION PADA ANALISIS SENTIMEN ULASA APLIKASI MAXIM DI GOOGLE PLAY STORE

Oleh

MARDIANSYAH

T3121022

SKRIPSI

Untuk memenuhi salah satu syarat ujian guna memperoleh gelar Sarjana Program Studi Teknik Informatika, ini telah disetujui oleh Tim Pembimbing.

Gorontalo, April 2025

Pembimbing I	Pembimbing II
Yasin Aril Mustofa, M.Kom	**Sunarto Taliki, M.Kom**
NIDN. 0926088503	**NIDN. 0906058301**

PENGESAHAN SKRIPSI

PERBANDINGAN METODE SUPPORT VECTOR MACHINE DAN LOGISTIC REGRESSION PADA ANALISIS SENTIMEN ULASA APLIKASI MAXIM DI GOOGLE PLAY STORE

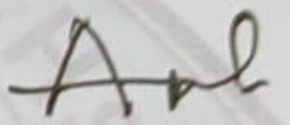
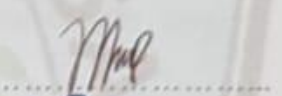
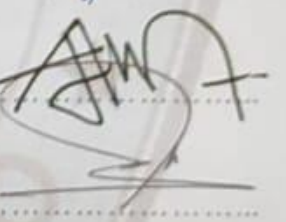
Oleh

MARDIANSYAH

T3121022

Diperikasa Oleh Panitia Ujian Strata Satu (S1)

Universitas Ichan Gorontalo

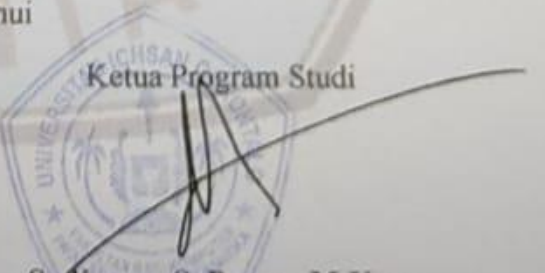
1. Ketua Penguji
   Amiruddin, M.Kom, MCF ..... 
2. Anggota
   Muis Nanja, M.Kom ..... 
3. Anggota
   Maryam Hasan, M.Kom ..... 
4. Anggota
   Yasin Aril Mustofa, M.Kom ..... 
5. Anggota
   Sunarto Taliki, M.Kom .....

Mengetahui

Dekan Fakultas Ilmu Komputer

Irvan Abraham Salihi, M.Kom
NIDN : 0918077302

Ketua Program Studi



Sudirman S. Panna, M.Kom
NIDN : 0924038025

PERNYATAAN SKRIPSI

Dengan ini saya menyatakan bahwa:

1. Karya tulis (Skripsi) saya ini adalah asli dan belum perna diajukan untuk mendapatkan gelar akademik (Sarjana) baik di Universitas Ichsan Gorontalo maupun di perguruan tinggi lainnya.
2. Karya tulis (skripsi) saya ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain,kecuali arahan dari Tim Pembimbing.
3. Dalam Karya tulis (Skripsi) saya ini tidak terdapat Karya atau pendapat yang telah dipublikasi orang lain, kecuali secara tertulis dicamtumkan sebagai acuan/sitasi dalam naskah dan dicamtumkan pula dalam daftar pustaka.
4. Peryataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam peryataan ini, makan saya sedia menerima sanki lainnya sesuai dengan norma-norma dengan yang berlaku di Universitas Ichsan Gorontalo.

Gorontalo, Mei 2025

Yang Membuat Peryataan,

MARDIANSYAH

ABSTRAK

MARDIANSYAH. T3121022. PERBANDINGAN METODE SUPPORT VECTOR MACHINE DAN LOGISTIC REGRESSION PADA ANALISIS SENTIMEN ULASAN APLIKASI MAXIM DI GOOGLE PLAY STORE

Penelitian ini bertujuan untuk menganalisis persepsi dan sentimen pengguna terhadap layanan aplikasi Maxim berdasarkan ulasan yang tersedia di Google Play Store serta membandingkan keakuratan metode *Support Vector Machine* (SVM) dan *Logistic Regression* dalam mengklasifikasikan sentimen tersebut. Data yang digunakan dalam penelitian ini berjumlah 10.000 ulasan pengguna yang diambil dari Google Play Store. Penelitian ini menggunakan pendekatan kuantitatif dengan metode deskriptif dan teknik *purposive sampling* dalam pemilihan data. Proses analisis dilakukan dengan membandingkan performa kedua algoritma menggunakan metrik evaluasi seperti akurasi, presisi, recall, dan F1-score. Hasil analisis menunjukkan bahwa mayoritas pengguna memberikan sentimen positif terhadap layanan aplikasi Maxim, yang ditunjukkan oleh frekuensi tinggi kata-kata seperti "driver", "ramah", dan "cepat". Dari sisi performa model, metode SVM menunjukkan hasil yang lebih unggul dengan akurasi sebesar 94,71%, sementara Logistic Regression memperoleh akurasi sebesar 92,04%. Selain itu, SVM juga menghasilkan nilai presisi, recall, dan F1-score yang lebih tinggi, serta menunjukkan kestabilan performa melalui evaluasi ROC AUC dan validasi silang, sehingga dapat disimpulkan bahwa SVM lebih efektif dalam mengklasifikasikan sentimen ulasan pengguna terhadap aplikasi Maxim.

Kata kunci: *SVM*, *Logistic Regression*, analisis sentimen, *Google Play Store*, aplikasi Maxim

ABSTRAK

MARDIANSYAH. T3121022. COMPARISON OF SUPPORT VECTOR MACHINE AND LOGISTIC REGRESSION METHODS IN SENTIMENT ANALYSIS OF MAXIM APPLICATION REVIEWS ON GOOGLE PLAY STORE

This study aims to evaluate user perceptions and sentiments regarding the services provided by the Maxim application, based on reviews from the Google Play Store. It compares the accuracy of the Support Vector Machine (SVM) and Logistic Regression methods in classifying these sentiments. It utilizes a dataset comprising 10,000 user reviews collected from the Google Play Store. Adopting a quantitative approach, the study employs the descriptive method and purposive sampling techniques for data selection. The analysis process involves assessing the performance of the two algorithms using evaluation metrics such as accuracy, precision, recall, and F1-score. The findings reveal that a majority of users express positive sentiments towards Maxim application services, highlighted by the frequent use of words such as "driver," "friendly," and "fast." In terms of model performance, the SVM method demonstrates superior results, achieving an accuracy of 94.71%, whereas Logistic Regression attained an accuracy of 92.04%. SVM also exhibits higher precision, recall, and F1-score values, alongside performance stability as shown in ROC AUC evaluations and cross-validation. It indicates that SVM is more effective in classifying user review sentiments about the Maxim application.

Keywords: SVM, Logistic Regression, sentiment analysis, Google Play Store, Maxim application

KATA PENGANTAR

Alhamdullilah penulis dapat menyelesaikan Usulan Penelitian ini dengan judul Perbandingan Metode *Support Vector Machine* dan *Logistic Regreesion* pada Analisis Sentimen Ulasan Aplikasi Maxim di *Google Play Store*, untuk memenuhi salah satu syarat Penyusunan Skripsi Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.

Penulis menyadari sepenuhnya bahwa skripsi ini tidak mungkin terwujud tanpa bantuan dan dorongan dari berbagai pihak baik bantuan moral maupun material. Untuk itu, dengan segala keikhlasan dan kerendahan hati, penulis mengucapkan banyak terima kasih dan penghargaan yang setinggi-tingginya kepada:

1. Bapak Dr. Abdul Gaffar Latjokke, M.Si, selaku Ketua Yayasan Pengembangan Ilmu Pengetahuan dan Teknologi (YPIPT) Ichsan Gorontalo.
2. Ibu Dr. Hj. Juriko Abdussamad, M.Si selaku Rektor Universitas Ichsan Gorontalo.
3. Bapak Irvan Abraham Salihi, M.Kom, selaku Dekan Fakultas Ilmu Komputer.
4. Bapak Sudirman Melangi, M.Kom, selaku Wakil Dekan I Bidang Akademik.
5. Ibu Irma Surya Kumala Idris, S.Kom, M.Kom, selaku Wakil Dekan II Bidang Administrasi Umum dan Ke.uangan Fakultas Ilmu Komputer Universitas Ichan Gorontalo.
6. Bapak Sudirman S.Panna, M.Kom selaku selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer.
7. Bapak Yasin Aril Mustofa, M.Kom, selaku Pembimbing Utama yang telah membimbing penulis selama mengerjakan Skripsi ini.
8. Bapak Sunarto Taliki, M.Kom, selaku Pembimbing Pendamping yang telah membimbing penulis selama mengerjakan Skripsi ini.
9. Bapak dan Ibu Dosen Universitas Ichsan Gorontalo yang telah mendidik dan mengajarkan berbagai ilmu kepada penulis
10. Orang Tua dan Saudara-saudara ku yang selama ini telah memberikan kasih sayang, Do'a dan selalu memberikan dorongan moral yang sangat besar

kepada penulis.

11. Teman-teman dan rekan seperjuangan di Jurusan Teknik Informatika yang telah membantu penulis dalam penyelesaian Skripsi ini.

Saran dan kritik, penulis harapkan dari dewan penguji dan semua pihak untuk penyempurnaan penulisan Skripsi. Semoga Skripsi ini dapat bermanfaat bagi pihak yang berkepentingan.

Gorontalo, April 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Transportasi online juga menjadi kebutuhan karena transportasi umum di beberapa daerah sulit dijangkau. Selain itu, transportasi online juga populer di daerah yang dilanda kemacetan, dengan transportasi online penumpang kini tidak perlu lagi mendekati ojek pangkalan atau tidak perlu lagi menunggu di pinggir jalan untuk mendapatkan taksi. Selain itu, penumpang juga tidak perlu terlibat dalam proses tawar-menawar karena tarif ditentukan berdasarkan jarak yang ditempuh [1]. Pemesanan ojek online bisa dilakukan pada smartphone masing – masing dan nantinya driver ojek online akan mendatangi titik jemput dari pemesanan yang tertera pada layanan ojek online aplikasi maxim.

Salah satu layanan transportasi online yang berkembang di Indonesia adalah Maxim, di Indonesia maxim pertama kali beroperasi pada tahun 2018. Perusahaannya psemakin melebarkan sayap dengan tidak hanya menjadi Perusahaan transportasi online yang berfokus ke taksi saja, melainkan juga jenis layanan angkutan lain seperti ojek atau mobil pada umumnya [2]. Maxim mengembangkan sendiri aplikasinya dan menciptakan sistem perangkat keras dan perangkat lunak yang memungkinkan mitra-mitranya terhubung kelayanan perusahaan, memperoses jutaan setiap hari, memantau kualitas kerja dan pelayanan, serta menganalisis dan mengoptimalkan bisnis mereka. Aplikasi Maxim menjadi pusat dari seluruh aktivitas operasional perusahaan, termasuk proses pemesanan, pelacakan lokasi, dan komunikasi antara pengguna dan pengemudi. Aplikasi ini juga menyediakan berbagai fitur yang dirancang untuk meningkatkan kenyamanan dan efisiensi layanan, seperti penjadwalan fleksibel bagi pengemudi, tarif yang kompetitif bagi pelanggan, serta sistem reservasi pemesanan. Melihat peran penting aplikasi ini dalam aktivitas bisnis Maxim, ulasan dan tanggapan pengguna terhadap aplikasi Maxim di *Google Play Store* menjadi sumber data yang sangat relevan dan penting untuk dianalisis dalam penelitian ini, khususnya dalam mengkaji persepsi dan kepuasan pengguna terhadap layanan yang diberikan.

Sebagai platform distribusi aplikasi, *Google Play Store* menyediakan fitur ulasan dan penilaian dari pengguna. Ulasan ini berfungsi sebagai sarana komunikasi antara pengguna dan pengembang, serta menjadi indikator kualitas layanan aplikasi [3]. Dengan menganalisis ulasan ini, perusahaan dapat memperoleh wawasan berharga untuk meningkatkan kualitas layanan, memahami kebutuhan dan keinginan pengguna, serta mengambil keputusan strategis yang lebih baik berdasarkan *feedback* langsung dari pengguna dan Hasil dari analisis ulasan ini diharapkan dapat mengidentifikasi tren sentimen pengguna, mengukur tingkat kepuasan pelanggan, mengungkapkan masalah atau kendala yang sering dihadapi pengguna, serta memberikan insight untuk pengembangan fitur atau perbaikan layanan yang lebih tepat sasaran [4].

Analisis sentimen merupakan studi komputasi opini-opini,sentimen,serta emosi yang diekspresikan dalam teks dan sebagai alat untuk mengklasifikasikan sebuah informasi yang berbentuk teks dalam kategori ulasan positif dan ulasan negatif pada aplikasi.[5] Untuk memantau dan menyortir ulasan tersebut sangat sulit dilakukan secara manual. Berdasarkan data terbaru 2024, aplikasi Maxim telah diunduh sebanyak 50 juta+ kali dengan 5,31 juta ulasan di *Google Play* [6]. Maka dari itu dibutuhkan sebuah metode yang bisa melakukan pekerjaan tersebut. Beranjak dari masalah di atas, Peningkatan Kualitas Layanan dengan mengetahui metode yang lebih akurat dalam analisis sentimen ulasan aplikasi Maxim, perusahaan dapat lebih cepat dan tepat merespon ulasan pelanggan, sehingga meningkatkan kualitas layanan yang diberikan. Dengan pemahaman yang lebih baik tentang sentimen pengguna, Maxim dapat membuat keputusan bisnis yang lebih baik dan strategis berdasarkan ulasan dan feedback pengguna. Adanya analisis sentimen yang lebih akurat, perusahaan dapat lebih memahami kebutuhan dan keinginan pengguna [7],

Ada beberapa metode komputasi yang dapat digunakan untuk menyelesaian masalah analisis sentimen antara lain: *Artificial Neural Network, Support Vector Machine*, *Learning Vector Quantization*, *K-Nearest Neighbor*, *Linear Discriminat Analysis* dan *Logistic Regreession* [7], Masing-masing metode memiliki karakteristik dan keunggulan tersendiri dalam mengklasifikasikan opini atau

sentimen dari teks ulasan pengguna. Pada penelitian [4] metode SVM *dan Logistic Regression* diterapkan untuk menganalisis sentimen pada ulasan pengguna aplikasi Google Meet. Hasilnya menunjukkan bahwa SVM memiliki akurasi lebih tinggi, yakni sebesar 87,02%, dibandingkan Logistic Regression yang digunakan dalam skenario yang sama. Sementara itu, dalam penelitian [8], Logistic Regression dan SVM digunakan untuk menganalisis sentimen pengguna Twitter terhadap layanan Steam. Penelitian tersebut menunjukkan bahwa Logistic Regression mampu mencapai akurasi sebesar 86%, yang membuktikan bahwa metode ini juga cukup andal dalam memodelkan variabel biner serta menangani data teks secara efektif. Berdasarkan kedua penelitian tersebut, dapat disimpulkan bahwa baik SVM maupun *Logistic Regression* memiliki performa yang kompetitif dan layak dibandingkan, sehingga menjadi dasar pemilihan kedua metode ini untuk dibandingkan lebih lanjut dalam penelitian ini.

Dengan berbagai pemaparan maka peneliti tertarik untuk melakukan penelitian dengan judul "Perbandingan Metode *Support Vector Machine* dan *Logistic Regression* pada Analisis Sentimen Ulasan Aplikasi Maxim di Google Play Store". Penelitian ini bertujuan untuk mengetahui metode yang lebih akurat dalam mengklasifikasikan sentimen ulasan pengguna aplikasi Maxim. Hasilnya diharapkan dapat menjadi referensi dalam pengembangan analisis sentimen yang efektif serta mendukung pengambilan keputusan strategis untuk meningkatkan kualitas layanan dan kepuasan pelanggan.

1.2 Identifikasi Masalah

Berdasarkan uraian diatas dapat di identifikasi masalah sebagai berikut:

1. Memantau dan menyortir ulasan secara manual sangat sulit dilakukan. Oleh karena itu, diperlukan metode yang dapat melakukan pekerjaan tersebut secara otomatis.
2. Berdasarkan perbedaan akurasi dari berbagai metode yang digunakan dalam penelitian sebelumnya, terdapat kebutuhan untuk membandingkan dua metode yang paling umum digunakan, yaitu SVM dan Logistic Regression, untuk menentukan metode yang paling akurat dalam analisis sentimen.

1.3 Rumusan Masalah

Berdasarkan latar belakng,maka permasalahan dapat dirumuskan yaitu sebagai berikut:

1. Bagaimana persepsi dan sentimen pengguna terhadap layanan aplikasi Maxim berdasarkan ulasan yang tersedia di Google Play Store?
2. Metode mana yang lebih akurat antara *Support Vector Machine* dan *Logistic Regression* dalam mengklasifikasikan sentimen ulasan pengguna aplikasi Maxim, yang dapat dibuktikan melalui perbandingan akurasi, presisi, recall, dan F1-score dari masing-masing metode pada dataset ulasan pengguna di Google Play Store?

1.4 Tujuan penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Untuk menganalisis persepsi dan sentimen pengguna terhadap layanan aplikasi Maxim berdasarkan ulasan yang tersedia di Google Play Store.
2. Untuk membandingkan keakuratan antara metode Support Vector Machine dan Logistic Regression dalam mengklasifikasikan sentimen ulasan pengguna aplikasi Maxim, yang dapat diukur melalui perbandingan akurasi, presisi, recall, dan F1-score dari kedua metode pada dataset ulasan pengguna di Google Play Store.

1.5 Manfaat penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Manfaat Teoritis: Penelitian ini memberikan kontribusi terhadap pengembangan ilmu pengetahuan di bidang analisis sentimen dan pembelajaran mesin, serta dapat menjadi referensi untuk penelitian selanjutnya yang berkaitan dengan klasifikasi teks dan analisis sentimen.
2. Manfaat Praktis: Penelitian ini dapat membantu Maxim dalam meningkatkan kualitas layanan dan pengambilan keputusan bisnis yang lebih baik berdasarkan pemahaman yang lebih akurat tentang sentimen pengguna.

BAB II
LANDASAN TEORI

2.1 Tinjauan Studi

Tabel 2. 1 Tinjauan Studi

No	Penelitian	Judul	Tahun	Hasil
1	Irma Surya Kumala Idris 1, Yasin Aril Mustofa 2, Irvan Abraham Salihi 3.[9]	Analisis Sentimen Terhadap Penggunaan Aplikasi Shopee Mengunakan Algoritma *Support Vector Machine* (SVM)	2023	Hasil penelitian menggunakan algoritma *Support Vector Machine* terbukti mampu menghasilkan kinerja yang cukup baik dengan hasil akurasi sebesar 98% dan f1-score sebesar 0.98 atau sebesar 98%.
2	Andry Novantika1,*, Sugiman 2.[4]	Analisis Sentimen Ulasan Pengguna Aplikasi VideoConference *Google Meet* menggunakan Metode SVM dan *Logistic Regression*	2022	Nilai akurasi yang didapatkan pada data ulasan aplikasi Google Meet untuk masing-masing kernel berturut-turut adalah 87,02%, 84,59%, 86,63% dan 71,12%, sementara untuk metode LogisticRegression sebesar 85,17%. Diperoleh kesimpulan bahwa algoritma terbaik dengan akurasi tertinggi adalahalgoritma SVM dengan kernel Linearyaitu sebesar 87,02%.
3	Muhammad Nur Akbar1, Nur Hasanahlmar'iyah Rusydi2, M. Hasrul H.3, NurulShaumi Ramadhanti4, Erfiana [1]	Sentiment Analysis Terhadap Review Aplikasi Maxim di Google Play Store Menggunakan Support Vector Machine (SVM)	2022	Penelitian ini bertujuan menganalisis review pengguna aplikasi maxim pada Google Play Store, menggunakan analisis sentimen. analisis review pengguna ini menggunakan metode Support Vector Machine (SVM) yang menghasilkan akurasi sebesar 79%.

No	Penelitian	Judul	Tahun	Hasil
4	Tamora Nonia Wijaya 1, Rini Indriati 2, Muhammad Najibulloh Muzaki 3.[10]	Analisis Sentimen Opini Publik Tentang Undang Undang Cipta Kerja Pada Twitter	2021	Performa terbaik yang diperoleh oleh Naive Bayes Classifier adalah akurasi sebesar 89.9%, precision sebesar 90%, recall sebesar 89.9%, dan f-1 score sebesar 89.9%. Hasil yang didapat menunjukkan bahwa masyarakat Indonesia 52.9% kontra dan 47.1% pro terhadap Undang-Undang Cipta Kerja.
5	Lutfi Budi Ilmawan a,1*dan Muhammad Aliyazid Mude b,2 [11]	Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store	2020	Hasil pengujian yang didapatkan dari metode 3-folds crossvalidation menghasilkan bahwa SVM Classifier memiliki nilai yang lebih tinggi jika dibandingkan dengan akurasi dari Naïve Bayes classifier untuk mengklasifikasikan ulasan tekstual berbahasa Indonesia pada Google Play Store, yakni SVM classifier mendapatkan akurasi sebesar 81,46% dan Naïve Bayes classifier sebesar 75,41%.
6	Edo Ridho Lidinillah 1, Tatang Rohana 2, Ayu Ratna Juwita 3.[8]	Analisis sentimen twitter terhadap steam menggunakan algoritma logistic regression dan support vector machine	2023	Dalam penelitian ini data yang terkait menggunakan platform Steam dikumpulkan sampai 4363 data ulasan positif dan negatif terhadap platform steam dalam twitter, Untuk menghitung menggunakan metode Confusion Matrix dalam 2 algoritma yang berbeda buat perbandingan, algoritma yang digunakan adalah Logistic Regression dan Support Vector Machine. Dari percobaan perhitungan 2 metode itu diketahui bahwa algoritma Super Vector Machine mendapatkan nilai yang optimal dengan accuracy 0.81, precision 0.85 serta recall 0.77.

2.2 Tinjauan Pustaka

2.2.1 Transportasi Online

Transportasi online adalah pelayanan jasa transportasi yang berbasis internet dalam setiap kegiatan transaksinya, Selain itu, transportasi online juga populer di daerah yang dilanda kemacetan, dengan transportasi online penumpang kini tidak perlu lagi mendekati ojek pangkalan atau tidak perlu lagi menunggu di pinggir jalan untuk mendapatkan taksi. Selain itu, penumpang juga tidak perlu terlibat dalam proses tawar-menawar karena tarif ditentukan berdasarkan jarak yang ditempuh.[1] Transportasi online menawarkan kemudahan, biaya yang lebih murah, kenyamanan dan keamanan yang lebih terjamin, maka tidak mengherankan jika banyak orang yang beralih dari moda transportasi konvensional ke moda transportasi online. Seiring dengan waktu, kehadiran transportasi online ini menimbulkan kecemburuan sosial bagi transportasi konvensional yang sudah ada sebelumnya, baik ojek, taksi, bus dan lain sebagainya. Transportasi online dituding sebagai biang kerok menurunnya pendapatan para pengemudi transportasi konvensional. Aksi protes, penolakan, penghadangan dan puncaknya adalah demo besar-besaran yang menolak kehadiran Gojek, Maxim dan Grab dilakukan oleh para pengemudi transportasi konvensional. Salahkah dengan adanya aplikasi online di bidang transportasi ini? Tentu saja tidak, karena kemajuan teknologi adalah sesuatu yang tidak bisa kita hindari dalam kehidupan ini. Dari aksi protes yang dilakukan pengemudi transportasi konvensional, melahirkan larangan beroperasi bagi perusahaan transportasi berbasis online melalui Keputusan Menteri Perhubungan Nomor UM.302/1/21/Phb/2015 karena dianggap bertentangan dengan Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan, namun kemudian Keputusan Menteri ini dicabut karena pernyataan Presiden bahwa alat transportasi berbasis aplikasi online masih dibutuhkan oleh masyarakat.[12]

2.2.2 Maxim

Maxim adalah perusahaan transportasi online asal Rusia, tepatnya di Chardnisk, pegunungan Ural yang sudah beroperasi sejak tahun 2003. Di Indonesia, Maxim pertama kali beroperasi pada tahun 2018. Perusahaannya semakin melebarkan sayap dengan tidak hanya menjadi Perusahaan transportasi online yang berfokus ke taksi saja, melainkan juga jenis layanan angkutan lain seperti ojek atau mobil pada umumnya. Maxim mengembangkan sendiri aplikasinya dan menciptakan system perangkat keras dan perangkat lunak yang memungkinkan mitra-mitranya terhubung kelayanan perusahaan, memperoses jutaan setiap hari, memantau kualitas kerja dan pelayanan, serta menganalisis dan mengoptimalkan bisnis mereka. Maxim memiliki kebijakan yang menguntungkan bagi pengemudi maupun pelanggan, seperti jadwal yang fleksibel untuk pengemudi, harga terjangkau serta sistem reservasi order untuk pelanggan.[5]

2.2.3 Google Play Store sebagai

Google play adalah sebagai wadah pengunduh aplikasi ini memiliki fitur yang menarik yaitu sebuah fitur yang dapat memberikan rating dan ulasan dari para pengguna aplikasi. Fungsi dari adanya ulasan tersebut, yaitu bisa digunakan sebagai tolak ukur yang cukup efektif dan efisien menemukan informasi. Ulasan tersebut bisa memberikan saran positif dan negatif.[1]

2.2.4 Analisis Sentimen

Analisi sentimen merupakan studi komputasi opini-opini,sentimen,serta emosi yang diekspresikan dalam teks dan sebagai alat untuk mengklasifikasikan sebuah informasi yang berbentuk teks dalam kategori ulasan positif dan ulasan negatif pada aplikasi.[5] Analisis Sentimen penting dilakukan dengan perkembangan yang pesat suatu opini di media sosial dan berbagai diskusi di ulasan atau suatu review produk dan layanan, karena pertumbuhan era digital yang sangat pesat membuat masyarakat pada zaman modern tidak dapatlepas dari penggunaan teknologi, salah satunya yaitu media sosial membuat orang mengungkapkan dan menuliskan pendapat yang tertulis dapat melalui media sosial atau melalui suatu ulasan sehingga analisis sentiment dapat dilakukan.[3]

Sentimen positif memberikan nilai yang baik, sementara sentiment negative memberikan nilai buruk dalam kontek teks. Penilitian di bidang dengan sentimen dari suatu data menjadi sangat penting dan dibutuhkan di era big data seperti saat ini. Analisis sentiment adalah tren tekini di bidang Natural Languange Processing (LNP).

2.2.5 Text Mining

Text Mining merupakan proses mengekstraksi suatu pola untuk diteliti yang datanya berasal dari sebuah teks. *Text Mining* merupakan disiplin ilmu yang didasarkan pada *information retrival, data mining*, *machine learning, statistic dan linguistic komputasi.*[9]. *Teks mining* digunakan dalam Teknik seperti kategorisasi. Eksraksi identitas, dan analisis sentimen untuk mengidentifikasi wawasan dan tren dalam volume data dasar yang tidak terstuktur atau semi-terstuktur. Objek dari teks mining ini biasanya berupa dokumen yang tidak terstruktur atau semi-terstruktur, dan proses informasi pada teks mining dapat menghasilkan analisis perasaan yang. Tahapan utama dalam text mining meliputi preprocessing data, seperti menghapus simbol, angka, dan kata tidak relevan (stopwords), serta mengubah teks menjadi representasi numerik menggunakan teknik seperti TF-IDF atau word embeddings. Selanjutnya, model pembelajaran mesin diterapkan untuk menganalisis data sesuai dengan kebutuhan, seperti klasifikasi atau klasterisasi, dengan hasil yang dievaluasi menggunakan metrik seperti akurasi dan F1-score.

2.2.6 Ekstraksi Fitur

Ekstraksi fitur dilakukan untuk mengubah data teks menjadi representasi numerik yang dapat digunakan oleh algoritma pembelajaran mesin, seperti menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*).[5] Rumus TF-IDF adalah:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t) \quad (2.\ 1)$$

imana:

- $TF(t, d)$ adalah frekuensi istilah t dalam dokumen d,

- $IDF(t)$ adalah logaritma dari total jumlah dokumen dibagi dengan jumlah dokumen yang mengandung istilah t.

2.2.7 Support Vector Machine

Support Vector Machine merupakan salah satu metode pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. SVM dikembangkan oleh Vapnik dan koleganya pada awal tahun 1990-an dan dikenal sebagai salah satu algoritma yang memiliki performa tinggi terutama dalam permasalahan klasifikasi biner. Prinsip dasar SVM adalah mencari *hyperplane* terbaik yang dapat memisahkan data ke dalam dua kelas berbeda dengan margin maksimal [9]. SVM bekerja dengan mengubah data input ke dalam ruang berdimensi lebih tinggi menggunakan fungsi kernel, lalu mencari hyperplane optimal yang memisahkan kelas dengan margin terlebar. Titik-titik data yang berada paling dekat dengan hyperplane disebut sebagai support vector, yang berperan penting dalam menentukan posisi hyperplane tersebut Dalam kasus klasifikasi linier, SVM mencari garis pemisah (hyperplane) yang dapat memaksimalkan jarak antara dua kelas. Sedangkan untuk data yang tidak dapat dipisahkan secara linier, SVM menggunakan fungsi kernel seperti *linear, polynomial, radial basis function* (RBF), atau *sigmoid* untuk memetakan data ke ruang fitur yang lebih tinggi agar menjadi separable. SVM memiliki keunggulan dalam menghadapi data berdimensi tinggi dan juga mampu menangani kasus di mana jumlah fitur lebih besar daripada jumlah data. Selain itu, SVM juga cukup efektif pada dataset yang memiliki noise kecil dan data yang tidak terlalu besar secara ukuran

2.2.8 Penerapan Algoritma *Support Vector Machine*

Berikut ini contoh penerapan algoritma Support Vector Machine pada penelitian [13] yang dapat dilihat pada beberapa tahapan di bawah ini.

1. Pengumpulan data

Data ulasan yang dikumpulkan menggunakan web scrapping berjumlah 279.877 review yang paling relevan dalam rentang waktu dari bulan Februari 2018 hingga bulan November 2022, yang ditampilkan pada tabel berikut.

Tabel 2. 2 *Scrapping* Data [13]

No	User	Score	Comment
1	Pengguna Google	5	cukup bagus apk nya bpjs Sangat cepat dan mudah
2	Pengguna Google	1	Bintang setengah ada ga?
3	Pengguna Google	5	Sangat cepat dan mudah
…	…	…	…
27984	Pengguna Google	3	Coba dlu nnti bintangny d tmbh klo bgs
27985	Pengguna Google	5	Cucokk mantep
27986	Pengguna Google	5	mantap gannnn

2. Preprocessing

Setelah pengumpulan data, kemudian tahap pra pemrosesan sebelum digunakan sebagai data latih dan data uji. Beberapa langkah dalam *pre-processing* yaitu *case folding, tokenizing, stop words* dan *normalization.*

a. Case folding

proses case folding, yaitu adanya perubahan kalimat atau kata menjadi lowercase (huruf kecil) ditampilkan pada Tabel berikut

Tabel 2. 3 *Case Folding*

Sebelum	Sesudah
cukup bagus apk nya bpjs	cukup bagus apk nya bpjs
Sangat cepat dan mudah	sangat cepat dan mudah
Bintang setengah ada ga?	bintang setengah ada ga?

b. Tokenizing

Hasil dari proses tokenizing, yaitu memisahkan setiap kata menjadi satuan kata dan menghilangkan simbol tertentu seperti tanda baca ditampilkan pada Tabel berikut.

Tabel 2. 4 Tokenizing

Sebelum	**Sesudah**
cukup bagus apk nya bpjs	[cukup, bagus, apk, nya, bpjs]
Sangat cepat dan mudah	[sangat, cepat, dan, mudah]
Bintang setengah ada ga?	[bintang, setengah, ada, ga]

c. Stopwords

Hasil dari proses stop words yaitu menghilangkan kata-kata yang tidak perlu atau tanpa tujuan dari teks ditampilkan pada Tabel

Tabel 2. 5 *Stopwords*

Sebelum	Sesudah
cukup bagus apk nya bpjs	[bagus, apk, bpjs]
Sangat cepat dan mudah	[cepat, mudah]
Bintang setengah ada ga?	[bintang, ga]

d. Normalization

Hasil dari proses normalization yaitu menghapus atau menghilangkan kata-kata yang memiliki imbuhan atau tambahan, dan membenarkan pengejaan yang kurang tepat menjadi kata yang sesuai ditampilkan pada Tabel berikut.

Tabel 2. 6 *Normalization*

Sebelum	**Sesudah**
cukup bagus apk nya bpjs	bagus aplikasi bpjs
Sangat cepat dan mudah	cepat mudah
Bintang setengah ada ga?	bintang, gak

3. Pelabelan Data

Setelah *pre-processing* digunakan metode *lexicon-based* untuk pelabelan tiap data yaitu pemberian skor terhadap setiap fitur. Beberapa hasil pelabelan *lexicon-based* ditampilkan pada Tabel berikut

Tabel 2. 7 Pelabelan Data

Ulasan	Nilai	Kategori
bagus apk nya bpjs	0,4125	Positif
Sangat cepat dan mudah	0,4404	Positif
Bintang setengah ada ga?	-0,2960	Negatif

4. Klasifikasi

Hasil klasifikasi algoritma SVM menggunakan pendekatan kernel linear dengan proporsi data latih dan data uji sebesar 0,8:0,2 Pengujian model ini didapatkan matriks konfusi yang ditampilkan pada Tabel dibawah ini. Tabel menunjukkan hasil klasifikasi dari kategori ulasan aplikasi JMO, didapatkan nilai precision, recall, F1-score dan support yang paling tinggi adalah ulasan berkategori positif berjumlah 17.571 dengan persentase 58,64%, sedangkan untuk ulasan negatif berjumlah 3.876 dengan persentase 12,58%, dan ulasan netral berjumlah 8.701 dengan persentase 28,78%. Yang menunjukkan adanya keseimbangan data, hasil klasifikasi ketiga ulasan sebagai berikut.

Tabel 2. 8 Hasil Klasifikasi

Kategori	Precision	Recall	F1-Score	Support
Negatif	0,82	0,94	0,88	3.876
Netral	0,98	0,99	0,99	8.701
Positif	0,99	0,95	0,97	17.571

5. Hasil Evaluasi

Hasil dari tahap evaluasi digunakan untuk menentukan nilai uji dari jenis yang berhasil diuji dengan menghitung satu ukuran spesifik terhadap dataset pengujian, yaitu data yang tidak digunakan dalam fase pengembangan model. Tabel dibawah

ini menunjukkan hasil pengujian menggunakan metode SVM, didapatkan nilai akurasi yang sangat tinggi yaitu 96%, disebabkan karena adnya penambahan kernel linear dan perbandingan data uji serta data latih yang sesuai. Dengan nilai presisi 92%, nilai keberhasilan suatu model 96% serta nilai perbandingan rata-rata antara presisi dan recall 94%

Tabel 2. 9 Hasil Evaluasi

Accuracy	**Precission**	**Recall**	**F1-Score**
0,96	0,92	0,96	0,94

Berdasarkan klasifikasi yang telah dilakukan, menunjukkan bahwa metode SVM terhadap analisis sentiment aplikasi Jamsostek Mobile memiliki hasil klasifikasi terbaik dengan akurasi sangat tinggi yaitu accuracy 0,96, sehingga dapat dikatakan bahwa aplikasi JMO ini merupakan aplikasi yang baik dan bagus untuk digunakan bagi para pengguna dan dapat sebagai bahan informasi bagi calon pengguna.

2.2.9 Logistic Regression

Logistic regression merupakan metode statistik yang digunakan untuk memodelkan hubungan antara satu atau lebih variabel independen dengan variabel dependen yang bersifat kategorikal (biasanya biner: 0 dan 1) . Metode ini sangat populer dalam klasifikasi karena kemampuannya untuk memprediksi probabilitas suatu peristiwa berdasarkan data input [14]. Berbeda dengan regresi linier yang hasil prediksinya berupa nilai kontinu, Logistic Regression menghasilkan nilai probabilitas antara 0 dan 1 melalui fungsi logistik (*sigmoid*). Fungsi ini membatasi *output* agar tetap berada dalam rentang probabilitas.

Fungsi logistik atau sigmoid dirumuskan sebagai:

Rumus dasar dari Logistic Regression adalah:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_1 x_2 + \cdots + \beta_n x_n)}} \qquad (2.\ 2)$$

Dimana:

- $P(y = 1|x)$ adalah probabilitas bahwa label y adalah 1 untuk input x
- β_0 adalah intecept

- $\beta_1, \beta_2 \ldots \beta_n$ adalah koefisien regresi
- $x_1, x_2 \ldots x_n$ adalah nilai dari fitur input
- e adalah bilangan eksponensial (sekitar 2,718)

Model *Logistic Regression* mengklasifikasikan data berdasarkan ambang batas (*threshold*), biasanya 0.5. Jika hasil dari fungsi *sigmoid* di atas lebih besar dari 0.5, maka prediksi kelasnya adalah 1 (positif); jika kurang dari 0.5, maka kelasnya adalah 0 (negatif). *Logistic Regression* banyak digunakan dalam analisis sentimen karena mampu menangani data teks yang telah direpresentasikan dalam bentuk numerik (misalnya melalui ekstraksi fitur TF-IDF), serta menghasilkan model yang dapat diinterpretasikan secara statistik

2.2.10 Penerapan Algoritma Logistik Regression

Berikut ini contoh penerapan algoritma logistic regression pada penelitian [15] yang dapat dilihat pada beberapa tahapan berikut.

1. Analisa data

   Data masukan dari penelitian ini berupa judul dari berbagai portal berita online dengan topik pemilu 2019 Dengan 395 record yang selanjutnya dilakukan proses labeling. Adapun hasil dari proses labeling tersebut didapatkan label positif berjumlah 215 record, data dengan label negatif berjumlah 85 record dan data dengan label netral berjumlah 95 record.

2. *Text pre-processing*

   Dalam proses *text mining*, teks dokumen yang digunakan harus dipersiapkan terlebih dahulu sebelum dapat digunakan untuk proses utama. Proses mempersiapkan dataset mentah disebut juga dengan proses *text preprocessing*. Text preprocessing berfungsi untuk mengubah data teks yang tidak terstruktur atau sembarang menjadi data yang terstruktur [15]. *Preprocessing* dilakukan untuk menghindari dataset yang kurang sempurna, terdapat noise pada dataset, data-data yang tidak konsisten dan mempercepat pemrosesan terhadap dokumen, alur text preprocessing yang *meliputi case folding, remove punctuation, stemming,* pembobotan kata. Berikut merupakan tabel hasil preprocessing.

Tabel 2. 10 Hasil Preprocessing[15]

Text	Hasil
Benarkah Polsek Tanjung Bumi Tak GubrisLaporan Tim Farid Alfauzi?	benar polsek tanjung bumi tak gubris lapor tim farid alfauzi
Bila Pendaftaran Dibuka, Farid Alfauzi Sudah Bisa Mendaftar Pilkada Bangkalan	bila daftar buka farid alfauzi daftar pilka da bangkal
Ratusan Baleho Bakal Calon Bupati Bangkalan Dirusak	ratus baleho bakal calon bupati bangkal rusak
Serikat Buruh Sepakat Deklarasi Damai Pemilu 2019	serikat buruh sepakat deklarasi damai pemilu 2019
Antisipasi Rawan Pelanggaran, Bawaslu Minta Masyarakat Aktif	antisipasi rawan langgar bawaslu minta masyarakat aktif

3. Handling Imbalanced Data

diketahui bahwa jumlah kelas pada penelitian ini tidakseimbang dimana kelas positif memiliki prosentase yang lebih banyak dibanding kelas negative dan netral. Sehingga perlu dilakukan *handling imbalanced* data agar model yang dihasilkan tidak condong ke satu kelas. Pada penelitian ini, peneliti melakukan *oversampling* menggunakan SMOTE. SMOTE merupakan Teknik yang digunakan untuk menyeimbangkan jumlah distribusi data pada kelas minoritas dengan cara meyeleksi data sampel tersebut hingga jumlah datanya menjadi seimbang dengan jumlah kelas mayoritas [15]. Adapun hasil dari oversampling tersebut adalah berikut

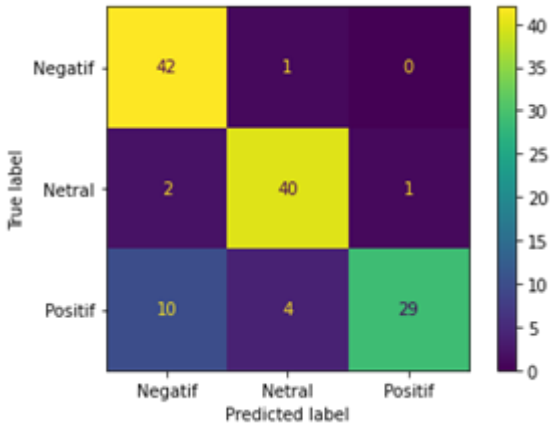
Kelas	Jumlah kelas sebelum oversampling	Jumlah kelas setelah oversampling
Positif	215	215
Negative	85	215
Netral	95	215

4. Klasifikasi Logistic Regression

Setelah melalui seluruh tahapan text *preprocessing* dan *oversampling*, selanjutnya dilakukan klasifikasi dengan salah satu algoritma *machine learning* yaitu *logistic regression*. Data dibagi menjadi dua bagian yaitu data latih dan data uji dengan prosentase 80% untuk data latih dan 20% untuk data uji. setelah data dibagi menjadi dua bagian, selanjutnya peneliti melakukan hyperparameter tuning untuk mendapatkan parameter terbaik. Teknik yang digunakan untuk *hyperparameter tuning* adalah *randomized search cross validation* dimana teknik ini dapat melakukan tuning parameter dengan waktu yang lebih singkat dibanding *grid search cross validation* [15]. Hasil dari tuning tersebut mendapatkan parameter terbaik yaitu C=493.529 dan *fit_intercept = False*. Sehingga dari kombinasi metode tersebut didapatkan skor latih 98% dan skor uji 86%.

5. Evaluasi

Hasil dari tahapan klasifikasi selanjutnya dilakukan evaluasi untuk mengetahui seberapa besar skor akurasi, *precision*, *recall* dan *f1 score*. Berdasarkan gambar dibawah ini, skor akurasi yang dihasilkan pada penelitian ini adalah 86%. Untuk menampilkan *confusion matrix*, peneliti menggunakan *library scikit learn* dengan memanggil fungsi *metrics*. Sedangkan untuk menampilkan perhitungan dari *confusion matrix*, peneliti menggunakan *library scikit learn* dengan memanggil fungsi *classification_report*.



Gambar 2. 1 Confusion Matriks [15]

	precision	recall	f1-score	support
0	0.78	0.98	0.87	43
1	0.89	0.93	0.91	43
2	0.97	0.67	0.79	43
accuracy			0.86	129
macro avg	0.88	0.86	0.86	129
weighted avg	0.88	0.86	0.86	129

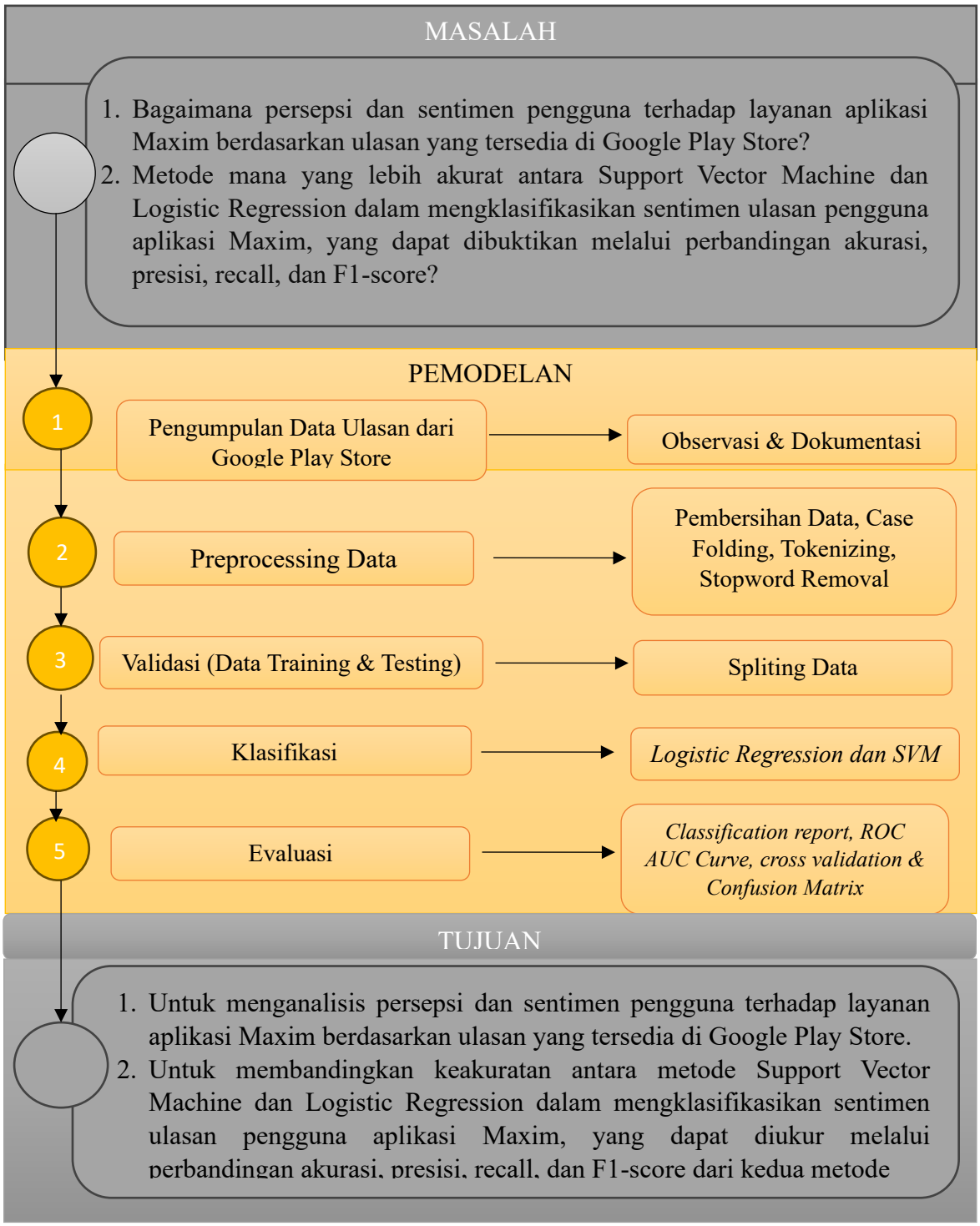
Gambar 2. 2 *Clasification Report* [15]

2.3 Perangkat lunak Pendukung

Berikut ini merupakan perangkat lunak yang digunakan dalam penelitian:

No	Nama Perangkat Lunak	Fungsi Penggunaan
1	Python	Bahasa pemrograman utama yang digunakan untuk proses preprocessing, pelatihan model, dan evaluasi.
2	Jupyter Notebook	Lingkungan pengembangan interaktif untuk menulis dan menjalankan kode Python.
3	Scikit-learn	Library machine learning yang digunakan untuk implementasi model SVM dan Logistic Regression serta evaluasi model.
4	Pandas	Library untuk manipulasi dan analisis data, khususnya dalam bentuk tabel (dataframe).
5	NumPy	Digunakan untuk komputasi numerik dan operasi array multidimensi.
6	Matplotlib & Seaborn	Library visualisasi data untuk menampilkan grafik hasil analisis.

2.4 Kerangka Pikir



Gambar 2. 3 Kerangka Pikir

BAB III
METODE PENELITIAN

3.1 Jenis, Metode, Subjek, Objek, Waktu, Dan Lokasi Penilitian

Dipandang dari Tingkat penerapanya, Penelitian ini merupakan penelitian kuantitatif dengan pendekatan analisis sentimen. Analisis dilakukan untuk mengetahui sentimen positif, dan negatif, dari ulasan aplikasi Maxim pada *Google Play Store* menggunakan metode *Logistic Regression* dan *Support Vector Machine* (SVM). Penelitian ini menggunakan metode analisis data sekunder, dengan demikian jenis penelitian ini adalah teknik *machine learning.* Subjek penelitian ini adalah ulasan pengguna aplikasi Maxim di *Google Play Store*. Pada Objek penelitian ini adalah ulasan dan rating aplikasi Maxim di *Google Play Store.* Penelitian ini dimulai dari Juli 2023 sampai dengan Januari 2024, Penelitian dilakukan secara *online* dengan pengambilan data ulasan dari *Google Play Store,* analisis data dilakukan di laptop Lenovo.

3.2 Pengumpulan Data

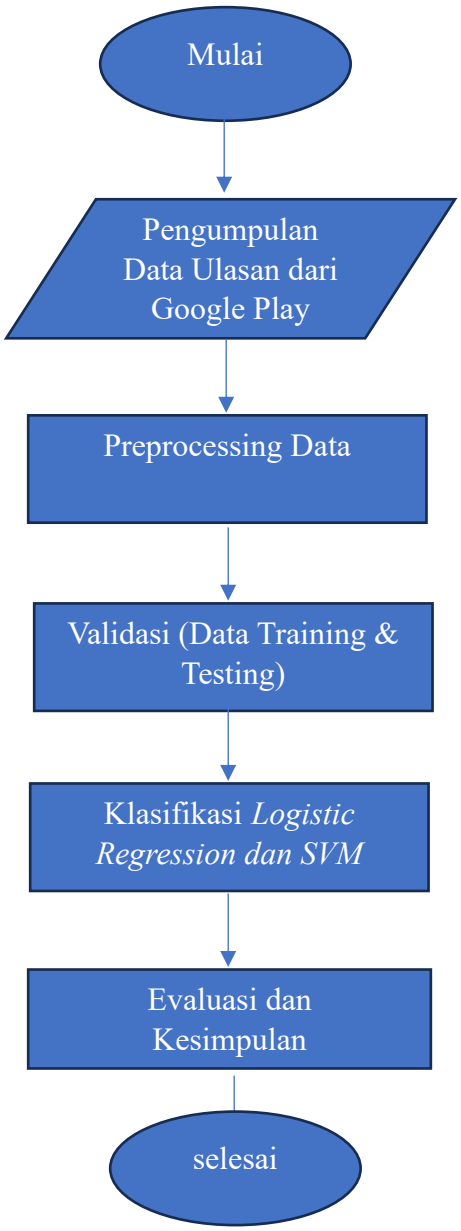
Data yang digunakan dalam penelitian ini berasal dari ulasan pengguna aplikasi Maxim yang tersedia di Google Play Store. Ulasan ini memberikan berbagai informasi tentang pengalaman pengguna, baik positif maupun negatif, yang akan digunakan sebagai basis untuk analisis sentimen.

3.3 Teknik Pengambilan Data

Teknik pengambilan data dalam penelitian ini dilakukan secara dokumentasi dengan memanfaatkan data ulasan pengguna aplikasi Maxim yang tersedia secara publik di Google Play Store. Data diambil menggunakan teknik web scraping untuk mengumpulkan ulasan berupa teks beserta label sentimennya, yang selanjutnya digunakan sebagai dataset dalam proses analisis sentimen.

3.4 Pemodelan

Model tahapan penelitian yang diusulkan ditunjukkan pada Gambar berikut ini:



Gambar 3. 1 Tahapan Penelitian

3.4.1 Pra pengolahan data

Sebelum data diolah, terlebih dahulu di lakukan: *Case folding*, *Spelling Normalization, Filtering, Tokenizing.* Hal ini dilakukan karena Mengubah semua teks ulasan menjadi huruf kecil untuk konsistensi. Memperbaiki kata-kata yang

salah eja dan mengubah singkatan menjadi bentuk lengkapnya. Menghapus kata-kata yang tidak penting (stopwords), simbol, angka, dan tanda baca. Dan Membagi teks ulasan menjadi potongan kata atau token.

3.4.2 Validasi (Data Training & Testing)

Data ulasan yang telah dipraolah kemudian dibagi menjadi dua set, yaitu data training dan data testing dengan perbandingan 80:20. Data *training* digunakan untuk melatih model machine learning, sedangkan data *testing* digunakan untuk menguji kinerja model.

3.4.3 Pengembangan Model

Pengembangan model dilakukan setelah data ulasan dari *Google Play Store* dikumpulkan dan diproses melalui tahap prapemrosesan. Proses pengembangan dimulai dengan pembagian data menjadi data training dan testing (validasi), kemudian dilakukan proses klasifikasi menggunakan dua algoritma machine learning, yaitu *Logistic Regression* dan *Support Vector Machine* (SVM). Model yang telah dilatih kemudian digunakan untuk memprediksi sentimen ulasan. Setiap hasil prediksi akan dievaluasi menggunakan metrik evaluasi *classification report,* dan *confusion matrix* untuk membandingkan performa masing-masing metode klasifikasi.

3.4.4 Evaluasi

Evaluasi dilakukan untuk menilai performa model klasifikasi yang telah dibangun menggunakan algoritma *Logistic Regression* dan *Support Vector Machine* (SVM). Evaluasi ini menggunakan metrik pengukuran seperti akurasi*, precision, recall, F1-score*, dan *confusion matrix*. Metrik tersebut digunakan untuk mengetahui seberapa baik model dalam mengklasifikasikan sentimen ulasan pengguna terhadap aplikasi Maxim, serta menjadi dasar dalam membandingkan performa dari kedua metode yang diuji.

BAB IV
HASIL PENELITIAN

4.1 Hasil Pengumpulan Data

Pada tahap awal penelitian, dilakukan pengumpulan data ulasan aplikasi Maxim yang terdapat di Google Play Store. Data yang berhasil dikumpulkan berjumlah sebanyak 10.000 ulasan. Setiap ulasan kemudian diberi pelabelan sentimen menggunakan metode *lexicon-based*, yang mengkategorikan ulasan tersebut ke dalam sentimen positif dan negatif, berdasarkan kamus kata yang telah ditentukan. Proses pelabelan ini bertujuan untuk menyediakan data berlabel yang akan digunakan sebagai dasar dalam perbandingan performa *metode Support Vector Machine* dan *Logistic Regression* pada analisis sentimen.

```python
from google_play_scraper import Sort, reviews
import pandas as pd
```

Gambar 4. 1 Coding Impor Library

Penjelasan gambar 4.1 pada berikut ini.

1. google_play_scraper: Library untuk melakukan scraping data aplikasi dari Google Play.
2. Sort: Digunakan untuk menentukan urutan pengambilan ulasan (misalnya berdasarkan relevansi atau terbaru).
3. reviews: Fungsi untuk mengambil ulasan pengguna dari suatu aplikasi.
4. pandas: Digunakan untuk menyimpan dan memproses data ulasan dalam bentuk tabel (DataFrame).

```python
all_reviews, _ = reviews(
    'com.taxsee.taxsee',
    lang='id',
    country='id',
    sort=Sort.MOST_RELEVANT,
    count=10000
)
```

Gambar 4. 2 Coding Ambil Ulasan Aplikasi

Penjelasan gambar 4.2 pada berikut ini.

1. 'com.taxsee.taxsee': ID unik aplikasi Maxim di Google Play.
2. lang='id': Hanya ambil ulasan yang ditulis dalam bahasa Indonesia.
3. country='id': Fokus pada ulasan dari pengguna di Indonesia.
4. sort=Sort.MOST_RELEVANT: Ambil ulasan yang dianggap paling relevan oleh sistem Google Play, bukan berdasarkan waktu terbaru.
5. count=10000: Ambil maksimal 10.000 ulasan pengguna.
6. all_reviews: List data ulasan (berisi banyak dictionary).

```python
data = [{
    'nama': review['userName'],
    'tgl': review['at'],
    'komentar': review['content'],
    'score': review['score'],
} for review in all_reviews]
```

Gambar 4. 3 Coding Strukturisasi ke DataFrame

Penjelasan gambar 4.3 pada berikut ini.

1. Bagian ini melakukan konversi hasil ulasan menjadi struktur tabular (baris dan kolom) yang siap dianalisis:
2. nama: Nama pengguna yang memberikan ulasan.
3. tgl: Tanggal ulasan dikirimkan.
4. komentar: Isi ulasan atau feedback pengguna.
5. score: Skor rating dari pengguna (1 sampai 5).

```
df = pd.DataFrame(data)
df.head()
```

	nama	tgl	komentar	score
2169	Pengguna Google	2025-05-06 23:30:12	saya langganan tetap tapu tolong donk untuk se...	5
8240	Pengguna Google	2025-05-06 22:08:48	Abang sangat ramah dah baik .. terimakasih abang	5
597	Pengguna Google	2025-05-06 21:53:46	cepat, tanggal dan pelayanan warrr biasa Terim...	5
6311	Pengguna Google	2025-05-06 21:27:48	pengemudi sering membatalkan jadi lama nyari p...	4
6807	Pengguna Google	2025-05-06 20:07:48	drivernya baik,perjalnan lancar tpt wktu	5

Gambar 4. 4 Coding menampilkan data

Penjelasan Gambar 4.4 dilihat pada bawah ini.

1. Fungsi pd.DataFrame() digunakan untuk membuat DataFrame, yaitu struktur data berbentuk tabel (baris dan kolom), dari list of dictionary data yang telah dibuat sebelumnya.
2. Setiap dictionary di dalam list data akan menjadi satu baris, dan key dari dictionary (nama, tgl, komentar, score) menjadi nama kolom.

```
data_mentah = df[['nama','tgl','komentar','score']]
data_sortir = data_mentah.sort_values(by='tgl',ascending=False)
data_sortir.head()
```

	nama	tgl	komentar	score
2169	Pengguna Google	2025-05-06 23:30:12	saya langganan tetap tapu tolong donk untuk se...	5
8240	Pengguna Google	2025-05-06 22:08:48	Abang sangat ramah dah baik .. terimakasih abang	5
597	Pengguna Google	2025-05-06 21:53:46	cepat, tanggal dan pelayanan warrr biasa Terim...	5
6311	Pengguna Google	2025-05-06 21:27:48	pengemudi sering membatalkan jadi lama nyari p...	4
6807	Pengguna Google	2025-05-06 20:07:48	drivernya baik,perjalnan lancar tpt wktu	5

Gambar 4. 5 Coding sortir data

Penjelasan Gambar 4.6 dilihat pada bawah ini.

1. Baris ini mengambil empat kolom utama dari DataFrame df, yaitu:
2. nama: Nama pengguna yang menulis ulasan.
3. tgl: Tanggal ulasan dibuat.
4. komentar: Isi dari ulasan (teks).
5. data_sortir = data_mentah.sort_values(by='tgl', ascending=False)
6. Fungsi sort_values() digunakan untuk mengurutkan data berdasarkan kolom tgl (tanggal ulasan).
7. ascending=False artinya data akan diurutkan dari tanggal terbaru ke tanggal terlama.
8. data_sortir.head() Memastikan proses sortir berhasil.

```
data = data.drop_duplicates(subset='komentar', keep='first')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9944 entries, 2169 to 9496
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   nama      9944 non-null   object
 1   tgl       9944 non-null   datetime64[ns]
 2   komentar  9944 non-null   object
 3   score     9944 non-null   int64
dtypes: datetime64[ns](1), int64(1), object(2)
memory usage: 388.4+ KB
```

Gambar 4. 6 Coding drop duplicates data

Penjelasan Gambar 4.7 dilihat pada bawah ini.

1. Fungsi drop_duplicates() digunakan untuk menghapus baris yang memiliki isi kolom komentar yang sama (duplikat).
2. subset='komentar' berarti pemeriksaan duplikasi hanya dilakukan terhadap kolom komentar saja, bukan seluruh baris.
3. keep='first' berarti jika ada komentar yang sama muncul lebih dari sekali, maka hanya baris pertama yang akan dipertahankan, dan sisanya dihapus.
4. Fungsi info() dari pandas digunakan untuk melihat ringkasan struktur DataFrame data setelah penghapusan duplikat.

No	Nama	Komentar	score
1	Pengguna Google	saya langganan tetap tapu tolong donk untuk semua draivernya lengkap atribut terutama malam hari	5
2	Pengguna Google	Abang sangat ramah dah baik .. terimakasih abang	5
3	Pengguna Google	cepat, tanggal dan pelayanan warrr biasa Terima kasih maxim 👍🙏	5
4	Pengguna Google	pengemudi sering membatalkan jadi lama nyari pengemudi yg mau	4
5	Pengguna Google	drivernya baik,perjalnan lancar tpt wktu	5
6	Pengguna Google	cepat dan amanah, pokonya mantap. driver ramah dan baik...	5
7	Pengguna Google	tolong non cash untuk makanan segera di nyalakan.. dan juga ada pilihan untuk tarik saldo..	1
8	Pengguna Google	apalah daftar doang ribet amat, udah berkali kali coba ngedaftar pake nomor tapi tetep ga	1

No	Nama	Komentar	score
		ada notip kode verifikasi pake beda nomor pun tetep aja ga ada muncul kode nya mau daftar harus jawab foto verifikasi mending dah 2/3, ini lebih dari 5 kali. ga jelas banget	
9	Pengguna Google	pelayanan bagus. driver ramah. cepat sampai dg selamat	5
10	Pengguna Google	buat saya maxim sangat membantu dan harga terjangkau.semoga maxsim makin sukses kedepan nya .	5
…	…	…	…
10000	Pengguna Google	Ya nih belum aktif ya aplikasi nya pantesan pada complain sama aplikasi nya pas login terus masukin no nya kagak dikirim" kode nya aturan sebelum download baca dulu ya ulusan nya... Buang" kouta aja ni... Sumpah nyesel banget udh download kalau tau gini 😑	1

Tabel 4. 1 Hasil Pengumpulan Data

1) Pelabelan Sentimen

Proses pelabelan sangat penting secara otomatis memberikan label positif atau negatif ke ribuan data ulasan berdasarkan jumlah kata bernada positif atau negatif dalam teks. Cara mendapatkan output positif atau negatif dapat dilihat berikut in.

1. Mengimpor pustaka pandas, yang digunakan untuk memanipulasi dan menganalisis data dalam bentuk tabel (DataFrame).
2. pd.read_csv(...): Membaca file berformat .tsv (tab-separated values), yaitu kamus kata-kata positif dan negatif.
3. header=None: Menyatakan bahwa file tidak memiliki baris header.
4. [0]: Mengambil kolom pertama dari file.

5. set(...): Mengubah hasil pembacaan ke dalam struktur data set, yang mempercepat pencarian kata saat pengecekan.
6. text.split(): Memecah kalimat menjadi daftar kata.
7. Sum (1 for word in ...): Menghitung berapa banyak kata yang muncul di dalam positive_lexicon dan negative_lexicon.
8. Jika jumlah kata positif lebih banyak atau sama dengan jumlah kata negatif → dikembalikan label 'positif', selain itu 'negatif'.
9. df['stem_text']: Kolom teks ulasan yang sudah melalui preprocessing dan stemming.
10. apply(determine_sentiment): Menerapkan fungsi determine_sentiment() ke setiap baris teks, lalu menyimpan hasilnya ke kolom baru 'sentimen'.
11. Menampilkan 11 baris pertama dari DataFrame, yang sekarang sudah memiliki kolom tambahan 'sentimen', berisi hasil klasifikasi manual berdasarkan kamus kata.

Pelabelan Sentimen

```
import pandas as pd

# Load lexicon positif dan negatif
positive_lexicon = set(pd.read_csv('positive.tsv', sep='\t', header=None)[0])
negative_lexicon = set(pd.read_csv('negative.tsv', sep='\t', header=None)[0])

# Fungsi penentuan sentimen
def determine_sentiment(text):
    positive_count = sum(1 for word in text.split() if word in positive_lexicon)
    negative_count = sum(1 for word in text.split() if word in negative_lexicon)

    if positive_count >= negative_count:
        return 'positif'
    else:
        return 'negatif'

# Terapkan ke kolom 'stem_text' dalam DataFrame
df['sentimen'] = df['stem_text'].apply(determine_sentiment)

# Tampilkan 11 data pertama
df.head(11)
```

Gambar 4. 7 Coding pelabelan sentimen

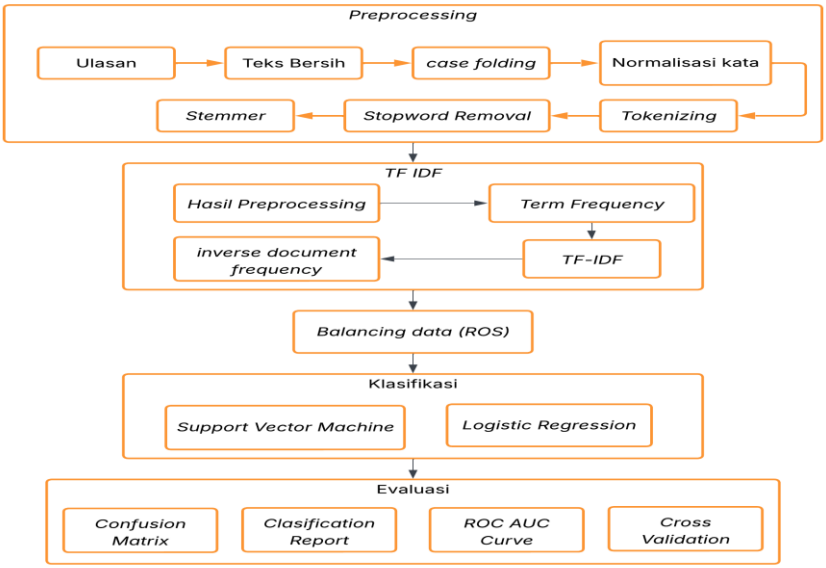
Tabel 4. 2 Hasil Pengumpulan sentiment

No	Nama	Komentar	label
1	Pengguna Google	saya langganan tetap tapu tolong donk untuk semua draivernya lengkap atribut terutama malam hari	negatif
2	Pengguna Google	Abang sangat ramah dah baik .. terimakasih abang	positif
3	Pengguna Google	cepat, tanggal dan pelayanan warrr biasa Terima kasih maxim 👍🙏	positif
4	Pengguna Google	pengemudi sering membatalkan jadi lama nyari pengemudi yg mau	positif
5	Pengguna Google	drivernya baik,perjalnan lancar tpt wktu	positif
6	Pengguna Google	cepat dan amanah, pokonya mantap. driver ramah dan baik...	positif
7	Pengguna Google	tolong non cash untuk makanan segera di nyalakan.. dan juga ada pilihan untuk tarik saldo..	positif
8	Pengguna Google	apalah daftar doang ribet amat, udah berkali kali coba ngedaftar pake nomor tapi tetep ga ada notip kode verifikasi pake beda nomor pun tetep aja ga ada muncul kode nya mau daftar harus jawab foto verifikasi mending dah 2/3, ini lebih dari 5 kali. ga jelas banget	positif
9	Pengguna Google	pelayanan bagus. driver ramah. cepat sampai dg selamat	negatif
10	Pengguna Google	buat saya maxim sangat membantu dan harga terjangkau.semoga maxsim makin sukses kedepan nya .	negatif
…	…	…	…

No	Nama	Komentar	label
10000	Pengguna Google	Ya nih belum aktif ya aplikasi nya pantesan pada complain sama aplikasi nya pas login terus masukin no nya kagak dikirim" kode nya aturan sebelum download baca dulu ya ulusan nya... Buang" kouta aja ni... Sumpah nyesel banget udh download kalau tau gini 😑	negatif

4.2 Hasil Pemodelan

Data ulasan yang telah dikumpulkan pada tahap sebelumnya kemudian melalui proses *preprocessing* untuk membersihkan dan menyiapkan data agar siap digunakan dalam pemodelan. Selanjutnya, fitur teks diubah menjadi representasi numerik menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*). Mengingat data yang digunakan mengalami ketidakseimbangan kelas, dilakukan penanganan dengan teknik *Random Over Sampler* untuk menyeimbangkan distribusi data. Setelah itu, dilakukan klasifikasi menggunakan dua metode, yaitu *Support Vector Machine* (SVM) dan *Logistic Regression* (LR).



Gambar 4. 8 Tahapan Pemodelan

Evaluasi performa model dilakukan dengan menggunakan beberapa metrik, antara lain *confusion matrix*, laporan klasifikasi, *ROC AUC Curve*, serta validasi silang (*cross-validation*) untuk memastikan keandalan hasil. Adapun tahapan dalam pemodelan dapat dilihat pada gambar di bawah ini.

4.2.1 Hasil preprocessing

Pada tahap preprocessing, data ulasan yang telah dikumpulkan dibersihkan dan dipersiapkan agar siap untuk proses analisis lebih lanjut. Proses ini meliputi beberapa langkah penting, yaitu text cleaning untuk menghilangkan karakter atau simbol yang tidak relevan, case folding untuk mengubah seluruh teks menjadi huruf kecil, word normalization untuk menyamakan bentuk kata, tokenization untuk memecah kalimat menjadi kata-kata, stopword removal untuk menghapus kata-kata umum yang tidak memiliki makna penting, serta stemming untuk mengubah kata ke bentuk dasarnya. Tahapan ini bertujuan untuk meningkatkan kualitas data sehingga representasi fitur yang dihasilkan pada tahap berikutnya menjadi lebih akurat dan efektif dalam proses klasifikasi.

1. *Text Cleaning*

   Pada tahap text cleaning, data ulasan yang telah dikumpulkan dibersihkan dari karakter-karakter yang tidak relevan seperti tanda baca berlebihan,

```python
import re
import unicodedata

def clean_text_advanced(text):
    # Menghapus angka
    text = re.sub(r'\d+', '', text)
    # Menghapus emoji
    text = ''.join(c for c in text if not unicodedata.decomposition(c))
    # Menghapus URL
    text = re.sub(r'https?://\S+', '', text)
    # Menghapus simbol dan tanda baca
    text = re.sub(r'[^\w\s]', ' ', text)
    # Menghapus huruf berulang lebih dari 2 kali dalam setiap kata
    text = re.sub(r'(.)\1{2,}', r'\1', text)
    # Menghapus kata berulang (misalnya 'mantap mantap' menjadi 'mantap')
    text = re.sub(r'\b(\w+)(?:\s+\1\b)+', r'\1', text)
    return text.strip()

# Terapkan ke dataset
```

Gambar 4. 9 Coding Text Cleaning

angka, simbol, dan spasi ganda. Proses ini bertujuan untuk menghilangkan elemen-elemen yang dapat mengganggu analisis teks sehingga hanya tersisa kata-kata yang bermakna. Dengan demikian, kualitas data menjadi lebih baik dan siap untuk diproses pada tahap selanjutnya.

Penjelasan gambar 4.10 dilihat pada berikut ini.

1. Import re: Modul untuk memproses teks dengan ekspresi reguler (regex).
2. Inmport unicodedata: Modul untuk mengakses karakter Unicode, digunakan di sini untuk menghapus emoji.
3. text = re.sub(r'\d+', '', text) menghapus semua karakter angka (0-9) dari teks.
4. text = ''.join(c for c in text if not unicodedata.decomposition(c)), Menghapus emoji dan karakter spesial dengan memeriksa apakah karakter tersebut memiliki *Unicode decomposition.*
5. text = re.sub(r'https?://\S+', '', text). Menghapus semua link atau alamat web dari komentar.
6. text = re.sub(r'[^\w\s]', ' ', text). Menghapus semua karakter selain huruf, angka, dan spasi (misalnya: titik, koma, tanda seru, hashtag, dll).
7. text = re.sub(r'(.)\1{2,}', r'\1', text). Menormalisasi ejaan yang dilebih-lebihkan, agar mudah dipahami model.
8. text = re.sub(r'\b(\w+)(?:\s+\1\b)+', r'\1', text). Menghindari pengaruh berlebihan dari pengulangan kata yang tidak menambah makna baru.
9. return text.strip().Menghapus spasi berlebih di awal dan akhir kalimat.
10. Menerapkan fungsi clean_text_advanced() ke seluruh kolom komentar dalam dataset.
11. Hasilnya disimpan dalam kolom baru bernama clean_text.

Tabel 4. 3 Hasil pembersihan teks

komentar	clean_text
sangat baik dan memuaskan	sangat baik dan memuaskan
Mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sma petanya tdk terlihat susah untuk tau keberadaanya maximnya	Mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sma petanya tdk terlihat susah untuk tau keberadaanya maximnya
tolong peta diberikan mode satelit seperti google map,agar lebih baik dalam mencari lokasi,terkadang costumer sulit menemukan titik yg pas	tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yg pas
keren....amanah...sukses selalu	keren amanah sukses selalu
mau login malah di suruh ngotak ngotakin gambar ga udah udah ga jelas	mau login malah di suruh ngotak ngotakin gambar ga udah ga jelas

1. *Case folding*

   Case folding adalah proses mengubah seluruh huruf dalam teks menjadi huruf kecil untuk menyamakan format dan menghindari duplikasi kata yang sama dengan bentuk berbeda.

```
[43]: def to_lower(text):
        text = text.lower()
        return text

      data['case_folding'] = data['clean_text'].apply(lambda x: to_lower(x))
      data[['clean_text','case_folding']].head(5)
```

Gambar 4. 10 Coding Case folding

Penjelasan gambar 4.11 dilihat pada berikut ini.

1. Menerima parameter text berupa string (kalimat hasil cleaning).
2. Menggunakan .lower() untuk mengubah seluruh karakter menjadi huruf kecil.
3. Mengembalikan hasil teks yang sudah dalam huruf kecil.
4. Menggunakan apply() untuk menerapkan fungsi to_lower() ke setiap baris pada kolom clean_text.
5. Hasilnya disimpan di kolom baru bernama case_folding.
6. clean_text: hasil teks yang telah dibersihkan dari angka, simbol, dsb.
7. case_folding: hasil dari clean_text yang sudah diubah menjadi huruf kecil.

Tabel 4. 4 hasil case folding

clean_text	case_folding
sangat baik dan memuaskan	sangat baik dan memuaskan
Mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sma petanya tdk terlihat susah untuk tau keberadaanya maximnya	mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sma petanya tdk terlihat susah untuk tau keberadaanya maximnya
tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yg pas	tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yg pas
keren amanah sukses selalu	keren amanah sukses selalu
mau login malah di suruh ngotak ngotakin gambar ga udah ga jelas	mau login malah di suruh ngotak ngotakin gambar ga udah ga jelas

1. *Word Normalization*

   Word normalization adalah proses menyamakan variasi kata menjadi bentuk dasar atau standar agar analisis teks menjadi lebih konsisten dan akurat.

```
45]: def replace_taboo_word(text, kamus_slag):
         if isinstance(text,str):
             words = text.split()
             replaced_words = []
             kalimat_baku = []
             kata_diganti = []
             kata_tidak_baku_hash = []
             for word in words:
                 if word in kamus_slag:
                     baku_word = kamus_slag[word]
                     if isinstance(baku_word,str) and all(char.isalpha() for char in baku_word):
                         replaced_words.append(baku_word)
                         kalimat_baku.append(baku_word)
                         kata_diganti.append(word)
                         kata_tidak_baku_hash.append(word)
                 else:
                     replaced_words.append(word)
             replaced_text = ' '.join(replaced_words)
         else:
             replaced_text = ''
             kalimat_baku = []
             kata_diganti = []
             kata_tidak_baku_hash = []
         return replaced_text, kalimat_baku, kata_diganti, kata_tidak_baku_hash

25]: #Data disimpan di excel
     kamus_data = pd.read_excel('kamuskatabaku.xlsx')
     kamus_tidak_baku = dict(zip(kamus_data['tidak_baku'],kamus_data['kata_baku']))

49]: data['normalisasi'], data['Kata_Baku'], data['Kata_Tidak_Baku'], data['Kata_Tidak_Baku_Hash']
     = zip(*data['case_folding'].apply(lambda x:replace_taboo_word(x,kamus_tidak_baku)))
     df = pd.DataFrame(data[['nama','tgl','score','komentar','clean_text','case_folding','normalisasi']])

     df[['case_folding','normalisasi']].head(50)
```

Gambar 4. 11 Coding Word Normalization

Penjelasan gambar 4.12 dilihat pada berikut ini.

1. text: kalimat yang akan dinormalisasi.
2. kamus_slag: dictionary yang berisi pasangan kata tidak baku sebagai key dan kata bakunya sebagai value.
3. if isinstance(text, str):. Mengecek apakah text adalah string. Ini menghindari error jika input bukan teks (misalnya None atau angka).
4. words: memecah teks menjadi list kata per kata.
5. replaced_words: akan menyimpan versi kata yang telah diganti atau tetap jika tidak perlu diganti.

6. kalimat_baku: kumpulan kata baku hasil penggantian. kata_diganti: daftar kata tidak baku yang berhasil diganti.
7. kata_tidak_baku_hash: duplikasi kata_diganti, kemungkinan digunakan untuk keperluan lain seperti hashing atau pelabelan.
8. replaced_text = ' '.join(replaced_words). Semua kata yang telah diperiksa (baik diganti atau tidak) digabung kembali menjadi kalimat utuh
9. else: replaced_text = '' kalimat_baku = [] kata_diganti = [] kata_tidak_baku_hash = []. Jika input text bukan string, maka semua output dikembalikan dalam bentuk kosong untuk menghindari error.
10. return replaced_text, kalimat_baku, kata_diganti, kata_tidak_baku_hash. Fungsi mengembalikan empat nilai sekaligus.

Tabel 4. 5 Hasil Word Normalization

case_folding	**normalisasi**
sangat baik dan memuaskan	sangat baik dan memuaskan
mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sma petanya tdk terlihat susah untuk tau keberadaanya maximnya	mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sama petanya tidak terlihat susah untuk tau keberadaanya maximnya
tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yg pas	tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yang pas
keren amanah sukses selalu	keren amanah sukses selalu

case_folding	normalisasi
mau login malah di suruh ngotak ngotakin gambar ga udah ga jelas	mau login malah di suruh ngotak ngotakin gambar tidak sudah tidak jelas

1. *Tokenizing*

Tokenizing adalah proses memecah teks menjadi unit-unit kecil seperti kata atau token agar dapat dianalisis secara lebih terperinci.

[51]:
```
def tokenize(text):
    tokens = text.split()
    return tokens

df['tokenize'] = df['normalisasi'].apply(lambda x:tokenize(x))

df[['normalisasi','tokenize']].head(5)
```

Gambar 4. 12 Coding Tokenizing

Penjelasan gambar 4.13 dilihat pada berikut ini.

1. text.split() adalah metode bawaan Python untuk memecah string berdasarkan spasi (default).
2. Output dari fungsi ini adalah list (daftar) kata-kata dalam kalimat tersebut.
3. Kolom normalisasi adalah teks yang sudah dibersihkan dan dinormalisasi (dari tahapan sebelumnya).
4. Fungsi apply(lambda x: ...) digunakan untuk menerapkan fungsi tokenize() ke setiap baris teks dalam kolom tersebut.
5. Hasilnya disimpan ke kolom baru bernama tokenize, yang berisi list token/kata dari setiap kalimat.
6. Menampilkan 5 baris pertama dari kolom normalisasi (teks bersih) dan tokenize (hasil tokenisasi).untuk memverifikasi apakah proses tokenisasi sudah berjalan dengan benar.

Tabel 4. 6 Hasil Tokenizing

normalisasi	tokenize
sangat baik dan memuaskan	['sangat', 'baik', 'dan', 'memuaskan']
mohon aplikasinya diperbaikki lagi lebih baik kayak itu alamat tujuannya sama petanya tidak terlihat susah untuk tau keberadaanya maximnya	['mohon', 'aplikasinya', 'diperbaikki', 'lagi', 'lebih', 'baik', 'kayak', 'itu', 'alamat', 'tujuannya', 'sama', 'petanya', 'tidak', 'terlihat', 'susah', 'untuk', 'tau', 'keberadaanya', 'maximnya']
tolong peta diberikan mode satelit seperti google map agar lebih baik dalam mencari lokasi terkadang costumer sulit menemukan titik yang pas	['tolong', 'peta', 'diberikan', 'mode', 'satelit', 'seperti', 'google', 'map', 'agar', 'lebih', 'baik', 'dalam', 'mencari', 'lokasi', 'terkadang', 'costumer', 'sulit', 'menemukan', 'titik', 'yang', 'pas']
keren amanah sukses selalu	['keren', 'amanah', 'sukses', 'selalu']
mau login malah di suruh ngotak ngotakin gambar tidak sudah tidak jelas	['mau', 'login', 'malah', 'di', 'suruh', 'ngotak', 'ngotakin', 'gambar', 'tidak', 'sudah', 'tidak', 'jelas']

1. *Stopword Removal*

   Stopword removal adalah proses menghapus kata-kata umum yang tidak memiliki makna penting dalam analisis, seperti kata sambung dan kata depan, agar fokus pada kata-kata yang lebih bermakna

```
61]: from nltk.corpus import stopwords
     nltk.download('stopwords')

     # Load stopwords Bahasa Indonesia
     stopword = set(stopwords.words('indonesian'))
     stopword.update(['ya','nya'])
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
63]: def remove_stopwords(text):
         """Menghapus stopwords dari teks."""
         return [word for word in text if word not in stopword]

     df['no_stopword'] = df['tokenize'].apply(lambda x: remove_stopwords(x))
     df[['tokenize','no_stopword']].head()
```

Gambar 4. 13 Coding Stopword Removal

Penjelasan gambar 4.14 dilihat pada berikut ini.

1. nltk.corpus adalah modul dari library NLTK (Natural Language Toolkit) yang menyediakan berbagai dataset teks, termasuk stopwords.
2. stopwords adalah daftar kata-kata umum seperti "dan", "atau", "yang" yang sering muncul tetapi tidak memberikan banyak informasi penting untuk analisis.
3. nltk.download('stopwords') Perintah ini digunakan untuk mengunduh dataset stopwords dari server NLTK ke lokal komputer Anda.
4. Perlu dijalankan satu kali (selama environment masih menyimpan hasil unduhan).
5. stopword = set(stopwords.words('indonesian')) Baris ini mengambil semua kata-kata umum (stopwords) dalam Bahasa Indonesia dari koleksi NLTK.
6. set() digunakan agar pencarian kata jadi lebih cepat (menggunakan hash dibanding list biasa).
7. stopword.update(['ya', 'nya']) Anda secara manual menambahkan kata "ya" dan "nya" ke daftar stopwords.

8. Fungsi ini menerima input text, yaitu list token/kata (bukan string kalimat utuh).
9. stopword adalah kumpulan kata-kata umum dalam Bahasa Indonesia (seperti "yang", "dan", "atau", "sudah", dll) yang tidak memiliki nilai analisis penting.
10. Fungsi mengembalikan list baru yang hanya berisi kata-kata penting (non-stopwords).
11. Kolom tokenize berisi daftar kata hasil pemecahan (tokenisasi) dari kalimat yang sudah dibersihkan dan dinormalisasi.
12. .apply(lambda x: remove_stopwords(x)) menjalankan fungsi remove_stopwords ke setiap baris dalam kolom tokenize.
13. Hasilnya disimpan di kolom baru bernama no_stopword.
14. df[['tokenize','no_stopword']].head(). tokenize: daftar kata sebelum stopword dihapus.
15. no_stopword: daftar kata setelah stopword dihapus.

Tabel 4. 7 Hasil *Stopword Removal*

tokenize	**no_stopword**
['sangat', 'baik', 'dan', 'memuaskan']	['memuaskan']
['mohon', 'aplikasinya', 'diperbaikki', 'lagi', 'lebih', 'baik', 'kayak', 'itu', 'alamat', 'tujuannya', 'sama', 'petanya', 'tidak', 'terlihat', 'susah', 'untuk', 'tau', 'keberadaanya', 'maximnya']	['mohon', 'aplikasinya', 'diperbaikki', 'kayak', 'alamat', 'tujuannya', 'petanya', 'susah', 'tau', 'keberadaanya', 'maximnya']
['tolong', 'peta', 'diberikan', 'mode', 'satelit', 'seperti', 'google', 'map', 'agar', 'lebih', 'baik', 'dalam', 'mencari', 'lokasi', 'terkadang',	['tolong', 'peta', 'mode', 'satelit', 'google', 'map', 'mencari', 'lokasi', 'terkadang', 'costumer', 'sulit', 'menemukan', 'titik', 'pas']

tokenize	no_stopword
'costumer', 'sulit', 'menemukan', 'titik', 'yang', 'pas']	
['keren', 'amanah', 'sukses', 'selalu']	['keren', 'amanah', 'sukses']
['mau', 'login', 'malah', 'di', 'suruh', 'ngotak', 'ngotakin', 'gambar', 'tidak', 'sudah', 'tidak', 'jelas']	['login', 'suruh', 'ngotak', 'ngotakin', 'gambar']

1. *Stemming*

   Stemming adalah proses mengubah kata-kata menjadi bentuk dasarnya untuk mengurangi variasi kata dan memudahkan analisis teks.

```
# Stemmer untuk Bahasa Indonesia
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stem_text(words):
    """Melakukan stemming pada list kata."""
    return ' '.join([stemmer.stem(word) for word in words])

df['stem_text'] = df['no_stopword'].apply(lambda x:stem_text(x))
df[['no_stopword','stem_text']].head()
```

Gambar 4. 15 Coding Stemmin

Penjelasan gambar 4.15 dilihat pada berikut ini.

1. StemmerFactory() adalah kelas dari library Sastrawi untuk membuat objek stemmer.
2. factory.create_stemmer() menghasilkan objek stemmer yang dapat digunakan untuk melakukan stemming pada kata atau kalimat dalam Bahasa Indonesia.
3. Fungsi ini menerima input berupa list kata (words)—hasil dari tahap sebelumnya (stopword removal).
4. Fungsi stemmer.stem(word) akan mengubah setiap kata menjadi bentuk dasarnya (contoh: berjalan → jalan).

5. Kemudian semua kata hasil stemming digabung kembali menjadi satu kalimat dengan ' '.join(...).
6. Fungsi stem_text() diterapkan ke setiap baris dalam kolom no_stopword yang berisi list kata tanpa stopword.
7. Hasilnya disimpan dalam kolom baru bernama stem_text.
8. df[['no_stopword','stem_text']].head(). Kolom no_stopword: list kata setelah stopword dihapus. Kolom stem_text: hasil konversi ke bentuk kata dasar.

Tabel 4. 8 Hasil Stemming

no_stopword	stem_text
['memuaskan']	muas
['mohon', 'aplikasinya', 'diperbaikki', 'kayak', 'alamat', 'tujuannya', 'petanya', 'susah', 'tau', 'keberadaanya', 'maximnya']	mohon aplikasi diperbaikki kayak alamat tuju peta susah tau keberadaanya maximnya
['tolong', 'peta', 'mode', 'satelit', 'google', 'map', 'mencari', 'lokasi', 'terkadang', 'costumer', 'sulit', 'menemukan', 'titik', 'pas']	tolong peta mode satelit google map cari lokasi terkadang costumer sulit temu titik pas
['keren', 'amanah', 'sukses']	keren amanah sukses
['login', 'suruh', 'ngotak', 'ngotakin', 'gambar']	login suruh ngotak ngotakin gambar

```
df = df.drop_duplicates(subset='stem_text', keep='first')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9664 entries, 0 to 9663
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   nama           9664 non-null   object
 1   tgl            9664 non-null   datetime64[ns]
 2   score          9664 non-null   int64
 3   komentar       9664 non-null   object
 4   clean_text     9663 non-null   object
 5   case_folding   9663 non-null   object
 6   normalisasi    9663 non-null   object
 7   tokenize       9664 non-null   object
 8   no_stopword    9664 non-null   object
 9   stem_text      9663 non-null   object
 10  sentimen       9664 non-null   object
dtypes: datetime64[ns](1), int64(1), object(9)
memory usage: 830.6+ KB
```

Gambar 4. 16 Coding data duplikat

Penjelasan gambar 4.16 dilihat pada berikut ini.

1. Menghapus baris-baris duplikat yang memiliki nilai identik pada kolom stem_text.
2. subset='stem_text': Menentukan bahwa pencarian duplikat hanya berdasarkan isi kolom stem_text (hasil stemming dari teks ulasan).
3. keep='first': Menyimpan hanya kemunculan pertama dari setiap nilai duplikat dan membuang sisanya.
4. df.info().Menampilkan informasi ringkas tentang DataFrame saat ini

```
df = df[df['stem_text'].apply(lambda x: isinstance(x, str))]

# Cek lagis
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9663 entries, 0 to 9663
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   nama           9663 non-null   object
 1   tgl            9663 non-null   datetime64[ns]
 2   score          9663 non-null   int64
 3   komentar       9663 non-null   object
 4   clean_text     9663 non-null   object
 5   case_folding   9663 non-null   object
 6   normalisasi    9663 non-null   object
 7   tokenize       9663 non-null   object
 8   no_stopword    9663 non-null   object
 9   stem_text      9663 non-null   object
 10  sentimen       9663 non-null   object
dtypes: datetime64[ns](1), int64(1), object(9)
memory usage: 905.9+ KB
```

Gambar 4. 17 Coding pembersihan data

Penjelasan gambar 4.17 dilihat pada berikut ini.

1. df['stem_text'].apply(...): Mengevaluasi setiap nilai dalam kolom stem_text. lambda x: isinstance(x, str):
2. Fungsi anonim yang mengembalikan True jika nilai x adalah string (str), dan False jika bukan. df[...]:
3. Hanya menyimpan baris-baris yang bernilai True (yakni, stem_text-nya bertipe string).
4. Menghindari error di tahapan pemrosesan berikutnya (seperti TF-IDF atau tokenisasi lanjutan) akibat nilai yang bukan string, seperti NaN, angka, None, atau list yang gagal diproses.
5. df.info() Menampilkan kembali struktur data setelah difilter:

```
df[['nama','score','komentar','clean_text','case_folding','normalisasi','tokenize','no_stopword','stem_text']]
.to_excel('dataframe/hasil_preprocessing.xlsx',index=False)
print('data berhasil disimpan di folder dataframe')
```

Gambar 4. 18 Coding data disimpan di folder dataframe

Penjelasan gambar 4.18 dilihat pada berikut ini.

1. .to_excel(...): Menyimpan data ke file Excel.
2. 'dataframe/hasil_preprocessing.xlsx': Lokasi dan nama file tujuan penyimpanan.
3. Disimpan dalam folder bernama dataframe (pastikan folder sudah ada atau dibuat sebelumnya).
4. index=False: Tidak menyertakan indeks DataFrame ke dalam file Excel, agar tabel lebih bersih.

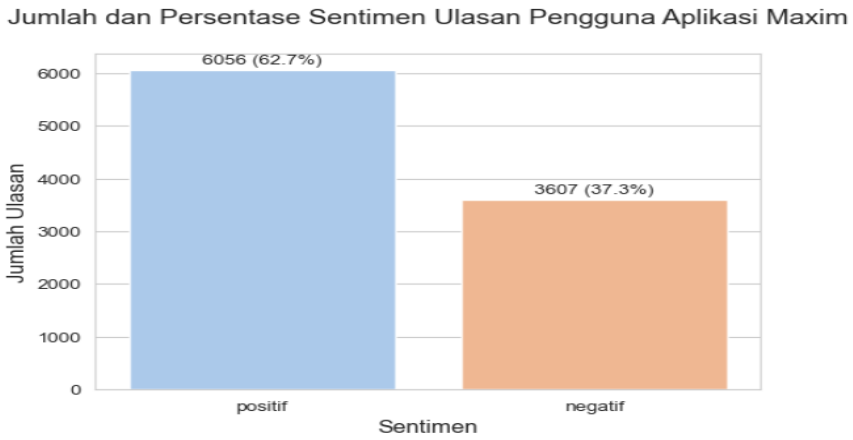
```
5]: import seaborn as sn
    import matplotlib.pyplot as plt

    # Hitung jumlah setiap label sentimen
    jml_sentimen = df['sentimen'].value_counts()
    total_ulasan = jml_sentimen.sum()

    # Hitung persentase setiap label sentimen
    persen_sentimen = (jml_sentimen / total_ulasan) * 100

    # Visualisasi diagram batang
    sn.set_style('whitegrid')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax = sn.barplot(
        x=jml_sentimen.index,
        y=jml_sentimen.values,
        palette='pastel',
        hue=jml_sentimen.index,
        dodge=False,
        legend=False
    )
    plt.title('Jumlah dan Persentase Sentimen Ulasan Pengguna Aplikasi Maxim', fontsize=14, pad=20)
    plt.xlabel('Sentimen', fontsize=12)
    plt.ylabel('Jumlah Ulasan', fontsize=12)

    # Tambahkan label jumlah dan persentase di atas batang
    for i, (count, percent) in enumerate(zip(jml_sentimen.values, persen_sentimen.values)):
        ax.text(i, count + 50, f'{count} ({percent:.1f}%)', ha='center', va='bottom')

    # Tampilkan plot
    plt.show()
```

Gambar 4. 19 Coding menghitung jumlah sentimen

Penjelasan gambar 4.19 dilihat pada berikut ini.

1. import seaborn as sn Library untuk membuat visualisasi statistik yang menarik dan informatif.
2. import matplotlib.pyplot as plt Digunakan untuk membuat dan menampilkan grafik.
3. df['sentimen'].value_counts(): Menghitung jumlah masing-masing label sentimen (misalnya: positif, negatif).
4. total_ulasan: Jumlah seluruh ulasan.
5. persen_sentimen: Persentase setiap label sentimen dari total ulasan.
6. sn.set_style('whitegrid'): Mengatur gaya visualisasi dengan latar putih dan grid halus.

7. plt.subplots(figsize=(6, 4)): Membuat canvas plot dengan ukuran 6x4 inci.
8. x: Label sentimen.
9. y : Jumlah masing-masing label.
10. palette='pastel': Menggunakan warna pastel agar visual tampak ringan dan tidak mencolok.
11. hue dan dodge=False: Memastikan tidak ada pemisahan ganda berdasarkan warna.
12. legend=False: Menonaktifkan legenda karena informasi sudah cukup jelas dari sumbu X.
13. plt.title(...), plt.xlabel(...), plt.ylabel(...) Menambahkan judul utama grafik dan label untuk sumbu X (kategori sentimen) dan Y (jumlah ulasan).
14. Menempatkan teks di atas masing-masing batang berisi: Jumlah ulasan (misal: 2300) Persentase total (misal: 46.0%) count + 50: Mengatur posisi label agar tidak menempel pada batang.
15. plt.show() Menampilkan hasil visualisasi secara interaktif.



Gambar 4. 20 Hasil Jumlah Sentimen

4.2.2 Hasil TF-IDF

Setelah tahap preprocessing selesai, data teks diubah menjadi representasi numerik menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*). Metode ini memberikan bobot pada setiap kata berdasarkan frekuensi kemunculannya dalam dokumen dan seberapa umum kata tersebut muncul di seluruh kumpulan data. Dengan demikian, TF-IDF mampu menyoroti kata-kata yang penting dan membedakan antar dokumen secara efektif. Dalam penelitian ini, hasil transformasi TF-IDF ditampilkan untuk 10 dokumen ulasan sebagai contoh representasi fitur numerik dari data teks. Setiap dokumen diubah menjadi vektor bobot yang merefleksikan pentingnya kata-kata dalam dokumen tersebut berdasarkan frekuensi kemunculan dan distribusinya di seluruh kumpulan data. Penyajian hasil ini bertujuan untuk memberikan gambaran bagaimana metode TF-IDF mengolah data teks menjadi bentuk yang dapat digunakan oleh algoritma klasifikasi.

1. *Term Frequency* (TF)

   Term Frequency (TF) merupakan ukuran frekuensi kemunculan suatu kata dalam sebuah dokumen. Semakin sering sebuah kata muncul dalam dokumen, maka nilai TF-nya akan semakin tinggi. TF digunakan untuk menilai seberapa penting kata tersebut dalam konteks dokumen tertentu. Adapun TF untuk 10 dokumen serta codinganya dapat dilihat di bawah ini.

```
# Menghitung term-document matrix
tf = CountVectorizer()
term_doc_matr = tf.fit_transform(df['stem_text'][:10])

# Membuat DataFrame dengan Terms sebagai baris dan dokumen sebagai kolom
df_term_doc = pd.DataFrame(
    term_doc_matr.toarray().T,
    index=tf.get_feature_names_out(),
    columns=[f'D{i+1}' for i in range(term_doc_matr.shape[0])]
).reset_index()

df_term_doc.insert(0, 'No', range(1, len(df_term_doc) + 1))
df_term_doc.rename(columns={'index': 'Terms'}, inplace=True)

# Menghitung panjang dokumen (jumlah kata per dokumen)
doc_lengths = term_doc_matr.toarray().sum(axis=1)
df_term_doc.loc[len(df_term_doc)] = ['-', 'Document Length'] + list(doc_lengths)

# Simpan ke file Excel
output_file = 'dataframe/tf_document.xlsx'
df_term_doc.to_excel(output_file, index=False, sheet_name='tf')

print(f"File berhasil disimpan: {output_file}")
display(df_term_doc.head(20).style.background_gradient(cmap='YlOrRd'))
```

```
File berhasil disimpan: dataframe/tf_document.xlsx
```

Gambar 4. 21 Coding Hasil Jumlah Sentimen

Penjelasan gambar 4.21 dilihat pada berikut ini.

1. tf = CountVectorizer() Digunakan untuk menghitung frekuensi kemunculan setiap kata (term) dalam dokumen.
2. fit_transform(...): Menerapkan perhitungan frekuensi kata terhadap 10 dokumen pertama dari kolom stem_text.
3. term_doc_matr.toarray().T: Mengubah matriks sparse menjadi array biasa dan **mentransposenya** agar baris = kata, kolom = dokumen.
4. tf.get_feature_names_out(): Mendapatkan daftar kata (fitur).
5. columns=[f'D{i+1}'...]: Kolom diberi nama D1, D2, ..., D10.
6. Menambahkan kolom No sebagai penomoran baris.
7. Ubah nama kolom index menjadi Terms (kata yang dihitung).
8. doc_lengths: Menghitung total kata dalam tiap dokumen.

9. df_term_doc.loc[...]: Menambahkan baris paling bawah untuk menunjukkan panjang (jumlah kata) dari setiap dokumen.
10. Menyimpan hasil ke dalam file Excel bernama tf_document.xlsx di dalam folder dataframe.
11. Menampilkan 20 baris pertama dari TDM dengan gradasi warna berdasarkan nilai menggunakan palet YlOrRd (kuning ke merah).
12. Membantu memvisualisasikan term mana yang paling sering muncul.

Tabel 4. 9 Term Frequency

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	alamat	0	1	0	0	0	0	0	0	0	0
2	amanah	0	0	0	1	0	0	0	0	0	0
3	aplikasi	0	1	0	0	0	0	1	1	0	0
4	bagus	0	0	0	0	0	1	0	0	1	0
5	bersih	0	0	0	0	0	0	0	0	0	1
6	cari	0	0	1	0	0	0	0	0	0	0
7	costumer	0	0	1	0	0	0	0	0	0	0
8	diperbaikki	0	1	0	0	0	0	0	0	0	0
9	gambar	0	0	0	0	1	0	0	0	0	0
10	google	0	0	1	0	0	0	0	0	0	0
11	harga	0	0	0	0	0	0	1	0	0	0
12	history	0	0	0	0	0	0	2	0	0	0
13	hujan	0	0	0	0	0	0	0	0	1	0
14	kayak	0	1	0	0	0	0	0	0	0	0
15	keberadaanya	0	1	0	0	0	0	0	0	0	0
16	kena	0	0	0	0	0	0	1	0	0	0
17	kendara	0	0	0	0	0	0	1	0	0	0
18	keren	0	0	0	1	0	0	0	0	0	0

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
19	layan	0	0	0	0	0	1	0	0	1	0
20	lihat	0	0	0	0	0	0	1	0	0	0
21	login	0	0	0	0	1	0	0	0	0	0
22	lokasi	0	0	1	0	0	0	0	0	0	0
23	lol	0	0	0	0	0	0	0	1	0	0
24	map	0	0	1	0	0	0	0	0	0	0
25	maximnya	0	1	0	0	0	0	0	0	0	0
26	mesan	0	0	0	0	0	0	1	0	0	0
27	mode	0	0	1	0	0	0	0	0	0	0
28	mohon	0	1	0	0	0	0	0	0	0	0
29	muas	1	0	0	0	0	0	0	0	0	0
30	ngotak	0	0	0	0	1	0	0	0	0	0
31	ngotakin	0	0	0	0	1	0	0	0	0	0
32	nopol	0	0	0	0	0	0	1	0	0	0
33	orang	0	0	0	0	0	0	0	0	0	1
34	pas	0	0	1	0	0	0	1	0	0	0
35	peta	0	1	1	0	0	0	0	0	0	0
36	satelit	0	0	1	0	0	0	0	0	0	0
37	sehat	0	0	0	0	0	0	0	0	0	1
38	sesuai	0	0	0	0	0	0	1	0	0	0
39	sukses	0	0	0	1	0	0	0	0	0	0
40	sulit	0	0	1	0	0	0	0	0	0	0
41	suruh	0	0	0	0	1	0	0	0	0	0
42	susah	0	1	0	0	0	0	0	0	0	0
43	tau	0	1	0	0	0	0	0	0	0	0
44	tembak	0	0	0	0	0	0	1	0	0	0
45	temu	0	0	1	0	0	0	0	0	0	0
46	tera	0	0	0	0	0	0	1	0	0	0
47	terima	0	0	0	0	0	0	0	0	1	0

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
48	terkadang	0	0	1	0	0	0	1	0	0	0
49	titik	0	0	1	0	0	0	0	0	0	0
50	tolong	0	0	1	0	0	0	0	0	0	0
51	tuju	0	1	0	0	0	0	0	0	0	0
52	update	0	0	0	0	0	0	0	1	0	0
53	wlpun	0	0	0	0	0	0	0	0	1	0
-	Document Length	1	11	14	3	5	2	14	3	5	3

Untuk menghindari bias terhadap kata-kata yang sangat sering muncul, dilakukan proses TF Normalisasi. TF Normalisasi menyesuaikan nilai Term Frequency agar berada dalam rentang tertentu, biasanya antara 0 dan 1, sehingga bobot kata menjadi lebih seimbang dan representatif dalam analisis yang dapat dilihat pada berikut.

```python
tf = CountVectorizer()
term_doc_matr = tf.fit_transform(df['stem_text'][:10])
```

Gambar 4. 22 Coding Menghitung Term Frequency (TF)

Penjelasan gambar 4.22 dilihat pada berikut ini.

1. CountVectorizer(): Alat untuk mengubah teks menjadi matriks frekuensi kata.
2. df['stem_text'][:10]: Hanya menggunakan 10 dokumen pertama.
3. term_doc_matr: Matriks TF mentah, bentuk awal sebelum normalisasi.

```python
doc_lengths = term_doc_matr.toarray().sum(axis=1)
```

Gambar 4. 23 Coding Hitung Panjang Dokumen

Penjelasan gambar 4.23 dilihat pada berikut ini.

1. Menghitung total kata (frekuensi total) dalam setiap dokumen.
2. Ini akan digunakan untuk melakukan normalisasi TF agar proporsional.

```python
tf_normalized = np.round(term_doc_matr.toarray().T / doc_lengths, decimals=3)
```

Gambar 4. 24 Coding Hitung TF Normalisasi

Penjelasan gambar 4.24 dilihat pada berikut ini.

1. term_doc_matr.toarray().T: Transpose matriks agar baris = term, kolom = dokumen.
2. Pembagian dengan doc_lengths: Membuat nilai TF menjadi relatif terhadap panjang dokumen.
3. np.round(..., decimals=3): Membulatkan hasil ke 3 angka desimal untuk presisi dan keterbacaan.

```python
df_tf_normalized = pd.DataFrame(
    tf_normalized,
    index=tf.get_feature_names_out(),
    columns=[f'D{i+1}' for i in range(term_doc_matr.shape[0])]
).reset_index()
```

Gambar 4. 25 Coding Buat DataFrame

Penjelasan gambar 4.25 dilihat pada berikut ini.

1. get_feature_names_out(): Mendapatkan daftar kata (term) dari data.
2. Baris = kata, kolom = dokumen (D1, D2, ..., D10).

```
# Menambahkan kolom nomor
df_tf_normalized.insert(0, 'No', range(1, len(df_tf_normalized) + 1))
df_tf_normalized.rename(columns={'index': 'Terms'}, inplace=True)

# Menambahkan baris panjang dokumen
df_tf_normalized.loc[len(df_tf_normalized)] = ['-', 'Document Length'] + list(doc_lengths)
```

Gambar 4. 26 Coding Tambahkan Kolom & Baris Tambahan

Penjelasan gambar 4.26 dilihat pada berikut ini.

1. Tambahkan kolom No untuk penomoran kata.
2. Ubah nama kolom index menjadi Terms (daftar kata).
3. Tambahkan baris paling bawah untuk menunjukkan panjang dokumen sebagai referensi analisis.

```
# Simpan ke file Excel
output_file = 'dataframe/tf_normalized.xlsx'
df_tf_normalized.to_excel(output_file, index=False, sheet_name='tf_normalized')
print(f"File berhasil disimpan: {output_file}")
```

Gambar 4. 27 Coding Simpan ke Excel

Penjelasan gambar 4.27 dilihat pada berikut ini.

1. Menyimpan hasil ke file Excel bernama tf_normalized.xlsx dalam folder dataframe.

```
# Menampilkan hasil dengan format warna
display(df_tf_normalized.head(12).style.background_gradient(cmap='YlOrRd'))
```

Gambar 4. 28 Coding Tampilkan Hasil dengan Warna

1. Menampilkan 12 baris pertama dengan gradasi warna:

2. Warna terang = nilai TF normalisasi tinggi (kata sering muncul).
3. Warna gelap = nilai TF rendah (kata jarang muncul).

Tabel 4. 10 TF Normalisasi

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	alamat	0	0,091	0	0	0	0	0	0	0	0
2	amanah	0	0	0	0,333	0	0	0	0	0	0
3	aplikasi	0	0,091	0	0	0	0	0,071	0,333	0	0
4	bagus	0	0	0	0	0	0,5	0	0	0,2	0
5	bersih	0	0	0	0	0	0	0	0	0	0,333
6	cari	0	0	0,071	0	0	0	0	0	0	0
7	costumer	0	0	0,071	0	0	0	0	0	0	0
8	diperbaikki	0	0,091	0	0	0	0	0	0	0	0
9	gambar	0	0	0	0	0,2	0	0	0	0	0
10	google	0	0	0,071	0	0	0	0	0	0	0
11	harga	0	0	0	0	0	0	0,071	0	0	0
12	history	0	0	0	0	0	0	0,143	0	0	0
13	hujan	0	0	0	0	0	0	0	0	0,2	0
14	kayak	0	0,091	0	0	0	0	0	0	0	0
15	keberadaanya	0	0,091	0	0	0	0	0	0	0	0
16	kena	0	0	0	0	0	0	0,071	0	0	0
17	kendara	0	0	0	0	0	0	0,071	0	0	0
18	keren	0	0	0	0,333	0	0	0	0	0	0
19	layan	0	0	0	0	0	0,5	0	0	0,2	0
20	lihat	0	0	0	0	0	0	0,071	0	0	0
21	login	0	0	0	0	0,2	0	0	0	0	0
22	lokasi	0	0	0,071	0	0	0	0	0	0	0
23	lol	0	0	0	0	0	0	0	0,333	0	0
24	map	0	0	0,071	0	0	0	0	0	0	0

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
25	maximnya	0	0,091	0	0	0	0	0	0	0	0
26	mesan	0	0	0	0	0	0	0,071	0	0	0
27	mode	0	0	0,071	0	0	0	0	0	0	0
28	mohon	0	0,091	0	0	0	0	0	0	0	0
29	muas	1	0	0	0	0	0	0	0	0	0
30	ngotak	0	0	0	0	0,2	0	0	0	0	0
31	ngotakin	0	0	0	0	0,2	0	0	0	0	0
32	nopol	0	0	0	0	0	0	0,071	0	0	0
33	orang	0	0	0	0	0	0	0	0	0	0,333
34	pas	0	0	0,071	0	0	0	0,071	0	0	0
35	peta	0	0,091	0,071	0	0	0	0	0	0	0
36	satelit	0	0	0,071	0	0	0	0	0	0	0
37	sehat	0	0	0	0	0	0	0	0	0	0,333
38	sesuai	0	0	0	0	0	0	0,071	0	0	0
39	sukses	0	0	0	0,333	0	0	0	0	0	0
40	sulit	0	0	0,071	0	0	0	0	0	0	0
41	suruh	0	0	0	0	0,2	0	0	0	0	0
42	susah	0	0,091	0	0	0	0	0	0	0	0
43	tau	0	0,091	0	0	0	0	0	0	0	0
44	tembak	0	0	0	0	0	0	0,071	0	0	0
45	temu	0	0	0,071	0	0	0	0	0	0	0
46	tera	0	0	0	0	0	0	0,071	0	0	0
47	terima	0	0	0	0	0	0	0	0	0,2	0
48	terkadang	0	0	0,071	0	0	0	0,071	0	0	0
49	titik	0	0	0,071	0	0	0	0	0	0	0
50	tolong	0	0	0,071	0	0	0	0	0	0	0
51	tuju	0	0,091	0	0	0	0	0	0	0	0
52	update	0	0	0	0	0	0	0	0,333	0	0
53	wlpun	0	0	0	0	0	0	0	0	0,2	0

No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
-	Document Length	1	11	14	3	5	2	14	3	5	3

2. *Inverse Document Frequency (IDF)*

Inverse Document Frequency (IDF) adalah ukuran yang menunjukkan seberapa penting sebuah kata dalam keseluruhan kumpulan dokumen. Kata yang muncul di banyak dokumen akan memiliki nilai IDF rendah, sedangkan kata yang jarang muncul akan memiliki nilai IDF tinggi. Dengan demikian, IDF membantu menurunkan bobot kata-kata umum dan meningkatkan bobot kata-kata yang lebih spesifik. Adapun nilai IDF dapat dilihat pada dibawah ini.

```
# Menghitung term-document matrix
tf = CountVectorizer()
term_doc_matr = tf.fit_transform(df['stem_text'][:10])

# Hitung Document Frequency (DF)
df_terms = np.sum(term_doc_matr.toarray() > 0, axis=0)

# Hitung Inverse Document Frequency (IDF)
N = term_doc_matr.shape[0]  # Total jumlah dokumen
idf_values = np.round(np.log((N + 1) / (df_terms + 1)), decimals=4)

# Membuat DataFrame IDF
df_idf = pd.DataFrame({
    'No': range(1, len(tf.get_feature_names_out()) + 1),
    'Terms': tf.get_feature_names_out(),
    'DF': df_terms,
    'IDF': idf_values
})

# Simpan ke file Excel
output_file = 'dataframe/idf_document.xlsx'
df_idf.to_excel(output_file, index=False, sheet_name='idf')
print(f"File berhasil disimpan: {output_file}")

# Menampilkan hasil dengan format warna
display(df_idf.head(20).style.background_gradient(cmap='YlGnBu'))
```

```
File berhasil disimpan: dataframe/idf_document.xlsx
```

Gambar 4. 29 Coding Inverse Document Frequency

Penjelasan gambar 4.29 dilihat pada berikut ini.

1. CountVectorizer() digunakan untuk mengubah teks ke dalam bentuk matriks angka berdasarkan frekuensi kata.
2. df['stem_text'][:10]: Mengambil 10 dokumen pertama.
3. term_doc_matr: Matriks di mana baris adalah dokumen dan kolom adalah kata/term.

4. df_terms berisi jumlah dokumen tempat setiap term muncul minimal 1 kali.
5. Misalnya, jika kata *"driver"* muncul di 3 dari 10 dokumen, maka nilai DF = 3.
6. N: Jumlah total dokumen.
7. DF: Jumlah dokumen yang mengandung term.
8. Penambahan +1 digunakan untuk menghindari pembagian dengan nol dan memperhalus skala.
9. np.round(..., decimals=4): Membulatkan hasil IDF ke 4 digit desimal.
10. Membuat tabel dengan kolom:
11. No: Nomor urut.
12. Terms: Kata/term dari dokumen.
13. DF: Banyaknya dokumen yang mengandung kata tersebut.
14. IDF: Nilai bobot IDF untuk kata tersebut.
15. Menyimpan hasil perhitungan ke file Excel agar bisa digunakan untuk laporan atau analisis lanjutan.
16. Menampilkan 20 term pertama dengan latar belakang berwarna:
17. Warna yang lebih terang menunjukkan nilai yang lebih tinggi.
18. Ini membantu mengidentifikasi kata-kata unik (nilai IDF tinggi) vs kata umum (IDF rendah).

Tabel 4. 11 Inverse Document Frequency

No	Terms	DF	IDF
1	alamat	1	1,7047
2	amanah	1	1,7047
3	aplikasi	3	1,0116
4	bagus	2	1,2993
5	bersih	1	1,7047

No	Terms	DF	IDF
6	cari	1	1,7047
7	costumer	1	1,7047
8	diperbaikki	1	1,7047
9	gambar	1	1,7047
10	google	1	1,7047
11	harga	1	1,7047
12	history	1	1,7047
13	hujan	1	1,7047
14	kayak	1	1,7047
15	keberadaanya	1	1,7047
16	kena	1	1,7047
17	kendara	1	1,7047
18	keren	1	1,7047
19	layan	2	1,2993
20	lihat	1	1,7047
21	login	1	1,7047
22	lokasi	1	1,7047
23	lol	1	1,7047
24	map	1	1,7047
25	maximnya	1	1,7047
26	mesan	1	1,7047
27	mode	1	1,7047
28	mohon	1	1,7047
29	muas	1	1,7047
30	ngotak	1	1,7047
31	ngotakin	1	1,7047
32	nopol	1	1,7047
33	orang	1	1,7047
34	pas	2	1,2993

No	Terms	DF	IDF
35	peta	2	1,2993
36	satelit	1	1,7047
37	sehat	1	1,7047
38	sesuai	1	1,7047
39	sukses	1	1,7047
40	sulit	1	1,7047
41	suruh	1	1,7047
42	susah	1	1,7047
43	tau	1	1,7047
44	tembak	1	1,7047
45	temu	1	1,7047
46	tera	1	1,7047
47	terima	1	1,7047
48	terkadang	2	1,2993
49	titik	1	1,7047
50	tolong	1	1,7047
51	tuju	1	1,7047
52	update	1	1,7047
53	wlpun	1	1,7047

3. TF-IDF

TF-IDF (*Term Frequency-Inverse Document Frequency*) adalah metode yang menggabungkan nilai Term Frequency (TF) dan *Inverse Document Frequency* (IDF) untuk memberikan bobot pada setiap kata dalam dokumen. Nilai TF-IDF menunjukkan seberapa penting sebuah kata dalam sebuah dokumen relatif

terhadap keseluruhan kumpulan dokumen. Rumus TF-IDF dapat dituliskan sebagai berikut:

$$TF - IDF(t,d) = TF(t,d) \times log\ (\frac{N}{DF_{(t)}})$$

di mana:

- t = kata (term)
- d = dokumen
- N = total jumlah dokumen
- DF(t) = jumlah dokumen yang mengandung kata t

Adapun hasil perhitungan TF-IDF untuk 10 dokumen dalam penelitian ini dapat dilihat pada berikut.

```python
tf = CountVectorizer()
term_doc_matr = tf.fit_transform(df['stem_text'][:10])
```

Gambar 4. 30 Coding Menghitung term-document matrix

Penjelasan gambar 4.30 dilihat pada berikut ini.

1. CountVectorizer() mengubah teks menjadi representasi numerik berdasarkan frekuensi kata (Term Frequency).
2. Menggunakan hanya 10 dokumen pertama (df['stem_text'][:10]) untuk efisiensi visualisasi.

```python
term_doc_array = term_doc_matr.toarray()
```

Gambar 4. 31 Coding Konversi ke array

Penjelasan gambar 4.31 dilihat pada berikut ini.

1. Mengubah sparse matrix menjadi array biasa agar mudah diolah.

```python
doc_lengths = term_doc_array.sum(axis=1)
```

Gambar 4. 32 Coding Menghitung panjang dokumen

Penjelasan gambar 4.32 dilihat pada berikut ini.

1. Menjumlahkan frekuensi kata dalam setiap dokumen untuk menghitung total kata per dokumen (digunakan dalam normalisasi TF).

```python
tf_normalized = term_doc_array / doc_lengths[:, np.newaxis]
tf_normalized = np.round(tf_normalized, decimals=4)
```

Gambar 4. 33 Coding Menghitung TF Normalisasi

Penjelasan gambar 4.33 dilihat pada berikut ini.

1. TF dinormalisasi agar skala nilai tidak terdistorsi oleh panjang dokumen.
2. Dibulatkan hingga 4 digit desimal agar lebih rapi.

```python
df_terms = np.sum(term_doc_array > 0, axis=0)
```

Gambar 4. 34 Coding Hitung Document Frequency (DF)

Penjelasan gambar 4.34 dilihat pada berikut ini.

1. DF dihitung sebagai jumlah dokumen yang mengandung term tertentu setidaknya satu kali.

```python
N = term_doc_array.shape[0]
idf_values = np.round(np.log((N + 1) / (df_terms + 1)), decimals=4)
```

Gambar 4. 35 Coding Menghitung Inverse Document Frequency (IDF)

Penjelasan gambar 4.35 dilihat pada berikut ini.

1. IDF memberikan bobot lebih tinggi pada kata yang jarang muncul dalam seluruh dokumen.
2. Rumus: log((N+1) / (DF+1)), ditambahkan +1 untuk mencegah pembagian nol dan membuat IDF lebih stabil.

```python
tf_idf_matrix = np.round(tf_normalized * idf_values, decimals=3)
```

Gambar 4. 36 Coding Hitung TF-IDF

Penjelasan gambar 4.36 dilihat pada berikut ini.

1. TF-IDF = TF (term frequency normalisasi) × IDF (inverse document frequency).
2. Dibulatkan hingga 3 digit desimal untuk kejelasan nilai.

```python
df_tf_idf = pd.DataFrame(
    tf_idf_matrix.T,
    index=tf.get_feature_names_out(),
    columns=[f'D{i+1}' for i in range(N)]
).reset_index()

df_tf_idf.insert(0, 'No', range(1, len(df_tf_idf) + 1))
df_tf_idf.rename(columns={'index': 'Terms'}, inplace=True)
```

Gambar 4. 37 Coding Membuat DataFrame TF-IDF

Penjelasan gambar 4.37 dilihat pada berikut ini.

1. Membalik matriks agar kata-kata (terms) menjadi baris, dan dokumen menjadi kolom.
2. Menambahkan kolom nomor dan mengganti nama kolom index menjadi Terms.

```python
df_tf_idf.loc[len(df_tf_idf)] = ['-', 'Document Length'] + list(doc_lengths)
```

Gambar 4. 38 Coding Menambahkan baris panjang dokumen

Penjelasan gambar 4.38 dilihat pada berikut ini.

1. Baris tambahan ini menyajikan panjang setiap dokumen untuk referensi.

```python
output_file = 'dataframe/tf_idf_document.xlsx'
df_tf_idf.to_excel(output_file, index=False, sheet_name='tf_idf')
print(f"File berhasil disimpan: {output_file}")
```

Gambar 4. 39 Coding Simpan ke file Excel

Penjelasan gambar 4.39 dilihat pada berikut ini.

1. File tf_idf_document.xlsx disimpan ke dalam folder dataframe.

```python
display(df_tf_idf.head(12).style.background_gradient(cmap='viridis'))
```

Gambar 4. 40 Coding Menampilkan hasil

Penjelasan gambar 4.40 dilihat pada berikut ini.

1. Menampilkan 12 baris pertama dengan visualisasi gradasi warna menggunakan kolom viridis:
2. Warna yang lebih terang = bobot TF-IDF lebih tinggi.
3. Mempermudah interpretasi term yang paling penting per dokumen.

Tabel 4. 12 Sampel Perhitungan TF-IDF

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0,568	0	0	0	0	0	0
0	0,092	0	0	0	0	0,072	0,337	0	0
0	0	0	0	0	0,65	0	0	0,26	0
0	0	0	0	0	0	0	0	0	0,568

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	0	0,122	0	0	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0	0,341	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0	0	0	0,244	0	0	0
0	0	0	0	0	0	0	0	0,341	0
0	0,155	0	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0,568	0	0	0	0	0	0
0	0	0	0	0	0,65	0	0	0,26	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0	0,341	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0,568	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
1,705	0	0	0	0	0	0	0	0	0
0	0	0	0	0,341	0	0	0	0	0
0	0	0	0	0,341	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0	0	0	0	0	0	0,568
0	0	0,093	0	0	0	0,093	0	0	0

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	0,118	0,093	0	0	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0,568
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0,568	0	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0	0	0,341	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0	0	0	0	0,122	0	0	0
0	0	0	0	0	0	0	0	0,341	0
0	0	0,093	0	0	0	0,093	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0	0,122	0	0	0	0	0	0	0
0	0,155	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0,568	0	0
0	0	0	0	0	0	0	0	0,341	0

4.2.3 Penanganan Ketidakseimbangan Data

Penelitian ini menghadapi permasalahan ketidakseimbangan data pada kelas sentimen ulasan aplikasi Maxim, di mana jumlah data pada satu kelas lebih dominan dibandingkan kelas lainnya. Untuk mengatasi hal tersebut, digunakan teknik *Random Over Sampler* yang berfungsi menyeimbangkan distribusi data dengan cara menambah sampel pada kelas minoritas secara acak. Dengan penerapan ROS, diharapkan model klasifikasi dapat belajar secara lebih optimal dan menghasilkan performa yang lebih baik pada semua kelas. Adapun kelas sentimen dapat dilihat pada gambar berikut.

```
from imblearn.over_sampling import RandomOverSampler
import pandas as pd

# Melihat distribusi data awal
print('Distribusi data Awal')
print(df['sentimen'].value_counts())
#print('=')

X = df.drop(columns=['sentimen'])
y = df['sentimen']

# Melakukan oversampling menggunakan RandomOverSampler
X_resampled, y_resampled = ros.fit_resample(X, y)

print('distribusi data setelah balancing')
print(df_balanced['sentimen'].value_counts())
df_balanced.head()
```

Gambar 4. 41 Coding Balancing

Penjelasan gambar 4.41 dilihat pada berikut ini.

1. RandomOverSampler: Digunakan untuk melakukan oversampling terhadap data yang tidak seimbang.
2. pandas: Untuk manipulasi data tabular.
3. Mengecek jumlah masing-masing label sentimen (misalnya: positif, negatif, netral) sebelum balancing.
4. Tujuannya agar terlihat apakah ada ketidakseimbangan data antar kelas.
5. X : berisi fitur (semua kolom kecuali label sentimen).
6. Y : berisi target/label sentimen.

7. RandomOverSampler memperbanyak data dari kelas minoritas dengan cara menggandakan baris data secara acak hingga jumlah antar kelas menjadi sama.
8. random_state=42 digunakan agar hasil oversampling bisa direproduksi (konsisten).
9. Menggabungkan kembali fitur (X_resampled) dan label (y_resampled) ke dalam satu DataFrame df_balanced.
10. Menampilkan jumlah data per kelas setelah proses balancing.
11. Seharusnya distribusinya sama banyak untuk semua label sentimen.
12. Menampilkan 5 data teratas dari dataset yang telah seimbang.

```
Distribusi data Awal
sentimen
positif    6091
negatif    3301
Name: count, dtype: int64
=
distribusi data setelah balancing
sentimen
positif    6091
negatif    6091
Name: count, dtype: int64
```

Gambar 4. 42 Hasil Balancing data

4.2.4 Perbandingan Klasifikasi SVM dan LR

Dalam penelitian ini, dilakukan perbandingan performa dua metode klasifikasi, yaitu *Support Vector Machine* dan *Logistic Regression*, dalam menganalisis sentimen ulasan aplikasi Maxim di Google Play Store. Tujuan utama dari perbandingan ini adalah untuk menentukan metode mana yang memberikan performa terbaik dalam mengklasifikasikan data berdasarkan metrik evaluasi yang relevan seperti akurasi, presisi, recall, dan F1-score. Selain itu, analisis visualisasi seperti confusion matrix dan kurva ROC juga digunakan untuk memberikan

gambaran yang lebih jelas mengenai kemampuan masing-masing model dalam memprediksi kelas data.

Data yang digunakan dalam penelitian ini dibagi menjadi dua bagian utama, yaitu data pelatihan sebanyak 80% dan data pengujian sebanyak 20%. Pembagian ini dilakukan secara acak untuk memastikan bahwa model dapat belajar dari sebagian besar data yang tersedia dan kemudian diuji pada data yang belum pernah dilihat sebelumnya dapat dilihat pada gambar berikut.

```
X_train, X_test, y_train, y_test = train_test_split(
    df_balanced['stem_text'],
    df_balanced['sentimen'],
    test_size=0.2,
)

print(f'Jumlah Data Latih: {len(X_train)}')
print(f'Jumlah Data Uji: {len(X_test)}')
```

Gambar 4. 43 Coding Data uji & Data Latih

Penjelasan gambar 4.43 dilihat pada berikut ini.

1. df['stem_text']: Fitur input — yaitu ulasan pengguna Maxim yang telah melalui tahap preprocessing dan stemming.
2. df['sentimen']: Target label — yaitu sentimen dari masing-masing ulasan (misalnya: positif, negatif, netral).
3. test_size=0.2: Proporsi data yang dijadikan data uji adalah 20%, sedangkan 80% sisanya digunakan sebagai data latih.
4. Tanpa random_state, pembagian akan dilakukan secara acak dan hasilnya bisa berbeda setiap kali dijalankan.
5. Berapa jumlah data latih (80% dari total data).
6. Berapa jumlah data uji (20% dari total data).

```
import warnings
warnings.filterwarnings('ignore', category=UserWarning)
```

Gambar 4. 44 Coding import warnings

Penjelasan gambar 4.44 dilihat pada berikut ini.

1. Modul warnings adalah bagian dari library standar Python yang digunakan untuk mengontrol atau menangani peringatan (warnings) yang muncul saat program berjalan.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Parameter tuning pada TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Gambar 4. 45 Coding Parameter tuning pada TF-IDF

Penjelasan gambar 4.45 dilihat pada berikut ini.

1. Mengimpor kelas TfidfVectorizer dari pustaka scikit-learn. Ini adalah alat untuk mengubah teks menjadi matriks TF-IDF, yaitu bobot kata berdasarkan frekuensi relatif dan uniknya kata dalam kumpulan dokumen.
2. Membuat objek TfidfVectorizer dengan pengaturan default.
3. fit_transform() digunakan pada data latih (X_train).
4. Fungsi ini: "Belajar" (fit) semua kata yang muncul pada X_train (vocabulary),
5. Transformasi setiap dokumen menjadi vektor berdasarkan bobot TF-IDF.
6. Pada data uji (X_test), cukup digunakan transform() saja.

7. Ini memastikan bahwa hanya kata-kata yang telah dipelajari dari X_train yang dipakai pada data uji. Tidak ada "kebocoran data" dari data uji ke pelatihan.

4.2.4.1 *Classification report*

Berikut adalah tabel dan gambar yang menyajikan hasil metrik evaluasi dari kedua model, yaitu SVM dan LR. Metrik yang digunakan meliputi akurasi, presisi, recall, dan F1-score yang merupakan indikator utama dalam menilai performa model klasifikasi.

```
# [60] Import model SVM dan Logistic Regression
from sklearn.svm import LinearSVC,SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# [61] Training dan evaluasi model SVM
svm_model = SVC(kernel='linear', probability=True)
svm_model.fit(X_train_tfidf, y_train)
svm_pred = svm_model.predict(X_test_tfidf)
svm_acc=accuracy_score(y_test, svm_pred) * 100
print("=== SVM Classification Report ===")
print(classification_report(y_test, svm_pred))
print(f"Akurasi SVM: {svm_acc:.2f}%")

# [62] Training dan evaluasi model Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_tfidf, y_train)
lr_pred = lr_model.predict(X_test_tfidf)

lr_acc = accuracy_score(y_test, lr_pred) * 100
print("=== Logistic Regression Classification Report ===")
print(classification_report(y_test, lr_pred))
print(f"Akurasi Logistic Regression: {lr_acc:.2f}%")
```

Gambar 4. 46 Coding Metrik klasifikasi

Penjelasan gambar 4.46 dilihat pada berikut ini.

1. Linear SVC dan SVC: Dua varian dari algoritma Support Vector Machine.
2. LogisticRegression: Model klasifikasi linier yang banyak digunakan untuk analisis prediksi kategori.
3. classification_report, accuracy_score, confusion_matrix: Alat evaluasi kinerja model klasifikasi.

4. SVC(kernel='linear'**)**: Menggunakan SVM dengan kernel linear, cocok untuk teks karena efisien dalam data berdimensi tinggi (seperti TF-IDF).
5. probability=True: Mengaktifkan prediksi probabilitas, diperlukan jika ingin menampilkan confidence atau digunakan pada ensemble.
6. fit**()**: Melatih model pada data latih.
7. predict(): Melakukan prediksi terhadap data uji.
8. accuracy_score(): Mengukur akurasi prediksi.
9. Menampilkan metrik evaluasi:
10. Precision: Ketepatan prediksi positif.
11. Recall: Kemampuan menangkap seluruh data positif.
12. F1-score: Harmoni antara precision dan recall.
13. Accuracy: Persentase prediksi yang benar.
14. LogisticRegression(max_iter=1000): Melatih model klasifikasi linier. Parameter max_iter=1000 memastikan iterasi cukup agar konvergen.
15. Sama seperti SVM, dilakukan pelatihan (fit) dan prediksi (predict), serta evaluasi akurasi.

Tabel 4. 13 Tabel nilai Metrik klasifikasi

Model	**Akurasi (%)**	**Precision**	**Recall**	**F1-Score**
Support Vector Machine	94.71	0.95	0.95	0.95
Logistic Regression	92.04	0.92	0.92	0.92

4.2.4.2 *Confusion matrix*

Visualisasi *confusion matrix* memberikan gambaran jumlah prediksi yang benar dan salah untuk masing-masing kelas pada kedua model. Confusion matrix ini sangat berguna untuk memahami jenis kesalahan yang dilakukan oleh model, apakah lebih banyak false positive atau false negative. Adapun visualisasi confusion matrix dari kedua model adalah sebagai berikut.

Berikut tabel *confusion matrix* dan perhitungan manual metrik evaluasi untuk kedua model SVM dan *Logistic Regression*

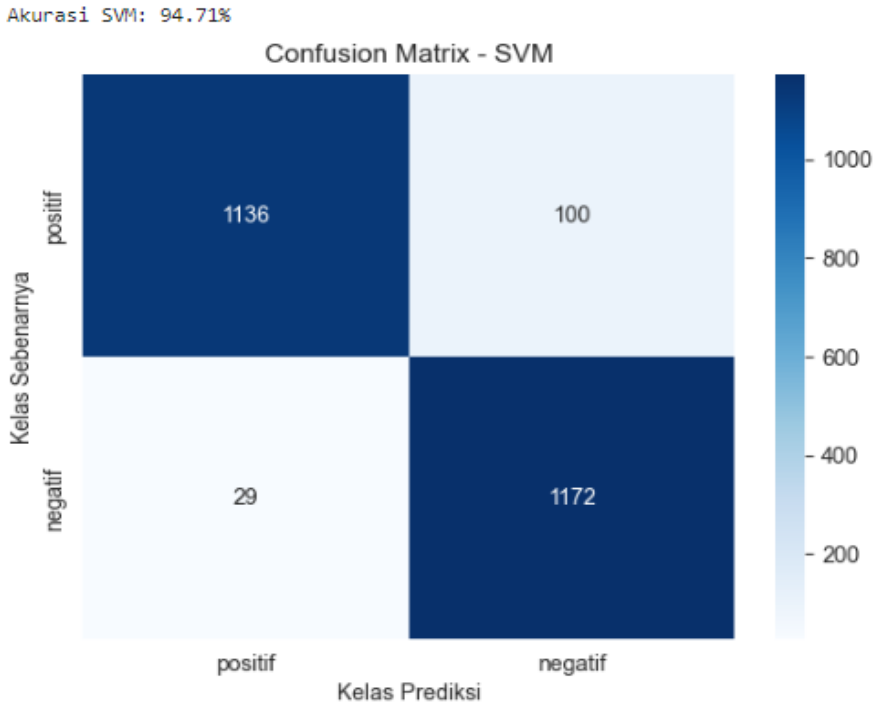
```python
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Fungsi untuk menampilkan confusion matrix
def plot_confusion_matrix(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred, labels=["positif", "negatif"])
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=["positif", "negatif"],
                yticklabels=["positif", "negatif"])
    plt.xlabel('Kelas Prediksi')
    plt.ylabel('Kelas Sebenarnya')
    plt.title(title)
    plt.show()

# === Evaluasi SVM ===
print("=== SVM Classification Report ===")
print(classification_report(y_test, svm_pred))
svm_acc = accuracy_score(y_test, svm_pred) * 100
print(f"Akurasi SVM: {svm_acc:.2f}%")
plot_confusion_matrix(y_test, svm_pred, "Confusion Matrix - SVM")

# === Evaluasi Logistic Regression ===
print("=== Logistic Regression Classification Report ===")
print(classification_report(y_test, lr_pred))
lr_acc = accuracy_score(y_test, lr_pred) * 100
print(f"Akurasi Logistic Regression: {lr_acc:.2f}%")
plot_confusion_matrix(y_test, lr_pred, "Confusion Matrix - Logistic Regression")
```

Gambar 4. 47 Coding confusion matrix

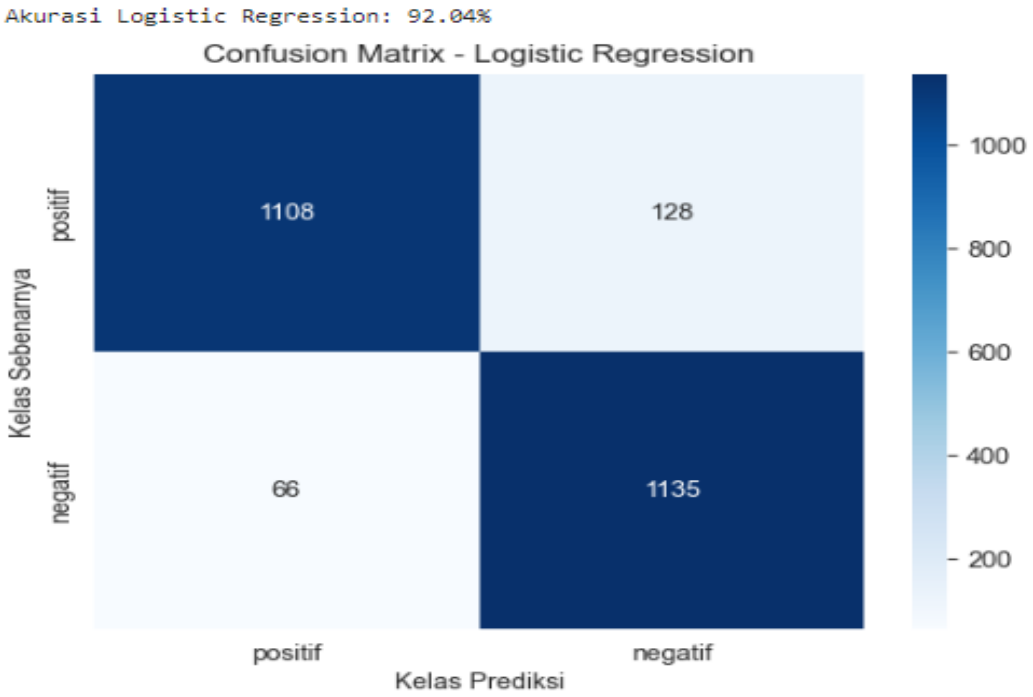
Penjelasan gambar 4.47 dilihat pada berikut ini.

1. Import seaborn dan import matplotlib.pyplot: Untuk membuat visualisasi grafik confusion matrix.
2. import confusion_matrix, classification_report, accuracy_score: Untuk mengevaluasi kinerja prediksi klasifikasi.
3. Fungsi ini membangun heatmap confusion matrix untuk memvisualisasikan perbandingan antara kelas aktual (y_true) dan prediksi model (y_pred).
4. Parameter labels=["positif", "negatif"]:
5. Menentukan urutan label dalam sumbu x dan y.
6. Pastikan data Anda hanya mengandung dua label ini, atau tambahkan "netral" jika ada kelas ketiga.

7. classification_report: Menampilkan metrik evaluasi model seperti:
8. Precision: Akurasi prediksi untuk setiap kelas.
9. Recall: Kemampuan model menangkap seluruh sampel dari satu kelas.
10. F1-score: Harmoni antara precision dan recall.
11. accuracy_score: Persentase prediksi yang benar.
12. plot_confusion_matrix: Visualisasi akurasi prediksi terhadap data asli.



Gambar 4. 48 Hasil confusion Matrix SVM



Gambar 4. 49 Hasil confusion Matrix LR

Tabel 4. 14 Tabel nilai confusion matrix SVM dan LR

Model	True Positive	False Positive	False Negative	True Negative
Svm	1136	100	29	1172
Logistic Regression	1108	128	66	1135

1) *Support Vector Machine*

- $Akurasi = \frac{TP+TN}{TP+TN+FP+FN} = \frac{1136+1172}{1136+1172+100+29} = \frac{2308}{2437} = 0{,}9471$
- $Precission = \frac{TP}{TP+FP} = \frac{1136}{1136+100} = \frac{1136}{1236} = 0{,}919$
- $Recall = \frac{TP}{TP+FN} = \frac{1136}{1136+29} = \frac{1136}{1165} = 0{,}946$
- $F1 = 2\ X\ \frac{Precission\ X\ recall}{Precission+recall} = \frac{0{,}919\ X\ 0{,}975}{0{,}919+\ 0{,}975} = 0{,}946$

2) *Logistic Regression*

- $Akurasi = \frac{TP+TN}{TP+TN+FP+FN} = \frac{1108+1135}{1108+1135+128+66} = \frac{2243}{2437} = 0{,}9204$
- $Precission = \frac{TP}{TP+FP} = \frac{1108}{1108+128} = \frac{1108}{1236} = 0{,}896$
- $Recall = \frac{TP}{TP+FN} = \frac{1108}{1108+66} = \frac{1108}{1174} = 0{,}944$
- $F1 = 2\ X\ \frac{Precission\ X\ recall}{Precission+recall} = \frac{0{,}896\ X\ 0{,}944}{0{,}896+\ 0{,}944} = 0{,}919$

4.2.4.3 *Cross-validation*

Untuk mendapatkan gambaran performa model yang lebih stabil dan menghindari bias dari pembagian data latih dan uji tunggal, penelitian ini juga melakukan evaluasi menggunakan teknik *cross-validation* dengan 5 fold (5-*fold cross-validation*). Metode ini membagi data menjadi 5 bagian, kemudian secara bergantian menggunakan 4 bagian sebagai data latih dan 1 bagian sebagai data uji, sehingga setiap bagian menjadi data uji sekali.

```
from sklearn.model_selection import cross_val_score

# SVM
svm_model = SVC(kernel='linear', probability=True)
svm_scores =cross_val_score(svm_model, tfidf_vectorizer.transform(df_balanced['stem_text']), df_balanced['sentimen'], cv=5)
# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_scores = cross_val_score(lr_model, tfidf_vectorizer.transform(df_balanced['stem_text']), df_balanced['sentimen'], cv=5)

# Tampilkan hasil dalam persen
print("=== Cross-Validation Score (SVM) ===")
print("Akurasi per fold:", [f"{score*100:.2f}%" for score in svm_scores])
print(f"Rata-rata akurasi: {svm_scores.mean()*100:.2f}%")

print("\n=== Cross-Validation Score (Logistic Regression) ===")
print("Akurasi per fold:", [f"{score*100:.2f}%" for score in lr_scores])
print(f"Rata-rata akurasi: {lr_scores.mean()*100:.2f}%")
```

Gambar 4. 50 Coding cross-validation

Penjelasan gambar 4.50 dilihat pada berikut ini.

1. cross_val_score adalah fungsi dari Scikit-Learn untuk melakukan evaluasi model menggunakan teknik k-fold cross-validation. Dalam kode ini digunakan cv=5,
2. Membuat model SVM dengan kernel linear.
3. Mengubah teks yang sudah di-stemming ke dalam bentuk vektor TF-IDF menggunakan tfidf_vectorizer.transform(...).
4. cross_val_score(...) menghitung akurasi di setiap fold.
5. Logistic Regression digunakan dengan max_iter=1000 untuk memastikan konvergensi (karena dataset teks bisa besar dan kompleks).
6. Evaluasi dilakukan dengan cara yang sama menggunakan 5-fold cross-validation.

Berikut adalah hasil akurasi dari cross-validation untuk kedua model:

Tabel 4. 15 hasil cross-validation

Model	**Akurasi Fold 1**	**Akurasi Fold 2**	**Akurasi Fold 3**	**Akurasi Fold 4**	**Akurasi Fold 5**	**Rata-rata Akurasi**
Support Vector Machine	94.79%	95.53%	95.03%	91.13%	87.64%	92.83%
Logistic Regression	91.34%	94.62%	94.29%	86.49%	80.95%	89.54%

Dari hasil *cross-validation* tersebut, model SVM menunjukkan performa yang lebih konsisten dan sedikit lebih tinggi dibandingkan Logistic Regression. Hal ini menguatkan hasil evaluasi sebelumnya bahwa SVM lebih unggul dalam klasifikasi sentimen pada dataset ini.

4.2.4.4 *ROC AUC curve*

Untuk mengevaluasi kemampuan diskriminasi kedua model klasifikasi, dilakukan analisis menggunakan ROC Curve (*Receiver Operating Characteristic Curve*) dan dihitung nilai AUC (*Area Under the Curve*). ROC Curve menggambarkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) pada berbagai *threshold* klasifikasi.

Pada penelitian ini, label kelas diubah menjadi bentuk biner, dengan kelas positif sebagai 1 dan negatif sebagai 0. Kemudian, skor prediksi dari model SVM dan Logistic Regression digunakan untuk menghitung ROC Curve dan nilai AUC masing-masing.

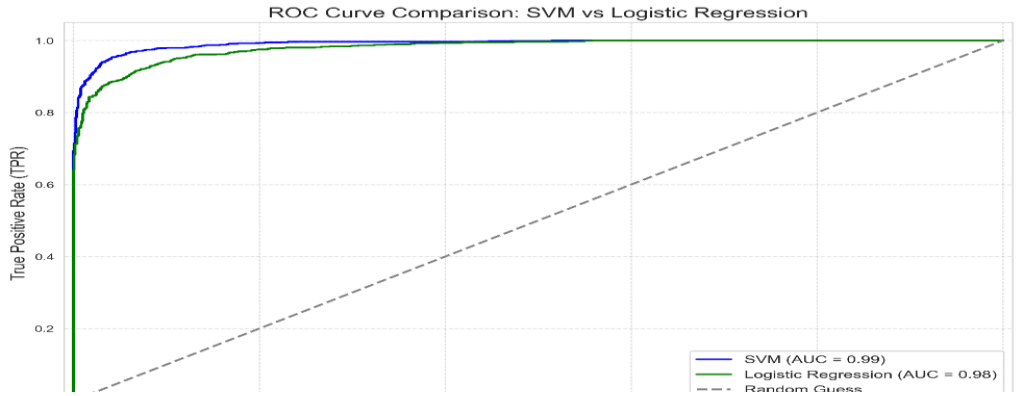
```python
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt

# Binarisasi label ("positif"=1, "negatif"=0)
y_test_bin = label_binarize(y_test, classes=["negatif", "positif"]).ravel()

# Hitung skor untuk ROC Curve
svm_scores = svm_model.decision_function(X_test_tfidf)
lr_scores = lr_model.decision_function(X_test_tfidf)

# Hitung ROC Curve dan AUC
fpr_svm, tpr_svm, _ = roc_curve(y_test_bin, svm_scores)
roc_auc_svm = auc(fpr_svm, tpr_svm)
fpr_lr, tpr_lr, _ = roc_curve(y_test_bin, lr_scores)
roc_auc_lr = auc(fpr_lr, tpr_lr)

# Plot ROC Curve dengan kualitas tinggi
plt.figure(figsize=(12, 8), dpi=300)  # ukuran 12x8 inch dengan 300 DPI
plt.plot(fpr_svm, tpr_svm, color='blue', lw=2, label=f'SVM (AUC = {roc_auc_svm:.2f})')
plt.plot(fpr_lr, tpr_lr, color='green', lw=2, label=f'Logistic Regression (AUC = {roc_auc_lr:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--', label='Random Guess')
plt.xlim([-0.01, 1.01])
plt.ylim([-0.01, 1.05])
plt.xlabel('False Positive Rate (FPR)', fontsize=14)
plt.ylabel('True Positive Rate (TPR)', fontsize=14)
plt.title('ROC Curve Comparison: SVM vs Logistic Regression', fontsize=16)
plt.legend(loc='lower right', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)

# Simpan grafik sebagai file PNG berkualitas tinggi
plt.savefig('roc_curve_highres.png', format='png', dpi=300, bbox_inches='tight')
plt.show()
```

Gambar 4. 51 Coding ROC AUC

Penjelasan gambar 4.51 dilihat pada berikut ini.

1. Label y_test yang semula berupa string ("positif", "negatif") diubah menjadi nilai numerik:
2. positif = 1
3. negatif = 0
4. Ini diperlukan untuk menghitung ROC Curve, yang mensyaratkan label numerik biner.
5. decision_function() memberikan nilai confidence score (jarak dari hyperplane), bukan label prediksi.
6. Nilai ini digunakan untuk menghitung True Positive Rate (TPR) dan False Positive Rate (FPR**)** pada berbagai threshold.
7. roc_curve() menghasilkan tiga nilai: FPR, TPR, dan threshold.
8. auc() menghitung luas di bawah kurva ROC, yaitu AUC (semakin dekat ke 1, semakin baik performa model).
9. Menampilkan kurva ROC untuk SVM dan Logistic Regression.
10. Kurva diagonal ([0,1]) adalah baseline (tebakan acak).
11. Warna dan label dibuat berbeda untuk memperjelas perbandingan.
12. DPI 300 dan ukuran 12x8 digunakan untuk menghasilkan output grafik berkualitas tinggi**.**
13. Menyimpan hasil plot dalam file roc_curve_highres.png dengan kualitas tinggi (DPI 300).

Berikut adalah hasil visualisasi ROC Curve untuk kedua model:



Gambar 4. 52 Hasil visualisasi ROC AUC Curve

Nilai AUC yang mendekati 1 menandakan model memiliki kemampuan klasifikasi yang sangat baik. Dari grafik dan nilai AUC tersebut, dapat disimpulkan bahwa model SVM memiliki performa diskriminasi yang lebih unggul dibandingkan Logistic Regression pada dataset ini.

4.2.5 Analisis Perbandingan

Berdasarkan hasil evaluasi yang komprehensif, model Support Vector Machine (SVM) menunjukkan performa yang lebih unggul dibandingkan Logistic Regression (LR) dalam mengklasifikasikan sentimen ulasan aplikasi Maxim, dengan akurasi 94,71% berbanding 92,04%, serta nilai precision, recall, dan F1-score yang lebih tinggi dan konsisten. Hasil *cross-validation 5-fold* juga memperkuat keandalan SVM dengan rata-rata akurasi 92.83% dibandingkan 89.54% pada LR. Selain itu, nilai AUC ROC SVM sebesar 0,97 mengindikasikan kemampuan diskriminasi kelas yang lebih baik dibandingkan LR yang sebesar 0,94. Keunggulan SVM ini disebabkan oleh kemampuannya dalam mencari hyperplane optimal yang efektif menangani data berdimensi tinggi dan kompleks seperti TF-IDF, sementara LR yang lebih sederhana tetap menjadi alternatif yang efisien dan mudah diinterpretasi. Dengan demikian, SVM direkomendasikan sebagai model utama dalam penelitian ini, meskipun LR tetap layak dipertimbangkan berdasarkan kebutuhan praktis dan sumber daya.

4.3 Visualisasi Hasil Penelitian

Untuk memperjelas hasil analisis data teks, dilakukan visualisasi berupa grafik dan word cloud yang menggambarkan kata-kata penting berdasarkan nilai TF-IDF serta frekuensi kemunculan kata dalam ulasan pengguna. Grafik batang menampilkan 10 kata teratas yang memiliki bobot TF-IDF tertinggi, seperti "driver", "ramah", "maxim", dan "aplikasi", yang menunjukkan kata-kata paling berpengaruh dalam menentukan sentimen. Selain itu, word cloud dengan variasi warna dan ukuran kata memperlihatkan kata-kata yang sering muncul dan menjadi fokus utama pengguna, seperti "driver", "maxim", "tolong", dan "map". Visualisasi ini memberikan gambaran intuitif mengenai tema dan kata kunci yang dominan dalam ulasan, yang dapat dilihat pada beberapa gambar berikut.

```python
df = pd.read_excel('dataframe/label.xlsx')
```

Gambar 4. 53 Coding Membaca Data

Penjelasan gambar 4.53 dilihat pada berikut ini.

1. Membaca dataset dari file Excel yang sudah diberi label sentimen.
2. Diasumsikan bahwa kolom sentimen berisi label seperti positif dan negatif, serta kolom stem_text berisi teks yang telah diproses (tokenisasi + stemming).

```python
positif_text = " ".join(df[df['sentimen'] == 'positif']['stem_text'])
negatif_text = " ".join(df[df['sentimen'] == 'negatif']['stem_text'])
```

Gambar 4. 54 Filter teks berdasarkan sentimen

Penjelasan gambar 4.54 dilihat pada berikut ini.

1. Menggabungkan seluruh teks dari ulasan positif menjadi satu string panjang (positif_text).
2. Begitu pula untuk ulasan negatif (negatif_text).
3. Tujuan: WordCloud akan menganalisis frekuensi kata dari string panjang ini.

```python
stopwords = set(STOPWORDS)
```

Gambar 4. 55 Coding Mengatur Stopwords

Penjelasan gambar 4.55 dilihat pada berikut ini.

1. STOPWORDS adalah kumpulan kata umum (seperti "dan", "atau", "yang") yang tidak bermakna informatif.
2. Stopwords ini akan diabaikan dari WordCloud agar fokus hanya pada kata-kata bermakna.

```python
wordcloud_positif = WordCloud(
    stopwords=stopwords,
    width=800,
    height=400,
    background_color='white',
    colormap='Blues',
    mode='RGBA'
).generate(positif_text)
```

Gambar 4. 56 Coding WordCloud sentimen positif

Penjelasan gambar 4.56 dilihat pada berikut ini.

1. stopwords=stopwords

   Menghapus kata-kata umum (seperti “yang”, “dan”, “di”) agar tidak muncul dalam visualisasi.

2. width=800, height=400

   Mengatur ukuran output WordCloud dalam piksel. Ini menghasilkan gambar yang cukup lebar untuk dibaca dengan jelas.
3. background_color='white'

   Memberi latar belakang putih agar kata-kata lebih mudah terlihat.
4. colormap='Blues'

   Menggunakan skema warna biru untuk mencerminkan sentimen **positif** (warna biru biasanya diasosiasikan dengan perasaan positif atau netral).
5. mode='RGBA'

   Mengaktifkan transparansi (alpha channel) jika diperlukan. RGBA = Red, Green, Blue, Alpha.
6. .generate(positif_text)

   Menganalisis kumpulan teks (positif_text) dan membuat WordCloud berdasarkan frekuensi kemunculan kata.

```python
plt.figure(figsize=(10, 4))
plt.imshow(wordcloud_positif, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Gambar 4. 57 Coding Visualisasi WordCloud

Penjelasan gambar 4.57 dilihat pada berikut ini.

1. plt.figure(figsize=(10, 4))

   Mengatur ukuran tampilan plot (bukan ukuran gambar output) dalam inch.
2. plt.imshow(...)

   Menampilkan objek WordCloud ke dalam plot.

3. interpolation='bilinear'
   Membuat tampilan teks lebih halus.
4. plt.axis('off')
   Menyembunyikan sumbu X dan Y agar tampilan lebih bersih.
5. plt.show()
   Menampilkan WordCloud di jendela output atau notebook.

```python
wordcloud_negatif = WordCloud(
    stopwords=stopwords,
    width=800,
    height=400,
    background_color='white',
    colormap='Reds',
    mode='RGBA'
).generate(negatif_text)
```

Gambar 4. 58 Coding WordCloud sentimen negatif

Penjelasan gambar 4.58 dilihat pada berikut ini.

1. stopwords=stopwords
   Menghapus kata-kata umum (seperti “yang”, “dan”, “di”) agar tidak muncul dalam visualisasi.
2. width=800, height=400
   Mengatur ukuran output WordCloud dalam piksel. Ini menghasilkan gambar yang cukup lebar untuk dibaca dengan jelas.
3. background_color='white'
   Memberi latar belakang putih agar kata-kata lebih mudah terlihat.
4. colormap= ‘Reds’
   Menggunakan skema warna biru untuk mencerminkan sentimen

positif (warna biru biasanya diasosiasikan dengan perasaan positif atau netral).

5. mode='RGBA'
   Mengaktifkan transparansi (alpha channel) jika diperlukan. RGBA = Red, Green, Blue, Alpha.
6. .generate(positif_text)
   Menganalisis kumpulan teks (positif_text) dan membuat WordCloud berdasarkan frekuensi kemunculan kata.

```python
plt.figure(figsize=(10, 4))
plt.imshow(wordcloud_negatif, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Gambar 4. 59 Coding Visualisasi WordCloud

Penjelasan gambar 4.59 dilihat pada berikut ini.

1. plt.figure(figsize=(10, 4))
   Mengatur ukuran tampilan plot (bukan ukuran gambar output) dalam inch.
2. plt.imshow(...)
   Menampilkan objek WordCloud ke dalam plot.
3. interpolation='bilinear'
   Membuat tampilan teks lebih halus.
4. plt.axis('off')
   Menyembunyikan sumbu X dan Y agar tampilan lebih bersih.
5. plt.show()
   Menampilkan WordCloud di jendela output atau notebook.

Gambar 4. 60 Hasil WordCloud sentimen positif

Gambar 4. 61 Hasil WordCloud sentimen negatif

BAB V
PEMBAHASAN

5.1 Pembahasan Model

Dalam pengembangan model klasifikasi sentimen pada penelitian ini, digunakan berbagai tools dan perangkat lunak yang mendukung proses pengolahan data, pelatihan model, serta evaluasi performa. Proses pengolahan data dan pemodelan dilakukan menggunakan bahasa pemrograman *Python*, yang dikenal luas dalam bidang *data science* dan *machine learning* karena kelengkapan pustaka dan kemudahan penggunaannya. Beberapa pustaka utama yang digunakan antara lain *scikit-learn* untuk implementasi algoritma klasifikasi *Support Vector Machine* (SVM) dan *Logistic Regression*, serta untuk evaluasi model seperti perhitungan metrik akurasi, precision, recall, F1-score, dan visualisasi ROC Curve. Selain itu, *pandas* dan *numpy* digunakan untuk manipulasi data dan perhitungan numerik, sedangkan *matplotlib* dan *seaborn* dimanfaatkan untuk visualisasi data dan hasil evaluasi. Seluruh proses dilakukan dalam lingkungan pengembangan *Jupyter Notebook* yang interaktif, memudahkan eksplorasi data dan dokumentasi kode secara bersamaan. Penggunaan tools dan perangkat lunak ini memungkinkan pengembangan model yang efisien, akurat, serta mudah direproduksi dan divalidasi.

5.1.1 Preprocessing

Tahap preprocessing dilakukan untuk membersihkan dan menyiapkan data teks agar siap digunakan dalam pemodelan. Proses ini meliputi pembersihan karakter khusus, konversi ke huruf kecil, tokenisasi, penghilangan stopwords, dan stemming menggunakan pustaka NLTK dan Sastrawi. Dengan preprocessing yang tepat, data menjadi lebih bersih dan representatif sehingga model dapat belajar dengan lebih efektif dan menghasilkan prediksi yang akurat.

1) *Text cleaning*

   Pada penelitian ini, text cleaning dilakukan untuk memastikan data yang digunakan lebih konsisten dan mudah diproses oleh algoritma klasifikasi. Proses ini membantu mengurangi noise dan meningkatkan kualitas fitur yang dihasilkan, sehingga model dapat belajar pola dengan lebih baik.

	komentar	clean_text
8468	sangat baik dan memuaskan	sangat baik dan memuaskan
7944	Mohon aplikasinya diperbaikki lagi lebih baik ...	Mohon aplikasinya diperbaikki lagi lebih baik ...
48	tolong peta diberikan mode satelit seperti goo...	tolong peta diberikan mode satelit seperti goo...
8045	keren....amanah...sukses selalu	keren amanah sukses selalu
1534	mau login malah di suruh ngotak ngotakin gamba...	mau login malah di suruh ngotak ngotakin gamba...

Gambar 5. 1 Text cleaning

2) *Case folding*

proses mengubah seluruh teks menjadi huruf kecil (*lowercase*) untuk menyamakan format kata dan menghindari duplikasi kata yang sama dengan perbedaan kapitalisasi. Misalnya, kata "Maxim", "maxim", dan "MAXIM" akan dianggap sama setelah case folding.

	clean_text	case_folding
8468	sangat baik dan memuaskan	sangat baik dan memuaskan
7944	Mohon aplikasinya diperbaikki lagi lebih baik ...	mohon aplikasinya diperbaikki lagi lebih baik ...
48	tolong peta diberikan mode satelit seperti goo...	tolong peta diberikan mode satelit seperti goo...
8045	keren amanah sukses selalu	keren amanah sukses selalu
1534	mau login malah di suruh ngotak ngotakin gamba...	mau login malah di suruh ngotak ngotakin gamba...

Gambar 5. 2 case folding

3) Normalisasi Kata

proses mengubah kata-kata dalam teks menjadi bentuk baku sesuai standar bahasa Indonesia yang diatur dalam KBBI.

6662	driver sangat ramah dan amanah	driver sangat ramah dan amanah
1447	aplikasi rusak masa mau order aja gak bisa a...	aplikasi rusak masa mau order saja tidak bisa ...
1875	terima kasih maxim pelayanannya ramah dan tepa...	terima kasih maxim pelayanannya ramah dan tepa...
1456	sa vat baik pk sopirnya dia maunturun ambikan ...	sa vat baik pakai sopirnya dia maunturun ambik...
477	map y ga akurat nyari alamat titik tujuan di c...	map ya tidak akurat mencari alamat titik tujua...

Gambar 5. 3 Word Normalization

4) *Tokenizing*

Memecah teks menjadi unit-unit kecil yang disebut token, yang biasanya berupa kata, karakter, atau sub-kata. Token ini merupakan dasar dari analisis teks dalam *Natural Language Processing* (NLP) karena memungkinkan komputer untuk memahami dan memproses bahasa manusia secara lebih terstruktur. Contohnya, kalimat "Saya suka makan" akan dipecah menjadi token-token ["Saya", "suka", "makan"].

	normalisasi	tokenize
8468	sangat baik dan memuaskan	[sangat, baik, dan, memuaskan]
7944	mohon aplikasinya diperbaikki lagi lebih baik ...	[mohon, aplikasinya, diperbaikki, lagi, lebih,...
48	tolong peta diberikan mode satelit seperti goo...	[tolong, peta, diberikan, mode, satelit, seper...
8045	keren amanah sukses selalu	[keren, amanah, sukses, selalu]
1534	mau login malah di suruh ngotak ngotakin gamba...	[mau, login, malah, di, suruh, ngotak, ngotaki...

Gambar 5. 4 Tokenizing

5) *Stopword Removal*

proses menghapus kata-kata umum yang sering muncul dalam teks tetapi tidak memberikan makna penting dalam analisis, seperti kata sambung, kata depan, dan kata ganti (misalnya: "dan", "di", "yang").

	tokenize	no_stopword
8468	[sangat, baik, dan, memuaskan]	[memuaskan]
7944	[mohon, aplikasinya, diperbaikki, lagi, lebih,...	[mohon, aplikasinya, diperbaikki, kayak, alama...
48	[tolong, peta, diberikan, mode, satelit, seper...	[tolong, peta, mode, satelit, google, map, men...
8045	[keren, amanah, sukses, selalu]	[keren, amanah, sukses]
1534	[mau, login, malah, di, suruh, ngotak, ngotaki...	[login, suruh, ngotak, ngotakin, gambar]

Gambar 5. 5 Stopword Removal

6) *Stemming*

proses mengubah kata-kata yang memiliki berbagai bentuk turunan menjadi bentuk dasarnya (akar kata). Tujuannya adalah untuk menyatukan variasi kata yang memiliki makna sama agar model dapat mengenali kata tersebut secara konsisten. Misalnya, kata "berlari", "lari", dan "pelari" akan distem menjadi "lari".

	no_stopword	stem_text
8468	[memuaskan]	muas
7944	[mohon, aplikasinya, diperbaikki, kayak, alama...	mohon aplikasi diperbaikki kayak alamat tuju p...
48	[tolong, peta, mode, satelit, google, map, men...	tolong peta mode satelit google map cari lokas...
8045	[keren, amanah, sukses]	keren amanah sukses
1534	[login, suruh, ngotak, ngotakin, gambar]	login suruh ngotak ngotakin gambar

Gambar 5. 6 Stemming

5.1.2 Pembobotan Kata

Dalam tahap pembobotan kata, penelitian ini menggunakan pustaka *scikit-learn* khususnya modul *TfidfVectorizer* untuk mengubah data teks hasil preprocessing menjadi representasi numerik berupa vektor fitur. *TfidfVectorizer* secara otomatis menghitung nilai TF-IDF untuk setiap kata dalam dokumen, sehingga kata-kata yang memiliki bobot tinggi adalah kata yang sering muncul dalam dokumen tertentu namun jarang muncul di dokumen lain. Pustaka ini dipilih karena kemudahan penggunaannya, efisiensi komputasi, serta integrasinya yang baik dengan algoritma *machine learning* di *scikit-learn*. Dengan menggunakan TF-IDF dari *scikit-learn*, proses ekstraksi fitur menjadi lebih cepat dan hasilnya dapat langsung digunakan untuk pelatihan model klasifikasi seperti SVM dan *Logistic Regression*.

1) Term Frequency

TF digunakan untuk menilai pentingnya kata tersebut dalam konteks dokumen tertentu. Pada tabel yang ditampilkan, setiap baris mewakili sebuah kata (term), dan setiap kolom mewakili sebuah dokumen (D1, D2, ..., D10). Angka di dalam tabel menunjukkan jumlah kemunculan kata tersebut dalam dokumen terkait.

Sebagai contoh, kata "amanah" muncul 1 kali di dokumen D3, sedangkan kata "history" muncul 2 kali di dokumen D8. Nilai TF ini menjadi dasar dalam perhitungan TF-IDF, di mana kata-kata yang sering muncul dalam dokumen

tertentu akan memiliki bobot yang lebih tinggi, namun bobot ini akan dikurangi jika kata tersebut juga sering muncul di banyak dokumen lain.

	No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	1	alamat	0	1	0	0	0	0	0	0	0	0
1	2	amanah	0	0	0	1	0	0	0	0	0	0
2	3	aplikasi	0	1	0	0	0	0	1	1	0	0
3	4	bagus	0	0	0	0	0	1	0	0	1	0
4	5	bersih	0	0	0	0	0	0	0	0	0	1
5	6	cari	0	0	1	0	0	0	0	0	0	0
6	7	costumer	0	0	1	0	0	0	0	0	0	0
7	8	diperbaikki	0	1	0	0	0	0	0	0	0	0
8	9	gambar	0	0	0	0	1	0	0	0	0	0
9	10	google	0	0	1	0	0	0	0	0	0	0

Gambar 5. 7 *Term Frequency*

2) TF Normalisasi

penyesuaian nilai *Term Frequency* (TF) agar skala frekuensi kata dalam sebuah dokumen menjadi lebih proporsional dan tidak terlalu dipengaruhi oleh panjang dokumen atau kata yang sangat sering muncul. Salah satu metode normalisasi yang umum digunakan adalah dengan membagi frekuensi kemunculan kata dengan jumlah kata total dalam dokumen tersebut, sehingga nilai TF menjadi relatif terhadap ukuran dokumen. Dengan cara ini, kata-kata yang muncul sangat sering dalam dokumen panjang tidak mendominasi bobot fitur secara berlebihan. Contoh perhitungan TF Normalisasi dapat dilihat pada gambar di bawah ini.

	No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	1	alamat	0.000000	0.091000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	2	amanah	0.000000	0.000000	0.000000	0.333000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	3	aplikasi	0.000000	0.091000	0.000000	0.000000	0.000000	0.000000	0.071000	0.333000	0.000000	0.000000
3	4	bagus	0.000000	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000000	0.200000	0.000000
4	5	bersih	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.333000
5	6	cari	0.000000	0.000000	0.071000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	7	costumer	0.000000	0.000000	0.071000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	8	diperbaikki	0.000000	0.091000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
8	9	gambar	0.000000	0.000000	0.000000	0.000000	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000
9	10	google	0.000000	0.000000	0.071000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	11	harga	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.071000	0.000000	0.000000	0.000000

Gambar 5. 8 TF Normalisasi

3) IDF

Kata-kata yang muncul di banyak dokumen akan memiliki nilai IDF rendah, sedangkan kata-kata yang jarang muncul akan memiliki nilai IDF tinggi, sehingga memberikan bobot lebih pada kata yang lebih unik dan informatif. Nilai IDF dihitung berdasarkan jumlah dokumen yang mengandung kata tersebut (Document Frequency/DF) dan total dokumen dalam korpus. nilai dan perhitungannya dapat dilihat pada gambar di bawah ini, yang menunjukkan DF dan IDF untuk setiap kata dalam dataset.

	No	Terms	DF	IDF
0	1	alamat	1	1.704700
1	2	amanah	1	1.704700
2	3	aplikasi	3	1.011600
3	4	bagus	2	1.299300
4	5	bersih	1	1.704700
5	6	cari	1	1.704700
6	7	costumer	1	1.704700
7	8	diperbaikki	1	1.704700
8	9	gambar	1	1.704700
9	10	google	1	1.704700
10	11	harga	1	1.704700

Gambar 5. 9 DF dan IDF

4) TF-IDF

Dengan mengalikan TF dan IDF, TF-IDF memberikan bobot yang tinggi pada kata-kata yang sering muncul dalam dokumen tertentu tetapi jarang muncul di dokumen lain, sehingga kata-kata tersebut dianggap lebih informatif. Pada gambar di bawah, setiap nilai menunjukkan bobot TF-IDF untuk kata tertentu dalam dokumen tertentu. Nilai yang lebih tinggi (ditandai dengan warna kuning, hijau, atau biru) menunjukkan kata yang lebih penting dalam dokumen tersebut. Misalnya, kata "amanah" memiliki bobot TF-IDF tinggi di dokumen D4, menandakan kata tersebut sangat relevan untuk dokumen tersebut.

	No	Terms	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	1	alamat	0.000000	0.155000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	2	amanah	0.000000	0.000000	0.000000	0.568000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	3	aplikasi	0.000000	0.092000	0.000000	0.000000	0.000000	0.000000	0.072000	0.337000	0.000000	0.000000
3	4	bagus	0.000000	0.000000	0.000000	0.000000	0.000000	0.650000	0.000000	0.000000	0.260000	0.000000
4	5	bersih	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.568000
5	6	cari	0.000000	0.000000	0.122000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	7	costumer	0.000000	0.000000	0.122000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	8	diperbaikki	0.000000	0.155000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
8	9	gambar	0.000000	0.000000	0.000000	0.000000	0.341000	0.000000	0.000000	0.000000	0.000000	0.000000
9	10	google	0.000000	0.000000	0.122000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	11	harga	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.122000	0.000000	0.000000	0.000000

Gambar 5. 10 Perhitungan TF-IDF

5.1.3 Klasifikasi SVM dan Logistic Regression

Pada tahap klasifikasi, penelitian ini menggunakan algoritma Support Vector Machine (SVM) dan Logistic Regression yang diimplementasikan dengan pustaka *scikit-learn* pada bahasa pemrograman *Python*. Kedua model ini dipilih karena keefektifannya dalam menangani data teks yang telah diolah menggunakan TF-IDF. Selanjutnya, hasil evaluasi performa kedua model akan ditampilkan dan dianalisis untuk menentukan model yang paling sesuai dalam mengklasifikasikan sentimen ulasan aplikasi Maxim.

```
                precision    recall  f1-score   support

      negatif       0.92      0.98      0.95      1201
      positif       0.98      0.92      0.95      1236

     accuracy                           0.95      2437
    macro avg       0.95      0.95      0.95      2437
weighted avg       0.95      0.95      0.95      2437

Akurasi SVM: 94.71%
=== Logistic Regression Classification Report ===
                precision    recall  f1-score   support

      negatif       0.90      0.95      0.92      1201
      positif       0.94      0.90      0.92      1236

     accuracy                           0.92      2437
    macro avg       0.92      0.92      0.92      2437
weighted avg       0.92      0.92      0.92      2437

Akurasi Logistic Regression: 92.04%
```

Gambar 5. 11 Hasil Laporan Klasifikasi

Hasil *classification report* menunjukkan bahwa model *Support Vector Machine* (SVM) memiliki performa lebih baik dengan akurasi 94,71%, precision dan recall yang lebih tinggi pada kedua kelas dibandingkan Logistic Regression yang memiliki akurasi 92,04% SVM mampu mengenali sentimen positif dan negatif dengan lebih akurat, ditunjukkan oleh nilai F1-score 0,95, sementara Logistic Regression mencapai 0,92. Dengan demikian, SVM lebih efektif dan direkomendasikan sebagai model utama dalam klasifikasi sentimen ulasan aplikasi Maxim.

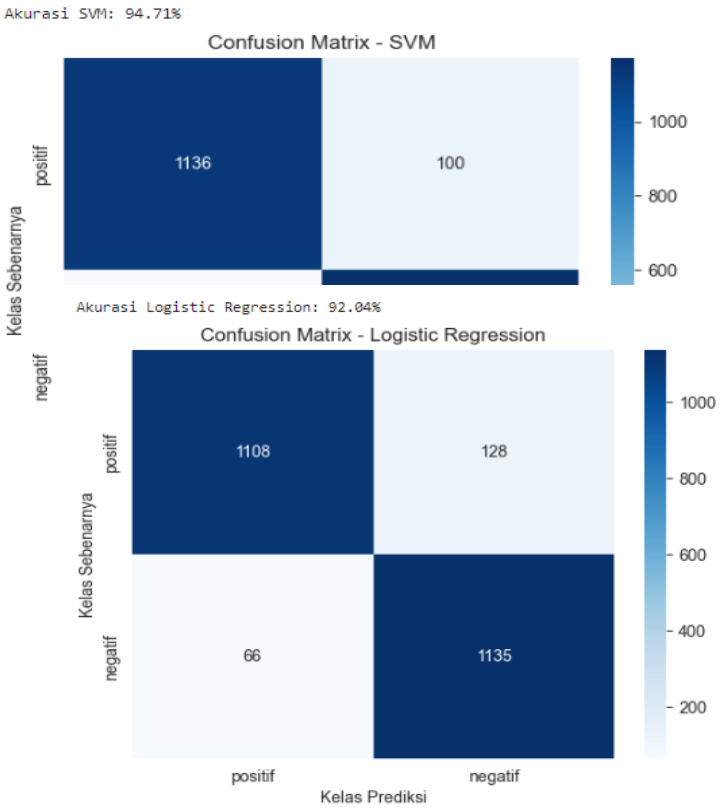
5.1.4 Evaluasi

Evaluasi performa model klasifikasi dilakukan dengan menggunakan beberapa metrik dan visualisasi penting, yaitu *confusion matrix*, ROC AUC, dan *cross-validation*. Metode ini bertujuan untuk mengukur akurasi, kemampuan membedakan kelas, serta kestabilan model pada data yang berbeda. Evaluasi

dilakukan pada dua model utama, yaitu *Support Vector Machine* (SVM) dan Logistic Regression, untuk menentukan model yang paling efektif dalam mengklasifikasikan sentimen ulasan aplikasi Maxim.

1) *Confusion Matrix*

Gambar 5. 12 Confusion matrix SVM



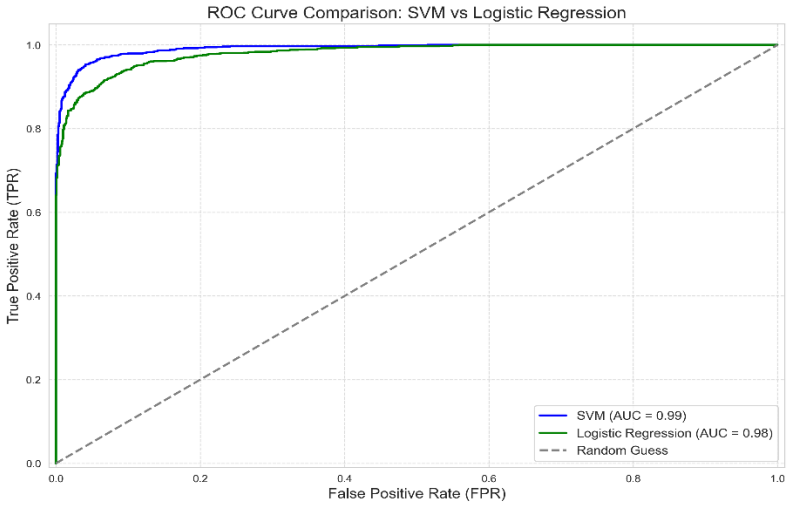
untuk model SVM menunjukkan bahwa model berhasil mengklasifikasikan dengan benar 1136 data positif dan 1172 data negatif. Kesalahan klasifikasi terjadi pada 100 data positif yang diprediksi negatif dan 29 data negatif yang diprediksi positif. Hal ini menandakan SVM memiliki tingkat kesalahan yang rendah dan kemampuan klasifikasi yang baik.

Gambar 5. 13 confusion matrix LR

Sedangkan untuk *Logistic Regression* menunjukkan 1108 data positif dan 1135 data negatif yang diklasifikasikan dengan benar. Kesalahan klasifikasi lebih tinggi dibanding SVM, yaitu 128 data positif diprediksi negatif dan 66 data negatif diprediksi positif. Ini mengindikasikan *Logistic Regression* memiliki

performa yang sedikit lebih rendah dibanding SVM dalam hal akurasi klasifikasi.

2) *ROC AUC Curve*



Gambar 5. 14 Evaluasi ROC AUC Curve

Kurva ROC memperlihatkan perbandingan performa kedua model dalam membedakan kelas positif dan negatif. Model SVM memiliki nilai AUC sebesar 0,99, yang menunjukkan kemampuan hampir sempurna dalam klasifikasi. *Logistic Regression* juga menunjukkan performa baik dengan AUC 0,98, namun sedikit di bawah SVM. Garis putus-putus menunjukkan baseline random guess yang jelas kalah dibanding kedua model.

3) *Cross Validation*

```
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['94.79%', '95.53%', '95.03%', '91.13%', '87.64%']
Rata-rata akurasi: 92.83%

=== Cross-Validation Score (Logistic Regression) ===
Akurasi per fold: ['91.34%', '94.62%', '94.29%', '86.49%', '80.95%']
Rata-rata akurasi: 89.54%
```

Gambar 5. 15 evaluasi cross-validation

Hasil *cross-validatio*n menunjukkan kestabilan performa model pada data yang berbeda. SVM memperoleh rata-rata akurasi sebesar 92,83% dengan variasi akurasi per fold yang relatif kecil, menandakan model ini konsisten dan andal. *Logistic Regression* memiliki rata-rata akurasi 89,54%, yang lebih rendah dan menunjukkan performa yang kurang stabil dibanding SVM.

Secara keseluruhan, evaluasi ini menegaskan bahwa model SVM lebih unggul dalam hal akurasi, kemampuan membedakan kelas, dan kestabilan performa dibandingkan *Logistic Regression* dalam klasifikasi sentimen ulasan aplikasi Maxim.

5.1.5 Uji Coba Data Kedua Metode SVM & LR

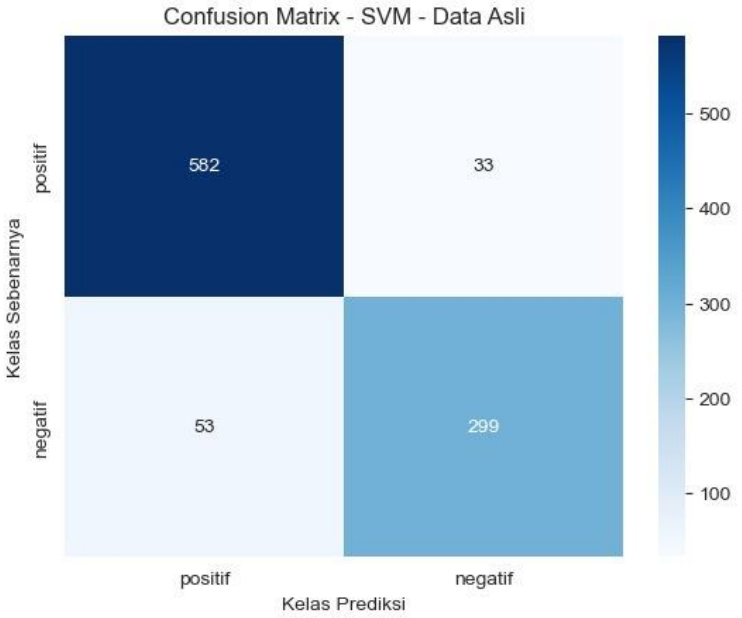
Uji coba data dilakukan data asli dan data balancing melalui proses klasifikasi, *confusion matrix* dan *cross-validation* sebanyak 5 kali uji coba.

1. Uji Coba Data Asli.

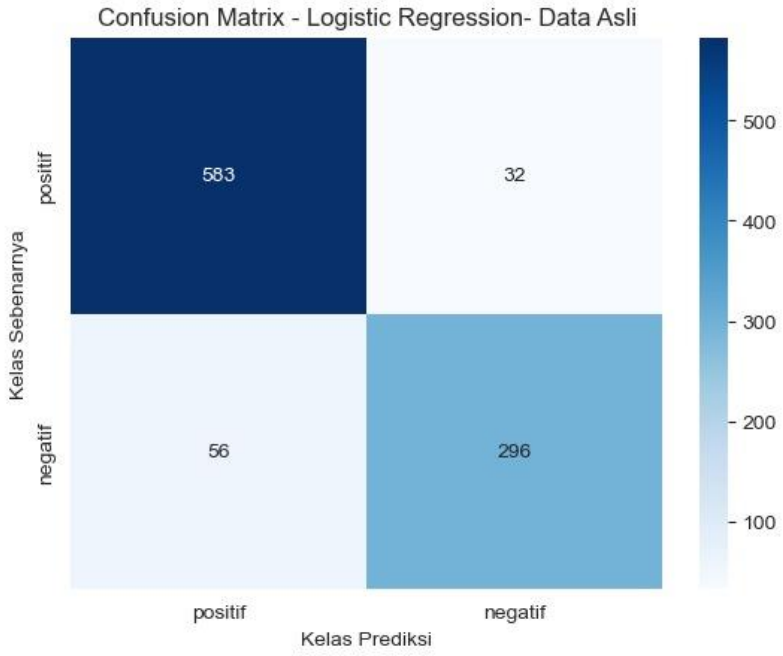
   Data yang digunkan 90% Data latih Dan 10% Data uji.

```
=== Menggunakan Data Asli ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.91      0.89      0.90       368
     positif       0.93      0.94      0.94       599

    accuracy                           0.92       967
   macro avg       0.92      0.92      0.92       967
weighted avg       0.92      0.92      0.92       967

Akurasi SVM: 92.24%
=== Menggunakan Data Asli ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.89      0.86      0.88       368
     positif       0.91      0.94      0.93       599

    accuracy                           0.91       967
   macro avg       0.90      0.90      0.90       967
weighted avg       0.91      0.91      0.91       967

Akurasi Logistic Regression: 90.69%
```

Gambar 5. 16 Hasil Klasifikasi



Gambar 5. 17 Hasil Confusion Matrix SVM



Gambar 5. 18 Hasil Confusion Matrix LR

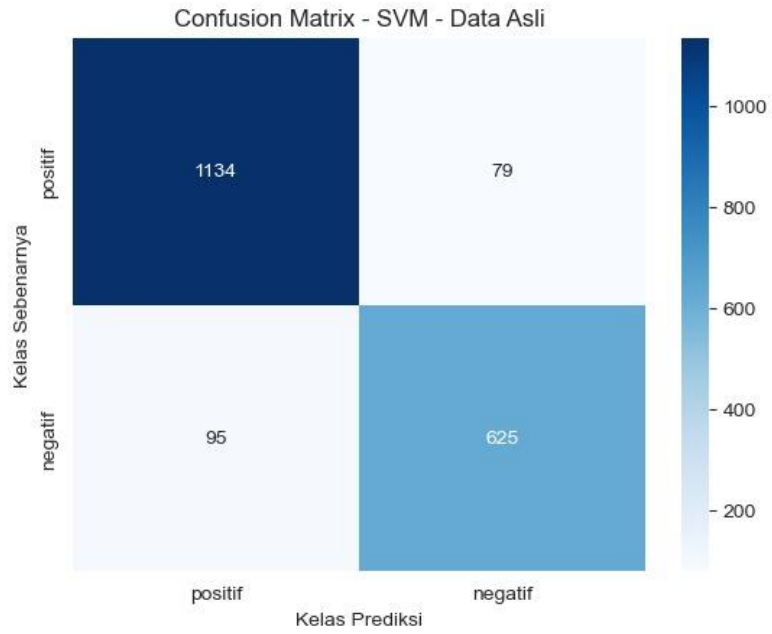
```
=== Menggunakan Data Asli ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['89.34%', '92.76%', '92.24%', '87.94%', '87.78%']
Rata-rata akurasi: 90.01%

=== Cross-Validation Score (Logistic Regression) ===
=== Mengunakan Data Asli ===
Akurasi per fold: ['85.93%', '89.76%', '91.36%', '86.02%', '85.30%']
Rata-rata akurasi: 87.67%
```

Gambar 5. 19 Hasil Cross-Validation

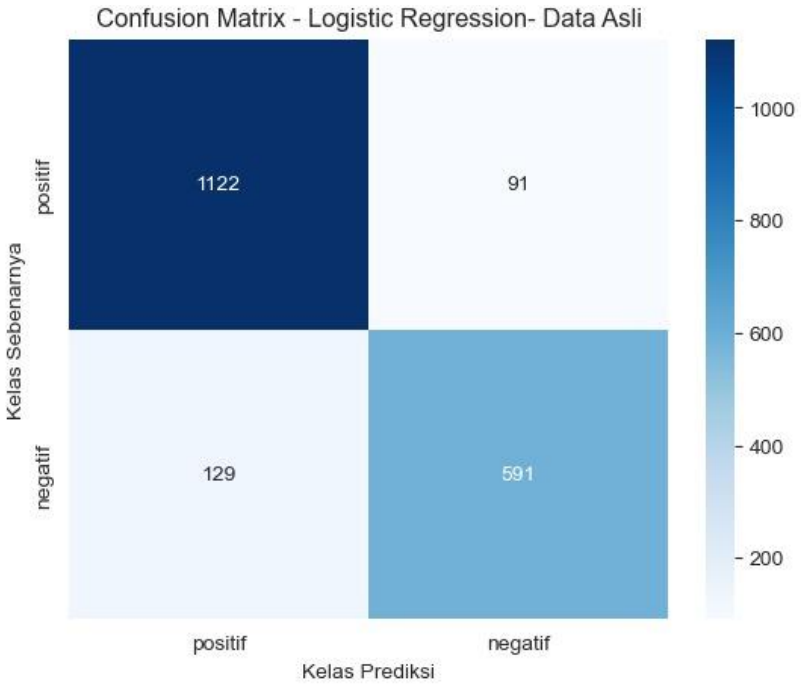
Data yang digunkan 80% Data latih Dan 20% Data uji.

```
=== Menggunakan Data Asli ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.89      0.87      0.88       720
     positif       0.92      0.93      0.93      1213

    accuracy                           0.91      1933
   macro avg       0.91      0.90      0.90      1933
weighted avg       0.91      0.91      0.91      1933

Akurasi SVM: 91.00%
=== Menggunakan Data Asli ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.87      0.82      0.84       720
     positif       0.90      0.92      0.91      1213

    accuracy                           0.89      1933
   macro avg       0.88      0.87      0.88      1933
weighted avg       0.89      0.89      0.89      1933

Akurasi Logistic Regression: 88.62%
```

Gambar 5. 20 Hasil Klasifikasi



Gambar 5. 22 Hasil Confusion Matrix SVM



Gambar 5. 21 Hasil Confusion Matrix LR

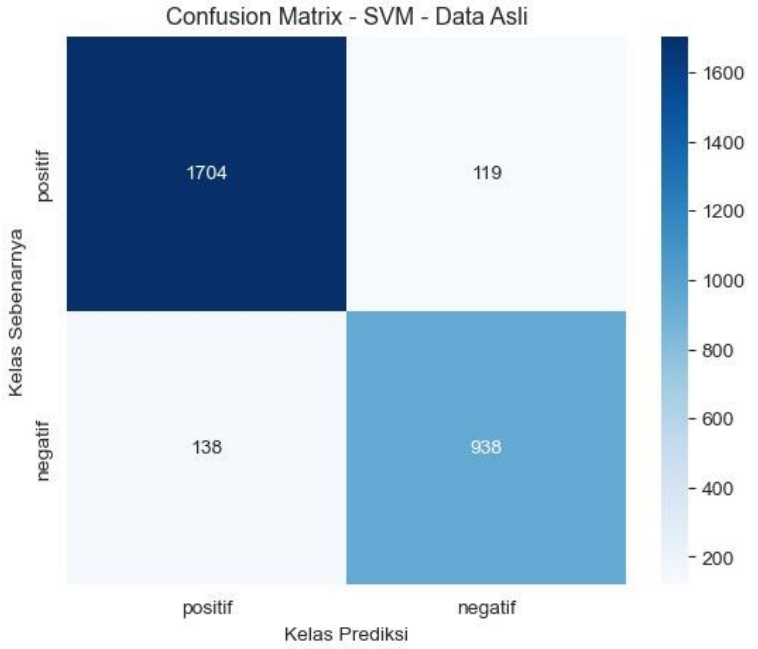
```
=== Menggunakan Data Asli ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['89.50%', '92.55%', '92.03%', '87.89%', '87.47%']
Rata-rata akurasi: 89.89%

=== Cross-Validation Score (Logistic Regression) ===
=== Mengunakan Data Asli ===
Akurasi per fold: ['86.03%', '89.81%', '91.21%', '86.08%', '85.30%']
Rata-rata akurasi: 87.68%
```

Gambar 5. 23 Hasil Cross-Validation

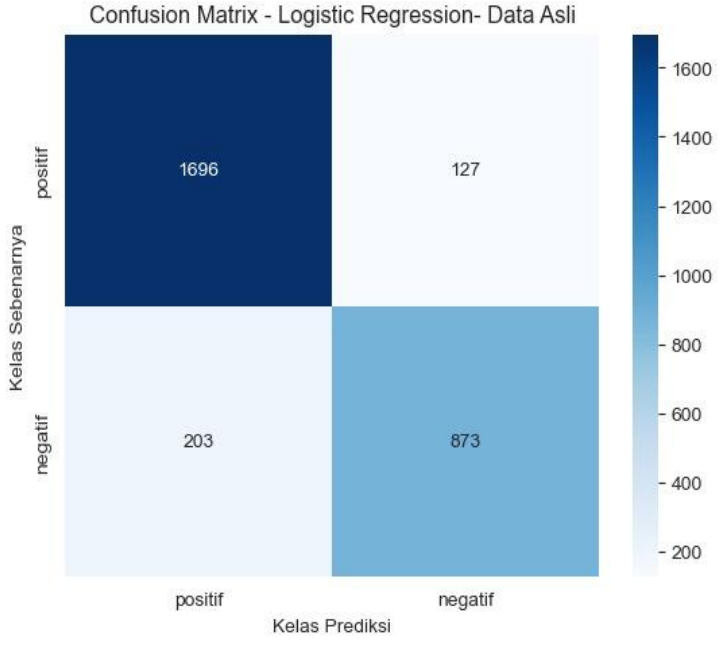
Data yang digunkan 70% Data latih Dan 30% Data uji.

```
=== Menggunakan Data Asli ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.89      0.87      0.88      1076
     positif       0.93      0.93      0.93      1823

    accuracy                           0.91      2899
   macro avg       0.91      0.90      0.90      2899
weighted avg       0.91      0.91      0.91      2899

Akurasi SVM: 91.13%
=== Menggunakan Data Asli ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.87      0.81      0.84      1076
     positif       0.89      0.93      0.91      1823

    accuracy                           0.89      2899
   macro avg       0.88      0.87      0.88      2899
weighted avg       0.89      0.89      0.89      2899

Akurasi Logistic Regression: 88.62%
```

Gambar 5. 24 Hasil Klasifikasi



Gambar 5. 25 Hasil Confusion Matrix SVM



Gambar 5. 26 Hasil Confusion Matrix LR

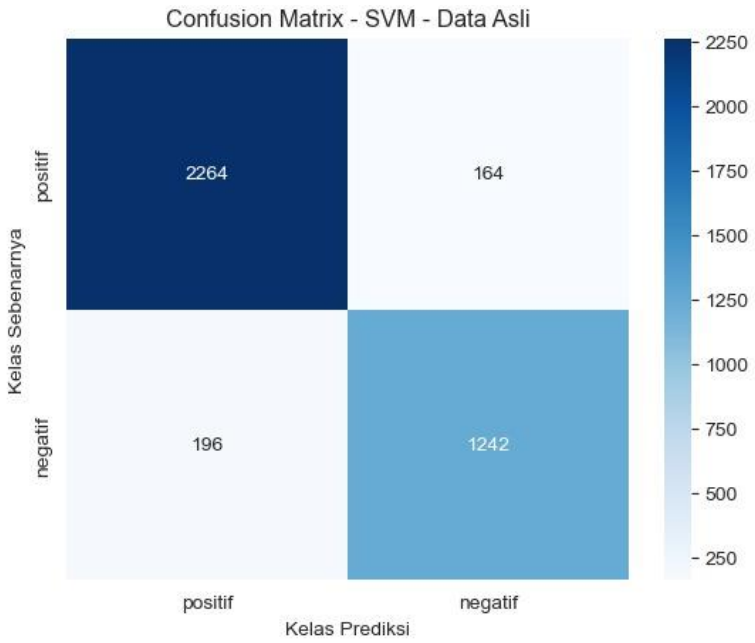
```
=== Menggunakan Data Asli ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['89.50%', '92.55%', '91.98%', '88.04%', '87.73%']
Rata-rata akurasi: 89.96%

=== Cross-Validation Score (Logistic Regression) ===
=== Mengunakan Data Asli ===
Akurasi per fold: ['86.19%', '89.91%', '91.15%', '86.23%', '85.30%']
Rata-rata akurasi: 87.76%
```

Gambar 5. 27 Hasil Cross-Validation

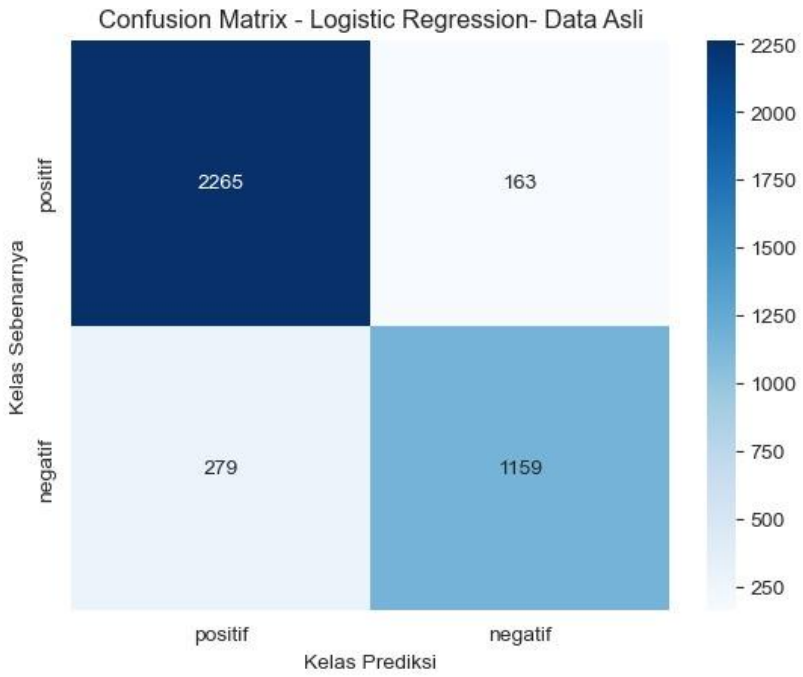
Data yang digunkan 60% Data latih Dan 40% Data uji.

```
=== Menggunakan Data Asli ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.88      0.86      0.87      1438
     positif       0.92      0.93      0.93      2428

    accuracy                           0.91      3866
   macro avg       0.90      0.90      0.90      3866
weighted avg       0.91      0.91      0.91      3866

Akurasi SVM: 90.69%
=== Menggunakan Data Asli ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.88      0.81      0.84      1438
     positif       0.89      0.93      0.91      2428

    accuracy                           0.89      3866
   macro avg       0.88      0.87      0.88      3866
weighted avg       0.89      0.89      0.88      3866

Akurasi Logistic Regression: 88.57%
```

Gambar 5. 28 Hasil Klasifikasi



Gambar 5. 29 Hasil Confusion Matrix SVM



Gambar 5. 30 Hasil Confusion Matrix LR

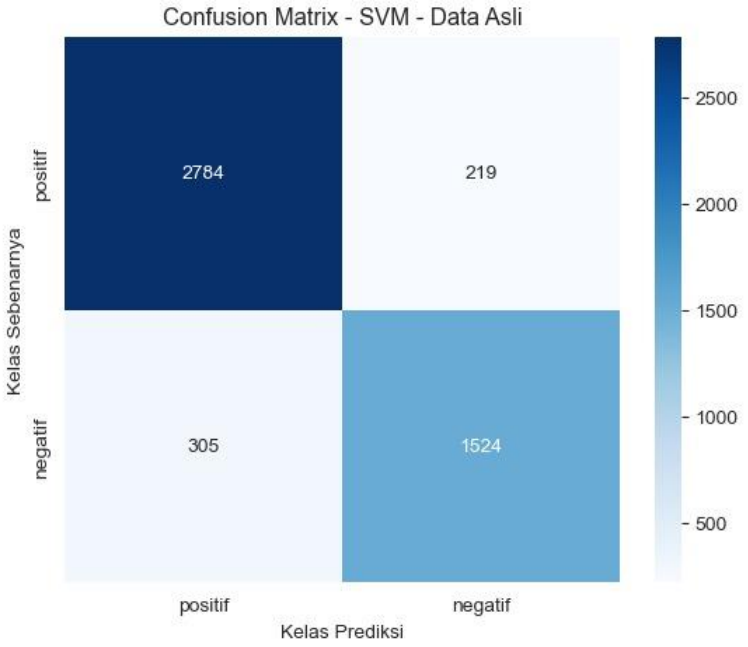
```
=== Menggunakan Data Asli ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['89.81%', '92.96%', '92.29%', '88.30%', '87.47%']
Rata-rata akurasi: 90.17%

=== Cross-Validation Score (Logistic Regression) ===
=== Mengunakan Data Asli ===
Akurasi per fold: ['86.24%', '89.96%', '91.36%', '85.87%', '85.51%']
Rata-rata akurasi: 87.79%
```

Gambar 5. 31 Hasil Cross-Validation

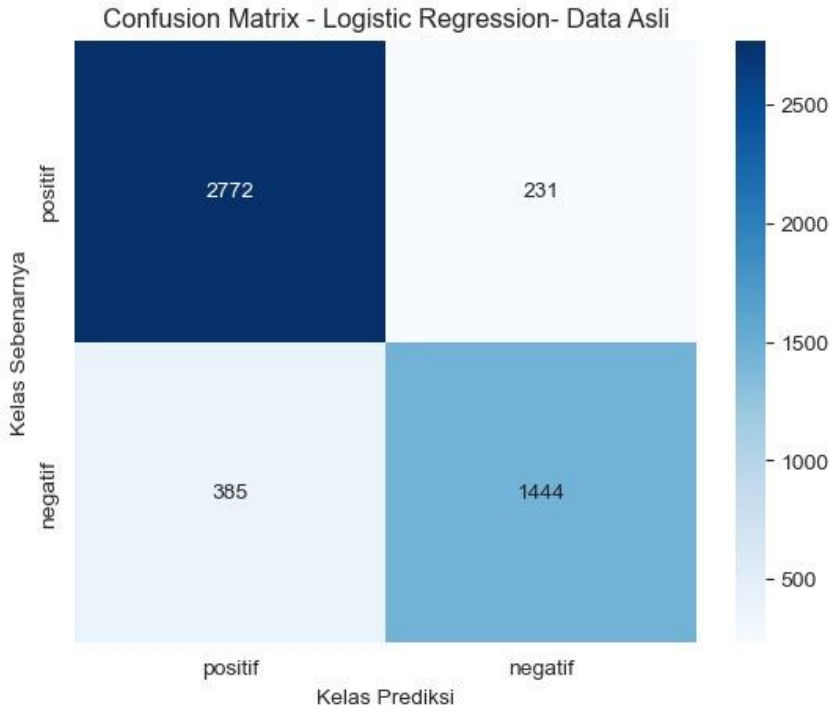
Data yang digunkan 50% Data latih Dan 50% Data uji.

```
=== Menggunakan Data Asli ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.87      0.83      0.85      1829
     positif       0.90      0.93      0.91      3003

    accuracy                           0.89      4832
   macro avg       0.89      0.88      0.88      4832
weighted avg       0.89      0.89      0.89      4832

Akurasi SVM: 89.16%
=== Menggunakan Data Asli ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.86      0.79      0.82      1829
     positif       0.88      0.92      0.90      3003

    accuracy                           0.87      4832
   macro avg       0.87      0.86      0.86      4832
weighted avg       0.87      0.87      0.87      4832

Akurasi Logistic Regression: 87.25%
```

Gambar 5. 32 Hasil Klasifikasi



Gambar 5. 33 Hasil Confusion Matrix SVM



Gambar 5. 34 Hasil Confusion Matrix LR

```
=== Data sudah Balancing ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['89.55%', '92.60%', '91.77%', '87.63%', '87.68%']
Rata-rata akurasi: 89.85%
=== Cross-Validation Score (Logistic Regression) ===
=== Data sudah Balancing ===
Akurasi per fold: ['86.19%', '90.17%', '91.05%', '85.97%', '85.30%']
Rata-rata akurasi: 87.74%
```

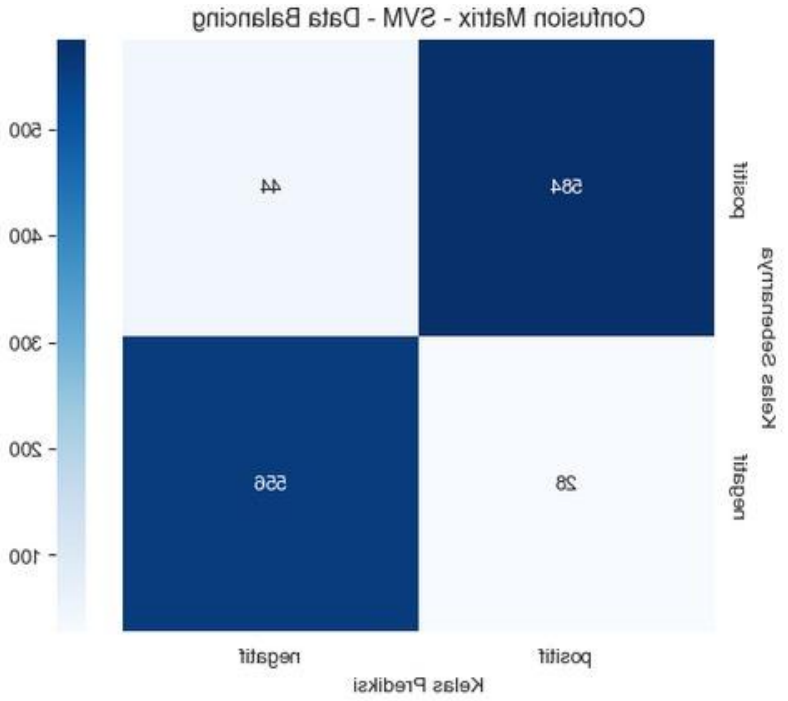
Gambar 5. 35 Hasil Cross-Validation

2. Uji Coba Data Balancing.

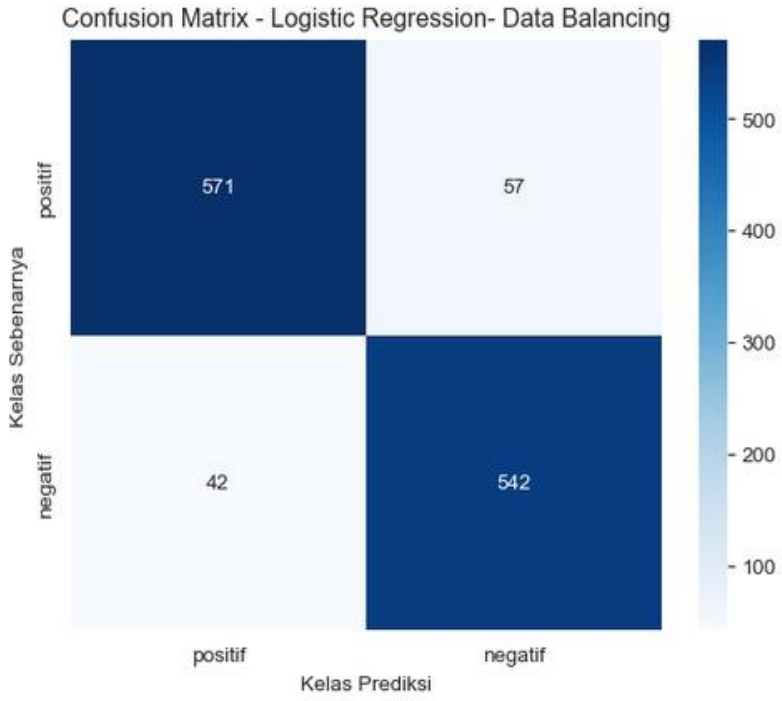
   Data yang digunkan 90% Data latih Dan 10% Data uji.

```
===  Data sudah Balancing ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.93      0.95      0.94       584
     positif       0.95      0.93      0.94       628

    accuracy                           0.94      1212
   macro avg       0.94      0.94      0.94      1212
weighted avg       0.94      0.94      0.94      1212

Akurasi SVM: 94.06%
=== Data sudah Balancing  ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.90      0.93      0.92       584
     positif       0.93      0.91      0.92       628

    accuracy                           0.92      1212
   macro avg       0.92      0.92      0.92      1212
weighted avg       0.92      0.92      0.92      1212

Akurasi Logistic Regression: 91.83%
```

Gambar 5. 36 Hasil Klasifikasi



Gambar 5. 37 Hasil Confusion Matrix SVM



Gambar 5. 38 Hasil Confusion Matrix LR

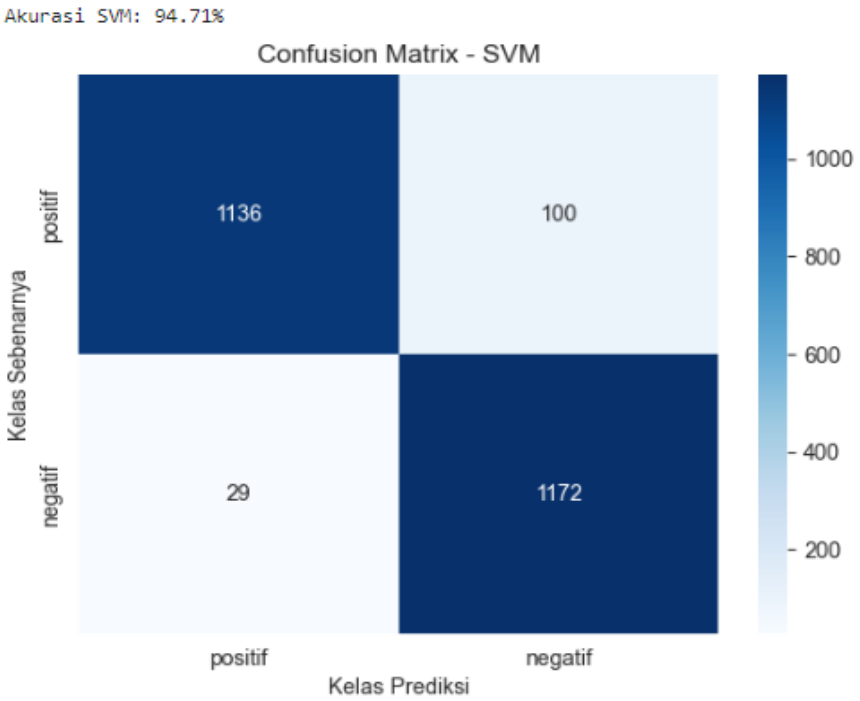
```
=== Data sudah Balancing ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['93.69%', '94.18%', '94.18%', '90.26%', '88.81%']
Rata-rata akurasi: 92.22%
=== Cross-Validation Score (Logistic Regression) ===
=== Data sudah Balancing ===
Akurasi per fold: ['90.43%', '94.14%', '93.64%', '85.38%', '83.36%']
Rata-rata akurasi: 89.39%
```

Gambar 5. 39 Hasil Cross-Validation

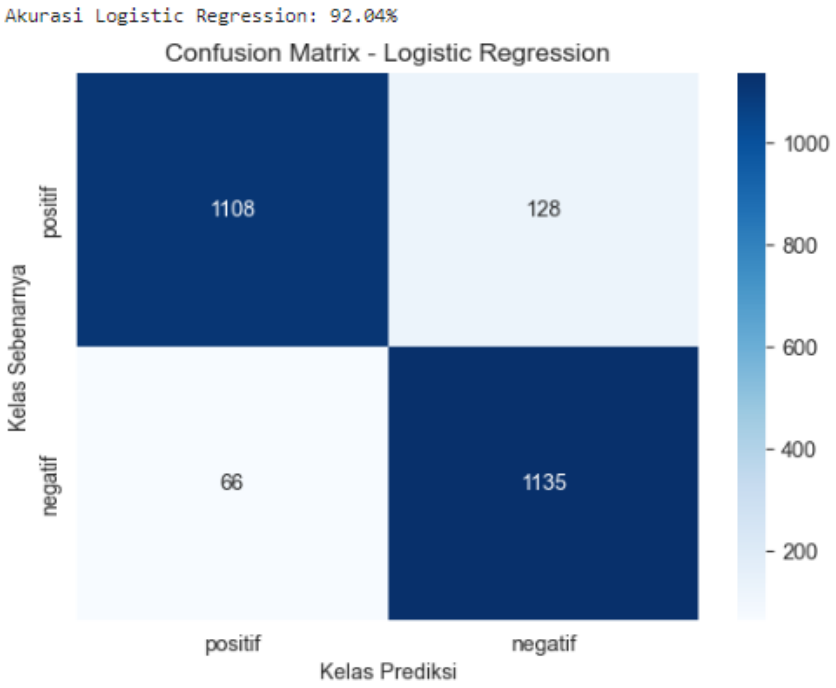
Data yang digunkan 80% Data latih Dan 20% Data uji.

```
              precision    recall  f1-score   support

     negatif       0.92      0.98      0.95      1201
     positif       0.98      0.92      0.95      1236

    accuracy                           0.95      2437
   macro avg       0.95      0.95      0.95      2437
weighted avg       0.95      0.95      0.95      2437

Akurasi SVM: 94.71%
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.90      0.95      0.92      1201
     positif       0.94      0.90      0.92      1236

    accuracy                           0.92      2437
   macro avg       0.92      0.92      0.92      2437
weighted avg       0.92      0.92      0.92      2437

Akurasi Logistic Regression: 92.04%
```

Gambar 5. 40 Hasil Klasifikasi



Gambar 5. 41 Hasil Confusion Matrix SVM



Gambar 5. 42 Hasil Confusion Matrix LR

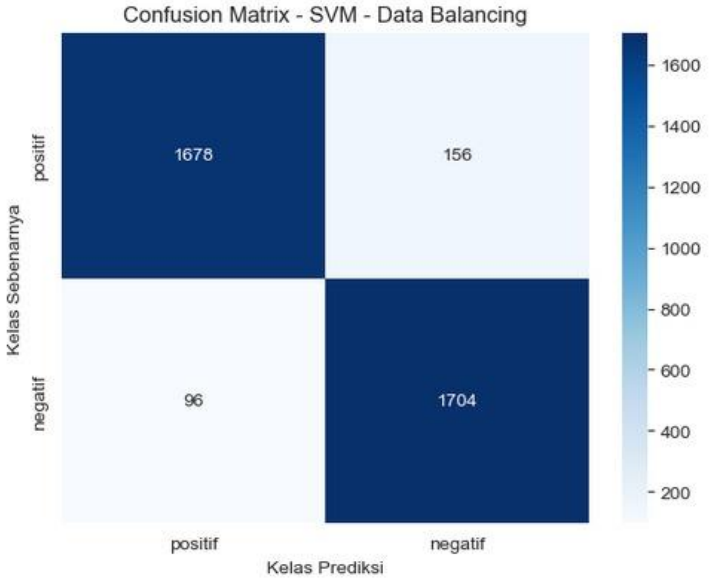
```
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['94.79%', '95.53%', '95.03%', '91.13%', '87.64%']
Rata-rata akurasi: 92.83%

=== Cross-Validation Score (Logistic Regression) ===
Akurasi per fold: ['91.34%', '94.62%', '94.29%', '86.49%', '80.95%']
Rata-rata akurasi: 89.54%
```

Gambar 5. 43 Hasil Cross-Validation

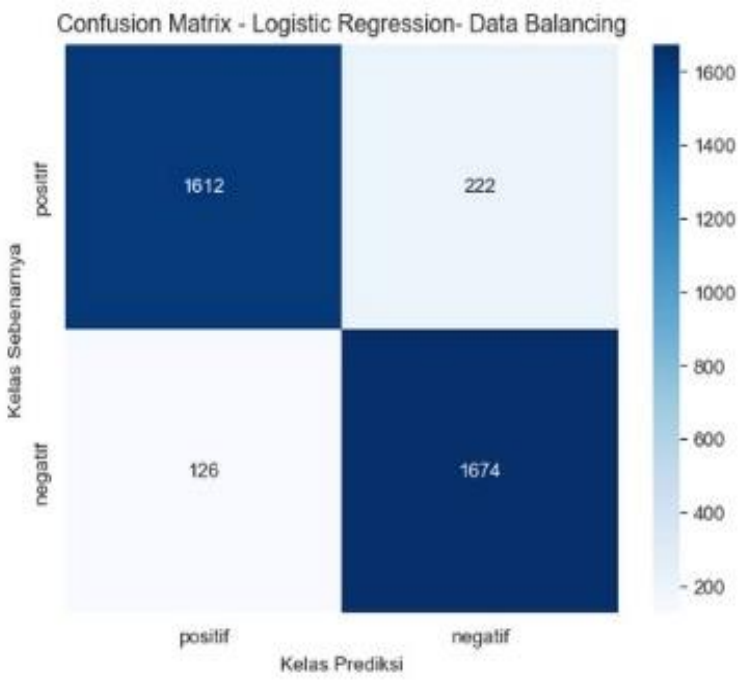
Data yang digunkan 70% Data latih Dan 30% Data uji.

```
===  Data sudah Balancing ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.92      0.95      0.93      1800
     positif       0.95      0.91      0.93      1834

    accuracy                           0.93      3634
   macro avg       0.93      0.93      0.93      3634
weighted avg       0.93      0.93      0.93      3634

Akurasi SVM: 93.07%
=== Data sudah Balancing  ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.88      0.93      0.91      1800
     positif       0.93      0.88      0.90      1834

    accuracy                           0.90      3634
   macro avg       0.91      0.90      0.90      3634
weighted avg       0.91      0.90      0.90      3634

Akurasi Logistic Regression: 90.42%
```

Gambar 5. 44 Hasil Klasifikasi



Gambar 5. 45 Hasil Confusion Matrix SVM



Gambar 5. 46 Hasil Confusion Matrix LR

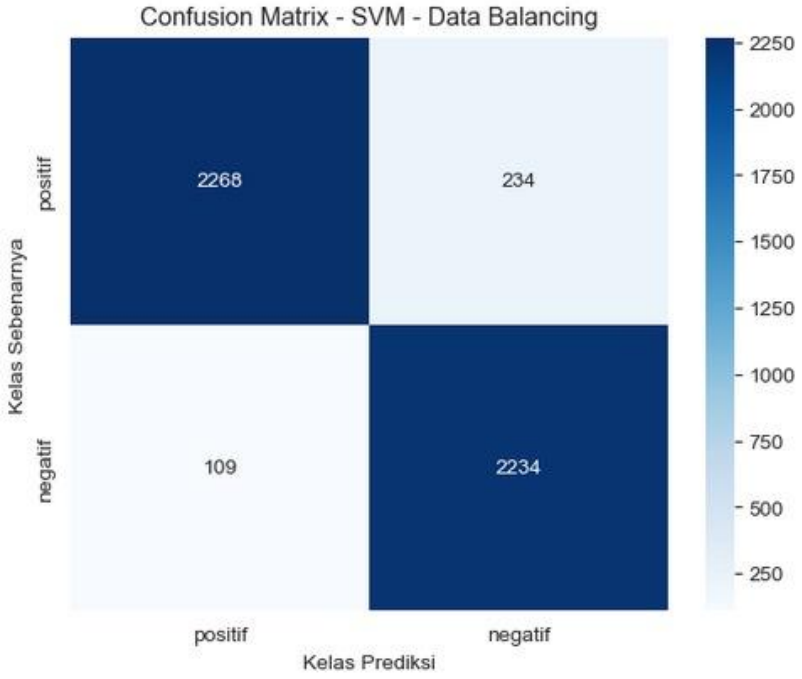
```
=== Data sudah Balancing ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['93.73%', '94.51%', '94.05%', '90.42%', '88.77%']
Rata-rata akurasi: 92.30%
=== Cross-Validation Score (Logistic Regression) ===
=== Data sudah Balancing ===
Akurasi per fold: ['90.51%', '94.02%', '93.68%', '85.71%', '83.20%']
Rata-rata akurasi: 89.42%
```

Gambar 5. 47 Hasil Cross-Validation

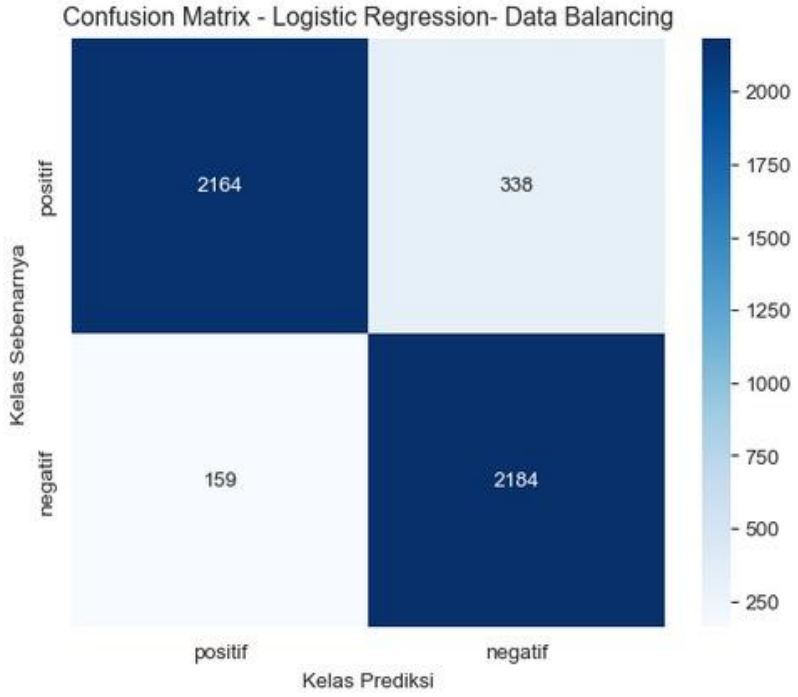
Data yang digunkan 60% Data latih Dan 40% Data uji.

```
===  Data sudah Balancing ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.91      0.95      0.93      2343
     positif       0.95      0.91      0.93      2502

    accuracy                           0.93      4845
   macro avg       0.93      0.93      0.93      4845
weighted avg       0.93      0.93      0.93      4845

Akurasi SVM: 92.92%
=== Data sudah Balancing  ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.87      0.93      0.90      2343
     positif       0.93      0.86      0.90      2502

    accuracy                           0.90      4845
   macro avg       0.90      0.90      0.90      4845
weighted avg       0.90      0.90      0.90      4845

Akurasi Logistic Regression: 89.74%
```

Gambar 5. 48 Hasil Klasifikasi



Gambar 5. 49 Hasil Confusion Matrix SVM



Gambar 5. 50 Hasil Confusion Matrix LR

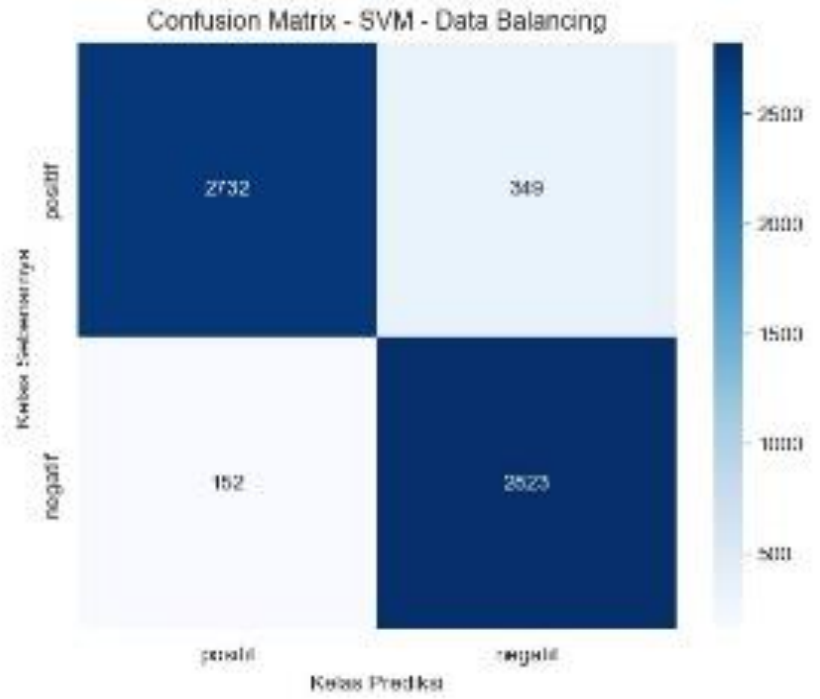
```
=== Data sudah Balancing ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['93.73%', '94.26%', '93.97%', '90.21%', '88.52%']
Rata-rata akurasi: 92.14%
=== Cross-Validation Score (Logistic Regression) ===
=== Data sudah Balancing ===
Akurasi per fold: ['90.84%', '94.06%', '93.64%', '85.38%', '83.28%']
Rata-rata akurasi: 89.44%
```

Gambar 5. 51 Hasil Cross-Validation

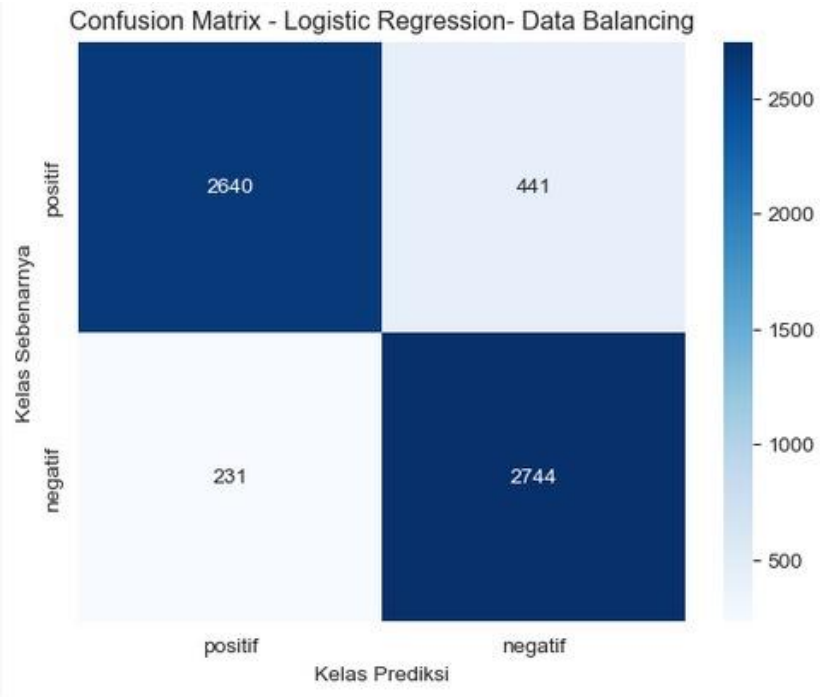
Data yang digunkan 50% Data latih Dan 50% Data uji.

```
===  Data sudah Balancing ===
=== SVM Classification Report ===
              precision    recall  f1-score   support

     negatif       0.89      0.95      0.92      2975
     positif       0.95      0.89      0.92      3081

    accuracy                           0.92      6056
   macro avg       0.92      0.92      0.92      6056
weighted avg       0.92      0.92      0.92      6056

Akurasi SVM: 91.73%
=== Data sudah Balancing  ===
=== Logistic Regression Classification Report ===
              precision    recall  f1-score   support

     negatif       0.86      0.92      0.89      2975
     positif       0.92      0.86      0.89      3081

    accuracy                           0.89      6056
   macro avg       0.89      0.89      0.89      6056
weighted avg       0.89      0.89      0.89      6056

Akurasi Logistic Regression: 88.90%
```

Gambar 5. 52 Hasil Klasifikasi



Gambar 5. 53 Hasil Confusion Matrix SVM



Gambar 5. 54 Hasil Confusion Matrix LR

```
=== Data sudah Balancing ===
=== Cross-Validation Score (SVM) ===
Akurasi per fold: ['94.02%', '94.30%', '94.01%', '90.38%', '88.60%']
Rata-rata akurasi: 92.26%
=== Cross-Validation Score (Logistic Regression) ===
=== Data sudah Balancing ===
Akurasi per fold: ['90.84%', '94.14%', '93.52%', '85.71%', '83.44%']
Rata-rata akurasi: 89.53%
```

Gambar 5. 55 Hasil Cross-Validation

Tabel 5. 1 Hasil Uji Data Asli

Uji data asli	Klasifikasi SVM	Klasifikasi LR	Cross Validation SVM	Cross Validation LR
90% data latih 10% data uji	92.24%	90.69%	90.01%	87.67%
80% data latih 20% data uji	91.00%	88.62%	89.89%	87.68%
70% data latih 30% data uji	91.13%	88.62%	92.35%	89.50%
60% data latih 40% data uji	90.69%	88.57%	90.17%	87.79%
50% data latih 50% data uji	89.16%	87.25%	89.85%	87.74%

Tabel 5. 2 Hasil Uji Balancing Data

Uji data balancing	Klasifikasi SVM	Klasifikasi LR	Cross Validation SVM	Cross Validation LR
90% data latih 10% data uji	94.06%	91.83%	92.22%	89.39%
80% data latih 20% data uji	94.71%	92.04%	92.83%	89.54%
70% data latih 30% data uji	93.07%	90.42%	92.30%	89.42%
60% data latih 40% data uji	92.92%	89.74%	92.14%	89.44%
50% data latih 50% data uji	91.73%	88.90%	92.26%	89.53%

Setelah melakukan uji coba data asli dengan data balancing menggunakan metode SVM dan LR sebanyak 5 kali. Secara keseluruhan, bahwa metode SVM lebih unggul dibandingkan dengan metode LR.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai jawaban dari tujuan penelitian ini:

1. Berdasarkan analisis ulasan pengguna yang diambil dari Google Play Store, dapat disimpulkan bahwa mayoritas pengguna memberikan persepsi positif terhadap layanan aplikasi Maxim. Hal ini terlihat dari dominasi kata-kata positif seperti "driver", "ramah", "cepat", dan "aplikasi" yang sering muncul dalam ulasan. Kata-kata tersebut mencerminkan kepuasan pengguna terhadap kualitas layanan, kecepatan respon, dan kemudahan penggunaan aplikasi. Namun, terdapat juga beberapa ulasan negatif yang menyoroti masalah seperti keterlambatan dan kendala teknis, sehingga gambaran sentimen pengguna cukup beragam namun cenderung positif secara keseluruhan.
2. Dalam mengklasifikasikan sentimen ulasan pengguna, kedua metode, yaitu *Support Vector Machine*dan *Logistic Regression*, berhasil memberikan hasil yang memuaskan. Namun, SVM menunjukkan performa yang lebih unggul dengan nilai akurasi mencapai 94,71%, presisi, recall, dan F1-score yang lebih tinggi dibandingkan *Logistic Regression* yang memiliki akurasi 92,04%. Hal ini menunjukkan bahwa SVM lebih efektif dalam membedakan sentimen positif dan negatif pada dataset ulasan Maxim. Keunggulan SVM ini juga didukung oleh hasil evaluasi lain seperti ROC AUC *dan cross-validation* yang menunjukkan kestabilan dan konsistensi performa model.

6.2 Saran

Berdasarkan kelemahan dan keterbatasan yang ditemukan selama penelitian, beberapa saran yang dapat diberikan untuk penelitian selanjutnya adalah:

1. Perluasan dataset dengan menambahkan ulasan dari berbagai platform lain selain Google Play Store agar hasil analisis lebih representatif dan komprehensif.

2. Eksplorasi metode klasifikasi lain yang lebih kompleks seperti *deep learning* (misalnya LSTM atau BERT) untuk meningkatkan akurasi dan kemampuan model dalam memahami konteks kalimat.
3. Penambahan analisis aspek sentimen (*aspect-based sentiment analysis*) untuk mendapatkan insight yang lebih detail mengenai aspek-aspek layanan aplikasi yang paling berpengaruh terhadap kepuasan pengguna.

DAFTAR PUSTAKA

[1] A. N. Hasanah and B. N. Sari, "Analisis Sentimen Ulasan Pengguna Aplikasi Jasa Ojek Online Maxim Pada Google Play Dengan Metode Naïve Bayes Classifier," *J. Inform. dan Tek. Elektro Terap.*, vol. 12, no. 1, pp. 90–96, 2024, doi: 10.23960/jitet.v12i1.3628.

[2] M. T. Diwandanu and L. M. Wisudawati, "Analisis Sentimen Terhadap Twit Maxim Pada Twitter Menggunakan R Programming Dan K Nearest Neighbors," *J. Ilm. Inform. Komput.*, vol. 28, no. 1, pp. 1–16, 2023, doi: 10.35760/ik.2023.v28i1.7909.

[3] R. Maulana, A. Voutama, and T. Ridwan, "Analisis Sentimen Ulasan Aplikasi MyPertamina pada Google Play Store menggunakan Algoritma NBC," *J. Teknol. Terpadu*, vol. 9, no. 1, pp. 42–48, 2023, doi: 10.54914/jtt.v9i1.609.

[4] A. Novantika, "Analisis sentimen ulasan pengguna aplikasi video conference google meet menggunakan metode svm dan logistic regression," *Prism. Pros. Semin. Nas. Mat.*, vol. 5, pp. 808–813, 2022, [Online]. Available: https://journal.unnes.ac.id/sju/index.php/prisma/

[5] R. Ramadhan, "Analisis Sentimen Pada Ulasan Aplikasi Di Google Play Store Dengan K-Nearest Neighbor," *J. Ekon. Vol. 18, Nomor 1 Maret201*, vol. 2, no. 1, pp. 41–49, 2020.

[6] D. Normawati and S. A. Prayogi, "Implementasi Naïve Bayes Classifier Dan Confusion Matrix Pada Analisis Sentimen Berbasis Teks Pada Twitter," *J. Sains Komput. Inform. (J-SAKTI*, vol. 5, no. 2, pp. 697–711, 2021.

[7] A. Saepudin, A. Faqih, and G. Dwilestari, "Perbandingan Algoritma Klasifikasi Support Vector Machine, Random Forest dan Logistic Regression Pada Ulasan Shopee," *J. Teknokompak*, vol. 18, no. 1, pp. 178–192, 2024, [Online]. Available: https://doi.org/10.33365/jtk.v18i1.3764

[8] E. R. Lidinillah, T. Rohana, and A. R. Juwita, "Analisis sentimen twitter terhadap steam menggunakan algoritma logistic regression dan support vector machine," *TEKNOSAINS J. Sains, Teknol. dan Inform.*, vol. 10, no. 2, pp. 154–164, 2023, doi: 10.37373/tekno.v10i2.440.

[9] I. S. K. Idris, Y. A. Mustofa, and I. A. Salihi, "Analisis Sentimen Terhadap Penggunaan Aplikasi Shopee Mengunakan Algoritma Support Vector Machine (SVM)," *Jambura J. Electr. Electron. Eng.*, vol. 5, no. 1, pp. 32–35, 2023, doi: 10.37905/jjeee.v5i1.16830.

[10] T. N. Wijaya, R. Indriati, and M. N. Muzaki, "Analisis Sentimen Opini Publik Tentang Undang-Undang Cipta Kerja Pada Twitter," *Jambura J. Electr. Electron. Eng.*, vol. 3, no. 2, pp. 78–83, 2021, doi: 10.37905/jjeee.v3i2.10885.

[11] L. B. Ilmawan and M. A. Mude, “Perbandingan Metode Klasifikasi Support Vector Machine dan Naïve Bayes untuk Analisis Sentimen pada Ulasan Tekstual di Google Play Store,” *Ilk. J. Ilm.*, vol. 12, no. 2, pp. 154–161, 2020, doi: 10.33096/ilkom.v12i2.597.154-161.

[12] E. Wahyusetyawati and Ng, “Dilema pengaturan transportasi online,” *J. RechtsVinding*, no. April, pp. 1–4, 2017, [Online]. Available: https://rechtsvinding.bphn.go.id

[13] T. Tinaliah and T. Elizabeth, “Analisis Sentimen Ulasan Aplikasi PrimaKu Menggunakan Metode Support Vector Machine,” *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 9, no. 4, pp. 3436–3442, 2022, doi: 10.35957/jatisi.v9i4.3586.

[14] L. Kaplow and S. Shavell, “Summary of Contents,” *Fairness versus Welf.*, pp. vii–viii, 2022, doi: 10.4159/9780674039315-001.

[15] A. R. Hidayati, A. S. Fitrani, M. A. Rosid, F. Sains, and U. Muhammadiyah, “Analisa Sentimen Pemilu 2019 Pada Judul Berita Online Menggunakan Metode Logistic Regression,” vol. 4, no. 2, pp. 298–305, 2023.

LAMPIRAN

Lampiran 1. Kode Program

```
!pip install google-play-scraper

!pip install pandas numpy nltk Sastrawi

!pip install matplotlib seaborn pillow wordcloud plotly

!pip install scikit-learn

# modul untuk scrape data

from google_play_scraper import Sort, reviews

# Modul untuk Preprocessing

import pandas as pd

from pandas import DataFrame

import numpy as np

import nltk

from nltk.corpus import words,stopwords

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

import string

import re

from collections import Counter

# Modul untuk Visualisasi

import matplotlib.pyplot as plt

import seaborn as sn

from PIL import Image

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import plotly.graph_objects as go

import plotly.express as px

# Modul untuk Proses Algoritma SVM

from sklearn import model_selection, metrics

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, precision_score, recall_score,
      f1_score, confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer,
      TfidfTransformer, TfidfVectorizer

from google_play_scraper import Sort, reviews
```

```
import pandas as pd
# Ambil ulasan aplikasi Maxim
all_reviews, _ = reviews(
    'com.taxsee.taxsee',
    lang='id',
    country='id',
    sort=Sort.MOST_RELEVANT,
    count=10000
)
# Buat DataFrame
data = [{
    'nama': review['userName'],
    'tgl': review['at'],
    'komentar': review['content'],
    'score': review['score'],
} for review in all_reviews]
df = pd.DataFrame(data)
df.head()
data_mentah = df[['nama','tgl','komentar','score']]
data_sortir = data_mentah.sort_values(by='tgl',ascending=False)
data_sortir.head()
#Cleaning Text
import re
import unicodedata
def clean_text_advanced(text):
  # Menghapus angka
  text = re.sub(r'\d+', '', text)
  # Menghapus emoji
  text = ''.join(c for c in text if not unicodedata.decomposition(c))
  # Menghapus URL
  text = re.sub(r'https?://\S+', '', text)
  # Menghapus simbol dan tanda baca
  text = re.sub(r'[^\w\s]', ' ', text)
```

```
    # Menghapus huruf berulang lebih dari 2 kali dalam setiap kata
    text = re.sub(r'(.)\1{2,}', r'\1', text)
    # Menghapus kata berulang (misalnya 'mantap mantap' menjadi 'mantap')
    text = re.sub(r'\b(\w+)(?:\s+\1\b)+', r'\1', text)
    return text.strip()
# Terapkan ke dataset
data['clean_text'] = data['komentar'].apply(lambda x: clean_text_advanced(x))
data[['komentar','clean_text']].head()
#Case Folding
def to_lower(text):
  text = text.lower()
  return text
data['case_folding'] = data['clean_text'].apply(lambda x: to_lower(x))
data[['clean_text','case_folding']].head(5)
#Nomalisasi kata
def replace_taboo_word(text, kamus_slag):
    if isinstance(text,str):
        words = text.split()
        replaced_words = []
        kalimat_baku = []
        kata_diganti = []
        kata_tidak_baku_hash = []
        for word in words:
            if word in kamus_slag:
                baku_word = kamus_slag[word]
                if isinstance(baku_word,str) and all(char.isalpha() for char in baku_word):
                    replaced_words.append(baku_word)
                    kalimat_baku.append(baku_word)
                    kata_diganti.append(word)
                    kata_tidak_baku_hash.append(word)
            else:
                replaced_words.append(word)
        replaced_text = ' '.join(replaced_words)
```

```
    else:
        replaced_text = ''
        kalimat_baku = []
        kata_diganti = []
        kata_tidak_baku_hash = []
    return replaced_text, kalimat_baku, kata_diganti, kata_tidak_baku_hash
#Tokenize
def tokenize(text):
    tokens = text.split()
    return tokens
df['tokenize'] = df['normalisasi'].apply(lambda x:tokenize(x))
df[['normalisasi','tokenize']].head(5)
from nltk.corpus import stopwords
nltk.download('stopwords')
# Load stopwords Bahasa Indonesia
stopword = set(stopwords.words('indonesian'))
stopword.update(['ya','nya'])
from nltk.corpus import stopwords
nltk.download('stopwords')
# Load stopwords Bahasa Indonesia
stopword = set(stopwords.words('indonesian'))
stopword.update(['ya','nya'])
# Stemmer untuk Bahasa Indonesia
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stem_text(words):
    """Melakukan stemming pada list kata."""
    return ' '.join([stemmer.stem(word) for word in words])
df['stem_text'] = df['no_stopword'].apply(lambda x:stem_text(x))
df[['no_stopword','stem_text']].head()
df.info()
df = df.drop_duplicates(subset='stem_text', keep='first')
df.info()
```

```
df = df[df['stem_text'].apply(lambda x: isinstance(x, str))]

# Cek lagis

df.info()

# Wordcloud hasil preprocessing

df_text = ' '.join(df['stem_text'].tolist())

stopwords = set(STOPWORDS)

# stopwords.update(['ya'])

wc = WordCloud(stopwords=stopwords,background_color='black', width=800,
height=400).generate(df_text)

plt.figure(figsize=(10, 5))

plt.imshow(wc, interpolation='bilinear')

plt.axis("off")

plt.title("WordCloud Visualization")

plt.show()

import pandas as pd

# Load lexicon positif dan negatif

positive_lexicon = set(pd.read_csv('positive.tsv', sep='\t', header=None)[0])

negative_lexicon = set(pd.read_csv('negative.tsv', sep='\t', header=None)[0])

# Fungsi penentuan sentimen

def determine_sentiment(text):

    positive_count = sum(1 for word in text.split() if word in positive_lexicon)

    negative_count = sum(1 for word in text.split() if word in negative_lexicon)

    if positive_count >= negative_count:

        return 'positif'

    else:

        return 'negatif'

# Terapkan ke kolom 'stem_text' dalam DataFrame

df['sentimen'] = df['stem_text'].apply(determine_sentiment)

# Tampilkan 11 data pertama

df.head(11)

X_train, X_test, y_train, y_test = train_test_split(

    df['stem_text'],

    df['sentimen'],
```

```
    test_size=0.2,
)
print(f'Jumlah Data Latih: {len(X_train)}')
print(f'Jumlah Data Uji: {len(X_test)}')
from sklearn.model_selection import cross_val_score

# SVM
svm_model = SVC(kernel='linear', probability=True)
svm_scores =cross_val_score(svm_model, tfidf_vectorizer.transform(df_balanced['stem_text']), df_balanced['sentimen'], cv=5)
# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_scores = cross_val_score(lr_model, tfidf_vectorizer.transform(df_balanced['stem_text']), df_balanced['sentimen'], cv=5)

# Tampilkan hasil dalam persen
print("=== Cross-Validation Score (SVM) ===")
print("Akurasi per fold:", [f"{score*100:.2f}%" for score in svm_scores])
print(f"Rata-rata akurasi: {svm_scores.mean()*100:.2f}%")

print("=== Cross-Validation Score (Logistic Regression) ===")
print("Akurasi per fold:", [f"{score*100:.2f}%" for score in lr_scores])
print(f"Rata-rata akurasi: {lr_scores.mean()*100:.2f}%")
```

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS ICHSAN GORONTALO
FAKULTAS ILMU KOMPUTER
SURAT KEPUTUSAN MENDIKNAS RI NOMOR 84/D/O/2001
Jl. Achmad Nadjamuddin No. 17 Telp (0435) 829975 Fax (0435) 829976 Gorontalo

SURAT REKOMENDASI BEBAS PLAGIASI

No. **077/FIKOM-UIG/R/V/2025**

Yang bertanda tangan di bawah ini :

Nama	: Irvan Abraham Salihi, M.Kom
NIDN	: 0928028101
Jabatan	: Dekan Fakultas Ilmu Komputer

Dengan ini menerangkan bahwa :

Nama Mahasiswa	:	Mardiansyah
NIM	:	T3121022
Program Studi	:	Teknik Informatika (S1)
Fakultas	:	Fakultas Ilmu Komputer
Judul Skripsi	:	Perbandingan Metode Support Vector Machine Dan Logistic Regression Pada Analisis Sentimen Ulasan Aplikasi Maxim Di Google Play Store

Sesuai hasil pengecekan tingkat kemiripan skripsi melalui aplikasi **Turnitin** untuk judul skripsi di atas diperoleh hasil *Similarity* sebesar **23%**, berdasarkan Peraturan Rektor No. 32 Tahun 2019 tentang Pendeteksian Plagiat pada Setiap Karya Ilmiah di Lingkungan Universitas Ichsan Gorontalo dan persyaratan pemberian surat rekomendasi verifikasi calon wisudawan dari LLDIKTI Wil. XVI, bahwa batas kemiripan skripsi maksimal 30%, untuk itu skripsi tersebut di atas dinyatakan **BEBAS PLAGIASI** dan layak untuk diujiankan.

Demikian surat rekomendasi ini dibuat untuk digunakan sebagaimana mestinya.

Gorontalo, 04 Mei 2025

Mengetahui
Dekan,

Irvan Abraham Salihi, M.Kom
NIDN. 0928028101

Tim Verifikasi,

Zulfrianto Y Lamasigi, M.Kom
NIDN. 0914089101

Terlampir :
Hasil Pengecekan Turnitin

KEMENTERIAN PENDIDIKAN TINGGI, SAINS DAN TEKNOLOGI
UNIVERSITAS ICHSAN GORONTALO
FAKULTAS ILMU KOMPUTER
SK MENDIKNAS NOMOR 84/D/O/2001
Jl. Achmad Nadjamudin, No.17, Telp.(0435) 829975, Fax (0435) 829976, Kota Gorontalo
Website : fikom-unisan.ac.id e-mail : info@fikom-unisan.ac.id

SURAT KETERANGAN PENELITIAN

Nomor : 070 /FIKOM-UIG/SKP/V/2025

Yang bertanda tangan dibawah ini :

N a m a	: Irvan Abraham Salihi, M. Kom
Jabatan	: Dekan Fakultas Ilmu Komputer

Dengan ini Menerangkan bahwa :

N a m a Mahasiswa	: Mardiansyah
N I M	: T3121022
Program Studi	: Teknik Informatika

Bahwa yang bersangkutan benar-benar telah melakukan penelitian tentang "**Perbandingan Metode Support Vector Machine Dan Logistic Regression Pada Analisis Sentimen Ulasa Aplikasi Maxim Di Google Play Store** " Guna untuk meyelesaikan Studi pada Program Studi Teknik Informatika Fakultas Ilmu Komputer, dan bersangkutan telah menyelesaikan penelitian Tersebut pada **TGL 25 April 2025** sesuai dengan waktu yang telah di tentukan.

Demikian Surat Keterangan ini dibuat dan digunakan untuk seperlunya.

Gorontalo, 02 Mei 2025

Dekan,

Irvan A. Salihi, S.Kom.,M.Kom

NIDN : 0928028101

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
UNIVERSITAS ICHSAN GORONTALO
FAKULTAS ILMU KOMPUTER
UPT. PERPUSTAKAAN FAKULTAS
SK. MENDIKNAS RI NO. 84/D/0/2001
Jl. Achmad Nadjamuddin No.17 Telp(0435) 829975 Fax. (0435) 829976 Gorontalo

SURAT KETERANGAN BEBAS PUSTAKA

No : 007/Perpustakaan-Fikom/IV/2025

Perpustakaan Fakultas Ilmu Komputer (FIKOM) Universitas Ichsan Gorontalo dengan ini menerangkan bahwa :

Nama Anggota	:	Mardiansyah
No. Induk	:	T3121022
No. Anggota	:	M202512

Terhitung mulai hari, tanggal : Sabtu, 19 April 2025, dinyatakan telah bebas pinjam buku dan koleksi perpustakaan lainnya.

Demikian keterangan ini di buat untuk di pergunakan sebagaimana mestinya.

Gorontalo, 19 April 2025
Mengetahui,
Kepala Perpustakaan

Apriyanto Alhamad, M.Kom
NIDN : 0924048601

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
UNIVERSITAS ICHSAN GORONTALO
FAKULTAS ILMU KOMPUTER
SK MENDIKNAS NOMOR 84/D/O/2001
Jl. Achmad Nadjamuddin No. 17 Telp. (0435) 829975 Fax (0435) 829976 Gorontalo

Berita Acara Perbaikan/Revisi Ujian PROPOSAL

Pada hari ini, Kamis 19 September 2024, Pukul 13.00-14.00 Wita. Telah dilaksanakan Ujian PROPOSAL mahasiswa Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.

Nama	:	Mardiansyah
Nim	:	T3121022
Pembimbing I	:	Yasin Aril Mustofa, M.Kom
Pembimbing II	:	Sunarto Taliki, M.Kom
Judul PROPOSAL	:	Perbandingan Metode Support Vector Machine Dan Logistic Regression Pada Analisis Sentimen Ulasan Aplikasi Maxim Di Google Play Store

Oleh Komite Seminar sebagai berikut :

No	Komite Seminar	Status	Tanda Tangan
1	Amiruddin, M.Kom, MCF	Ketua	
2	Muis Nanja, M.Kom	Anggota	
3	Maryam Hasan, M.Kom	Anggota	
4	Yasin Aril Mustofa, M.Kom	Anggota	
5	Sunarto Taliki, M.Kom	Anggota	

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS ICHSAN GORONTALO
FAKULTAS ILMU KOMPUTER
SK MENDIKNAS NOMOR 84/D/O/2001
JL. Achmad Nadjamuddin No. 17 Telp. (0435) 829975 Fax (0435) 829976 Gorontalo

Berita Acara Perbaikan/Revisi Ujian SKRIPSI

Pada hari ini, Senin 5 Mei 2025, Pukul 13.00-14.30 Wita. Telah dilaksanakan Ujian SKRIPSI mahasiswa Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.

Nama	:	Mardiansyah
Nim	:	T3121022
Pembimbing I	:	Yasin Aril Mustofa, M.Kom
Pembimbing II	:	Sunarto Taliki, M.Kom
Judul SKRIPSI	:	Perbandingan Metode Support Vector Machine Dan Logistic Regression Pada Analisis Sentimen Ulasan Aplikasi Maxim Di Google Play Store

Oleh Komite Seminar sebagai berikut :

No	Komite Seminar	Status	Tanda Tangan
1	Amiruddin, M.Kom, MCF	Ketua	
2	Muis Nanja, M.Kom	Anggota	
3	Maryam Hasan, M.Kom	Anggota	
4	Yasin Aril Mustofa, M.Kom	Anggota	
5	Sunarto Taliki, M.Kom	Anggota	

23% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Top Sources

0% Internet sources
14% Publications
17% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
168 suspect characters on 2 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

BIODATA MAHASISWA MARDIANSYAH
UNIVERSITAS ICHSAN GORONTALO
JURUSAN TEKNIK INFORMATIKA

Nama	:	Mardiansyah
Tempat/Tgl lahir	:	Palu, 29 Maret 1997
Jenis Kelamin	:	Pria
Gol Darah	:	B
Alamat	:	Jl. Ahmad A Wahad
RT/RW	:	000/000
Kel/Desa	:	Desa Luhu
Kec	:	Telaga
Agama	:	Islam
Suku	:	Bugis
Status Perkawinan	:	Belum Menikah
Pekerjaan	:	Mahasiswa
Kewarganegaraan	:	WNI
Riwayat Kependidikan	:	
SD	:	SDN 1 Bulila
SMP	:	SMP N 1 Telaga
SMA	:	SMA N 1 Telaga
Perguruan Tinggi	:	Universitas Ichsan Gorontalo