IMPLEMENTASI ALGORITMA *NAIVE BAYES* DAN *RANDOM FOREST* PADA ANALISIS SENTIMEN ULASAN APLIKASI *GETCONTACT* DI *GOOGLE PLAYSTORE*

SKRIPSI

Diajukan untuk Memenuhi Tugas Akhir dan Melengkapi Syarat
Guna Memperoleh Gelar Sarjana Strata Satu (S-1) dalam
Teknologi Informasi

Diajukan oleh:

SHAUQI NAZMI FAUZAN

NIM : 2108096074

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI WALISONGO

SEMARANG

2025

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Shauqi Nazmi Fauzan

NIM : 2108096074

Jurusan : Teknologi Informasi

Menyatakan bahwa skripsi yang berjudul:

IMPLEMENTASI ALGORITMA *NAIVE BAYES* DAN *RANDOM FOREST* PADA ANALISIS SENTIMEN ULASAN APLIKASI *GETCONTACT* DI *GOOGLE PLAYSTORE*

Secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 25 Juni 2025

Pembuat Pernyataan

Shauqi Nazmi Fauzan

NIM: 2108096074

LEMBAR PENGESAHAN

KEMENTRIAN AGAMA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof Dr. Hamka kampus III Ngaliyan
Semarang Telp. 7601295 Fax.7615387

LEMBAR PENGESAHAN

Naskah skripsi berikut ini:

Judul : **IMPLEMENTASI ALGORITMA *NAIVE BAYES* DAN *RANDOM FOREST* PADA ANALISIS SENTIMEN ULASAN APLIKASI *GETCONTACT* DI *GOOGLE PLAYSTORE***

Penulis : **Shauqi Nazmi Fauzan**

NIM : 2108096074

Jurusan : Teknologi Informasi

Telah diujikan dalam sidang *tugas akhir* oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Ilmu Teknologi Informasi.

Semarang, 30 Juni 2025

DEWAN PENGUJI

Penguji I,	Penguji II,
Hery Mustofa, M.Kom.	**Mokhamad Iklil M., M.Kom.**
NIP. 198703172019031007	NIP. 198808072019031010
Penguji III,	Penguji IV,
Nur Cahyo H. W., S.T.,M.Kom.	**Dr. Masy Ari Ulinuha, M.T**
NIP. 197312222006041001	NIP. 198108122011011007
Pembimbing I,	Pembimbing II,
Mokhamad Iklil M., M.Kom.	**Adzhal Arwani M., M.Kom**
NIP. 198808072019031010	NIP. 199107032019031006

NOTA PEMBIMBING

Semarang, 04/05/2025

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. wr. wb.

Dengan ini diberitahukan, bahwa saya telah melakukan bimbingan, arahan, dan koreksi naskah skripsi dengan :

Judul	:	IMPLEMENTASI ALGORITMA NAIVE BAYES DAN RANDOM FOREST PADA ANALISIS SENTIMEN ULASAN APLIKASI GETCONTACT DI GOOGLE PLAYSTORE
Penulis	:	Shauqi Nazmi Fauzan
NIM	:	2108096074
Jurusan	:	Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. wr. wb.

Pembimbing I,

Mokhamad Iklil Mustofa M.Kom
NIP. 198808072019031010

NOTA PEMBIMBING

Semarang, 04/05/2025

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. wr. wb.

Dengan ini diberitahukan, bahwa saya telah melakukan bimbingan, arahan, dan koreksi naskah skripsi dengan :

Judul	:	IMPLEMENTASI ALGORITMA NAIVE BAYES DAN RANDOM FOREST PADA ANALISIS SENTIMEN ULASAN APLIKASI GETCONTACT DI GOOGLE PLAYSTORE
Penulis	:	Shauqi Nazmi Fauzan
NIM	:	2108096074
Jurusan	:	Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. wr. wb.

Pembimbing II,

Adzhal Arwani Mahfudh, M.Kom
NIP. 199107032019031006

LEMBAR PERSEMBAHAN

Dengan rasa syukur yang mendalam, dengan telah diselesaikanya Tugas akhir ini, penulis mempersembahkan kepada:

1. Kedua orang tua penulis yang senantiasa memberikan semangat, dukungan penuh dan doa kepada penulis.
2. Segenap civitas akedemik UIN Walisongo Semarang, staff pengajar, karyawan, dan seluruh mahasiswa semoga selalu dalam keadaan sehat dan tetap semangat dalam beraktivitas mengisi hari-harinya di kampus tercinta UIN Walisongo Semarang.
3. Teman teman penulis yang selalu memberikan support kepada penulis dari awal hinggal akhir.

ABSTRAK

Perkembangan teknologi digital mempermudah komunikasi, namun sekaligus meningkatkan risiko penipuan digital. Getcontact hadir untuk membantu pengguna mengidentifikasi dan memblokir panggilan spam. Penelitian ini menganalisis sentimen pengguna terhadap Getcontact berdasarkan 2.000 ulasan di *Google Playstore* dengan algoritma *Naive Bayes* dan *Random Forest*. Ulasan dikumpulkan melalui web scraping, lalu menjalani preprocessing (cleaning, tokenizing, stopword removal, stemming). Fitur teks diekstraksi menggunakan TF-IDF, dan ketidakseimbangan kelas diatasi dengan SMOTE. Data kemudian dibagi menjadi set latih dan uji untuk mengevaluasi performa kedua model.

Hasil evaluasi menunjukkan bahwa *Naive Bayes* mencapai akurasi 87,4%, presisi 88,9%, recall 87,4%, dan F1-score 87,5%. Sedangkan *Random Forest* memperoleh akurasi 89,6%, presisi 91,2%, recall 89,6%, dan F1-score 89,7%. Berdasarkan metrik tersebut, *Random Forest* lebih unggul dalam mengklasifikasikan sentimen ulasan ke dalam tiga kategori (negatif, netral, positif). Temuan ini diharapkan membantu pengembang memahami persepsi pengguna dan meningkatkan kualitas layanan Getcontact.

Kata kunci: Analisis Sentimen, Getcontact, Multinomial *Naive Bayes*, *Random Forest*, *Google Playstore*.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan Laporan Tugas Akhir Skripsi yang berjudul "IMPLEMENTASI ALGORITMA *NAIVE BAYES* DAN *RANDOM FOREST* PADA ANALISIS SENTIMEN ULASAN APLIKASI GETCONTACT DI *GOOGLE PLAYSTORE*" dengan baik dan tepat waktu. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, suri teladan umat manusia, yang telah membimbing kita menuju jalan yang terang dan penuh ilmu pengetahuan.

Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Walisongo Semarang.Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah memberikan dukungan, bimbingan, dan doa selama proses penyusunan skripsi ini, antara lain kepada:

1. Prof. Dr. H. Musahadi, M.Ag., selaku Dekan Fakultas Sains dan Teknologi UIN Walisongo Semarang.

2. Dr. Khotibul Umam, S.T., M.Kom., selaku Ketua Program Studi Teknologi Informasi.
3. Bapak Mokhamad Iklil Mustofa M.Kom., selaku Dosen Pembimbing I yang telah memberikan bimbingan, arahan, serta masukan yang sangat berarti selama penyusunan skripsi.
4. Bapak Adzhal Arwani Mahfudh, M.Kom., selaku Dosen Pembimbing II yang juga telah memberikan banyak masukan dan koreksi demi kesempurnaan skripsi ini.
5. Seluruh dosen, staf, dan civitas akademika di Fakultas Sains dan Teknologi UIN Walisongo, khususnya Program Studi Teknologi Informasi, atas ilmu dan dukungan yang telah diberikan.
6. Orang tua dan keluarga tercinta yang selalu memberikan doa, semangat, serta kasih sayang yang tiada henti.
7. Eka Juni Rahayu yang telah setia membersamai dan memberikan semangat serta dukungan tanpa henti selama proses penyusunan skripsi ini
8. Teman-teman seperjuangan Aby, Ardan, Yudha, Daffa, Adi, Akmal dan anggota tim kontrakan yang selalu memberikan support yang baik dan positif.

9. Serta semua pihak yang telah membantu secara langsung maupun tidak langsung yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang bersifat membangun sangat penulis harapkan untuk perbaikan di masa mendatang. Akhir kata, penulis berharap semoga skripsi ini dapat memberikan manfaat, baik bagi penulis sendiri maupun bagi para pembaca sekalian.

MOTTO

Aku mencintai permasalahanku. Karena aku tahu, yang memberi permasalahan kepadaku pun mencintaiku.

(Maulana Jalaludin Rumi)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

A. Latar Belakang

Semua aspek kehidupan manusia dipermudah oleh teknologi informasi dan komunikasi. Dengan teknologi saat ini, komunikasi menjadi sangat cepat sehingga seolah-olah tanpa jarak. Media online, khususnya aplikasi komunikasi, memungkinkan pengguna berkomunikasi jarak jauh dengan menggunakan internet. Namun, WhatsApp tidak dapat mengetahui apakah orang yang berkomunikasi dengannya adalah orang yang aman atau penipuan.

Penipuan digital adalah jenis kejahatan yang meningkat pesat di Indonesia seiring dengan digitalisasi. Penelitian ini menunjukkan bahwa masyarakat sangat rentan terhadap penipuan digital; 98,3% dari 1.671 responden pernah menerima pesan penipuan digital, baik satu maupun lebih. Penipuan berkedok hadiah (91,2%), pinjaman ilegal (74,8%), pengiriman tautan/link yang mengandung virus atau malware (65,2%), penipuan berkedok krisis keluarga (59,8%), dan investasi ilegal (56,8%) adalah modus pesan

penipuan yang paling sering mereka terima. Jaringan seluler (SMS atau telepon) adalah cara komunikasi yang paling banyak digunakan dalam penipuan (64,1%). Ini karena mudah, murah, dan merupakan fitur utama telepon, sehingga memiliki jangkauannya yang luas. Media sosial (12,3%), aplikasi chat (9,1%), situs web (8,9%), dan email (3,8%) adalah media terbanyak selanjutnya (Kurnia -Rahayu -Engelbertus et al. n.d.).

Menurut *Getcontact*.com, Get Contact adalah aplikasi "Penyekat Spam" dan "Maklumat Pemanggil" untuk pengguna. Pengguna hanya dapat berkomunikasi dengan orang yang mereka pilih melalui Get Contact, yang menghilangkan panggilan yang mengganggu. Pengguna dapat membedakan panggilan yang diterima dari nomor-nomor yang tidak didaftarkan dalam Kenalan. Jutaan orang di seluruh dunia menggunakan aplikasi Get Contact, terutama di Indonesia. Di playstore, lebih dari 2 juta pengguna telah memberikan ulasan untuk Get Contact.

Aplikasi seperti Get Contact, yang berfungsi untuk melindungi pengguna dari panggilan atau pesan penipuan, secara tidak langsung mencerminkan ajaran Islam untuk menjauhkan diri dari tipu daya. Adapun ayat

yang menerangkannya, terdapat pada Q.S An-Nahl: 94 yang berbunyi:

وَلَا تَتَّخِذُوْٓا اَيْمَانَكُمْ دَخَلًاۢ بَيْنَكُمْ فَتَزِلَّ قَدَمٌۢ بَعْدَ ثُبُوْتِهَا وَتَذُوْقُوا السُّوْۤءَ بِمَا صَدَدْتُّمْ عَنْ سَبِيْلِ اللّٰهِ ۚ وَلَكُمْ عَذَابٌ عَظِيْمٌ

Artinya: *"Dan janganlah kamu jadikan sumpah-sumpahmu sebagai alat penipu di antaramu, yang menyebabkan kaki(mu) tergelincir setelah tegaknya (kukuh), dan kamu akan merasakan keburukan (di dunia) karena kamu menghalangi (manusia) dari jalan Allah, dan kamu akan mandapat azab yang besar."*

Dalam ayat diatas, mengandung pesan moral yang sangat berkaitan dengan kasus penipuan digital. Ayat-ayat ini memberikan pengingat penting untuk menghindari merusak orang lain dengan penipuan, mempertahankan kepercayaan orang lain, dan menghindari keburukan bagi diri sendiri dan masyarakat.

Dalam penelitian ini, aplikasi Get Contact menunjukkan tingkat kenyamanan yang rendah. Tingkat kepuasan pengguna dipengaruhi oleh masalah kenyamanan pengguna, yang dipengaruhi oleh pengalaman pengguna. Memiliki hak untuk menentukan tingkat kepuasan pelanggan adalah prinsip membangun pengalaman pengguna (aturan pelanggan). Jika target audiens tidak memiliki kepuasan, aturan, dan

kenyamanan dalam berinteraksi dengan produk, sistem, atau layanan apa pun, tingkat pengalaman pengguna akan sangat berkurang.

Lebih dari seratus juta unduhan telah diunduh untuk *Getcontact* di *playstore*. Karena aplikasi *Getcontact* sangat populer, penelitian ini akan memeriksanya. Analisis sentimen adalah bidang yang menganalisis perasaan dan emosi orang terhadap sesuatu. Analisis ulasan pengguna di *Google Playstore* diperlukan untuk menilai umpan balik pengguna terhadap layanan *Getcontact* (Figueroa et al. 2019).

Sebagai salah satu pendekatan yang relevan dalam analisis sentimen terhadap ulasan pengguna aplikasi *Getcontact*, dengan menggunakan metode klasifikasi probabilitas sederhana seperti *Naive Bayes* dan *Random Forest.* Metode *Naive Bayes* memiliki fungsi untuk menghitung sekumpulan kemungkinan dengan menjumlahkan frekuensi dan kombinasi nilai dari kumpulan data yang diberikan (Kawani Gigih Putra 2019). Sedangkan metode *Random Forest,* penggabungan hasil dari beberapa pohon keputusan untuk menghasilkan klasifikasi yang lebih akurat dan stabil.

Algoritma *Naive Bayes* dan *Random Forest* masing-masing memiliki keunggulan dan keterbatasan. Tujuan penelitian ini adalah untuk membandingkan kinerja kedua algoritma tersebut dalam klasifikasi sentimen ulasan untuk aplikasi *Getcontact* yang tersedia di *Google Playstore*. Python dipilih karena implementasinya yang mudah dan dukungannya terhadap proses text mining, yang mencakup tahap preprocessing hingga pengelompokan sentimen (Tanggraeni and Sitokdana 2022). Ulasan dikategorikan ke dalam kategori sentimen positif, negatif, dan netral untuk melakukan klasifikasi (Aditiya, Enri, and Maulana 2022).

B. Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan, penulis menemukan bahwa analisis sentimen perlu dilakukan terhadap ulasan pengguna aplikasi *Getcontact* di *Google Playstore*. Tujuan dari analisis ini adalah untuk mengetahui bagaimana pengguna melihat aplikasi tersebut dan menemukan aspek mana yang perlu diperbaiki atau dikembangkan.

C. Rumusan Masalah

Berdasarkan uraian dari latar belakang tersebut, maka pokok dari rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana implementasi algoritma *Naive Bayes* dan *Random Forest* digunakan dalam analisis sentimen pengguna aplikasi *Getcontact* di *Google Play store*?
2. Bagaimana perbandingan performa algoritma *Naive Bayes* dan *Random Forest* dalam analisis sentimen terhadap aplikasi *Getcontact* di *Google Play store*?

D. Tujuan Penelitian

Berdasarkan rumusan masalah telah diberikan, maka tujuan penelitian ini adalah sebagai berikut :

1. Menerapkan algoritma *Naive Bayes* dan *Random Forest* untuk menganalisis sentimen pengguna aplikasi *Getcontact* di *Google Playstore*.
2. Membandingkan performa yang dihasilkan oleh metode *Naive Bayes* dan *Random Forest* dalam mengkalisifikasi ulasan pengguna *Getcontact* kedalam sentimen positif, negatif, dan netral.

E. Batasan Penelitian

Peneliti menetapkan sejumlah batasan masalah yang diperlukan agar penelitian dapat dilakukan secara terarah dan objektif. Batasan-batasan ini adalah sebagai berikut:

1. Data yang dikumpulkan berasal dari komentar atau ulasan pengguna pada aplikasi *Getcontact* yang berasal dari *Google Play store*.
2. Metode *Naive Bayes* dan *Random Forest* digunakan untuk mengklasifikasikan hasil analisis sentimen ini.
3. Ulasan pengguna tentang aplikasi *Getcontact* akan dikategorikan menjadi tiga jenis sentimen yaitu sentimen positif, negatif, dan netral.
4. Penelitian ini menggunakan data berupa 2.000 ulasan pengguna yang diambil dari aplikasi *Getcontact*.
5. Bahasa pemrograman python dan *software google colab* digunakan dalam penelitian ini.
6. Data ulasan pengguna aplikasi *Getcontact* di *Google Playstore* dikumpulkan dalam rentang waktu 31 Desember 2024 - 19 Januari 2025.

F. Manfaat Penelitian

Adapun beberapa manfaat dari penelitian ini, yaitu berdasarkan manfaat secara teoritis dan manfaat secara praktis:

1. Manfaat Teoritis:

a. Melalui analisis ulasan yang dikategorikan menjadi positif, negatif, dan netral, kami dapat

membantu pihak perusahaan memahami persepsi pengguna terhadap aplikasi *Getcontact*.

b. Hasil analisis sentimen dapat digunakan sebagai referensi untuk mempertahankan kualitas, memperbaiki kesalahan, dan menilai pengembangan ke arah yang lebih baik.

2. Manfaat Praktis:

a. Mengkategorikan ulasan di aplikasi *Getcontact* ke dalam kategori negatif, positif, atau netral.

b. mengevaluasi kinerja metode *Random Forest* dan *Naive Bayes* untuk mengkategorikan ulasan pada aplikasi *Getcontact* yang tersedia di *Google Playstore*.

c. Penelitian tentang analisis sentimen dapat digunakan sebagai referensi untuk penelitian lain yang berkaitan dengan subjek ini.

BAB II
LANDASAN PUSTAKA

A. Konsep dan Penggunaan Analisis Sentimen

Analisis sentimen adalah teknik yang digunakan untuk mengekstrak data opini dan memahami serta mengolah tekstual data secara otomatis untuk mengidentifikasi sentimen yang melekat pada sebuah opini (Sari and Wibowo 2019). Analisis yang dilakukan terhadap data teks dari berbagai sudut pandang untuk mengidentifikasi pendapat positif atau negatif dari sejumlah besar teks dengan konteks yang beragam (Purba and Rizki Padya 2023).

Lima langkah utama biasanya terlibat dalam analisis sentimen: *crawling data*, *pre-processing*, *feature selection*, *classification*, dan *evaluation*. Tujuan dari proses ini adalah untuk mengubah data yang tidak terorganisir menjadi data yang terorganisir dengan baik. Analisis sentimen memiliki banyak manfaat, yaitu dapat digunakan untuk menilai dan memberi inspirasi di berbagai bidang. Karena popularitas dan manfaat analisis sentimen, penulisan dan penggunaan berbasis analisis sentimen meningkat pesat (Purba and Rizki Padya 2023

Analisis sentimen digunakan dalam banyak bidang, seperti pemasaran, penelitian konsumen, politik, dan kesehatan. Misalnya, perusahaan dapat memanfaatkan analisis ini untuk mengetahui bagaimana konsumen melihat produk atau layanan mereka, dan di sektor publik, analisis ini dapat membantu memahami pendapat masyarakat tentang kebijakan tertentu. Tujuan analisis sentimen berguna untuk menentukan opini atau komentar karena memiliki kecenderungan negatif atau positif. Ini dapat digunakan sebagai referensi untuk meningkatkan layanan atau kualitas produk (Suryono, Emha, and Luthfi 2020).

B. *Google Playstore* sebagai Sumber Data

Google Playstore adalah platform untuk konten digital yang memungkinkan pengguna smartphone Android mengunduh aplikasi dan produk digital lainnya. Selain itu, *Google Playstore* adalah salah satu platform aplikasi android terpopuler dan terbesar, dan memungkinkan pengguna menilai aplikasi dan memberikan ulasan mereka (Riansyah Ramadhan and Agus Sugianto 2024).

Ulasan pengguna sering mengandung informasi penting seperti laporan bug dan permintaan fitur yang dapat digunakan pengembang untuk meningkatkan dan

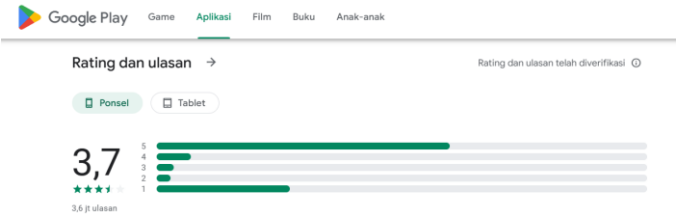
mempertahankan aplikasi game, film, e-book, dan lainnya, baik yang berbayar maupun gratis (Braja and Kodar 2023). Singkatnya, *Google Playstore* berfungsi sebagai platform yang membantu pengembang aplikasi berhubungan dengan pengguna perangkat Android secara tidak langsung.

C. Gambaran Umum *Getcontact*

Getcontact saat ini menjadi salah satu program komunikasi terpopuler di Indonesia. Sebagai platform yang inovatif, *Getcontact* bertujuan untuk membuat proses pengendalian pesan telepon menjadi lebih mudah bagi pengguna. Dengan fitur identifikasi, pengguna dapat dengan mudah mengetahui siapa yang sedang menelepon mereka bahkan jika nomor tersebut tidak terdaftar dalam daftar kontak. Ini membantu mengurangi kebingungan dan meningkatkan kepercayaan diri dalam menerima panggilan, terutama dari nomor yang tidak dikenal.

Inilah mengapa *Getcontact* semakin populer di antara pengguna yang mengutamakan kemudahan dan keamanan dalam berkomunikasi. Aplikasi *Getcontact* telah diunduh lebih dari 100 juta kali, menunjukkan betapa populernya di kalangan pengguna. Dengan rating rata-rata 3,7, aplikasi ini berada di kategori yang cukup

tinggi dibandingkan aplikasi sejenisnya. Selain itu, ada lebih dari 3,6 juta ulasan yang tersedia di *Google Playstore*, menjadikannya sumber data yang sangat berharga untuk mempelajari persepsi dan perasaan pengguna terhadap aplikasi.



Gambar 2. 1 Rating Aplikasi Getcontact

D. *Text mining*

Text mining merupakan proses ekstraksi informasi yang bernilai dari data teks yang bersifat tidak terstruktur atau semi-terstruktur. Dalam penerapannya, *text mining* memanfaatkan berbagai teknik seperti *machine learning*, *data mining*, *natural language processing* (NLP), manajemen pengetahuan, serta pencarian informasi untuk menggali pola dan wawasan dari teks secara otomatis (Ridwansyah 2022a). Tujuan utama *Text mining* adalah untuk mengubah teks yang tidak terstruktur menjadi format yang dapat diproses oleh komputer untuk analisis lebih lanjut.

Teknik ini bekerja dengan menghasilkan banyak variasi dari kumpulan data teks yang tersedia, lalu mencari pola yang sesuai dengan prediksi dari kumpulan data teks saat ini. Proses kerja *Text mining* hampir sama dengan penelitian data mining, tetapi pola yang digunakan dalam *Text mining* berasal dari kumpulan bahasa alami yang tidak terstruktur, sedangkan pola dalam data mining berasal dari database yang terstruktur (Ridwansyah 2022b).

E. *Scrapping* Data

Crawling data atau *scrapping* data, juga disebut sebagai proses otomatis untuk mengumpulkan dan mengindeks data dari berbagai sumber. *scrapping* data adalah sebuah proses menggali jauh ke dalam internet atau target tertentu untuk mengambil data (Duei Putri, Nama, and Sulistiono 2022). Tujuan *scrapping* data ini adalah untuk mengumpulkan data dari ulasan aplikasi *Getcontact* di *Google Playstore*, yang merupakan sumber data yang diperlukan untuk penelitian ini. Cara melakukan *scrapping* data ini adalah dengan membuat program yang secara otomatis mengumpulkan ulasan berdasarkan kata kunci yang relevan. Data dari kumpulan ulasan tersebut akan digunakan untuk

menganalisis sentimen menggunakan teknik *Naive Bayes*.

F. *Text preprocessing*

Dalam proses data mining, transformasi teks adalah tahap ketiga, dan tujuan dari tahap ini adalah membuat data yang berupakteksjksiap untuknndiproses. Dalam penelitian ini, tahap transformasi teks mencakup transformasi case, tokenizing, filtering,stopwords, dan stemming (Tanggraeni and Sitokdana 2022). Proses *Text preprocessing* adalah sebagai berikut:

1. *Cleaning*

   Cleaning data adalah proses membersihkan kalimat dari karakter emoji, tanda baca, spasi kosong, dan *single char* (karakter tunggal. Dalam pengolahan teks, *cleaning* data adalah proses penting yang bertujuan untuk membuat data mentah lebih terstruktur dan siap untuk dianalisis.

2. *Case Folding*

   Case folding adalah proses penyamaan case dalam sebuah dokumen. Tidak semua dokumen teks menggunakan huruf kapital secara konsisten. Akibatnya, case folding diperlukan untuk mengubah teks secara keseluruhan dalam dokumen menjadi

bentuk standar, biasanya dalam bentuk huruf kecil atau lowercase.

3. *Tokenizing*

   Tokenizing adalah suatu fase di mana string kata dipotong berdasarkan konstruksi kata tersebut. Hasil proses *cleaning* dan case folding kemudian dilanjutkan dengan proses tokenizing, di mana kalimat dipenggal kata perkata.

4. *Stopword removal*

   Stopword removal merupakan proses untuk menghilangkan kata-kata atau istilah yang dianggap tidak penting dan maknanya tidak berpengaruh, seperti kata-kata seperti "ke", "di", "pada", dan sebagainya.

5. *Stemming*

   Stemming adalah salah satu tahap *text praprocessing*, stemming bertujuan untuk menghilangkan imbuhan seperti awalan (prefix), akhiran (suffix), atau sisipan (infix) dari kata dan mengubahnya menjadi bentuk dasar atau kata asalnya.

G. TF-IDF

Analisis sentimen melibatkan data mentah berupa teks yang tidak dapat dibaca langsung oleh

komputer. Oleh karena itu, diperlukan teknik untuk mengubah data teks menjadi sekumpulan angka yang dapat diproses oleh sistem komputer. Metode TF-IDF *(Term Frequency Invers Document Frequency)* adalah teknik yang digunakan untuk memberikan bobot pada setiap kata untuk mengetahui seberapa dekat suatu kata (term) dengan dokumen tertentu (Andreswari, Suranti, and Aulia Trianggara 2024).

TF bertujuan untuk mengetahui seberapa sering suatu kata muncul dalam sebuah dokumen. Sedangkan IDF bertujuan untuk mengukur seberapa penting sebuah kata dalam konteks koleksi dokumen yang lebih besar, kata-kata yang muncul lebih jarang biasanya memiliki IDF yang lebih tinggi.

Berikut persamaan TF-IDF ditunjukkan pada persamaan 2.1

$$TFIDF(d,t) = TF(d,t).IDF(t) \quad (2.1)$$

Dalam hal ini, d adalah dokumen dan t adalah kata, dan TF(d,t) adalah jumlah kata yang ada di setiap dokumen yang dirumuskan pada persamaan 2.2.

$$TF(d,t) = \frac{jumlah\ kata\ t\ pada\ dokumen\ d}{total\ kata\ pada\ dokumen\ d} \qquad (2.2)$$

Dengan menggunakan rumus yang ditemukan dalam persamaan 2.3, IDF menghitung jumlah dokumen yang mengandung kata tertentu.

$$IDF(t) = log\frac{total\ dokumen}{jumlah\ yang\ mengandung\ kata\ t} \qquad (2.3)$$

H. SMOTE

Penggunaan algoritma klasifikasi tanpa mempertimbangkan ketidakseimbangan data cenderung membuat model lebih fokus pada kelas mayoritas dan mengabaikan kelas minoritas (Chawla et al. 2002). Dalam situasi tertentu, data yang tidak seimbang (*imbalanced*) tidak menjadi isu serius jika perbandingan jumlah antar kelas tidak terlalu mencolok. Namun, ketidakseimbangan menjadi masalah ketika terdapat perbedaan yang signifikan antara jumlah data di kelas mayoritas dan minoritas, karena hal ini dapat memengaruhi akurasi hasil evaluasi model.

Salah satu solusi untuk mengatasi permasalahan data yang tidak seimbang adalah dengan menerapkan metode *Synthetic Minority Oversampling Technique*

(SMOTE). SMOTE merupakan pendekatan yang cukup populer dalam menangani ketidakseimbangan data. Metode ini dikembangkan dari teknik oversampling, di mana prosesnya melibatkan pembuatan sampel baru dari kelas minoritas. Tujuannya adalah untuk menyeimbangkan distribusi data dengan melakukan sampling ulang pada data dari kelas minoritas (Siringoringo 2018). Proses pembuatan data sintetis merujuk pada persamaan 2.4.

$$x_{new} = x_i + (x_j|x_i)\delta \quad (2.4)$$

Keterangan :

x_{new} = Data sintesis yang diciptakan

x_i = Sampel kelas minoritas yang menjadi oversampling.

x_j = Data terdekat dari sampel minoritas

δ = Bilangan acak dari rentang 0 dan 1

I. Klasifikasi

Klasifikasi merupakan teknik yang didasarkan pada kategori yang telah ditetapkan sebelumnya, digunakan untuk menentukan kelas atau kelompok tertentu yang cocok untuk data tertentu (Yolanda Paramitha et al. 2023). Data dikelompokkan ke dalam kelas tertentu berdasarkan karakteristik yang serupa.

Ketika objek diklasifikasikan ke dalam tiga atau lebih kelas, itu disebut klasifikasi kompleks. Dalam kasus ini, penelitian ini diklasifikasikan ke dalam tiga kelas: positif, negatif, dan netral.

Klasifikasi terdiri dari dua tahap utama. Pertama, model dibangun dengan algoritma tertentu untuk membentuk model klasifikasi dari data latih yang memiliki label atau kelas. Tahap kedua adalah penggunaan model, di mana model yang telah dibuat diuji pada data baru yang label atau kelasnya belum diketahui (Duei Putri, Nama, and Sulistiono 2022). Jika tingkat akurasi yang tinggi diperoleh melalui perbandingan data sebenarnya dan hasil prediksi sistem, model tersebut dapat digunakan untuk memprediksi kelas pada data yang belum diketahui kelasnya (Prajamukti and Mega Santoni n.d.).

J. *Split Validation Data*

Metode validasi yang dikenal sebagai *split validation* data membagi dataset secara acak menjadi dua bagian. Bagian data pertama digunakan sebagai data latih (*training*) dan bagian data kedua digunakan sebagai data uji (*testing*).

Data latih (*training*) adalah data yang digunakan selama proses pembelajaran. Data uji (*testing*) adalah

data yang tidak digunakan selama proses pembelajaran dan digunakan untuk menguji kinerja model (Turmudi Zy, Adji Ardiansyah, and Maulana 2021).

K. *Naive Bayes classifier*

Berdasarkan teorema Bayes, NaiveBayes Classifier menggunakan metode probabilitas dan statistik untuk memprediksi peluang di masa depan berdasarkan pengalaman masa lalu. Algoritma *Naive Bayes* menggunakan teori probabilitas dalam cabang matematika untuk menentukan peluang terbesar dari suatu klasifikasi. Ini dilakukan dengan menganalisis frekuensi masing-masing klasifikasi berdasarkan data pelatihan dan tidak bergantung pada aturan tertentu (Retnosari et al. 2021).

Berikut adalah bentuk umum dari teorema Bayes (Prawira, Arisandi, and Sutrisno 2022):

$$P(H|X) = \frac{P(X|H).P(H)}{P(X)} \qquad (2.5)$$

Keterangan :

X = Data dengan kelas yang belum diketahui.

H = hipotesa data X yang merupakan suatu kelas spesifik.

P(H|X) = Peluang hipotesis H berdasarkan kondisi X (Posteriori prob)

P(H) = Peluang hipotesis H (prior prob)

P(X|H) = Peluang X berdasarkan kondisi tersebut

P(X) = Peluang dari X

Salah satu kelebihan Algoritma *Naive Bayes* adalah kemudahan penggunaan karena proses perhitungannya sederhana, cepat, dan efisien dalam penggunaan ruang. Selain itu, algoritma ini sangat tahan terhadap fitur yang tidak relevan dan membutuhkan sedikit data pelatihan untuk mengestimasi parameter yang diperlukan, seperti variansi variabel dan rata-rata. Oleh karena itu, algoritma ini cocok untuk menganalisis data yang sangat besar. Namun, salah satu kelemahan algoritma ini adalah ketergantungannya pada probabilitas kondisional, hasil prediksi juga akan menjadi nol jika probabilitas kondisional nol. Ini dapat memengaruhi akurasi prediksi.

L. *Random Forest*

Random Forest adalah sebuah algoritma yang dikembangkan dari model algoritme *decision*. Ini melatih setiap pohon fikiran menggunakan sampel unik (Aldean et al., 2022). *Random Forest* algoritme dibuat oleh J. Ross Quinlan sebagai pengembangan dari pendekatan ID3, yang digunakan untuk membuat pohon keputusan, dan digunakan untuk mengklasifikasikan sejumlah besar data. *Random Forest* dianggap sebagai

solusi yang tepat untuk masalah klasifikasi dalam *machine learning* dan data *mining*. Beberapa keunggulan metode ini, menurut Pamuji dan Ramdhan, termasuk tingkat kesalahan yang relatif rendah, kinerja yang baik, dan kemampuan untuk menangani data yang sangat besar.

M. Evaluasi

Evaluasi adalah proses untuk menilai kinerja model klasifikasi. Ini melakukan pengukuran melalui pengujian untuk menentukan kualitas model yang dievaluasi dan bertujuan untuk memastikan bahwa model yang digunakan adalah akurat (Phafiandita et al. 2022). Berikut pada tabel 2.1 beberapa metode evaluasi yang dapat digunakan untuk mengukur keakuratan:

Tabel 2. 1 Confusion Matrix

		True Class	
		True	*False*
Predicted Class	*True*	*TP*	*FP*
	False	*FN*	*TN*

Empat nilai yang digunakan dalam perhitungan untuk confusion matrix tersebut adalah:

1. TP (true positive), yaitu data yang diprediksi positif dan faktanya data itu positif.

2. TN (true negative), yaitu data yang diprediksi negatif dan faktanya data itu negatif.
3. FP (false positive), yaitu data yang diprediksi positif dan faktanya data itu negative.
4. FN (false negative), yaitu data yang diprediksi negatif dan faktanya data itu postif.

Untuk mengevaluasi kinerja *Naive Bayes* Classifier, nilai akurasi, presisi, *recall*, dan f1 *score* dihitung (Ahmad et al. 2020).

1. Akurasi

   Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai actual. Akurasi dapat dihitung menggunakan rumus yang tercantum pada persamaan 2.6.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.6)$$

2. Presisi

   Presisi adalah tingkat ketepatan atau kesesuaian antara informasi yang dicari dan hasil yang dihasilkan oleh sistem. Ketika seseorang mencari informasi dalam sistem dan sistem memberikan sejumlah dokumen, tingkat ketepatan dokumen tersebut menunjukkan relevansi, atau sejauh mana dokumen tersebut memenuhi kebutuhan pencari informasi.

Nilai presisi diperoleh dari Persamaan dan didefinisikan sebagai rasio prediksi benar positif dibandingkan dengan hasil yang diprediksi secara keseluruhan positif. Presisi dapat dihitung menggunakan rumus yang tercantum pada persamaan 2.7.

$$Presisi = \frac{TP}{TP+FP} \quad (2.7)$$

3. *Recall*

Recall adalah metrik yang mengukur kemampuan model untuk dengan tepat mengidentifikasi semua data yang termasuk dalam kategori nilai positif murni. Dengan kata lain, *recall* menunjukkan seberapa berhasil model menangkap semua instance yang relevan dari kelas tertentu. *Recall* dapat dihitung menggunakan rumus yang tercantum pada persamaan 2.8.

$$Recall = \frac{TP}{TP+FN} \quad (2.8)$$

4. *F1 score*

F1 score adalah nilai rata-rata harmonis untuk presisi dan *recall* sistem segmentasi yang diuji. Ini digunakan untuk menyeimbangkan kedua elemen tersebut dan memberikan gambaran yang lebih adil tentang kinerja model, terutama dalam kasus di mana ada trade-off antara *precision* dan *recall*. *F1*

score dapat dihitung menggunakan rumus yang tercantum pada persamaan 2.9.

$$F1 - Score = 2\frac{(Presisi\ .\ Recall)}{Presisi+Recall} \quad (2.9)$$

Karena output sentimen yang dihasilkan terdiri dari tiga kelas yaitu, negatif, netral, dan positif. Penelitian ini menggunakan *multiclass confusion matrix* 3x3. *Multiclass confusion matrix* ini digambarkan pada Tabel 2.2.

Tabel 2. 2 Multiclass Confusion Matrix

		True Class		
		Negative	*Netral*	*Positive*
Predicted Class	*Negative*	*T Neg*	*F NegNet*	*F NegPos*
	Netral	*F NetNeg*	*T Net*	*F NetPos*
	Positif	*F PosNeg*	*F PosNet*	*T Pos*

Dalam matrix confusion multiclass ini, sembilan nilai digunakan sebagai dasar perhitungan, yaitu:

1. T Pos (*True Positve*), yaitu jumlah data yang diprediksi sebagai positif dan faktanya positif.
2. T Net (True Netral), yaitu jumlah data yang diprediksi sebagai netral dan faktanya netral,
3. T Neg (*True Negative*), yaitu jumlah data yang diprediksi sebagai negatif dan faktanya negatif.
4. F PosNet (*False Positive Netral*), yaitu jumlah data yang diprediksi positif dan faktanya netral.

5. F PosNeg (*False Positive Negative*), yaitu jumlah data yang diprediksi positif dan faktanya negatif.
6. F NetPos (*False Netral Positive*), yaitu jumlah data yang diprediksi netal dan faktanya positif.
7. F NetNeg (*False Netral Negative*), yaitu jumlah data yang diprediksi netral dan faktanya negatif.
8. F NegPos (*False Negative Positive*), yaitu jumlah data yang diprediksi negatif dan faktanya positif.
9. F NegNet (*False Negative Netral*), yaitu jumlah data yang diprediksi negatif dan faktanya netral.

N. Penelitian Terdahulu

Berikut detail dari rujukan penelitian ditunjukan pada tabel 2.3

Tabel 2. 3 Penelitian Terdahulu

No	Pustaka	Topik	Metode	Objek	Klasifikasi dan Hasil
1	(Noer Azzahra et al. 2024)	Klasifikasi pesan SMS menjadi spam atau ham menggunakan algoritma *Naive Bayes*	*Naive Bayes*	Pesan SMS	2 Kelas (Spam dan Ham). Akurasi 93,2%, presisi 93,7%, *recall* 93,2%, *F1-Score* 91,6%,

					ROC 97,3%.
2	(Hermanto et al. 2024)	Perbandingan algoritma Naïve Bayes dan Support Vector Machine (SVM) untuk analisis sentimen ulasan pengguna aplikasi *Getcontact*	*Naive Bayes* dan SVM	Aplikasi *Getcontact*	2 Kelas (Positif dan Negatif). Naïve Bayes: Akurasi 82,97%, AUC 0,500. SVM: Akurasi 78,00%, AUC 0,926.
3.	(Aldean et al. 2022)	Analisis sentimen masyarakat terhadap vaksinasi Sinovac di Twitter	*Random Forest*	Tweet tentang vaksin Sinovac	2 Kelas (Positif dan Negatif). Akurasi 79%, *Precision* 85%, *Recall* 90%, *F1-Score* 88%. Sentimen mayoritas positif (71,33%).

Studi sebelumnya menunjukkan bahwa algoritma *Naive Bayes*, Support Vector Machine (SVM), dan *Random Forest* banyak digunakan untuk menganalisis sentimen pada berbagai objek, seperti ulasan aplikasi, pesan SMS, dan media sosial. Studi (Hermanto et al. 2024) membandingkan kemampuan Naïve Bayes dan SVM dalam analisis sentimen ulasan aplikasi *Getcontact*. Penelitian lain (Noer Azzahra et al. 2024) menemukan bahwa *Naive Bayes* lebih baik dalam mengklasifikasikan sentimen positif dan negatif daripada SVM. Selain itu, (Aldean et al. 2022) menggunakan algoritma *Random Forest* untuk menganalisis persepsi masyarakat terhadap vaksinasi Sinovac di Twitter. Algoritma ini menghasilkan hasil klasifikasi yang sebagian besar positif.

Berdasarkan hasil-hasil tersebut, peneliti ingin melakukan analisis sentimen pada ulasan aplikasi *Getcontact* di *Google Playstore* dengan menggunakan algoritma *Naive Bayes* dan *Random Forest*. Tujuan penelitian ini adalah untuk membandingkan bagaimana kedua algoritma tersebut bekerja dalam mengkategorikan sentimen ulasan pengguna ke dalam tiga kategori, yaitu positif, negatif, dan netral. Diharapkan penelitian ini akan menghasilkan hasil

analisis yang valid, akurat, dan relevan untuk membantu pengembangan algoritma klasifikasi yang lebih baik dengan menggunakan dataset ulasan pengguna aplikasi *Getcontact* yang belum pernah dianalisis sebelumnya.

BAB III
METODOLOGI PENELITIAN

A. Metode Pengumpulan Data

1. Studi Pustaka

Penulis melakukan studi pustaka dengan memanfaatkan buku, jurnal, skripsi, dan sumber sejenis untuk mempelajari konsep, analisis, serta permasalahan yang berkaitan dengan penelitian. Selain itu, penulis juga mengumpulkan data secara daring melalui internet dengan mengunjungi situs web yang relevan dengan analisis sentimen, text mining, serta implementasi algoritma *Naive Bayes* dan *Random Forest*.

2. Observasi

Tahap observasi dilakukan oleh peneliti dengan tujuan mengamati secara langsung ulasan yang terdapat di *Google Playstore* terkait aplikasi *Getcontact*. Selain itu, peneliti juga menggunakan aplikasi tersebut sebagai pengguna. Dari observasi yang dilakukan, peneliti memperoleh hasil sebagai berikut:

a. Terdapat informasi mengenai berbagai kendala yang dialami oleh pengguna aplikasi *Getcontact*.
b. Ditemukan informasi terkait kelebihan serta kekurangan dari aplikasi *Getcontact*.

B. Kebutuhan Perangkat Penelitian

Untuk memastikan bahwa proses sistem dapat berjalan dengan baik, pengimplementasian sistem membutuhkan beberapa perangkat lunak dan perangkat keras yang mendukung. Berikut adalah kebutuhan yang digunakan dalam penelitian ini:

1. Kebutuhan Perangkat Keras:

Tabel 3. 1 Kebutuhan Perangkat Keras

No	Perangkat Keras	Spesifikasi
1	Device	Lenovo LOQ 15APH8
2	Processor	AMD Ryzen™ 5 7640HS (4.3/5.0 GHz)
3	Memori (RAM)	16.00 GB
4	Monitor	15.6 Inch
5	Keyboard dan Mouse	Normal

2. Kebutuhan Perangkat Lunak:

Tabel 3. 2 Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Windows 11 Home Single Language
2	Bahasa Pemrograman	Python
3	Ms. Office	Ms. Word & Ms. Excel
4	Browser	Chrome
5	Google Drive	Google Colab

C. Langkah Analisis

Tahapan analisis data dalam penelitian ini adalah sebagai berikut:

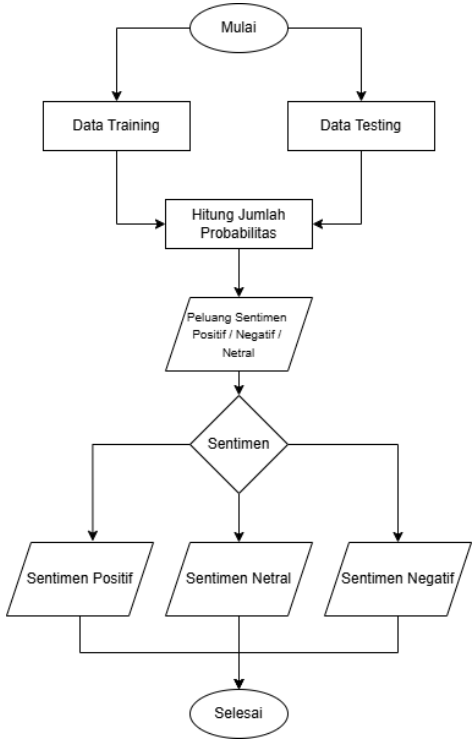
1. Scrapping data ID aplikasi *Getcontact* dari *Google Playstore*
    a. Mengambil ID aplikasi *Getcontact*
    b. Scrapping data menggunakan ID aplikasi *Getcontact* menggunakan *python*
    c. Menyimpan data dalam bentuk csv
2. Mengumpulkan data ulasan aplikasi dari *Google Playstore*
    a. Melakukan pelabelan data dengan nilai positif, negatif, dan netral
    b. Membaca data csv dalam *python*

3. Text processing
    a. Melakukan *cleaning* data
    b. Melakukan *case folding*
    c. Melakukan *tokenizing*
    d. Melakukan *stopword removal*
    e. Melakukan *stemming*
4. Melakukan TF-IDF pada data
5. Melakukan *split validation* data
6. Implementasi algoritma klasifikasi
    a. *Naive Bayes*

        Untuk analisis sentimen, metode klasifikasi *Naive Bayes* terdiri dari beberapa langkah:

        1) Identifikasi fitur adalah proses memilih kata atau frasa yang berfungsi sebagai indikator suatu sentimen.
        2) menghitung peluang, dengan menghitung berapa kali fitur tertentu muncul dalam sebuah ulasan
        3) Asumsi independensi berarti bahwa setiap kata dianggap tidak memiliki hubungan apa pun dengan kata lainnya.
        4) Berdasarkan ulasan yang tersedia, kemungkinan untuk setiap kategori (positif,

negatif, atau netral) dihitung untuk menentukan probabilitas kategori.



Gambar 3. 1 Alur Naive Bayes

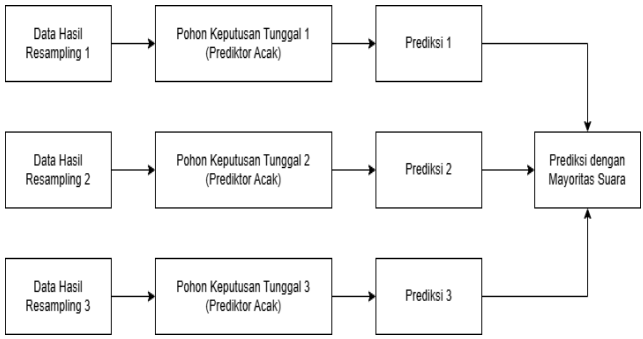
b. *Random Forest*

Berikut merupakan tahapan dalam melakukan pengklasifikasian dengan menggunakan *Random Forest*:

1) Untuk menghasilkan beberapa subset data dari dataset awal, sampel acak diambil

dengan pengembalian dari dataset asli berukuran N. Teknik bootstrap digunakan untuk melakukan resampling.

2) Dengan pemilihan fitur acak (mtry), satu pohon keputusan dibangun dari setiap subset data.
3) Setelah pemilihan m secara acak, pohon dibangun tanpa melakukan proses pemangkasan (pruning).
4) Langkah 1–3 diulangi sebanyak n kali hingga menghasilkan sebuah *forest* (hutan) dengan n pohon.
5) Penentuan suatu kelas dilakukan melalui cara *majority vote* (suara mayoritas).
6) Langkah selanjutnya adalah menemukan parameter *mtry* terbaik untuk menghasilkan *out of bag error* (nilai error misklasifikasi) yang stabil dan menentukan tingkat *variabel importance* (kepentingan variabel).
7) Setelah nilai *mtry* optimal ditemukan, prediksi dibuat menggunakan data *testing*.

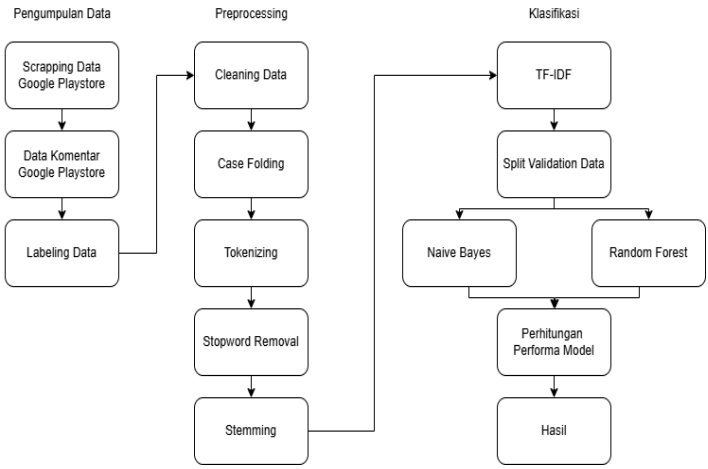


Gambar 3. 2 Alur Sederhana Random Forest

7. menggunakan library python untuk melakukan perhitungan *akurasi*, *presisi*, *recall*, dan *f1 score* untuk setiap jenis klasifikasi, dan kemudian membandingkan hasil yang diperoleh.
8. menggunakan metode *Naive Bayes* dan *Random Forest*, dan memberikan visualisasi untuk mempermudah pemahaman.

D. Alur Pengerjaan Penelitian

Perancangan alur penelitian ini menjelaskan gambaran umum mengenai tahapan-tahapan yang dilakukan oleh peneliti, mulai dari awal hingga akhir. Proses tersebut dapat dilihat pada Gambar 3.3.



Gambar 3. 3 Alur Pengerjaan Penelitian

1. Pengambilan Data

   Teknik scrapping data dilakukan dengan membuat aplikasi terlebih dahulu dan kemudian menampilkan informasi tentang aplikasi dengan memasukkan keyword "Aplikasi *Getcontact*" di *Google Playstore*. Setelah aplikasi *Getcontact* muncul, ID aplikasi diambil melalui halaman *Google Playstore*, yang bernama app.source.*Getcontact*. Data dikumpulkan dari 31 Desember 2024 hingga 19 Januari 2025. Data yang dikumpulkan termasuk nama pengguna, rating, komentar, dan informasi lainnya seperti ulasan pengguna aplikasi *Getcontact* yang diambil dari situs *Google Play* berbahasa

Indonesia. Selanjutnya, data ini disimpan dalam file.csv.

Gambar 3. 4 Pengambilan ID Aplikasi

2. Labeling Ulasan

Selanjutnya, data yang telah dikumpulkan diklasifikasikan berdasarkan sentimen yang terkandung di dalamnya. Tiga kategori sentimen digunakan dalam penelitian ini, yaitu positif, negatif, dan netral. Pelabelan merupakan langkah penting dalam proses klasifikasi data.

Tabel 3. 3 Penerapan Labeling

Ulasan	Sentimen	Label
Ada telpon masuk tidak di kenal, sangat membantu sekali untuk bisa tau namanya si penelpon	positif	2
Bagus tapi terbatas.	netral	1
Maaf ya ratingnya tak turunkan. Saya sudah berlangganan setiap bulan tp nama2 orang yg telp tidak	negatif	0

pernah muncul. Jadi percuma, buang2 uang		

3. Preprocessing

Tahap ini bertujuan untuk membuat data mentah bersih sehingga dapat digunakan dalam proses klasifikasi. Data yang diperoleh dari *scrapping* ulasan pengguna di *Google Playstore* digunakan untuk melakukan tahapan *preprocessing*. Proses *preprocessing* termasuk:

a. *Cleaning*

Langkah penting dalam proses preprocessing teks adalah *cleaning*, yang bertujuan untuk menghilangkan karakter yang tidak penting seperti tanda baca, emoji, URL, dan pengulangan karakter. Untuk membuat proses pengolahan data selanjutnya lebih mudah, proses penyederhanaan teks dilakukan. Sebagai contoh, emoji, simbol, dan tanda baca sering digunakan terlalu banyak dalam ulasan produk. Proses membersihkan menghapus elemen-elemen tersebut agar teks menjadi lebih rapi dan siap untuk dianalisis. Tabel 3.4 menunjukkan contoh hasil *cleaning*.

Tabel 3. 4 Penerapan Pada Tahap Cleaning

Input Process	*Output Process*

Sayang sekali,aplikasi ini harusnya bagus,cuman sekarang serba berbayar dan banyak iklan! padahal dulu udah bagus sekarang malah mengecewakan.saya harap aplikasi ini bisa seperti dulu lagi	Sayang sekali aplikasi ini harusnya bagus cuman sekarang serba berbayar dan banyak iklan padahal dulu udah bagus sekarang malah mengecewakan saya harap aplikasi ini bisa seperti dulu lagi

b. *Case Folding*

Tujuan dari tahapan ini adalah untuk menyeragamkan bentuk kata agar lebih konsisten, sehingga seluruh huruf dalam suatu kata, kalimat, atau teks disama-samakan menjadi huruf kecil (lowercase). Tabel 3.5 berikut menunjukkan hasil proses *case folding*.

Tabel 3. 5 Penerapan Pada Tahap Case Folding

Input Process	*Output Process*
Sayang sekali aplikasi ini harusnya bagus cuman sekarang serba berbayar dan banyak iklan padahal dulu udah bagus sekarang malah mengecewakan	sayang sekali aplikasi ini harusnya bagus cuman sekarang serba berbayar dan banyak iklan padahal dulu udah bagus sekarang

saya harap aplikasi ini bisa seperti dulu lagi	malah mengecewakan saya harap aplikasi ini bisa seperti dulu lagi

c. *Tokenizing*

Proses memecah teks menjadi bagian-bagian kecil yang terdiri dari kata atau token dikenal sebagai *tokenizing*. Tanda baca dan karakter khusus biasanya dihapus pada tahap ini karena dianggap tidak penting untuk analisis lebih lanjut. Tabel 3.6 menunjukkan contoh hasil tokenizing.

Tabel 3. 6 Penerapan Pada Tahap Tokenizing

Input Process	*Output Process*
sayang sekali aplikasi ini harusnya bagus cuman sekarang serba berbayar dan banyak iklan padahal dulu udah bagus sekarang malah mengecewakan saya harap aplikasi ini bisa seperti dulu lagi	['sayang', 'sekali', 'aplikasi', 'ini', 'harusnya', 'bagus', 'cuman', 'sekarang', 'serba', 'berbayar', 'dan', 'banyak', 'iklan', 'padahal', 'dulu', 'udah', 'bagus', 'sekarang', 'malah', 'mengecewakan', 'saya', 'harap', 'aplikasi', 'ini', 'bisa', 'seperti', 'dulu', 'lagi']

d. *Stopword Removal*

Untuk menghilangkan kata-kata umum yang tidak penting atau tidak relevan dengan informasi yang diperlukan, proses stopword removal dilakukan. Proses ini bertujuan untuk mengurangi jumlah kata yang harus disimpan sistem. Tabel 3.7 berikut menunjukkan contoh hasil proses penghapusan stopword.

Tabel 3. 7 Penerapan Pada Tahap Stopword Removal

Input Process	*Output Process*
['sayang', 'sekali', 'aplikasi', 'ini', 'harusnya', 'bagus', 'cuman', 'sekarang', 'serba', 'berbayar', 'dan', 'banyak', 'iklan', 'padahal', 'dulu', 'udah', 'bagus', 'sekarang', 'malah', 'mengecewakan', 'saya', 'harap', 'aplikasi', 'ini', 'bisa', 'seperti', 'dulu', 'lagi']	['sayang', 'aplikasi', 'bagus', 'cuman', 'serba', 'berbayar', 'iklan', 'udah', 'bagus', 'mengecewakan', 'harap', 'aplikasi']

e. *Stemming*

Tujuan dari tahap stemming adalah untuk menyederhanakan kata berulang menjadi satu kata dasar saja dan mengubah

kata yang memiliki imbuhan menjadi bentuk kata dasar. Tabel 3.8 berikut menunjukkan contoh hasil dari proses stemming.

Tabel 3. 8 Penerapan Pada Tahap Stemming

Input Process	*Output Process*
['sayang', 'aplikasi', 'bagus', 'cuman', 'serba', 'berbayar', 'iklan', 'udah', 'bagus', 'mengecewakan', 'harap', 'aplikasi']	['sayang', 'aplikasi', 'bagus', 'cuman', 'serba', 'bayar', 'iklan', 'udah', 'bagus', 'kecewa', 'harap', 'aplikasi']

4. Ekstrasi Fitur dengan TF-IDF

Setelah data selesai melalui tahap *preprocessing*, langkah berikutnya adalah membuat fitur yang membantu proses klasifikasi. Metode pembobotan TF-IDF (*Term Frequency-Inverse Document Frequency*) digunakan untuk pembuatan fitur dalam penelitian ini. Setiap kata dalam ulasan diberi bobot melalui proses ini untuk menunjukkan seberapa penting kata tersebut dalam dokumen. Bab sebelumnya memberikan penjelasan tentang rumus perhitungan TF-IDF.

5. Uji model

Teknik validasi yang disebut *split validation* membagi dataset secara acak menjadi tahap

pelatihan data dan pengujian data untuk mengevaluasi kinerja model yang dihasilkan. Sebagian 20% dari data latih digunakan sebagai data uji. Pengambilan data uji dilakukan secara acak menggunakan bantuan library Python. Uji model dilakukan setelah proses pelatihan selesai, untuk mengevaluasi kinerja model. Hasil dari uji model ini menunjukkan seberapa baik performa metode yang diterapkan.

6. Implementasi Algoritma Klasifikasi

   Setelah melalui tahap preprocessing dan ekstraksi fitur menggunakan TF-IDF, data dilanjutkan ke tahap pengujian dengan membagi dataset menjadi data training dan data testing untuk mengevaluasi akurasi klasifikasi menggunakan metode *Naive Bayes* dan *Random Forest*. Data yang telah diproses dan diekstraksi fitur-fiturnya akan diklasifikasikan ke dalam tiga kategori output, yaitu positif, negatif, dan netral. Pada proses pengklasifikasian, metode *Naive Bayes* dan *Random Forest* digunakan untuk mengukur ketepatan klasifikasi. Kategori dengan skor probabilitas tertinggi akan dianggap sebagai hasil prediksi untuk setiap ulasan.

7. Evaluasi Model

   Untuk mengetahui seberapa baik kinerja model, evaluasi model dilakukan dengan melihat kinerja metode melalui *multiclass confusion matrix*. Pada dasarnya, confusion matrix membandingkan hasil klasifikasi sistem dengan hasil klasifikasi sebenarnya. Di sini, data uji diuji dengan data latih, menghasilkan prediksi kelas. Prediksi kelas ini kemudian dibandingkan dengan kelas sebenarnya dari data uji yang sebelumnya disembunyikan. Hasil pemeriksaan ini menunjukkan kinerja model, termasuk akurasi, presisi, *recall*, dan *F1 score*.

8. Visualisasi

   Salah satu langkah dalam proses analisis sentimen adalah menampilkan data hasil pengklasifikasian menggunakan *wordcloud*. Tujuan visualisasi ini adalah untuk mengekstraksi informasi yang diimplementasikan dalam bentuk topik-topik yang paling sering disampaikan oleh pengguna aplikasi. Dengan demikian, memberikan gambaran umum tentang kata-kata yang paling banyak digunakan dalam ulasan, sehingga lebih mudah untuk menganalisis pola sentimen secara keseluruhan.

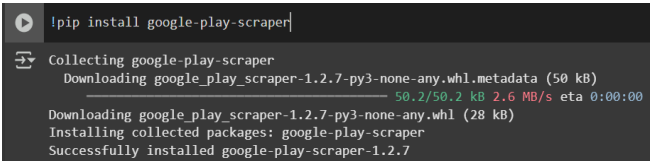
BAB IV
HASIL DAN PEMBAHASAN

A. Pengambilan Data Ulasan *Google Playstore*

Teknik web scraping digunakan untuk mengumpulkan data ulasan pengguna untuk aplikasi Getcontact yang tersedia di Google Play Store. Proses ini dilakukan menggunakan bahasa pemrograman Python, dan langkah-langkah berikut:

1. Install *Google Play Scrapper Package*

   Tahap awal dalam proses scraping adalah melakukan instalasi library google-play-scraper, yang berfungsi untuk mempermudah pengambilan data ulasan aplikasi dari *Google Playstore*.



Gambar 4. 1 Package Google Playstore

2. Pemanggilan Library

   Untuk proses pengambilan data dengan teknik scraping, library harus diinstal untuk mengambil fitur penting seperti review, username, gambar user, konten, skor, dan lainnya.

```
from google_play_scraper import app
import pandas as pd
import numpy as np
```

Gambar 4. 2 Pemanggilan Library Scrapping

3. Menyalin ID Aplikasi *Getcontact* pada *Google Playstore*

Proses pertama dimulai dengan membuka aplikasi *Getcontact* di *Google Playstore*. Selanjutnya, identifikasi aplikasi yang akan digunakan untuk proses scraping disalin dan dimasukkan ke dalam kode program untuk mengumpulkan data ulasan. Gambar 4.3 menunjukkan bagaimana ID aplikasi ditampilkan.

Gambar 4. 3 Tampilan ID Aplikasi Getcontact melalui Google Playstore

4. Scrapping Ulasan Aplikasi Getcontact

Proses analisis sentimen dimulai dengan menyalin ID aplikasi Getcontact dari Google Play

Store dan memasukkan jumlah ulasan yang akan digunakan. Setelah itu, ID tersebut dimasukkan dan dijalankan ke dalam kode program. Proses scrapping data dapat dilanjutkan, seperti yang ditunjukkan pada Gambar 4.4.

```
# Scrape ulasan terbaru
result, continuation_token = reviews(
    'app.source.getcontact',
    lang='id',  # Bahasa ulasan (Indonesia)
    country='id',  # Lokasi ulasan (Indonesia)
    sort=Sort.NEWEST,  # Mengambil ulasan terbaru
    count=2000,  # Jumlah ulasan yang ingin diambil
    filter_score_with=None  # Ambil semua skor/rating
```

Gambar 4. 4 Tahap Scrapping Data Ulasan

Penggunaan library `google_play_scraper` dalam bahasa pemrograman Python dilakukan untuk mengambil ulasan dari aplikasi yang akan dianalisis sentimennya di Google Play Store. Pada penelitian ini, aplikasi yang diambil ulasannya memiliki ID `'app.source.getcontact'`. Parameter `lang='id'` dan `country='id'` menunjukkan bahwa ulasan yang diambil berasal dari pengguna di Indonesia dan menggunakan Bahasa Indonesia. Pengaturan `sort=Sort.NEWEST` digunakan untuk mengambil ulasan berdasarkan urutan yang paling terbaru. Sedangkan

`count=2000` menunjukkan bahwa jumlah maksimum ulasan yang diambil adalah sebanyak 2000 ulasan. Hasil dari pemanggilan fungsi `reviews()` akan disimpan dalam variabel `result`, dan apabila diperlukan pengambilan data lanjutan, maka digunakan `continuation_token` untuk melanjutkan proses scraping ulasan tambahan.

5. Menyimpan Data Scrapping ke Dataframe

   Library pandas dan numpy digunakan untuk mengolah data yang dikumpulkan dari proses scraping. Pertama, data dibagi menjadi list, dan kemudian diubah menjadi dataframe.

```
data = pd.DataFrame(np.array(result), columns=['review'])
data = data.join(pd.DataFrame(data.pop('review').tolist()))
data
```

Gambar 4. 5 Pengubahan Ulasan menjadi Dataframe

Dataframe dibuat dari hasil ulasan yang telah disimpan dalam variabel `result` dengan menggunakan fungsi `np.array()` dan `pd.DataFrame()`, lalu diberi nama kolom `review`. Selanjutnya, kolom `review` dipindahkan dan dipecah menjadi beberapa kolom baru dengan terlebih dahulu mengubah isinya menjadi list menggunakan `.tolist()`. Hasil list tersebut

kemudian dikonversi kembali menjadi DataFrame baru dan digabungkan ke DataFrame awal menggunakan fungsi `.join()`.

6. Mengambil Data yang dibutuhkan

   Pada tahap ini, dipilih kolom-kolom tertentu dari DataFrame sorteddata untuk membentuk DataFrame baru bernama `df`.

```
df = sorteddata[['content','score','Year','Month','Day']]
df
```

Gambar 4. 6 Pemilihan Kolom Data

Kolom yang dipilih meliputi `'content'`, `'score'`, `'Year'`, `'Month'`, dan `'Day'`.

7. Menyimpan Data Scrapping dalam Format csv

   DataFrame yang telah diproses kemudian disimpan dalam format CSV dengan nama file `"review_getcontact_scrapped.csv"`.

```
df.to_csv('review_getcontact_scrapped.csv')
```

Gambar 4. 7 Menyimpan Data ke csv

Parameter index=False dapat ditambahkan untuk menghindari penyimpanan indeks baris ke dalam file csv.

Berikut hasil data scrapping yang telah disimpan dalam bentuk file csv:

	Column1	content	score	Year	Month	Day
2	1999	Ampas	1	2024	12	31
3	1998	bagus	5	2024	12	31
4	1997	premium semuaa, ga bagus	1	2024	12	31
5	1996	keren	5	2024	12	31
6	1995	males harus bayar padahal dulu gratis	1	2024	12	31
7	1994	Happy new year 2025 I'm so happy 😀	5	2024	12	31
8	1993	ga bagus	5	2024	12	31
9	1992	bagus buat cek nomer² sdm rendah kerja kga, nipu orang via tlpn aja bisanya, tysm min	5	2024	12	31
10	1991	Tidak bisa di gunakan. Harus premium	1	2024	12	31
11	1990	Dipersulit ketika login ulang	1	2024	12	31
12	1989	apa apa harus premium	1	2024	12	31
13	1988	hmm kok berbayar yaa kirain gratis	5	2024	12	31
14	1987	ga banget ya aplikasi nya sekarang, buat cek nomor orang harus premium dulu, ga kaya sebelumnya	1	2024	12	31
15	1986	SEKARANG BERBAYAR JDI MALESSSSS	1	2024	12	31
16	1985	Getcontact saya GK bisa dibukak	1	2024	12	31
17	1984	Aplikasi nya sangat buruk,saya mengetik nomor sendiri tapi yang keluar malah nama orang lain,ga ba	1	2024	12	31
18	1983	Ok	5	2024	12	31
19	1982	sangat membantu	5	2024	12	31
20	1981	knp byr cuy	1	2024	12	31
21	1980	membantu	5	2024	12	31
22	1979	berbayar	1	2024	12	31
23	1978	sangat membantu	5	2024	12	31
24	1977	oke	5	2024	12	31
25	1976	ngelag bgst	1	2024	12	31
26	1975	busuk maksa bayar	1	2024	12	31
27	1974	Kualitas sekarang jelek, tidak makin bagus tapi makin buruk tolong diperbaiki lagi setiap mau lihat ko	2	2024	12	31
28	1973	Udh download mlh gbsa verifikasi	1	2024	12	31

Gambar 4. 8 Hasil Ekspor Data ke csv

B. Pelabelan Data Ulasan

Setelah data ulasan terkumpul, langkah selanjutnya adalah melakukan pemberian label sentimen pada setiap komentar untuk mengelompokkan ulasan ke dalam kategori positif, netral, atau negatif. Penentuan label dilakukan berdasarkan isi dari masing-masing komentar. Label positif diberikan apabila ulasan menunjukkan kepuasan atau pengalaman baik, seperti pada komentar: "Ada telpon masuk tidak dikenal, sangat membantu sekali untuk bisa tau namanya si penelpon." Label netral diberikan jika ulasan bersifat tengah atau tidak secara jelas menunjukkan pujian maupun keluhan, contohnya: "Bagus tapi terbatas." Sedangkan label negatif diberikan untuk ulasan yang mengandung ketidakpuasan atau keluhan, misalnya: "Maaf ya

ratingnya tak turunkan. Saya sudah berlangganan setiap bulan tapi nama-nama orang yang telp tidak pernah muncul. Jadi percuma, buang-buang uang."

Dalam penelitian ini, proses pelabelan dilakukan secara manual dengan bantuan seorang guru kelas 6 di SD Negeri Harapan Mulya 03 yang memiliki kemampuan dalam memahami konteks bahasa dan makna dari setiap ulasan. Meskipun begitu, karena jumlah ulasan yang dianalisis mencapai 2000, proses pelabelan memerlukan waktu yang cukup lama. Untuk menjaga keakuratan hasil, setiap komentar perlu dibaca dan dievaluasi satu per satu, terutama pada ulasan yang mengandung kalimat ambigu atau mencampurkan pujian dan keluhan. Berikut ini merupakan hasil proses pelabelan sentimen yang telah dilakukan.

	A	B	C	D	E	F
1	Column1	content	score	Year	Month	Day
2	1999	Ampas	negatif	2024	12	31
3	1998	bagus	positif	2024	12	31
4	1997	premium semuaa, ga bagus	negatif	2024	12	31
5	1996	keren	positif	2024	12	31
6	1995	males harus bayar padahal dulu gratis	negatif	2024	12	31
7	1994	Happy new year 2025 I'm so happy 😁	positif	2024	12	31
8	1993	ga bagus	negatif	2024	12	31
9	1992	bagus buat cek nomer² sdm rendah kerja kga, nipu orang via tlpn aja bisanya, tysm min	positif	2024	12	31
10	1991	Tidak bisa di gunakan. Harus premium	negatif	2024	12	31
11	1990	Dipersulit ketika login ulang	negatif	2024	12	31
12	1989	apa apa harus premium	negatif	2024	12	31
13	1988	hmm kok berbayar yaa kirain gratis	negatif	2024	12	31
14	1987	ga banget ya aplikasi nya sekarang, buat cek nomor orang harus premium dulu, ga kaya sebelumnya	negatif	2024	12	31
15	1986	SEKARANG BERBAYAR JDI MALESSSSS	negatif	2024	12	31
16	1985	Getcontact saya GK bisa dibukak	negatif	2024	12	31
17	1984	Aplikasi nya sangat buruk,saya mengetik nomor sendiri tapi yang keluar malah nama orang lain,ga be	negatif	2024	12	31
18	1983	Ok	netral	2024	12	31
19	1982	sangat membantu	positif	2024	12	31
20	1981	knp byr cuy	negatif	2024	12	31
21	1980	membantu	positif	2024	12	31
22	1979	berbayar	netral	2024	12	31
23	1978	sangat membantu	positif	2024	12	31
24	1977	oke	netral	2024	12	31
25	1976	ngelag bgst	negatif	2024	12	31
26	1975	busuk maksa bayar	negatif	2024	12	31
27	1974	Kualitas sekarang jelek, tidak makin bagus tapi makin buruk tolong diperbaiki lagi setiap mau lihat ko	negatif	2024	12	31
28	1973	Udh download mlh gbsa verifikasi	negatif	2024	12	31

Gambar 4. 9 Pelabelan Data

C. *Text Preprocessing*

Sebelum data dianalisis secara lebih mendalam, diperlukan tahap awal berupa preprocessing atau pra-pemrosesan data. Tahapan ini bertujuan untuk membersihkan dan merapikan data agar menjadi lebih terstruktur serta siap untuk dianalisis. Dengan melakukan preprocessing, kualitas data akan meningkat sehingga mendukung hasil penelitian yang lebih akurat dan terpercaya.

Adapun tahapan preprocessing yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. *Cleaning*

   Pembersihan data (cleaning data) dilakukan untuk menghilangkan elemen-elemen yang tidak diperlukan dalam ulasan, seperti emoji, tanda baca, tautan (URL), hashtag, dan karakter tidak relevan lainnya. Berikut ini merupakan kode program yang digunakan dalam proses cleaning data:

Gambar 4. 10 Proses Cleaning

Pada tahap pembersihan data (cleaning), dilakukan proses pengolahan langsung pada kolom *content* untuk menghilangkan elemen-elemen yang tidak relevan atau mengganggu dalam analisis. Proses ini dilakukan dengan menggunakan metode chaining pada fungsi `str.replace()` dari library pandas. Beberapa elemen yang dihapus meliputi URL atau tautan `(https\S+)`, mention pengguna seperti @username `(@\S+)`, hashtag `(#\S+)`, singkatan dengan tanda kutip tunggal `('\w+)`, serta tanda baca dan karakter non-alfabet dengan pola `([^\w\s])`. Selain itu, karakter angka tertentu seperti angka '2' yang muncul setelah spasi juga dihapus menggunakan pola `(\s(2))`.

2. *Case Folding*

Tahapan ini bertujuan untuk menyamakan bentuk penulisan kata agar lebih konsisten, yaitu

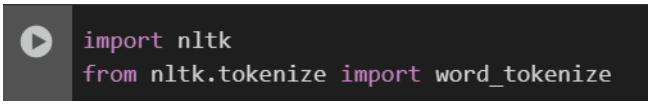
dengan mengubah seluruh huruf dalam kata, kalimat, atau teks menjadi huruf kecil (lowercase).

Gambar 4. 11 Proses Case Folding

Kode `` `df['content'] = df['content'].str.lower()` `` digunakan untuk mengonversi seluruh teks dalam kolom *content* menjadi huruf kecil. Proses ini dilakukan agar tidak terjadi perbedaan penulisan antara huruf kapital dan huruf kecil, sehingga data menjadi lebih konsisten dan seragam untuk keperluan analisis lebih lanjut.

3. *Tokenizing*

Tahapan ini dilakukan dengan memanfaatkan library nltk yang tersedia dalam bahasa pemrograman Python. Pada tahap ini digunakan fungsi `` `word_tokenize` `` dari modul `nltk.tokenize` untuk memecah teks ulasan menjadi bagian-bagian kata (token) yang lebih kecil.



Gambar 4. 12 Menginstall Library nltk

Berikut ini adalah source code yang digunakan untuk melakukan proses tokenizing.

```
from nltk.tokenize import RegexpTokenizer
regexp = RegexpTokenizer('\w+')
df['content_token']=df['content'].apply(regexp.tokenize)
df.head(3)
```

Gambar 4. 13 Proses Tokenizing

Potongan kode di atas menggunakan `RegexpTokenizer` dari library `nltk` untuk melakukan tokenisasi teks ulasan. Tokenizer ini dirancang untuk mengekstrak hanya karakter alfanumerik (`\w+`), sehingga tanda baca dan karakter khusus lainnya akan diabaikan. Dengan menerapkan fungsi `regexp.tokenize` pada kolom `content`, setiap kalimat dalam ulasan akan diubah menjadi daftar kata-kata terpisah yang disimpan dalam kolom baru bernama `content_token`.

Tokenisasi merupakan langkah awal yang penting dalam Natural Language Processing (NLP), karena berfungsi sebagai dasar untuk tugas-tugas lanjutan seperti analisis sentimen, klasifikasi teks, dan pemodelan bahasa.

Berikut hasil dari code program cleaning data, case folding dan tokenizing.

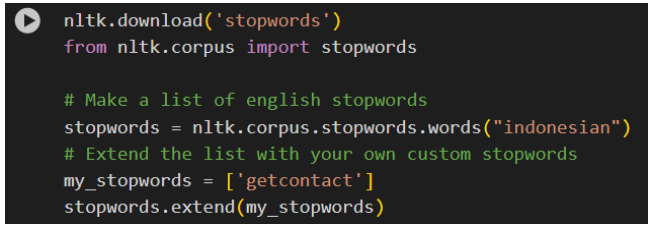
	Column1	content	score	Year	Month	Day	sentiment	content_token
0	1999	ampas	negatif	2024	12	31	-1	[ampas]
1	1998	bagus	positif	2024	12	31	1	[bagus]
2	1997	premium semuaa ga bagus	negatif	2024	12	31	-1	[premium, semuaa, ga, bagus]
3	1996	keren	positif	2024	12	31	1	[keren]
4	1995	males harus bayar padahal dulu gratis	negatif	2024	12	31	-1	[males, harus, bayar, padahal, dulu, gratis]
5	1994	happy new year 025 i so happy	positif	2024	12	31	1	[happy, new, year, 025, i, so, happy]
6	1993	ga bagus	negatif	2024	12	31	-1	[ga, bagus]
7	1992	bagus buat cek nomer³ sdm rendah kerja kga ni...	positif	2024	12	31	1	[bagus, buat, cek, nomer³, sdm, rendah, kerja,...
8	1991	tidak bisa di gunakan harus premium	negatif	2024	12	31	-1	[tidak, bisa, di, gunakan, harus, premium]
9	1990	dipersulit ketika login ulang	negatif	2024	12	31	-1	[dipersulit, ketika, login, ulang]

Gambar 4. 14 Hasil Tokenizing

4. *Stopword Removal*

Tahapan berikutnya adalah stopword removal, yaitu proses menghapus kata-kata umum yang sering muncul dalam teks tetapi tidak memberikan makna penting dalam analisis, seperti "yang", "dan", "di", serta kata-kata umum lainnya.

Pada tahap ini, library NLTK yang telah diinstal sebelumnya kembali dimanfaatkan untuk membantu mengidentifikasi dan menghapus kata-kata tersebut. Proses ini bertujuan agar analisis teks menjadi lebih fokus pada kata-kata yang memiliki makna signifikan. Berikut merupakan kode yang digunakan untuk menjalankan proses stopword removal ini:



Gambar 4. 15 Proses Stopword Removal

kode di atas mengimpor modul `nltk.corpus` untuk mengakses daftar kata umum (stopwords). Langkah awal dilakukan dengan mengunduh daftar stopwords menggunakan `nltk.download('stopwords')`. Selanjutnya, daftar stopwords dalam bahasa Indonesia dimuat ke dalam variabel stopwords menggunakan `stopwords.words("indonesian")`. Daftar ini berisi kata-kata umum yang biasanya tidak memiliki makna signifikan dalam analisis. Selain itu, daftar stopwords diperluas dengan menambahkan kata kustom, dalam hal ini kata "getcontact", ke dalam daftar menggunakan `stopwords.extend(my_stopwords)`. Berikut merupakan hasil dari penerapan kode stopword removal tersebut.

	Column1	content	score	Year	Month	Day	sentiment	content_token
0	1999	ampas	negatif	2024	12	31	-1	[ampas]
1	1998	bagus	positif	2024	12	31	1	[bagus]
2	1997	premium semuaa ga bagus	negatif	2024	12	31	-1	[premium, semuaa, ga, bagus]
3	1996	keren	positif	2024	12	31	1	[keren]
4	1995	males harus bayar padahal dulu gratis	negatif	2024	12	31	-1	[males, bayar, gratis]
5	1994	happy new year 025 i so happy	positif	2024	12	31	1	[happy, new, year, 025, i, so, happy]
6	1993	ga bagus	negatif	2024	12	31	-1	[ga, bagus]
7	1992	bagus buat cek nomer⁴ sdm rendah kerja kga ni...	positif	2024	12	31	1	[bagus, cek, nomer⁴, sdm, rendah, kerja, kga, ...
8	1991	tidak bisa di gunakan harus premium	negatif	2024	12	31	-1	[premium]
9	1990	dipersulit ketika login ulang	negatif	2024	12	31	-1	[dipersulit, login, ulang]

Gambar 4. 16 Hasil Stopword Removal

5. *Stemming*

Untuk melakukan stemming pada penelitian ini, digunakan library Sastrawi yang tersedia dalam bahasa pemrograman Python. Berikut ditampilkan gambar proses instalasi library Sastrawi.

```
!pip install Sastrawi
```

Gambar 4. 17 Menginstall Library Sastrawi

Berikut adalah kode program yang digunakan untuk melakukan proses stemming.

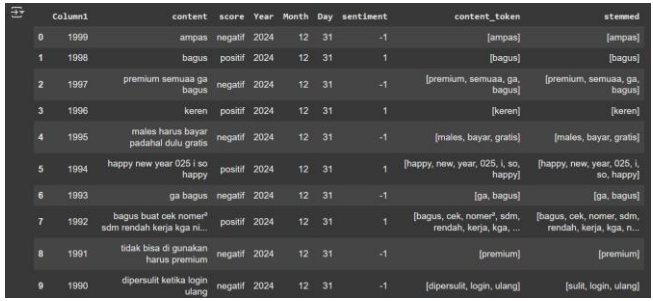
```
# import Sastrawi package
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()
df['stemmed'] = df['content_token'].apply(lambda x: [stemmer.stem(y) for y in x])
```

Gambar 4. 18 Proses Stemming

Pertama, library Sastrawi diimpor dan objek stemmer dibuat menggunakan `StemmerFactory`

dari pustaka tersebut. Setelah objek stemmer siap, proses stemming dilakukan pada kolom `content_token` di dalam DataFrame. Setiap elemen dalam kolom tersebut, yang berupa daftar token (kata-kata), akan diproses satu per satu menggunakan fungsi `stem()` dari objek stemmer. Hasil stemming dari setiap kata kemudian disimpan ke dalam kolom baru bernama stemmed.

Berikut hasil stemming data yang telah dilakukan.



	Column1	content	score	Year	Month	Day	sentiment	content_token	stemmed
0	1999	ampas	negatif	2024	12	31	-1	[ampas]	[ampas]
1	1998	bagus	positif	2024	12	31	1	[bagus]	[bagus]
2	1997	premium semuaa ga bagus	negatif	2024	12	31	-1	[premium, semuaa, ga, bagus]	[premium, semuaa, ga, bagus]
3	1996	keren	positif	2024	12	31	1	[keren]	[keren]
4	1995	males harus bayar padahal dulu gratis	negatif	2024	12	31	-1	[males, bayar, gratis]	[males, bayar, gratis]
5	1994	happy new year 025 i so happy	positif	2024	12	31	1	[happy, new, year, 025, i, so, happy]	[happy, new, year, 025, i, so, happy]
6	1993	ga bagus	negatif	2024	12	31	-1	[ga, bagus]	[ga, bagus]
7	1992	bagus buat cek nomer² sdm rendah kerja kga ni...	positif	2024	12	31	1	[bagus, cek, nomer², sdm, rendah, kerja, kga, ...	[bagus, cek, nomer, sdm, rendah, kerja, kga, n...
8	1991	tidak bisa di gunakan harus premium	negatif	2024	12	31	-1	[premium]	[premium]
9	1990	dipersulit ketika login ulang	negatif	2024	12	31	-1	[dipersulit, login, ulang]	[sulit, login, ulang]

Gambar 4. 19 Hasil Stemming

Setelah melalui serangkaian tahapan preprocessing, dataset siap untuk analisis lebih lanjut. Tahapan preprocessing ini meliputi pembersihan teks dari elemen yang tidak relevan seperti URL, tanda baca, karakter khusus, dan emoji. Proses case folding dilakukan untuk mengubah teks menjadi huruf kecil, menghindari perbedaan antara

huruf besar dan kecil. Selanjutnya, teks ditokenisasi menjadi kata-kata, diikuti dengan penghapusan stopwords yang tidak memberikan kontribusi signifikan. Terakhir, stemming dilakukan untuk mengubah kata ke bentuk dasarnya, memudahkan pencarian, dan mengurangi variasi kata.

Dengan demikian, tahap preprocessing data menjadi langkah krusial dalam mempersiapkan data untuk analisis dan pemodelan teks.

D. Ekstrasi Fitur

Dalam tahapan ekstraksi fitur, dibutuhkan *library* yang mendukung proses pengolahan data, seperti library *sklearn* (*scikit-learn*) dan *imblearn*. *Library* ini memiliki peran penting dalam membantu proses *preprocessing* data serta pelatihan model yang dibutuhkan dalam *machine learning* dan data *science*. Pada tahap ekstraksi fitur ini, digunakan beberapa kelas dari *sklearn*, yaitu *TfidfVectorizer* dari modul `sklearn.feature_extraction.text` untuk mengubah data teks menjadi representasi numerik berbasis TF-IDF, serta `train_test_split` dari `sklearn.model_selection` untuk membagi data menjadi data latih dan data uji. Selain itu, untuk mengatasi ketidakseimbangan data, digunakan metode

SMOTE (*Synthetic Minority Oversampling Technique*) yang diimpor dari *library* `imblearn.over_sampling`.

Dalam tahap ekstraksi fitur, sistem melakukan proses transformasi data teks menjadi representasi numerik menggunakan metode TF-IDF (*Term Frequency–Inverse Document Frequency*). Proses ini dilakukan dengan mengimpor kelas *TfidfVectorizer* dari library `sklearn.feature_extraction.text`. Dalam implementasinya, data teks diambil dari kolom '*text_string*' dan label sentimen dari kolom '*sentiment*' pada dataset. Selanjutnya, teks pada kolom tersebut diubah menjadi vektor TF-IDF menggunakan fungsi `fit_transform` dari objek *TfidfVectorizer*. Proses ini menghasilkan fitur numerik yang siap digunakan untuk proses pelatihan model klasifikasi. Tampilan pembobotan TF-IDF dapat dilihat pada Gambar 4.20.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Ambil fitur dan label dari dataframe
X = df['text_string']
y = df['sentiment']

# Inisialisasi dan transformasi teks dengan TF-IDF
tfidf = TfidfVectorizer()
X_final = tfidf.fit_transform(X)
```

Gambar 4. 20 Proses Pembobotan TFIDF

Sebelum dilakukan pembagian data, terlebih dahulu dilakukan proses penanganan data tidak seimbang menggunakan metode SMOTE (*Synthetic Minority Oversampling Technique*). SMOTE berfungsi untuk menyeimbangkan jumlah sampel antara kelas mayoritas dan kelas minoritas dengan cara menghasilkan data sintetis dari kelas minoritas.

Setelah data berhasil diseimbangkan, langkah selanjutnya adalah proses pembagian data menjadi data latih (training) dan data uji (testing) menggunakan fungsi `train_test_split`. Pada proses ini, data uji dialokasikan sebesar 20% dari total data, sedangkan 80% sisanya digunakan sebagai data latih. Pembagian ini bertujuan untuk mempermudah proses pelatihan dan evaluasi model klasifikasi. Proses pembagian data setelah penanganan ketidakseimbangan ditunjukkan pada Gambar 4.21.

```
# Handling imbalanced using SMOTE
from imblearn.over_sampling import SMOTE # Handling Imbalanced
smote = SMOTE()
x_sm,y_sm = smote.fit_resample(X_final,y)
X_train , X_test , y_train , y_test = train_test_split(x_sm , y_sm , test_size=0.2,random_state=42)
```

Gambar 4. 21 Proses Proses split data 80:20 setelah SMOTE

Hasil dari tahapan pembobotan menggunakan TF-IDF menghasilkan representasi dokumen dalam bentuk matriks, di mana setiap baris merepresentasikan

satu dokumen dan setiap kolom merepresentasikan kata-kata unik yang muncul di seluruh korpus teks. Nilai dalam matriks tersebut menunjukkan skor TF-IDF dari masing-masing kata dalam dokumen tertentu.

Untuk menampilkan hasil pembobotan ini, digunakan fungsi `get_feature_names_out()` dari objek TfidfVectorizer untuk mengambil daftar kata yang terbentuk. Selanjutnya, dilakukan iterasi terhadap beberapa dokumen untuk melihat kata-kata yang memiliki skor TF-IDF tertinggi pada masing-masing dokumen. Proses ini ditunjukkan pada Gambar 4.22.

```
feature_names = tfidf.get_feature_names_out()
max_docs = 5

for doc_index in range(min(max_docs, x_sm.shape[0])):
    doc_tfidf = x_sm[doc_index]
    nonzero_indices = doc_tfidf.nonzero()[1]

    tfidf_scores = [(feature_names[idx], doc_tfidf[0, idx]) for idx in nonzero_indices]
    tfidf_scores.sort(key=lambda x: x[1], reverse=True)

    print(f"\nDokumen {doc_index}:")
    for word, score in tfidf_scores:
        print(f"  {word}: {score:.4f}")
```

```
Dokumen 0:
  ampas: 1.0000

Dokumen 1:
  bagus: 1.0000

Dokumen 2:
  semuaa: 0.8756
  bagus: 0.3781
  premium: 0.3006

Dokumen 3:
  keren: 1.0000

Dokumen 4:
  males: 0.7405
  gratis: 0.5644
  bayar: 0.3648
```

Gambar 4. 22 Hasil Pembobotan TF-IDF

Berikut adalah contoh data dari penelitian ini yang menggunakan tiga komentar untuk dilakukan pembobotan TF-IDF secara manual:

(Doc 1) = " males harus bayar padahal dulu gratis"

(Doc 2) = "Dipersulit ketika login ulang"

(Doc 3) = "hmm kok berbayar yaa kirain gratis"

Kemudian, hasil preprocessing dari komentar di atas adalah sebagai berikut:

(Doc 1) = ['males', 'bayar', 'gratis']

(Doc 2) = ['sulit', 'login', 'ulang']

(Doc 3) = ['hmm', 'bayar', 'yaa', 'kirain', 'gratis']

Langkah selanjutnya adalah melakukan perhitungan menggunakan metode TF-IDF untuk membentuk representasi vektor kata yang akan diberikan bobot. Dalam proses ini, terdapat dua komponen utama, yaitu Term Frequency (TF) dan Inverse Document Frequency (IDF). TF berfungsi untuk menghitung frekuensi kemunculan kata dalam setiap dokumen, sedangkan IDF digunakan untuk mengurangi bobot dari kata-kata yang terlalu sering muncul di banyak dokumen, karena dianggap kurang memberikan informasi yang spesifik. Perhitungan TF-IDF dimulai dengan menghitung nilai TF terlebih dahulu. Contoh

perhitungan TF secara manual ditampilkan pada Tabel 4.1.

Tabel 4. 1 Contoh Perhitungan TF

Token	TF		
	Doc 1	Doc 2	Doc 3
males	1	0	0
bayar	1	0	1
gratis	1	0	1
sulit	0	1	0
login	0	1	0
ulang	0	1	0
hmm	0	0	1
yaa	0	0	1
kirain	0	0	1

Berdasarkan tabel sebelumnya, nilai Document Frequency (DF) telah diperoleh. Sebagai contoh, jumlah dokumen yang digunakan dalam tabel adalah tiga komentar, sehingga total dokumen (D) = 3. Langkah berikutnya adalah menghitung nilai Inverse Document Frequency (IDF) dan TF-IDF menggunakan rumus yang telah dijelaskan pada bab sebelumnya. Hasil

perhitungan tersebut kemudian akan diterapkan dan ditampilkan dalam Tabel 4.2.

Tabel 4. 2 Contoh Perhitungan TF dan TFIDF

Token	Df	D/Df (D=3)	IDF (log(D/Df))	IDF +1	TF*IDF		
					D1	D2	D3
males	1	3	log(3)=0.477	1.477	1.477	0	0
bayar	2	1.5	log(1.5)=0.176	1.176	1.176	0	1.176
gratis	2	1.5	log(1.5)=0.176	1.176	1.176	0	1.176
sulit	1	3	log(3)=0.477	1.477	0	1.477	0
login	1	3	log(3)=0.477	1.477	0	1.477	0
ulang	1	3	log(3)=0.477	1.477	0	1.477	0
hmm	1	3	log(3)=0.477	1.477	0	0	1.477
yaa	1	3	log(3)=0.477	1.477	0	0	1.477
kirain	1	3	log(3)=0.477	1.477	0	0	1.477

Dengan demikian, setelah dilakukan perhitungan bobot TF-IDF secara manual terhadap ketiga komentar yang telah diproses, hasilnya dapat direpresentasikan dalam bentuk array sebagai berikut:

```
Array([
 [1.176, 0, 1.176, 0, 0, 0, 0, 0, 0],
 [0, 1.477, 0, 1.477, 1.477, 0, 0, 0, 0],
 [0, 0, 1.176, 0, 0, 1.477, 1.477, 1.477, 1.477]
])
```

Berdasarkan hasil perhitungan TF-IDF di atas, setiap dari ketiga dokumen telah berhasil direpresentasikan dalam bentuk vektor. Setiap kolom pada hasil tersebut mencerminkan nilai bobot dari masing-masing kata yang muncul dalam keseluruhan teks, sehingga menggambarkan kontribusi setiap kata terhadap dokumen yang bersangkutan.

E. Klasifikasi *Naive Bayes* dan *Random Forest*

Setelah melalui tahapan *preprocessing* dan ekstraksi fitur, dataset dilanjutkan ke tahap pembelajaran (*learning*) dengan menggunakan dua algoritma klasifikasi, yaitu *Naive Bayes* dan *Random Forest*. Pada tahap ini, sistem dilatih menggunakan 80% data latih dari total data yang tersedia, sedangkan 20% sisanya digunakan sebagai data uji. Tujuannya adalah untuk mengukur performa masing-masing algoritma dalam mengklasifikasikan sentimen pengguna terhadap aplikasi. Proses pelatihan dan pengujian dilakukan dengan bantuan library *scikit-learn* (*sklearn*), yang juga digunakan pada tahap ekstraksi fitur sebelumnya.

Adapun library *scikit-learn* (*sklearn*) yang digunakan dalam proses klasifikasi pada penelitian ini meliputi MultinomialNB, RandomForestClassifier, *accuracy_score*, *precision_score*, *recall_score*, *f1_score*,

classification_report, dan *confusion_matrix* yang digunakan untuk mengevaluasi kinerja model klasifikasi.

Langkah awal dalam proses klasifikasi dimulai dengan menyiapkan *library* `*sklearn*`. Karena Google Colab sudah menyediakan *library* ini secara bawaan, maka tidak diperlukan proses instalasi tambahan. Pengguna cukup langsung melakukan *import* terhadap beberapa kelas yang diperlukan. Adapun *source code* untuk mengimpor *library sklearn* pada tahap klasifikasi ditampilkan pada Gambar 4.37.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

Gambar 4. 23 Import Library Sklearn pada proses klasifikasi

Setelah tahap sebelumnya selesai, proses selanjutnya adalah melakukan klasifikasi menggunakan metode *Naive Bayes* dan *Random Forest*. Berikut adalah *source code* untuk proses klasifikasi ditampilkan pada Gambar 4.24.

```python
# Convert sparse matrices to dense arrays (if applicable)
X_train_dense = X_train.toarray()
X_test_dense = X_test.toarray()

# Initialize models without random_state
nb_model = MultinomialNB()
rf_model = RandomForestClassifier()

# Train models
nb_model.fit(X_train_dense, y_train)
rf_model.fit(X_train_dense, y_train)

# Predictions
nb_pred = nb_model.predict(X_test_dense)
rf_pred = rf_model.predict(X_test_dense)

# Evaluation function
def evaluate_model(y_test, predicted, model_name):
    print(f"=== Evaluasi {model_name} ===")
    print("Akurasi:", accuracy_score(y_test, predicted))
    print("Presisi:", precision_score(y_test, predicted, average="weighted"))
    print("Recall:", recall_score(y_test, predicted, average="weighted"))
    print("F1-score:", f1_score(y_test, predicted, average="weighted"))
    print(f'Confusion matrix:\n{confusion_matrix(y_test, predicted)}')
    print('=====================================================\n')
    print(classification_report(y_test, predicted, zero_division=0))
    print("\n")

# Evaluate each model
evaluate_model(y_test, nb_pred, "Multinomial Naive Bayes")
evaluate_model(y_test, rf_pred, "Random Forest")
```

Gambar 4. 24 Klasifikasi Metode Random Forest dan Naive Bayes

Proses dimulai dengan mengubah data latih (`X_train`) dan data uji (`X_test`), yang sebelumnya masih berbentuk matriks sparse hasil dari transformasi TF-IDF, menjadi array berdimensi penuh (dense array) menggunakan fungsi `.toarray()`. Langkah ini diperlukan karena beberapa algoritma pembelajaran mesin, seperti *Naive Bayes* dan *Random Forest*, lebih optimal saat menggunakan representasi data dalam bentuk array. Setelah itu, dilakukan inisialisasi dua

model klasifikasi, yaitu `MultinomialNB()` untuk algoritma *Naive Bayes* dan `RandomForestClassifier()` untuk algoritma *Random Forest*, tanpa menetapkan nilai `random_state`.

Kedua model kemudian dilatih menggunakan data latih (`X_train_dense`) dan label yang sesuai (`y_train`) dengan metode `.fit()`. Setelah proses pelatihan selesai, digunakan metode `.predict()` untuk melakukan prediksi pada data uji (`X_test_dense`), sehingga diperoleh hasil prediksi yang disimpan dalam variabel `nb_pred` untuk model *Naive Bayes* dan `rf_pred` untuk model *Random Forest*. Untuk mengevaluasi kinerja masing-masing model, digunakan fungsi `evaluate_model()` yang telah didefinisikan secara manual. Fungsi ini menerima tiga parameter, yaitu label asli (`y_test`), hasil prediksi (`predicted`), dan nama model yang dievaluasi (`model_name`).

F. Uji Model

Pada tahap pengujian model, dilakukan evaluasi untuk mengetahui sejauh mana tingkat akurasi model dalam melakukan proses klasifikasi. Hasil dari proses ini ditampilkan melalui *multiclass confusion matrix*, yang menggambarkan perbandingan antara label prediksi

(*predicted class*) dengan label sebenarnya (*true class*). Dalam penelitian ini digunakan *confusion matrix* dengan format multi-kelas karena klasifikasi dilakukan terhadap tiga kategori sentimen, yaitu negatif, netral, dan positif. Oleh karena itu, confusion matrix yang digunakan berbentuk matriks 3 x 3, yang secara rinci disajikan pada Tabel 4.3.

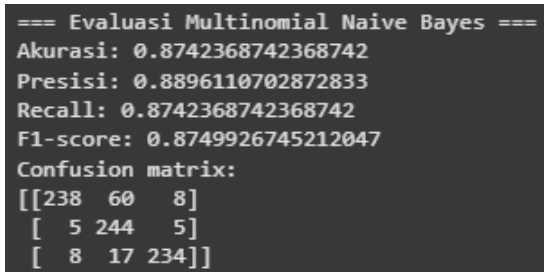
Tabel 4. 3 Multiclass Confusion Matrix 3x3

		True Class		
		Negative	*Netral*	*Positive*
Predicted Class	*Negative*	*T Neg*	*F NegNet*	*F NegPos*
	Netral	*F NetNeg*	*T Net*	*F NetPos*
	Positif	*F PosNeg*	*F PosNet*	*T Pos*

Untuk menilai sejauh mana tingkat ketepatan model dalam melakukan klasifikasi, diperlukan perhitungan nilai akurasi. Akurasi ini diperoleh dengan menghitung jumlah prediksi yang sesuai dengan label sebenarnya, kemudian dibagi dengan total keseluruhan data yang digunakan pada proses pengujian. Rumus untuk menghitung nilai akurasi tersebut dapat dilihat pada Persamaan 4.1.

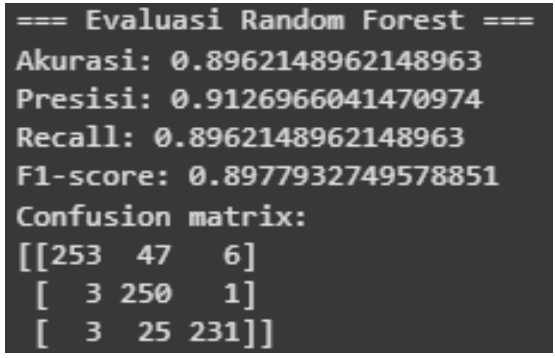
$$Akurasi = \frac{TPos+TNeg+TNet}{Total\ daya\ yang\ diuji} \qquad (4.1)$$

Setelah melalui tahapan pengujian model menggunakan algoritma *Naive Bayes*, diperoleh hasil berupa nilai akurasi serta *multiclass confusion matrix* berukuran 3 x 3 yang disajikan pada Gambar 4.25.



Gambar 4. 25 Hasil Uji Model Naive Bayes

Selanjutnya, dilakukan tahapan pengujian model menggunakan algoritma *Random Forest*, diperoleh hasil berupa nilai akurasi serta *multiclass confusion matrix* berukuran 3 x 3 yang disajikan pada Gambar 4.26.



Gambar 4. 26 Hasil Uji Model Random Forest

Berdasarkan hasil pengujian model yang ditampilkan melalui multiclass confusion matrix berukuran 3x3, diperoleh nilai akurasi sebesar 0.874 atau 87,4% untuk algoritma Multinomial *Naive Bayes*, dan 0.896 atau 89,6% untuk algoritma *Random Forest*. Hasil ini menunjukkan bahwa model *Random Forest* memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan *Naive Bayes* dalam melakukan klasifikasi sentimen pada data yang digunakan. Tahapan selanjutnya adalah melakukan evaluasi lebih lanjut terhadap kinerja kedua model dengan melihat metrik lain seperti presisi, *recall*, dan *F1-Score* untuk mendapatkan gambaran yang lebih menyeluruh mengenai efektivitas model dalam mengklasifikasikan data ke dalam tiga kelas, yaitu negatif, netral, dan positif.

G. Evaluasi Model

Setelah dilakukan proses pengujian model, tahap selanjutnya adalah evaluasi model untuk mengetahui kinerja dari model yang telah dibangun. Dalam penelitian ini, evaluasi dilakukan dengan menghitung sejumlah metrik performa, yaitu akurasi, presisi, *recall*, dan *F1-Score*, yang diperoleh berdasarkan hasil multiclass confusion matrix berukuran 3x3. Sebelumnya, pada tahap pengujian model telah

diperoleh hasil confusion matrix 3x3 yang ditampilkan pada Tabel 4.4.

Tabel 4. 4 Hasil Multiclass Confusion Matrix Naive Bayes

		True Class		
		Negative	*Netral*	*Positive*
Predicted Class	*Negative*	*238*	*60*	*8*
	Netral	*5*	*244*	*5*
	Positif	*8*	*17*	*234*

Selanjutnya, pengujian model dengan algoritma *Random Forest* juga menghasilkan confusion matrix 3x3 yang disajikan secara terpisah pada Tabel 4.5.

Tabel 4. 5 Hasil Multiclass Confusion Matrix Random Forest

		True Class		
		Negative	*Netral*	*Positive*
Predicted Class	*Negative*	*253*	*47*	*6*
	Netral	*3*	*250*	*1*
	Positif	*3*	*25*	*231*

Selanjutnya, peneliti melakukan perhitungan secara manual untuk memperoleh nilai akurasi dari model *Naive Bayes*, berdasarkan data pada tabel *multiclass confusion matrix* yang telah disajikan

sebelumnya. Dengan demikian, hasil perhitungan akurasi secara manual dapat dijelaskan sebagai berikut:

$$Akurasi = \frac{TPos+TNeg+TNet}{Total\ daya\ yang\ diuji} \text{x } 100\%$$

$$Akurasi = \frac{238+244+234}{238+60+8+5+244+5+8+17+234} \text{x} 100\%$$

$$Akurasi = \frac{716}{819} \text{x} 100\%$$

$$Akurasi = 0{,}874 \text{x} 100\%$$

$$Akurasi = 87{,}4\%$$

Setelah itu, perhitungan manual juga dilakukan untuk menentukan nilai akurasi dari model *Random Forest*. Perhitungan ini didasarkan pada data multiclass confusion matrix yang telah ditampilkan sebelumnya. Hasil dari perhitungan akurasi secara manual untuk model *Random Forest* dapat dijelaskan sebagai berikut:

$$Akurasi = \frac{TPos+TNeg+TNet}{Total\ daya\ yang\ diuji} \text{x } 100\%$$

$$Akurasi = \frac{253+250+231}{253+47+6+3+250+1+3+25+231} \text{x} 100\%$$

$$Akurasi = \frac{734}{819} \text{x} 100\%$$

$$Akurasi = 0{,}896 \text{x} 100\%$$

$$Akurasi = 89{,}6\%$$

Untuk menghitung metrik performa model lainnya, seperti nilai presisi, *recall*, dan *F1-Score*, diperlukan penentuan nilai *true positive* (TP), *true*

negative (TN), *false positive* (FP), dan *false negative* (FN) untuk masing-masing kelas yang ada pada *multiclass confusion matrix* berukuran 3x3. Namun, pada *confusion matrix* dengan ukuran 3x3, proses ini cukup kompleks karena melibatkan lebih dari dua kelas. Oleh karena itu, untuk mempermudah dalam menghitung metrik tersebut, dilakukan konversi matriks menjadi bentuk 2x2 untuk setiap kelas secara terpisah. Hasil konversi *Naive Bayes* ditampilkan sebagai berikut:

- Kelas Negatif *Naive Bayes*

Tabel 4. 6 Perhitungan Nilai Kelas Negatif Naive Bayes

True/Pred	Negatif	Bukan
Negatif	TP=238	FN=(60+8)=68
Bukan	FP=(5+8)=13	TN=(244+5+17+234)=500

- Kelas Netral *Naive Bayes*

Tabel 4. 7 Perhitungan Nilai Kelas Netral Naive Bayes

True/Pred	Netral	Bukan
Netral	TP=244	FN=(5+5)=10
Bukan	FP=(60+17)=77	TN=(238+8+8+234)=488

- Kelas Positif *Naive Bayes*

Tabel 4. 8 Perhitungan Nilai Kelas Positif Naive Bayes

True/Pred	Positif	Bukan
Positif	TP=234	FN=(8+17)=25
Bukan	FP=(8+5)=13	TN=(238+60+5+244)=547

Hasil konversi *Random Forest* ditampilkan sebagai berikut:

- Kelas Negatif *Random Forest*

Tabel 4. 9 Perhitungan Nilai Kelas Negatif Random Forest

True/Pred	Negatif	Bukan
Negatif	TP=253	FN=(47+6)=53
Bukan	FP=(3+3)=6	TN=(250+1+25+231)=507

- Kelas Netral *Random Forest*

Tabel 4. 10 Perhitungan Nilai Kelas Netral Random Forest

True/Pred	Netral	Bukan
Netral	TP=250	FN=(3+1)=4
Bukan	FP=(47+25)=72	TN=(253+6+3+231)=493

- Kelas Positif *Random Forest*

Tabel 4. 11 Perhitungan Nilai Kelas Positif Random Forest

True/Pred	Positif	Bukan

Positif	TP=231	FN=(25+3)=28
Bukan	FP=(6+1)=7	TN=(253+47+3+250)=553

Setelah diperoleh nilai *true positive*, *true negative*, *false positive*, dan *false negative*, maka berdasarkan rumus yang telah dijelaskan pada bab sebelumnya, nilai *precision*, *recall*, dan *F1-Score* dari hasil *multiclass confusion matrix* pada algoritma *Naive Bayes* dapat dirangkum ke dalam Tabel 4.12 sebagai berikut:

Tabel 4. 12 Hasil Perhitungan Performa Naive Bayes

Class	True Positive (TP)	False Positive (FP)	False Negative (FN)	*Precision*	*Recall*	*F1-Score*
Negative	238	13	68	95%	78%	85%
Netral	244	77	10	76%	96%	85%
Positive	234	13	25	95%	90%	92%

Berdasarkan Tabel 4.12, nilai *precision* untuk kelas negatif adalah sebesar 95%, kelas netral sebesar 76%, dan kelas positif sebesar 95%. Sementara itu, nilai *recall* pada kelas negatif mencapai 78%, kelas netral 96%, dan kelas positif sebesar 90%. Adapun nilai *F1-Score* diperoleh masing-masing sebesar 85% untuk

kelas negatif, 85% untuk kelas netral, dan 92% untuk kelas positif.

Hasil *multiclass confusion matrix* pada algoritma *Random Forest* selanjutnya dapat dirangkum ke dalam Tabel 4.13.

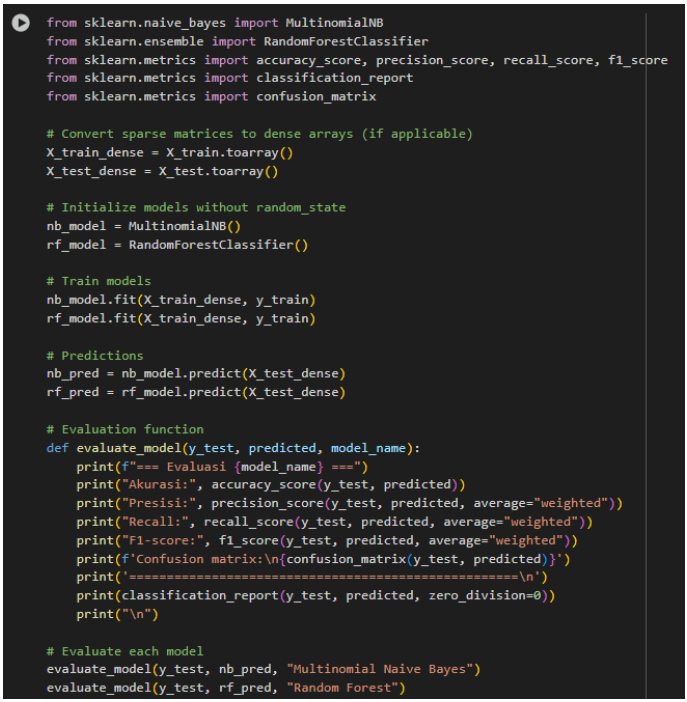
Tabel 4. 13 Hasil Perhitungan Performa Random Random Forest

Class	True Positive (TP)	False Positive (FP)	False Negative (FN)	*Precision*	*Recall*	*F1-Score*
Negative	253	6	53	98%	83%	90%
Netral	250	72	4	78%	98%	87%
Positive	231	7	28	97%	89%	93%

Berdasarkan Tabel 4.13, nilai precision pada hasil klasifikasi menggunakan algoritma *Random Forest* menunjukkan bahwa kelas negatif memiliki nilai precision sebesar 98%, kelas netral sebesar 78%, dan kelas positif sebesar 97%. Untuk nilai recall, kelas negatif memperoleh 83%, kelas netral 98%, dan kelas positif sebesar 89%. Adapun nilai F1-Score masing-masing adalah 90% untuk kelas negatif, 87% untuk kelas netral, dan 93% untuk kelas positif.

Untuk menghitung nilai akurasi, *precision*, *recall*, dan *F1-Score* pada suatu sistem klasifikasi, digunakan

bantuan *source code* seperti yang ditunjukkan pada Gambar 4.27.

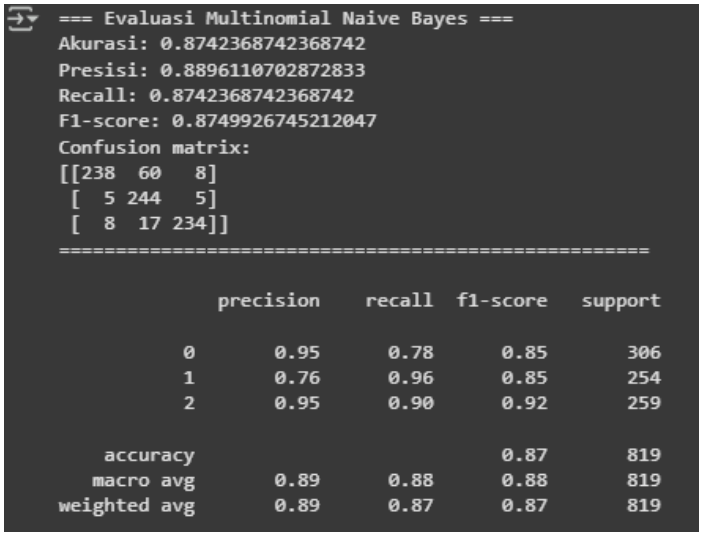


```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

# Convert sparse matrices to dense arrays (if applicable)
X_train_dense = X_train.toarray()
X_test_dense = X_test.toarray()

# Initialize models without random_state
nb_model = MultinomialNB()
rf_model = RandomForestClassifier()

# Train models
nb_model.fit(X_train_dense, y_train)
rf_model.fit(X_train_dense, y_train)

# Predictions
nb_pred = nb_model.predict(X_test_dense)
rf_pred = rf_model.predict(X_test_dense)

# Evaluation function
def evaluate_model(y_test, predicted, model_name):
    print(f"=== Evaluasi {model_name} ===")
    print("Akurasi:", accuracy_score(y_test, predicted))
    print("Presisi:", precision_score(y_test, predicted, average="weighted"))
    print("Recall:", recall_score(y_test, predicted, average="weighted"))
    print("F1-score:", f1_score(y_test, predicted, average="weighted"))
    print(f'Confusion matrix:\n{confusion_matrix(y_test, predicted)}')
    print('=====================================================\n')
    print(classification_report(y_test, predicted, zero_division=0))
    print("\n")

# Evaluate each model
evaluate_model(y_test, nb_pred, "Multinomial Naive Bayes")
evaluate_model(y_test, rf_pred, "Random Forest")
```

Gambar 4. 27 Source Code Perhitungan Naive Bayes dan Random Forest

Sehingga hasil dari kode program tersebut diperoleh performa model klasifikasi menggunakan algoritma *Naive Bayes*, ditinjau dari setiap kelas berdasarkan nilai *precision*, *recall*, dan *F1-Score*. Nilai-nilai ini ditampilkan dalam bentuk angka desimal dengan rentang antara 0 hingga 1, yang apabila

dikonversikan ke dalam bentuk persentase menunjukkan tingkat keberhasilan model dalam melakukan klasifikasi. Semakin mendekati angka 1, maka semakin baik pula performa model tersebut. Evaluasi lengkap hasil klasifikasi dengan algoritma *Naive Bayes* ini dapat dilihat pada Gambar 4.28.



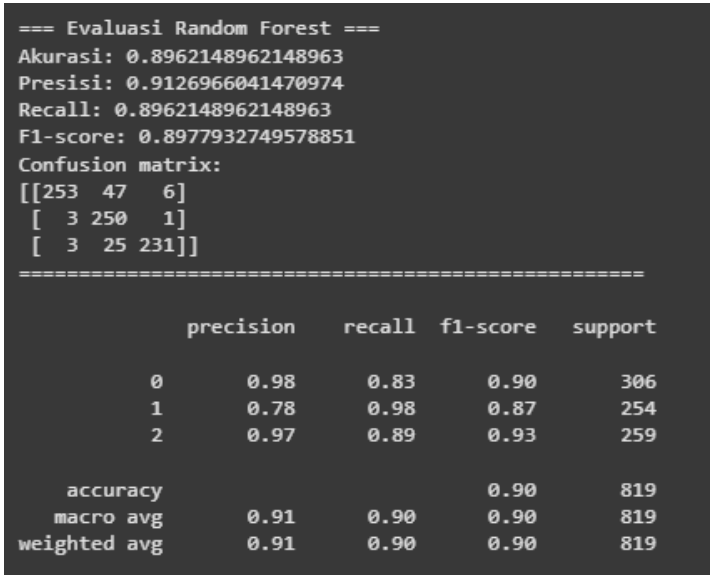
```
=== Evaluasi Multinomial Naive Bayes ===
Akurasi: 0.8742368742368742
Presisi: 0.8896110702872833
Recall: 0.8742368742368742
F1-score: 0.8749926745212047
Confusion matrix:
[[238  60   8]
 [  5 244   5]
 [  8  17 234]]
==================================================

              precision    recall  f1-score   support

           0       0.95      0.78      0.85       306
           1       0.76      0.96      0.85       254
           2       0.95      0.90      0.92       259

    accuracy                           0.87       819
   macro avg       0.89      0.88      0.88       819
weighted avg       0.89      0.87      0.87       819
```

Gambar 4. 28 Hasil Pengukuran Evaluasi Performa Naive Bayes

Hasil dari evaluasi model menunjukkan bahwa nilai *precision*, *recall*, dan *F1-Score* pada setiap kelas dalam algoritma *Naive Bayes* memiliki tingkat performa yang bervariasi. Berdasarkan hasil evaluasi, tingkat keberhasilan sistem dalam menemukan ketepatan prediksi atau yang disebut dengan *precision*, diperoleh

untuk kelas negatif sebesar 95%, kelas netral sebesar 76%, dan kelas positif sebesar 95%. Untuk mengukur sejauh mana sistem mampu menemukan kembali informasi yang relevan atau *recall*, diperoleh hasil sebesar 78% untuk kelas negatif, 96% untuk kelas netral, dan 90% untuk kelas positif. Nilai *F1-Score* yang merupakan harmonisasi antara *precision* dan *recall*, diperoleh sebesar 85% untuk kelas negatif, 85% untuk kelas netral, dan 92% untuk kelas positif. Secara keseluruhan, nilai rata-rata berbobot (weighted average) yang diperoleh dari proses evaluasi yaitu *precision* sebesar 89%, *recall* sebesar 87%, dan *F1-Score* sebesar 87%.

Selanjutnya, hasil evaluasi model menggunakan algoritma *Random Forest* juga menunjukkan performa klasifikasi berdasarkan *precision*, *recall*, dan *F1-Score* untuk masing-masing kelas. Sama seperti pada metode sebelumnya, nilai evaluasi ini dihitung berdasarkan hasil prediksi dibandingkan dengan label sebenarnya, yang dirangkum dalam bentuk *multiclass confusion matrix.* Hasil evaluasi lengkap dari algoritma *Random Forest* ditampilkan pada Gambar 4.29.



Gambar 4. 29 Hasil Pengukuran Evaluasi Performa Random Forest

Hasil dari evaluasi model menunjukkan bahwa nilai *precision*, *recall*, dan *F1-Score* pada setiap kelas dalam algoritma *Random Forest* memiliki performa yang cukup baik dan merata. Berdasarkan hasil evaluasi, tingkat ketepatan prediksi sistem atau *precision* diperoleh sebesar 98% untuk kelas negatif, 78% untuk kelas netral, dan 97% untuk kelas positif. Dari sisi *recall*, atau kemampuan sistem dalam menemukan kembali informasi yang relevan, diperoleh nilai sebesar 83% untuk kelas negatif, 98% untuk kelas netral, dan 89% untuk kelas positif. Sementara itu, nilai *F1-Score* yang

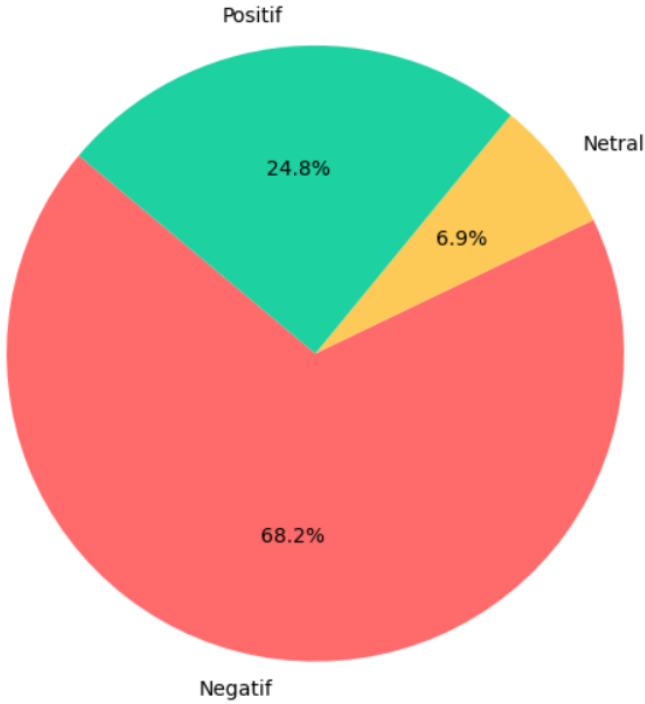
merupakan rata-rata harmonis dari *precision* dan *recall*, diperoleh sebesar 90% untuk kelas negatif, 87% untuk kelas netral, dan 93% untuk kelas positif. Secara keseluruhan, berdasarkan hasil evaluasi, nilai rata-rata berbobot (weighted average) dari *Random Forest* adalah *precision* sebesar 91%, *recall* sebesar 90%, dan *F1-Score* sebesar 90%.

H. Visualisasi

Tahap akhir dalam penelitian ini adalah memvisualisasikan hasil analisis sentimen menggunakan diagram lingkaran dan wordcloud. Wordcloud digunakan untuk menggambarkan frekuensi kata yang sering muncul berdasarkan hasil klasifikasi sentimen, sedangkan diagram lingkaran digunakan untuk menunjukkan proporsi masing-masing kategori sentimen. Visualisasi ini bertujuan untuk memberikan gambaran mengenai jumlah serta distribusi sentimen dalam ulasan pengguna terhadap aplikasi Getcontact di *Google Playstore*. Selain itu, wordcloud membantu mengidentifikasi kata-kata yang paling sering digunakan dalam ulasan.

Berdasarkan hasil akhir dari proses preprocessing data, diperoleh total 1365 komentar dengan sentimen negatif, 139 komentar netral, dan 496

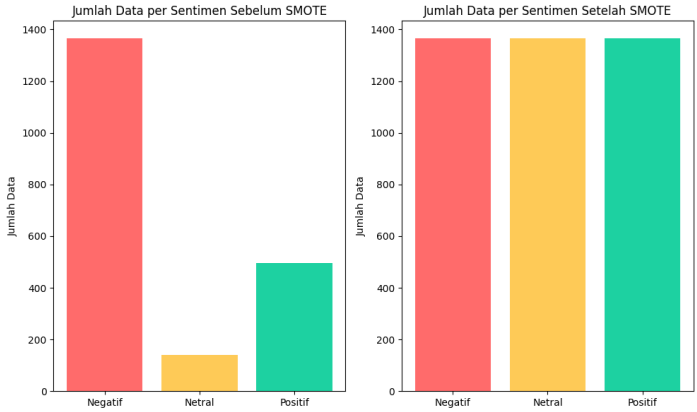
komentar positif. Persentase distribusi data berdasarkan masing-masing sentimen tersebut divisualisasikan dalam Gambar 4.30.



Gambar 4. 30 Persentase Pada Sentimen

Pada Gambar 4.30 dapat disimpulkan bahwa sentimen negatif mendominasi hasil analisis dengan persentase sebesar 68.2%, diikuti oleh sentimen positif sebesar 24.8%, dan sentimen netral sebesar 6.9%. Persentase ini menunjukkan bahwa mayoritas pengguna aplikasi Getcontact di *Google Playstore* memberikan

tanggapan yang cenderung negatif terhadap aplikasi tersebut.



Gambar 4. 31 Jumlah Data Setelah SMOTE

Pada gambar 4.31 ditunjukkan pada grafik sebelah kanan, setelah penyeimbangan menggunakan metode SMOTE, jumlah data untuk ketiga kategori sentimen menjadi sama rata, masing-masing dengan 1.365 data, sehingga total data adalah 4.095 setelah penyeimbangan. Tujuan penyeimbangan ini adalah untuk mengatasi masalah ketidakseimbangan kelas yang dapat mengganggu kinerja model klasifikasi, sehingga model dapat belajar dari masing-masing kategori sentimen secara seimbang.

Selanjutnya, peneliti ingin mengetahui kata-kata yang paling sering muncul dalam ulasan pengguna

aplikasi Getcontact di *Google Play Store*. Oleh karena itu, dilakukan visualisasi data menggunakan wordcloud yang ditampilkan pada Gambar 4.31.

Gambar 4. 32 Wordcloud Kata Populer Ulasan Pengguna

Berdasarkan Gambar 4.31, terlihat bahwa kata-kata yang paling menonjol dibicarakan oleh pengguna aplikasi Getcontact di Google Play Store ditampilkan dengan ukuran huruf yang lebih besar. Kata-kata seperti “premium”, “harus”, “bisa”, dan “dulu” menjadi kata yang paling sering muncul dalam ulasan. Selain itu, terdapat pula kata lain seperti “aplikasi”, “bagus”, “berbayar”, dan “login” dan masih banyak lainnya.

Selanjutnya, peneliti juga ingin mengetahui visualisasi wordcloud berdasarkan dua kategori sentimen dengan jumlah terbanyak, yaitu sentimen positif dan sentimen negatif. Gambar 4.32 menampilkan

wordcloud yang merepresentasikan kata-kata yang paling sering muncul pada masing-masing sentimen tersebut.

Gambar 4. 33 Wordcloud pada Sentimen Negatif dan Positif

Pada Gambar 4.32 dapat disimpulkan bahwa pada sentimen negatif, kata-kata seperti “harus”, “premium”, “dulu”, dan “bisa” merupakan kata yang paling sering muncul, yang mengindikasikan keluhan pengguna terhadap fitur berbayar aplikasi. Sementara itu, pada sentimen positif, kata “bagus”, “sangat”, dan “membantu” mendominasi, menunjukkan bahwa pengguna merasa puas dan terbantu dengan layanan yang diberikan oleh aplikasi Getcontact. Ukuran kata yang lebih besar menunjukkan frekuensi kemunculan yang lebih tinggi dalam komentar pengguna.

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Beberapa kesimpulan dapat dibuat berdasarkan penelitian yang telah dilakukan:

1. Penelitian ini membuktikan bahwa algoritma *Naive Bayes* dan *Random Forest* dapat digunakan untuk menganalisis sentimen ulasan pengguna aplikasi Getcontact di Google Play Store. Proses klasifikasi dimulai dari pengumpulan data melalui teknik web scraping, dilanjutkan dengan tahapan preprocessing, ekstraksi fitur menggunakan TF-IDF, dan penanganan data tidak seimbang menggunakan metode SMOTE. Selanjutnya dilakukan klasifikasi menggunakan kedua algoritma. Dataset awal dibagi menjadi 80% data latih dan 20% data uji.
2. Dari total 2000 ulasan yang dikumpulkan, diperoleh 1365 komentar bersentimen negatif, 496 komentar bersentimen positif, dan 139 komentar bersentimen netral. Berdasarkan hasil klasifikasi, sentimen negatif mendominasi dengan persentase sebesar 68.2%, diikuti oleh sentimen positif sebesar 24.8%, dan sentimen netral sebesar 6.9%. Hal ini menunjukkan bahwa mayoritas pengguna

memberikan tanggapan negatif terhadap aplikasi Getcontact.

3. Evaluasi performa model menunjukkan bahwa algoritma *Random Forest* menghasilkan klasifikasi yang lebih baik dibandingkan *Naive Bayes*. *Random Forest* mencatat akurasi sebesar 89,6%, dengan nilai precision 91,3%, recall 89,6%, dan F1-score 89,8%. Sementara itu, *Naive Bayes* memiliki akurasi 87,4%, precision 88,9%, recall 87,4%, dan F1-score 87,5%. Berdasarkan hasil tersebut, *Random Forest* terbukti lebih unggul dalam mengklasifikasikan sentimen ulasan pengguna aplikasi Getcontact di *Google Playstore*.

B. Saran

Penulis menyarankan beberapa hal berikut berdasarkan penelitian yang dilakukan:

1. Penelitian ini hanya menggunakan data ulasan pengguna aplikasi Getcontact dari *Google Play Store*. Untuk pengembangan penelitian selanjutnya, disarankan untuk mengambil data dari berbagai platform lain seperti *App Store* atau media sosial guna memperoleh hasil analisis sentimen yang lebih beragam dan menyeluruh.

2. Meskipun algoritma *Random Forest* menunjukkan performa yang lebih baik dibandingkan *Naive Bayes* dalam penelitian ini, disarankan untuk membandingkan lebih banyak metode klasifikasi lainnya seperti *Support Vector Machine* (SVM) atau *K-NN*, *Decision Tree*, dan lain sebagainya, agar diperoleh model yang lebih optimal dalam akurasi dan kecepatan pemrosesan.
3. Disarankan untuk menambahkan koleksi kamus khusus yang memuat kata tidak baku atau istilah gaul dalam proses preprocessing. Hal ini penting mengingat banyaknya ulasan yang menggunakan bahasa informal atau tidak standar, terutama dalam konteks media sosial dan ulasan aplikasi. Penanganan terhadap kata-kata tersebut dapat meningkatkan akurasi analisis sentimen secara keseluruhan.

DAFTAR PUSTAKA

Aditiya, Piqih, Ultach Enri, and Iqbal Maulana. 2022. "Analisis Sentimen Ulasan Pengguna Aplikasi Myim3 Pada Situs Google Play Menggunakan Support Vector Machine." *JURIKOM (Jurnal Riset Komputer)* 9(4): 1020. doi:10.30865/jurikom.v9i4.4673.

Ahmad, Amar, Uin Alauddin Makassar, Jl Sultan Alauddin No, and Kota Makassar. 2020. 08 *Media Sosial Dan Tantangan Masa Depan Generasi Milenial*.

Aldean Muhammad Yusril, Paradise, and Nugraha Novanda Alim Setya. 2022. "Analisis Sentimen Masyarakat Terhadap Vaksinasi-19 Di Twitter Menggunakan Metode Random Classifier (Studi Kasus: Vaksin Sinovac)." doi:https://doi.org/10.20895/inista.v4i2.575.

Andreswari, Delsy, Dewi Suranti, and Dimas Aulia Trianggara. 2024. "Application Of Text Mining In Grouping Thesis Topics Using TF-IDF Method Based On Thesis Abstract Penerapan Text Mining Dalam Pengelompokkan Topik Skripsi Menggunakan Metode TF-IDF Berdasarkan Abstrak Skripsi ARTICLE HISTORY." *Jurnal Komputer Indonesia* 3(2): 69–78. doi:10.37676/jki.v3i2.

Braja, Alex Sander P., and Achmad Kodar. 2023. "Implementasi Fine-Tuning BERT Untuk Analisis Sentimen Terhadap Review Aplikasi PUBG Mobile Di Google Play Store." *J I M P - Jurnal Informatika Merdeka Pasuruan* 7(3): 120. doi:10.51213/jimp.v7i3.779.

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. 16 Journal of Artificial Intelligence Research *SMOTE: Synthetic Minority Over-Sampling Technique*.

Duei Putri, Dianati, Gigih Forda Nama, and Wahyu Eko Sulistiono. 2022. "Analisis Sentimen Kinerja Dewan

Perwakilan Rakyat (DPR) Pada Twitter Menggunakan Metode *Naive Bayes* Classifier." *Jurnal Informatika dan Teknik Elektro Terapan* 10(1). doi:10.23960/jitet.v10i1.2262.

Figueroa, Ismael, Cristhy Jiménez, Hector Allende-Cid, and Paul Leger. 2019. "Developing Usability Heuristics with PROMETHEUS: A Case Study in Virtual Learning Environments." *Computer Standards and Interfaces* 65: 132–42. doi:10.1016/j.csi.2019.03.003.

Hermanto, Fahlapi Riza, Kuntoro Antonius Yadi, and Asra Taufik. 2024. "Perbandingan Algoritma Klasifikasi Analisis Sentimen Pengguna Aplikasi Dalam Pencegahan PenipuanOnline."

Kawani Gigih Putra. 2019. "Journal of Informatics, Information System, Software Engineering and Applications." 1(2): 73–081. doi:10.20895/INISTA.V1I2.

Kurnia -Rahayu -Engelbertus, Novi, Wendratama Zainuddin, Muda Z Monggilo -Acniah, Damayanti Dewa, Ayu Diah, and Angendari -Firya Qurratu'ain Abisono. *MODUS, MEDIUM, DAN REKOMENDASI PENIPUAN DIGITAL DI INDONESIA*.

Noer Azzahra, Fathimah, Tatang Rohana, Ayu Ratna Juwita, Perjuangan Karawang, Karawang Jl HSRonggo Waluyo, Telukjambe Timur, and Jawa Barat. 2024. "Penerapan Metode *Naive Bayes* Dalam Klasifikasi Spam SMS Menggunakan Fitur Teks Untuk Mengatasi Ancaman Pada Pengguna." *Journal of Information System Research (JOSH)* 5(3): 880. doi:10.47065/josh.v5i3.5070.

Phafiandita, Adisna Nadia, Ayu Permadani, Alsa Sukma Pradani, and M. Iqbal Wahyudi. 2022. "Urgensi Evaluasi Pembelajaran Di Kelas." *JIRA: Jurnal Inovasi dan Riset Akademik* 3(2): 111–21. doi:10.47387/jira.v3i2.262.

Prajamukti, Reino, and Mayanda Mega Santoni. *KLASIFIKASI DAN ANALISIS SENTIMEN PADA DATA TWITTER MENGGUNAKAN ALGORITMA NAÏVE BAYES (STUDI KASUS: TIMNAS INDONESIA SENIOR, U-23, DAN U-19)*. https://t.co/SAbaU6Prrz.

Prawira, Arya, Desi Arisandi, and Tri Sutrisno. 2022. "Penerapan Algoritma *Naive Bayes* Dan Multiple Linear Regression Untuk Prediksi Status Dan Plafon Kredit (Studi Kasus: Bank ABC)." *Journal on Education* 05(01).

Purba, Mariana, and Inka Rizki Padya. 2023. 8 Inka Rizki Padya *ANALISIS SENTIMEN MARKETPLACE DI ERA SOCIETY 5.0 MENGGUNAKAN ALGORITMA NAIVE BAYES*.

Retnosari, Rita, Program Studi, Sistem Informasi, Stmik Nusa, and Mandiri Jakarta. 2021. "ANALISIS KELAYAKAN KREDIT USAHA MIKRO BERJALAN PADA PERBANKAN DENGAN METODE *NAIVE BAYES*."

Riansyah Ramadhan, Gery, and Castaka Agus Sugianto. 2024. 8 Jurnal Mahasiswa Teknik Informatika *ANALISIS SENTIMEN ULASAN APLIKASI DANA DI GOOGLE PLAY STORE MENGGUNAKAN ALGORITMA NAÏVE BAYES*.

Ridwansyah, Tengku. 2022a. "KLIK: Kajian Ilmiah Informatika Dan Komputer Implementasi Text Mining Terhadap Analisis Sentimen Masyarakat Dunia Di Twitter Terhadap Kota Medan Menggunakan K-Fold Cross Validation Dan Naïve Bayes Classifier." *Media Online* 2(5): 178–85. https://djournals.com/klik.

Ridwansyah, Tengku. 2022b. "KLIK: Kajian Ilmiah Informatika Dan Komputer Implementasi Text Mining Terhadap Analisis Sentimen Masyarakat Dunia Di Twitter Terhadap Kota Medan Menggunakan K-Fold Cross Validation Dan Naïve Bayes Classifier." *Media Online* 2(5): 178–85. https://djournals.com/klik.

Sari, Fransiska Vina, and Arief Wibowo. 2019. "ANALISIS SENTIMEN PELANGGAN TOKO ONLINE JD.ID MENGGUNAKAN METODE NAÏVE BAYES CLASSIFIER BERBASIS KONVERSI IKON EMOSI." *Jurnal SIMETRIS* 10(2).

Siringoringo, Rimbun. 2018. 3 *KLASIFIKASI DATA TIDAK SEIMBANG MENGGUNAKAN ALGORITMA SMOTE DAN K-NEAREST NEIGHBOR*.

Suryono, Sigit, Dan Emha, and Taufiq Luthfi. 2020. *Analisis Sentimen Pada Twitter Dengan Menggunakan Metode Naïve Bayes Classifier*.

Tanggraeni, Artanti Inez, and Melkior N. N. Sitokdana. 2022. "Analisis Sentimen Aplikasi E-Government Pada Google Play Menggunakan Algoritma Naïve Bayes." *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)* 9(2): 785–95. doi:10.35957/jatisi.v9i2.1835.

Turmudi Zy, Ahmad, Lutfi Adji Ardiansyah, and Donny Maulana. 2021. "PELITA TEKNOLOGI Implementasi Algoritma Naïve Bayes Dalam Mendiagnosa Penyakit Angin Duduk." *Jurnal Pelita Teknologi* 16(1): 52–65.

Yolanda Paramitha, Nabilla, Aang Nuryaman, Ahmad Faisol, Eri Setiawan, dan Dina Eka Nurvazly, Jurusan Matematika, Fakultas Mipa, et al. 2023. 04 Jurnal Siger Matematika *Klasifikasi Penyakit Stroke Menggunakan Metode Naïve Bayes*.
https://www.kaggle.com/datasets/zzettrkalpakbal/full-filled-.

LAMPIRAN-LAMPIRAN

Lampiran 1 Pengesahan Lembar Ujian Komprehensif

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof Dr. Hamka Kampus III Ngaliyan
Semarang Telp. 7601295 Fax.7615387

PENGESAHAN UJIAN KOMPREHENSIF

Naskah proposal skripsi berikut ini:

Judul	: IMPLEMENTASI ALGORITMA NAIVE BAYES DAN RANDOM FOREST PADA ANALISIS SENTIMEN ULASAN APLIKASI GETCONTACT DI GOOGLE PLAYSTORE
Penulis	: Shauqi Nazmi Fauzan
NIM	: 2108096074
Jurusan	: Teknologi Informasi

Telah diujikan dalam sidang komprehensif oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo Semarang pada Kamis, 20 Februari 2025.

Semarang, 7 Maret 2025

DEWAN PENGUJI

Penguji I,	Penguji II,
Hery Mustofa, M.Kom.	Mokhamad Iklil Mustofa, M.Kom
NIP. 198703172019031007	NIP. 198808072019031010
Penguji III,	Penguji IV,
Nur Cahyo Hendro Wibowo, S.T.,M.Kom.	Dr. Masy Ari Ulinuha, M.T
NIP. 197312222006041001	NIP. 198108122011011007

Lampiran 2 Daftar Riwayat Hidup

RIWAYAT HIDUP

A. Identitas Diri

1. Nama Lengkap : Shauqi Nazmi Fauzan
2. Tempat & Tanggal Lahir : Jakarta, 12 November 2002
3. Alamat Rumah : Jl. Utan Panjang III RT.005 RW.0007
4. No HP : 087832163686
5. Email : shauqi.nazmi.fauzan12@gmail.com

B. Riwayat Pendidikan

1. Sekolah Dasar (SD) Negeri Kebon Kosong 01 Pagi
2. Sekolah Menengah Pertama (SMP) Negeri 59 Jakarta
3. Sekolah Menengah Atas (SMA) Negeri 15 Jakarta

Semarang, 10 Juni 2023

Shauqi Nazmi Fauzan
NIM. 2108096074

Lampiran 3 Source Code

```
import warnings
warnings.filterwarnings("ignore")

!pip install google-play-scraper

from google_play_scraper import app
import pandas as pd
import numpy as np

from google_play_scraper import Sort, reviews
from datetime import datetime

# Scrape ulasan terbaru
result, continuation_token = reviews(
    'app.source.getcontact',
    lang='id',  # Bahasa ulasan (Indonesia)
    country='id',  # Lokasi ulasan (Indonesia)
    sort=Sort.NEWEST,  # Mengambil ulasan
terbaru
    count=2000,  # Jumlah ulasan yang ingin
diambil
    filter_score_with=None  # Ambil semua
skor/rating
)

# Menentukan rentang waktu
start_date = datetime(2024, 12, 31)
end_date = datetime(2025, 1, 19)

# Filter ulasan berdasarkan rentang waktu
filtered_reviews = [
```

```
    review for review in result if start_date
<= review['at'] <= end_date
]

sentimen = []
for index, row in df.iterrows():
    if row['score'] == 'positif':
        sentimen.append(2)
    elif row['score'] == 'netral':
        sentimen.append(1)
    else:
        sentimen.append(0)
df['sentiment'] = sentimen
df.head()
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Cleaning
df['content'] = (
    df['content']
    .str.replace('https\S+', ' ', case=False,
regex=True)  # Menghapus URL
    .str.replace('@\S+', ' ', case=False,
regex=True)       # Menghapus mention
    .str.replace('#\S+', ' ', case=False,
regex=True)       # Menghapus hashtag
    .str.replace("\'\w+", ' ', case=False,
regex=True)      # Menghapus singkatan
    .str.replace("[^\w\s]", ' ', case=False,
regex=True)    # Menghapus tanda baca
```

```
    .str.replace("\s(2)", ' ', case=False,
regex=True)      # Menghapus angka '2' dengan
spasi
)
# Case folding
df['content'] = df['content'].str.lower()
# impor word_tokenize dari modul nltk
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
regexp = RegexpTokenizer('\w+')
df['content_token']=df['content'].apply(regexp.
tokenize)
df.head(10)
nltk.download('stopwords')
from nltk.corpus import stopwords

# Make a list of english stopwords
stopwords =
nltk.corpus.stopwords.words("indonesian")
# Extend the list with your own custom
stopwords
my_stopwords = ['getcontact']
stopwords.extend(my_stopwords)

# Remove stopwords
df['content_token'] =
df['content_token'].apply(lambda x: [item for
item in x if item not in stopwords])
df.head(10)

!pip install Sastrawi
# import Sastrawi package
```

```
from Sastrawi.Stemmer.StemmerFactory import
StemmerFactory

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()
df['stemmed'] =
df['content_token'].apply(lambda x:
[stemmer.stem(y) for y in x])

df['stemmed'] =
df['content_token'].apply(lambda x:
[stemmer.stem(y) for y in x]) # Stem every
word.
df.head(10)

df['text_string'] = df['stemmed'].apply(lambda
x: ' '.join([item for item in x if
len(item)>3]))
df.head(5)

from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.model_selection import
train_test_split

# Ambil fitur dan label dari dataframe
X = df['text_string']
y = df['sentiment']

# Inisialisasi dan transformasi teks dengan TF-
IDF
tfidf = TfidfVectorizer()
```

```
X_final = tfidf.fit_transform(X)

# Handling imbalanced using SMOTE
from imblearn.over_sampling import SMOTE #
Handling Imbalanced
smote = SMOTE()
x_sm,y_sm = smote.fit_resample(X_final,y)
X_train , X_test , y_train , y_test =
train_test_split(x_sm , y_sm ,
test_size=0.2,random_state=42)

feature_names = tfidf.get_feature_names_out()
max_docs = 5

for doc_index in range(min(max_docs,
x_sm.shape[0])):
    doc_tfidf = x_sm[doc_index]
    nonzero_indices = doc_tfidf.nonzero()[1]

    tfidf_scores = [(feature_names[idx],
doc_tfidf[0, idx]) for idx in nonzero_indices]
    tfidf_scores.sort(key=lambda x: x[1],
reverse=True)

    print(f"\nDokumen {doc_index}:")
    for word, score in tfidf_scores:
        print(f"  {word}: {score:.4f}")

from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score
```

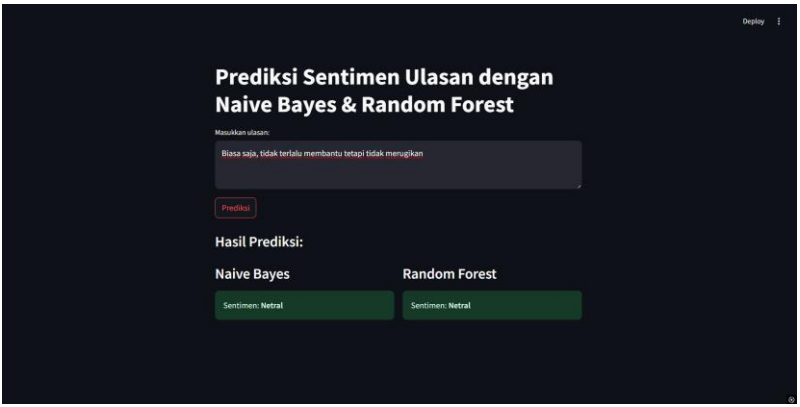
```
from sklearn.metrics import
classification_report
from sklearn.metrics import confusion_matrix

# Convert sparse matrices to dense arrays (if
applicable)
X_train_dense = X_train.toarray()
X_test_dense = X_test.toarray()

# Initialize models without random_state
nb_model = MultinomialNB()
rf_model = RandomForestClassifier()

# Train models
nb_model.fit(X_train_dense, y_train)
rf_model.fit(X_train_dense, y_train)

# Predictions
nb_pred = nb_model.predict(X_test_dense)
rf_pred = rf_model.predict(X_test_dense)

# Evaluation function
def evaluate_model(y_test, predicted,
model_name):
    print(f"=== Evaluasi {model_name} ===")
    print("Akurasi:", accuracy_score(y_test,
predicted))
    print("Presisi:", precision_score(y_test,
predicted, average="weighted"))
    print("Recall:", recall_score(y_test,
predicted, average="weighted"))
    print("F1-score:", f1_score(y_test,
predicted, average="weighted"))
```

```
    print(f'Confusion
matrix:\n{confusion_matrix(y_test,
predicted)}')
    print('==================================
================\n')
    print(classification_report(y_test,
predicted, zero_division=0))
    print("\n")

# Evaluate each model
evaluate_model(y_test, nb_pred, "Multinomial
Naive Bayes")
evaluate_model(y_test, rf_pred, "Random
Forest")

from wordcloud import WordCloud
from collections import Counter
import matplotlib.pyplot as plt

def generate_wordcloud(text_series, title, ax):
    # Gabungkan seluruh teks
    full_text = "
".join(text_series.astype(str))

    # Hitung frekuensi kata
    word_freq = Counter(full_text.split())

    # Ambil top 100 kata paling sering
    most_common_words =
dict(word_freq.most_common(100))

    # Buat WordCloud dari kata-kata terbanyak
```

```
    wordcloud = WordCloud(width=800,
height=400,
background_color='white').generate_from_frequen
cies(most_common_words)

    ax.imshow(wordcloud,
interpolation='bilinear')
    ax.set_title(title)
    ax.axis('off')

# Siapkan figure untuk 3 wordcloud
fig, axes = plt.subplots(1, 3, figsize=(22, 6))

# 1. WordCloud Semua Komentar
generate_wordcloud(df['content'], "Topik Umum
yang Banyak Diperbincangkan", axes[0])

# 2. WordCloud Sentimen Negatif
generate_wordcloud(df[df['sentiment'] ==
0]['content'], "WordCloud Sentimen Negatif",
axes[1])

# 3. WordCloud Sentimen Positif
generate_wordcloud(df[df['sentiment'] ==
2]['content'], "WordCloud Sentimen Positif",
axes[2])

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt
```

```
# Hitung jumlah data untuk masing-masing
sentimen
sentiment_counts =
df['sentiment'].value_counts().sort_index()  #
pastikan 0,1,2 urut
labels = ['Negatif', 'Netral', 'Positif']
colors = ['#ff6b6b', '#feca57', '#1dd1a1']

# Buat pie chart
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=labels,
autopct='%1.1f%%', colors=colors,
startangle=140)
plt.axis('equal')  # untuk membuat pie chart
lingkaran
plt.show()
```

Lampiran 4 Tampilan Web Prediksi Sentimen Ulasan



Lampiran 5 Source Code Web

```
import streamlit as st
import joblib


# Load model dan vectorizer
nb_model = joblib.load("nb_smote_model.pkl")
rf_model = joblib.load("rf_smote_model.pkl")
tfidf = joblib.load("tfidf_smote.pkl")

st.title("Prediksi Sentimen Ulasan dengan Naive
Bayes & Random Forest")

# Input dari user
user_input = st.text_area("Masukkan ulasan:")

if st.button("Prediksi"):
    if user_input.strip() == "":
        st.warning("Silakan masukkan teks ulasan
terlebih dahulu.")
    else:
        # Transformasi teks dengan tfidf
        input_vect = tfidf.transform([user_input])

        # Prediksi Naive Bayes
        pred_nb = nb_model.predict(input_vect)[0]
        # Prediksi Random Forest
        pred_rf = rf_model.predict(input_vect)[0]

        # Mapping label (ubah sesuai label yang
digunakan model Anda)
```

```
        label_map = {0: "Negatif", 1: "Netral", 2:
"Positif"}
        sentiment_nb = label_map.get(pred_nb,
str(pred_nb))
        sentiment_rf = label_map.get(pred_rf,
str(pred_rf))

        st.write("### Hasil Prediksi:")
        col1, col2 = st.columns(2)
        with col1:
            st.subheader("Naive Bayes")
            st.success(f"Sentimen:
**{sentiment_nb}**")
        with col2:
            st.subheader("Random Forest")
            st.success(f"Sentimen:
**{sentiment_rf}**")
```