UNIVERSITAS GUNADARMA

FAKULTAS ILMU KOMPUTER & TEKNOLOGI INFORMASI

ANALISIS SENTIMEN PADA ULASAN APLIKASI AMAZON SHOPPING DI GOOGLE PLAY STORE MENGGUNAKAN NAIVE BAYES CLASSIFIER

Disusun Oleh :

Nama : Elmo Allistair Heriyanto

NPM : 12118220

Jurusan : Sistem Informasi

Pembimbing : Dr. Ernianti Hasibuan., SKom., MSc.

Diajukan Guna Melengkapi Sebagian Syarat

Dalam Mencapai Gelar Sarjana Strata Satu (S1)

JAKARTA

2022

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini,

Nama	: Elmo Allistair Heriyanto
NPM	: 12118220
Judul Skripsi	: Analisis Sentimen Pada Ulasan Aplikasi Amazon Shopping di Google Play Store Menggunakan Naive Bayes Classifier
Tanggal Sidang	: 03 September 2022
Tanggal Lulus	: 03 September 2022

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

Demikian pernyataan ini dibuat dengan sebenarnya dan penuh dengan kesadaran.

Bogor, 27 Agustus 2022

(Elmo Allistair H)

LEMBAR PENGESAHAN

KOMISI PEMBIMBING

NO	NAMA	KEDUDUKAN
1.	Dr. Ernianti Hasibuan., SKom., MSc.	Ketua
2.	Dr. Dyah Cita Irawati., SSi., MM.	Anggota
3.	Dr. Ety Sutanty., SKom., MMSI.	Anggota

Tanggal Sidang : *03/09/2022*

PANITIA UJIAN

NO	NAMA	KEDUDUKAN
1.	Dr. Ravi Ahmad Salim	Ketua
2.	Prof. Dr. Wahyudi Priyono	Sekretaris
3.	Dr. Ernianti Hasibuan., SKom., MSc.	Anggota
4.	Dr. Dyah Cita Irawati., SSi., MM.	Anggota
5.	Dr. Ety Sutanty., SKom., MMSI.	Anggota

Tanggal Lulus : *03/09/2022*

Mengetahui,

Pembimbing **Kepala Bagian Sidang Ujian**

(Dr. Ernianti Hasibuan, SKom., MSc.) **(Dr. Edi Sukirman, SSi., MM., M.I.Kom)**

ABSTRAK

Elmo Allistair Heriyanto. 12118220

ANALISIS SENTIMEN PADA ULASAN APLIKASI AMAZON SHOPPING DI GOOGLE PLAY STORE MENGGUNAKAN NAIVE BAYES CLASSIFIER.

Skripsi. Jurusan Sistem Informasi. Fakultas Ilmu Komputer dan Teknologi Informasi. Universitas Gunadarma. 2022
Kata Kunci : analisa sentimen, amazon shopping, *machine learning*, naive bayes
(xii+50+Lampiran)

Analisis sentimen atau penggalian opini adalah studi yang menganalisis pendapat, pemikiran, dan kesan orang mengenai berbagai topik, subjek, dan produk atau layanan. Perkembangan sosial media membuat tersedianya data-data opini publik tersebut yang dapat ditemukan dengan mudah di internet seperti ulasan, forum diskusi, blog, twitter, sosial media, dan lain-lain. Besarnya volume data tersebut menyebabkan adanya kebutuhan akan sistem otomatis untuk mengklasifikasi data tersebut berdasarkan aspek yang berbeda dikarenakan pengklasifikasian data secara manual merupakan proses yang memakan waktu. Pada penelitian ini akan dilakukan analisis sentimen dengan pendekatan berbasis machine learning menggunakan algoritma Naive Bayes menggunakan data ulasan pengguna pada aplikasi Amazon Shopping yang terdapat pada Google Play Store. Hasil klasifikasi menggunakan keempat algoritma Naive Bayes menghasilkan akurasi rata-rata sebesar 82.15%, *precision* sebesar 72.25%, *recall* sebesar 83.49%, dan *f1-score* sebesar 77.41%. Multinomial NB menghasilkan akurasi terbaik diantara keempat algoritma Naive Bayes yang digunakan, yaitu sebesar 86.74%. Nilai *precision*, *recall*, dan *f1-score* berturut-turut adalah 78.82%, 85.90%, dan 82.21%.

Daftar Pustaka (2017 – 2022)

ABSTRACT

Elmo Allistair Heriyanto. 12118220

SENTIMENT ANALYSIS OF AMAZON SHOPPING APPLICATION REVIEWS ON GOOGLE PLAY STORE USING NAIVE BAYES CLASSIFIER

Thesis. Information Systems. Faculty of Computer Science and Information Technology. Gunadarma University. 2022.
Keywords : sentiment analysis, amazon shopping, machine learning, naive bayes
(xii+50+Attachment)

Sentiment analysis or opinion mining is a study that analyzes people's opinions, thoughts and impressions on various topics, subjects, and products or services. The development of social media makes public opinion data available which can be found easily on the internet such as reviews, discussion forums, blogs, twitter, social media, and others. The large volume of data causes the need for an automatic system to classify the data based on different aspects because classifying data manually is a time-consuming process. In this study, sentiment analysis will be carried out with a machine learning-based approach using the Naive Bayes algorithm using user review data on the Amazon Shopping application found on the Google Play Store. The classification results using the four Naive Bayes algorithms produce an average accuracy of 82.15%, precision of 72.25%, recall of 83.49%, and f1-score of 77.41%. Multinomial NB produces the best accuracy among the four Naive Bayes algorithms used, which is 86.74%. The precision, recall, and f1-score values are 78.82%, 85.90%, and 82.21%, respectively.

Bibliography (2017 – 2022)

KATA PENGANTAR

Puji dan syukur penulis haturkan kehadirat Allah SWT, karena berkat rahmat dan hidayah-Nya-lah saya dapat menyelesaikan skripsi yang berjudul "Analisis Sentimen pada Ulasan Amazon Shopping di Google Play Store dengan Naive Bayes Classifier". Skripsi ini dibuat untuk memenuhi tugas akhir perkuliahan dan sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Strata 1 di Program Sistem Informasi Fakultas Ilmu Komputer dan Teknologi Informasi di Universitas Gunadarma.

Penulis menyadari bahwa skripsi masih jauh dari sempurna. Oleh karena itu, penulis berharap dapat belajar lebih banyak lagi dalam mengimplementasikan ilmu yang didapatkan. Skripsi ini tentunya tidak mungkin terselesaikan tanpa adanya dukungan, bantuan, bimbingan, dan nasehat dari berbagai pihak selama penyusunan skripsi ini. Pada kesempatan ini penulis mengucapkan terima kasih, kepada :

1. Prof. Dr. Hj. E. S. Margianti, SE., MM., selaku Rektor Universitas Gunadarma.
2. Prof. Dr. Rer. Nat. Achmad Benny Mutiara, SSi., SKom., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma.
3. Dr. Setia Wirawan, SKom., MMSI., selaku Ketua Jurusan Sistem Informasi Universitas Gunadarma.
4. Dr. Edi Sukirman, SSi., MM., M.I.Kom., selaku Kepala Bagian Sidang Ujian Universitas Gunadarma.
5. Dr. Ernianti Hasibuan., SKom., MSc., selaku Dosen Pembimbing saya yang telah memberikan arahan dan dukungan selama proses penelitian.
6. Para Dosen Universitas Gunadarma khususnya Fakultas Ilmu Komputer dan Teknologi Informasi yang telah memberikan ilmu pengetahuannya kepada penulis.

7. Seluruh rekan perjuangan Mahasiswa/i Universitas Gunadarma Angkatan 2018, khususnya keluarga besar kelas 1KA08 dan 4KA17 yang saling mendukung selama 4 tahun.
8. Teman-teman komunitas Python Indonesia yang telah memberi masukkan dalam menulis program.
9. Bapak Heri dan Ibu Muliawati, selaku Orang Tua penulis yang selalu memberikan dukungan penuh dan doa selama proses penyusunan penulisan.
10. Serta semua pihak yang tidak dapat penulis sebutkan satu persatu, atas bantuan, saran dan masukan yang telah diberikan.

Jakarta, 27 Agustus 2022

(Elmo Allistair Heriyanto)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

1. PENDAHULUAN

1.1 Latar Belakang

Analisis sentimen atau penggalian opini adalah studi yang menganalisis pendapat, pemikiran, dan kesan orang mengenai berbagai topik, subjek, dan produk atau layanan. Sentimen mengacu pada hal positif atau negatif yang diungkapkan dalam teks. Analisis sentimen bertujuan untuk mengotomatisasi tugas mengidentifikasi opini dan sentimen yang orang-orang ungkapkan, dan kemudian mengklasifikasikan polaritas sentimennya [32]. Sistem analisis sentimen dapat diterapkan di hampir setiap bisnis dan domain sosial karena karena opini merupakan pusat psikologi manusia dan merupakan pengaruh utama dari perilaku kita.

Sejak awal tahun 2000, analisis sentimen telah berkembang menjadi salah satu area penelitian paling aktif dalam Natural Language Processing [16], aktivitas industri seputar analisis sentimen juga berkembang secara pesat. Pengimplementasian analisis sentimen telah menyebar ke banyak domain, mulai dari bisnis, layanan kesehatan, pemilihan umum, analisis produk, hingga riset pasar. Di masa lalu, ketika sebuah bisnis membutuhkan opini publik atau pelanggan, mereka akan melakukan survei dan jajak pendapat. Setelah perkembangan sosial media, pelaku bisnis tidak lagi bergantung pada cara-cara tersebut dikarenakan tersedianya data-data opini publik yang dapat ditemukan dengan mudah di internet seperti ulasan, forum diskusi, blog, twitter, sosial media, dan lain-lain. Besarnya volume data tersebut menyebabkan adanya kebutuhan akan sistem otomatis untuk mengklasifikasi data tersebut berdasarkan aspek yang berbeda dikarenakan pengklasifikasian data secara manual merupakan proses yang memakan waktu.

Ulasan merupakan salah satu bentuk penilaian seseorang terhadap suatu produk atau jasa. Analisis Sentimen dapat membantu kita untuk mengetahui opini pelanggan melalui ulasan yang mereka tuliskan mengenai suatu produk atau jasa. Ulasan tersebut dapat menjadi informasi berharga untuk pelanggan lain. Misalnya,

sebelum membeli suatu produk, kebanyakan orang cenderung mencari melalui ulasan tentang produk tersebut yang akan membantu mereka dalam membuat pilihan. Berdasarkan survey yang dilakukan oleh BrightLocal [3], 79% konsumen mengatakan mereka mempercayai ulasan online lebih dari rekomendasi pribadi dari teman atau keluarga.

Terdapat beberapa pendekatan yang umum digunakan untuk melakukan analisis sentimen, antara lain dengan menggunakan *machine learning*, berbasis leksikon, hybrid dan teknik lainnya seperti *aspect-based approach*, *transfer learning*, dan *multimodal sentiment analysis* [1]. Pada penelitian ini akan dibuat sebuah sistem untuk mengklasifikasi sentimen dengan pendekatan berbasis *machine learning* menggunakan algoritma Naive Bayes menggunakan data ulasan pengguna aplikasi Amazon Shopping yang terdapat pada Google Play Store. Pendekatan menggunakan *machine learning* dipilih karena kemampuannya untuk melakukan analisa dengan cepat, sehingga dapat mengurangi waktu dan upaya untuk mengklasifikasi ulasan. Adapun tujuan yang ingin dicapai adalah mengetahui performa yang dihasilkan algoritma Naive Bayes dalam melakukan klasifikasi sentimen sekalis mengetahui perbandingan performanya dengan algoritma berbasis *machine learning* lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dari penelitian ini adalah sebagai berikut:

1) Bagaimana cara melakukan klasifikasi sentimen menggunakan *machine learning*?
2) Bagaimana sentimen pengguna terhadap aplikasi Amazon Shopping?
3) Bagaimana performa yang dihasilkan pengklasifikasi Naive Bayes dalam melakukan klasifikasi sentimen?
4) Bagaimana cara meningkatkan performa model pengklasifikasi sentimen?
5) Bagaimana perbandingan performa klasifikasi yang dihasilkan algoritma Naive Bayes dengan algoritma berbasis *machine learning* lainnya?

1.3 Ruang Lingkup

Agar pembahasan penelitian ini tidak menyimpang dari apa yang telah dirumuskan, maka diperlukan batasan-batasan. Adapun batasan-batasan dalam penelitian ini adalah:

1) Data ulasan yang digunakan adalah ulasan aplikasi Amazon Shopping yang didapatkan di Google Play Store yang berjumlah 2147 data ulasan berbahasa inggris.
2) Algoritma Naive Bayes yang digunakan adalah Bernoulli, Complement, Gaussian, dan Multinomial Naive Bayes.
3) Algoritma pembanding berbasis *machine learning* lainnya yang digunakan sebagai perbandingan adalah Support Vector, Decision Tree, Random Forest, dan Logistic Regression.
4) Sentimen ulasan dikategorikan menjadi 2 kategori: positif dan negatif.
5) *Development environment* penelitian dilakukan dalam layanan Google Collaboratory dengan menggunakan *runtime* Python versi 3.7.13.

1.4 Tujuan Penelitian

Beberapa tujuan yang ingin dicapai dengan dilakukannya penelitian ini adalah sebagai berikut:

1) Mengetahui sentimen pengguna terhadap aplikasi Amazon Shopping.
2) Mengetahui performa klasifikasi yang dihasilkan Naive Bayes Classifier.
3) Mempelajari cara untuk meningkatkan performa model *machine learning*.
4) Membandingkan performa algoritma bayesian dan non-bayesian berbasis *machine learning* lainnya dalam melakukan klasifikasi sentimen.

1.5 Metode Penelitian

Metode penelitian berisi tahapan-tahapan yang akan dilakukan agar pelaksanaan penelitian sesuai dengan rencana dan mendapatkan hasil yang diharapkan. Adapun tahapan dalam penelitian ini adalah sebagai berikut:

1) Penyiapan data

   Tahapan pertama yang perlu dilakukan adalah menyiapkan data. Kegiatan yang dilakukan pada tahapan ini antara lain pengambilan data ulasan aplikasi Amazon Shopping yang tersedia di Google Play Store. Data yang didapatkan akan dibersihkan dan diberi label sentimen terlebih dahulu sebelum dilanjutkan ke tahap berikutnya.

2) *Text pre-processing*

   Pre-processing dilakukan untuk membersihkan *noise* untuk mendapatkan informasi seakurat mungkin dari teks ulasan. Proses-proses yang dilakukan antara lain: *tokenization*, *case folding*, penghapusan *stopwords* dan *punctuation, part-of-speech tagging*, dan *lemmatization*.

3) Implementasi klasifikasi sentimen

   Setelah data teks selesai diproses, selanjutnya akan dilakukan analisa sentimen menggunakan *machine learning*, algoritma yang digunakan adalah Naive Bayes Classifier, hasil analisis akan diklasifikasikan menjadi dua kategori: positif dan negatif. Tahapan ini meliputi pengekstraksian fitur pada data ulasan, pembagian data menjadi set data latih dan uji, kemudian dilakukannya klasifikasi sentimen itu sendiri.

4) Evaluasi hasil

   Kinerja algoritma klasifikasi diringkas menggunakan teknik *confusion matrix*. Hasil analisis sentimen akan dievaluasi menggunakan matrik evaluasi akurasi, *precision*, *recall*, dan *f1-score*.

5) Peningkatan kinerja model

   Model yang telah dievaluasi dan diketahui kinerja klasifikasinya kemudian akan dilakukan percobaan untuk meningkatkan kinerja tersebut. Percobaan yang akan dilakukan antara lain menggunakan pembobotan TF-IDF dan n-gram yang berbeda saat melakukan pengekstraksian fitur, serta penggunaan persentase pembagian data yang berbeda.

1.4 Sistematika Penulisan

Penulisan skripsi dengan judul "Analisis Sentimen pada Ulasan Aplikasi Amazon Shopping di Google Play Store Menggunakan Naive Bayes Classifier" ini dibagi menjadi lima bab, yang akan disusun berdasarkan urutan sebagai berikut:

1. PENDAHULUAN

   Bagian pendahuluan akan membahas permasalahan yang melatarbelakangi dilakukannya penelitian ini beserta dengan rumusan masalah, ruang lingkup, tujuan penelitian, metode penelitian dan sistematika penulisan.

2. TINJAUAN PUSTAKA

   Bagian tinjauan pustaka menguraikan dasar-dasar teori yang berhubungan dengan topik penelitian beserta dengan literatur penelitian terkait yang telah dilakukan sebelumnya.

3. METODE PENELITIAN

   Bagian metode penelitian membahas secara rinci tahapan-tahapan yang dilakukan penulis dalam melakukan penelitian agar mencapai hasil yang diharapkan.

4. HASIL DAN PEMBAHASAN

   Bagian hasil dan pembahasan menjelaskan tentang hasil pengolahan data dan hasil pengimplementasian dan pengujian model, serta menjabarkan hasil evaluasi model yang digunakan.

5. PENUTUP

   Bagian penutup berisi kesimpulan yang merupakan rangkuman dari hasil penelitian yang dilakukan beserta kumpulan saran-saran yang dapat digunakan untuk peneliti selanjutnya di masa depan.

2. TINJAUAN PUSTAKA

2.1 *Machine Learning*

Machine learning dapat didefinisikan secara luas sebagai metode komputasi yang menggunakan pengalaman untuk meningkatkan kinerja atau membuat prediksi yang akurat [18]. Pengalaman tersebut mengacu pada informasi yang biasanya berbentuk data elektronik yang dikumpulkan dan tersedia untuk analisis. Data ini bisa dalam bentuk digitalisasi perangkat pelatihan yang sudah berlabel, atau jenis informasi lain yang diperoleh melalui interaksi dengan lingkungan. *Machine learning* merupakan bagian dari disiplin ilmu kecerdasan bagian (AI) dapat diaplikasikan dengan sangat luas, beberapa diantaranya adalah klasifikasi teks atau dokumen, *natural language processing*, *speech processing*, *computer vision*, dan lain-lain.

Tujuan praktis utama dari *machine learning* adalah menghasilkan prediksi yang akurat dan merancang algoritma yang efisien dan kuat untuk menghasilkan prediksi tersebut bahkan untuk permasalahan skala besar [18]. Berdasarkan teknik pembelajarannya, tipe-tipe machine learning dapat dibedakan menjadi [22] :

1. *Supervised learning*, tipe ini menggunakan data yang sudah diberi label yang digunakan untuk permasalahan klasifikasi dan regresi.
2. *Unsupervised learning*, menggunakan data masukan yang tidak diberi, diklasifikasikan ataupun dikategorikan labelnya.
3. S*emi-supervised learning*, menggunakan data berlabel untuk mempelajari struktur data yang tidak berlabel dan mencoba membuat prediksi.
4. *Reinforcement learning*, adalah subbidang khusus dalam *machine learning* yang berkaitan dengan bagaimana agen perangkat lunak harus menemukan urutan tindakan yang berkontribusi untuk mencapai tujuan tertentu [12].

2.2 *Natural Language Processing*

NLP (*Natural Language Processing*) merupakan pemrosesan komunikasi manusia berbasis mesin yang bertujuan untuk mengajarkan mesin bagaimana cara memproses dan memahami bahasa manusia, sehingga memungkinkan saluran komunikasi yang mudah antara manusia dan mesin [2]. NLP telah banyak diterapkan di berbagai area, salah satunya adalah analisis sentimen, analisis sosial media, *chatbots*, penerjemah, ekstraksi informasi dan lain-lain. NLP dan *machine learning* merupakan disiplin ilmu yang saling terkait. Dalam banyak kasus, output dari teknik NLP seperti *text preprocessing* digunakan sebagai input untuk *machine learning*, dan sebaliknya seperti menerapkan *supervised machine learning* ke konstruksi kamus untuk mengidentifikasi *entity* dengan NLP [11].

Aplikasi terkait NLP dibangun menggunakan sejumlah besar data yang disebut corpus. Corpus didefinisikan sebagai kumpulan materi bahasa alami tertulis atau lisan, disimpan di dalam komputer, dan digunakan untuk mengetahui bagaimana bahasa digunakan [29]. Beberapa manfaat penggunaan corpus antara lain membantu beberapa analisis statistik seperti distribusi frekuensi dan memvalidasi aturan linguistik seperti menemukan tata bahasa (*grammar*) yang salah. Aplikasi NLP terkadang menggunakan satu corpus sebagai masukan, atau menggunakan lebih dari satu corpus yang disebut corpora.

2.3 *Text Mining*

Text mining, juga disebut *text analytics*, adalah teknik yang mengubah data tidak terstruktur menjadi data terstruktur menggunakan bantuan NLP untuk menyempurnakan hasil analisis yang dilakukan menggunakan algoritma *machine learning* [15]. *Text mining* mengidentifikasi fakta, hubungan, dan pernyataan dalam kumpulan data tekstual yang besar. Informasi yang diekstrak dapat diubah menjadi bentuk terstruktur yang dapat dianalisis lebih lanjut. Penerapan dari text mining yang paling umum adalah analisa sentimen, *text categorization*, *text clustering*, dan *document summarization*. Tugas-tugas tersebut pada umumnya memerlukan *text preprocessing* sebelum dilakukan [28].

2.4 *Text Preprocessing*

Text Preprocessing merupakan metode untuk membersihkan data teks yang tidak terstruktur dan membuatnya siap untuk diproses ke dalam model. *Preprocessing* merupakan bagian integral dari aplikasi berbasis NLP. Langkah-langkah ini terdiri dari penghapusan karakter atau bagian teks yang tidak relevan dan tidak diinginkan berdasarkan beberapa pola yang diamati, namun tetap mempertahankan maksud asli dari teks tersebut [20]. Beberapa langkah-langkah *text preprocessing* dalam NLP antara lain [8]:

1) *Tokenization,* proses ini mengacu pada prosedur pemisahan kalimat kata-kata penyusunnya. Contoh pada kalimat "Saya membaca buku.", setelah meneruskan kalimat ini ke program tokenisasi, kata/token yang diekstrak akan menjadi: "Saya", "membaca", "buku" dan ".". Contoh tersebut mengekstrak satu token pada satu waktu, token ini disebut *unigram*. Token juga dapat diekstrak menjadi dua (*bigrams*) dan tiga (*trigrams*) token sekaligus.
2) *Case folding,* merupakan tahapan mengkonversi teks menjadi satu bentuk yang standar, yaitu *uppercase* atau *lowercase*. Strategi yang paling umum digunakan adalah mengkonversi semua huruf menjadi *lowercase*.
3) Penghapusan *punctuations*, tanda baca merupakan simbol yang digunakan untuk memisahkan kalimat untuk memperjelas artinya. Simbol-simbol ini dapat mempengaruhi hasil klasifikasi sentimen terutama jika teknik yang digunakan bergantung pada frekuensi kemunculan kata dan frasa, karena tanda baca sering digunakan dalam teks.
4) Penghapusan *stop words*, *stop words* adalah kata-kata umum yang hanya digunakan untuk mendukung konstruksi kalimat. Contoh *stopwords* dari bahasa inggris adalah: "*a*", "*am*", dan "*the*". Penghapusan *stop words* dari analisis dilakukan karena kata-kata tersebut sangat sering muncul dan kehadirannya tidak mempengaruhi makna kalimat yang ada di dalamnya.
5) Penandaan PoS, PoS (*part of speech*) merupakan proses penandaan kata-kata dalam kalimat ke dalam bagian ucapannya dan akan dilabeli di akhir. Penanda PoS dapat membantu mengurangi ambiguitas, yang merupakan

masalah utama di NLP. PoS diekstrak dari token yang membentuk kalimat, sehingga kita dapat menyaring PoS yang menarik dan menganalisisnya. Penn TreeBank [19] merupakan kumpulan tag yang umum digunakan untuk bahasa Inggris.

6) *Lemmatization*, proses ini berkaitan dengan *stemming*, yang merupakan proses pengelompokan bentuk-bentuk infleksi yang berbeda dari sebuah kata sehingga mereka dapat dianalisis sebagai satu item. Sebagai tambahan, lematisasi melakukan pemeriksaan tambahan dengan melihat kamus untuk mengekstrak bentuk dasar dari sebuah kata, pemeriksaan ini juga membuat proses menjadi lebih lambat.

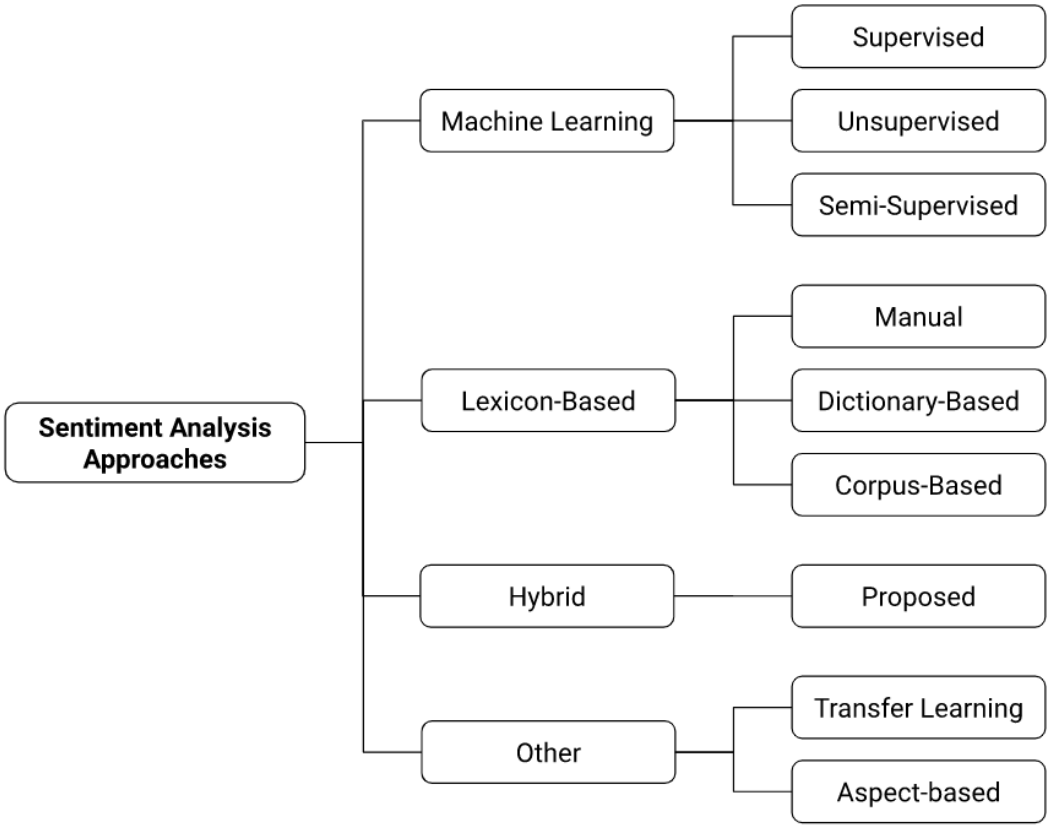
2.5 Analisis Sentimen

Analisis sentimen atau juga disebut penggalian opini merupakan bidang studi yang menganalisis pendapat, sentimen, penilaian, sikap, dan emosi seseorang terhadap suatu entitas yang dapat berupa produk, layanan, organisasi, individu, acara, masalah, atau topik yang diungkapkan dalam teks tertulis [16]. Analisis sentimen telah memperluas penelitian NLP secara signifikan karena telah memperkenalkan banyak masalah penelitian yang menantang yang belum pernah dipelajari sebelumnya. Analisis sentimen merupakan masalah analisis semantik, tetapi sangat terfokus dan terbatas karena sistem analisis sentimen tidak perlu sepenuhnya "memahami" setiap kalimat atau dokumen dan hanya perlu memahami beberapa aspek saja, misalnya opini positif dan negatif serta sasarannya.

Terdapat banyak pendekatan untuk melakukan analisa sentimen, diantaranya adalah [1]:

1) *Machine learning*, merupakan pendekatan yang paling banyak digunakan. Pendekatan ini bergantung pada algoritma *machine learning* dan fitur linguistik untuk melakukan klasifikasi sentimen.
2) *Lexicon-based*, menggunakan leksikon sentimen yang telah disiapkan sebelumnya untuk menilai dokumen dengan menggabungkan skor sentimen dari semua kata dalam dokumen.

3) Hybrid, menggabungkan pendekatan leksikon dan *machine learning* yang menggabungkan keluaran analisis leksikal dengan fleksibilitas pendekatan *machine learning* untuk mengatasi ambiguitas dan mengintegrasikan konteks kata-kata sentimen.
4) Pendekatan lainnya, seperti *transfer learning* yang menggunakan kesamaan data, distribusi data, tugas model, dan sebagainya untuk menerapkan pengetahuan yang sudah dipelajari dalam satu domain ke domain baru.



Gambar 2.1 Pendekatan-pendekatan analisis sentiment [1].

2.6 Ekstraksi Fitur

Ekstraksi fitur adalah proses pengurangan dimensi di mana kumpulan awal data mentah direduksi menjadi kelompok yang lebih mudah dikelola untuk diproses. Ekstraksi fitur mencoba untuk mengekstrak satu set fitur (dimensi) yang sama sekali baru dari pola data daripada memilih fitur dari atribut yang ada [7]. Setiap instruksi yang kita masukkan ke komputer akan diubah menjadi digit biner, hal ini diperlukan karena Komputer hanya memahami digit biner, yaitu 0 dan 1. Demikian pula model *machine learning* yang cenderung hanya memahami data numerik. Oleh karena itu, data teks perlu diubah menjadi bentuk numerik. Dalam kasus ini, data ulasan yang dibersihkan akan direpresentasikan menjadi bentuk numerik menggunakan metode *Bag-of-Words* dan TF-IDF.

Pembuatan *features* dalam teks dilakukan menggunakan metode *Bag-of-Words* dan TF-IDF. *Bag-of-Words* merupakan representasi yang mengubah teks menjadi vektor dengan panjang tetap dengan menghitung berapa kali setiap kata muncul dalam dokumen, proses ini juga sering disebut sebagai vektorisasi. *Term Frequency-Inverse Document Frequency* (TF-IDF) menggunakan semua token dalam kumpulan data sebagai kosakata. Frekuensi kemunculan suatu token dari kosakata dalam setiap dokumen terdiri dari *Term Frequency* (TF) dan jumlah dokumen di mana token muncul akan menentukan nilai *Inverse Document Frequency* (TF-IDF).

2.7 Naive Bayes Classifier

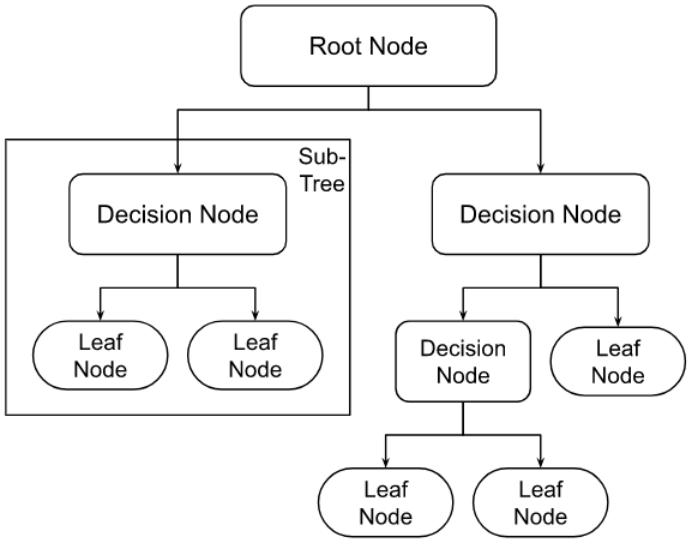
Naive Bayes adalah algoritma probabilistik yang biasanya digunakan untuk masalah klasifikasi. Naive Bayes Classifier bukanlah algoritma tunggal, melainkan kumpulan algoritma klasifikasi di mana semuanya memiliki prinsip yang sama berdasarkan Teorema Bayes [17] yang menggambarkan probabilitas suatu peristiwa berdasarkan pengetahuan sebelumnya atau probabilitas tertentu lainnya yang diketahui dari peristiwa itu. Pengklasifikasi Naive Bayes yang digunakan dalam penelitian ini adalah:

1. Multinomial Naive Bayes, model ini bekerja pada konsep *term frequency* yang berarti berapa kali sebuah kata muncul dalam dokumen. Multinomial NB memiliki kemampuan untuk mengklasifikasikan data yang tidak dapat direpresentasikan secara numerik.
2. Bernoulli Naive Bayes, model ini berguna jika vektor fitur yang digunakan adalah biner (yaitu 0 dan 1). Contoh paling umum adalah memeriksa apakah sebuah kata akan muncul dalam dokumen atau tidak (0 jika tidak, 1 jika iya). Bernoulli NB dapat memberikan hasil yang lebih baik dalam kasus di mana menghitung frekuensi kemunculan suatu kata kurang penting.
3. Gaussian Naive Bayes, model ini digunakan ketika fitur memiliki nilai kontinu. Gaussian NB mengasumsikan bahwa semua fitur mengikuti distribusi gaussian yaitu distribusi normal.
4. Complement Naive Bayes, model ini dirancang untuk mengoreksi "asumsi berat" yang dibuat oleh pengklasifikasi Multinomial Naive Bayes yang tidak bekerja dengan baik pada kumpulan data yang tidak seimbang. Pengklasifikasi ini tidak menghitung probabilitas item milik kelas tertentu, melainkan menghitung probabilitas item milik semua kelas

2.8 Algoritma Pembanding

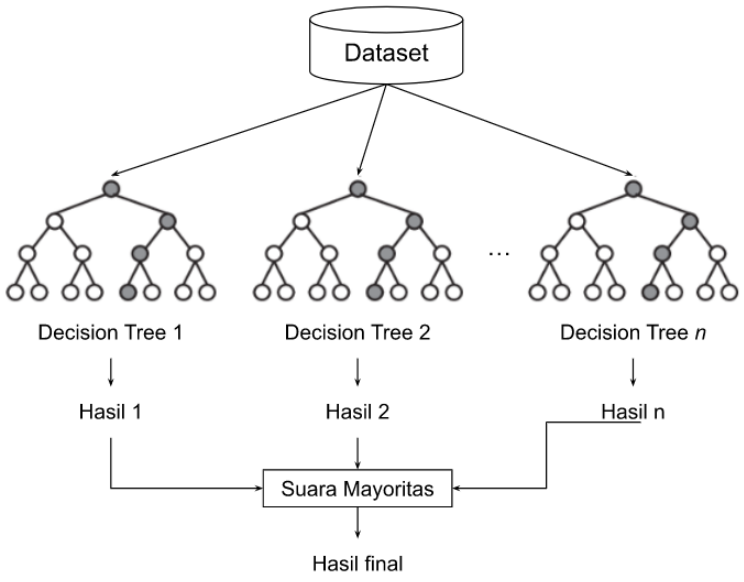
Algoritma pembanding digunakan sebagai upaya mencari model dengan akurasi terbesar sekaligus sebagai perbandingan dengan algoritma Naive Bayes yang digunakan. Algoritma-algoritma yang dipilih adalah yang antara lain:

1. *Decision Tree*, model ini memiliki struktur seperti pohon di mana simpul teratas dianggap sebagai akar pohon yang secara rekursif dibagi pada rangkaian simpul keputusan dari akar sampai simpul terminal atau simpul keputusan tercapai [14]. Bagian atas *decision tree* dikenal sebagai *root node* yang sesuai dengan variabel prediktor terbaik.



Gambar 2.2 Ilustrasi dari *decision tree* [5].

2. *Random Forest*, model ini menggunakan beberapa pohon untuk menghitung rata-rata (regresi) atau suara mayoritas (klasifikasi) di simpul daun terminal saat membuat prediksi [14]. Pohon klasifikasi dan regresi dibangun menggunakan set data pelatihan yang dipilih secara acak dan subset acak dari variabel prediktor untuk hasil pemodelan [25].



Gambar 2.3 Ilustrasi dari *random forest*.

3. *Logistic Regression*, model *logistic regression* menguji hubungan antara satu atau lebih faktor independen untuk meramalkan variabel data dependen. Berbeda dengan *linear regression*, *logistic regression* memprediksi hasil sebagai probabilitas kelas default [23]. Regresi logistik digunakan ketika variabel dependen adalah kategoris, yang merupakan model paling umum untuk data respons biner. Dalam hal kemudahan komputasi, *logistic regression* adalah model terbaik di antara model linier umum untuk data respons biner [9].
4. *Support Vector Machine*, merupakan model yang memanfaatkan titik batas yang memisahkan kelas yang dikenal sebagai vektor pendukung untuk klasifikasi data yang tidak berlabel. Tujuan utama SVM adalah untuk menemukan bidang data optimal yang membagi dua kelas titik data, algoritma ini memberikan akurasi yang lebih baik dalam kasus dimana ruang dimensi tinggi [24].

2.9 *Confusion Matrix*

Confusion matrix merupakan teknik untuk meringkas kinerja algoritma klasifikasi yang merupakan matriks berukuran n x n yang menunjukkan klasifikasi yang diprediksi dan aktual, di mana n adalah jumlah kelas yang berbeda. Beberapa istilah yang terdapat dalam *confusion matrix* antara lain:

- *True Positive* (TP) adalah jumlah prediksi negatif yang benar;
- *False Positive* (FN) adalah jumlah prediksi positif yang salah;
- *True Negative* (TN) adalah jumlah prediksi negatif yang salah;
- *False Negative* (FN) adalah jumlah prediksi positif yang benar.

		PREDICTED CLASS	
		Positive	**Negative**
ACTUAL CLASS	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Tabel 2.1 Representasi *confusion matrix*.

Akurasi, *precision*, *recall*, dan *F-1 score* dapat dihitung menggunakan informasi yang diberikan dalam *confusion matrix*. Akurasi merupakan seberapa akurat model dapat mengklasifikasi dengan benar, nilai ini dapat dihitung dengan membagi jumlah prediksi yang benar dengan jumlah total sampel. *Precision* menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model, nilai *precision* yang besar menunjukkan rendahnya nilai *false positive*. *Recall* adalah rasio pengamatan positif yang diprediksi dengan benar dengan semua pengamatan di kelas yang sebenarnya, nilai *recall* yang besar menunjukkan rendahnya nilai *false negative*. Menggabungkan nilai *precision* dan *recall* akan menghasilkan metrik tunggal yang dikenal sebagai *F1-score*, yang merupakan rata-rata harmonik tertimbang dari *precision* dan *recall*. Tabel 2.2 menunjukkan metrik pengukuran kinerja model *machine learning* beserta formula untuk menghitungnya.

Metrik	Formula
Akurasi	$\frac{TP + TN}{TP + TN + FP + FN}$
Recall	$\frac{TP}{TP + FN}$
Precision	$\frac{TP}{TP + FP}$
F1-score	$2 * \frac{Recall * Precision}{Recall + Precision}$

Tabel 2.2 Formula metrik evaluasi.

2.10 Python

Python adalah bahasa pemrograman *interpreted* tingkat tinggi, berorientasi objek, dengan semantik dinamis yang dibuat oleh Guido van Rossum pada tahun 1991. Python merupakan *general-purpose language*, yang artinya dapat digunakan

untuk menyelesaikan berbagai macam masalah berbeda dan tidak memiliki domain aktivitas tertentu. Menurut indeks *PopularitY of Programming Language* (PYPL) [21] dan *The Importance of Being Earnest* (TIOBE) [30], bahasa Python menempati peringkat pertama sebagai bahasa pemrograman paling populer pada saat ini. Popularitas bahasa Python telah tumbuh sebanyak 11% dalam 5 tahun terakhir. Beberapa perusahaan besar yang menggunakan Python antara lain: Google, YouTube, BitTorrent, NASA, dan lain-lain. Fitur terpenting dalam Python adalah mendukung beberapa paradigma pemrograman, termasuk pemrograman berorientasi objek, imperatif dan fungsional atau gaya prosedural [26].

2.11 *Natural Language Toolkit*

Natural Language Toolkit (NLTK) merupakan salah satu *library* Python paling populer untuk melakukan tugas NLP. NLTK dikembangkan oleh Steven Bird dan Edward Loper dari *University of Pennsylvania* yang dibuat untuk mendukung penelitian di NLP dan dilengkapi dengan serangkaian sumber daya untuk mempelajari NLP [13]. NLTK digunakan dalam penelitian ini untuk melakukan *text preprocessing* pada teks ulasan.

2.12 Amazon Shopping

Amazon Shopping merupakan aplikasi *e-commerce* yang dimiliki perusahaan Amazon, aplikasi ini tersedia di Google Play App dan Apple App Store. Amazon merupakan perusahaan multinasional yang berfokus pada *e-commerce, cloud computing, digital streaming* dan *artificial intelligence* yang didirikan oleh Jeff Bezos pada tahun 1994. Amazon Shopping memiliki pengguna aktif bulanan rata-rata sebanyak 98.07 juta pengguna yang menjadikannya aplikasi belanja paling populer kedua di Amerika serikat pada tahun 2021 [6].

2.13 Google Play Store

Google Play (sebelumnya dikenal sebagai Android Market) adalah etalase distribusi resmi untuk aplikasi Android dan media digital lainnya, seperti musik, film, dan buku, dari Google. Google Play tersedia di perangkat seluler dan tablet yang menjalankan sistem operasi Android, perangkat Chrome OS yang didukung, maupun melalui web. Selama kuartal pertama tahun 2022, lebih dari 3,29 juta aplikasi seluler tersedia di Google Play Store, turun hampir 30 persen dibandingkan kuartal sebelumnya yang berjumlah 4,67 juta aplikasi [4]. Jumlah aplikasi yang tersedia dapat berfluktuasi karena Google secara teratur menghapus aplikasi dengan beberapa alasan objektif.

2.14 Penelitian Terdahulu

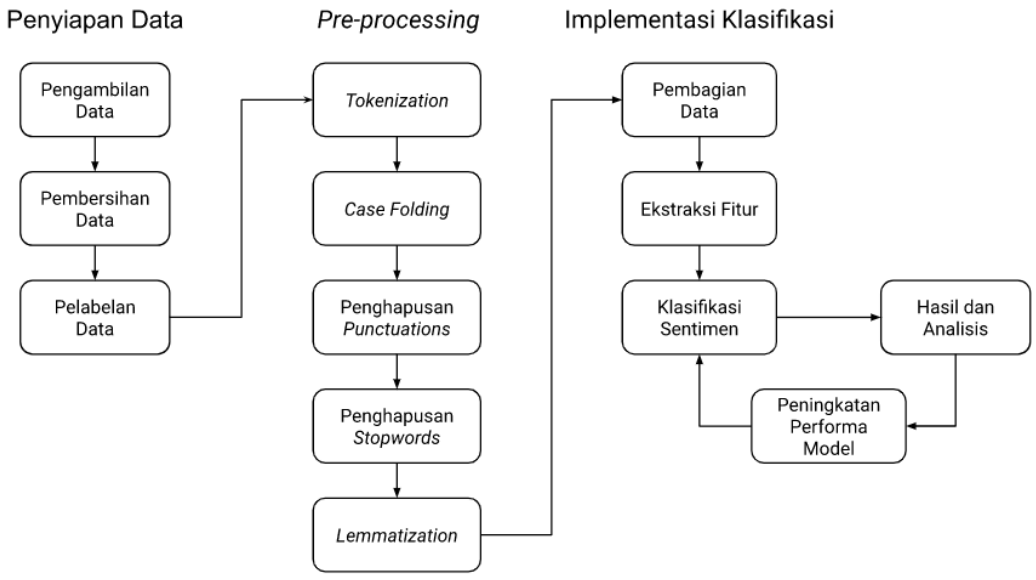
Beberapa penelitian serupa telah dilakukan sebelumnya, diantaranya penelitian [10] yang melakukan analisa sentimen menggunakan Naive Bayes untuk mengkategorikan ulasan positif dan negatif dari pelanggan Amazon atas kategori produk yang berbeda seperti ponsel, aksesoris, alat musik dan barang elektronik. Penelitian tersebut mampu menghasilkan akurasi rata-rata di atas 90% dengan mencoba beberapa simulasi yang berbeda seperti *cross validation*, rasio data latih-uji, dan teknik ekstraksi fitur yang berbeda.

Selanjutnya pada penelitian [27], berhasil dibangun sebuah sistem yang dapat mengklasifikasikan opini ulasan produk menjadi sentimen positif dan negatif dengan memanfaatkan rating. Dataset yang digunakan adalah data ulasan bahan makanan and makanan *gourmet* dari Amazon sebanyak 50.000 data. Hasil klasifikasi sistem dengan menerapkan metode *labelling average* dengan menggunakan algoritma Multinomial Naive Bayes mampu mengklasifikasikan sentimen positif dan negatif dengan akurasi sebesar 80,48%.

3. METODE PENELITIAN

3.1 Gambaran Alur Penelitian

Alur penelitian ini dimulai dari penyiapan data, pada tahapan ini dilakukan pengambilan data ulasan aplikasi Amazon Shopping pada Google Play Store, data yang dikumpulkan akan dibersihkan dan diberikan label sentimen. Data yang telah disiapkan selanjutnya akan melalui tahapan *preprocessing* agar data siap dimasukkan ke dalam model, pada tahapan ini akan dilakukan beberapa proses secara urut. Tahapan selanjutnya adalah dilakukan implementasi analisis sentimen, data ulasan akan dibagi menjadi set data latih dan uji yang kemudian dilakukan ekstraksi fitur untuk mengubah data mentah tersebut fitur numerik. Klasifikasi sentimen dilakukan menggunakan algoritma Naive Bayes, setelah hasil klasifikasi didapatkan, model kemudian akan dievaluasi dan dilakukan beberapa percobaan untuk meningkatkan performanya. Adapun tahapan penelitian ini digambarkan secara urut pada Gambar 3.1



Gambar 3.1 Gambaran alur penelitian.

3.2 Penyiapan Data Penelitian

Tahapan pertama dalam penelitian ini adalah penyiapan data, kegiatan penyiapan data dimulai dengan pengambilan data ulasan mentah pada aplikasi Amazon Shopping yang terdapat di Google Play Store. Setelah data dikumpulkan, data perlu dibersihkan terlebih dahulu, proses ini meliputi penghapusan data yang hilang atau terduplikat dan pemformatan ulang set data yang digunakan sehingga hanya berisi data-data yang dibutuhkan. Langkah terakhir adalah pelabelan data ulasan ke dalam dua kelas, yaitu positif dan negatif.

3.2.1 Pengambilan Data

Proses pengambilan data dilakukan menggunakan *library* google-play-scraper versi 1.2. *Library* ini menyediakan API untuk melakukan pengambilan data ulasan di Google Play Store dengan mudah menggunakan bahasa pemrograman Python tanpa *external dependencies*. Pengambilan data dilakukan pada tanggal 6 Juni 2022. Setelah proses pengambilan data, ulasan yang diambil akan dimasukkan ke dalam dataset yang kemudian disimpan dalam file berformat .csv dengan nama file "*reviews_raw.csv*".

3.2.2 Pembersihan Data

Proses pembersihan data dilakukan untuk memastikan keakuratan dan konsistensi data ulasan mentah yang disiapkan. Proses pembersihan data meliputi pengecekan dan penghapusan data jika terdapat konten data ulasan yang hilang maupun terduplikat. Ulasan yang digunakan sebagai masukkan hanya data ulasan yang bahasa inggris, sehingga ulasan yang tidak berbahasa inggris juga akan dihapuskan. Set data yang dimiliki juga perlu diformat, langkah ini dilakukan dengan penghapusan kolom-kolom yang tidak relevan dalam proses penelitian.

3.2.3 Pelabelan Data

Klasifikasi sentimen dilakukan menggunakan algoritma tipe *supervised machine learning*, dimana dibutuhkan data berlabel untuk melatih algoritma agar menghasilkan prediksi secara akurat. Proses pelabelan data dilakukan secara manual terhadap semua ulasan yang diambil. Label sentimen yang diberikan adalah negatif (-1) atau positif (1), label ini membentuk representasi dari kelas objek apa yang dimiliki ulasan dan membantu model *machine learning* untuk belajar mengidentifikasi sentimen pada data tanpa label. Sekumpulan aturan berikut digunakan untuk menentukan label sentimen:

- Jika ulasan cenderung negatif, label yang diberikan adalah negatif.
- Jika ulasan cenderung positif, label yang diberikan adalah positif.
- Jika ulasan cenderung netral, jumlah kata positif atau negatif terbanyak menentukan label sentimennya.
- Rating yang diberikan pengulas digunakan sebagai pertimbangan.

Data ulasan yang telah diberikan label sentimen kemudian akan disimpan dalam file csv baru bernama “reviews_labelled.csv”. Adapun contoh daftar ulasan beserta label sentimen yang diberikan dapat dilihat pada Tabel 3.1.

content	sentiment
You can find almost anything with this app. Very easy to navigate. Good response to search criteria.	1
I mean it has good quality stuff but it says free delivery but then you have to pay delivery and i think their should be more things that has free delivery but over all its a good app	1
App is useless. Half the time it doesn't start. Takes multiple attempts to open. App is up to date, cleared data and cache. I've even recently bought a new phone and the problem persists.	-1
Can't cancel email notifications from this app. I get push notifications and emails too. So annoying.	-1

Tabel 3.1 Contoh teks ulasan dan label sentimen yang diberikan.

3.3 *Text Preprocessing*

Data teks ulasan perlu dibersihkan kembali agar siap dimasukkan ke dalam model untuk melakukan klasifikasi. Hasil preprocessing juga akan direpresentasikan dalam bentuk *word clouds* untuk mendapatkan gambaran tentang distribusi kelas secara keseluruhan berdasarkan kata-kata yang paling sering muncul. Proses *text preprocessing* terdiri dari lima tahapan yang akan dilakukan secara urut, yaitu:

3.3.1 *Tokenization*

Sebelum teks review diproses lebih lanjut, teks perlu disegmentasi ke dalam unit linguistik seperti kata, tanda baca, angka, alfanumerik, dll. Hal ini penting agar makna teks dapat dengan mudah ditafsirkan dengan menganalisis kata-kata yang ada dalam teks. Tokenisasi memecahkan teks menjadi unit yang lebih kecil yang disebut *token* yang dapat berupa kata, karakter, atau subkata. Tahapan *tokenization* dilakukan menggunakan metode word_tokenize() dari NLTK yang merupakan fungsi pembungkus untuk memanggil fungsi tokenize() pada *instance* kelas TreebankWordTokenizer.

Langkah-langkah dalam melakukan *tokenization* menggunakan NLTK adalah:

1. Lakukan installasi *library* NLTK menggunakan pip3.
2. Lakukan instalasi model Punkt Tokenizer menggunakan NLTK,
3. Impor modul nltk.word_tokenize untuk melakukan tokenisasi,
4. Terapkan metode word_tokenize() pada setiap teks ulasan.

3.3.2 Case Folding

Tahapan *preprocessing* kedua yang dilakukan adalah *case folding*. *Case folding* dilakukan dengan mengubah semua karakter dalam dokumen menjadi huruf yang sama, baik huruf besar semua atau huruf kecil semua. Teks pada data ulasan yang digunakan tidak selalu konsisten dalam penggunaan huruf kapital, oleh karena itu akan dilakukan perubahan semua huruf besar (*uppercase*) menjadi huruf kecil (*lowercase*).

Langkah-langkah dalam melakukan *case folding* untuk konversi semua huruf menjadi huruf kecil adalah sebagai berikut:

1. Baca teks ulasan yang telah melalui proses tokenisasi.
2. Terapkan metode casefold() pada setiap teks ulasan.

3.3.3 *Punctuations Removal*

Daftar *punctuation* didapat dari modul string.punctuation yang berisi semua set karakter *punctuations* yang telah didefinisikan oleh modul string di Python 3. Adapun daftar tanda baca yang akan didefinisikan antara lain:

```
!"#$%&'()*+, -./:;<=>?@[\]^_`{|}~
```

Proses penghapusan *punctuations* dari teks ulasan dilakukan menggunakan metode str.replace(). Metode ini memerlukan dua parameter, yaitu string yang akan diganti, yang

dalam hal ini adalah daftar *punctuations* yang telah didefinisikan. Parameter kedua adalah karakter yang akan diganti, yang dalam hal ini adalah string kosong (“”) untuk menghapus *punctuation* tersebut dalam teks ulasan.

Langkah-langkah untuk menghapus *punctuations* dalam teks ulasan adalah:

1. Impor daftar *punctuations* dari modul string.punctuation.
2. Baca teks ulasan yang telah diubah ke huruf kecil.
3. Hapus *punctuations* yang terdapat pada teks ulasan dengan metode str.replace().

3.3.4 *Stopwords Removal*

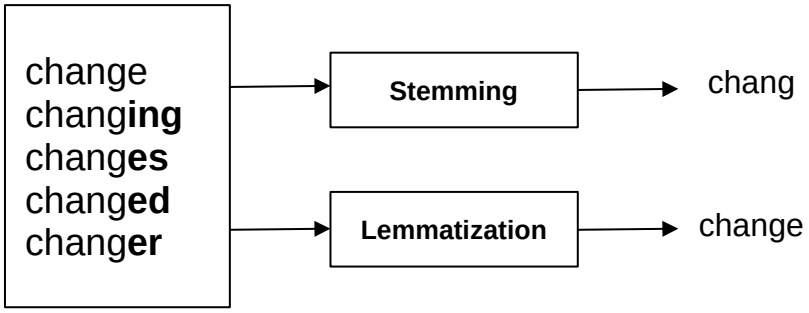
Proses penghapusan *stopwords* dimulai dengan mendefinisikan terlebih daftar kata-kata yang termasuk ke dalam *stopwords* dalam bahasa inggris. Daftar kosakata s*topwords* yang digunakan dikumpulkan dari beberapa library yaitu NLTK (179 kata), Gensim (337 kata), dan SpaCy (326 kata) yang dikombinasikan dan menghasilkan daftar *stopwords* sebanyak 413 kata. Penghapusan *stopwords* dilakukan dengan mengecek setiap kata dalam teks ulasan, jika kata tersebut terdapat dalam daftar *stopwords*, maka kata tersebut akan dihapuskan.

Langkah-langkah untuk menghapus *stopwords* dalam teks ulasan adalah:

1. Install *library* Gemsim dan SpaCy.
2. Impor dan muat daftar kosakata *stopwords* dari *library* NLTK (nltk.corpus.stopwords), Gemsim (gensim.parsing.preprocessing.STOPWORDS), dan SpaCy (spacy.Defaults.stop_words).
3. Simpan kombinasi *stopwords* yang dihasilkan dari ketiga *library*.
4. Baca data ulasan yang telah dihapus tanda bacanya.
5. Hapus kata yang terdapat dalam daftar *stopwords* pada setiap teks ulasan.

3.3.5 *Normalization*

Lemmatization dan *stemming* merupakan kasus khusus dari normalisasi yang memiliki tujuan sama untuk mengurangi bentuk-bentuk infleksional dan bentuk-bentuk kata yang terkait secara turunan menjadi bentuk dasar yang sama. Perbedaannya adalah *stemming* hanya menghilangkan atau menghentikan beberapa karakter terakhir dari sebuah kata, yang dapat menyebabkan arti dan ejaan yang salah. *Lemmatization* mempertimbangkan konteks dan mengubah kata ke bentuk dasar yang bermakna, yang disebut *lemma*. Contoh perbedaan hasil *stemming* dan *lemmatization* menggunakan kata “*change*” dan kata turunannya dapat dilihat pada Gambar 3.2.



Gambar 3.2 Contoh perbedaan hasil *stemming* dan *lemmatization*

Berdasarkan uraian tersebut, *lemmatization* dipilih sebagai metode untuk melakukan normalisasi kata agar mendapatkan bentuk kata yang lebih informatif. NLTK Lemmatizer yang digunakan dalam tahapan ini adalah dengan metode WordNetLemmatizer (nltk.stem.WordNetLemmatizer.lemmatize) yang akan dibantu menggunakan tag PoS, hal ini penting dilakukan karena tanpa memahami apakah sebuah kata digunakan sebagai kata kerja, kata benda, atau kata sifat dalam sebuah kalimat, hasil lematisasi akan kurang efektif. Metode WordNetLemmatizer memerlukan parameter "*wordnet*" dan "*pos*" untuk melakukan lematisasi. Parameter *wordnet* membutuhkan nilai tag PoS dalam kalimat, tag tersebut menggunakan konstanta PoS yang terdiri dari "a", "s", "r", "n", "v" untuk untuk tag ADJ (*Adjective*), ADJ SAT (*Adjective Sattelite*), ADV (*Adverb*), NOUN (*Noun*), VERB (*Verb*).

Langkah-langkah untuk melakukan *lemmatization* dengan bantuan tag PoS adalah sebagai berikut:

1. Impor *library* nltk.stem.WordNetLemmatizer dan nltk.corpus.wordnet
2. Dapatkan tag PoS untuk setiap token pada teks ulasan
3. Ubah tag PoS menjadi tag wordnet (ADJ/VERB/NOUN/ADV)
4. Lakukan lematisasi dengan fungsi lemmatize() pada *instance* WordNetLemmatizer() menggunakan parameter tag wordnet untuk setiap tokennya.

3.4 Implementasi Klasifikasi Sentimen

Klasifikasi sentimen dilakukan setelah data ulasan telah melalui proses *text preprocessing*. Tahapan ini dimulai dari melakukan ekstraksi fitur pada teks ulasan menggunakan representasi *Bag-of-Words* dan pembobotan kata TF-IDF, pembagian data menjadi set data latih dan uji, melatih model *machine learning*, kemudian memprediksi sentimen dan mengujinya menggunakan model yang sudah dilatih.

3.4.1 Ekstraksi Fitur

Data teks ulasan perlu direpresentasikan menjadi data numerik sebelum digunakan untuk melatih model *machine learning* dan mengujinya, hal ini dapat dilakukan dengan ekstraksi fitur. Ekstraksi Fitur dilakukan dengan menggunakan Bag-of-Words dan TF-IDF yang kemudian akan dibandingkan hasil keduanya berdasarkan akurasi klasifikasi sentimen yang dihasilkan. Langkah pertama dalam ekstraksi fitur adalah mengambil daftar semua kata dalam dokumen, dalam contoh berikut, terdapat tiga baris dokumen yang akan diperlakukan sebagai corpus.

Doc1: "Amazon Prime is amazing"
Doc2: "Best place to shop"
Doc3: "Amazing choice to shop"

Berdasarkan contoh corpus yang diberikan, daftar kosakata unik yang didapatkan adalah sebanyak 9 kata, yaitu “amazing”, “*amazon*”, “*best*”, “*choice*”, “*is*”, “*place*” “*prime*”, “*shop*”, dan “*to*” dari corpus yang berisi 12 kata. *Bag-of-Words* dilakukan dengan menghitung berapa kali setiap kata muncul di setiap dokumen. Dokumen teks diubah menjadi vektor yang akan digunakan sebagai masukan atau keluaran untuk model *machine learning*. Metode penilaian yang paling sederhana adalah dengan menandai keberadaan kata sebagai nilai *boolean* jika kata tersebut terdapat pada dokumen, 0 jika tidak ada, 1 jika ada.

Doc	amazing	amazon	best	choice	is	place	prime	shop	to
1	1	1	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1	1
3	1	0	0	1	0	0	0	1	1

Tabel 3.2 Representasi *bag-of-words* untuk corpus yang diberikan.

Berikut adalah representasi setiap dokumen dengan panjang 9 kata sesuai dengan jumlah kosakata yang didapatkan, dengan tiap posisi dalam vektor mewakili nilai setiap kata.

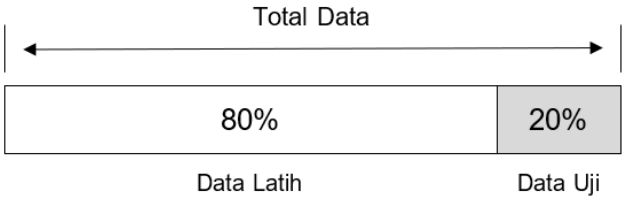
Doc1: [1 1 0 0 1 0 1 0 0]
Doc2: [0 0 1 0 0 1 0 1 1]
Doc3: [1 0 0 1 0 0 0 1 1]

Permasalahan dengan penilaian berdasarkan frekuensi kemunculan suatu kata adalah kata-kata yang sangat sering muncul tidak selamanya mengandung informasi yang relevan dibandingkan kata yang lebih jarang muncul namun memiliki informasi yang lebih relevan. Berdasarkan hal tersebut, akan digunakan pendekatan lain yaitu TF-IDF yang akan menskala ulang frekuensi kata berdasarkan seberapa sering kata tersebut muncul dalam dokumen, hal ini dimaksudkan untuk mengukur seberapa relevan suatu istilah dalam dokumen tertentu. Array berikut mewakili vektor yang dibuat dengan ketiga dokumen menggunakan vektorisasi TF-IDF:

```
[ [ 0.4020402   0.52863461  0.          0.          0.52863461
    0.          0.52863461  0.          0. ]
  [ 0.          0.          0.5628291   0.          0.
    0.5628291   0.          0.4280460   0.42804604 ]
  [ 0.4598535   0.          0.          0.60465213  0.
    0.          0.          0.4598535   0.45985353]]
```

3.4.2 Pembagian Data

Saat melatih model machine learning, model tidak dapat dilatih menggunakan satu set data, diperlukan adanya set data lain untuk mengukur kinerja dari model yang digunakan. Data ulasan akan dibagi menjadi set data latih dan uji. Data latih digunakan untuk melatih dan membuat model mempelajari fitur/pola tersembunyi dalam data. Sementara data uji merupakan data terpisah yang digunakan untuk menguji performa model setelah menyelesaikan pelatihan. Setelah melalui tahap prediksi, model dievaluasi dengan membandingkannya dengan keluaran aktual yang ada dalam set data uji. Data dibagi dengan rasio 80:20, dimana 80% digunakan untuk data latih, dan 20% untuk data uji.



Gambar 3.3 Ilustrasi pembagian data menjadi data latih dan uji dengan rasio 80:20.

3.4.3 Klasifikasi Sentimen

Proses klasifikasi sentimen dilakukan menggunakan sekelompok algoritma Naive Bayes, algoritma tersebut merupakan metode pembelajaran *supervised* berdasarkan penerapan teorema Bayes dengan asumsi independensi fitur yang kuat (*naive*). Pengimplementasian klasifikasi ini dilakukan menggunakan modul dan kelas yang telah disediakan oleh *library* scikit-learn (sklearn.naive_bayes). Adapun algoritma bayesian yang akan digunakan adalah Multinomial, Bernoulli, Gaussian, dan Complement Naive Bayes.

3.5 Evaluasi Model

Hasil klasifikasi sentimen yang didapatkan akan dievaluasi untuk mengetahui bagaimana kinerja dari model yang digunakan. Evaluasi model disajikan menggunakan tabel *confusion matrix*, indikator-indikator yang digunakan adalah: *True Positive* (TP) yang menunjukkan jumlah kelas negatif yang benar, *False Positive* (FN) untuk jumlah kelas positif yang salah, *True Negative* (TN) untuk jumlah kelas negatif yang salah, dan *False Negative* (FN) untuk jumlah kelas positif yang benar. Setelah nilai indikator tersebut didapatkan, nilai akurasi, *precision*, *recall* dan *f1-score* dapat dihitung

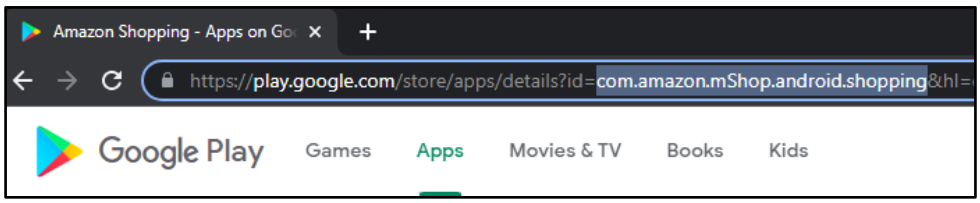
3.6 Peningkatan Performa Model

Setelah model dievaluasi dan kinerja dari model-model yang digunakan, beberapa percobaan akan dilakukan untuk meningkatkan kinerja dari model. Percobaan ini akan dilakukan menggunakan model yang menghasilkan akurasi terbesar. Upaya yang dilakukan untuk peningkatan akurasi ini diantaranya adalah menggunakan pembobotan TF-IDF untuk melakukan ekstraksi fitur, akurasi model juga diuji berdasarkan perbedaan n-gram dan perbedaan persentase pembagian set data latih dan uji. Klasifikasi juga akan dilakukan menggunakan algoritma *machine learning* lainnya yang bukan termasuk ke dalam keluarga pengklasifikasi Naive Bayes. Penghitungan performa klasifikasi ini ditunjukkan untuk membandingkan performa pengklasifikasi Naive Bayes dengan algoritma berbasis *machine learning* lainnya. Algoritma non-bayesian yang digunakan adalah Support Vector, Decision Tree, Random Forest, dan Logistic Regression.

4. HASIL DAN PEMBAHASAN

4.1 Pengambilan Data

Data-data ulasan diambil menggunakan *google-play-scraper*, fungsi yang digunakan adalah *reviews()* yang hanya mengambil data ulasan pada aplikasi yang dituju. Beberapa parameter wajib didefinisikan sebelum mengambil data ulasan, yaitu parameter *id*, *lang*, dan *country*. Parameter *id* yang merupakan Google Play ID diperlukan sebagai pengenal unik untuk aplikasi yang akan diambil data ulasannya. ID ini dapat ditemukan di parameter *id* dalam URL dari halaman web Play Store untuk aplikasi yang diinginkan. Sebagai contoh, "com.amazon.mShop.android.shopping" adalah Google Play ID untuk aplikasi Amazon Shopping.



Gambar 4.1 Google Play ID untuk aplikasi Amazon Shopping.

Pengaturan untuk bahasa dan negara yang digunakan adalah pengaturan default, yaitu bahasa "en" (English) dan negara "us" (United States). Selain itu, data ulasan juga diurutkan berdasarkan tanggal terbaru ditulisnya ulasan. Source code untuk melakukan pengumpulan data review dapat dilihat pada Gambar 4.2.

```
# Scraping amazon reviews from play store
from google_play_scraper import Sort, reviews_all

amazon_reviews = reviews(
    "com.amazon.mShop.android.shopping",
    lang="en",
    country="us",
    sort=Sort.NEWEST
)
```

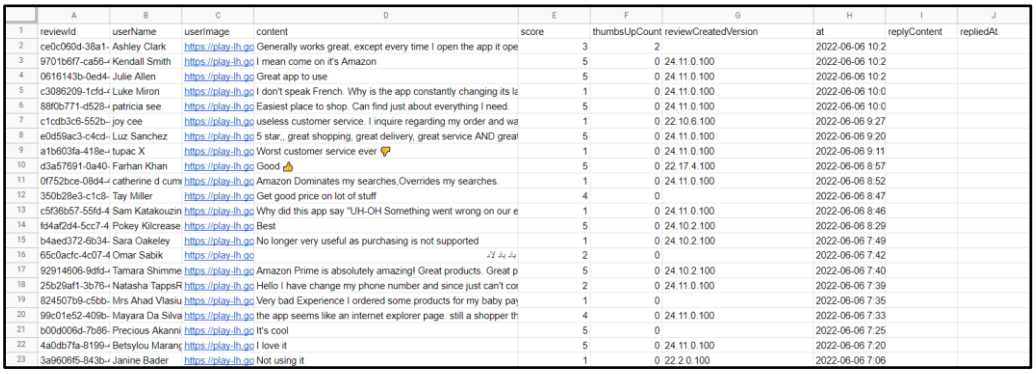
Gambar 4.2 *Source code* untuk mengambil ulasan aplikasi Amazon Shopping

Setelah melalui proses pengambilan data, dihasilkan data berupa kumpulan *dictionary* berisi informasi setiap ulasan yang ditulis. Kolom-kolom data yang dihasilkan untuk setiap ulasan dijelaskan pada Tabel 4.1.

Kolom	Keterangan
at	Tanggal saat ulasan diberikan
content	Teks ulasan yang diberikan pengulas
repliedAt	Tanggal saat pesan balasan dikirim
replyContent	Pesan balasan yang diberikan pihak pengembang kepada pengulas
reviewCreatedVersion	Versi dari aplikasi saat ulasan diberikan
reviewId	Pengenal unik untuk ulasan
score	Rating yang diberikan pengulas
thumbsUpCount	Jumlah pengguna lain yang telah memberikan ulasan acungan jempol
userImage	URL untuk gambar profil pengulas
userName	Nama pengguna penulis ulasan

Tabel 4.1 Kolom yang dihasilkan dalam proses pengumpulan data.

Menggunakan parameter yang diberikan, dihasilkan data ulasan mentah sebanyak 2234 ulasan yang ditulis dalam 6 bulan terakhir saat data diambil, yaitu pada rentang 01-05-2022 sampai 06-06-2022. Data tersebut kemudian disimpan dalam format DataFrame untuk mempermudah pengolahan lebih lanjut dan juga disimpan dalam format file .csv dengan nama "*reviews_raw.csv*". Hasil data ulasan mentah yang diambil dapat dilihat pada Gambar 4.3.



	A	B	C	D	E	F	G	H	I	J
1	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt
2	ce0c060d-38a1-	Ashley Clark	https://play-lh.go	Generally works great, except every time I open the app it ope	3	2		2022-06-06 10:2		
3	9701b6f7-ca56-4	Kendall Smith	https://play-lh.go	I mean come on it's Amazon	5	0	24.11.0.100	2022-06-06 10:2		
4	0616143b-0ed4-	Julie Allen	https://play-lh.go	Great app to use	5	0	24.11.0.100	2022-06-06 10:2		
5	c3086209-1cfd-4	Luke Miron	https://play-lh.go	I don't speak French. Why is the app constantly changing its la	1	0	24.11.0.100	2022-06-06 10:0		
6	88f0b771-d528-4	patricia see	https://play-lh.go	Easiest place to shop. Can find just about everything I need.	5	0	24.11.0.100	2022-06-06 10:0		
7	c1cdb3c6-552b-	joy cee	https://play-lh.go	useless customer service. I inquire regarding my order and wa	1	0	22.10.6.100	2022-06-06 9:27		
8	e0d59ac3-c4cd-	Luz Sanchez	https://play-lh.go	5 star,, great shopping, great delivery, great service AND great	5	0	24.11.0.100	2022-06-06 9:20		
9	a1b603fa-418e-4	tupac X	https://play-lh.go	Worst customer service ever 👎	1	0	24.11.0.100	2022-06-06 9:11		
10	d3a57691-0a40-	Farhan Khan	https://play-lh.go	Good 👍	5	0	22.17.4.100	2022-06-06 8:57		
11	0f752bce-08d4-4	catherine d cum	https://play-lh.go	Amazon Dominates my searches,Overrides my searches.	1	0	24.11.0.100	2022-06-06 8:52		
12	350b28e3-c1c8-	Tay Miller	https://play-lh.go	Get good price on lot of stuff	4	0		2022-06-06 8:47		
13	c5f36b57-55fd-4	Sam Katakouzin	https://play-lh.go	Why did this app say "UH-OH Something went wrong on our e	1	0	24.11.0.100	2022-06-06 8:46		
14	fd4af2d4-5cc7-4	Pokey Kilcrease	https://play-lh.go	Best	5	0	24.10.2.100	2022-06-06 8:29		
15	b4aed372-6b34-	Sara Oakeley	https://play-lh.go	No longer very useful as purchasing is not supported	1	0	24.10.2.100	2022-06-06 7:49		
16	65c0acfc-4c07-4	Omar Sabik	https://play-lh.go	يد يد لا	2	0		2022-06-06 7:42		
17	92914606-9dfd-4	Tamara Shimme	https://play-lh.go	Amazon Prime is absolutely amazing! Great products. Great p	5	0	24.10.2.100	2022-06-06 7:40		
18	25b29af1-3b76-4	Natasha TappsR	https://play-lh.go	Hello I have change my phone number and since just can't cor	2	0	24.11.0.100	2022-06-06 7:39		
19	824507b9-c5bb-	Mrs Ahad Vlasiu	https://play-lh.go	Very bad Experience I ordered some products for my baby pay	1	0		2022-06-06 7:35		
20	99c01e52-409b-	Mayara Da Silva	https://play-lh.go	the app seems like an internet explorer page. still a shopper th	4	0	24.11.0.100	2022-06-06 7:33		
21	b00d006d-7b86-	Precious Akanni	https://play-lh.go	It's cool	5	0		2022-06-06 7:25		
22	4a0db7fa-8199-4	Betsylou Maran	https://play-lh.go	I love it	5	0	24.11.0.100	2022-06-06 7:20		
23	3a9606f5-843b-4	Janine Bader	https://play-lh.go	Not using it	1	0	22.2.0.100	2022-06-06 7:06		

Gambar 4.3 Dataset *reviews_raw.csv*.

4.2 Pembersihan Data

Proses pembersihan meliputi penghapusan ulasan yang tidak berbahasa inggris dan ulasan yang terlalu singkat dan tidak bermakna. Kelengkapan data akan dicek, jika terdapat ulasan yang terduplikat ataupun hilang, data tersebut akan dihapus. Kolom *content* yang berisi ulasan yang ditulis dan kolom *score* yang berisi rating yang diberikan pengulas diperlukan untuk proses selanjutnya. Kolom *score* dibutuhkan karena akan digunakan sebagai pertimbangan dalam memberikan skor sentimen terhadap ulasan yang ditulis dalam proses pelabelan data. Sebanyak 88 buah data ulasan telah dihapus dalam tahapan pembersihan sehingga menyisakan sebanyak 2147 ulasan. Data ulasan yang telah dibersihkan kemudian disimpan dalam file csv dengan nama "reviews_clean.csv". Hasil dataset yang sudah dibersihkan dapat dilihat pada Gambar 4.4.

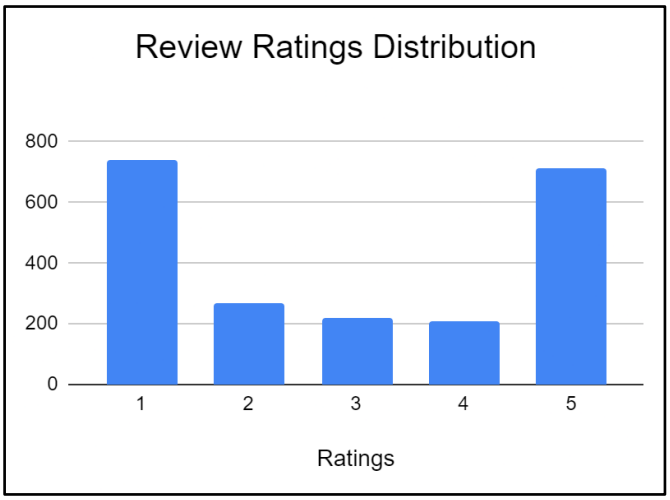
	content	score
0	Generally works great, except every time I ope...	3
1	Amazon Prime is absolutely amazing! Great prod...	5
2	Hello I have change my phone number and since ...	2
3	Outstanding orders? Too difficult to find. Sma...	3
4	I like it alot, very easy, i love the variety ...	4
...	...	...
2142	I buy from Amazon all the time and am generall...	4
2143	Just lil question why do u guys made me update...	1
2144	A bit slow on my phone otherwise very similar ...	4
2145	I made a legitimate order after verifying thei...	1
2146	They make Unsubscribing prime membership very ...	1

2147 rows × 2 columns

Gambar 4.4 Dataset *reviews_clean*.

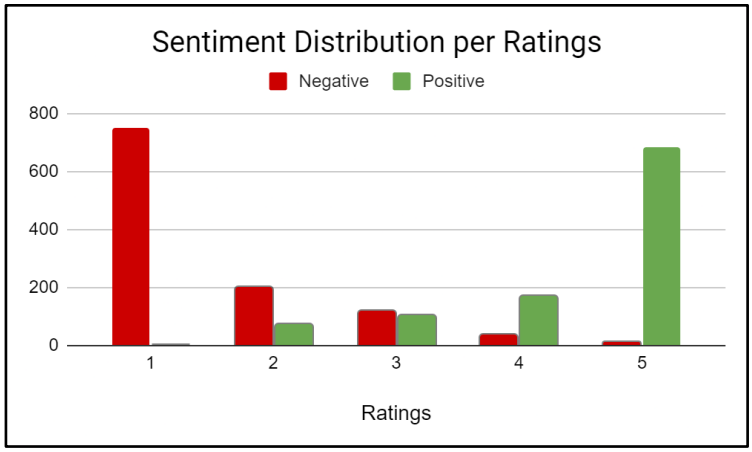
4.3 Pelabelan Data

Pelabelan data dilakukan secara manual dengan beberapa aturan yang telah didefinisikan pada Bab 3.2.3. Rating yang diberikan pengulas menjadi referensi penting bagi untuk menentukan sentimen yang akan diberikan. Adapun distribusi rating dari keseluruhan dataset dapat dilihat pada Gambar 4.5.



Gambar 4.5 Distribusi rating yang diberikan pengulas.

Hasil akhir dari proses pelabelan menghasilkan data sebanyak 991 ulasan bersentimen positif dan 1156 bersentimen negatif dengan persentase masing-masing 46.2% dan 53.8% dari keseluruhan data. Distribusi sentimen berdasarkan rating adalah sebagai berikut:



Gambar 4.6 Distribusi sentimen ulasan berdasarkan rating.

Label sentimen yang diberikan diisikan pada kolom baru baru bernama "sentiment", setelah semua teks ulasan diberikan label sentimen, dataset akan disimpan dalam file csv baru bernama "reviews_labelled.csv". Isi dari dataset yang berisi teks ulasan yang telah diberikan label sentimen positif (1) dan negatif (-1) dapat dilihat pada Gambar 4.7.

	content	score	sentiment
0	Generally works great, except every time I ope...	3	-1
1	Amazon Prime is absolutely amazing! Great prod...	5	1
2	Hello I have change my phone number and since ...	2	-1
3	Outstanding orders? Too difficult to find. Sma...	3	-1
4	I like it alot, very easy, i love the variety ...	4	1
...	...	...	...
2142	I buy from Amazon all the time and am generall...	4	1
2143	Just lil question why do u guys made me update...	1	-1
2144	A bit slow on my phone otherwise very similar ...	4	-1
2145	I made a legitimate order after verifying thei...	1	-1
2146	They make Unsubscribing prime membership very ...	1	-1

2147 rows × 3 columns

Gambar 4.7 Dataset *reviews_labelled.csv*.

4.4 Hasil *Text Pre-Processing*

Data ulasan mentah akan melalui serangkaian tahapan *preprocessing* secara urut sebelum dapat dimasukkan ke dalam model. Tahapan-tahapan tersebut antara lain: *tokenization*, *case folding*, *punctuations removal*, *stopwords removal*, dan *lemmatization* yang akan dilakukan secara urut. Tabel 4.2 memperlihatkan contoh perubahan yang terjadi pada salah satu teks, dari teks ulasan awal sampai ke hasil tahap *lemmatization*.

Tahapan	Hasil
Teks Awal	“Amazon Prime is absolutely amazing! Great prices. Shipped right to your door!”
Tokenization	["Amazon", "Prime", "is", "absolutely", "amazing", "!", "Great", "prices", ".", "Shipped", "right", "to", "your", "door", "!"]
Case Folding	“amazon prime is absolutely amazing! great prices. shipped right to your door!”
Punctuations Removal	“amazon prime is absolutely amazing great prices shipped right to your door”
Stopwords Removal	“amazon prime absolutely amazing great prices shipped right door”
Normalization	“amazon prime absolutely amazing great price shipped right door”

Tabel 4.2 Contoh perubahan pada teks ulasan pada setiap tahapan *preprocessing*.

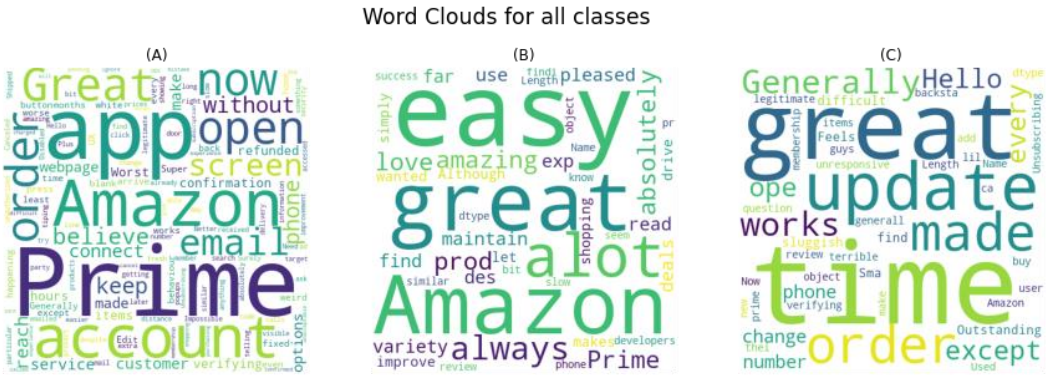
Teks ulasan akhir yang dihasilkan setelah melalui semua proses *preprocessing* beserta label sentimennya disimpan menjadi file csv baru “*reviews_postprocessing.csv*”, set data tersebut nantinya akan digunakan sebagai sebagai masukan dalam proses klasifikasi sentimen. Adapun isi dari dataset tersebut yang menyimpan teks hasil proses *preprocessing* dapat dilihat pada Gambar 4.8.

	content	sentiment
0	generally work great time open app open blank white scre...	-1
1	amazon prime absolutely amaze great product great price ...	1
2	hello change phone number connect tomy account plus tip ...	-1
3	outstanding order difficult small upgrade sort outstandi...	-1
4	like alot easy love variety item choose	1
...	...	...
2142	buy amazon time generally happy lately receive item have...	1
2143	lil question u guy update payment product id pay product...	-1
2144	bit slow phone similar webpage like delivery option visi...	-1
2145	legitimate order verify account security email receive e...	-1
2146	unsubscribing prime membership difficult charge prime pu...	-1

2147 rows × 2 columns

Gambar 4.8 Dataset *reviews_postprocessing.csv*.

Hasil preprocessing kemudian divisualisasikan dalam bentuk *wordcloud* yang memberikan representasikan daftar kata yang sering muncul untuk setiap kelas dalam data ulasan. Gambar 4.9 Memvisualisasikan *word clouds* untuk semua kelas (A), kelas positif (B), dan kelas negatif (C).



Gambar 4.9 Representasi *word clouds* untuk setiap kelas.

4.5 Ekstraksi Fitur

Ekstraksi fitur dilakukan menggunakan metode *Bag-of-Words* dan TF-IDF. Pembuatan fitur dengan *Bag-of-Worlds* dilakukan menggunakan fungsi CountVectorizer(), sedangkan metode TF-IDF dilakukan menggunakan fungsi TfidfVectorizer() yang disediakan dalam *library* sklearn dengan jumlah n-gram yang digunakan adalah 1 (unigram).

```
# Load review dataset
df_reviews = pd.read_csv("reviews_post-processing.csv")
reviews = df_reviews["content"]
sentiment = df_reviews["sentiment"]

# creating document-term matrix
# CountVectorizer
cv = CountVectorizer(ngram_range=(1,1))
cv_fit.fit_transform(reviews)
# TfidfVectorizer
tfidf = TfidfVectorizer(ngram_range=(1,1))
tfidf_fit = tfidf.fit_transform(reviews)
```

Gambar 4.10 *Source code* proses ekstraksi fitur.

Tabel 4.3 menunjukkan daftar kata-kata dengan frekuensi kemunculan terbesar yang didapatkan melalui proses *Bag-of-Words*. Sedangkan daftar istilah dengan nilai pembobotan TF-IDF terbesar dalam data ulasan dapat dilihat pada Tabel 4.4.

No	Kata	Frekuensi
1	amazon	1605
2	app	1436
3	order	716
4	item	651
5	time	584
6	prime	423
7	love	392
8	easy	382
9	service	371
10	use	369

Tabel 4.3 Daftar 10 kata dengan frekuensi kemunculan terbanyak.

No	Istilah	Skor TF-IDF
1	open	0.353
2	screen	0.248
3	popups	0.224
4	behaviour	0.224
5	ux	0.213
6	surely	0.205
7	target	0.205
8	generally	0.174
9	press	0.171
10	white	0.171

Tabel 4.4 Daftar 10 istilah dengan nilai TF-IDF terbesar.

4.6 Pembagian Data

Pembagian data dilakukan menggunakan fungsi train_test_split yang disediakan oleh *library* Sklearn. Parameter *train_size* yang digunakan adalah sebesar 0.2, yang merepresentasikan proporsi pemisahan data untuk set data uji sebanyak 20% dari keseluruhan set data, yang sisa datanya sebanyak 80% akan dimasukkan ke dalam set data latih. Parameter *random_state* didefinisikan agar setiap kali model dieksekusi, data acak yang dipisahkan untuk data latih dan uji akan selalu sama.

```
# CountVectorizer
cv_split = train_test_split(cv_fit, sentiment,
                            test_size=0.2, random_state=5)
# TfidfVectorizer
tfidf_split = train_test_split(tfidf_fit, sentiment,
                               test_size=0.2, random_state=5)
```

Gambar 4.11 *Source code* untuk proses pembagian data.

Setelah melalui tahapan pembagian data, data ulasan yang berjumlah 2147 dibagi menjadi dua bagian, yaitu 1717 buah data latih yang akan digunakan untuk melatih model *machine learning* untuk proses klasifikasi sentimen dan 429 buah data uji yang akan digunakan sebagai mengevaluasi performa model. Gambar 4.10 menunjukkan data ulasan yang telah dibagi menjadi set data latih (A) dan set data uji (B)

```
0       Difficult to find what I want. And the tracking doesn't ...
1       This app over the last year or two has become extremely ...
2       Feels sluggish, unresponsive at times, backstack doesn't...
3       so quick, easy & reliable, as long as you use common sen...
4       I do 99% of my shopping here & have for several years no...
                                ...
1712    This app used to run fine, but now you can't buy digital...
1713    Service: 4-5, app: around 1.5. Looks like a 90s webpage,...
1714    Just wanted to let the developers know of a problem that...
1715    Convenient way to get all things I need delivered to my ...
1716    Why did the app change my language? I can't spend money ...
Name: content, Length: 1717, dtype: object
```

(A) Data Latih

```
0       Sucks lately. Both the web page and the app suck after l...
1       Amazon has made shopping during this time of uncertainty...
2       It is the worst, it doesn't even want to open. The app d...
3       Love it when it works but it keeps signing me out. It do...
4       Amazon shopping is that "right from the convenience of y...
                                ...
425     Recently started to automatically switch to French. Solu...
426     Terrible costumer service! I returned a tablet and Amazo...
427     The app is so-so, a little glitchy at times, and half of...
428     Can't exactly trust the tracking info. (Least not for me...
429     Order history purposely broken so it makes it more diffi...
Name: content, Length: 430, dtype: object
```

(B) Data Uji

Gambar 4.12 Data ulasan yang telah dibagi menjadi dua set, yaitu data latih dan data uji.

4.7 Klasifikasi Sentimen dengan Naive Bayes

Pengimplementasian klasifikasi sentimen menggunakan Naive Bayes dimulai dengan mengkompilasi model, hal ini dilakukan mengimpor modul dan kelas yang telah dikompilasi dari sklearn. Daftar kelas yang diimpor dari *library* sklearn adalah BernoulliNB, ComplementNB, GaussianNB, dan MultinomialNB. Setelah model-model yang akan digunakan didefinisikan, langkah selanjutnya adalah melakukan *fitting* pada fungsi klasifikasi dengan memasukkan data latih pada model. Langkah terakhir adalah melakukan prediksi label sentimen menggunakan data uji.

```
# importing models
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB

def fitting_model(classifier, X_train, Y_train):
    return classifier.fit(X_train.todense(), Y_train)

def perform_prediction(classifier, X_test):
    return classifier.predict(X_test.todense())

# specify the classifiers
classifiers = {
    "Bernoulli NB": BernoulliNB(),
    "Complement NB": ComplementNB(),
    "Gaussian NB": GaussianNB(),
    "Multinominal NB": MultinomialNB()

}
```

Gambar 4.13 *Source code* implementasi klasifikasi sentimen dengan Naive Bayes.

4.8 Pengujian Model

Model yang telah dilatih dan dapat memprediksi sentimen digunakan sebagai penentu untuk pengujian data menggunakan set data uji sebanyak 426 ulasan. Setiap ulasan diprediksi apakah termasuk ke dalam kelas positif atau negatif. Berikut adalah contoh hasil prediksi klasifikasi sentimen pada set data uji yang didapatkan menggunakan algoritma Multinomial Naive Bayes.

Teks	Prediksi	Aktual
Sucks lately. Both the web page and the app suck after l…	-1	-1
Amazon has made shopping during this time of uncertainty…	1	1
It is the worst, it doesn't even want to open. The app d…	-1	-1
Love it when it works but it keeps signing me out. It do…	1	-1
Amazon shopping is that 'right from the convenience of y…	1	1

Tabel 4.5 Pengujian menggunakan set data uji.

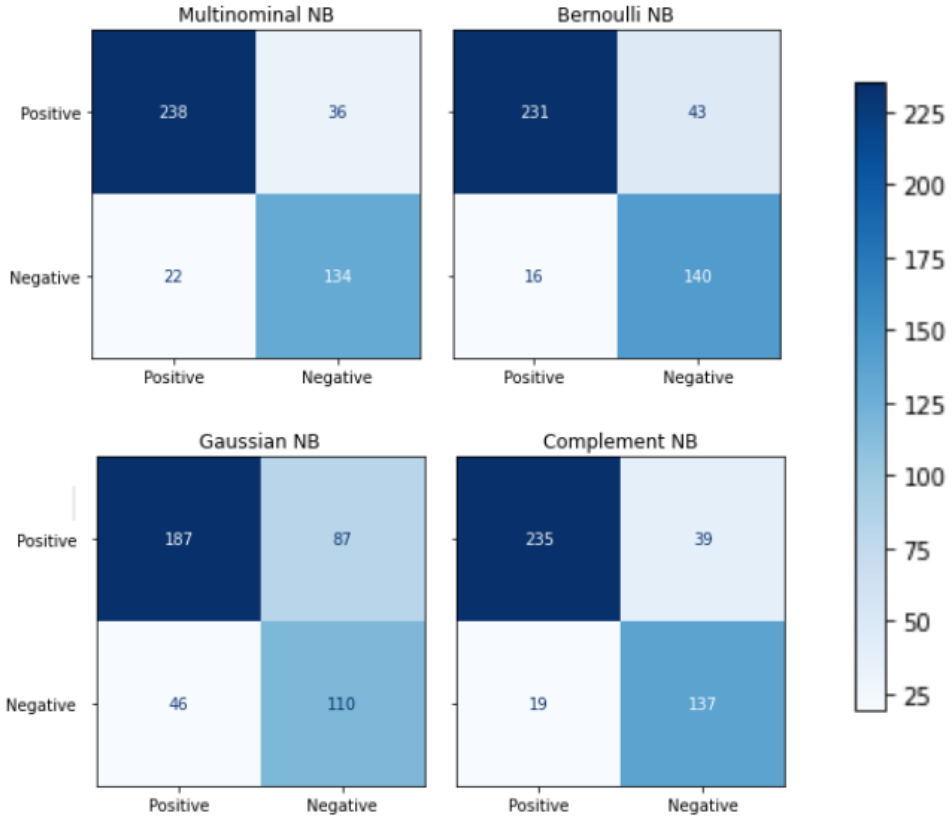
Klasifikasi sentimen juga dilakukan menggunakan masukan berupa data ulasan baru yang tidak terdapat pada set data uji yang dimiliki. Adapun hasil pengujian tersebut dapat dilihat pada Tabel 4.6.

Teks	Prediksi	Aktual
App never loads, need to use the website instead.	-1	-1
I shop amazing all the time absolutely love the app	1	1
Great and excellent app for a secure shopping	1	1
Frustrating considering you can’t purchase kindle books	-1	-1
You can find almost everything you need or want...	1	1

Tabel 4.6 Pengujian menggunakan set data baru.

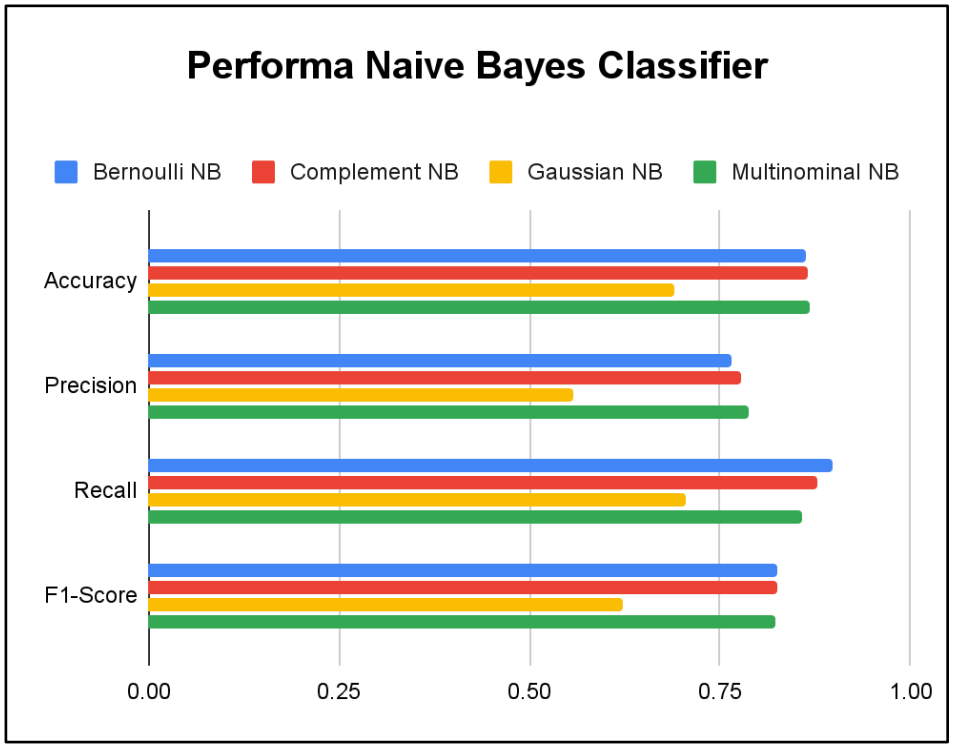
4.9 Evaluasi Hasil

Model yang telah dilatih dan diuji perlu dievaluasi untuk menghitung performa klasifikasi yang dihasilkan. Proses evaluasi dari model dilakukan menggunakan *confusion matrix*. Performa pengklasifikasi dapat dilihat setelah mengetahui besarnya nilai akurasi, *precision*, *recall*, dan *f1-score* yang dihitung menggunakan nilai *true positive* (TP), *true negative* (TN), *false positive* (FP), false negative (FN) dari *confusion matrix*. Performa model yang baik ditentukan dengan besarnya nilai-nilai tersebut, semakin besar maka semakin baik. *Confusion matrix* yang dihasilkan dari setiap model dapat dilihat pada Gambar 4.14.



Gambar 4.14 *Confusion matrix* untuk setiap model Naive Bayes.

Nilai TP, TN, FP, FN yang didapatkan dari *confusion matrix* setiap model selanjutnya digunakan untuk menghitung nilai akurasi, *precision*, *recall*, dan *f1-score*. Setelah melalui proses perhitungan, akurasi tertinggi dari keempat algoritma Naive Bayes yang didapatkan adalah sebesar 86.74% dengan menggunakan Multinomial Naive Bayes. Sedangkan Gaussian Naive Bayes menghasilkan akurasi terkecil, yaitu 69.07%. Rata-rata keempat model Naive Bayes menghasilkan akurasi, *precision*, *recall* dan *f1-score* masing-masing sebesar 82.15%, 72.25%, 83.49% dan 77.41%. Berikut adalah perbandingan performa klasifikasi dari keempat model.



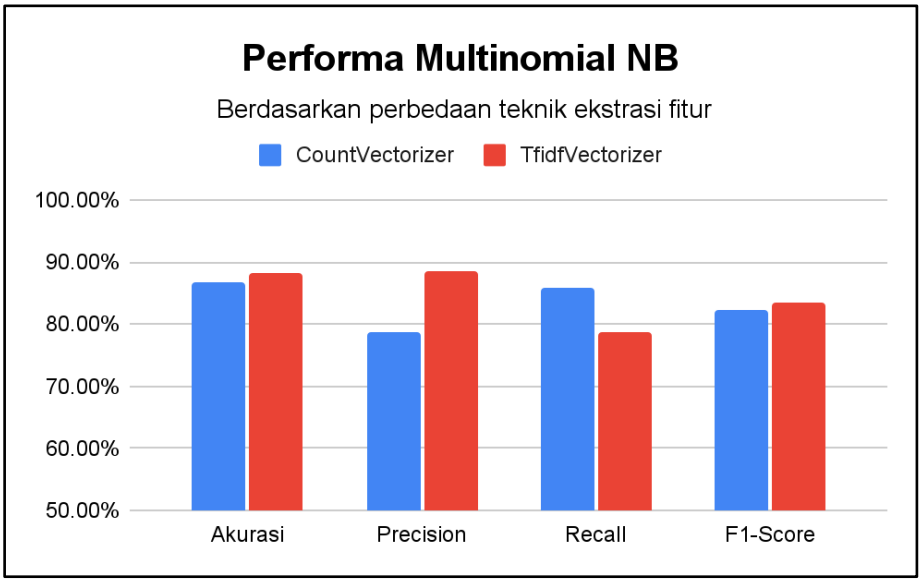
Gambar 4.15 Perbandingan performa klasifikasi algoritma Naive Bayes.

4.10 Hasil Percobaan Peningkatan Akurasi Model

Setelah mendapatkan hasil evaluasi keempat model Naive Bayes, didapatkan model dengan nilai akurasi terbesar, yaitu Multinomial Naive Bayes sebesar 86.74%. Model tersebut kemudian akan dilakukan proses lanjutan untuk percobaan peningkatan akurasinya.

4.10.1 Klasifikasi Menggunakan TF-IDF

Percobaan peningkatan akurasi model yang pertama dilakukan dengan menggunakan pembobotan TF-IDF untuk melakukan ekstraksi fitur yang sebelumnya dilakukan menggunakan metode *Bag-of-Word*. Setelah membandingkan kedua teknik ekstraksi fitur, didapatkan akurasi Multinomial NB dengan menggunakan seleksi fitur TF-IDF menghasilkan akurasi sebesar 88.37%, lebih besar dibandingkan metode *Bag-of-Word* yang sebesar 86.74%. Gambar 4.16 Menunjukkan data perbandingan nilai akurasi, *precision*, *recall*, dan *f1-score* untuk kedua teknik ekstraksi fitur dengan model Multinomial NB.

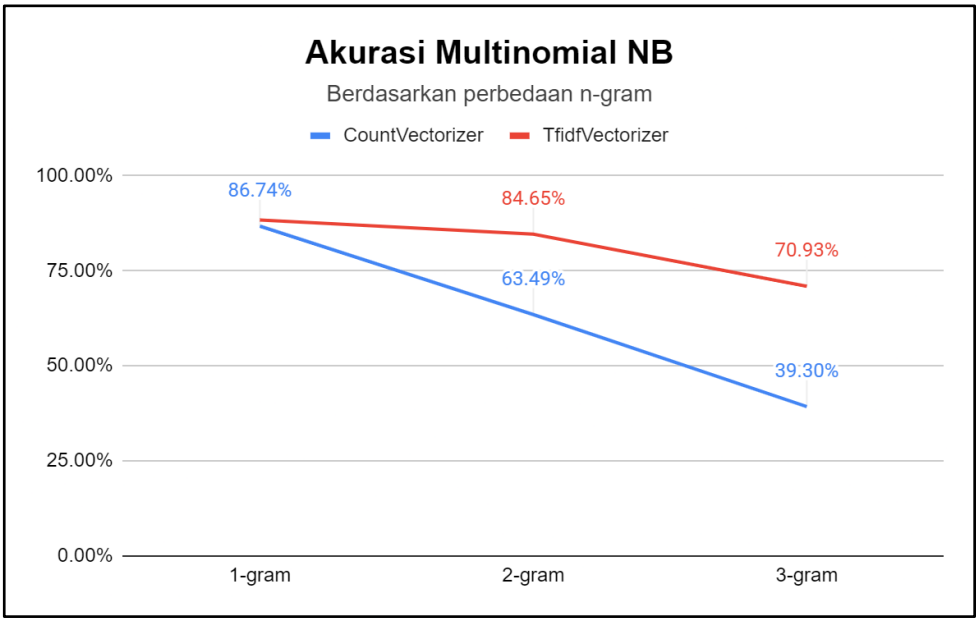


Gambar 4.16 Perbandingan performa teknik ekstraksi fitur untuk model Multinomial NB.

Model Bernoulli dan Complement NB juga mengalami peningkatan akurasi masing-masing menjadi 86.98% dan 88.37% dari nilai akurasi sebelumnya sebesar 86.51%, dan 86.74%. Namun, terdapat pengurangan akurasi pada model Gaussian NB yang sebelumnya sebesar 69.07% menjadi 67.23%. Model Multinomial NB kembali mendapatkan nilai akurasi terbesar diantara keempat model menggunakan pembobotan TF-IDF yaitu sebesar 88.37%.

4.10.2 Klasifikasi Menggunakan n-gram yang Berbeda

Model Multinomial NB diuji kembali menggunakan n-gram yang berbeda saat melakukan ekstraksi fitur, token-token dibagi menjadi 1-gram (*unigram)*, 2-gram (*bigram)*, dan 3-gram (*trigram*). Pendefinisian n-gram yang digunakan dapat dilakukan dengan merubah parameter *ngram_range()* dengan batas bawah dan batas atas kisaran nilai n-gram. Misalnya *ngram_range* yang bernilai (1, 1) berarti hanya *unigram*, (2, 2) berarti *bigram*, dan (3, 3) berarti hanya *trigram*. Setelah dilakukan klasifikasi sentimen menggunakan n-gram yang berbeda, didapatkan akurasi masing-masing sebesar 82.1%, 69.0%, dan 46.2% menggunakan *Bag-of-Words*, dan 82.7%, 75.2%, dan 54.6% untuk TF-IDF.

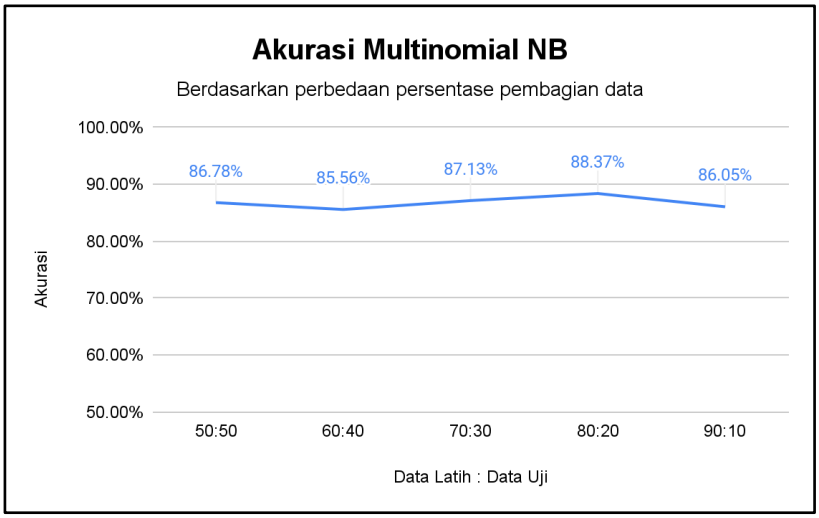


Gambar 4.17 Akurasi Multinomial NB berdasarkan perbedaan n-gram.

Pengamatan berdasarkan hasil yang didapatkan menunjukkan jumlah n-gram yang digunakan berbanding terbalik dengan akurasi prediksi sentimen yang didapatkan, model mengalami penurunan akurasi seiring besarnya n-gram yang digunakan. Selain itu juga membuktikan kembali bahwa ekstraksi fitur menggunakan metode TF-IDF menghasilkan akurasi yang lebih besar dari metode *Bag-of-Words*.

4.10.3 Klasifikasi Menggunakan Persentase Pembagian Data yang Berbeda

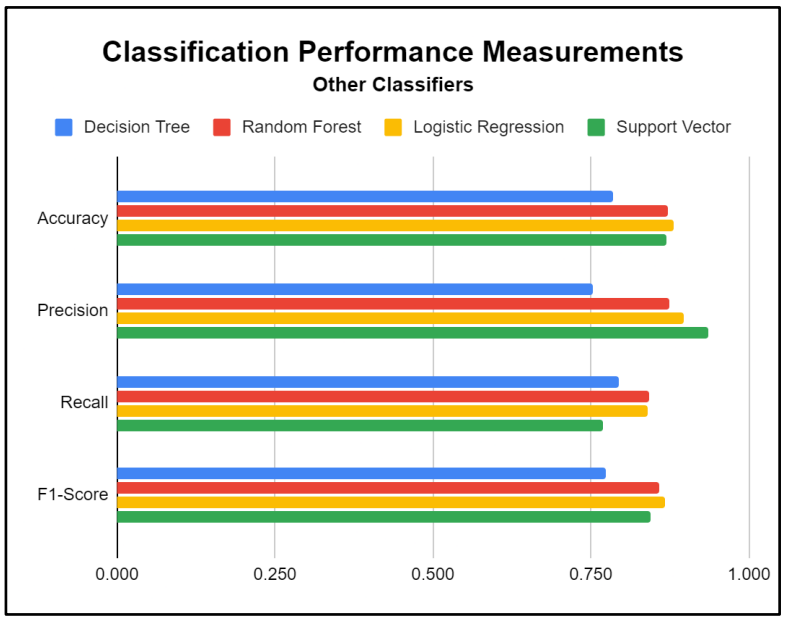
Persentase pembagian data sebesar 80:20 digunakan dalam proses pembagian set data, yang berarti sebanyak 20% dari keseluruhan data akan digunakan sebagai set data uji, dan 80% sebagai set data latih. Pada percobaan berikut dilakukan percobaan penggunaan persentase pembagian data untuk persentase data uji sebanyak 10%-50% dari keseluruhan data. Perubahan persentase pembagian data dilakukan dengan merubah nilai parameter *test_size* saat menjalankan fungsi *train_test_split()*, nilai yang uji adalah rentang 0.1 sampai 0.5 yang mewakili persentase jumlah data uji yang akan dibagi sebanyak 10%-50%. Hasil akurasi yang didapatkan untuk setiap persentase adalah sebagai berikut:



Gambar 4.18 Akurasi Multinomial NB berdasarkan perbedaan persentase pembagian data.

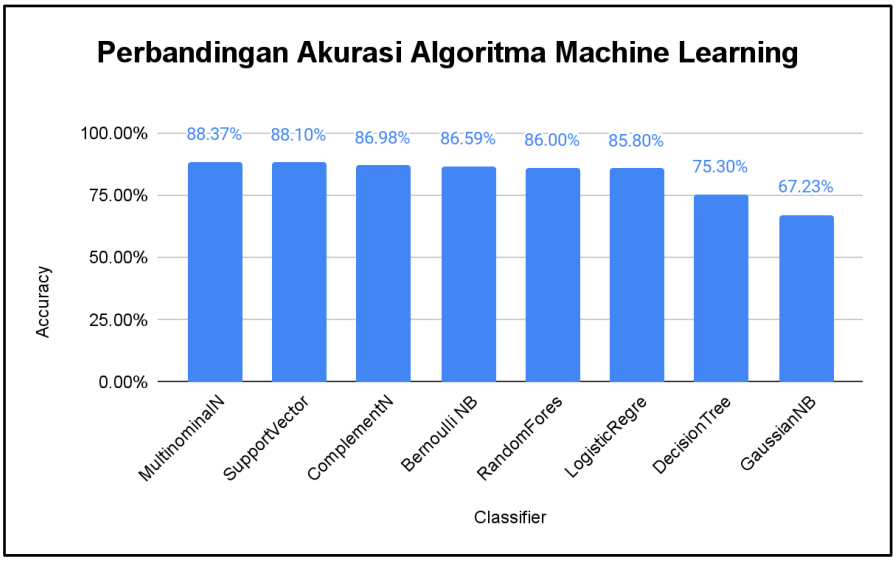
4.10.4 Klasifikasi Menggunakan Algoritma *Machine Learning* non-Bayesian

Pada tahapan ini dilakukan tugas klasifikasi sentiment yang sama menggunakan algoritma *machine learning* non-bayesian. Teknik ekstraksi fitur yang digunakan adalah pembobotan TF-IDF dengan n-gram sebesar 1, dan persentase pembagian data sebanyak 80:20. Perbandingan performa klasifikasi algoritma non-bayesian yang gunakan digambarkan pada Gambar 4.19.



Gambar 4.19 Perbandingan performa klasifikasi algoritma non-bayesian.

Hasil performa semua algoritma *machine learning* yang digunakan kemudian dibandingkan berdasarkan tingkat akurasinya. Menggunakan pengaturan yang sama, dihasilkan perbandingan performa akurasi sebagai berikut dalam Gambar 4.20.



Gambar 4.20 Perbandingan akurasi semua algoritma *machine learning* yang digunakan.

5. PENUTUP

5.1 Kesimpulan

Berdasarkan rangkaian tahapan yang telah dilakukan, terdapat beberapa kesimpulan yang dapat diambil dari penelitian ini antara lain:

1. Ulasan pada aplikasi Amazon Shopping di Google Play Store memiliki sebanyak 1156 (53.8%) ulasan bersentimen negatif, dan 991 (46.2%) ulasan bersentimen negatif dari keseluruhan ulasan sebanyak 2147. Hal ini menunjukkan sentimen pengguna cenderung negatif.
2. Hasil klasifikasi menggunakan keempat algoritma Naive Bayes menghasilkan akurasi rata-rata sebesar 82.15%, *precision* sebesar 72.25%, *recall* sebesar 83.49%, dan *f1-score* sebesar 77.41%. Multinomial NB menghasilkan performa terbaik diantara keempat algoritma Naive Bayes yang digunakan, yaitu sebesar 86.74%. Nilai *precision*, *recall*, dan *f1-score*nya berturut-turut adalah 78.82%, 85.90%, dan 82.21%.
3. Ekstraksi fitur menggunakan metode TF-IDF menghasilkan akurasi lebih besar dibandingkan metode *Bag of Words*, akurasi Multinomial NB meningkat dari 86.74% menjadi 88.37%. Nilai n-gram berbanding terbalik dengan akurasi model yang dihasilkan, semakin besar nilai *n*, akurasi model yang dihasilkan semakin kecil. Model Multinomial NB menghasilkan akurasi sebesar 88.37% untuk *unigram*, 84.65% untuk *bigram*, dan 70.93% untuk *trigram*. Selain itu, pengujian dengan menggunakan persentase data uji sebanyak 10-50% menunjukkan bahwa persentase 20% menghasilkan akurasi terbesar, yaitu 88.37%.
4. Hasil perbandingan terhadap kedelapan algoritma *machine learning* yang digunakan menunjukkan Naive Bayes memiliki performa yang bersaing dibandingkan dengan algoritma lainnya. Akurasi tertinggi diperoleh sebesar 88.37% menggunakan Multinomial NB, sedangkan terbesar kedua adalah Support Vector Machine dengan nilai 88.10%.

5.2 Saran

Penelitian yang telah dilakukan masih jauh dalam kata sempurna, beberapa saran yang dapat penulis sampaikan dalam tulisan ini adalah:

1. Menggunakan data ulasan dengan jumlah yang lebih banyak dalam melatih model *machine learning*.
2. Membuat sistem klasifikasi sentimen dengan antarmuka web maupun desktop agar lebih mudah digunakan dan dipahami pengguna.
3. Mencoba lebih banyak teknik untuk meningkatkan akurasi model, seperti menerapkan *k-Fold Cross-Validation*.

DAFTAR PUSTAKA

[1] Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. Knowledge-Based Systems, 226, 107134. doi:10.1016/j.knosys.2021.107134.

[2] Borka, K. R., Hora, S., Jain, T., Wambugu, M. (2019). Deep Learning for Natural Language Processing. Packt Publishing Ltd: Birmingham. ISBN: 9781838553678.

[3] BrightLocal. (2020). Local Consumer Review Survey 2020. https://www.brightlocal.com/research/local-consumer-review-survey-2020/. Diakses pada 15 April 2022.

[4] Ceci, L. (2022). Number of available apps in the Google Play Store from 2nd quarter 2015 to 1st quarter 2022. Statista. https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/. Diakses tanggal 5 Juli 2022

[5] Charbuty, B., & Abdulazeez, A. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. Journal of Applied Science and Technology Trends, 2(01), 20 - 28. https://doi.org/10.38094/jastt20165.

[6] Chevalier, S. (2021). Leading shopping apps in the U.S. 2021, by monthly active users [Infographic]. Statista. https://www.statista.com/statistics/1239046/top-saas-countries-list/. Diakses tanggal 18 Mei 2022.

[7] Ghojogh, B., Samad, M. N., Mashhadi, S. A., Kapoor, T., Ali, W., Karray, F., & Crowley, M. (2019). Feature selection and feature extraction in pattern analysis: A literature review. arXiv preprint arXiv:1905.02845.

[8] Ghosh, S., Gunning, D. (2019). Natural Language Processing Fundamentals: Build intelligent applications that can interpret the human language to deliver impactful results. Packt Publishing Ltd: Birmingham. ISBN: 9781789955989.

[9] Golpour, P., Ghayour-Mobarhan, M., Saki, A., Esmaily, H., Taghipour, A., Tajfard, M., … Ferns, G. A. (2020). Comparison of Support Vector Machine, Naïve Bayes and Logistic Regression for Assessing the Necessity for Coronary Angiography. International Journal of Environmental Research and Public Health, 17(18), 6449. doi:10.3390/ijerph17186449.

[10] Haque, T. U., Saber, N. N., & Shah, F. M. (2018). Sentiment analysis on large scale Amazon product reviews. 2018 IEEE International Conference on Innovative Research and Development (ICIRD). doi:10.1109/icird.2018.8376299.

[11] Hutchinson, T. (2020). Natural language processing and machine learning as practical toolsets for archival processing. Records Management Journal, 30(2), 155–174. doi:10.1108/rmj-09-2019-0055.

[12] Jovel J, Greiner R. An Introduction to Machine Learning Approaches for Biomedical Research. Front Med (Lausanne). 2021 Dec 16;8:771607. doi: 10.3389/fmed.2021.771607. PMID: 34977072; PMCID: PMC8716730.

[13] Kedia, A., Rasu, M. (2020). Hands-On Python Natural Language Processing: Explore tools and techniques to analyze and process text with a view to building real-world NLP applications. Packt Publishing Ltd: Birmingham. ISBN: 9781838982584.

[14] Kirasich, Kaitlin; Smith, Trace; and Sadler, Bivin (2018) "Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets," SMU Data Science Review: Vol. 1: No. 3, Article 9.

[15] Kumar, S., Kar, A. K., & Ilavarasan, P. V. (2021). Applications of text mining in services management: A systematic literature review. International Journal of Information Management Data Insights, 1(1), 100008. doi:10.1016/j.jjimei.2021.100008.

[16] Liu, Bing. (2020). Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge University Press. ISBN: 9781108486378.

[17] Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9, 381-386.

[18] Mohri, M., Rostamizadeh, A., Talwalkar, A. (2018). Foundations of Machine Learning Second Edition. The MIT Press: Cambridge, MA. ISBN 9780262039406.

[19] Penn Treebank P.O.S Tags. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html. Diakses pada 12 Mei 2022.

[20] Prakash, C., Chittimalli P. K., Naik, R. (2022). Domain Specific Text Preprocessing for Open Information Extraction. ISEC 2022: 15th

Innovations in Software Engineering Conference. Article 28, 1–5.https://doi.org/10.1145/3511430.3511456.

[21] PYPL Index. (2022). PYPL PopularitY of Programming Language. Diakses pada 09 Mei 2022. http://pypl.github.io/PYPL.html.

[22] Retnoningsih E, Pramudita R. 2020. Mengenal Machine Learning Dengan Teknik Supervised dan Unsupervised Learning Menggunakan Python. Bina Insani ICT Journal. 7 (2): 156-165.

[23] Saim, M. M, Hassan A., (2022). Comparative study of machine learning algorithms (SVM, Logistic Regression and KNN) to predict cardiovascular diseases. E3S Web Conf. 351 01037. DOI: 10.1051/e3sconf/202235101037.

[24] Sharma, H., Saraswat, M., Yadav, A., Kim, J. H., & Bansal, J. C. (Eds.). (2021). Congress on Intelligent Systems. Advances in Intelligent Systems and Computing. doi:10.1007/978-981-33-6981-8.

[25] Speiser, J. L., Miller, M. E., Tooze, J., & Ip, E. (2019). A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling. Expert Systems with Applications. doi:10.1016/j.eswa.2019.05.028

[26] Srinath, K. R. (2017). Python – The Fastest Growing Programming Language. International Research Journal of Engineering Technology (IRJET), 04(12), 354–357.

[27] Tama, V. O., Sibaroni, Y., & Adiwijaya. (2019). Labeling Analysis in the Classification of Product Review Sentiments by using Multinomial Naive Bayes Algorithm. Journal of Physics: Conference Series, 1192, 012036. doi:10.1088/1742-6596/1192/1/012036.

[28] Tao, D., Yang, P., & Feng, H. (2020). Utilization of text mining as a big data analysis tool for food science and nutrition. Comprehensive Reviews in Food Science and Food Safety, 19(2), 875–894. doi:10.1111/1541-4337.12540.

[29] Thanaki, J. (2018). Python Natural Language Processing. Packt Publishing Ltd: Birmingham. ISBN: 9781787121423.

[30] TIOBE Index. (2022). TIOBE - The Software Quality Company. Diakses pada 09 Mei 2022. https://www.tiobe.com/tiobe-index/.

[31] Tran, H., (2019). Survey of Machine Learning and Data Mining Techniques used in Multimedia System. The University of Texas at Dallas. doi:10.13140/RG.2.2.20395.49446/1.

[32] Vanaja, S., Belwal, M. (2018). Aspect-Level Sentiment Analysis on E-Commerce Data. Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018). doi:10.1109/ICIRCA.2018.8597286.

LISTING PROGRAM

```
# -*- coding: utf-8 -*-
"""SentimentAnalysis.ipynb

# 1. Data Preparation

## 1.1 Data Collection


- Library used : [google-play-scraper
1.1.0](https://pypi.org/project/google-play-scraper/)
- App Source : [Amazon Shopping - Apps on Google
Play](https://play.google.com/store/apps/details?id=com.amazon.mShop.andr
oid.shopping)
"""

# Installing library google-play-scraper
!pip install google-play-scraper

# Import libraries
import pandas as pd
import numpy as np

# Scraping amazon reviews from play store
from google_play_scraper import Sort, reviews_all

amazon_reviews = reviews(
    "com.amazon.mShop.android.shopping",
    lang="en",
    country="us",
    sort=Sort.NEWEST
)

# Create dataset from scraped reviews
reviews = pd.DataFrame(np.array(amazon_reviews), columns=["review"])
df_reviews =
df_reviews.join(pd.DataFrame(reviews.pop("review").tolist()))

# Save raw reviews dataset as csv file
df_reviews.to_csv("amazon-reviews_raw.csv", index=False)

"""## 1.2 Data Cleaning

Steps:
- Check for duplicated row
- Check for missing value
- Only keep neccessary column(s)
- Rename columns
- Remove newline, special character, and emojis from review

"""

# Load raw reviews dataset
import pandas as pd
import re

df_raw = pd.read_csv("reviews_raw.csv")
df_raw.info()
```

```
# Check duplicated row
print(df_raw.duplicated().sum())

# Check missing value
print(df_raw["content"].isna().sum().sum())

def remove_emojis(data):
    emoj = re.compile("["
        u"\U00002700-\U000027BF"  # Dingbats
        u"\U0001F600-\U0001F64F"  # Emoticons
        u"\U00002600-\U000026FF"  # Miscellaneous Symbols
        u"\U0001F300-\U0001F5FF"  # Miscellaneous Symbols And Pictographs
        u"\U0001F900-\U0001F9FF"  # Supplemental Symbols and Pictographs
        u"\U0001FA70-\U0001FAFF"  # Symbols and Pictographs Extended-A
        u"\U0001F680-\U0001F6FF"  # Transport and Map Symbols
                      "]+", re.UNICODE)
    return re.sub(emoj, "", data)

# Keep neccessary columns
df_clean = df_raw[["content", "score"]].copy()

# Rename columns
df_clean.rename(columns={"content":"review", "score":"rating"},
inplace=True)

# Remove numbers
df_clean["review"].replace(r"\d+","", regex=True, inplace=True)

# Remove emojis
df_clean["review"] = df_clean["review"].apply(remove_emojis)

# Preview cleaned dataset
df_clean.head()

# Save dataset to csv file
df_clean.to_csv("reviews_clean.csv", index=False)

# manually cleaned
df_clean = pd.read_csv("reviews_clean.csv")
df_clean

"""## 1.3 Data Labelling

Labelling rules:
- If the review is likely to be negative, the label is negative.
- If the review is likely to be positive, the label is positive.
- If the review tends to be neutral, the label is positive.
- If the review seems ambiguous, the number of positive or negative words
determines the sentiment label.
- The rating given by reviewers is also used by researchers as a
consideration.
"""

# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Import labelled data
df_labelled = pd.read_csv("temp.csv")
df_labelled

# Sentiment distribution
```

```
df_labelled["sentiment"].value_counts()

# Rating distribution
df_labelled["score"].value_counts(sort=False).sort_index()

# Sentiment distribution per rating
df_labelled.groupby("score")["sentiment"].value_counts().sort_index()

"""## Data Splitting"""

# Split data to train and test data
import pandas as pd
from sklearn.model_selection import train_test_split

reviews = pd.read_csv("temp.csv")[["content","sentiment"]]
df_train, df_test = train_test_split(reviews,
                                     test_size=0.2,
                                     random_state=5)

# Size each data
print(f"Train size : {df_train.shape[0]}")
print(f"Test size : {df_test.shape[0]}")

# Save splitted data to .csv
df_train.reset_index(drop=True, inplace=True)
df_test.reset_index(drop=True, inplace=True)
df_train.to_csv("train-dataset.csv", index=False)
df_test.to_csv("test-dataset.csv", index=False)

print(df_train)
print(df_test)

"""# 2. Text Preprocessing

Steps:

- Case Folding
- Remove Punctuation
- Remove Stopwords
- Lemmatization

## 2.1 Preparing Resources
"""

# Import libraries
import nltk
import spacy
import string
from gensim.parsing.preprocessing import STOPWORDS as sw_gensim
from nltk.corpus import stopwords as sw_nltk
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Download packages
nltk.download("punkt") # tokenization
nltk.download("averaged_perceptron_tagger") # pos tagging
nltk.download("stopwords") # stopwords
nltk.download("wordnet") # lemmatization
nltk.download("omw-1.4") # lemmatization

# Create temporary dataset
import pandas as pd
```

```
df_labelled = pd.read_csv("reviews_labelled2.csv")
df_temp = pd.DataFrame(df_labelled["content"].copy())
df_temp

pd.set_option('display.max_colwidth', 60)

def show_difference(col1, col2, df=df_temp):
    """Show difference between pre and post processing"""
    # Get modified-only rows
    df_mod = df[df[col1] != df[col2]]
    df_mod = df[df[col1].str.len() < 80]
    # Get 3 random samples
    diff = df_mod.sample(3)[[col1, col2]]
    diff.rename(columns={col1:"before", col2:"after"}, inplace=True)
    return diff

"""## 2.2 Case folding

Converts all review text to lowercase
"""

# Case folding
def case_folding(text):
    return text.lower()

df_temp["lower"] = df_temp["content"].apply(case_folding)
show_difference("content", "lower")

"""## 2.3 Removing punctuation

List of punctuation from `string.punctuation`:

> !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
"""

# Remove punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans("", "", string.punctuation))

df_temp["punct"] = df_temp["lower"].apply(remove_punctuation)
show_difference("lower", "punct")

"""## 2.4 Removing stopwords

Stopwords used:

- NLTK (179 words)
- Gensim ([337
words](https://tedboy.github.io/nlps/_modules/gensim/parsing/preprocessin
g.html))
- Spacy ([326
words](https://github.com/explosion/spaCy/blob/master/spacy/lang/en/stop_
words.py))

Total combined = 413 words
"""

# load stopwords
sp = spacy.load('en_core_web_sm')
words_spacy = sp.Defaults.stop_words # spacy
words_nltk = sw_nltk.words("english") # nltk
words_gensim = sw_gensim # gensim
```

```
# Stopwords combined
stopwords = words_nltk + list(words_gensim) + list(words_spacy) # 843

# Remove duplicated words
stopwords = sorted(list(set(stopwords)))

# Total stopwords
print(len(stopwords))

# Stopwords sample
print(stopwords[:10])

# Remove Stopwords
def remove_stopwords(text):
    # return " ".join([w for w in text if w not in stop_words])
    return " ".join([w for w in text.split() if w not in stopwords])

df_temp["stopwords"] = df_temp["punct"].apply(remove_stopwords)
show_difference("punct", "stopwords")

"""## 2.5 Lemmatization"""

# Lemmatization
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

lemmatizer = WordNetLemmatizer()

def nltk_pos_tagger(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

def lemmatize(pos_tags):
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(pos_tags))
    wordnet_tagged = map(lambda x: (x[0], nltk_pos_tagger(x[1])),
nltk_tagged)
    lemmatized_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatized_sentence.append(word)
        else:
            lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))
    return " ".join(lemmatized_sentence)

lemmatizer = WordNetLemmatizer()
df_temp["lemmatize"] = df_temp["stopwords"].apply(lemmatize)
show_difference("stopwords", "lemmatize")

df_post = pd.DataFrame()
df_post["content"] = df_temp["lemmatize"]
df_post["sentiment"] = df_labelled["sentiment"]
df_post.head()
# Save post processing dataset to csv file
df_post.to_csv("reviews_post-processing.csv", index=False)
```

```
df_post

""" \
 \
 

## 2.6 Word Clouds

Library used: [wordcloud
1.8.2.2](https://amueller.github.io/word_cloud/generated/wordcloud.WordCl
oud.html)

Documentation:
[wordcloud.WordCloud](https://amueller.github.io/word_cloud/generated/wor
dcloud.WordCloud.html)

Labels:
- (A) Wordcloud for all classes
- (B) Wordcloud for positive class
- (C) Wordcloud for negative class
"""

# get texts for wordcloud
wordcloud_all = df_labelled["content"].values
wordcloud_pos = df_labelled[df_labelled["sentiment"] == 1]["content"]
wordcloud_neg = df_labelled[df_labelled["sentiment"] == -1]["content"]

# visualize wordclouds
cloud_content = {"A":wordcloud_all, "B":wordcloud_pos, "C":wordcloud_neg}
fig, ax = plt.subplots(1,3, figsize=(15,5))
fig.suptitle("Word Clouds for all classes", fontsize="xx-large");
for i,(key, text) in enumerate(cloud_content.items()):
    wordcloud = WordCloud(background_color="white",
                          height=400).generate(str(text))
    ax[i].imshow(wordcloud);
    ax[i].axis("off");
    ax[i].title.set_text(f"({key})")

"""# 3. Sentiment Analysis


"""

import warnings
warnings.filterwarnings('ignore')
# pd.set_option("display.float_format", lambda x: "%.02f" % x)

"""## 3.1 Compiling Models

Import naive bayes precompiled classes from sklearn

[Documentation](https://scikit-learn.org/stable/modules/naive_bayes.html)
"""

# importing models
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB

# specify the classifiers
classifiers = {
```

```
    "Bernoulli NB": BernoulliNB(),
    "Complement NB": ComplementNB(),
    "Gaussian NB": GaussianNB(),
    "Multinominal NB": MultinomialNB()
}

"""## 3.2 Perform Sentiment Classification"""

# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_recall_fscore_support

def sentiment_analysis(classifier, data_split):
    """Perform sentiment analysis"""
    X_train, X_test, Y_train, Y_test = data_split
    # fitting model
    classifier.fit(X_train.todense(), Y_train)
    # perform prediction
    predicted = classifier.predict(X_test.todense())

    # accuracy
    accuracy = accuracy_score(Y_test, predicted)
    # precision, recall, f-measure and support for each class
    scores = list(precision_recall_fscore_support(Y_test, predicted,
average='binary')[:-1])
    # confusion matrix
    cf_matrix = confusion_matrix(Y_test, predicted)
    # text report showing main classification metrics
    report = classification_report(Y_test, predicted)

    return [[accuracy]+scores, report, cf_matrix]

# Load review dataset
import pandas as pd
df_reviews = pd.read_csv("reviews_post-processing.csv")
reviews = df_reviews["content"]
sentiment = df_reviews["sentiment"]

# creating document-term matrix
# CountVectorizer
cv = CountVectorizer(ngram_range=(1,1))
cv_fit = cv.fit_transform(reviews)
# TfidfVectorizer
tfidf = TfidfVectorizer(ngram_range=(1,1))
tfidf_fit = tfidf.fit_transform(reviews)

# splitting data

# CountVectorizer
cv_split = train_test_split(cv_fit, sentiment,
                            test_size=0.2, random_state=5)
# TfidfVectorizer
tfidf_split = train_test_split(tfidf_fit, sentiment,
```

```
                                test_size=0.2, random_state=5)

# dictionary for saving classifier performances
performance = {}

# perform sentiment analysis for each classifier and save the result to
dict
for i, (key, classifier) in enumerate(classifiers.items()):
    result = sentiment_analysis(classifier, cv_split)
    performance[key] = {}
    performance[key]["scores"] = result[0]
    performance[key]["report"] = result[1]
    performance[key]["matrix"] = result[2]

reviews_sample = [
    "Sucks lately. Both the web page and the app suck after last update.
Its slow, glitches, freezeing when watching videos or kicks u out of app,
if not fixed will have to shop else where for holidays",
    "Amazon has made shopping during this time of uncertainty. A good
experience with timely deliveries.",
    "It is the worst, it doesn't even want to open. The app doesn't work.
It is giving me problems,it always closes up when i try to open it.",
    "Love it when it works but it keeps signing me out. It does this
every few months but it says I don't have amazon unlimited even though we
pay for it",
    "Amazon shopping is that 'right from the convenience of your bed'
shopping that removes every hassle and limitation. Not to mention,
shipping is easily facilitated through that 'front door drop off' system!
Can't you tell? I am a happy Amazon shopper 😊!",
]

print(classifier)
classifier.predict(cv.transform(reviews_sample))

new_reviews_sample = [
    "App never loads, need to use the website instead.",
    "I shop amazing all the time absolutely love the app it's so
convenient",
    "Great and excellent app for a secure shopping.",
    "Frustrating considering you can no longer purchase kindle books",
    "Brilliant service no complaint what so ever",
    "You can find almost everything you need or want..."
]

print(classifier)
classifier.predict(cv.transform(new_reviews_sample))

"""# 4. Model Evaluation"""

pd.set_option("display.float_format", lambda x: "%.04f" % x)

"""## 4.1 Classification Report"""

for i, (key, perf) in enumerate(performance.items()):
    print(key)
    print(perf["report"])

"""### 3.4.2 Scores"""

# create dataframe to store scores for each classifiers
df_scores = pd.DataFrame(columns=["classifier", "accuracy", "precision",
"recall", "fscore"])
```

```
df_scores["classifier"] = classifiers.keys()
df_scores.set_index(["classifier"], inplace=True)

# add scores to dataframe
for i, (key, perf) in enumerate(performance.items()):
    df_scores.iloc[i,:] = perf["scores"]

df_scores

print(f"Average Accuracy    = {df_scores['accuracy'].mean():.3f}")
print(f"Average Precision   = {df_scores['precision'].mean():.3f}")
print(f"Average Recall      = {df_scores['recall'].mean():.3f}")
print(f"Average F-Score     = {df_scores['fscore'].mean():.3f}")

"""### 3.4.3 Confusion Matrix"""

labels = ["Positive", "Negative"]
fig, axes = plt.subplots(1, 4, figsize=(20, 5), sharey='row')

for i, (key, perf) in enumerate(performance.items()):
    cf_matrix = perf["matrix"]
    disp = ConfusionMatrixDisplay(cf_matrix, display_labels=labels)
    disp.plot(ax=axes[i], xticks_rotation=0,
              cmap=plt.cm.Blues, values_format="g", colorbar=False)
    disp.ax_.set_title(key)
    disp.ax_.set_xlabel("")
    disp.ax_.set_ylabel("")

plt.subplots_adjust(wspace=0.1, hspace=0.1)
fig.text(0.45, 0.95, "Confusion Matrix - Naive Bayes Classifiers",
ha="center", fontsize="xx-large")
fig.text(0.45, 0.04, "Predicted Label", ha="center", fontsize="x-large")
fig.text(0.05, 0.50, "True Label", va="center", rotation='vertical',
fontsize="x-large")
fig.colorbar(disp.im_, ax=axes)
plt.show()

"""# 5. Improving Accuracy

## 5.1 Trying TF-IDF
"""

from sklearn.feature_extraction.text import TfidfVectorizer
corpus=["Amazon Prime is amazing",
        "Best place to shop",
        "Amazing choice to shop"]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(X.toarray())

pd.set_option("display.float_format", lambda x: "%.04f" % x)
df_tfidf = pd.DataFrame(columns=["classifier", "accuracy", "precision",
"recall", "fscore"])
df_tfidf["classifier"] = classifiers.keys()
df_tfidf.set_index(["classifier"], inplace=True)

tfidf = TfidfVectorizer(ngram_range=(1,1)).fit_transform(reviews)
tfidf_split = train_test_split(tfidf, sentiment, test_size=0.25,
random_state=5)

for i, (key, classifier) in enumerate(classifiers.items()):
    scores = sentiment_analysis(classifier, tfidf_split)[0]
```

```
    df_tfidf.iloc[i,:] = scores

df_tfidf

print(f"Average Accuracy    = {df_scores['accuracy'].mean():.3f}")
print(f"Average Precision   = {df_scores['precision'].mean():.3f}")
print(f"Average Recall      = {df_scores['recall'].mean():.3f}")
print(f"Average F-Score     = {df_scores['fscore'].mean():.3f}")

# CountVectorizer and TfidfVectorizer Accuracy Comparisson
vectorizer_comp = pd.DataFrame()
vectorizer_comp["CountVectorizer"] = df_scores["accuracy"]
vectorizer_comp["TfidfVectorizer"] = df_tfidf["accuracy"]

print("Accuracy")
print("CountVectorizer   = ", vectorizer_comp["CountVectorizer"].mean())
print("TfidfVectorizer   = ", vectorizer_comp["TfidfVectorizer"].mean())

vectorizer_comp

def get_tfidf_score(corpus, n=5):
    """ get top n terms with highest tf-idf score """
    tfidf_vectorizer=TfidfVectorizer(use_idf=True)
    tfidf_vectorizer_vectors=tfidf_vectorizer.fit_transform(corpus)
    # get transformed feature names
    features = tfidf_vectorizer.get_feature_names_out()
    # get the first vector out (for the first document)
    first_vector_tfidfvectorizer = tfidf_vectorizer_vectors[0]
    # store tfidf scores to dataframe
    df_tfidf = pd.DataFrame(first_vector_tfidfvectorizer.T.todense(),
                            index= features, columns=["tfidf_score"])
    # sort by score
    df_tfidf.sort_values(by=["tfidf_score"], ascending=False,
inplace=True)
    return df_tfidf.head(n)

get_tfidf_score(reviews, 10)

"""## 5.2 Trying different n-gram"""

def get_most_frequent_words(corpus, ngram, n=10):
    """ get top n words in text corpus by its occurrence """
    vec = CountVectorizer(ngram_range = (ngram, ngram)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    # sum of each word occurence
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, i]) for word, i in
vec.vocabulary_.items()]
    # sort by occurrence
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    # convert to dataframe
    freq_words = pd.DataFrame(words_freq[:n], columns=["word", "count"])

    return freq_words

# shows most frequest words by n-gram
print(f"Unigram : \n {get_most_frequent_words(reviews, ngram=1)}\n")
print(f"Bigram  : \n {get_most_frequent_words(reviews, ngram=2)}\n")
print(f"Trigram : \n {get_most_frequent_words(reviews, ngram=3)}")

def acc_with_different_ngrams(vectorizer, corpus, classifiers,
ngram_start, ngram_stop):
    # create dataframe to store classifiers accuracy for each grams
```

```
    df_ngrams = pd.DataFrame(columns=["classifier", "unigram", "bigram",
"trigram"])
    df_ngrams["classifier"] = classifiers.keys()
    df_ngrams.set_index(["classifier"], inplace=True)
    # perform calculation for each n-gram
    for n in range(ngram_start,ngram_stop+1):
        cv = vectorizer(ngram_range=(n,n)).fit_transform(reviews)
        cv_split = train_test_split(cv, sentiment, test_size=0.2,
random_state=5)
        # perform sentiment analysis for each classifier
        for i, (key, classifier) in enumerate(classifiers.items()):
            accuracy = sentiment_analysis(classifier, cv_split)[0][0]
            df_ngrams.iloc[i,n-1] = accuracy

    return df_ngrams

print("With CountVectorizer")
print(acc_with_different_ngrams(CountVectorizer, reviews, classifiers, 1,
3))
print("\nWith TfidfVectorizer")
print(acc_with_different_ngrams(TfidfVectorizer, reviews, classifiers, 1,
3))

"""## 5.3 Different Data Split Percentage"""

for n in range(1, 6, 1):
    tfidf_split = train_test_split(tfidf_fit, sentiment, test_size=n/10,
random_state=5)
    acc=sentiment_analysis(MultinomialNB(), tfidf_split)[0][0]
    print(f"{n/10} {acc}")

"""## 5.4 Trying different classifiers"""

# import classifiers
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# specify the classifiers
classifiers_2 = {
    "Support Vector": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Logistic Regression": LogisticRegression()
}

# create dataframe to store performance each classifiers
df_otherclass = pd.DataFrame(columns=["classifier", "acc_cv",
"acc_tfidf"])
df_otherclass["classifier"] = classifiers_2.keys()
df_otherclass.set_index(["classifier"], inplace=True)

# perform sentiment analysis for each classifier
for i, (key, classifier) in enumerate(classifiers_2.items()):
    acc_cv = sentiment_analysis(classifier, cv_split)[0][0]
    acc_tfidf = sentiment_analysis(classifier, tfidf_split)[0][0]
    df_otherclass.iloc[i,:] = acc_cv, acc_tfidf

df_otherclass
```

LAMPIRAN OUTPUT PROGRAM

Hasil Klasifikasi dengan Algoritma Naive Bayes

Classifier	Accuracy	Precision	Recall	F1-Score
Bernoulli NB	0.8628	0.7650	0.8974	0.8260
Complement NB	0.8651	0.7784	0.8782	0.8253
Gaussian NB	0.6907	0.5584	0.7051	0.6232
Multinomial NB	0.8674	0.7882	0.8590	0.8221
Average	0.8215	0.7225	0.8349	0.7741

Hasil Klasifikasi Menggunakan TF-IDF

Classifier	Accuracy	Precision	Recall	F1-Score
Bernoulli NB	0.8659	0.7778	0.8883	0.8294
Complement NB	0.8698	0.8426	0.8426	0.8426
Gaussian NB	0.6723	0.5443	0.6548	0.5945
Multinomial NB	0.8837	0.8857	0.7868	0.8333
Average	0.8229	0.7626	0.7931	0.7750

Hasil Akurasi Klasifikasi Menggunakan n-gram yang Berbeda

Classifiers	CountVectorizer			TfidfVectorizer		
	1-gram	2-gram	3-gram	1-gram	2-gram	3-gram
Multinomial NB	0.8674	0.6349	0.3930	0.8837	0.8465	0.7093
Bernoulli NB	0.8628	0.7047	0.4837	0.8659	0.7047	0.4837
Gaussian NB	0.6907	0.7744	0.5860	0.6723	0.7791	0.5860
Complement NB	0.8651	0.6470	0.3860	0.8698	0.6767	0.4047
Average	0.8215	0.6903	0.4622	0.8229	0.7518	0.5459

Hasil Klasifikasi Menggunakan Persentase Pembagian Data yang Berbeda

%Data uji	Accuracy	Precision	Recall	F1-Score
10%	0.860	0.950	0.745	0.835
20%	0.884	0.946	0.795	0.864
30%	0.871	0.932	0.767	0.841
40%	0.856	0.914	0.747	0.822
50%	0.868	0.917	0.770	0.837

Hasil Klasifikasi dengan Algoritma *Machine Learning* non-Bayesian (TF-IDF)

Classifier	Accuracy	Precision	Recall	F1-Score
Support Vector	0.8659	0.7778	0.8883	0.8294
Decision Tree	0.8698	0.8426	0.8426	0.8426
Random Forest	0.6723	0.5443	0.6548	0.5945
Logistic Regression	0.8837	0.8857	0.7868	0.8333