ANALISIS SENTIMEN KINERJA POLRI DI MEDIA SOSIAL TWITTER

Disusun Oleh:

N a m a : Fazma Rizqy Alfi Bahri
NIM : 17523211

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS SENTIMEN KINERJA POLRI DI MEDIA SOSIAL TWITTER

TUGAS AKHIR

Disusun Oleh:

N a m a : Fazma Rizq Alfi Bahri

NIM : 17523211

Yogyakarta, 26 Juli 2023

Pembimbing,

(Arrie Kurniawardhani, S.Si, M.Kom.,)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS SENTIMEN KINERJA POLRI DI MEDIA SOSIAL TWITTER

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 26 Juli 2023

Tim Penguji

Ketua Penguji

Arrie Kurniawardhani, S.Si, M.Kom., _____

Anggota 1

ANDHIKA GIRI PERSADA, S.Kom., M.Eng. _____

Anggota 2

KURNIAWAN DWI IRIANTO, S.T., M.Sc. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(DHOMAS HATTA FUDHOLI, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Fazma Rizqy Alfi Bahri

NIM : 17523211

Tugas akhir dengan judul:

ANALISIS SENTIMEN KINERJA POLRI DI MEDIA SOSIAL TWITTER

Menyatakan bahwa keseluruhan komponen yang ada dalam tugas akhir ini beserta isinya merupakan hasil karya saya sendiri, tanpa campur tangan orang lain. Apabila di kemudian hari terbukti bahwa beberapa bagian dari karya ini ada yang bukan merupakan hasil karya saya sendiri, tugas akhir yang saya ajukan sebagai hasil karya saya sendiri ini siap untuk ditarik kembali dan saya siap untuk menanggung apapun resiko dan konsekuensi yang muncul.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 26 Juli 2023

(Fazma Rizqy Alfi Bahri)

HALAMAN PERSEMBAHAN

Alhamdulillahirabbil'alamin, beribu puji dan syukur penulis panjatkan kepada *Allah SWT*, Tuhan alam semesta, pemberi rezeki dan rahmat yang tiada habisnya, yang dengan izinnya diberikan kemudahan serta kelancaran dalam berbagai aspek sehingga proses pengerjaan Tugas Akhir ini dapat diselesaikan dengan baik sehingga selesai tepat pada waktunya. Tulisan ini dipersembahkan untuk berbagai individu yang telah membantu dalam pembuatan tugas akhir ini, tetapi yang paling utama adalah untuk kedua orang tua penulis yang tidak henti-hentinya memanjatkan doa dan selalu memberi dukungan dalam berbagai aspek di kehidupan yang penulis jalani. Terutama pada bagian pembiayaan dari awal penulis masuk kuliah hingga pada tahap akhir ini.

Rasa terima kasih yang sangat besar juga tak lupa penulis persembahkan untuk istri penulis yang bernama Devina Putri Rahayu. Karena beliaulah dengan doa dan semangat serta dukungan yang diberikan kepada penulis dengan tiada hentinya-lah yang dapat memotivasi penulis untuk menyelesaikan tugas akhir ini.

Tak lupa juga penulis mempersembahkan rasa terima kasih kepada ibu dosen pembimbing, ibu Arrie Kurniawardhani, S.Si, M.Kom., beserta seluruh dosen program studi Informatika Universitas Islam Indonesia, karena dengan pelajaran dan bimbingan yang telah mereka berikan, penulis dapat untuk sampai hingga titik penyelesaian tugas akhir ini.

Penulis juga mengucapkan terima kasih untuk semua keluarga besar penulis yang terus mendukung penulis agar dapat menyelesaikan tugas akhir ini.

Penulis berdoa agar semua bantuan baik fisik maupun moril yang telah diberikan kepada penulis dibalas dengan sebaik-baiknya balasan oleh Allah SWT, aamiin yra.

HALAMAN MOTO

Penulis ingin menceritakan sesuatu yang membuat penulis termotivasi untuk menyelesaikan tugas akhir ini. 3 semester sebelumnya, penulis mengambil topik tugas akhir mengenai pengembangan gim. Namun, di perjalanan dalam menyelesaikannya penulis ternyata tidak dapat mengerjakannya dengan maksimal sehingga penulis mengulangi kembali pengerjaan tugas akhir. Penulis sendiri sangat termotivasi untuk menyelesaikan tugas akhir ini agar dapat menyelesaikan studi yang diambil oleh penulis secepatnya.

‘Barangsiapa yang memulai sesuatu, hendaklah mengerjakannya dengan baik hingga selesai’

-Fazma Rizqy Alfi Bahri

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, segala puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberi penulis kesehatan, kemampuan untuk berpikir dan menyelesaikan masalah, serta nikmat-nikmat lainnya sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul "Analisis Sentimen Kinerja Polri Di Media Sosial Twitter". Shalawat dan salam juga tak lupa penulis persembahkan kepada nabi besar kita Muhammad SAW.

Penulis menyusun laporan tugas akhir ini guna memenuhi kewajiban penulis sebagai mahasiswa pada jenjang Strata 1 (S1) pada Jurusan Informatika Universitas Islam Indonesia untuk dapat menyelesaikan studi yang penulis ambil. Laporan ini tidak dapat penulis selesaikan dengan baik apabila tidak ada dukungan dan motivasi yang telah diberikan oleh berbagai pihak. Oleh karenanya, penulis hendak menyampaikan rasa terima kasih yang sangat besar kepada:

1. Allah SWT, berkat rahmat dan hidayah yang diberikan oleh-Nya yang terus-menerus mengiringi langkah penulis dalam menuntut ilmu dengan memberikan kemampuan, kekuatan, serta kesehatan untuk penulis..
2. Orang tua penulis, Ayah dan Mama yang selalu mendo'akan penulis dan selalu memberi support baik berupa moril maupun materil untuk penulis sehingga penulis dapat menyelesaikan studi dengan baik
3. Istri Penulis, Devina Putri Rahayu yang sangat berperan besar dalam memberikan penulis dukungan dan segala bentuk bantuan yang diperlukan oleh penulis sehingga tugas akhir ini dapat diselesaikan dengan baik.
4. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Ibu Arrie Kurniawardhani, S.Si., M.Kom. selaku dosen pembimbing tugas akhir yang membimbing dan memberikan arahan untuk penulis dalam penyusunan laporan tugas akhir dan berbagai langkah pengerjaannya.
6. Seluruh Dosen dan Staff Karyawan Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang sangat berperan penting selama masa perkuliahan yang dijalani oleh penulis karena ilmu, dukungan, serta bantuan yang diberikan kepada penulis

Semoga semua pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini diberikan kesehatan, rezeki, serta hal-hal baik oleh Allah SWT. Penulis memohon maaf sebesar-besarnya apabila selama proses penyelesaian tugas akhir ini terjadi kekhilafan d an kesalahan yang diperbuat oleh penulis. Semoga tugas akhir ini dapat menjadi bermanfaat kepada dunia pendidikan dan dunia teknologi, terutama pada bidang informatika.

Wassalamualaikum Warahmatullahi Wabarakatuh.

Yogyakarta, 26 Juli 2023

(Fazma Rizqy Alfi Bahri)

SARI

Twitter merupakan sebuah platform media sosial yang jumlah pengguna aktifnya mencapai puluhan juta setiap harinya. Institusi yang dalam beberapa waktu terakhir ini kerap dijadikan sebagai bahan perbincangan oleh khalayak ramai adalah kinerja Kepolisian Republik Indonesia (POLRI). Seiring dengan banyaknya cuitan yang muncul mengenai institusi ini, dibutuhkan *sentiment analysis* untuk melakukan pemetaan terhadap berbagai sentimen yang muncul dari berbagai pihak. Oleh karenanya, dilakukan analisis sentimen mengenai tanggapan masyarakat terhadap kinerja POLRI melalui penelitian ini dan membandingkan dua metode klasifikasi yaitu *Naïve Bayes Classifier* (NBC) dan *Support Vector Machine* (SVM). Data mengenai sentimen masyarakat terhadap POLRI tersebut bermanfaat untuk dijadikan sebagai bahan evaluasi untuk meningkatkan kinerja di masa yang akan datang. Terdapat total 2814 data *tweet* yang terdiri atas 2313 *tweet* positif, 337 *tweet* netral, dan 164 *tweet negatif*. Skor akurasi yang diperoleh kedua algoritma ini masing-masing 91,4% untuk *Naïve Bayes Classifier* dan 91,1% untuk *Support Vector Machine*, hal ini dapat terjadi karena dilakukannya penyeimbangan terhadap data latih dengan metode SMOTE. Waktu prediksi yang dibutuhkan oleh *Naïve Bayes Classifier* lebih sedikit dibandingkan dengan *Support Vector Machine* yaitu hanya 0.00 detik. Sedangkan *Support Vector Machine* membutuhkan waktu sekitar 0.06 detik. Didasari oleh pengelompokkan data *tweet* mengenai kinerja POLRI di media sosial Twitter menurut label datanya, ditemukan bahwa sentimen mengenai kinerja POLRI adalah positif. Kata yang paling sering muncul dalam keseluruhan data yang dikumpulkan dari media sosial Twitter adalah polri, kerja, apresiasi, usut, tuntas, maksimal, kerja.

Kata kunci: *Naïve Bayes, Support Vector Machine* (SVM), Twitter, Polri.

GLOSARIUM

Preprocessing	Proses untuk mengubah suatu data agar bentuknya menjadi terstruktur
Klasifikasi	Sebuah metode yang untuk melakukan pengelompokan dengan cara sistematis terhadap data untuk diletakkan pada kategori yang sebelumnya telah ditetapkan
Data Crawling	Sebuah cara untuk mendapatkan datapada suatu web atau aplikasi dengan proses otomatis yang memanfaatkan alat bernama "*crawler"*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era perkembangan teknologi modern ini yang melesat jauh dari era sebelumnya membuat durasi peredaran berbagai informasi dapat berlangsung dengan sangat cepat. Bermacam jenis informasi beredar di berbagai platform digital, contohnya adalah media sosial, yaitu contohnya adalah Twitter. Definisi Twitter sendiri ialah sebuah platform digital yang layanannya menyediakan sarana untuk berkomunikasi antar teman, kolega, keluarga, atau siapapun untuk dapat tetap terkoneksi melalui pesan yang ditukarkan dengan cepat dan frekuensinya sering (Twitter, 2022).

Beragam informasi dalam bentuk teks menghiasi linimasa Twitter. Beragam individu hingga lembaga negara melakukan *tweet*. Salah satu lembaga negara tersebut yang memiliki akun Twitter adalah Kepolisian Republik Indonesia (POLRI). Hingga saat ini, opini yang muncul dari masyarakat sebagai pengguna Twitter mengenai kinerja POLRI sangatlah beragam. Opini tersebut bervariasi mulai dari pujian atas kinerja bagus yang dilakukan oleh POLRI, hingga opini negatif berupa hujatan yang ditujukan kepada Institusi Kepolisian.

Berbagai berita yang muncul mengenai Institusi Kepolisian ini jenisnya sangatlah banyak. Bermacam berita yang beredar ini menghasilkan banyak perbedaan tanggapan mengenai institusi ini. Masifnya jumlah opini-opini masyarakat ini dapat dipergunakan sebagai bahan untuk evaluasi agar kinerja Institusi Kepolisian menjadi lebih baik di masa yang akan datang.

Untuk dapat melakukan pemanfaatan terhadap informasi-informasi yang masif ini, dibutuhkan analisis yang tepat agar kumpulan informasi ini dapat digunakan untuk mendukung keputusan yang akan diambil oleh Institusi Kepolisian. Banyak sekali metode untuk menganalisis berbagai kicauan di Twitter, salah satunya adalah metode analisis sentimen. Analisis sentimen, atau yang biasanya diketahui sebagai *opinion mining,* merupakan sebuah studi komputasional mengenai sentimen, opini, dan emosi yang disampaikan dalam kata-kata atau kalimat (Villavicencio *et al.,* 2021).

Berdasarkan penjabaran di atas, untuk dapat melakukan analisis sentimen diperlukan sebuah mesin agar dapat menganalisis *tweet* pada media sosial Twitter. Adapun untuk dapat melakukan analisis sentimen terhadap hal ini yaitu dengan cara mengklasifikasi teks dengan menerapkan *text mining*. Metode analisis sentimen yang dimanfaatkan dalam penelitian ini

adalah *Naïve Bayes Classifier* (NBC) dan *Support Vector Machine* (SVM). Metode NBC sendiri adalah sebuah teknik pembelajaran mesin yang basisnya adalah probabilistik. Dalam pengklasifikasian teks, metode NBC menjadi salah satu pilihan dikarenakan metodenya sederhana dan juga performansi serta akurasi yang dimilikinya tinggi (D. G. Nugroho et al., 2015). Sedangkan metode SVM dijelaskan sebagai salah satu metode milik *supervised learning* yang dapat melakukan generalisasi pada proses klasifikasi terhadap suatu pola. Kelebihan yang dimiliki oleh metode ini ialah dapat melakukan identifikasi terhadap *hyperplane* terpisah yang mengoptimalkan margin antar kelas (Kristiyanti, 2015).

Dalam algoritma klasifikasi *Naïve Bayes* ini, ada dua tahapan pada proses pengklasifikasian teks. Tahapan pertama adalah dilakukannya analisis pada sampel dokumen. Proses analisis ini terdiri dari pemilahan kosa kata, yaitu kosa kata yang memiliki kemungkinan untuk muncul dalam koleksi dokumen sampel. Setelah pemilahan kosa kata dilakukan, selanjutnya dilakukan penentuan probabilitas pada tiap kategori berlandaskan dokumen sampelnya. Tahapan selanjutnya yaitu tahap klasifikasi, yang mana akan ditentukan nilai kategori suatu dokumen berlandaskan kosa kata yang timbul dalam dokumen yang telah diklasifikasi (Giovani et al., 2020).

Pada penelitian ini, dilakukan proses analisis sentimen untuk dapat melihat kecenderungan berbagai opini yang beredar di media sosial Twitter terhadap kinerja POLRI mengandung kelas sentimen yaitu positif, negatif, atau netral serta untuk dapat mengetahui seberapa baik akurasi yang dimanifestasikan oleh metode *naïve bayes* saat digunakan untuk menganalisis sentimen opini kinerja POLRI.

1.2 Rumusan Masalah

Didasari oleh latar belakang yang sudah dijelaskan di atas, berikut merupakan beberapa pertanyaan dalam penelitian ini:

a. Bagaimana metode *Naïve Bayes* dan *Support Vector Machine* (SVM) dapat mengategorikan *tweet* mengenai kinerja Institusi Kepolisian Negara Republik Indonesia (POLRI) yang terdapat di media sosial Twitter dengan tepat?
b. Berapa perolehan skor akurasi yang didapatkan dari penerapan metode Naïve Bayes dan *Support Vector Machine* (SVM) dalam pengategorian tweet mengenai kinerja Institusi Kepolisian Negara Republik Indonesia (POLRI) yang terdapat di jejaring sosial Twitter?

c. Di antara kedua algoritma *Naïve Bayes* dan *Support Vector Machine*, algoritma mana yang memiliki performa lebih baik dalam melakukan klasifikasi?

1.3 Tujuan Penelitian

Tujuan dari diadakannya penelitian ini yaitu untuk:

a. Mengetahui apakah penggunaan metode *Naïve Bayes* dan *Support Vector Machine* dapat mengategorikan *tweet* mengenai kinerja Institusi Kepolisian Negara Republik Indonesia (POLRI) yang terdapat di media sosial Twitter dengan tepat.
b. Mengetahui perolehan skor akurasi yang didapatkan dari penerapan metode *Naïve Bayes* dan *Support Vector Machine* dalam pengategorian *tweet* mengenai kinerja Institusi Kepolisian Negara Republik Indonesia (POLRI) yang terdapat di jejaring sosial Twitter.
c. Melakukan perbandingan performa di antara dua algoritma *Naïve Bayes* dan *Support Vector Machine* dalam melakukan proses klasifikasi.

1.4 Batasan Masalah

Untuk membatasi agar pembahasan pada penelitian ini tidak keluar dari topik yang seharusnya, penulis memberikan batasan-batasan dalam penelitian ini. Batasan penelitian tersebut adalah:

a. *Tweet* yang diambil serta dilakukan analisis terhadapnya hanya tweet yang menggunakan Bahasa Indonesia.
b. Metode klasifikasi yang digunakan adalah metode Naïve Bayes dan *Support Vector Machine*.
c. *Tweet* hanya diklasifikasikan menjadi 3 jenis sentimen, yaitu positif, netral, dan negatif
d. T*weet* yang diambil hanya mulai dari tanggal 28 September hingga 6 Oktober 2022 dan 11 Desember hingga 20 Desember 2022.

1.5 Manfaat Penelitian

Penelitian ini dapat memberikan manfaat berupa dijadikannya penelitian ini sebagai referensi pada penelitian mengenai topik yang sejenis di masa yang akan datang. Penelitian ini juga dapat menunjukkan metode mana yang lebih optimal untuk mengklasifikasikan *tweet* mengenai kinerja polri melalui analisa yang dilakukan. Proses analisis sentimen mengenai *tweet* yang membahas kinerja Institusi Kepolisian Negara Republik Indonesia (POLRI) terdapat di aplikasi Twitter dengan harapan penelitian ini dapat bermanfaat untuk menjadi referensi dalam pengambilan keputusan untuk dapat meningkatkan kinerja POLRI.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Pada bagian ini, akan dilakukan pembahasan mengenai latar belakang dilakukannya penelitian ini, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan

BAB II TINJAUAN PUSTAKA

Dalam bab ini dipaparkan penelitian yang terlebih dahulu dilakukan mengenai topik analisis sentimen dan dilakukan perbandingan terhadap topik analisis sentimen yang mengunakan berbagai metode algoritma. Teori-teori mendasar yang memiliki hubungan dengan penelitian yang dimanfaatkan untuk melakukan pemecahan masalah pada penelitian ini turut dibahas dalam bab ini.

BAB III METODOLOGI PENELITIAN

Bab ini berisikan penjelasan mengenai prosedur yang dilalui dalam melakukan penelitian ini.

BAB IV HASIL DAN PEMBAHASAN

Pada bagian ini, hasil dari pengkajian yang telah dilaksanakan oleh peneliti yaitu berupa data yang telah selesai diolah akan dipresentasikan menjadi sebuah informasi.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan yang peneliti dapatkan dari proses pengolahan.

BAB II
TINJAUAN PUSTAKA

2.1 Analisis Sentimen

Analisis sentimen atau yang dapat disebut juga sebagai *opinion mining* dapat dijabarkan sebagai sebuah riset komputasional yang secara tekstual mengekspresikan emosi, opini, dan sentimen (Ratnawati, 2018). Metode analisis sentimen ini memiliki objektif utama, yaitu untuk dapat memahami, mengekstrak dan melakukan pengolahan suatu data berbentuk tekstual dengan tujuan untuk memperoleh informasi sentimen yang tersemat dalam suatu bentuk kalimat, yaitu kalimat opini (Buntoro, 2017). Frasa "analisis sentimen" sendiri muncul pertama kali pada tahun 2003 oleh Nasukawa dan Yi. Untuk frasa "*opinion mining*" sendiri muncul di tahun yang sama, yaitu 2003 oleh Dave, Lawrence, dan Pennock. (Harijiatno, 2019).

2.2 *Natural Language Processing* (NLP)

Natural Language Processing atau yang biasa disingkat dengan (NLP) merupakan sebuah bidang keilmuan pada Ilmu Komputer yang berurusan dengan pemrosesan bahasa manusia baik dalam bentuk teks atau percakapan (Purwarianti et al., 2016). Tujuan bidang keilmuan ini sendiri ialah untuk merancang serta membangun aplikasi yang mana interaksi antara manusia dan mesin melalui pemanfaatan bahasa alami dapat difasilitasi dengan baik. Pada ilmu NLP ini terdapat beberapa bidang, antara lain *Document Classification, Machine Translation, Question Answering Systems (QAS), Summarization,* dan *Speech Recognition* (Rachman, 2021).

2.3 Text mining

Text mining adalah bagian dari tahapan proses analisis mengenai data berbentuk teks yang datanya bersumber dari suatu dokumen. Dalam pengklasifikasian dokumen tekstual, konsep *text mining* ini dimanfaatkan untuk mengklasifikasi kumpulan dokumen tersebut agar sinkron dengan subjeknya masing-masing (Darwis et al., 2021). Tujuan dari dilakukannya proses *text mining* ini sendiri adalah untuk mengambil dan mengumpulkan kata, serta untuk memperoleh informasi. Kemudian, pada hasil yang didapat dari proses *text mining* ini dapat dilakukan proses analisis. Yang mana, pada proses ini analisa yang dilakukan memiliki nilai yang dapat dimanfaatkan untuk kepentingan tertentu (Farmadiansyah et al., 2021). Sampai hari ini, penerapan *text mining* sudah dilakukan di berbagai macam bidang, di antaranya:

a. *Summarization* (Peringkasan)

Dari sisi pembaca, penerapan *text mining* berupa *summarization* ini bermanfaat untuk menghemat waktu dikarenakan prosesnya yang berupa peringkasan terhadap suatu dokumen.

b. *Concept Linking* (Penautan Topik)

Concept Linking diterapkan dengan cara menghubungkan bermacam dokumen terkait dengan cara mengidentifikasi konsep-konsep yang akan dipergunakan. Maka dari itu, informasi yang tidak ditemukan dengan metode pencarian tradisional akan dibantu proses penemuannya dengan menerapkan metode ini.

c. Question Answering

Proses penerapan *Question Answering* adalah dengan mencocokkan pola yang berbasis *knowledge* untuk dapat menemukan jawaban-jawaban yang reliabel pada pertanyaan yang diajukan.

d. Information Extraction

Pada bidang ini, *text mining* diterapkan sebagai identifikasi terhadap frase-frase kunci dan hubungan di dalam teks dengan cara mencocokkan pola dalam teks berdasarkan urutan yang telah ditetapkan

e. Clustering

Di bidang *Clustering*, dokumen-dokumen yang sebelumnya tidak memiliki kategori yang telah ditetapkan akan dikelompokkan berdasarkan kemiripannya

f. Topic Tracking

Pada bidang *Topic Tracking,* penerapan *text mining* dapat diterapkan sebagai alat untuk memprediksi dokumen-dokumen lain yang diminati oleh user berlandaskan oleh berbagai dokumen yang telah dilihat dan berdasarkan profil user tersebut (Imron, 2019).

2.4 Machine Learning

Machine Learning dapat dijabarkan sebagai sebuah ilmu untuk menciptakan sebuah sistem yang dapat dengan otodidak berfungsi dan belajar sendiri tanpa perlu manusia melakukan pemrograman berulang kali terhadapnya. Ilmu ini dikategorikan sebagai salah satu disiplin ilmu dalam bidang ilmu Kecerdasan Buatan, atau yang biasanya dikenal sebagai *Artificial Intelligent* (AI) (Imron, 2019).

Berdasarkan prosedur yang dimiliki oleh algoritmanya, teknik dalam *Machine Learning* diklasifikasikan menjadi 4 tipe, yaitu *Supervised Machine Learning, Unsupervised Machine Learning, Semi-Supervised Machine Learning,* dan yang terakhir adalah *Reinforcement Machine Learning*. Keempat teknik dalam *Machine Learning* di atas dapat dijabarkan sebagai berikut:

1. *Supervised Machine Learning*

   Jenis teknik yang pertama dalam *Machine Learning* bernama *Supervised Machine Learning*. Teknik ini memanfaatkan ilmu yang diperoleh dari data yang ada, baik data sebelumnya maupun data pada saat ini dengan bantuan label untuk memprediksi kejadian di masa depan.

   Pendekatan ini dimulai dari proses *training* yang dilakukan pada *dataset* dan *Machine Learning* ini sendiri mengembangkan fungsi yang telah disimpulkan untuk memperkirakan nilai yang muncul pada hasil keluarannya. Sistem ini mampu untuk menyediakan hasil terhadap sebuah data yang dimasukkan dengan proses pelatihan yang memadai.

   Algoritma *Machine Learning* melakukan perbandingan terhadap hasil yang didapatkan dengan hasil yang aktual dan digunakan untuk mengidentifikasi kesalahan yang muncul untuk mengganti modelnya berdasarkan hasil yang ada.

2. *Unsupervised Machine Learning*

   Teknik dalam *Machine Learning* yang kedua bernama *Unsuperivsed Machine Learning*. Teknik ini dipergunakan ketika tidak dilakukan klasifikasi terhadap data yang digunakan pada proses *training* dan tidak juga diberi label. *Unsupervised Machine Learning* menganalisa bagaimana sistem dapat menyimpulkan sebuah fungsi untuk menjelaskan pola yang tersembunyi dalam data yang belum dilabeli. Sistem yang dimiliki oleh teknik ini tidak mengidentifikasi hasil luarannya secara tepat, tetapi menemukan datanya dan menuliskan hasil pengamatan yang didapat dari *dataset* untuk menemukan pola yang tersembunyi tersebut dari data yang belum diberi label.

3. *Semi-Supervised Machine Learning*

   Teknik *Semi-Supervised Machine Learning* terletak di antara teknik *supervised* dan *unsupervised* ketika menggunakan data yang telah dilabeli dan yang belum dilabeli untuk keperluan proses pelatihan datanya. Lazimnya, teknik ini memperhitungkan kuantitas yang lebih besar pada data yang belum dilabeli dan kuantitas yang lebih kecil pada data

yang sudah diberi label. Jenis teknik ini dapat menyesuaikan untuk memperoleh akurasi yang lebih tinggi.

4. *Reinforcement Machine Learning*

   Reinforcement Machine Learning berinteraksi dengan lingkungan sekelilingnya berdasarkan tindakan yang dilakukan dan menempatkan galat atau imbalan yang didapatkan. Fitur umum dari teknik *Reinforcement Machine Learning* ini adalah penundaan imbalan yang didapatkan serta melakukan pencarian yang melewati proses *trial and error*. Fitur tersebut memungkinkan sistem dan program perangkat lunak untuk mengidentifikasi perilaku yang ideal dari sebuah konteks yang spesifik untuk meningkatkan kinerja (Saravanan & Sujatha, 2019).

2.5 Media Sosial

Menurut Anang Sugeng Cahyono dalam (Istiani & Islamy, 2020), media sosial merupakan sebuah medium yang berbasis daring. Media ini memungkinkan seluruh penggunanya dapat berpartisipasi dengan sangat mudah dalam keseluruhan aktivitasnya. Besarnya manfaat dari sosial media pada saat ini tidak dapat dipungkiri pengaruhnya dalam tatanan sosial saat ini. Mengenai beberapa fungsi dari media sosial akan dijabarkan sebagai berikut:

1. Sebagai media untuk mendukung perluasan dan demokratisasi dalam bidang informasi dan pengetahuan serta dapat melakukan transformasi dari pengguna isi suatu pesan menjadi pencipta pesan tersebut.
2. Sebagai perluasan dari interaksi antar manusia secara sosial yang tidak terbatas oleh cakupan geografis dengan memanfaatkan teknologi berupa internet dan web.
3. Sebagai media untuk mendukung transformasi jenis komunikasi dari yang bentuknya searah yang dilakukan antara sebuah institusi media ke berbagai macam khalayak yaitu sebagai media penyiaran (*one to many)* menjadi komunikasi berwujud dialogis antar kelompok maupun individu (*many to many)* (Purbohastuti, 2017).

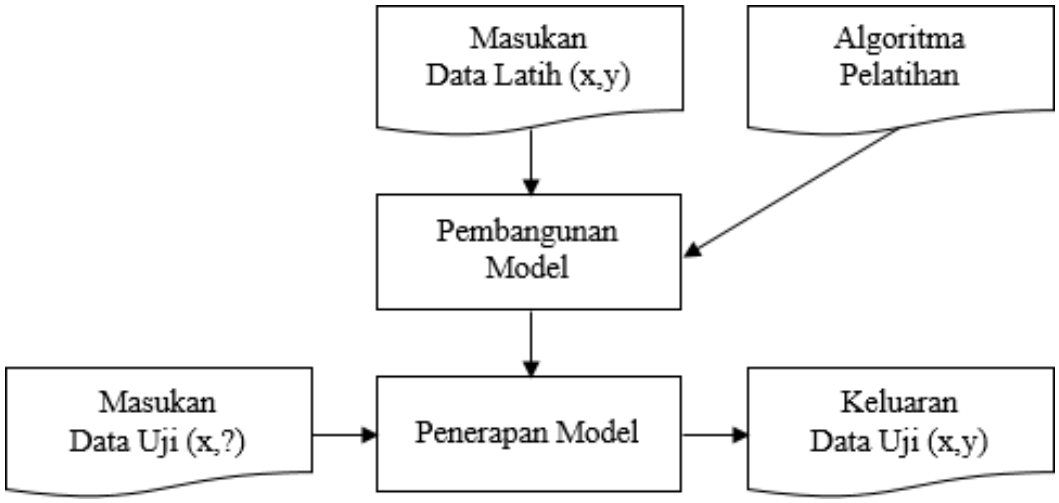
2.6 Klasifikasi

Menurut Han & Kamber yang disadur dalam (Febriyanti, 2020), klasifikasi merupakan sebuah teknik untuk melakukan analisis terhadap data yang dipergunakan untuk dapat menghasilkan sebuah model prediksi yang berguna untuk mendeskripsikan sebuah label ataupun kelas data. Penjabaran lain mengenai klasifikasi juga dijelaskan oleh Prasetyo dalam

(Imron, 2019) yaitu sebuah pekerjaan menilai sebuah objek data untuk dapat dimasukkan ke dalam kelas tertentu dari sejumlah kelas yang tersedia.

Dalam prosesnya, terdapat 2 langkah dalam pemrosesannya. Langkah pertama adalah fase *learning* (fase *training)*, pada fase ini, terjadi proses pembuatan algoritma klasifikasi dengan tujuan untuk melakukan analisis terhadap data *training* dan hasil representasinya berbentuk aturan klasifikasi. Sedangkan langkah kedua dalam proses ini adalah aturan klasifikasi diperkirakan akurasinya dengan menggunakan data tes (Mulia, 2021).

Kotak hitam memiliki makna yang sama dengan model dalam klasifikasi, di mana terdapat suatu model yang berfungsi untuk menerima masukan dan model tersebut mampu untuk melakukan pemikiran mengenai masukan tersebut dan menghasilkan jawaban sebagai bentuk luaran dari hasil pemikiran model tersebut. Kerangka kerja dalam klasifikasi ditunjukkan pada Gambar 2.1. Pada gambar tersebut data latih (x,y) disediakan untuk dapat dipergunakan sebagai data pembangun model. Lalu, untuk mengetahui kelas y yang sebenarnya, dilakukan penerapan terhadap model yang telah dibuat untuk dapat melakukan prediksi kelas dari data uji (x,?).



Gambar 2.1 Proses Klasifikasi

Sumber: (Imron, 2019)

Framework yang dipertunjukkan pada Gambar 2.1 meliputi dua tahapan proses, yaitu proses induksi dan proses deduksi. Proses Induksi atau yang dapat disebut juga dengan proses pelatihan merupakan tahapan dalam untuk membangun model klasifikasi berdasarkan data latih yang telah diberikan. Tahapan yang kedua yaitu proses deduksi, atau yang disebut juga dengan proses prediksi. Proses ini merupakan sebuah langkah penerapan model yang telah dibangun pada proses induksi untuk dapat mengetahui kelas yang sesungguhnya dari data uji.

2.7 Naïve Bayes Classifier

Naive Bayes Classifier merupakan sebuah metode *classifier* yang berlandaskan probabilitas serta Teorema Bayesian yang diasumsikan bahwa setiap Variabel X sifatnya adalah bebas atau *independent*. Dapat dikatakan juga, diasumsikan dalam NBC bahwa tidak ada keterkaitan antara keberadaan sebuah variabel dengan keberadaan variabel lainnya (Buntoro, 2017).

Cara kerja metode NBC ini adalah dengan melakukan prediksi terhadap peluang di masa depan berlandaskan pengalaman pada masa lalu. Metode *Naïve bayes* ini merupakan salah satu metode klasifikasi yang digunakan secara luas. Model NBC memiliki basis matematis yang kokoh serta efisiensi klasifikasi yang stabil. Dalam waktu yang bersamaan, beberapa parameter dari model NBC butuh untuk diestimasikan. Parameter-parameter ini sedikit kurang sensitif terhadap data yang hilang (*missing data)* (J. Liu et al., 2016). Adapun *Naïve Bayes* ini sendiri dirumuskan dalam Persamaan 2.1.

$$P(H|X) = \frac{P(H|X)P(H)}{P(X)} \qquad (2.1)$$

Keterangan :

X = Data dengan *class* yang belum diketahui

H = Hipotesis data X merupakan suatu *class* spesifik

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

P(H) = Probabilitas hipotesis H

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

P(X) = Probabilitas dari X (Muslehatin et al., 2017).

Aturan Bayes dalam (Imron, 2019) adalah sebagai berikut:

Jika terdapat P(H1|X) lebih kecil dari (<) P(H2|X), maka X masuk ke dalam klasifikasi sebagai H2. Pernyataan P(H1|X) mengindikasikan probabilitas yang dimiliki oleh hipotesis H1 berdasarkan kondisi x terjadi, demikian juga dengan H2. Maka dari itu, dapat ditarik kesimpulan bahwa klasifikasi yang berasal dari x sesuai dengan probabilitas yang paling besar di antara probabilitas x terhadap keseluruhan kelas.

Naïve Bayes Classifier sendiri dijadikan pilihan sebagai metode klasifikasi dalam penelitian ini karena beberapa alasan, yang pertama adalah waktu pelatihan yang dibutuhkan oleh metode ini sangat singkat, sehingga mempercepat proses pelatihan dan menghemat *resource* dari komputer secara keseluruhan. Alasan kedua adalah performanya yang cukup bagus dalam melakukan klasifikasi, seperti pada penelitian yang dilakukan oleh (Normawati & Prayogi, 2021), dimana pemanfaatan *Naïve Bayes* menghasilkan akurasi sebesar 82%, precision 93%, dan recall sebesar 52%. Penelitian lain yang menggunakan metode ini yaitu (Mulia, 2021), pemanfaatan metode ini menghasilkan nilai akurasi yang lebih besar, yaitu 91% dengan nilai *recall* dan *precision* sebesar 91% dan 82%. Alasan terakhir dalam memilih metode ini adalah *Naïve Bayes Classifier* membutuhkan data yang relatif kecil untuk dapat dilatih. Proses pelatihan metode ini sangat cepat, dan membutuhkan ruang penyimpanan yang kecil pada saat proses pelatihan dan klasifikasi (Al-Aidaroos et al., 2010).

2.8 *Support Vector Machine* (SVM)

Pada tahun 1992, dilakukan presentasi yang berlokasi di Annual Workshop on Computational Learning Theory mengenai sebuah model klasifikasi bernama *Support Vector Machine* yang dikembangkan oleh Guyon, Boser, dan Vapnik. Secara faktual, konsep yang paling mendasar dari SVM ini sendiri merupakan gabungan dari variasi teori mengenai komputasi yang telah ada puluhan tahun sebelum 1992, contohnya adalah margin *hyperplane*, kernel, dan juga konsep pendukung lainnya (A. S. Nugroho, 2007).

Dalam permasalahan yang terkait dengan pengklasifikasian, SVM melakukan identifikasi terhadap *hyperplane* yang terpisah yang mengoptimalkan margin di antara dua kelas secara maksimal. Target utama dari model SVM ini adalah untuk memperkirakan sebuah fungsi klasifikasi dengan mempergunakan data pelatihan input dan output dari kedua kelas tersebut $(x_1, y_1), \ldots, (x_n, y_n) \in R^m \times \{\pm 1\}$.

Tujuan yang ingin dicapai oleh fungsi klasifikasi ini sendiri adalah untuk mendirikan sebuah persamaan *hyperplane* yang melakukan pemisahan terhadap data *training* dan meninggalkan keseluruhan titik pada kelas yang sama dalam sisi yang sama. Juga, fungsi klasifikasi ini memaksimalkan jarak minimum antara *hyperplane* dan kedua kelas yang ada $(w,\ b)$. Di mana w merepresentasikan bobot vektor yang mewujudkan sebuah margin fungsional dari 1 pada poin positif x^+dan juga untuk poin negatif x^- (Chou et al., 2014)

Alasan dalam memutuskan untuk memilih metode *Support Vector Machine* ini sebagai *classifier* adalah kemampuannya untuk menghasilkan akurasi yang tinggi pada hasil proses klasifikasi. Hal ini dibuktikan dengan penelitian yang dijalankan oleh (Kristiyanti, 2015) dengan topik Review Produk Kosmetik dengan menggunakan metode seleksi fitur SVM dan *Particle Swarm Optimization*. Akurasi yang dihasilkan pada penelitian ini dengan pemanfaatan SVM dinilai sangat baik, karena mencapai angka 89%. Metode ini juga memiliki kelebihan lain yaitu mampunya metode ini untuk mengidentifikasi *hyperplane* terpisah yang mengoptimalkan margin antar kelas.

2.9 Term Frequency – Inverse Document Frequency

Term-Frequency-Inverse Document Frequency yang biasa disingkat dengan (*TF-IDF*) merupakan sebuah proses untuk dapat mentransformasi data dari bentuk tekstual ke dalam bentuk numerik untuk dapat dilakukan pembobotan pada tiap fitur atau kata. Dalam sebuah dokumen, *TF-IDF* yang merupakan sebuah ukuran statistik ini dimanfaatkan untuk melakukan evaluasi terhadap bobot kepentingan sebuah kata.

Term-Frequency (TF) merupakan frekuensi munculnya kata pada setiap dokumen yang diberikan dan *TF* ini sendiri menunjukkan seberapa penting sebuah kata tersebut dalam dokumen tersebut. *Document Frequency* (DF) menunjukkan seberapa umum suatu kata muncul dalam dokumen-dokumen yang diperiksa. *Inverse Document Frequency* (IDF) sendiri merupakan nilai *inverse* (kebalikan) dari DF.

Nilai pembobotan pada tiap kata dari pemrosesan yang memanfaatkan *TF-IDF* ini merupakan hasil dari perkalian antara IDF dengan TF. Bobot kata akan semakin besar jika frekuensi munculnya kata tersebut adalah sering dan apabila kata tersebut muncul dalam banyak dokumen, bobotnya akan semakin kecil (Septian et al., 2019).

Perhitungan bobot dari setiap token x pada dokumen dapat dirumuskan di Persamaan 2.2.

$$W_{dx} = tf_{dx} \times IDF_x \quad (2.2)$$

Keterangan:

W_{dx} : bobot dari x (kata) dalam satu dokumen

tf_{dx} : frekuensi kemunculan x (kata) dalam dokumen

IDF_x : *Invesed Document Frequency*

Nilai IDF_x didapatkan dari perhitungan yang dirumuskan dalam Persamaan 2.3.

$$IDF_x = \log(\frac{N}{n_x}) \quad (2.3)$$

Keterangan:

N : jumlah semua dokumen

n_x : jumlah dokumen yang mengandung kata x

Setelah bobot (W) di masing-masing kata yang terdapat dalam dokumen diketahui, tahapan selanjutnya adalah tahap *sorting*. Yang mana ketika nilai bobotnya semakin besar, maka derajat kesamaan dokumen pada kata kunci tersebut juga akan semakin membesar, dan dapat juga berlaku sebaliknya (Qorita, 2022).

2.10 Confusion Matrix

Confusion Matrix dapat dijabarkan sebagai tabel yang berisikan pengelompokan jumlah data uji yang benar dan yang salah. Contoh dari tabel *Confusion Matrix* untuk diterapkan pada proses klasifikasi positif dan negatif ditunjukkan pada Tabel 2.1.

Tabel 2. 1 Contoh Tabel Confusion Matrix

		Kelas Prediksi	
		Positif	Negatif
Kelas Sebenarnya	Positif	TP	FN
	Negatif	FP	TN

Keterangan:

TP *(True Positive)* = Jumlah dokumen yang berasal dari kelas Positif yang diklasifikasikan dengan tepat sebagai kelas Positif

TN *(True Negative)* = Jumlah dokumen yang berasal dari kelas Negatif yang diklasifikasikan dengan tepat sebagai kelas Negatif

FP *(False Positive)* = Jumlah dokumen yang berasal dari kelas Negatif yang salah diklasifikasikan dengan tepat sebagai kelas Positif

FN *(False Negative)* = Jumlah dokumen yang berasal dari kelas Positif yang salah diklasifikasikan sebagai kelas Negatif (Normawati & Prayogi, 2021).

2.11 Performance Evaluation Measure

Tujuan dari dilakuknannya *Performance Evaluation Measure* adalah untuk melakukan evaluasi terhadap model yang sebelumnya telah dibuat dan untuk menunjukkan performa sebenarnya berlandaskan data yang dihasilkan oleh algoritma yang dipergunakan. Banyak metode perhitungan agar nilai PEM dapat ditemukan, metode tersebut biasanya diterapkan secara kombinasi maupun parsial. Berikut ditunjukkan beberapa perhitungan dalam PEM antara lain:

a. *Precision*

Precision merupakan tingkat ketepatan antara *request* yang diajukan oleh pengguna dengan jawaban yang diberikan oleh sistem. *Precision* dirumuskan dalam Persamaan 2.4.

$$Precision = \frac{TP}{TP+FP} \qquad (2.4)$$

b. *Accuracy*

Accuracy merupakan pembanding antara informasi yang menjadi jawaban dari sistem dengan informasi secara keseluruhan. *Accuracy* dirumuskan dalam Persamaan 2.5.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \qquad (2.5)$$

c. *Recall*

Recall merupakan ukuran ketepatan antara informasi yang sebelumnya sudah pernah dipanggil dengan informasi yang sama. *Recall* dirumuskan dalam Persamaan 2.6 (Imron, 2019).

$$Recall = \frac{TP}{TP+FN} \qquad (2.6).$$

d. *F1-Score*

F1-Score merupakan sebuah nilai yang didapatkan dari proses penghitungan nilai rataan harmonik dari kedua ukuran ketepatan yaitu *recall* dan *precision*. Skor tertinggi yang terdapat dalam penghitungan *f1-score* adalah 1 dan skor terendahnya adalah 0. *F1-Score* dirumuskan dalam Persamaan 2.7 (Irfan, 2022).

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (2.7).$$

2.12 SMOTE

Teknik *Synthetic Minority Oversampling Technique* (SMOTE) dijabarkan sebagai metode yang melakukan penyeimbangan terhadap jumlah distribusi pada data sampel pada kelas

minoritas. Metode yang digunakan sebagai penyeimbang jumlah data sampel adalah penyeleksian data. Data diseleksi dan diseimbangkan hingga selaras dengan jumlah data sampel yang terdapat dalam kelas mayoritas (Siringoringo, 2018). Proses *oversampling* ini menghasilkan replikasi kelas minoritas secara acak yang berasal dari *dataset*. Teknik ini menyebabkan proses pelatihan yang dilakukan terhadap model dilakukan secara berulang kali dan cara ini tidak efektif untuk mengatasi masalah *over-fitting* pada *dataset*.

Terdapat tiga tahapan yang dilalui oleh teknik SMOTE dalam melakukan *oversampling*. Tahapan pertama adalah memilih sampel dari sebuah kelas minoritas secara acak, selanjutnya pada tahapan kedua dilakukan pemilihan terhadap sebuah sampel dari *k-neighbouring* sampelnya. Tahapan yang terakhir adalah membuat sampel baru dengan menggunakan *Linear Interpolation* yang berasal dari dua sampel yang dipilih. *Linear Interpolation* sendiri memperkirakan sebuah titik dari dua titik yang telah ada dengan cara menggambar sebuah garis linear dari titik berikut, $(x_a,\ y_a)$ dan $(x_b,\ y_b)$. *Linear Interpolation* dirumuskan dalam Persamaan 2.8 (Nurhopipah et al., 2021) .

$$y = y_a + (y_b + y_a)(x - x_a)(x_b - x_a) \quad (2.8).$$

2.13 Penelitian Serupa

Pada proses pengerjaan penelitian ini, terdapat penelitian-penelitian yang sudah dilakukan sebelumnya dan terdapat juga penelitian serupa yang dijadikan acuan dalam pengerjaan penelitian ini. Pengacuan terhadap hasil riset sebelumnya mengenai topik yang terkait dengan penelitian ini yaitu analisis sentimen merupakan sebuah tindakan yang esensial dengan tujuan mencegah adanya duplikasi dalam penelitian yang dilaksanakan penulis.

Sebuah penelitian yang dilaksanakan oleh (Duei Putri et al., 2022) mengenai analisis sentimen pada platform Twitter mengenai kinerja Dewan Perwakilan Rakyat Republik Indonesia menghasilkan Berdasarkan pemanfaatan algoritma Naïve Bayes pada penelitian ini, diperoleh hasil berupa *accuracy score* sebesar 0.8 atau 80%. Hal ini menunjukkan bahwa sistem dapat memprediksi 80% secara tepat dari total data *testing* yang berjumlah 20%. Hasil analisis yang didapat memungkinkan untuk digunakan untuk melakukan prediksi terhadap *dataset* baru dengan *labeling* yang tidak perlu terlebih dahulu dilakukan. Dari hasil analisis sistem berdasarkan data hasil *crawling* yang berjumlah sebanyak 1546 data, terdapat klasifikasi tweet mengenai DPR sebanyak 758 negatif, 95 positif, dan 693 netral.

Kajian mengenai analisis sentimen turut dilakukan oleh (Vitandy et al., 2019) dengan memilih topik berupa Evaluasi Kinerja Dosen yang memanfaatkan metode *Term Frequency-*

Inverse Document Frequency dan *Naïve Bayes Classifier*. Berdasarkan hasil penelitian yang telah dilakukan terhadap kuesioner pada Semester Genap 2016/2017, Ganjil 2017/2018, dan Genap 2017/2018, terdapat komentar positif sebanyak 837 komentar, negatif sebanyak 296 komentar, dan netral sebanyak 364 komentar. Selain dari itu, setelah klasifikasi dilakukan dengan memanfaatkan metode *TF-IDF* dan Naïve Bayes Classifier, dilaksanakan juga pengujian terhadap hasil klasifikasi yang ada pada setiap semesternya. Hasil dari proses pengujian tersebut dirata-rata dan memperoleh hasil berupa Recall sebesar 80,3%, F1-Score sebesar 80%, Accuracy sebesar 80,1%, dan Precision sebesar 80,3%.

Sebuah penelitian yang dilakukan oleh (Amirul Haj et al., 2020) dengan topik bahasan analisa sentimen yang ditujukan terhadap kinerja Komisi Pemilihan Umum pada saat terjadinya pemilihan umum di tahun 2019. Dalam penelitian ini, terdapat tingkat akurasi berbeda yang didapatkan dari 2 skenario. Pada skenario 1, algoritma *K-Means* hanya dimanfaatkan tanpa adanya bantuan dari algoritma stemming dan normalisasi kata. Hasil akurasi yang didapat pada skenario 1 adalah 84%. Pada skenario 2, algoritma *K-Means* dikombinasikan dengan algoritma *Levensthein Distance* yang dimanfaatkan sebagai algoritma normalisasi kata dan algoritma *Confix Stripping Stemmer* sebagai algoritma stemming yang mana pada skenario 2 ini terjadi peningkatan akurasi menjadi 86%.

Dalam sebuah penelitian yang dijalankan oleh (Dermawan et al., 2021.), sebuah penelitian yang juga membahas topik analisis sentimen mengenai kinerja kegiatan kepala daerah Kota Lubuk Linggau yang menghasilkan sebuah kesimpulan bahwa terdapat 150 komentar baik yang positif dan negatif pada *dataset* yang digunakan dalam penelitian ini dan menghasilkan akurasi sebesar 83%. Pada hasil klasifikasi komentar masyarakat, masih terdapat kesalahan pada Predict yaitu adanya 5 kesalahan dengan metode *Naïve Bayes* dari data actual aslinya.

Topik serupa turut diteliti oleh (Permana & Makmun, 2020) namun dengan objek penelitian kinerja dosen dengan memanfaatkan Algoritma *K-Nearest Neighbor* (KNN). KNN dimanfaatkan sebagai pendekatan dalam proses analisis sentimen terhadap teks opini yang dijalankan. Hasil dari prediksi sentimen yang dilakukan pada penelitian ini dari keseluruhan kelas sentimen, nilai akurasinya mencapai 80%, *recall* bernilai 0.889 dan *precision* bernilai 0.909.

Sebuah penelitian yang dijalankan oleh (Qudsi et al., 2021) mengenai analsis sentimen tentang data saran mahasiswa terhadap kinerja departemen yang terdapat pada perguruan tinggi dengan memanfaatkan algoritma *Convolutional Neural Network* (CNN). Dari hasil pengujian yang telah dilaksanakan pada penelitian ini, didapat sebuah kesimpulan. *DoubleMax* CNN

dengan filter yang berukuran 5 memiliki hasil yang terbaik pada proses klasifikasi 2 dan 3 kelas sentimen. *DoubleMax* CNN mendapatkan F1-*Score* dan akurasi sebesar 98% pada klasifikasi 2 kelas sentimen.

Dalam proses klasifikasi 3, nilai F1-*Score* dan akurasi yang didapat oleh *DoubleMax* CNN sebesar 90% dan 91%. Jika dilakukan perbandingan terhadap metode klasik *Machine Learning* seperti *Naïve Bayes,* SVM, dan *Logic Regression*, rataan *DoubleMax* CNN dalam bekerja adalah 17% lebih baik dalam melakukan klasifikasi sentimen data saran.

Pada tahun 2019, (Ali et al., 2019) menganalisis sentimen masyarakat terhadap kinerja Presiden Indonesia dalam aspek ekonomi, kesehatan, dan pembangunan dengan basis opini yang berasal dari Twitter. Penelitian ini menghasilkan kesimpulan sebagai berikut Pada Tahap *pre-processing* dengan menggunakan *Non-Standard Word Handling* yang memanfaatkan 10 *fold cross validation* memiliki hasil lebih tinggi dengan rata-rata 84,75% yang mana selisih yang didapat tidak jauh yaitu 0,07%.

Akurasi yang dimiliki oleh *Lexicon SentiWordnet* yang digunakan pada pengujian ekstraksi fitur lebih tinggi jika dibandingkan dengan fitur ekstraksi lainnya yaitu 84.75% dengan selisih 0.13% dengan metode *TF-IDF* dan penggabungan antara *Lexicon SentiWordnet* dan *TF-IDF*. Proses klasifikasi yang memanfaatkan algoritma *Naïve Bayes* menunjukkan hasil yaitu adanya 52,7% masyarakat memberikan opini positif terkait dengan kinerja Presiden Republik Indonesia pada aspek ekonomi, sebanyak 53,52% opini positif diberikan pada aspek kesehatan dan 63,88% opini positif diberikan pada aspek pembangunan.

Gambaran singkat tentang penelitian yang membahas topik analisis sentimen disajikan dalam Tabel 2.2 yang berisikan tentang perbandingan nilai akurasi dan metode yang dipergunakan pada penelitian mengenai topik serupa yang telah dilakukan sebelumnya.

Tabel 2. 2 Penelitian Terdahulu

No	Nama Peneliti	Judul	Metode	*Dataset*	Nilai Akurasi
1	(Duei Putri et al., 2022)	Analisis Sentimen Kinerja Dewan Perwakilan Rakyat (DPR) pada Twitter Menggunakan Metode *Naive Bayes Classifier*	NBC	Twitter	80%
2	(Vitandy et al., 2019)	Analisis Sentimen Evaluasi Kinerja Dosen menggunakan *Term Frequency-Inverse Document Frequency* dan *Naïve Bayes Classifier*	NBC	Kuesioner Sistem Informasi Akademik	80,1%
		Analisis Sentimen Kinerja KPU Pemilu	*K-Means*	Facebook	84%

3	(Amirul Haj et al., 2020)	2019 Menggunakan Algoritma *K-Means* Dengan Algoritma *Confix Stripping Stemmer*	*K-Means* dengan *Confix Stripping Stemmer*		86%
4	(Dermawan et al., 2021)	Analisis Sentimen Komentar Kinerja Kegiatan Kepala Daerah Kota Lubuk Linggau	NBC	Facebook	83%
5	(Permana & Makmun, 2020)	Analisis Sentimen pada Teks Opini Penilaian Kinerja Dosen dengan Pendekatan Algoritma KNN	KNN	Dokumen teks opini dari Mahasiswa	80%
6	(Qudsi et al., 2021)	Analisis Sentimen pada Data Saran Mahasiswa Terhadap Kinerja Departemen di Perguruan Tinggi Menggunakan *Convolutional Neural Network*	*Simple* CNN 2 Kelas	Data saran dari BP3M Politeknik Caltex Riau	97%
			DoubleMax CNN 2 Kelas		98%
			Simple CNN 3 Kelas		91%
			DoubleMax CNN 3 Kelas		91%
7	(Ali et al., 2019)	Analisis Sentimen Masyarakat Terhadap Kinerja Presiden Indonesia Dalam Aspek Ekonomi, Kesehatan, dan Pembangunan Berdasarkan Opini dari Twitter	NBC	Twitter	90,50%

BAB III
METODOLOGI PENELITIAN

3.1 Alur Penelitian

Alur dari riset mengenai analisis sentimen terhadap kinerja POLRI ditampilkan pada Gambar 3.1. Rangkaian proses tersebut dimulai dari pengumpulan data, lalu melabeli data yang telah didapatkan dari proses pengumpulan data, selanjutnya data yang telah dilabeli dilakukan *preprocessing* terhadapnya. Setelah data bersih dan siap dipergunakan, data tersebut dimasukkan ke dalam proses ekstraksi fitur.

Ketika proses ekstraksi fitur selesai dilakukan, selanjutnya adalah melakukan proses *Imbalanced Data Handling* terlebih dahulu. Alasannya dikarenakan *dataset* yang digunakan pada penelitian ini tidak seimbang dalam pembagian kelasnya, sehingga hal ini dapat mempengaruhi performa model dalam melakukan klasifikasi. Setelah data diseimbangkan pembagian kelas sentimennya, dilakukan proses pelatihan terhadap model klasifikasi yang memanfaatkan algoritma *Naïve Bayes* dan *Support Vector Machine*.

Setelah model dilatih, proses selanjutnya adalah melakuakn evaluasi terhadap performa model dalam melakukan prediksi. Cara untuk mengevaluasi performa model adalah dengan membuat *Confusion Matrix* dan *Performance Evaluation Measure.* Dalam *Performance Evaluation Measure,* berisikan nilai *accuracy, precision, recall,* dan *f1-score*. Ketika proses evaluasi terhadap model selesai, langkah selanjutnya adalah melakukan proses penyimpanan model. Hal ini dilakukan dengan tujuan agar ketika sewaktu-waktu ingin menggunakan model klasifikasi ini kembali, tidak perlu untuk melakukan pelatihan ulang terhadap data. Keseluruhan proses yang dilakukan pada penelitian ini memanfaatkan Jupyter Notebook dengan bahasa pemrograman *Python* 3.11.3. Berikut ditunjukkan spesifikasi perangkat keras dan perangkat lunak beserta *environment* yang dipergunakan dalam penelitian ini.

1. Analisis kebutuhan *hardware*

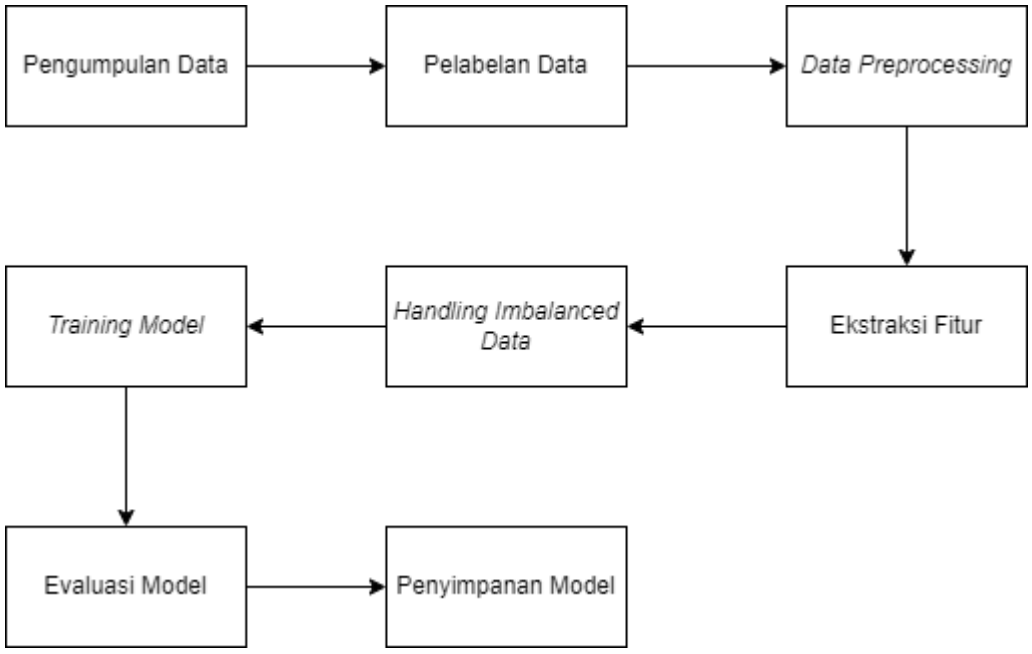
Perangkat keras yang dipergunakan dalam melakukan penelitian ini, di antaranya:

a. Processor : 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.3GHz
b. GPU : NVIDIA GeForce RTX 3060 Laptop GPU
c. RAM : 16.0 GB
d. SSD : 500 GB

2. Analisis kebutuhan *software*

Perangkat lunak yang dipergunakan dalam melakukan penelitian ini, di antaranya:

a. Python 3.11.3 (tags/v3.11.3:f3909b8)
b. Windows 11 Home Single Language 64-bit (10.0, Build 22621)
c. Tweepy 4.14.0, Pandas 1.5.3, Re 2.2.1, Numpy 1.23.5, Sastrawi 1.0.1
d. NLTK 3.8.1, Swifter 1.3.4, Sklearn 1.2.2, Imblearn 0.10.1, Pickle 4.0
e. Python Environment *not set*.



Gambar 3. 1 Alur Proses Analisis Sentimen

3.2 Pengumpulan Data

Data penelitian ini bersumber dari kumpulan *tweet* yang ada di media sosial Twitter yang membahas tentang kinerja POLRI. Batasan yang ditetapkan yaitu hanya *tweet* yang menggunakan Bahasa Indonesia yang dipilih. T*weet* yang diambil hanya mulai dari tanggal 28 September hingga 6 Oktober 2022 dan 11 Desember hingga 20 Desember 2022.

Proses *Crawling Data* dilakukan pada kedua periode ini dengan alasan kejadian kanjuruhan sedang menjadi pembahasan yang cukup panas. Pada masa tersebut proses sidang terhadap kasus Ferdi Sambo juga sedang dilakukan sehingga banyak cuitan-cuitan di *Twitter* yang muncul membahas topik ini. Elemen dalam data *tweet* yang termasuk dalam analisis hanyalah teks keseluruhan dari *tweet* tersebut. *Hashtag*, *emoji*, dan lain-lain tidak dimasukkan ke dalam analisis. Media sosial *Twitter* dijadikan pilihan dalam mengumpulkan data karena

basis aplikasinya yang berjenis teks, sehingga pengumpulan data menjadi lebih mudah. Dari media sosial *Twitter* akan dilakukan pengambilan data dengan proses *scraping* yang memanfaatkan *Library* Python bernama Tweepy.

Digunakan *search query* berupa kata "kinerja polri" untuk melakukan pencarian *tweet* dari aplikasi Twitter. Terdapat 3275 *tweet* yang didapatkan dari hasil penggabungan antara hasil *crawling* data pada bulan oktober dan hasil pada bulan desember. Namun, sebelum dilakukannya *pre-processing* terhadap data, dijalankan proses penghapusan data *tweet* yang memiliki isi duplikat dengan kode *Python.*

3.3 Pelabelan Data

Tahap Pelabelan Data merupakan tahap selanjutnya setelah data dikumpulkan melalui proses *Crawling Data* untuk dapat dilakukan pelabelan secara manual terhadapnya. Pada penelitian ini, proses pelabelan dilakukan oleh penulis secara manual dengan membaca satu persatu data *tweet* hasil *crawling*. Kemudian, masing-masing data *tweet* dilabeli sesuai sentimen yang terkandung didalamnya dan dilabeli ke dalam kelas sentimen positif, netral, dan negatif.

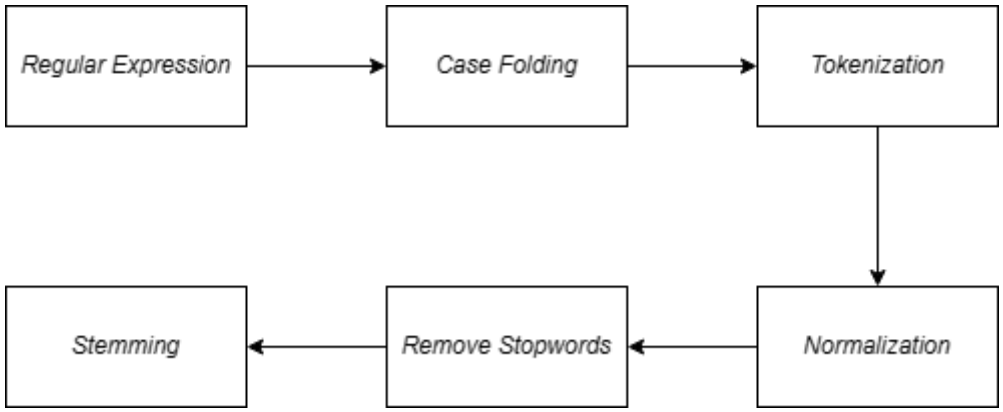
Adapun kriteria yang ditetapkan untuk menentukan sentimen suatu kalimat itu termasuk ke dalam label positif, negatif, atau netral ketika dilakukan proses pelabelan dirujuk dari buku yang ditulis oleh (B. Liu, 2012), yang memiliki judul: *Sentiment Analysis and Opinion mining*. Pada buku ini, dinyatakan bahwa kata-kata sentimen diartikan sebagai kata yang terdapat di dalam sebuah bahasa yang dipergunakan untuk mengekspresikan sentimen positif maupun negatif.

Beberapa kata yang menggambarkan sentimen positif contohnya adalah bagus, luar biasa, semangat, bahagia. Kata yang menggambarkan sentimen yang negatif di antaranya adalah buruk, jelek, mengerikan (B. Liu, 2012). Sentimen netral digambarkan dengan penggunaan kata yang tidak mengandung sentimen apapun, maka *tweet* tersebut dilabeli dengan sentimen netral. (Sorgente, Antonio et al., 2022). Sebagian besar kata-kata sentimen terdiri atas kata sifat dan kata keterangan, namun kata benda dan kata kerja juga dapat dipergunakan untuk mengekspresikan berbagai macam sentimen. Selain kata-kata, terdapat juga frasa dan idiom yang menunjukkan sentimen, contohnya: tinggi hati, gelap mata, darah daging.

3.4 Pre-processing Data

Tahapan ini merupakan tahap yang berfungsi sebagai penyeleksi data yang telah dikumpulkan untuk dapat diubah menjadi bentuk yang terstruktur. Pada tahapan ini juga dilakukan pembersihan terhadap data agar tidak lagi mengandung kata atau karakter yang tidak terdapat makna berarti di dalamnya. Tujuan pembersihan ini sendiri adalah untuk dapat memudahkan proses analisis yang akan dilakukan selanjutnya. Proses *Data Preprocessing* ini sendiri dilakukan dengan memanfaatkan *software Python.*

Mengenai tahapan *Preprocessing* ini, terdapat beberapa langkah dalam pengerjaannya. Langkah pertama adalah menerapkan *Regular Expressions,* kemudian melakukan *Case Folding,* selanjutnya melakukan *Tokenization* pada data. Setelah data ditokenisasi, dilakukan proses *Normalization* untuk kemudian dilakukan proses *Remove Stopwords*, dan diakhiri dengan melakukan *Stemming.* Berikut ditunjukkan alur tahapan *preprocessing* pada Gambar 3.2.



Gambar 3. 2 Tahapan *Data Preprocessing*

Berikut dipaparkan langkah-langkah dalam tahapan *Preprocessing*:

a. *Regular Expression*

Dalam tahap preprocessing, *regular expression* memiliki tujuan untuk melakukan *fetching* karakter huruf alphabet ‘a’ sampai ‘z’ yang terdapat dalam data *tweet*. Proses ini menghilangkan karakter lain di luar alphabet seperti *hashtag*, *username*, angka, tanda baca. Pada penelitian ini, *Regular Expression* dipergunakan karena data *tweet* yang didapatkan mengandung banyak sekali karakter di luar alphabet yang disebabkan oleh platform Twitter itu sendiri yang merupakan sarana untuk bertukar opini,

sehingga banyak karakter bebas yang digunakan untuk mengekspresikan diri dalam bentuk *tweet*.

b. Case Folding

Case Folding dapat didefinisikan sebagai sebuah proses yang melakukan pengubahan huruf yang terdapat dalam data menjadi kecil atau besar semua sesuai dengan kebutuhan dalam proses pengolahan data. Penggunaan *case folding* pada penelitian ini adalah pengubahan huruf yang ada pada *tweet* menjadi kecil seluruhnya. Tahapan *Case Folding* ini perlu dilakukan untuk mengurangi variasi pada data dan menjaga serta memastikan konsistensi data pada pemrosesan teks.

c. Tokenization

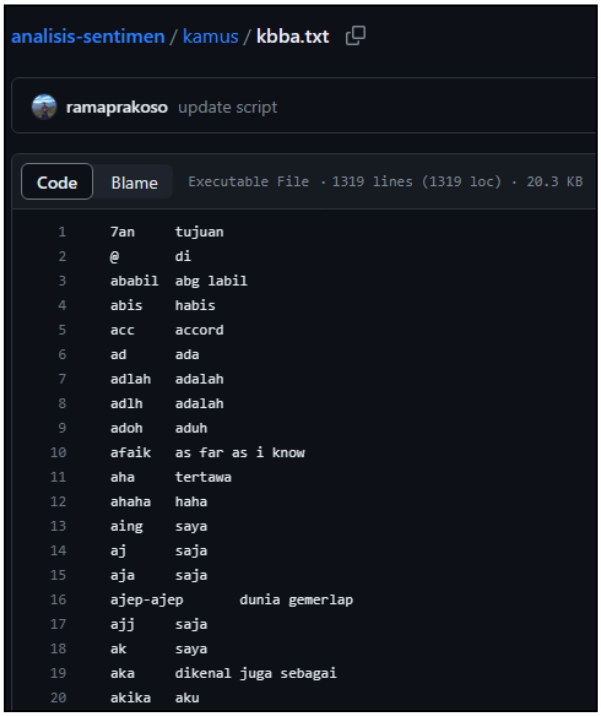
Pada tahapan ini, terjadi pemotongan terhadap kata pada dokumen atau dapat dijelaskan juga sebagai proses pemisahan teks menjadi potongan kalimat atau kata yang biasa disebut sebagai token agar data menjadi lebih mudah untuk dianalisa. Dipergunakannya proses *tokenization* pada penelitian ini adalah untuk dapat mempermudah pengolahan data pada proses ekstraksi fitur dikarenakan pada tahapan ini fitur-fitur tersebut dipergunakan sebagai *input* dalam algoritma analisis sentimen.

d. Normalization

Proses *Normalization* berfungsi sebagai pengubah bentuk suatu kata yang tidak baku menjadi bentuk bakunya. Kata-kata yang biasa diubah dalam proses ini adalah bahasa sehari-hari yang pada umumnya menggunakan bahasa tidak baku dan singkatan-singkatan tertentu. Kegunaan proses *normalization* dalam penelitian ini adalah untuk melakukan generalisasi terhadap ekspresi sentimen yang ada pada setiap data *tweet*.

Ketika data sudah di-normalisasi, maka model analisis sentimen akan lebih mudah dalam mengenali pola-pola yang ada pada data dengan efektif. Kamus yang digunakan dalam proses ini adalah Kamus Besar Bahasa Alay (KBBA) yang dibuat oleh Rama Prakoso. Kamus ini sendiri berbentuk file .txt yang kemudian diubah oleh penulis secara manual menjadi bentuk file .csv.

File ini berisikan kata tidak baku dan kata baku yang dipisahkan dengan tanda koma (Prakoso, 2017). Gambar 3.3 menunjukkan isi dari kamus ini ketika file ini masih berbentuk .txt dalam sumber aslinya. Gambar 3.4 menunjukkan kamus ini telah dikonversikan ke dalam bentuk .csv.



Gambar 3. 3 Isi Kamus KBBA dalam Bentuk File Txt.

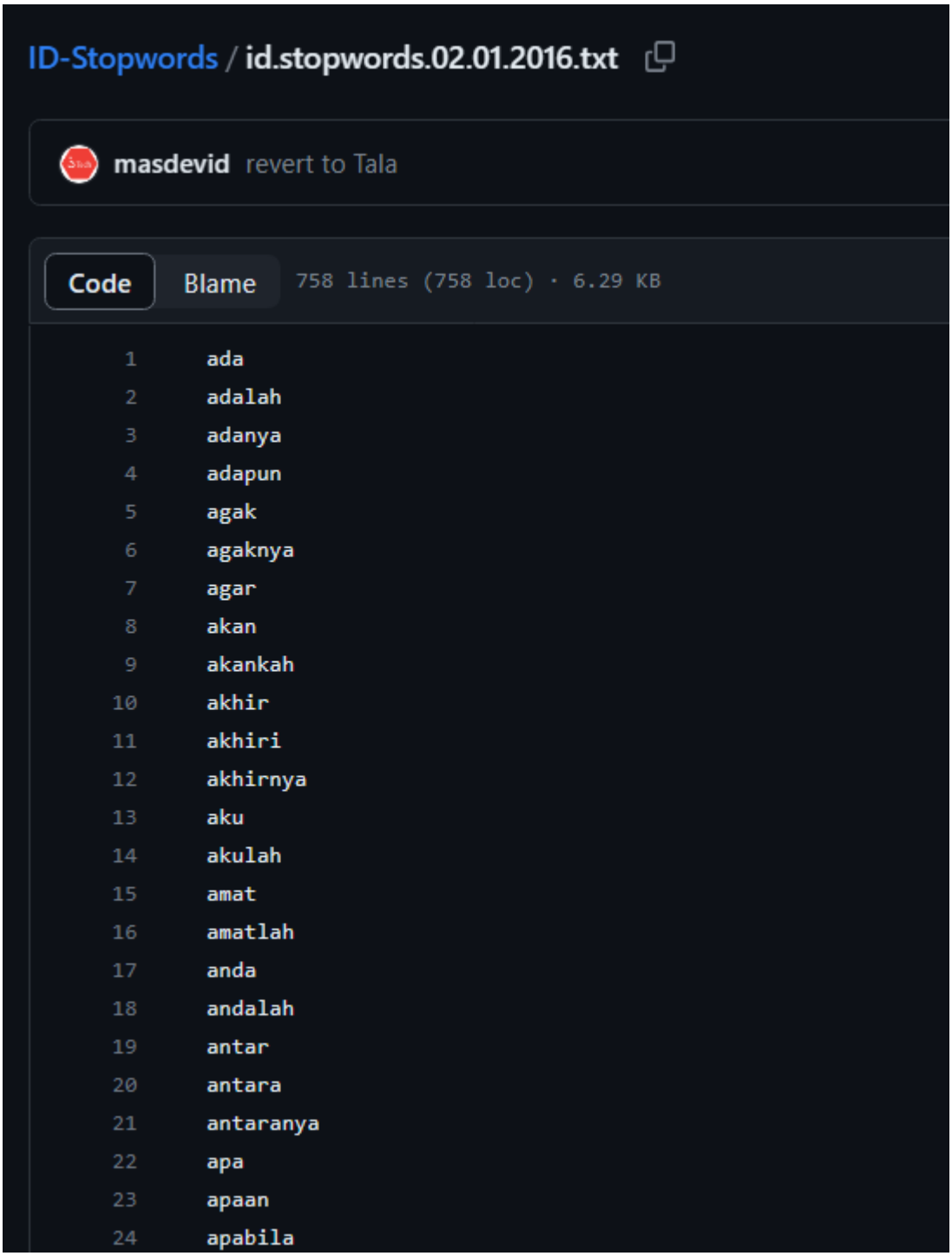
1	@	di
2	\n	dan
3	5f	maaf
4	7an	tujuan
5	ababil	abg labil
6	abis	habis
7	acc	diterima
8	ad	ada
9	adaa	ada
10	adlah	adalah
11	adlh	adalah
12	adoh	aduh
13	afaik	as far as i know
14	aga	apa
15	agr	agar
16	aha	tertawa
17	ahaha	haha
18	ahir	akhir
19	aing	saya
20	aj	aja
21	aj	saja
22	aja	saja

Gambar 3. 4 Kamus KBBA dalam Bentuk File Csv.

e. Remove Stopwords

Proses *Remove Stopwords* berfungsi untuk melakukan penghapusan kata-kata yang tidak memiliki arti atau yang kurang bermakna dalam dokumen. Salah satu tuuan dari penerapan tahap *preprocessing* ini adalah untuk mengurangi *noise* pada data. Penggunaan kata-kata pada *stopwords* seperti "yang, di, ke" dalam kehidupan manusia sehari-harinya cukup lazim. Namun, pada algoritma komputer kata-kata ini tidak berkontribusi terhadap sentimen yang muncul dalam data *tweet* yang diolah. Pada proses *Remove Stopwords* ini, digunakan kamus bernama ID-Stopwords yang dibuat oleh user github bernama masdevid. Terdapat 758 kata yang terkandung sebagai *stopwords* pada kamus ini.

File kamus ini berisikan kata yang tidak memiliki arti atau kata yang kurang memiliki makna dalam dokumen (Tala, 2003). Alamat Github kamus ini dapat dilihat pada tautan berikut (GitHub - masdevid/ID-Stopwords: Stopwords collection of Bahasa Indonesia collected from many sources.) Gambar 3. 5 menunujukkan isi dari kamus *stopwords* ini ketika file ini berada dalam sumber aslinya. Gambar 3. 6 menunjukkan isi dari kamus yang telah dikonversikan ke dalam bentuk .csv.



Gambar 3. 5 Isi dari Kamus ID-Stopwords dalam Bentuk File Txt.

	A
1	ada
2	adalah
3	adanya
4	adapun
5	agak
6	agaknya
7	agar
8	akan
9	akankah
10	akhir
11	akhiri
12	akhirnya
13	aku
14	akulah
15	amat
16	amatlah
17	anda
18	andalah
19	antar
20	antara
21	antaranya
22	apa
23	apaan
24	apabila
25	apakah
26	apalagi
27	apatah
28	artinya

Gambar 3. 6 Isi dari Kamus ID-Stopwords dalam Bentuk File Csv.

f. Stemming

Stemming adalah tahapan dalam *Preprocessing* yang mengubah setiap kata dari hasil proses *Remove Stopwords* yang terdapat imbuhan di dalamnya menjadi kata dasar. Salah satu alasan dipergunakannya *stemming* dalam penelitian ini yakni kemampuan proses ini untuk dapat meningkatkan performa model dan tidak membutuhkan banyak *resource* untuk berjalan dalam sistem. Metode yang dipergunakan oleh tahap *stemming* untuk memperingan kerja sistem adalah dengan mengurangi jumlah kata unik yang harus dikandung oleh sistem (Kowalski, 2011).

3.5 Ekstraksi Fitur

Setelah proses *Preprocessing* terhadap data selesai dilakukan, tahapan selanjutnya adalah dilakukannya ekstraksi fitur. Pada langkah ini, terdapat dua tahapan dalam prosesnya yaitu pembuatan *word vector* dan pembobotan terhadap setiap kata pada setiap kalimat atau dokumen dengan memanfaatkan *TF-IDF* yang dirumuskan dalam Persamaan (2.2). Setelah proses ekstraksi fitur selesai dilaksanakan, *dataset* siap untuk dipergunakan dalam proses selanjutnya yaitu *Training Model*.

3.6 Imbalanced Data Handling

Tahapan ini dijalankan untuk melakukan penanganan terhadap jumlah dari kelas *dataset* yang tidak seimbang sehingga nantinya akan mempengaruhi kinerja model dan berpotensi untuk memberikan hasil yang tidak maksimal dalam melakukan prediksi terhadap data yang di-inputkan. Metode *Imbalanced Data Handling* yang dipergunakan dalam penelitian ini adalah *Synthetic Minority Over-sampling Technique* (SMOTE).

3.7 Training Model

Pada proses *Training Model* ini, dipergunakan metode klasifikasi yang memanfaatkan algoritma *naïve bayes* dan *Support Vector Machine*. Data yang dipergunakan dalam proses klasifikasi sentimen ini berasal dari hasil pemrosesan data di tahap *Data Preprocessing* dan ekstraksi fitur yang menggunakan *TF-IDF* dalam proses pembobotan katanya. Output yang dihasilkan dari proses ini ialah sebuah model *Machine Learning* yang dapat melakukan pengelompokkan sentimen terhadap data *tweet.* Kumpulan data *tweet* ini dimasukkan sebagai input dan akan diklasifikasikan berdasarkan tiga sentimen kelas, yaitu sentimen positif, negatif, dan netral.

Dilakukan pembagian pada data *tweet* yang digunakan pada proses ini, yaitu sebesar 80% data digunakan sebagai data *training* dan 20% data digunakan sebagai data *testing*. Rasio pembagian kedua jenis data ini dirujuk dari penelitian-penelitian sebelumnya yang juga menggunakan pembagian 80:20 untuk data *training* dan data *testing*-nya. Beberapa penelitian tersebut antara lain (Febriyanti, 2020), yang menggunakan rasio pembagian 80:20 di antara ke-5 pembagian data latih dan data uji yang dilakukan. Adapun rasio pembagian data selain 80:20 yang dilakukan pada penelitian ini yaitu 60:40, 70:30, 85:15, dan 90:10. Dalam (Ratnawati, 2018), pembagian data dengan rasio 80-20 dilakukan dalam metode pengujian *K-Fold Cross Validation* yang dilakukan terhadap 500 *tweet* termasuk dengan data uji. Penelitian yang

dilakukan oleh (Mulia, 2021) memisahkan data *training* dan *testing* dengan rasio 80:20 dari total 5055 data dalam tahapan evaluasi model *Naïve Bayes Classifier*.

3.8 Evaluasi Model

Dalam proses ini, perhitungan dilakukan pada model yang telah dibuat pada tahapan sebelumnya untuk dapat dilakukan penilaian terhadapnya. Terdapat lima aspek perhitungan yang dijadikan sebagai objek penilaian, yaitu *Accuracy, Precision, Recall, F1-score,* dan *Confusion Matrix*. Aspek yang pertama yaitu *Accuracy*, merupakan perhitungan yang mengukur ketepatan dalam melakukan klasifikasi terhadap data. Aspek selanjutnya yaitu *Precision*, yang mana pengukuran dilakukan terhadap tingkat kepastian dalam melakukan klasifikasi pada data. Aspek selanjutnya adalah *Recall*, yang menghitung rasio kelas prediksi dari tiap-tiap label yang mengklasifikasikan suatu data itu benar terhadap data yang salah. Spek selanjutnya yaitu *F1-Score* merupakan suatu parameter perhitungan dalam proses evaluasi model yang mendapatkan hasilnya dari proses penghitungan nilai rataan harmonik yang berasal dari dua ukuran ketepatan yaitu *recall* dan *precision*. Aspek perhitungan yang terakhir adalah *Confusion Matrix*, yaitu sebuah tabel yang berisi tentang pengelompokan jumlah data uji yang benar serta yang salah.

3.9 Penyimpanan Model

Setelah dilakukannya pelatihan dan evaluasi terhadap model, prosedur yang dilakukan selanjutnya adalah menyimpan model yang sebelumnya telah dilatih dan dievaluasi dengan maksud agar tidak perlu lagi melakukan pelatihan ulang pada model. Modul bawaan *Python* yang dimanfaatkan dalam proses penyimpanan model ini bernama *pickle*.

BAB IV
HASIL DAN PEMBAHASAN

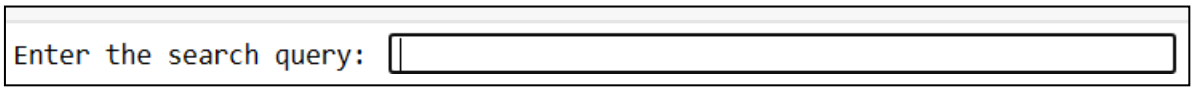
4.1 Pengumpulan Data

Data tweet yang dikumpulkan untuk dapat dipergunakan pada penelitian ini diambil dengan cara melakukan *Crawling* yang memanfaatkan bantuan kode pemrograman Python dan *Library* bernama tweepy. Output dari proses *Crawling* ini adalah file .json yang nantinya akan dilakukan pengkonversian menjadi bentuk file .csv dengan tujuan untuk membuat file hasil dari proses *crawling* ini dapat dijadikan kembali ke dalam bentuk excel (.xlsx) untuk selanjutnya dilakukan pelabelan terhadap data hasil *crawling*. Kode pemrograman untuk melakukan proses ini ditampilkan pada Gambar 4.1 dan cara kerja dari kode ini ditunjukkan secara berurutan mulai dari Gambar 4.2, 4.3, 4.4, 4.5, 4.6, dan 4.7.

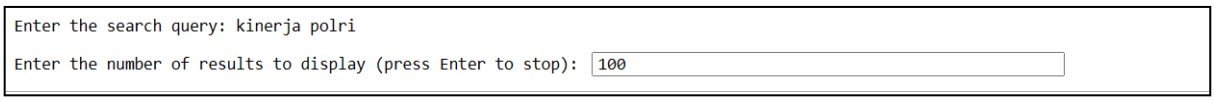
```
1. import tweepy, json, sys, os
2. consumer_key = "Masukkan API Key"
3. consumer_secret = " Masukkan API Secret Key "
4. access_token = "Masukkan Access Token Key"
5. access_token_secret = " Masukkan Access Token Secret Key"
6. auth = tweepy.OAuthHandler(consumer_key, consumer_secret,
   access_token, access_token_secret)
7. auth.set_access_token(access_token,   access_token_secret)
8. api = tweepy.API(auth, wait_on_rate_limit=True)
9. current_dir = os.getcwd()
10. file_path  =  os.path.join(current_dir)
11. query = input('Enter the search query: ')
12. while True:
13.count = input('Enter the number of results to display (press
   Enter to stop): ')
14.         if count == '':
15.             break
16.     try:
17.         count = int(count)
18.         break
19.     except ValueError:
20.         print('Invalid input. Please enter a valid number.')
21.output_file_name = input('Enter the output file name (without
   extension): ')
22.tweets  =  tweepy.Cursor(api.search_tweets,  q=query,  lang='id',
   tweet_mode='extended').items(count)
23.     print('Searching  for  tweets...\n')
24.     with open(output_file_name + '.json', 'w') as f:
```

```
        for i, tweet in enumerate(tweets):
            try:
                json.dump(tweet._json,  f)
                f.write('\n')
                sys.stdout.write('\rRetrieved tweets: ' + str(i+1)
    + '/' + str(count))
                sys.stdout.flush()
                if i+1 == count:
                    break
        except tweepy.TweepError as e:
            print('Encountered an error:', e)
            break
print('\n\nDisplaying ' + str(i+1) + ' results.')
print(f"\nDone, tweet scraping process is done and the file is stored
    in {os.path.abspath(file_path)}")
```

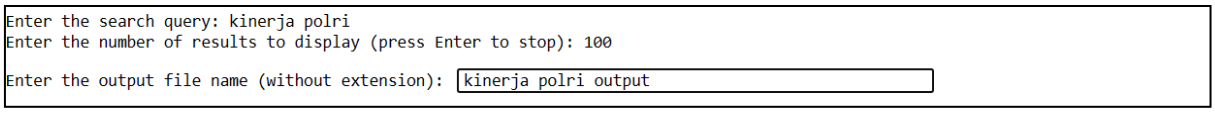
Gambar 4. 1 Kode Pemrograman Crawling Data



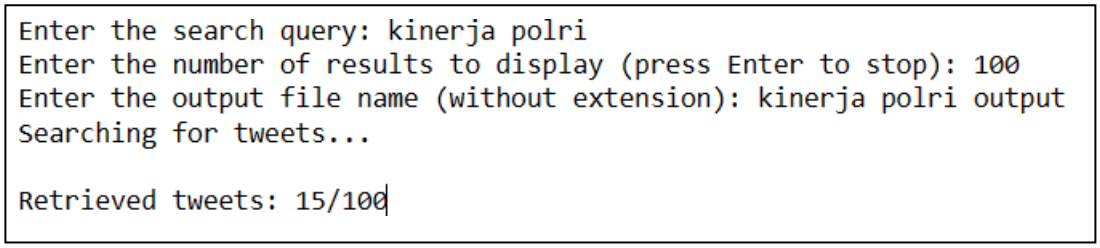
Gambar 4. 2 Input Kueri Pencarian Data Tweet



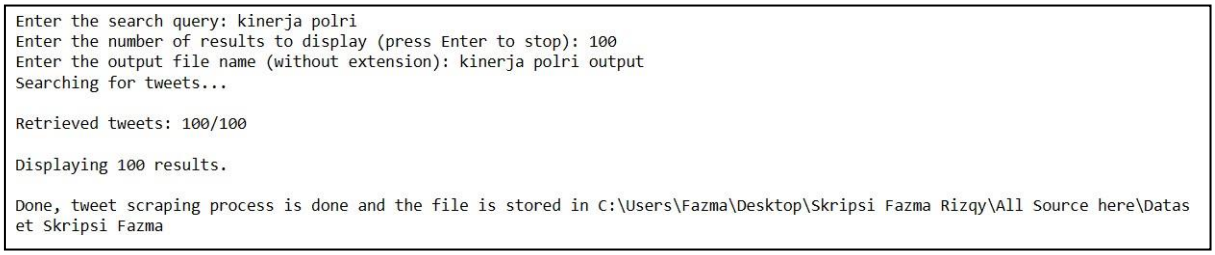
Gambar 4. 3 Input Jumlah Tweet yang Dibutuhkan



Gambar 4. 4 Input Nama File Output



Gambar 4. 5 Proses Crawling Tweet



Gambar 4. 6 Data Tweet Berhasil Dikumpulkan dalam Bentuk File .JSON

Gambar 4. 7 Isi File .JSON yang Berisikan Tweet Hasil Crawling

File .json ini berisikan tentang parameter yang terkandung pada setiap baris data hasil dari proses *crawling*. Beberapa parameter tersebut dijabarkan sebagai berikut, "created_at" berisikan mengenai stempel waktu yang menandakan kapan *tweet* tersebut dibuat. "id" berisikan nomor identifikasi unik yang terdapat pada *tweet.*

"id_str" *string* disini merepresentasikan nomor identifikasi unik yang dimiliki *tweet.* "full_text" berisikan teks komplit yang terdapat pada *tweet.* "truncated" berisikan sebuah *boolean* yang mengindikasikan bahwa suatu data *tweet* dipotong untuk mempersingkat

panjangnya. “display_text_range” berisikan daftar yang mengindikasikan titik awal dan akhir sebuah indeks pada teks yang ditampilkan dalam *tweet* orisinilnya.

“entitites” berisikan *metadata* yang berhubungan dengan entitas yang ada dalam *tweet*, contohnya adalah *hashtag, user mention*, dan URL. “hashtags” berisikan tentang *hashtag* apa saja yang ada dalam *tweet*. “symbols” berisikan mengenai simbol-simbol yang ada dalam *tweet*.

Langkah pertama yang dilakukan pada saat menjalankan kode ini adalah melakukan import *library* yang diperlukan untuk melakukan *crawling* terhadap data pada platform Twitter, yaitu tweepy, json, sys, dan os. Setelah *library* di-*import*, selanjutnya adalah melakukan *setting* terhap kredensial otentikasi yang dibutuhkan untuk mengakses API Twitter. Kode yang berisikan consumer_key, consumer_secret, access_token, dan access_token_secret didapatkan melalui pembuatan akun developer lewat platform Twitter. Setelah itu, pada baris kode ke-6 dan ke-7 dibuat sebuah objek bernama OAuthHandler dengan kredensial otentikasi yang telah diberikan sebelumnya dan juga menetapkan *access token* untuk objek OAuthHandler tersebut.

Pada baris kode ke-8, dibuat sebuah objek API dengan memanfaatkan *authentication handler* dan menetapkan nilai parameter wait_on_rate_limit menjadi True agar membuat skrip kode melakukan proses tunggu ketika API sudah mencapai *rate limit*. Dilakukan pengambilan direktori file di mana kode disimpan sekarang menggunakan fungsi os.getcwd() pada baris ke-9 dan dibuat pula jalur file penyimpanan dengan menyatukan direktori yang dipergunakan sekarang pada baris ke-10.

Setelah API siap, maka pada baris kode ke-11 dilakukan *prompt* untuk pengguna agar memasukkan kata kunci *tweet* yang ingin dicari. Baris selanjutnya yaitu baris 12 hingga 20 berfungsi untuk menangani input yang dimasukkan oleh user, memastikan bahwa angka jumlah *tweet* yang ingin dicari merupakan jenis input angka yang valid.

Pada baris ke-21, kode memberikan *prompt* kepada user untuk memasukkan nama file output tanpa menambahkan ekstensi filenya, dikarenakan pada kode sudah dilakukan *set* ke dalam bentuk file .json. Baris selanjutnya yaitu 22 memiliki fungsi untuk mencari data *tweet* dengan mempergunakan *cursor object* yang telah disediakan oleh Tweepy, di mana dilakukan spesifikasi terhadap kueri pencarian yang menetapkan bahwa *tweet* yang diambil hanya yang berbahasa Indonesia (id) dan tweet_mode menjadi ‘*extended’* agar *tweet* yang didapatkan berbentuk penuh.

Baris kode ke-23 mencetak teks yang mendandakan bahwa proses pencarian sedang dilakukan. Baris ke-24 hingga 35 melakukan pengulangan terhadap *tweet* yang diterima, dan

menyimpan data *tweet* dalam bentuk .json dan memperlihatkan perkembangan proses *Crawling* menggunakan fungsi sys.stdout.write().

Baris kode ke-36 mencetak angka dari hasil *tweet* yang telah didapatkan. Dan baris terakhir yaitu 37 menampilkan sebuah pesan yang menandakan bahwa proses *Crawling tweet* sudah selesai dilakukan, lalu program menunjukkan lokasi direktori file hasil *Crawling* yang berbentuk .json disimpan.

Adapun dalam pengambilannya terdapat beberapa tahapan sebelum data yang diambil ini berhasil diolah menjadi *dataset* yang berguna untuk melakukan pengujian dalam penelitian ini. Data yang diekstraksi dari media sosial Twitter terdiri dari hasil proses *Crawling* data sebanyak dua kali. Proses yang pertama dimulai dari tanggal 28 September hingga 6 Oktober 2022. Proses yang kedua dimulai dari tanggal 11 hingga 20 Desember 2022.

Proses seleksi data dilakukan dengan memberikan batasan yaitu hanya *tweet* yang mengandung kata kunci terkait dengan "Kinerja POLRI" dan berbahasa Indonesia yang dipergunakan. Dikarenakan proses *Crawling* yang dilakukan sebanyak 2 kali menghasilkan 2 buah file .json yang terpisah, dilakukan penggabungan terhadap 2 file tersebut ke dalam bentuk .xlsx agar pemrosesan *dataset* dapat lebih mudah dilakukan. Kode untuk melakukan penggabungan terhadap 2 file .json tersebut serta proses penyortiran kolom yang ada pada file .json menjadi hanya yang mengandung teks ditunjukkan pada Gambar 4.8.

```
1. #INPUT HASIL CRAWLING BULAN OKTOBER & DESEMBER
2. import pandas as pd
3. df_oktober = pd.read_json('berisi path beserta nama file .json
   hasil crawling di bulan oktober', lines=True)
4. df_desember = pd.read_json('berisi path beserta nama file .json
   hasil crawling di bulan desember', lines=True)
5. # GABUNG ISI DARI KEDUA FILE JSON
6. gabung_df = pd.concat([df_oktober, df_desember], axis=0)
7. #PILIH KOLOM YANG DIGUNAKAN
8. gabung_df  =  pd.DataFrame(gabung_df['text'])
9. #HAPUS DUPLIKAT
10.gabung_df.drop_duplicates(subset="text",  keep  =  'first',
   inplace = True)
11. #EXTRACT FILE JSON MENJADI BENTUK EXCEL
12.gabung_df.to_excel('Berisi path tujuan file output hasil
   penggabungan kedua hasil crawling bulan oktober dan desember')
```

Gambar 4. 8 Kode Pemrograman Penggabungan Dua Hasil Crawling.

Pertama-tama, dilakukan *import library* pandas untuk dapat mengubah *dataset* menjadi objek Python yang memiliki kolom dan baris atau biasanya dikenal dengan sebutan *DataFrame*. Pada baris ketiga dan keempat dimasukkan input file hasil *Crawling* yang masih berbentuk .json ke dalam dua *DataFrame* yang berbeda agar dapat dilakukan penggabungan isi dari kedua *dataset* tersebut.

Selanjutnya, pada baris keenam dilakukan penggabungan kedua dataframe dengan memanfaatkan fungsi pandas berupa concatenate yang menggabungkan keduanya dengan mengikuti sumbu tertentu yang dapat berupa baris atau kolom. *Value* yang dimiliki oleh parameter axis di sini adalah 0 karena penggabungan dilakukan berdasarkan baris data yang mana *output*-nya sendiri menghasilkan *dataset* yang berisi gabungan 2 file .json dalam 1 kolom. Jika Axis diubah menjadi 1, maka hasil outputnya menjadi 2 kolom yang berisi data dari df_oktober dan df_desember yang disajikan dalam bentuk 2 kolom yang terpisah. Gambar 4.9 menunjukkan contoh jika nilai "axis=1" dan Gambar 4.10 juga berisi contoh jika nilai "axis=0".

Pada baris kode kedelapan, dilakukan pemilihan kolom yang digunakan untuk pengolahan data. Di sini kolom yang diambil hanya kolom 'text' yang berisi data teks hasil *Crawling* dari *tweet* yang ada di *Twitter*. Pada baris kesepuluh dan baris keduabelas, dilakukan penghapusan data duplikat, dan selanjutnya data yang sebelumnya berbentuk .json diubah ke dalam bentuk *dataframe*, lalu diekstrak ke dalam bentuk excel. Tujuan dari disimpannya data ke dalam bentuk excel ialah untuk dapat mempermudah proses selanjutnya pada penelitian ini dan juga agar dapat dibaca oleh manusia dengan nyaman apabila data tersebut berbentuk excel. Data *tweet* yang awalnya berjumlah 3275 berhasil direduksi menjadi 2859 setelah melakukan proses *drop duplicates* ditunjukkan pada Gambar 4. 12 dan Gambar 4. 13.

GABUNG ISI DARI KEDUA FILE JSON

```
In [15]: gabung_df = pd.concat([df_oktober, df_desember], axis=1)
```

PILIH KOLOM YANG DIGUNAKAN

```
In [16]: gabung_df = pd.DataFrame(gabung_df['text'])
```

```
In [17]: gabung_df
```

Out[17]:

	text	text
0	@DivHumas_Polri Yahhh lombanya dihapus ges ter...	Selamat Petang Samosir ,\nApakah kamu merasa m...
1	Halo Sobat Polri..\nPendaftaran Lomba Menulis ...	Trending Indonesia - 20/12/2022 18:00 WIB\n1. ...
2	coba temanya diganti mas @DivHumas_Polri "kine...	Tren Topik Indonesia - 20 Desember 2022 18:00 ...
3	@Avatar_Rokupang @an__noying @kadeharakzuha @c...	Selamat Sore Sigi ,\nSaat anda jatuh di hari ...
4	RT @dphfaunderscore: @DivHumas_Polri ((perbaik...	Selamat Sore Pacitan ,\nOrang yang melakukan ...
...	...	...
1853	@siapkawalpolri Kinerja Polri sudah sangat bai...	NaN
1854	giat pelaksanaan Pembuatan Laporan hasil Tim ...	NaN
1855	@siapkawalpolri Nahh gini nihh apresiasi sekal...	NaN
1856	Wih mantep banget nih dengan kinerja dari para...	NaN
1857	Wah bangga lho gue melihat kinerja Polri sekar...	NaN

1858 rows × 2 columns

Gambar 4. 9 Contoh Nilai "axis=1" Dalam Penyajian 2 Kolom.

GABUNG ISI DARI KEDUA FILE JSON

```
In [8]: gabung_df = pd.concat([df_oktober, df_desember], axis=0)
```

PILIH KOLOM YANG DIGUNAKAN

```
In [9]: gabung_df = pd.DataFrame(gabung_df['text'])
```

```
In [10]: gabung_df
```

Out[10]:

	text
0	@DivHumas_Polri Yahhh lombanya dihapus ges ter...
1	Halo Sobat Polri..\nPendaftaran Lomba Menulis ...
2	coba temanya diganti mas @DivHumas_Polri "kine...
3	@Avatar_Rokupang @an__noying @kadeharakzuha @c...
4	RT @dphfaunderscore: @DivHumas_Polri ((perbaik...
...	...
1412	@aarifasophia Apresiasi untuk kinerja polri da...
1413	@johanhowan @ListyoSigitP @DivHumas_Polri @kom...
1414	@AfganHendrakus1 Semangat pak. Semakin banyak ...
1415	https://t.co/3KS1X6AX24
1416	Bhabinkamtibmas melakukan Sosialisasi kepada w...

3275 rows × 1 columns

Gambar 4. 10 Contoh Nilai "axis = 0" Dalam Penyajian 2 Kolom.

HAPUS DUPLIKAT

```
In [6]: gabung_df.drop_duplicates(subset="text", keep = 'first', inplace = True)
```

EXTRACT FILE MENJADI BENTUK EXCEL

```
In [7]: gabung_df.to_excel('Direktori File dan nama file.xlsx')
```

Gambar 4. 11 Penghapusan Data Duplikat dan Ekstraksi File ke Excel.

	text
0	@DivHumas_Polri Yahhh lombanya dihapus ges ter...
1	Halo Sobat Polri..\nPendaftaran Lomba Menulis ...
2	coba temanya diganti mas @DivHumas_Polri "kine...
3	@Avatar_Rokupang @an__noying @kadeharakzuha @c...
4	RT @dphfaunderscore: @DivHumas_Polri ((perbaik...
...	...
1412	@aarifasophia Apresiasi untuk kinerja polri da...
1413	@johanhowan @ListyoSigitP @DivHumas_Polri @kom...
1414	@AfganHendrakus1 Semangat pak. Semakin banyak ...
1415	https://t.co/3KS1X6AX24
1416	Bhabinkamtibmas melakukan Sosialisasi kepada w...

3275 rows × 1 columns

Gambar 4. 12 Hasil Penggabungan Data Hasil Crawling Sebelum Drop Duplikasi

	text
0	@DivHumas_Polri Yahhh lombanya dihapus ges ter...
1	Halo Sobat Polri..\nPendaftaran Lomba Menulis ...
2	coba temanya diganti mas @DivHumas_Polri "kine...
3	@Avatar_Rokupang @an__noying @kadeharakzuha @c...
4	RT @dphfaunderscore: @DivHumas_Polri ((perbaik...
...	...
1412	@aarifasophia Apresiasi untuk kinerja polri da...
1413	@johanhowan @ListyoSigitP @DivHumas_Polri @kom...
1414	@AfganHendrakus1 Semangat pak. Semakin banyak ...
1415	https://t.co/3KS1X6AX24
1416	Bhabinkamtibmas melakukan Sosialisasi kepada w...

2859 rows × 1 columns

Gambar 4. 13 Hasil Penggabungan Data Hasil Crawling Setelah Drop Duplikasi

1	
2	@DivHumas_Polri Yahhh lombanya dihapus ges ternyata huhu sedih padahal
3	Halo Sobat Polri..Pendaftaran Lomba Menulis Artikel Berita Polri 2022 telah di
4	coba temanya diganti mas @DivHumas_Polri "kinerja polisi" sub tema "polisi
5	@Avatar_Rokupang @an__noying @kadeharakzuha @collegemenfess Kalau c
6	RT @dphfaunderscore: @DivHumas_Polri ((perbaiki citra polri)) LUCU, YANG E
2812	@aarifasophia Apresiasi untuk kinerja polri dalam mengungkap ini. Tuntaska
2813	@johanhowan @ListyoSigitP @DivHumas_Polri @kompolnas_ri @mohmahfu
2814	@AfganHendrakus1 Semangat pak. Semakin banyak kasus yg keungkap, sem
2815	Bhabinkamtibmas melakukan Sosialisasi kepada warga tentang *Kartu Barco

Gambar 4. 14 Data Hasil Crawling yang Sudah Disimpan Ke Dalam Bentuk Excel

Jumlah *row* dalam Excel pada Gambar 4.13 adalah 2860, sedangkan jumlah data setelah diduplikasi adalah 2859. Hal ini terjadi dikarenakan baris pertama pada file excel ini berisikan *heading* atau judul kolom, sehingga penghitungan jumlah data dimulai dari baris ke-2. Setelah melakukan konversi bentuk *dataframe* ke dalam bentuk Excel, maka data sudah siap untuk dilabeli secara manual terlebih dahulu sebelum melakukan proses *pre-processing*. Tahapan ini membutuhkan pengamatan langsung dari manusia agar label yang diberikan tepat terhadap data teks tersebut.

4.2 Hasil Implementasi Pelabelan Data

Setelah proses pengambilan dan pengumpulan data selesai dilakukan, selanjutnya dilakukan proses *Labelling*. Proses ini dijalankan secara manual terhadap 2859 *tweet* yang akan membagi data menjadi tiga kelas label, yaitu positif, negatif, dan netral. Ketika proses pelabelan dilakukan secara manual, penulis menemukan data *tweet* yang tidak relevan dengan topik utama yaitu bahasan mengenai analisis sentimen terhadap kinerja POLRI.

Selanjutnya penulis melakukan penghapusan *tweet* secara manual sebanyak 45 *tweet*. Sehingga, jumlah data yang awalnya adalah 2859 *tweet* berkurang menjadi 2814. Pada Tabel 4.1 disajikan data *tweet* yang dihapus karena tidak relevan dengan bahasan utama.

Tabel 4. 1 Tweet yang Dihapus Karena Tidak Relevan

No.	*Tweet* yang Dihapus
1.	https://t.co/X5tZ17gNw8
2.	itu kliatany yg rusuh kebanyakan bocah2 kemarin sore, pd sok jagoan.yg gak ikut rusuh ikut jadi korban. jadi poli… https://t.co/WlG1KFRpar

3.	@CNNIndonesia Sesama Baswedan memang wajib saling bela tapi sebagai ASN Polri tak elok komentari kinerja KPK pel @nazaqistsha
4.	@RadioElshinta Coba pak jokowi iseng gtu ya tiba2 story survei gtu di ig, tanpa bilg siapa2 langsung post aja.
5.	Kinerja Polri https://t.co/fZz3Hy9N04

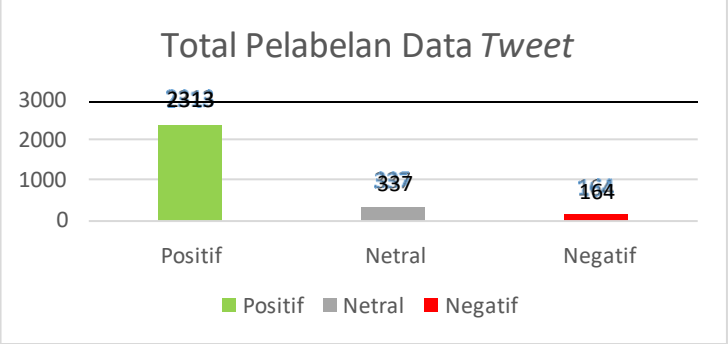
Dari proses *Labeling* yang dilakukan terhadap 2814 data *tweet*, dirincikan bahwa terdapat 2313 *tweet* yang dilabeli dengan sentimen positif, 337 *tweet* dilabeli sebagai *tweet* yang memiliki sentimen netral, dan 164 *tweet* diberi label sebagai negatif. Penggunaan Angka {1,2,3} pada proses *labeling* merepresentasikan kelas sentimen dari masing-masing data. Angka 1 merepresentasikan sentimen positif, Angka 2 netral, dan Angka 3 negatif.

Tabel 4.2 menunjukkan beberapa contoh *tweet* dalam proses *Labeling* yang telah dilakukan. Dalam Gambar 4.15 terdapat representasi bentuk chart dari pembagian *tweet* ke dalam kelas sentimen setelah dilakukan pelabelan. Setelah dilabeli, kemudian file yang sebelumnya merupakan gabungan hasil proses *crawling* disimpan kembali ke dalam bentuk excel. File ini kemudian digunakan dalam tahapan *preprocessing*.

Tabel 4. 2 Contoh Tweet dalam Proses Labeling

Tweet	**Label**
Selalu dukung bapak2 polri, semangat ✊ ✊ ✊ maju terus!! Berantas pelaku kejahatan ! Polri kinerja maksimal	Positif
🥰 💪 Mantap dan solid, Polri Kinerja Maksimal bersama TNI semakin menunjukkan kinerja yang bagus untuk negara tercinta https://t.co/GA8E57azql	Positif
Refleksi Kinerja Reformasi Polri https://t.co/YuBT8oMbe3	Netral
@YASOzatulo @_MbakSri_ @DivHumas_Polri LsM PENJARA (Pemantau Kinerja Aparatur Negara) itu yg bs mengatasi nya.	Netral

@detikcom Ini menandakan kinerja polri tidak bisa di andalkan 🙏 Kalo masyarakat yg berisanksi, trus fungsi polisi apa pak? @PolhukamRI	Negatif
Anggaran Pencitraan Besar, Kinerja 'Tak Maksimal' https://t.co/rclS2phH0P	Negatif



Gambar 4. 15 Distribusi Kelas Data Tweet

4.3 Hasil Implementasi *Data Preprocessing*

Karakteristik dari data *tweet* yang telah didapatkan dari proses sebelumnya yaitu *Crawling* memiliki sifat tidak terstruktur, oleh karena itu pada tahapan ini dilakukan *Preprocessing* untuk mempersiapkan data agar dapat diolah dengan baik pada proses analisis sentimen. Terdapat *Library* yang diharuskan untuk dipersiapkan terlebih dahulu sebelum digunakan dalam proses *Data Preprocessing*. Kode program untuk dapat memulai proses Import *Library* serta mengunduh file penyokong pada proses *Preprocessing* ditunjukkan dalam Gambar 4.16.

```
1.      import pandas as pd
2.      import re
3.      import numpy as np
4.      import string
5.      import Sastrawi
```

Gambar 4. 16 Import *Library* untuk Preprocessing

Library yang dimanfaatkan dalam tahapan *preprocessing* di antaranya adalah Pandas, Regular Expression atau yang disingkat sebagai re dalam Gambar 4.13, NumPy, *String*, dan Sastrawi. Kelima *Library* ini memiliki fungsi yang berbeda dalam pemanfaatannya:

1. Pandas

   Library pertama yang digunakan dalam tahapan *preprocessing* adalah pandas. *Library* ini menyediakan struktur data dan fungsi yang dapat menangani data yang terstruktur dengan efisien, khususnya data *tabular* dalam bentuk *DataFrame*. Pandas memungkinkan pengguna untuk menjalankan *task* seperti *sorting, filtering, merging,* dan *aggregating* data secara mudah. Pandas sendiri menggabungkan metode komputasi berperforma tinggi yang dimiliki oleh NumPy dengan kapabilitas manipulasi data yang fleksibel dari *spreadsheets* dan database relasional (Mckinney, 2018).

2. *Regular Expression* (RE)

   Regular Expression (RE) merupakan sebuah abstraksi dari pencarian *keyword* yang memungkinkan identifikasi terhadap teks dilakukan dengan memanfaatkan pola dibandingkan dengan *string* yang tepat itu sendiri. *Library* ini memungkinkan penggunanya untuk mengekstrak, mencari, dan memanipulasi teks berdasarkan pola-pola atau aturan yang spesifik(Chapman & Stolee, 2016).

3. NumPy

   NumPy merupakan sebuah *library* Python *open-source* yang dikembangkan oleh komunitas, yang menyediakan objek *array* Python multidimensional dengan fungsi *array-aware* yang beroperasi di atasnya. Sifat simpel yang melekat terhadap NumPy ini menjadikan *array* NumPy merupakan sebuah format pertukaran *de facto* untuk data *array* di Python. *Library* ini menyediakan objek *array N-dimensional* yang kuat dan jangkauan fungsi yang luas untuk melakukan operasi matematika pada *array* dengan efisien. NumPy beroperasi pada *array in-memory* menggunakan *Central Processing Unit* (CPU) (Harris et al., 2020)

4. *String*

   String merupakan sebuah modul bawaan yang dimiliki oleh Python. Modul ini berisi beberapa fungsi untuk memproses *string* standar Python. Fungsi yang disediakan oleh modul ini adalah manipulasi serta transformasi *string* dasar.

5. Sastrawi

Sastrawi merupakan sebuah *library stemmer* untuk pemrosesan bahasa natural (NLP) dalam Bahasa Indonesia. Sastrawi dimanfaatkan untuk mengatasi masalah dalam mengganti antara kata dengan kata ke dalam kata dasar. *Library* ini menyediakan fungsi yang cakupannya luas, termasuk *stemming*, *stopword removal,* dan *task* spesifik NLP yang lainnya (Rosid et al., 2020).

Tahapan *Data Preprocessing tweet* hasil *Crawling* dilakukan dengan memanfaatkan bahasa pemrograman yaitu Python. Alur proses pada tahapan ini antara lain adalah *Regular Expression, Case Folding, Tokenization, Normalization, Remove Stopwords,* dan *Stemming*. Berikut ditunjukkan implementasi tahapan-tahapan tersebut dalam *Data Preprocessing* yang dilakukan:

1. Regular Expression

Regular Expression digunakan sebagai proses dalam tahapan *preprocessing* yang pertama dikarenakan data yang terdapat pada *tweet* memiliki bermacam karakter di luar dari alphabet ‘a’ sampai dengan ‘z’ yang dinilai tidak memberikan pengaruh terhadap nilai sentimen suatu data *tweet* dan juga bertujuan untuk dapat mengurangi beban dalam proses yang dilakukan.

Sebelum melakukan tahapan ini, kembali dilakukan *loading data* terhadap file hasil proses *labeling* terlebih dahulu. File yang digunakan untuk tahapan preprocessing berbeda dengan hasil penggabungan 2 file .json hasil *crawling* yang telah didapatkan pada proses dalam Gambar 4.6. Isi dari file yang akan dimasukkan ke dalam variabel tweet_polridf ini adalah data *tweet* yang telah dilakukan pelabelan pada masing-masing datanya secara manual. Berikut ditunjukkan proses *load data* pada Gambar 4.17

```
1.      def load_data():
2.      data = pd.read_excel('Gabungan Dataset Lengkap Hasil
Labeling.xlsx)#ubah    nama file sesuai dengan nama file kalian
3.          return data
4.      tweet_polridf = load_data()
5.      tweet  =  pd.DataFrame(tweet_polridf[['text']])
```

Gambar 4. 17 Kode Pemrograman Loading Data

Pada proses *loading data* dalam tahapan *data preprocessing*, digunakan file excel yang telah dilabeli secara manual pada tahapan 4.2 sebelumnya. File ini bernama Gabungan *Dataset* Lengkap Hasil Labeling.xlsx. Isi dari file ini adalah data *tweet* yang belum diproses dan label yang disematkan pada masing-masing *tweet*. Label pada masing-masing

tweet merepresentasikan kelas sentimen data *tweet* tersebut. Label 1 merepresentasikan sentimen positif, Label 2 merepresentasikan sentimen netral, dan Label 3 merepresentasikan sentimen negataif. Isi dari file ini ditunjukkan pada Gambar 4. 18.

	text	Label
2	@DivHumas_Polri Yahhh lombanya dihapus ges ternyata huhu sedih padahal udah nyiapin pujian buat kinerja polri	1
3	Halo Sobat Polri..Pendaftaran Lomba Menulis Artikel Berita Polri 2022 telah dibuka looh.. Lomba ini diselenggaraka... https://t.co/HLnjH9GTaX	2
4	coba temanya diganti mas @DivHumas_Polri "kinerja polisi" sub tema "polisi yang bengis, dan intimidatif" https://t.co/AEJL5UWe4m	3
5	@Avatar_Rokupang @an__noying @kadeharakzuha @collegemenfess Kalau d pikir2 6 bulan itu dapat apa??? naikkan lah mas... https://t.co/bPw22ZmlMe	3
6	RT @dphfaunderscore: @DivHumas_Polri ((perbaiki citra polri)) LUCU, YANG BENAR ITU PERBAIKI KINERJA DAN OTOMATIS CITRANYA BAIK. BUKAN MALAH...	3
7	RT @el_camarino: @DivHumas_Polri "Lapor pak, media memberitakan kinerja kepolisian semakin buruk, apakah kita harus mengevaluasi diri dan m...	3
8	RT @lita_hartoyo: @DivHumas_Polri Apresiasi kinerja 😍😍😍Humanis, berintegritas dan inspiratif 😘😘😘 https://t.co/LvegFUSZlF	1
9	@vivitwij Apresiasi kinerja polri..	1
10	Pendaftaran Lomba Menulis Artikel Berita Polri 2022 telah dibuka.Lomba ini diselenggarakan untuk para Jurnalis/ War... https://t.co/l8G6eC8EjB	1

Gambar 4. 18 Isi dari file Gabungan *Dataset* Lengkap Hasil Labeling.xlsx

Pada baris kode ke-1, fungsi load_data() didefinisikan. Baris kode ke-2 berisikan sebuah fungsi bernama pd.read_excel() yang berada di dalam fungsi load_data(). Fungsi pd.read_excel yang berasal dari *library* pandas digunakan untuk membaca sebuah file excel yang bernama 'Nama File hasil Labeling.xlsx', yang mana file ini berisikan data *tweet* yang telah dilabeli. Baris ke-3 berisi variabel data yang dimiliki oleh *DataFrame* yang kemudian di-*return* oleh fungsi load_data().

Diluar fungsi yang telah ada, pada baris ke-4 fungsi load_data() dipanggil dan nilai *return*-nya disematkan pada sebuah variabel yang bernama tweet_polridf. Baris ke-5 berisikan pembuatan sebuah objek *DataFrame* baru yang disebut tweet dengan cara memilih hanya kolom 'text' dari *DataFrame* tweet_polridf menggunakan dua tanda kurung siku ([['text']]).

Setelah data dimuat, diterapkan *preprocessing* pada tahap pertama yaitu *Regular Expression*. Contoh dari penerapan *Regular Expression* pada tahap *preprocessing* ditunjukkan pada kode pemrograman dalam Gambar 4.19 dan Tabel 4.3 sebagai perbandingan data sebelum dan sesudah dilakukan penerapan *Regular Expression*

```
1. def text_clean(tweet, pattern):
2.     r = re.findall(pattern, tweet)
3.     for i in r:
4.         tweet = re.sub(i, '', tweet)
5.
6.     tweet = re.sub(r'[0-
   9]+|^RT[\s]+|<.*?>|#+|\u2026|\u0022|\n|\u00B2|\u00B3|\u00B9|
   \u2070|\u2074-\u2079|:|-|/|,|@[^\s]+|\b[a-zA-
   Z]\b|”|“|!!|&amp|"|'','',tweet)
7.     tweet = tweet.translate(str.maketrans(' ','
   ',string.punctuation))
```

```
8.      emoji_pattern = re.compile("["\U0001F600-\U0001F64F"
   "\U0001F300-\U0001F5FF""\U0001F680-\U0001F6FF""\U0001F1E0-
   \U0001F1FF""\U00002500-\U00002BEF""\U00002702-
   \U000027B0""\U000024C2-\U0001F251""\U0001f926-
   \U0001f9af""\U00010000-\U0010ffff""\u2640-\u2642""\u2600-
   \u2B55""\u200d""\u23cf""\u23e9""\u231a""\ufe0f""\u3030"
   "]+", flags=re.UNICODE)
9.      return emoji_pattern.sub('', tweet)
10.tweet_polridf['text sudah bersih'] =
   np.vectorize(text_clean)(tweet_polridf['text'],
   "(?:\@|http?\://|https?\://|www)\S+")
```

Gambar 4. 19 Kode Pemrograman Regular Expression

Pada baris ke-1 kode pemrograman *Regular Expression*, terdapat sebuah fungsi yang bernama text_clean didefinisikan dengan dua parameter, yaitu tweet dan pattern. Pada baris kode ke-2, variabel r di-*assign* dengan hasil dari mengaplikasikan fungsi re.findall pada pattern dan tweet. Fungsi ini mencari semua kemunculan pattern dalam tweet dan mengembalikan data berupa daftar data yang cocok dengan pencarian yang dimaksudkan.

Pada baris ke-3, sebuah *loop* dimulai untuk berjalan berulang pada setiap elemen I dalam *list* r. Didalam loop yang telah dimulai sebelumnya, pada baris ke-4 digunakan fungsi re.sub untuk menggantikan kemunculan i dengan *string* kosong dalam tweet. Proses ini secara efektif menghilangkan pattern yang cocok dari tweet.

Baris kode ke-5 melakukan substitusi pada variabel tweet secara komprehensif yang memanfaatkan *Regular Expression* yang panjang untuk mencocokkan dengan berbagai pola. Pola yang dimaksud seperti angka, indikator *retweet* (RT), *tag* HTML(<.*?>), karakter *unicode*, tanda baca, dan lain-lain. Setiap pola yang dicocokkan kemudian digantikan dengan *string* kosong (' '), dengan tujuan untuk menghapuskan pola tersebut dari *tweet*.

Dalam baris kode ke-6, digunakan method translate() bersamaan dengan str.maketrans() untuk menghapus semua karakter tanda baca dari variabel tweet. Konstanta *string*.punctuation berisikan sebuah *string* dengan keseluruhan karakter tanda baca. Baris kode ke-7 melakukan penyusunan terhadap pola *Regular Expression* bernama emoji_pattern yang memiliki fungsi untuk menghapuskan karakter emoji dari tweet. Pola ini merupakan kombinasi dari berbagai macam *unicode* yang menggambarkan berbagai karakter emoji.

Baris kode ke-8 memanfaatkan method re.sub() dengan menggunakan emoji_pattern untuk menghapus semua karakter emoji dari tweet. Baris kode yang terakhir yaitu baris

ke-9 mengaplikasikan fungsi text_clean pada kolom 'text' pada *DataFrame* tweet_polridf. Dalam pengaplikasiannya, digunakan fungsi np.vectorize() pada setiap baris dari kolom 'text', mengirimkan pattern yang telah disediakan sebagai argumen. Hasil dari implementasi proses *Regular Expression* terdapat pada Tabel 4.3.

Tabel 4. 3 Penerapan Regular Expression pada data tweet

Data *tweet*	**Data *tweet* setelah penerapan *Regular Expression***
Selalu dukung bapak2 polri, semangat ✊ ✊ ✊ maju terus!! Berantas pelaku kejahatan ! Polri kinerja maksimal	Selalu dukung bapak polri semangatmaju terus Berantas pelaku kejahatan Polri kinerja maksimal
🥰 💪 Mantap dan solid, Polri Kinerja Maksimal bersama TNI semakin menunjukkan kinerja yang bagus untuk negara tercinta https://t.co/GA8E57azql	Mantap dan solid Polri Kinerja Maksimal bersama TNI semakin menunjukkan kinerja yang bagus untuk negara tercinta
Refleksi Kinerja Reformasi Polri https://t.co/YuBT8oMbe3	Refleksi Kinerja Reformasi Polri
@YASOzatulo @_MbakSri_ @DivHumas_Polri LsM PENJARA (Pemantau Kinerja Aparatur Negara) itu yg bs mengatasi nya.	LsM PENJARA Pemantau Kinerja Aparatur Negara itu yg bs mengatasi nya
@detikcom Ini menandakan kinerja polri tidak bisa di andalkan 🙏 Kalo masyarakat yg berisanksi, trus fungsi polisi apa pak? @PolhukamRI	Ini menandakan kinerja polri tidak bisa di andalkan Kalo masyarakat yg berisanksi trus fungsi polisi apa pak
Anggaran Pencitraan Besar, Kinerja Tak Maksimal https://t.co/rclS2phH0P	Anggaran Pencitraan Besar Kinerja Tak Maksimal

2. Case Folding

Setelah data dibersihkan melalui proses penerapan *Regular Expression*, dilakukan proses *case folding* dengan tujuan untuk penyeragaman huruf pada data keseluruhan data *tweet* menjadi huruf kecil. Kode pemrograman untuk proses ini ditunjukkan pada Gambar 4.20.

```
1. tweet_polridf['Sudah case folding'] = tweet_polridf['text
   sudah bersih'].apply(lambda tweet: tweet.lower())
```

Gambar 4. 20 Kode Pemrograman Case Folding

Arti dari kode pada Gambar 4.20 adalah dilakukan pengambilan pada data teks *tweet* di kolom 'text sudah bersih'. Kemudian, data teks yang telah diambil tadi dikonversi menjadi huruf kecil seluruhnya dengan memanfaatkan fungsi lambda. Setelah dikonversi, hasil pengolahan data *tweet* disematkan ke kolom baru yaitu 'Sudah case folding' pada *DataFrame* tweet_polridf. Dalam Tabel 4.4 ditunjukkan perbandingan data sebelum dan sesudah dilakukan penerapan *Case Folding* pada data.

Tabel 4. 4 Penerapan Case Folding pada data tweet

Data *tweet*	**Data *tweet* setelah penerapan *Case Folding***
Selalu dukung bapak polri semangatmaju terus Berantas pelaku kejahatan Polri kinerja maksimal	selalu dukung bapak polri semangatmaju terus berantas pelaku kejahatan polri kinerja maksimal
Mantap dan solid Polri Kinerja Maksimal bersama TNI semakin menunjukkan kinerja yang bagus untuk negara tercinta	mantap dan solid polri kinerja maksimal bersama tni semakin menunjukkan kinerja yang bagus untuk negara tercinta
Refleksi Kinerja Reformasi Polri	refleksi kinerja reformasi polri
LsM PENJARA Pemantau Kinerja Aparatur Negara itu yg bs mengatasi nya	lsm penjara pemantau kinerja aparatur negara itu yg bs mengatasi nya
Ini menandakan kinerja polri tidak bisa di andalkan Kalo masyarakat yg berisanksi trus fungsi polisi apa pak	ini menandakan kinerja polri tidak bisa di andalkan kalo masyarakat yg berisanksi trus fungsi polisi apa pak
Anggaran Pencitraan Besar Kinerja Tak Maksimal	anggaran pencitraan besar kinerja tak maksimal

3. Tokenization

Setelah dilakukan penyeragaman huruf pada keseluruhan data, dilakukan proses *Tokenization*. Fungsi dari proses ini adalah untuk memenggal frasa, simbol, kata, dan komponen krusial lainnya (disebut dengan token) dari sebuah teks. Manfaat dari proses *Tokenization* ini sendiri yaitu mempermudah pendeteksian prosedur perhitungan dari frekuensi munculnya setiap kata dalam corpus.

Proses *Tokenization* ini sendiri dijalankan dengan memanfaatkan *Library* yang terdapat pada bahasa *Python* yaitu nltk. Kode pemrograman untuk proses ini ditunjukkan pada Gambar 4.21 dan Tabel 4.4 sebagai perbandingan data sebelum dan sesudah dilakukan penerapan *Tokenization* pada data.

```
1. from nltk.tokenize import TweetTokenizer
2. tweet_tokenizer = TweetTokenizer(r'\w+|$[0-9]+|\S+')
3. tweet_polridf['Sudah ditokenisasi'] = tweet_polridf['Sudah case
   folding'].apply(tweet_tokenizer.tokenize)
```

Gambar 4. 21 Kode Pemrograman Tokenization

Baris kode ke-1 pada tahapan *tokenization* ini mengimpor kelas TweetTokenizer dari modul nltk.tokenize. NLTK merupakan sebuah *library* yang ada di Python yang menyediakan *tools* untuk mengolah data bahasa manusia berbentuk teks. Kelas TweetTokenizer secara khusus didesain untuk memisahkan data *tweet* ke dalam bentuk huruf secara terpisah atau yang disebut dengan token.

Dalam baris kode ke-2 diperlihatkan bahwa kelas TweetTokenizer dibuat dan disematkan ke dalam variabel tweet_tokenizer. Argumen yang diberikan pada TweetTokenizer merupakan sebuah *Regular Expression* yang mendefinisikan aturan dari proses tokenisasi. Pada kode ini, *Regular Expression* r'\w+|$[0-9]+|\S+' memiliki arti bahwa *tokenizer* akan memisahkan teks ke dalam berbagai bentuk urutan. Contohnya seperti urutan karakter alfanumerik (\w+), urutan karakter angka ($[0-9]+), atau urutan karakter non whitespace (\S+).

Baris kode ke-3 mengaplikasikan method tweet_tokenizer.tokenize pada setiap entri dalam kolom 'Sudah case folding' pada *DataFrame* tweet_polridf. Hasil dari pengaplikasian method ini tersimpan dalam sebuah kolom baru yang bernama 'Sudah ditokenisasi'. Method apply dalam pandas sendiri dipergunakan utuk mengaplikasikan sebuah fungsi melewati sumbu dari *DataFrame*. Pada Tabel 4.4 ditunjukkan implementasi proses *Tokenization* pada data *tweet*.

Tabel 4.4 Penerapan *Tokenization* pada data *tweet*

Data *tweet*	**Data *tweet* setelah penerapan *Tokenization***
selalu dukung bapak polri semangatmaju terus berantas pelaku kejahatan polri kinerja maksimal	['selalu', 'dukung', 'bapak', 'polri', 'semangatmaju', 'terus', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']

mantap dan solid polri kinerja maksimal bersama tni semakin menunjukkan kinerja yang bagus untuk negara tercinta	['mantap', 'dan', 'solid', 'polri', 'kinerja', 'maksimal', 'bersama', 'tni', 'semakin', 'menunjukkan', 'kinerja', 'yang', 'bagus', 'untuk', 'negara', 'tercinta']
refleksi kinerja reformasi polri	['refleksi', 'kinerja', 'reformasi', 'polri']
lsm penjara pemantau kinerja aparatur negara itu yg bs mengatasi nya	['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'itu', 'yg', 'bs', 'mengatasi', 'nya']
ini menandakan kinerja polri tidak bisa di andalkan kalo masyarakat yg berisanksi trus fungsi polisi apa pak	['ini', 'menandakan', 'kinerja', 'polri', 'tidak', 'bisa', 'di', 'andalkan', 'kalo', 'masyarakat', 'yg', 'berisanksi', 'trus', 'fungsi', 'polisi', 'apa', 'pak']
anggaran pencitraan besar kinerja tak maksimal	['anggaran', 'pencitraan', 'besar', 'kinerja', 'tak', 'maksimal']

4. Normalization

Setelah proses *tokenization* selesai diimplementasikan pada data *tweet*, langkah selanjutnya adalah dengan melakukan *Normalization* untuk mengubah bentuk kata dari yang tidak baku menjadi baku sesuai dengan kaidah struktur KBBI (Kamus Besar Bahasa Indonesia). Kode pemrograman untuk proses ini ditunjukkan pada Gambar 4.22 dan Tabel 4.5 sebagai perbandingan data sebelum dan sesudah dilakukan penerapan *Normalization* pada data.

```
1. normalisasi_kata = pd.read_csv("kbba.csv")
2. normalisasi_kata_dict={}
3. for index, row in normalisasi_kata.iterrows():
4.     if row[0] not in normalisasi_kata_dict:
5.         normalisasi_kata_dict[row[0]] = row[1]
6. def normalized_term(document):
7.     return [normalisasi_kata_dict[term] if term in
   normalisasi_kata_dict else term for term in document]
8. tweet_polridf['Sudah dinormalisasi'] = tweet_polridf['Sudah
   ditokenisasi'].apply(normalized_term)
```

Gambar 4. 22 Kode Pemrograman Normalization

Baris kode ke-1 menunjukkan proses pembacaan sebuah file csv bernama “kbba.csv” ke dalam sebuah *DataFrame* pandas bernama normalisasi_kata. File CSV ini berisikan pasangan kata, yaitu bentuk tidak baku dan bentuk baku dari sebuah kata. Baris kode ke-2 menginisialisasi sebuah kamus kosong bernama normalisasi_kata_dict. Kamus ini

nantinya digunakan untuk menyimpan pasangan kata dari *DataFrame* normalisasi_kata, sehingga pencarian bentuk baku sebuah huruf dapat dilakukan secara efisien.

Dalam baris kode ke-3, dimulai sebuah *loop* yang berjalan secara berulang di setiap baris pada *DataFrame* normalisasi_kata. Untuk setiap barisnya, indeks dari baris tersebut disematkan pada variabel index dan data dari baris tersebut disematkan ke variabel row. Baris kode ke-4 melakukan pengecekan untuk mengetahui apakah elemen pertama variabel row belum menjadi sebuah *key* dalam kamus normalisasi_kata_dict.

Baris kode ke-5 menambah sebuah entri pada kamus normalisasi_kata_dict. *Key*-nya sendiri merupakan elemen pertama dari row (kata yang belum dinormalisasi), dan *value*-nya adalah elemen kedua dari row (kata yang sudah dinormalisasi). Baris kode ke-6 mendefiniskan sebuah fungsi bernama normalized_term. Fungsi ini mengambil daftar kata-kata yang direferensikan sebagai document dan mengembalikannya ke dalam daftar baru. Pada daftar baru ini setiap kata digantikan dengan bentuk yang sudah dinormalisasi.

Baris kode ke-7 merupakan *body* dari fungsi normalized_term. *Body* ini merupakan sebuah *comprehension list* yang berjalan melewati term dalam document. Jika term terdapat dalam kamus normalisasi_kata_dict, maka term akan digantikan bentuknya ke dalam bentuk yang sudah dinormalisasi. Jika tidak terdapat dalam kamus, maka tidak akan ada perubahan pada term.

Baris kode ke-8 mengaplikasikan fungsi normalized_term pada setiap entri dalam kolom ‘Sudah ditokenisasi’ pada *DataFrame* tweet_polridf. Hasil pengaplikasian fungsi tersebut berupa token pada setiap data *tweet* yang telah dinormalisasi. Token-token tersebut disimpan ke dalam sebuah kolom baru yang bernama ‘Sudah dinormalisasi’. Dalam Tabel 4.5 ditunjukkan penerapan *Normalization* pada data *tweet*.

Tabel 4. 5 Penerapan Normalization pada data tweet

Data *tweet*	**Data *tweet* setelah penerapan *Normalization***
['selalu', 'dukung', 'bapak', 'polri', 'semangatmaju', 'terus', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']	['selalu', 'dukung', 'bapak', 'polri', 'semangatmaju', 'terus', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']
['mantap', 'dan', 'solid', 'polri', 'kinerja', 'maksimal', 'bersama', 'tni', 'semakin', 'menunjukkan', 'kinerja', 'yang', 'bagus', 'untuk', 'negara', 'tercinta']	['mantap', 'dan', 'solid', 'polri', 'kinerja', 'maksimal', 'bersama', 'tni', 'semakin', 'menunjukkan', 'kinerja', 'yang', 'bagus', 'untuk', 'negara', 'tercinta']
['refleksi', 'kinerja', 'reformasi', 'polri']	['refleksi', 'kinerja', 'reformasi', 'polri']

['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'itu', 'yg', 'bs', 'mengatasi', 'nya']	['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'itu', 'yang', 'bisa', 'mengatasi', 'nya']
['ini', 'menandakan', 'kinerja', 'polri', 'tidak', 'bisa', 'di', 'andalkan', 'kalo', 'masyarakat', 'yg', 'berisanksi', 'trus', 'fungsi', 'polisi', 'apa', 'pak']	['ini', 'menandakan', 'kinerja', 'polri', 'tidak', 'bisa', 'di', 'andalkan', 'kalau', 'masyarakat', 'yang', 'berisanksi', 'terus', 'fungsi', 'polisi', 'apa', 'pak']
['anggaran', 'pencitraan', 'besar', 'kinerja', 'tak', 'maksimal']	['anggaran', 'pencitraan', 'besar', 'kinerja', 'tak', 'maksimal']

5. Remove Stopwords

Setelah kata-kata pada data *tweet* dikembalikan ke dalam bentuk bakunya, selanjutnya adalah proses *Remove Stopwords*. Proses ini sendiri bertujuan untuk menghilangkan kata yang apabila dihapuskan tidak memiliki pengaruh apa-apa terhadap informasi yang disampaikan dalam suatu kalimat. Contoh dari *Stopwords* dalam Bahasa Indonesia sendiri seperti "kamu", "saya", "yang", "untuk", dan kata lain yang sejenis. Proses ini menggunakan *Library Python* yang sama seperti pada proses *Tokenization* yaitu nltk. Kode pemrograman untuk proses ini ditunjukkan pada Gambar 4.23 dan Tabel 4.6 sebagai perbandingan data sebelum dan sesudah dilakukan penerapan *Remove Stopwords* pada data.

```
1. from nltk.corpus import stopwords
2. stopword = stopwords.words('indonesian')
3. daftar_stopwords = pd.read_csv("ID-Stopwords.csv",
   names=["stopwords"], header=None)
4. stopword.extend(["dehh","wkwk","dpt","dsuruh","kl","hahaha","nya","
   pke","dehh","jg","yg","yahaha","hahaha","haha","hehe","km","ia","ms
   ih","ja","ddpn","dsuruh","dlu","terhubung","ng","lah"])
5. stopword.extend(daftar_stopwords["stopwords"][0].split(' '))
6. stopword = set(stopword)
7. def stopwords(tweet_stopwords):
8.     tweet_stopwords = [word for word in tweet_stopwords if word not
   in stopword]
9.     return tweet_stopwords
10. tweet_polridf['Sudah Remove Stopwords'] = tweet_polridf['Sudah
   dinormalisasi'].apply(stopwords)
```

Gambar 4. 23 Kode Pemrograman Remove Stopwords

Baris kode ke-1 pada tahapan *Remove Stopwords* memiliki fungsi untuk mengimpor modul stopwordsdari *package* nltk.corpus. Isi dari modul stopwords ini sendiri merupakan daftar huruf yang kerap digunakan dalam berbagai bahasa. Seringkali, kata-kata yang cukup umum ini di-*filter* pada saat menjalankan *task* berupa pengolahan bahasa alami. Baris kode ke-2 berisikan daftar *stopwords* dalam Bahasa Indonesia dengan cara memanggil method words dalam modul stopwords dengan argumen 'indonesian'.

Baris ke-3 melakukan pembacaan file .csv bernama ID-Stopwords.csv ke dalam bentuk *DataFrame* pandas yang bernama daftar_stopwords. Argumen names=["stopwords"] mengatur nama kolom menjadi "stopwords". Kode header=none mengindikasikan bahwa file .csv ini tidak memiliki baris header. Baris kode ke-4 berisikan tambahan kata-kata untuk memperpanjang daftar kata yang termasuk dalam *stopwords*.

Baris kode ke-5 memperpanjang *list* stopword dengan kata-kata yang terdapat pada baris pertama *DataFrame* daftar_stopwords, yang mana kata-kata ini dipisahkan dengan spasi. Baris kode ke-6 mengkonversikan *list* stopword ke dalam sebuah *set*. Baris kode ke-7 mendefinisikan sebuah fungsi bernama stopwords yang mengambil kata-kata dari tweet_stopwords untuk dijadikan sebagai argumen.

Baris kode ke-8 merupakan sebuah *comprehension list* yang melakukan penyaringan terhadap kata-kata *stopwords* dalam tweet_stopwords. Baris kode ini membuat sebuah *list* baru yang hanya berisikan kata-kata yang tidak termasuk ke dalam *set* stopword. Baris kode ke-9 mengembalikan *list* kata yang telah dilakukan penyaringan dari fungsi stopwords.

Baris kode yang terakhir yaitu baris ke-10 mengaplikasikan fungsi stopwordske dalam setiap entri yang ada pada kolom 'Sudah dinormalisasi' yang terdapat pada *DataFrame* tweet_polridf. Hasil pemrosesan pada kode ini adalah sebuah *list* yang berisikan kata dari setiap data *tweet* yang sudah dibersihkan dari *stopwords*. Kemudian, hasilnya disimpan dalam sebuah kolom baru yang bernama 'Sudah Remove Stopwords'. Pada Tabel 4.6 ditunjukkan penerapan *Remove Stopwords* pada data *tweet*.

Tabel 4. 6 Penerapan Remove Stopwords pada data tweet

Data *tweet*	**Data *tweet* setelah penerapan *Remove Stopwords***
['selalu', 'dukung', 'bapak', 'polri', ' semangatmaju', 'terus', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']	['dukung', 'polri', ' semangatmaju', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']

['mantap', 'dan', 'solid', 'polri', 'kinerja', 'maksimal', 'bersama', 'tni', 'semakin', 'menunjukkan', 'kinerja', 'yang', 'bagus', 'untuk', 'negara', 'tercinta']	['mantap', 'solid', 'polri', 'kinerja', 'maksimal', 'tni', 'kinerja', 'bagus', 'negara', 'tercinta']
['refleksi', 'kinerja', 'reformasi', 'polri']	['refleksi', 'kinerja', 'reformasi', 'polri']
['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'itu', 'yang', 'bisa', 'mengatasi', 'nya']	['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'mengatasi']
['ini', 'menandakan', 'kinerja', 'polri', 'tidak', 'bisa', 'di', 'andalkan', 'kalau', 'masyarakat', 'yang', 'berisanksi', 'terus', 'fungsi', 'polisi', 'apa', 'pak']	['menandakan', 'kinerja', 'polri', 'andalkan', 'masyarakat', 'berisanksi', 'fungsi', 'polisi']
['anggaran', 'pencitraan', 'besar', 'kinerja', 'tak', 'maksimal']	['anggaran', 'pencitraan', 'kinerja', 'maksimal']

6. Stemming

Langkah terakhir dalam tahapan *preprocessing* adalah melakukan *Stemming*. Proses *stemming* merupakan mengembalikan kata yang ada dalam suatu teks menjadi bentuk kata dasarnya. Proses *Stemming* ini sendiri memanfaatkan *Library* pada *Python* yaitu Sastrawi.

Seperti yang didefinisikan oleh laman Github resmi Sastrawi, *library* ini sendiri bekerja dengan cara menghapus imbuhan yang masih merekat pada suatu kata. Sehingga, kata tersebut berubah menjadi bentuk dasarnya (Librarian, 2017),. Kode pemrograman untuk proses ini ditunjukkan pada Gambar 4. 24 dan Tabel 4.7 sebagai perbandingan data sebelum dan sesudah dilakukan penerapan *Stemming* pada data.

```
1. from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
2. import swifter
3. factory = StemmerFactory()
4. stemmer = factory.create_stemmer()
5.  def stemmer_kata(istilah):
6.     return stemmer.stem(istilah)
7. istilah_dict = {}
8. twit = tweet_polridf['Sudah Remove Stopwords']
9. for doc in twit:
10.     for istilah in doc:
11.         if istilah not in istilah_dict:
12.             istilah_dict[istilah]='  '
13.     print(len(istilah_dict))
14. for istilah in istilah_dict:
```

```
15.     istilah_dict[istilah] = stemmer_kata(istilah)
16.     print (istilah,":",istilah_dict[istilah])
17.     print(len(istilah_dict))
18. def  get_stemmed_kata(doc):
19.     return[istilah_dict[istilah] for istilah in doc]
20. tweet_polridf['Sudah Stemming'] = tweet_polridf['Sudah Remove
    Stopwords'].apply(get_stemmed_kata)
```

Gambar 4. 24 Kode Pemrograman Stemming

Pertama-tama, dilakukan impor kelas StemmerFactory dari modul Sastrawi.Stemmer.StemmerFactory pada baris ke-1. Sastrawi merupakan sebuah *library* untuk melakukan *stemming* terhadap teks Bahasa Indonesia. Baris ke-2 melakukan impor modul swifter, yang memungkinkan fungsi apply pada pandas bekerja dengan lebih cepat.

Baris ke-3 kode ini adalah membuat sebuah *instance* dari kelas StemmerFactory dan menyematkannya ke dalam variabel factory. Baris ke-4 berisikan mengenai pemanggilan method create_stemmer pada *object* factory untuk membuat sebuah *stemmer* yang kemudian disematkan pada variabel stemmer. Baris kode ke-5 mendefinisikan sebuah fungsi bernama stemmer_kata yang mengambil sebuah argumen tunggal, yaitu istilah.

Baris kode ke-6 merupakan *body* dari fungsi stemmer_kata. *Body* ini memanggil method stem pada objek stemmer dengan istilah sebagai argumen untuk kemudian hasilnya dikembalikan. Dalam baris kodeke-7 dibuat sebuah kamus kosong yang bernama istilah_dict. Baris kode ke-8 menyematkan kolom 'Sudah Remove Stopwords' pada *DataFrame* tweet_polridf dalam variabel twit.

Baris kode ke-9 memulai sebuah perulangan yang mengiterasi satu sama lain pada setiap item yang ada pada variabel twit. Baris kode ke-10 memulai sebuah perulangan for *nested* yang mengiterasi antar dalam dokumen yang sedang diolah (doc). Baris ke-11 melakukan pengecekan apakah huruf yang ada dalam variabel istilah belum termasuk sebagai *key* dalam kamus istilah_dict. Bar is kode ke-12 berfungsi untuk menetapkan nilai dari *key* istilah dalam kamus istilah_dict ke dalam sebuah karakter spasi tunggal.

Baris ke-13 mencetak nomor dari *key* dalam kamus istilah_dict. Baris kode ke-14 menunjukkan dimulainya perulangan for yang mengiterasi setiap *key* dalam kamus istilah_dict. Baris ke-15 memanggil fungsi stemmer_kata dengan istilah sebagai argumennya, dan menyematkan hasil sebagai *value* untuk *key* istilah di dalam kamus istilah_dict.

Baris kode ke-16 mencetak variabel istilah dan *value* yang cocok ke dalam kamus istilah_dict. Baris kode ke-17 mencetak jumlah *key* yang terdapat pada kamus istilah_dict. Baris kode ke-18 mendefinisikan sebuah fungsi yang bernama get_stemmed_kata yang mengambil satu argumen, (doc). Baris kode ke-19 merupakan *body* dari fungsi get_stemmed_kata. *Body* ini mengembalikan daftar kata pada (doc) yang telah di-*stemmed* dengan cara mencari setiap huruf yang ada dalam kamus istilah_dict.

Pada baris kode yang terakhir ini, yaitu baris ke-20, diaplikasikan fungsi get_stemmed_kata pada setiap item yang ada di kolom 'Sudah Remove Stopwords'. Letak kolom ini terdapat pada *DataFrame* tweet_polridf. Hasil dari tahapan *stemming* ini adalah kata-kata yang ada pada setiap data *tweet* disimpan dalam sebuah kolom baru yang bernama 'Sudah Stemming'. Tabel 4.7 menunjukkan Penerapan *Stemming* pada data *tweet*.

Tabel 4. 7 Penerapan Stemming pada Data Tweet

Data *tweet*	**Data *tweet* setelah penerapan *Stemming***
['dukung', 'polri', ' semangatmaju', 'berantas', 'pelaku', 'kejahatan', 'polri', 'kinerja', 'maksimal']	['dukung', 'polri', 'semangatmaju', 'berantas', 'laku', 'jahat', 'polri', 'kerja', 'maksimal']
['mantap', 'solid', 'polri', 'kinerja', 'maksimal', 'tni', 'kinerja', 'bagus', 'negara', 'tercinta']	['mantap', 'solid', 'polri', 'kerja', 'maksimal', 'tni', 'kerja', 'bagus', 'negara', 'cinta']
['refleksi', 'kinerja', 'reformasi', 'polri']	['refleksi', 'kerja', 'reformasi', 'polri']
['lsm', 'penjara', 'pemantau', 'kinerja', 'aparatur', 'negara', 'mengatasi']	['lsm', 'penjara', 'pantau', 'kerja', 'aparatur', 'negara', 'atas']
['menandakan', 'kinerja', 'polri', 'andalkan', 'masyarakat', 'berisanksi', 'fungsi', 'polisi']	['tanda', 'kerja', 'polri', 'andal', 'masyarakat', 'berisanksi', 'fungsi', 'polisi']
['anggaran', 'pencitraan', 'kinerja', 'maksimal']	['anggar', 'citra', 'kerja', 'maksimal']

4.4 Hasil Implementasi Ekstraksi Fitur

Tahap selanjutnya setelah *dataset* telah selesai dibersihkan melalui *Data Preprocessing* adalah Ekstraksi Fitur. Proses ini dilakukan dengan memanfaatkan *library* python yang bernama sklearn. Pada *library* ini dimanfaatkan modul sklearn.feature_extraction.text dan kelas TfidfVectorizer untuk melakukan pembobotan teks dengan menggunakan *TF-IDF*. Ekstraksi Fitur sendiri memiliki fungsi untuk melakukan pengubahan bentuk data dari kata

menjadi angka yang nantinya akan digunakan sebagai data latih untuk model prediksi sentimen. Namun, sebelum proses ekstraksi fitur dijalankan terdapat pemisahan antara data latih dan data uji. Hal ini dilakukan agar data yang dijadikan sebagai data pelatihan tidak bercampur dengan data uji, sehingga model analisis sentimen dapat berjalan dengan baik. Pemisahan data uji dan data latih dibagi dengan rasio sebesar 80:20, yang mana data latih sebesar 80% dan data uji sebesar 20%. Kode pemrograman untuk melakukan proses pemisahan data ini ditunjukkan pada Gambar 4.25.

```
1. from sklearn.model_selection import train_test_split
2. X = tweet_polridf['Sudah Stemming']
3. y = tweet_polridf['Label']
4. X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
5. joined_train = [' '.join(tokens) for tokens in X_train]
6. joined_test = [' '.join(tokens) for tokens in X_test]
```

Gambar 4. 25 Kode Pemrograman Proses Pemisahan Data

Pada baris kode ke-1, modul scikit-learn yang bernama sklearn.model_selection melakukan impor *library* train_test_split yang berguna untuk proses pemisahan data latih dan data uji. Baris kode ke-2 menyematkan data teks pada kolom 'Sudah Stemming' pada variabel X. Baris kode ke-3 menyematkan label yang terdapat pada setiap data teks yang terdapat pada kolom 'Label' ke dalam variabel y. Baris kode ke-4 memisahkan data ke dalam data uji dan data latih, dengan test_size 0.2 yang berarti data tes diambil sejumlah 20 persen dari keseluruhan data. Dengan kata lain, data dibagi rasionya menjadi 80:20. Baris kode ke-5 melakukan penggabungan token yang terdapat pada X_train untuk digunakan sebagai data latih. Baris kode ke-5 melakukan penggabungan token yang terdapat pada X_test untuk digunakan sebagai data uji.

Pada tahapan ini, data tidak perlu ditokenisasi karena data *tweet* yang telah diolah dalam kolom 'Sudah Stemming' sudah berbentuk *token*. Proses dari implementasi ekstraksi fitur ini sendiri ditunjukkan dalam kode pemrograman pada Gambar 4.26.

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2. tfidf_vectorizer = TfidfVectorizer()
3. X_train_tfidf = tfidf_vectorizer.fit_transform(joined_train)
4. X_test_tfidf = tfidf_vectorizer.transform(joined_test)
```

Gambar 4. 26 Kode Pemrograman Ekstraksi Fitur

Baris kode ke-1 melakukan impor *library* TfidfVectorizer yang berasal dari modul sklearn.feature_extraction.text. Kelas ini digunakan untuk mengubah data dari bentuk teks menjadi representasi matriks TF-IDF. Baris kode ke-2 membuat *instance* dari kelas TfidfVectorizer and menyimpannya ke dalam variabel tfidf_vectorizer. Baris kode ke-3 mengubah data latih yang terkandung dalam (joined_train) ke dalam representasi matriks *TF-IDF*. Baris kode ke-4 mengubah data uji yang terkandung dalam (joined_test) ke dalam bentuk matriks *TF-IDF*.

Ukuran matriks hasil *TF-IDF* ini adalah 2251 × 2012. Angka 2251 sendiri merepresentasikan baris pada matriks. Angka ini merupakan jumlah dokumen dalam file kita, yaitu data latih yang kita gunakan. Setiap baris dalam matriks ini sesuai dengan dokumen yang berbeda-beda. Angka 2012 merepresentasikan kolom pada matriks. Angka ini merupakan ukuran kosakata dari keseluruhan dokumen dalam file ini. Setiap kolom pada matriks ini sesuai dengan *term*-nya.

	aamiin	abad	abalabal	abdi	acab	acara	acung	ada	adaaa	adaada	...	ylbhi	yok	yosua	yudo	yuuk	za	zero	zona	zulham	zulkifli
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2246	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2247	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2248	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2249	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2250	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2251 rows × 2012 columns

Gambar 4. 27 Perubahan Data Teks Menjadi Bentuk Angka Pembobotan

	Row	Column	Value
0	0	benerbener	0.745193
1	0	kerja	0.111565
2	0	maksimal	0.429972
3	0	polri	0.127595
4	0	salut	0.480713
...	...	...	...
19562	2250	kerja	0.102030
19563	2250	komitmentuntas	0.617084
19564	2250	masyarakat	0.278121
19565	2250	polri	0.116689
19566	2250	suka	0.617084

Gambar 4. 28 Hasil Pembobotan TF-IDF yang bernilai selain 0

Dalam matriks *tf-idf* yang terdapat dalam Gambar 4. 27 dan Gambar 4. 28, ditunjukkan bahwasanya terdapat kata yang memiliki nilai nol dan tidak nol. Nilai nol sendiri mengartikan bahwa kata ini tidak terdapat dalam sebuah dokumen, sehingga nilai pembobotannya adalah nol. Dapat diartikan pula nilai nol ini sendiri menunjukkan tidak adanya kontribusi dan relevansi kata tersebut dalam suatu dokumen. Untuk nilai tidak nol, diartikan bahwa kata tersebut memiliki bobot *tf-idf* karena kata tersebut memiliki kontribusi dalam suatu dokumen. Nilai tidak nol juga mengindikasikan bahwa suatu kata memiliki kepentingan dalam dokumen tersebut.

4.5 Imbalanced Data Handling

Pada saat tahapan ekstraksi fitur telah selesai dilakukan dan dihasilkan bentuk data berupa *feature* untuk dapat diolah dalam proses klasifikasi, terdapat ketimpangan dalam label yang disematkan pada data yang mengakibatkan perbandingan antara kelas positif, netral, dan negatif pada data sangat timpang. Untuk menanganinya, dilakukan *handling* terhadap data yang *imbalanced* menggunakan *Synthetic Minority Oversampling Technique* (SMOTE) yang bekerja dengan menghasilkan sampel buatan dari proses pemilihan *nearest neighbors* secara acak dan menghubungkan sampel buatan dalam garis antara contoh data asli dengan data *neighbor* yang dipilih. Sampel buatan ini dibuat dengan cara menginterpolasi nilai yang ada

dalam fitur. Data yang digunakan oleh SMOTE ini merupakan data yang telah diubah bentuknya ke dalam bentuk angka pembobotan.

Langkah pertama yang dilakukan sebelum melakukan *Imbalanced Data Handling* ini adalah *import library* yang dibutuhkan. *Library* yang digunakan dalam proses ini di antaranya adalah SMOTE yang berasal dari modul imblearn.over_sampling. Kode pemrograman proses *Imbalanced Data Handling* ini ditampilkan pada Gambar 4. 29

```
1. from imblearn.over_sampling import SMOTE
2. smote = SMOTE(random_state=42)
3. X_train_resampled_imb, y_train_resampled_imb =
   smote.fit_resample(X_train_tfidf, y_train)
```

Gambar 4. 29 *Imbalanced Data Handling* Serta Pemisahan Data Latih Hasil *Oversampling*

Baris pertama pada kode ini melakukan impor terhadap kelas SMOTE dari imblearn.over_sampling. Baris ke-2 membuat sebuah *instance* dari kelas SMOTE dengan random_state=42. Baris kode ke-3 mengaplikasikan SMOTE untuk melakukan sampel ulang terhadap data latih X_train_tfidf dan labelnya yaitu y_train. Proses ini menghasilkan sampel buatan untuk kelas minoritas, yang disimpan ke dalam X_train_resampled_imb dan y_train_resampled_imb.

Tujuan dari dilakukannya proses ini ialah untuk menyeimbangkan jumlah kelas data yang timpang agar selaras dengan kelas mayoritas. Proses ini dilakukan dengan cara *resample* fitur X_train dan label y_train untuk menyeimbangkan pembagian kelas. Data yang telah di-*oversample* disimpan dalam X_train_resampled dan y_train_resampled, yang mana dapat digunakan untuk melatih model pembelajaran mesin dengan data yang *balance*.

Setelah dilakukannya *oversampling* pada data latih, terdapat penambahan pada data latih sebagai hasil dari proses *oversampling* yang membuat data latih sintetis berdasarkan data latih yang telah ada sebelumnya. Data latih sebelum proses *oversampling* berjumlah 2251, dengan jumlah data pada kelas positif sebanyak 1857 data, kelas netral sebanyak 262 data, dan kelas negatif sebanyak 132. Setelah proses *oversampling* selesai dilakukan, jumlah data latih bertambah menjadi 5580. Dengan kata lain, jumlah data latih mengalami peningkatan sebesar 147,9% dari sebelumnya. Jumlah data pada masing-masing kelas setelah proses *oversampling* selesai dilakukan adalah pada kelas positif berjumlah 1860, kelas netral sebanyak 1860, dan kelas negatif sebesar 1860. Data uji sebelum proses *oversampling* berjumlah 583, dan setelah proses *oversampling* jumlah data uji tidak berubah. Hal ini disebabkan oleh proses

oversampling yang hanya mereplikasi data latih saja. Perubahan jumlah data latih dapat dilihat pada Gambar 4.31 dan Gambar 4.32.

```
In [17]: print(X_train)
```

```
      0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1095  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1130  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1294  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
860   0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

      zulkifli
1744       0.0
2185       0.0
2370       0.0
2513       0.0
1025       0.0
...        ...
1638       0.0
1095       0.0
1130       0.0
1294       0.0
860        0.0

[2251 rows x 2432 columns]
```

Gambar 4. 30 Jumlah Data Latih Sebelum *Oversampling*

```
print(X_train_resampled_imb)
```

```
  (0, 197)      0.7451926561793791
  (0, 859)      0.11156520063784851
  (0, 1055)     0.4299718930247526
  (0, 1380)     0.12759526506800406
  (0, 1541)     0.48071273201767256
  (1, 16)       0.47811732997039547
  (1, 815)      0.6311262627479997
  (1, 859)      0.08632814975120515
  (1, 1055)     0.16635419359097842
  (1, 1380)     0.09873206956433786
  (1, 1611)     0.30660540653473933
  (1, 1801)     0.4839445166650423
  (2, 205)      0.4369895249992086
  (2, 438)      0.45688230299032534
  (2, 859)      0.130736473709337
  (2, 1055)     0.2519289561924301
  (2, 1380)     0.29904226267020845
  (2, 1611)     0.46432721871245647
  (2, 1836)     0.463370748784275
  (3, 859)      0.16124272774859258
  (3, 1090)     0.43953017725126586
  (3, 1324)     0.513916695261766
  (3, 1380)     0.1844106268777687
  (3, 1541)     0.6947635259996292
  (4, 100)      0.16820354204872137
  :     :
  (5577, 1425)  0.24051960671046888
  (5577, 1808)  0.350420518749823
  (5578, 100)   0.028877571974437466
  (5578, 109)   0.05439556991962945
  (5578, 130)   0.05742554623414062
  (5578, 141)   0.16407879262766226
  (5578, 247)   0.2839132433709869
  (5578, 326)   0.2867683700569855
  (5578, 331)   0.2867683700569855
  (5578, 336)   0.08890774472811186
  (5578, 489)   0.27228190292305104
  (5578, 685)   0.16239054042128892
  (5578, 859)   0.013728405906103084
  (5578, 908)   0.2620035909197328
  (5578, 1320)  0.2867683700569855
  (5578, 1380)  0.0157009495840598
  (5578, 1946)  0.49503424757159664
  (5579, 287)   0.20057803361627285
  (5579, 384)   0.3472439101198617
  (5579, 859)   0.10004912265110132
  (5579, 1084)  0.3472439101198617
  (5579, 1380)  0.05721225907139061
  (5579, 1850)  0.2823325207522846
  (5579, 1900)  0.3472439101198617
  (5579, 1936)  0.6944878202397234
```

Gambar 4. 31 Jumlah Data Latih Setelah *Oversampling*

4.6 Training Model

Proses klasifikasi yang dipergunakan dalam penelitian ini memanfaatkan beberapa metode klasifikasi dengan tujuan untuk mengetahui perbandingan hasil yang didapat di antara dua metode berikut: Naïve Bayes Classifier dan Support Vector *Machine*. Langkah pertama yang dilakukan adalah pemanggilan *library* sklearn, selanjutnya adalah pembuatan fungsi yang memiliki tujuan untuk melakukan *training* pada model yang telah dibuat dengan memanfaatkan data *training* yang telah dipisahkan sebelumnya. Dilakukan pelatihan terhadap dua jenis metode klasifikasi yaitu *Naïve Bayes* dan *Support Vector Machine* dengan *hyperparameter default*.

Metode *Naïve Bayes* dalam penelitian ini menggunakan *classifier* MultinomialNB dan *Support Vector Machine* menggunakan SVC sebagai *classifier*-nya. MultinomialNB memiliki *hyperparameter* bernama alpha yang memiliki *value default* 1.0 dan fit_prior yang berupa boolean sehingga *value default*-nya adalah True. SVC memiliki lima *hyperparameter,* yaitu C yang bernilai *default* 1.0, kernel bernilai *default* 'rbf', degree bernilai *default* 3, gamma bernilai *default* 'scale' yang menggunakan 1/(n_features * X.var()), dan yang terakhir adalah coef0 yang memiliki nilai *default* 0.0. Kode pemrogramannya dapat dilihat pada Gambar 4.33

```
1. from sklearn.naive_bayes import
   MultinomialNB
2. from sklearn.svm import SVC
3. naive_bayes_model = MultinomialNB()
4. naive_bayes_model.fit(X_train_resampled_imb,
   y_train_resampled_imb)
5. svm_model = SVC()
6. svm_model.fit(X_train_resampled_imb,
   y_train_resampled_imb)
```

Gambar 4. 32 Kode *Training* Data Naïve Bayes dan Support Vector *Machine*

Baris kode ke-1 memiliki fungsi untuk mengimpor model *classifier* MultinomialNB milik model *Naïve Bayes*. Baris kode ke-2 melakukan impor *classifier* SVC yaitu *Support Vector Classifier* milik model SVM.

Baris kode ke-3 berfungsi untuk membuat sebuah *instance* dari *Multinomial Naïve Bayes Classifier* dan menyematkannya pada variabel naive_bayes_model. Naïve Bayes merupakan sebuah algoritma probablistik yang dipergunakan untuk klasifikasi teks, dan varian multinomialnya cocok digunakan untuk fitur lainnya, seperti penghitungan huruf pada data teks. Baris kode ke-4 melatih *Multinomial Naïve Bayes* pada data *training*. Method fit()

digunakan untuk melatih *classifier*, dimana X_train_resampled_imb merepresentasikan fitur *training*, dan y_train_resampled_imb merepresentasikan label dari target data yang dituju.

Baris kode ke-5 membuat sebuah *instance* dari *classifier Support Vector Machine* (SVM) dan menyematkannya pada variabel svm_model. Baris kode ke-6 melatih model SVM pada data *training*. Seperti model *Naïve Bayes,* method fit() juga dipergunakan untuk melatih *classifier* ini sendiri. Dimana X_train_resampled_imb merepresentasikan fitur untuk *training*, sementara y_train_resampled_imb merepresentasikan taret label yang dimaksud.

4.7 Hasil Evaluasi Model

Setelah dilakukan pelatihan terhadap model dengan menggunakan dua metode klasifikasi yaitu *Naïve Bayes* dan *Support Vector Machine* dengan memanfaatkan data latih, selanjutnya adalah proses evaluasi terhadap performa model. menggunakan data uji yang telah dibagi rasionya sebanyak 20% dari keseluruhan *dataset*. Pertama-tama, dilakukan uji performa terhadap model *Naïve Bayes* dan SVM dengan SMOTE. Dalam berkas kode penelitian ini, dilakukan pemisahan terhadap kedua file *jupyter notebook* yang menerapkan *oversampling* dengan SMOTE dan yang tidak menggunakan SMOTE. Alasannya adalah untuk menghindari biasnya hasil yang didapat karena percampuran penggunaan model. Kode pemrograman untuk proses pengujian performa antara *Naïve Bayes* dan SVM ini ditunjukkan pada Gambar 4.33 dan Gambar 4.34.

```
1. from sklearn.metrics import accuracy_score, precision_score,
   recall_score, f1_score
2. from sklearn.metrics import confusion_matrix as
   confusion_matrix_imb
3. import time
4. start_time_imb = time.time()
5. y_pred_nb_imb  =  naive_bayes_model_imb.predict(X_test_tfidf)
6. end_time_imb = time.time()
7. prediction_time_nb_imb = end_time_imb - start_time_imb
8. accuracy_nb_imb  =  accuracy_score(y_test,  y_pred_nb_imb)
9. precision_nb_imb = precision_score(y_test, y_pred_nb_imb,
   average='weighted')
10. recall_nb_imb = recall_score(y_test, y_pred_nb_imb,
   average='weighted')
11. #Membuat Confusion Matrix
12. f1_nb_imb = f1_score(y_test, y_pred_nb_imb, average='weighted')
13. nb_y_pred_imb_imb  =  naive_bayes_model_imb.predict(X_test_tfidf)
14.nb_confusion_matrix_imb = confusion_matrix_imb(y_test,
   nb_y_pred_imb_imb)
```

Gambar 4. 33 Kode Pemrograman Pengujian Performa *Naïve Bayes*

Baris kode ke-1 pada kode ini mengimpor metrik evaluasi dari modul sklearn.metrics. Baris kode ke-2 mengimpor fungsi confusion_matrix dari sklearn.metrics dan memberikan alias sebagai confusion_matrix_imb. Baris kode ke-3 mengimpor modul time untuk melacak waktu eksekusi. Baris ke-4 merekam waktu awal dalam mengukur waktu yang dibutuhkan pada saat melakukan prediksi. Baris ke-5 memprediksi label dengan menggunakan model *Naïve Bayes* dalam *dataset* uji yang direpresentasikan sebagai X_test_tfidf.

Baris ke-6 merekam waktu selesainya model beroperasi. Baris ke-7 meghitung waktu yang dihabiskan untuk melakukan prediksi yang disimpan dalam variabel prediction_time_nb. Baris kode ke-8 menghitung akurasi dari prediksi *Naïve Bayes* (y_pred_nb_imb) dengan membandingkannya dengan label sebenarnya y_test. Baris kode ke-9 menghitung skor presisi terbobot. Baris kode ke-10 menghitung nilai *recall* terbobot.

Baris kode ke-12 menghitung skor f1 terbobot. Baris kode ke-13 melakukan prediksi lagi dan menyimpannya ke dalam nb_y_pred_imb_imb untuk kemudian dibuat *confusion matrix*nya. Baris kode ke-14 menghitung *confusion matrix* menggunakan label sesungguhnya dan prediksi *Naïve Bayes*.

```
1. import time
2. start_time = time.time()
3. y_pred_svm = svm_model.predict(X_test_tfidf)
4. end_time = time.time()
5. prediction_time_svm = end_time - start_time
6. accuracy_svm = accuracy_score(y_test, y_pred_svm)
7. precision_svm = precision_score(y_test, y_pred_svm,
   average='weighted')
8. recall_svm = recall_score(y_test, y_pred_svm, average='weighted')
9. f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
10. svm_y_pred_imb = svm_model.predict(X_test_tfidf)
11.svm_confusion_matrix = confusion_matrix(y_test, svm_y_pred_imb)
```

Gambar 4. 34 Kode Pemrograman Pengujian Performa SVM

Baris kode ke-1 mengimpor modul time untuk melacak waktu eksekusi model SVM. Baris ke-2 merekam waktu awal dalam mengukur waktu yang dibutuhkan pada saat melakukan prediksi. Baris ke-3 memprediksi label dengan menggunakan model *Support Vector Machine* dalam *dataset* uji yang direpresentasikan sebagai X_test_tfidf.

Baris ke-4 merekam waktu selesainya model beroperasi. Baris ke-5 meghitung waktu yang dihabiskan untuk melakukan prediksi yang disimpan dalam variabel prediction_time_svm. Baris kode ke-6 menghitung akurasi dari prediksi *Support Vector Machine* (y_pred_svm)

dengan membandingkannya dengan label sebenarnya y_test. Baris kode ke-7 menghitung skor presisi terbobot. Baris kode ke-8 menghitung nilai *recall* terbobot.

Baris kode ke-9 menghitung skor f1 terbobot. Baris kode ke-10 melakukan prediksi lagi dan menyimpannya ke dalam svm_y_pred_imb untuk mengetahui nilai *confusion matrixnya*. Baris kode ke-13 menghitung *confusion matrix* menggunakan label sesungguhnya dan prediksi *Support Vector Machine*.

Pengujian model di atas menghasilkan nilai performa kedua model yang dijadikan perbandingan. Hasil pengujian kedua model yaitu *Naïve Bayes* dan SVM ditunjukkan dalam Gambar 4. 35 dan Gambar 4. 36. Kedua hasil pengujian tersebut dibandingkan dengan performa model saat data tidak dilakukan *Imbalanced Data Handling*. Pada kondisi ini, kelas label masing-masing data sangat timpang dan tidak diseimbangkan jumlahnya.

```
Naive Bayes:
Accuracy: 0.9147424511545293
Precision: 0.9231014310990135
Recall: 0.9147424511545293
F1-Score: 0.9177719244891562

SVM:
Accuracy: 0.911190053285968
Precision: 0.902397497747305
Recall: 0.911190053285968
F1-Score: 0.9029946547679536
```

Gambar 4. 35 Hasil Pengujian Terhadap Performa Model *Naïve* Bayes dan SVM dengan SMOTE dalam Melakukan Klasifikasi

```
Naive Bayes:
Accuracy: 0.9023090586145648
Precision: 0.908868996605275
Recall: 0.9023090586145648
F1-Score: 0.8810245843096547

SVM:
Accuracy: 0.9094138543516874
Precision: 0.905812237068854
Recall: 0.9094138543516874
F1-Score: 0.8956279349281182
```

Gambar 4. 36 Hasil Pengujian Terhadap Performa Model *Naïve* Bayes dan SVM tanpa SMOTE dalam Melakukan Klasifikasi

Berdasarkan Gambar 4. 35 dan 4. 36, kedua metode klasifikasi yaitu *Naïve Bayes* dan *Support Vector Machine* mendapatkan nilai akurasi yang sangat baik, yaitu mencapai 90%. Dapat juga dilihat peningkatan sebesar 1% pada nilai akurasi model *Naïve Bayes* ketika menerapkan SMOTE untuk menangani data yang tidak seimbang yaitu dari 90% meningkat hingga 91%. Model SVM juga mengalami peningkatan serupa dalam nilai akurasinya, yaitu

sebesar 0,2% yang asalnya 90,9% mengalami peningkatan menjadi 91,1% ketika menggunakan SMOTE.

Aspek-aspek lain juga mengalami peningkatan ketika menerapkan SMOTE, seperti *precision* pada *Naïve Bayes* yang meningkat dari 90% menjadi 92%. Aspek *recall* dan *f1-score* juga mengalami peningkatan pada masing-masing metode klasifikasi, yaitu nilai *recall* pada *Naïve Bayes* meningkat dari 90,2% menjadi 91,4%, nilai *f1-score* pada *Naïve Bayes* meningkat dari 88,1% menjadi 91,7%.

Peningkatan nilai *recall* dan *f1-score* juga terjadi pada *classifier* SVM, untuk *recall* meningkat dari 90,9% menjadi 91,1%. Nilai *f1-score* meningkat dari 89,5% menjadi 90,2%. Di antara aspek yang mengalami kenaikan saat diterapkan *oversampling* dengan SMOTE, terdapat aspek yang mengalami penurunan ketika menerapkan SMOTE. Aspek tersebut adalah *precision* pada SVM mengalami penurunan dari 90,5% menjadi 90,2%. Pada Gambar 4. 37 ditunjukkan waktu pengujian Metode *Naïve Bayes* dan Gambar 4. 38 ditunjukkan waktu pengujian Metode *Support Vector Machine*.

```
Naive Bayes Prediction time: 0.00 seconds
```

Gambar 4. 37 Hasil Waktu Pengujian Metode Naïve Bayes

```
SVM Prediction time: 0.06 seconds
```

Gambar 4. 38 Hasil Waktu Pengujian Metode Support Vector *Machine*

Berdasarkan waktu pengujian kedua model dalam melakukan prediksi terhadap data uji, *Naïve Bayes* terlihat lebih unggul daripada *Support Vector Machine*. Hal ini disebabkan waktu prediksi yang dibutuhkan oleh model SVM lebih lama 0.06 detik dibandingkan *Naïve Bayes* yang menjalankan proses pengujian ini secara instan yaitu 0.00 detik. Lamanya waktu pengujian dari setiap modelnya bergantung pada banyaknya data yang digunakan sebagai data uji. Pada penelitian ini, data uji yang digunakan hanya sebanyak 583 data.

Selanjutnya adalah pembuatan *Confusion Matrix.* Proses ini dilakukan untuk mengetahui jumlah dari pengelompokkan data uji antara data yang benar dengan data yang salah. Kode Pemrograman pembuatan *Confusion Matrix* dapat dilihat pada Gambar 4. 39.

```
1. nb_y_pred_imb = naive_bayes_model.predict(X_test_tfidf)
2. nb_confusion_matrix = confusion_matrix(y_test, nb_y_pred_imb)
3. print("Confusion Matrix - Naive Bayes:")
4. print(nb_confusion_matrix_imb)
5. svm_y_pred_imb = svm_model.predict(X_test_tfidf)
6. svm_confusion_matrix = confusion_matrix(y_test, svm_y_pred_imb)
7. print("Confusion Matrix - SVM:")
8. print(svm_confusion_matrix_imb)
```

Gambar 4. 39 Kode Pemrograman Confusion Matrix.

Baris kode ke-1 memprediksi nilai target dengan menggunakan *Naïve Bayes* pada data uji X_test. Nilai prediksinya disimpan dalam variabel nb_y_pred_imb. Pada baris kode ke-2, *confusion matrix* dikomputasikan untuk model *Naïve Bayes*. Fungsi confusion_matrix mengambil label sebenarnya y_test dan label terprediksi nb_y_pred_imb lalu menghitung *confusion matrix* dari label tersebut. Hasil *confusion matrix* ini disimpan dalam variabel nb_confusion_matrix. Baris kode ke-3 mencetak *header* untuk output *confusion matrix Naïve Bayes*.

Baris kode ke-4 mencetak nilai *confusion matrix* untuk model *Naïve Bayes*. Baris kode ke-5 memprediksi nilai target dengan menggunakan model SVM pada data uji, dan prediksinya disimpan dalam svm_y_pred_imb. Baris kode ke-6 membuat *confusion matrix* untuk model SVM dengan menggunakan pendekatan serupa dengan baris kode ke-2. Baris kode ke-7 mencetak *header* untuk output *confusion matrix* SVM. Baris kode ke-8 nilai *confusion matrix* untuk model SVM.

Proses evaluasi terhadap model yang telah dibangun dijalankan ketika proses pelatihan terhadap model sudah selesai dilakukan. Tujuan dari proses evaluasi ini sendiri adalah guna melakukan penghitungan terhadap performa masing-masing metode yang digunakan dalam proses klasifikasi, khususnya *Naïve Bayes* dan *Support Vector Machine* sebagai metode yang dipergunakan dalam penelitian ini. Hasil *Confusion Matrix* yang disajikan dalam bentuk tabel berukuran 3x3 dan juga menggunakan SMOTE dapat dilihat dalam Tabel 4.8 dan 4.9. dan *Confusion Matrix* tanpa SMOTE ditunjukkan pada Tabel 4. 10 serta 4. 11.

Tabel 4. 8 Confusion Matrix Naïve Bayes Classifier dengan SMOTE

	Predicted Label		
True Label	Positif	Netral	Negatif
Positif	429	11	13
Netral	8	58	8

Negatif	6	2	28

Tabel 4. 9 Confusion Matrix Support Vector *Machine* dengan SMOTE

	Predicted Label		
True Label	Positif	Netral	Negatif
Positif	444	5	4
Netral	15	56	3
Negatif	20	3	13

Tabel 4. 10 Confusion Matrix Naïve Bayes Classifier tanpa SMOTE

	Predicted Label		
True Label	Positif	Netral	Negatif
Positif	450	3	0
Netral	21	53	0
Negatif	30	1	5

Tabel 4. 11 Confusion Matrix Support Vector *Machine* tanpa SMOTE

	Predicted Label		
True Label	Positif	Netral	Negatif
Positif	451	2	0
Netral	20	51	3
Negatif	26	0	10

4.8 Wordcloud

Setelah diketahui nilai perbandingan performa dari kedua model melalui proses evaluasi model, dilakukan analisis untuk mengetahui kata apa yang banyak diperbincangkan oleh masyarakat mengenai kinerja POLRI. Pengumpulan data dilakukan pada 2 periode waktu, yaitu pada tanggal 28 September - 6 Oktober 2022 dan 11 - 20 Desember 2022. Maka dari itu, data *tweet* yang menjadi input merupakan topik yang sedang hangat dibahas pada periode tersebut.

Untuk proses pembuatan *wordcloud* ini, digunakan data latih yang pemrosesannya hanya sampai pada proses *Remove Stopwords*, dengan tujuan agar kata-kata yang dimunculkan tidak diubah ke dalam bentuk aslinya. Sehingga, maksud dari suatu kata masuk ke dalam kelas sentimen dapat dipahami. Kode pemrograman untuk memisahkan data latih dan data uji yang hanya akan dipakai dalam *wordcloud* ditunjukkan pada Gambar 4. 40.

```
1. XX = tweet_polridf['Sudah Remove Stopwords']
2. yy = tweet_polridf['Label']
3. XX_train, XX_test, yy_train, yy_test = train_test_split(XX, yy,
   test_size=0.2, random_state=42)
4. joined_train_wc = [' '.join(tokens) for tokens in XX_train]
5. joined_test_wc = [' '.join(tokens) for tokens in XX_test]
```

Gambar 4. 40 Kode Pemrograman Pemisahan Data Latih dan Data Uji untuk *Wordcloud*.

Baris kode ke-1 melakukan penyematan data teks yang sudah dihapus *stopwords*-nya ke dalam variabel XX. Baris kode ke-2 menyematkan label data ke dalam variabel yy. Baris kode ke-3 memisahkan antara data latih dengandata uji. Rasio pemisahannya adalah 80% untuk data latih dan 20% untuk data uji. Digunakan random_state untuk memastikan pemisahan data yang konsisten setiap kode dijalankan. Baris ke-4 mengobinasikan kata dalam data latih ke dalam bentuk *string* tunggal. Baris ke-4 mengobinasikan kata dalam data uji ke dalam bentuk *string* tunggal. Untuk kode pemrograman pembuatan *wordcloud*-nya terdapat pada Gambar 4. 41.

```
1. from wordcloud import WordCloud
2. import matplotlib.pyplot as plt
3. text = ' '.join(joined_train_wc)
4. wordcloud = WordCloud(width=1920, height=1080,
   background_color='white', max_words=300,
   collocations=False).generate(text)
5. plt.figure(figsize=(10, 5))
6. plt.imshow(wordcloud, interpolation='bilinear')
7. plt.axis('off')
8. plt.show()
```

Gambar 4. 41 Kode Pemrograman Pembuatan *Wordcloud*.

Baris kode ke-1 melakukan impor kelas WordCloud dari *library* wordcloud. Baris kode ke-2 mengimpor modul pyplot dengan alias plt dari *library matplotlib* untuk membuat plot dan visual data. Baris kode ke-3 mengombinasikan data teks dari joined_train_wc. Isi dari variabel joined_train_wc sendiri merupakan data *string* yang berasal dari penggabungan data berbentuk token. Baris kode ke-4 membuat sebuah WordCloud dengan pengaturan sebagai berikut: width dan height mengatur dimensi dari gambar hasil output. Background_color menentukan warna

latar dari *wordcloud* tersebut. Max_words membatasi jumlah huruf yang ditampilkan pada *wordcloud*, collocations di-set ke False untuk membuat *wordcloud* menjadi terpisah setiap katanya. Fungsi generate() memproses data teks yang diberikan untuk membuat *wordcloud*.

Baris kode ke-5 menyetel ukuran figure plot dengan menggunakan plt.figure(figsize=(10, 5)). Baris kode ke-6 menampilkan gambar *wordcloud* dengan menggunakan plt.imshow(wordcloud, interpolation='bilinear'), dengan interpolasi "bilinear". Baris kode ke-7 mematikan tampilan sumbu dengan plt.axis('off'). Baris kode ke-8 menampilkan plot *wordcloud* menggunakan plt.show(). Berikut hasil *wordcloud* untuk keseluruhan kelas sentimen data latih yang ditunjukkan pada Gambar 4. 42 dan persentase kemunculan kata dalam *wordcloud* untuk keseluruhan kelas data latih ditunjukkan pada Gambar 4. 43.

Gambar 4. 42 *Wordcloud* untuk Keseluruhan Kelas Data Latih.

```
10 Kata Teratas dalam Keseluruhan Sentimen pada Data Latih:
1. polri: 10.37%
2. kinerja: 10.07%
3. maksimal: 3.64%
4. masyarakat: 1.66%
5. tuntas: 1.61%
6. apresiasi: 1.60%
7. kapolri: 1.49%
8. kejahatan: 1.26%
9. mengapresiasi: 1.26%
10. mengusut: 1.21%
```

Gambar 4. 43 Persentase Kemunculan Kata pada Data Latih

Dapat dilihat pada gambar di atas bahwa kata yang paling sering muncul dalam pembicaraan masyarakat di media sosial Twitter mengenai kinerja POLRI adalah polri, kinerja, maksimal, masyarakat, tuntas, apresiasi, kapolri, kejahatan, apresiasi, dan mengusut. Persentase kemunculan tertinggi ada pada kata "polri" dengan persentase 10.37%. Selain menampilkan data dari keseluruhan sentimen, turut ditampilkan juga visualisasi data dari masing-masing kelas sentimen. Kode Pemrograman untuk membuat *wordcloud* sesuai dengan kelasnya ditunjukkan pada Gambar 4. 44.

```
1. positive_text = ' '.join([' '.join(text_wc) for text_wc, label in
   zip(XX_train, yy_train) if label == 1])
2. neutral_text = ' '.join([' '.join(text_wc) for text_wc, label in
   zip(XX_train, yy_train) if label == 2])
3. negative_text = ' '.join([' '.join(text_wc) for text_wc, label in
   zip(XX_train, yy_train) if label == 3])
```

```
4. positive_wordcloud = WordCloud(width=1920, height=1080,
   random_state=42, background_color='white', max_words=300,
   collocations=False).generate(positive_text)
5. neutral_wordcloud = WordCloud(width=1920, height=1080,
   random_state=42, background_color='white', max_words=300,
   collocations=False).generate(neutral_text)
6. negative_wordcloud = WordCloud(width=1920, height=1080,
   random_state=42, background_color='white', max_words=300,
   collocations=False).generate(negative_text)
7. plt.figure(figsize=(10, 5))
8. plt.imshow(positive_wordcloud, interpolation='bilinear')
9. plt.title('Positive Sentiment')
10. plt.axis('off')
11. plt.show()
12. plt.figure(figsize=(10, 5))
13. plt.imshow(neutral_wordcloud, interpolation='bilinear')
14. plt.title('Neutral Sentiment')
15. plt.axis('off')
16. plt.show()
17. plt.figure(figsize=(10, 5))
18. plt.imshow(negative_wordcloud, interpolation='bilinear')
19. plt.title('Negative Sentiment')
20. plt.axis('off')
21.plt.show()
```

Gambar 4. 44 Kode Pemrograman untuk Menampilkan Keseluruhan Kelas.

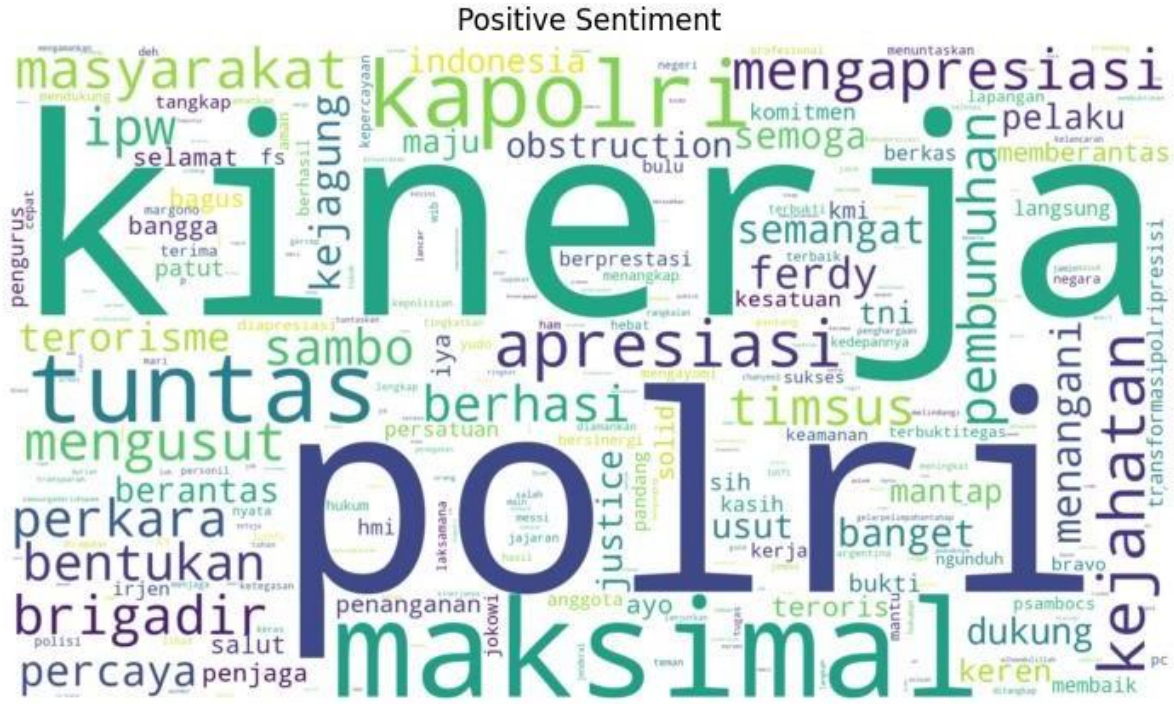
Baris kode ke-1 menampilkan variabel positive_text yang berisikan daftar sampel teks positif yang digabungkan. Baris kode ini menyaring data pada XX_train dan yy_train serta hanya memilih sampel teks yang memiliki label 1, yaitu sentimen positif. Setelah itu, baris kode ini menyatukan kata-kata yang terdapat pada data sampel teks menjadi sebuah *string* yang dipisahkan oleh spasi. Baris kode ke-2 memiliki fungsi yang sama seperti baris kode ke-1, namun baris kode ini memilih sampel teks yang memiliki label 2, yaitu sentimen netral. Baris kode ke-3 memiliki fungsi yang sama seperti baris kode di atasnya, yaitu memilih sampel teks yang memiliki label 3, yaitu sentimen negatif.

Baris kode ke-4 melakukan pembuatan *wordcloud* untuk sentimen positif sekaligus mengatur detailnya, seperti width, height, random_state, dan background_color. Baris kode ke-5 membuat *wordcloud* data sentimen netral sekaligus mengatur detailnya, persis seperti baris kode ke-4. Baris kode ke-6 membuat *wordcloud* untuk sentimen data negatif serta melakukan pengaturan yang sama seperti pembuatan *wordcloud* positif dan netral.

Baris kode ke-7 mengatur ukuran figur *wordcloud* yang akan dimunculkan. Baris kode ke-8 menampilkan gambar *wordcloud* dengan menggunakan plt.imshow(wordcloud,

interpolation='bilinear'), dengan interpolasi “bilinear”. Baris kode ke-9 menuliskan judul *wordcloud* yaitu ‘Positive Sentiment’, yang berarti urutan plot *wordcloud* yang pertama ini akan menunjukkan *wordcloud* berisi data yang memiliki sentimen positif. Baris kode ke-10 mematikan tampilan sumbu dengan plt.axis(‘off’). Baris kode ke-11 menampilkan plot *wordcloud* menggunakan plt.show().

Baris kode ke-12 sampai dengan 16 berfungsi untuk menampilkan *wordcloud* yang berisi sentimen data netral, dengan pengaturan yang sama persis dilakukan seperti pada plot *wordcloud* sentimen positif. Baris kode ke-17 hingga 21 berfungsi untuk menampilkan *wordcloud* yang berisikan sentimen data negatif, dengan pengaturan yang sama seperti kedua sentimen data sebelumnya. Adapun untuk *wordcloud* setiap kelas sentimen ditunjukkan pada beberapa gambar di bawah. Gambar pertama yaitu *wordcloud* sentimen positif, ditunjukkan pada Gambar 4. 45 dan persentase kemunculan katanya ditunjukkan pada Gambar 4. 46.



Gambar 4. 45 *Wordcloud* untuk Kelas Sentimen Positif.

```
10 Kata Teratas dalam Sentimen Positif pada Data Latih:
1. kinerja: 10.96%
2. polri: 10.93%
3. maksimal: 4.59%
4. tuntas: 2.02%
5. kapolri: 1.80%
6. apresiasi: 1.62%
7. kejahatan: 1.60%
8. mengapresiasi: 1.58%
9. masyarakat: 1.55%
10. mengusut: 1.53%
```

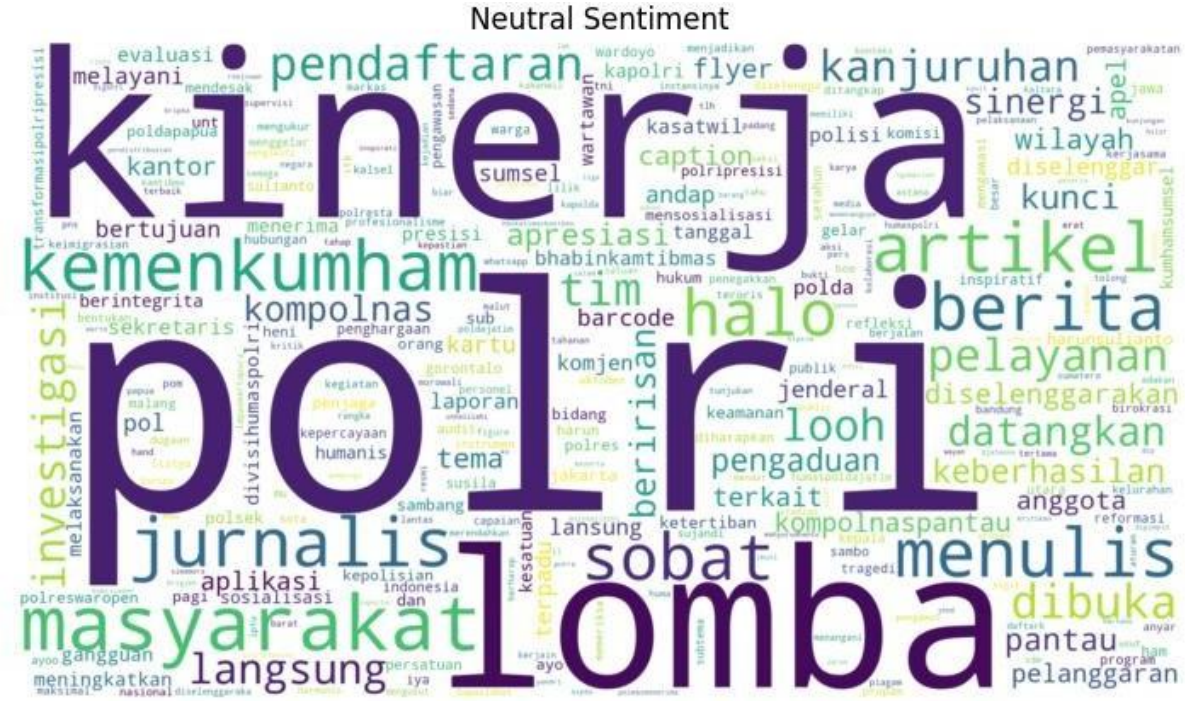
Gambar 4. 46 Persentase Kemunculan Kata dalam Sentimen Positif pada Data Latih

Dalam visualisasi data *wordcloud* untuk sentimen positif, terdapat banyak sekali kata yang mengandung nuansa positif terhadap kinerja POLRI. Ditunjukkan pada gambar *wordcloud* 4. 45 bahwa kata-kata berikut sering muncul dalam sentimen positif, di antaranya kinerja, polri, maksimal, tuntas, kapolri, apresiasi, kejahatan, mengapresiasi, masyarakat, dan mengusut. Terdapat pula kata yang seharusnya tidak termasuk ke dalam sentimen positif, yaitu "pembunuhan", "terorisme", "teroris" seperti yang terlihat pada di bawah kata "mengapresiasi".

Hal ini dapat terjadi karena kata-kata tersebut terkandung dalam *tweet* yang memiliki sentimen positif. Contoh *tweet* yang sudah dihapus *stopwords*-nya adalah seperti ini:

“ipw mengapresiasi kinerja timsus bentukan kapolri berhasi mengusut tuntas perkara pembunuhan brigadir”, “dukung berantas terorisme polri kinerja maksimal semangat polri”, dan “bantu polri kinerja maksimal memerangi teroris negeri”. *Tweet* tersebut dilabeli ke dalam kelas positif karena bermaksud mengapresiasi kinerja polri dalam memberantas kejahatan, contohnya adalah pembunuhan dan terorisme.

Persentase kemunculan tertinggi ada pada kata “kinerja” dengan persentase 10.96%. Kelas sentimen kedua yang akan ditunjukkan adalah netral. Gambar kedua yaitu *wordcloud* sentimen netral, ditunjukkan pada Gambar 4. 47 dan persentase kemunculan katanya pada Gambar 4. 48.



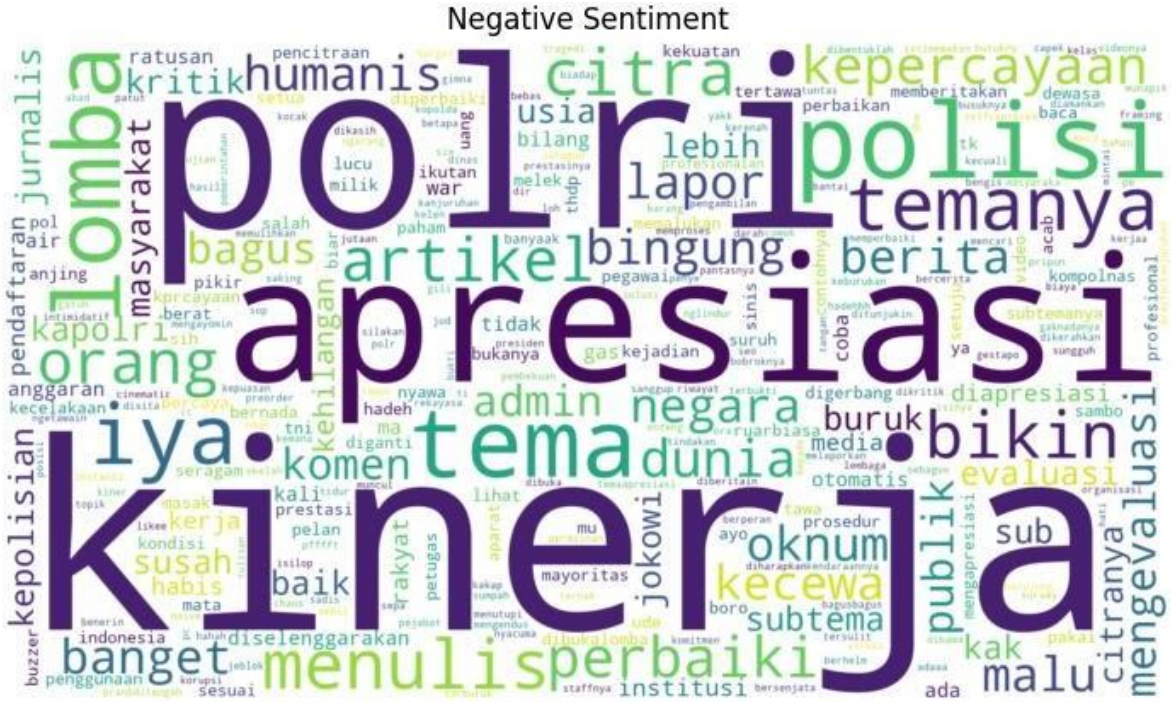
Gambar 4. 47 *Wordcloud* untuk Kelas Sentimen Netral.

```
10 Kata Teratas dalam Sentimen Netral pada Data Latih:
1. polri: 8.34%
2. kerja: 5.66%
3. lomba: 3.82%
4. masyarakat: 2.71%
5. kemenkumham: 2.11%
6. artikel: 2.02%
7. jurnalis: 1.99%
8. tulis: 1.99%
9. layan: 1.93%
10. berita: 1.84%
```

Gambar 4. 48 Persentase Kemunculan Kata dalam Sentimen Netral pada Data Latih

Pada pemetaan visual data *wordcloud* untuk sentimen netral, terdapat banyak kata-kata yang tidak merujuk ke sentimen manapun. Data *tweet* yang diklasifikasi pada sentimen ini pada umumnya hanya memberikan informasi umum mengenai POLRI. Kata yang sering muncul

pada sentimen ini yaitu polri, kerja, lomba, masyarakat, kemenkumham, artikel, jurnalis, tulis, layan, berita. Kata "polri" menjadi yang paling sering muncul dalam kelas sentimen netral pada data latih, yaitu dengan persentase kemunculan 8.34%. Kelas sentimen terakhir yang akan ditunjukkan pada gambar di bawah adalah negatif. Gambar ketiga yaitu *wordcloud* sentimen negatif, ditunjukkan pada Gambar 4. 49 dan persentase kemunculannya pada Gambar 4. 50.



Gambar 4. 49 *Wordcloud* untuk Kelas Sentimen Negatif.

```
10 Kata Teratas dalam Sentimen Negatif pada Data Latih:
1. kinerja: 10.32%
2. polri: 8.21%
3. apresiasi: 4.01%
4. polisi: 1.72%
5. tema: 1.43%
6. lomba: 1.34%
7. iya: 1.05%
8. temanya: 1.05%
9. menulis: 0.86%
10. bikin: 0.76%
```

Gambar 4. 50 Persentase Kemunculan Kata dalam Sentimen Negatif pada Data Latih

Dalam visualisasi data latih yang memiliki sentimen negatif dalam *wordcloud*, terdapat bermacam kata negatif. Data *tweet* yang diklasifikasikan ke dalam sentimen ini merupakan kata yang terkandung dalam *tweet* yang memiliki tendensi negatif terhadap kinerja POLRI. Kata yang paling sering muncul dalam *wordcloud* negatif ini beberapa contohnya kinerja, polri,

apresiasi, polisi, tema, lomba, iya, temanya, menulis bikin. Kata paling sering muncul dalam *wordcloud* berisi sentimen negatif pada data latih adalah kinerja, dengan persentase 10.32%.

Alasan dibalik kata-kata yang termasuk ke sentimen lain juga terkandung dalam sentimen negatif seperti kerja, polri, apresiasi adalah pada saat proses pelabelan data, *tweet* yang dilabeli negatif mengandung kata yang sama dengan *tweet* positif. Label sentimen *tweet* yang timpang juga mempengaruhi hal ini. Dikarenakan dalam hasil pelabelan *tweet* positif lebih banyak jumlahnya dibandingkan dengan sentimen netral atau negatif. Contoh dari *tweet* tersebut adalah "@maxwalden_ @Eno_Bening Najis tema nya apresiasi kinerja polri. Kocak". Pada *tweet* ini, terdapat kata "apresiasi", "kinerja", dan "polri". Data *tweet* selanjutnya juga mengandung kata-kata positif meskipun *tweet* ini memiliki label negatif. Contohnya adalah "@DivHumas_Polri bikin lomba menulis tpi judulnya tentang apresiasi kinerja polisi. eallaah si pengen diapresiasii". *Tweet* ini mengandung kata "apresiasi", "kinerja", dan "polisi". Selain dari *wordcloud* yang telah dipaparkan di atas, dilakukan juga pembuatan *wordcloud* yang berisikan data uji. *Wordcloud* ini dibuat dengan tujuan untuk memperlihatkan kata apa saja yang paling sering muncul dalam data uji. Kode pemrograman untuk memperlihatkan *wordcloud* berisikan data uji ini dapat dilihat pada Gambar 4. 51.

```
1. text_test = ' '.join(joined_test_wc)
2. wordcloud = WordCloud(width=1920, heiBght=1080,
   background_color='white', max_words=300,
   collocations=False).generate(text_test)
3. plt.figure(figsize=(10, 5))
4. plt.imshow(wordcloud, interpolation='bilinear')
5. plt.axis('off')
6. plt.show()
```

Gambar 4. 51 Kode Pemrograman Pembuatan *Wordcloud* pada Data Uji.

Baris kode ke-1 mengombinasikan isi dari *list* joined_test_wc ke dalam sebuah string tunggal bernama text_test, dimana huruf dipisahkan dengan spasi. Baris ke-2 berisikan kode untuk membuat sebuah objek *wordcloud* bernama Wordcloud berikut dengan pengaturannya. Hal yang diatura adalah lebar, tinggi, warna latar belakang, serta jumlah huruf maksimal yang ditampilkan pada *wordcloud*. Fungsi generate() memproses text_test untuk membuat *wordcloud*. Baris kode ke-3 menyiapkan figur untuk plot menggunakan matplotlib, dengan mengkhususkan ukurannya menjadi 10 unit lebar, dan 5 unit untuk tingginya.

Baris kode ke-4 menampilkan gambar *wordcloud* yang telah dibuat menggunakan fungsi imshow(). Objek wordcloud berisikan data *wordcloud*, dan *method* interpolasi 'bilinear' digunakan untuk meningkatkan kualitas gambar *wordcloud*. Baris kode ke-5 meniadakan label

sumbu dan *ticks*, untuk menyediakan output visual tanpa gangguan yang tidak perlu. Baris kode ke-6 menampilkan visualisasi *wordcloud* yang telah diplot ke dalam sebuah jendela grafis. *Wordcloud*-nya sendiri dapat dilihat pada Gambar 4. 52 dan persentase kemunculan kata dalam *wordcloud* untuk keseluruhan data uji dapat dilihat pada Gambar 4. 53.

Gambar 4. 52 *Wordcloud* untuk Keseluruhan Data Uji.

```
10 Kata Teratas dalam Keseluruhan Sentimen pada Data Uji:
1. polri: 9.94%
2. kinerja: 9.67%
3. maksimal: 3.59%
4. tuntas: 1.77%
5. apresiasi: 1.75%
6. masyarakat: 1.73%
7. kapolri: 1.41%
8. mengapresiasi: 1.31%
9. mengusut: 1.25%
10. bentukan: 1.24%
```

Gambar 4. 53 Persentase Kemunculan Kata pada Data Uji.

Seperti yang terlihat pada kedua gambar di atas, kata yang frekuensi kemunculannya paling sering dalam keseluruhan data uji adalah polri, kinerja, maksimal, tuntas, apresiasi, masyarakat, kapolri, mengapresiasi, mengusut, dan bentukan. Persentase tertinggi terdapat pada kata "polri" dengan jumlah persentase kemunculan sebesar 9.94% diantara keseluruhan data uji. Setelah *wordcloud* untuk menampilkan data uji selesai dibuat, selanjutnya adalah pembuatan *wordcloud* berdasarkan hasil klasifikasi dari model prediksi *Naïve Bayes* dan *Support Vector Machine*. Hal ini ditujukan untuk mengetahui kata apa saja yang paling sering muncul dalam

hasil prediksi *Naïve Bayes* dan SVM dalam mengklasifikasi data *tweet* ke dalam masing-masing kelas sentimennya. Berikut ditampilkan kode pemrograman untuk menampilkan *wordcloud* kedua hasil prediksi pada Gambar 4. 54.

```
positive_text_nb = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_nb_imb) if label == 1])
neutral_text_nb = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_nb_imb) if label == 2])
negative_text_nb = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_nb_imb) if label == 3])
positive_text_svm = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_svm_imb) if label == 1])
neutral_text_svm = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_svm_imb) if label == 2])
negative_text_svm = ' '.join([' '.join(text_test) for text_test,
label in zip(X_test, y_pred_svm_imb) if label == 3])
positive_wordcloud_nb = WordCloud(width=1920, height=1080,
random_state=42, background_color='white',
max_words=150).generate(positive_text_nb)
neutral_wordcloud_nb = WordCloud(width=1920, height=1080,
random_state=42, background_color='white', max_words=150,
collocations=False).generate(neutral_text_nb)
negative_wordcloud_nb = WordCloud(width=1920, height=1080,
random_state=42, background_color='white', max_words=150,
collocations=False).generate(negative_text_nb)
positive_wordcloud_svm = WordCloud(width=1920, height=1080,
random_state=42, background_color='white', max_words=150,
collocations=False).generate(positive_text_svm)
neutral_wordcloud_svm = WordCloud(width=1920, height=1080,
random_state=42, background_color='white', max_words=150,
collocations=False).generate(neutral_text_svm)
negative_wordcloud_svm = WordCloud(width=1920, height=1080,
random_state=42, background_color='white', max_words=150,
collocations=False).generate(negative_text_svm)
plt.figure(figsize=(10, 5))
plt.imshow(positive_wordcloud_nb, interpolation='bilinear')
plt.title('Positive Sentiment')
plt.axis('off')
plt.show()
plt.figure(figsize=(10, 5))
plt.imshow(neutral_wordcloud_nb, interpolation='bilinear')
plt.title('Neutral Sentiment')
plt.axis('off')
plt.show()
plt.figure(figsize=(10, 5))
plt.imshow(negative_wordcloud_nb, interpolation='bilinear')
plt.title('Negative Sentiment')
```

```
26. plt.axis('off')
27. plt.show()
28. plt.figure(figsize=(10, 5))
29. plt.imshow(positive_wordcloud_svm, interpolation='bilinear')
30. plt.title('Positive Sentiment')
31. plt.axis('off')
32. plt.show()
33. plt.figure(figsize=(10, 5))
34. plt.imshow(neutral_wordcloud_svm, interpolation='bilinear')
35. plt.title('Neutral Sentiment')
36. plt.axis('off')
37. plt.show()
38. plt.figure(figsize=(10, 5))
39. plt.imshow(negative_wordcloud_svm, interpolation='bilinear')
40. plt.title('Negative Sentiment')
41. plt.axis('off')
42.plt.show()
```

Gambar 4. 54 Kode Pemrograman Pembuatan *Wordcloud* pada Hasil Prediksi *Naïve Bayes* dan SVM

Baris kode ke-1 membuat sebuah *string* yang berisikan seluruh sampel teks bersentimen positif dari X_test, dimana model *Naïve Bayes* (y_pred_nb_imb) memprediksi label dengan angka 1 (positif). Sampel teks disatukan dengan spasi. Baris kode ke-2 mirip dengan baris ke-1, yaitu membuat *string* yang berisikan sampel teks dengan sentimen netral dan diprediksi dengan label 2 (netral). Baris kode ke-3 membuat sebuah *string* yang berisikan sampel teks dengan sentimen negatif yang diprediksi dengan label 3 (negatif).

Baris kode ke-4 hingga baris kode ke-6 keseluruhannya berfungsi sama seperti baris 1 hingga baris 3 secara berurutan, yaitu dengan memprediksi teks dengan label positif, netral, dan negatif. Yang membedakan adalah baris ke-4 dan ke-6 ini memprediksi data sampel teks dengan algoritma SVM (y_pred_svm_imb). Selanjutnya baris kode ke-7 hingga ke-12 membuat objek *wordcloud* untuk setiap kategori senimen dengan menggunakan *library WordCloud*. Setiap objek *wordcloud* di-set dengan pengaturan spesifik seperti lebar, tinggi, random_state, warna latar belakang, dan jumlah maksimal kata yang akan ditampilkan.

Baris kode ke-13 mengatur ukuran figur pada plot dengan menggunakan matplotlib.pyplot. Baris kode ke-14 hingga 27 menampilkan *wordcloud* untuk sentimen positif, netral, dan negative dari hasil prediksi algoritma *Naïve bayes*. Setiap *wordcloud* ditampilkan dengan menggunakan plt.imshow() dengan interpolasi yang spesifik, lalu diikuti dengan mengatur judul *wordcloud* an menghapus plot sumbunya. Setelah itu, plt.show() dipanggil untuk

menampilkan plotnya. Baris kode ke-28 hingga 42 berfungsi mirip dengan baris ke-14 dan 27. Baris kode ini menampilkan *wordcloud* untuk sentimen yang diprediksi oleh SVM secara berurutan; positif, netral, dan negatif. *Wordcloud* yang berisi sentimen positif hasil prediksi algoritma *Naïve Bayes* dapat dilihat pada Gambar dan persentase kemunculan kata pada *wordcloud* ini ditunjukkan pada Gambar 4 .55 dan Gambar 4. 56.

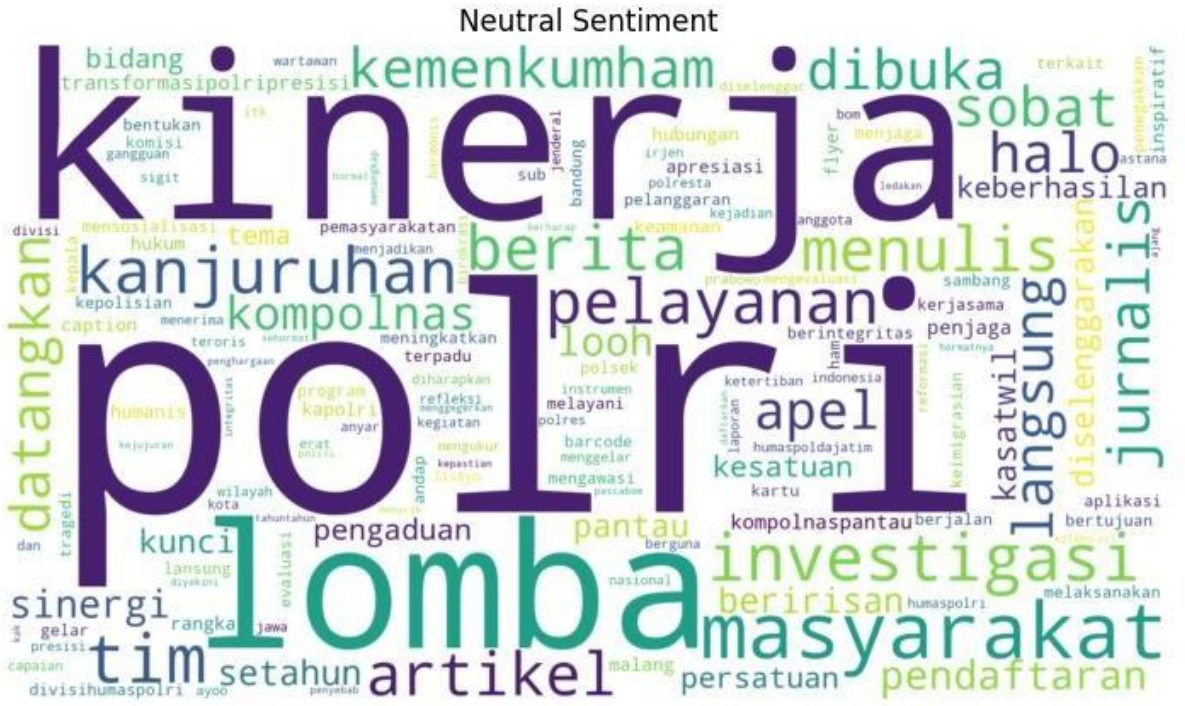
Gambar 4. 55 *Wordcloud* untuk Kelas Sentimen Positif pada Hasil Prediksi *Naïve Bayes*.

```
10 Kata Teratas dalam Sentimen Positif pada Hasil Prediksi Naive Bayes :
1. kinerja: 10.96%
2. polri: 10.93%
3. maksimal: 4.59%
4. tuntas: 2.02%
5. kapolri: 1.80%
6. apresiasi: 1.62%
7. kejahatan: 1.60%
8. mengapresiasi: 1.58%
9. masyarakat: 1.55%
10. mengusut: 1.53%
```

Gambar 4. 56 Persentase Kemunculan Kata dalam Sentimen Positif pada Hasil Prediksi *Naïve Bayes*

Seperti yang ditunjukkan oleh Gambar di atas, dibuat sebuah *wordcloud* yang berasal dari proses prediksi terhadap data uji oleh *Naïve Bayes* untuk menampilkan kata yang memiliki sentimen positif. Kata-kata yang sering muncul berdasarkan prediksi *Naïve Bayes* antara lain kinerja, polri, maksimal, tuntas, kapolri, apresiasi, kejahatan, mengapresiasi, masyarakat, dan

mengusut. Kata “kinerja” yang memiliki persentase kemunculan paling tinggi yaitu sebesar 10.96%. Kata yang mengandung sentimen diluar positif seperti “pembunuhan”, “terorisme” adalah kata yang pada data *tweet*-nya dilabeli sebagai positif. Contoh *tweet*-nya terdapat pada keterangan untuk Gambar 4. 46. Selanjutnya ditunjukkan *wordcloud* dengan sentimen netral yang berasal dari hasil prediksi *Naïve Bayes* dan persentase kemunculan katanya pada Gambar 4. 57 dan 4. 58.



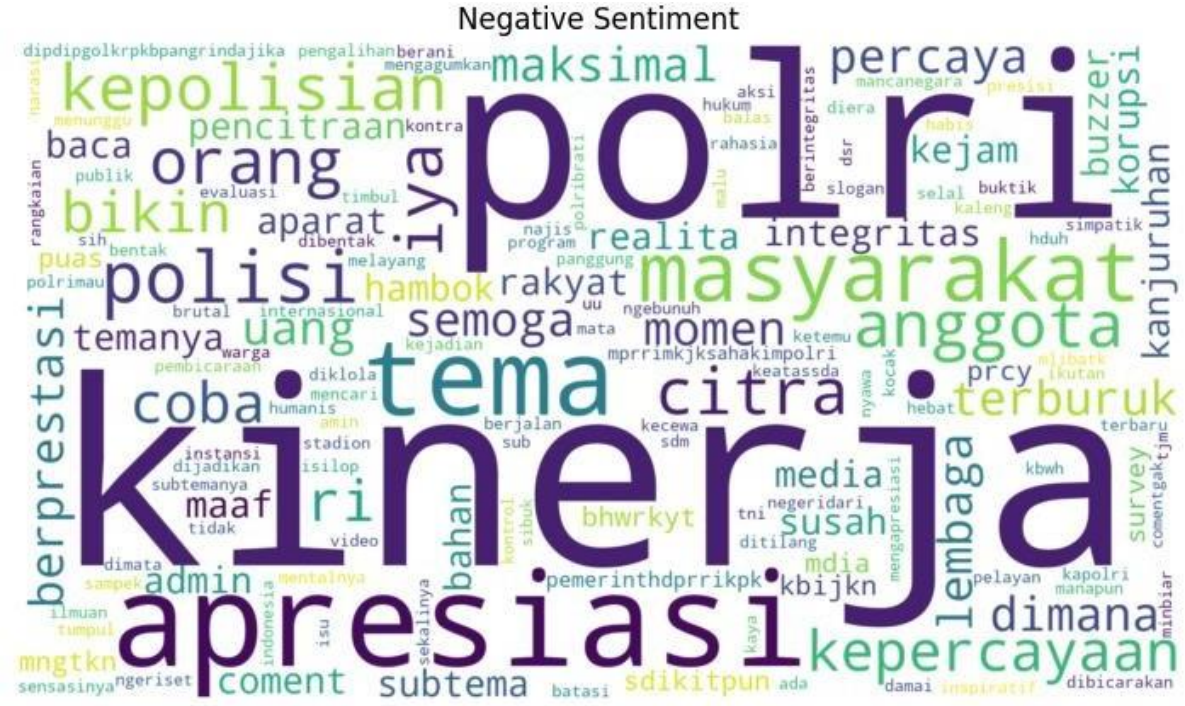
Gambar 4. 57 *Wordcloud* untuk Kelas Sentimen Netral pada Hasil Prediksi *Naïve Bayes*.

```
10 Kata Teratas dalam Sentimen  Netral pada Hasil Prediksi Naive Bayes :
1. polri: 8.86%
2. kinerja: 5.80%
3. lomba: 3.06%
4. masyarakat: 2.74%
5. tim: 1.97%
6. investigasi: 1.86%
7. kanjuruhan: 1.75%
8. pelayanan: 1.64%
9. menulis: 1.64%
10. berita: 1.64%
```

Gambar 4. 58 Persentase Kemunculan Kata dalam Sentimen Netral pada Hasil Prediksi *Naïve Bayes*

Dalam pembuatan *wordcloud* untuk kelas sentimen netral pada hasil prediksi yang dilakukan oleh algoritma *Naïve Bayes*, digunakan data *tweet* yang tidak memiliki

kecenderungan ke sentimen manapun. Kata yang paling sering muncul pada *wordcloud* ini yaitu polri, kinerja, lomba, masyarakat, tim, investigasi, kanjuruhan, pelayanan, menulis, dan berita. Kata yang paling sering muncul dalam *wordcloud* berisi sentimen netral pada hasil prediksi dengan *Naïve Bayes* adalah "polri", yaitu dengan persentase kemunculan sebesar 8.86%. Selanjutnya ditunjukkan *wordcloud* dengan sentimen negatif yang berasal dari hasil prediksi *Naïve Bayes* dan persentase kemunculan katanya pada Gambar 4. 59 dan 4. 60.



Gambar 4. 59 *Wordcloud* untuk Kelas Sentimen Negatif pada Hasil Prediksi *Naïve Bayes*.

```
10 Kata Teratas dalam Sentimen Negatif pada Hasil Prediksi Naive Bayes :
1. kinerja: 10.17%
2. polri: 7.69%
3. apresiasi: 4.71%
4. tema: 2.98%
5. masyarakat: 1.74%
6. polisi: 1.24%
7. iya: 0.99%
8. kepolisian: 0.99%
9. anggota: 0.99%
10. citra: 0.99%
```

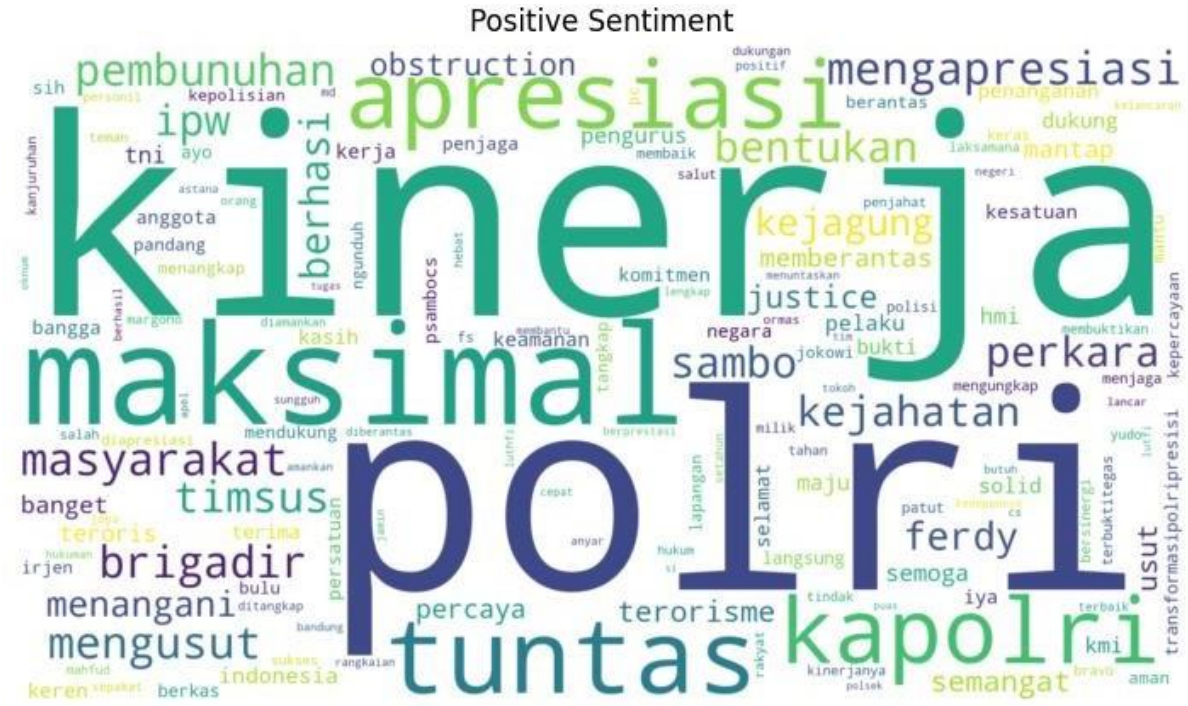
Gambar 4. 60 Persentase Kemunculan Kata dalam Sentimen Negatif pada Hasil Prediksi *Naïve Bayes*

Setelah membuat *wordcloud* yang berisikan *tweet* dengan sentimen positif dari hasil prediksi *Naïve Bayes*, dibuat sebuah *wordcloud* yang berisikan kata yang mengandung

sentimen negatif. Kata-kata yang paling sering muncul dalam *wordcloud* ini adalah kinerja, polri, apresiasi, tema, masyarakat, polisi, iya, kepolisian, anggota, dan citra. Kata yang paling sering muncul dalam *wordcloud* hasil prediksi *Naïve Bayes* unuk kelas sentimen *negatif* adalah "kinerja", yaitu dengan persentase kemunculan sebesar 10.17%.

Pada penelitian ini, terdapat algoritma lain yang digunakan sebagai *classifier*, yaitu *Support Vector Machine*. Maka dari itu, turut dibuat *wordcloud* yang berasal dari hasil prediksi SVM yang diklasifikasikan berdasarkan kelas sentimennya. *Wordcloud* hasil prediksi SVM untuk data *tweet* positif ditunjukkan pada Gambar 4. 61 dan persentase kemunculan kata dalam

wordcloud ini ditampilkan pada Gambar 4. 62.



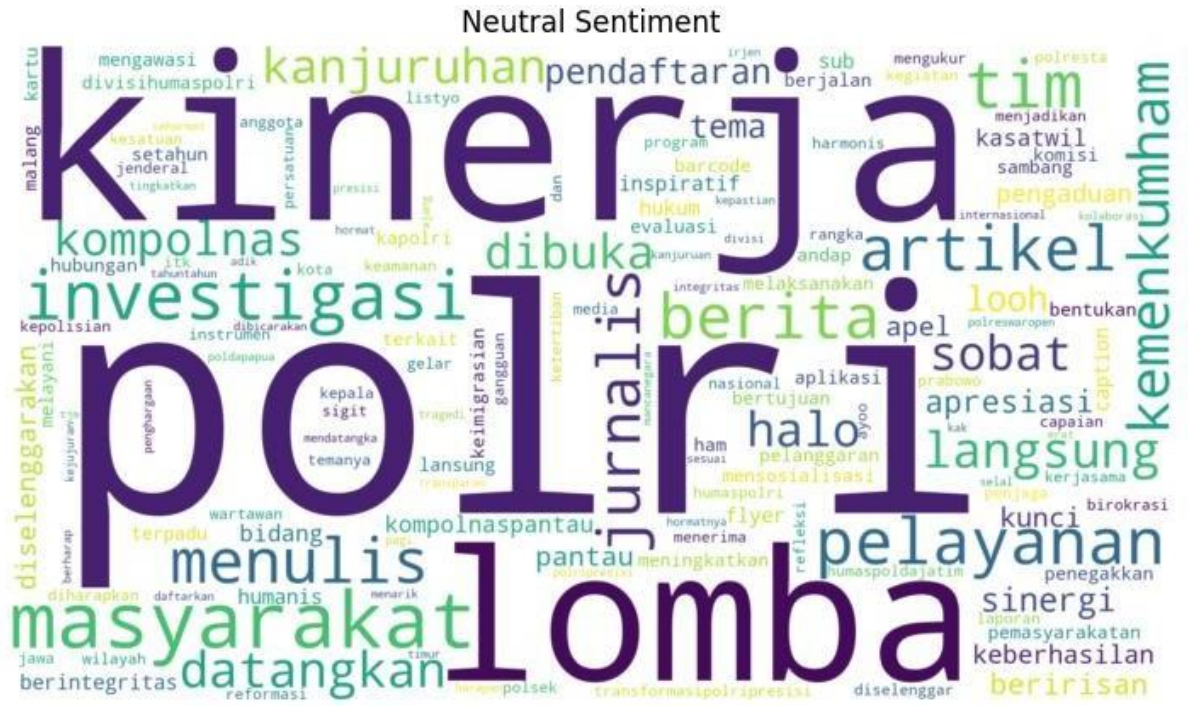
Gambar 4. 61 *Wordcloud* untuk Kelas Sentimen Positif pada Hasil Prediksi SVM

```
10 Kata Teratas dalam Sentimen Positif pada Hasil Prediksi SVM
1. kinerja: 10.29%
2. polri: 10.18%
3. maksimal: 4.37%
4. tuntas: 2.15%
5. apresiasi: 1.71%
6. kapolri: 1.62%
7. mengapresiasi: 1.57%
8. masyarakat: 1.55%
9. mengusut: 1.53%
10. brigadir: 1.48%
```

Gambar 4. 62 Persentase Kemunculan Kata dalam Sentimen Positif pada Hasil Prediksi SVM

Wordcloud pertama yang ditampilkan pada Gambar 4. 61 dibuat dengan hasil prediksi metode SVM terhadap data *tweet* yang memiliki sentimen positif. Kata yang frekuensi penggunaannya sering dan tercetak besar dalam *wordcloud* adalah sebagai berikut kinerja, polri, maksimal, tuntas, apresiasi, kapolri, mengapresiasi, masyarakat, mengusut, dan brigadir. Kata bersentimen negatif yang dikategorikan sebagai positif merupakan kata yang terdapat pada data *tweet* yang berlabel positif. Untuk contoh dari *tweet* tersebut juga dapat dilihat pada keterangan untuk Gambar 4. 46. Sementara kata yang paling sering muncul dalam *wordcloud* berisi *tweet* dengan kelas sentimen positif pada hasil prediksi SVM adalah "kinerja" dengan persentase kemunculan sebesar 10.29%. Selanjutnya dibuat *wordcloud* kedua yaitu yang

berisikan data *tweet* dengan sentimen netral dari hasil prediksi SVM yang ditunjukkan pada Gambar 4. 63 dan persentase kemunculan katanya pada Gambar 4. 64.



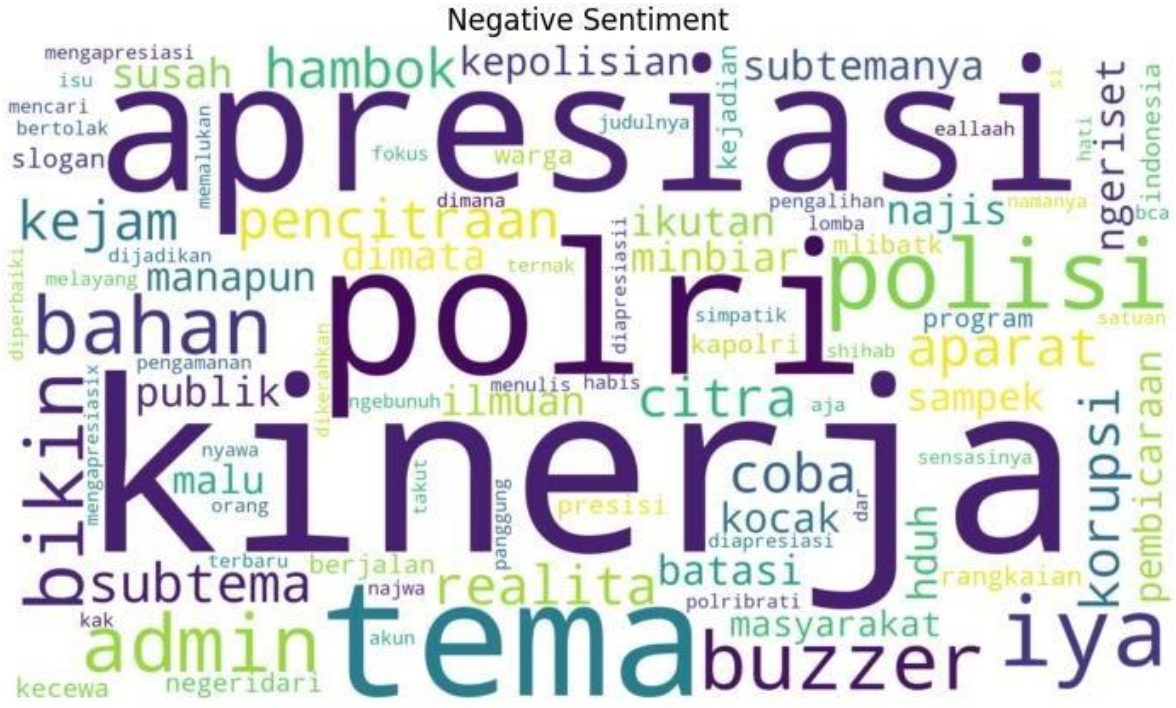
Gambar 4. 63 *Wordcloud* untuk Kelas Sentimen Netral pada Hasil Prediksi SVM.

```
10 Kata Teratas dalam Sentimen Netral pada Hasil Prediksi SVM
1. polri: 9.33%
2. kinerja: 5.93%
3. lomba: 3.53%
4. masyarakat: 2.90%
5. investigasi: 2.14%
6. tim: 2.02%
7. pelayanan: 1.89%
8. menulis: 1.77%
9. artikel: 1.77%
10. berita: 1.77%
```

Gambar 4. 64 Persentase Kemunculan Kata dalam Sentimen Netral pada Hasil Prediksi SVM

Wordcloud selanjutnya yang dibuat dari hasil prediksi algoritma SVM berisikan *tweet* yang memiliki kelas sentimen netral. Kata yang paling sering muncul dalam *wordcloud* ini antara lain polri, kinerja, lomba, masyarakat, investigasi, tim, pelayanan, menulis, artikel, dan berita.

Kata yang paling sering muncul pada *wordcloud* ini adalah "polri" dengan persentase kemunculan sebesar 9.33%. *Wordcloud* terakhir yang dibuat dari hasil prediksi SVM berisikan data *tweet* dengan sentimen negatif. *Wordcloud* ini ditunjukkan pada Gambar 4. 65 dan persentase kemunculan katanya pada Gambar 4.66.



Gambar 4. 65 *Wordcloud* untuk Kelas Sentimen Negatif pada Hasil Prediksi SVM.

```
10 Kata Teratas dalam Sentimen Negatif pada Hasil Prediksi SVM
1. kinerja: 11.56%
2. apresiasi: 6.12%
3. polri: 6.12%
4. tema: 4.76%
5. polisi: 2.04%
6. iya: 2.04%
7. kejam: 1.36%
8. korupsi: 1.36%
9. hambok: 1.36%
10. citra: 1.36%
```

Gambar 4. 66 Persentase Kemunculan Kata dalam Sentimen Negatif pada Hasil Prediksi SVM

Wordcloud terakhir yang berisikan data *tweet* yang memiliki sentimen negatif dibuat dari hasil prediksi algoritma SVM. Kata yang paling sering muncul dalam *wordcloud* ini ialah kinerja, apresiasi, polri, tema, polisi, iya, kejam, korupsi, *hambok*, dan citra. Terdapat kata *hambok* pada sentimen negatif yang merupakan penggunaan bahasa jawa, contoh penggunaannya adalah "*hambok yo ngono*". Persentase kata *hambok* mencapai 1.36% dari keseluruhan data *wordcloud* karena jumlah data uji yang memiliki sentimen negatif hanya berjumlah 36 *tweet*.Kata yang memiliki persentase terbesar, atau yang dapat diartikan sebagai yang paling sering muncul dalam *wordcloud* ini adalah "kinerja", dengan persentase kemunculan sebesar 11.56%.

4.9 Penyimpanan Model

Setelah selesai melakukan proses pembuatan *wordcloud*, diperlukan proses penyimpanan dengan maksud untuk tidak melatih ulang model yang akan digunakan. Modul *pickle* bawaan Python menjadi pilihan penyimpanan model yang berbentuk *.pkl.* Kode Pemrograman untuk melakukan penyimpanan model dengan menggunakan *pickle* ditunjukkan pada Gambar 4. 67.

```
1. import pickle
2.  with open ('naive_bayes_no_smote.pkl', 'wb') as r:
3.     pickle.dump(naive_bayes_model,  r)
4.  with open ('SVM_no_smote.pkl', 'wb') as r:
5.     pickle.dump(svm_model, r)
6.
7. with open ('naive_bayes_smote.pkl', 'wb') as r:
8.     pickle.dump(naive_bayes_model_imb,  r)
9. with open ('SVM_smote.pkl', 'wb') as r:
10.     pickle.dump(svm_model_imb,  r)
```

Gambar 4. 67 Kode Pemrograman Penyimpanan Model

Baris ke-1 dari kode penyimpanan model ini berfungsi untuk mengimpor *library* pickle, yaitu *library* dengan fungsi untuk menyimpan model yang telah dibuat. Baris ke-2 menyimpan sebuah model *Naïve Bayes* ke dalam file Bernama 'naïve_bayes_no_smote.pkl' menggunakan pickle.dump() yang terletak pada Baris ke-3. Baris ke-4 menyimpan model SVM ke dalam sebuah file Bernama 'SVM_no_smote.pkl' menggunakan pickle.dump() yang terletak pada Baris ke-5. Kode dari Baris ke-2 hingga Baris ke-5 merupakan penyimpanan model dalam bentuk yang belum dilakukan *Imbalanced Data Handling* dengan SMOTE terhadapnya.

Baris ke-7 dari kode ini menyimpan sebuah model *Naïve Bayes* yang Bernama 'naïve_bayes_smote.pkl' menggunakan pickle.dump() yang terletak pada Baris kode ke-8. Baris ke-9 dari kode ini menympan sebuah model SVM dengan nama 'SVM_smote.pkl' menggunakan pickle.dump() yang terletak pada baris kode ke-10. Fungsi dari Baris kode ke-7 hingga ke-10 ini adalah menyimpan model yang sudah dilakukan *Imbalanced Data Handling* terhadapnya.

4.10 Hasil Analisis Sentimen

Berdasarkan penelitian yang telah dilakukan terhadap *tweet* yang membahas mengenai kinerja POLRI, berikut disajikan analisis model *Naïve Bayes Classifier* dan *Support Vector Machine* yang telah dibuat untuk melakukan pengkategorian terhadap *tweet* yang membahas mengenai kinerja POLRI. Sesuai dengan rumusan masalah penelitian yaitu mempertanyakan

bagaimana kedua metode klasifikasi dapat mengategorikan *tweet* mengenai kinerja polri dengan tepat, maka diterapkan kedua model ini ke dalam model prediksi.

Cara kerja dari model ini adalah dengan memproses data *tweet* yang dimasukkan ke dalam model prediksi, lalu model akan melakukan *preprocessing* hingga ekstraksi fitur terhadap data input. Sehingga, model dapat mengategorikan ke kelas mana suatu *tweet* tentang kinerja POLRI berada. Model yang ditunjukkan pada Subbab 4.10 ini merupakan model yang telah diterapkan SMOTE terhadapnya. Kode Pemrograman dari model prediksi ini ditunjukkan pada Gambar 4. 68.

```
1. b = input('Masukkan Data Tweet: ')
2. b = text_clean(b,"(?:@|http?://|https?://|www)\S+")
3. b = b.lower()
4. b = tweet_tokenizer.tokenize(b)
5. b = normalized_term(b)
6. b = stopwords(b)
7. b = [stemmer_kata(word) for word in b]
8. b = [' '.join(b)]
9. b = vectorizer.transform(b)
10.b = b.toarray()
11.
12.# Define the mapping dictionary
13.sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}
14.
15.# Perform predictions
16.naive_bayes_prediction = naive_bayes_model_imb.predict(b)
17.svm_prediction = svm_model_imb.predict(b) # Assuming a_dense is
   defined and computed
18.
19.# Map numerical predictions to sentiment labels
20.naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
21.svm_sentiment = sentiment_mapping[svm_prediction[0]]
22.
23. # Print the results
24.print("Prediksi Sentimen Tweet Metode Naive Bayes : ",
   naive_bayes_sentiment)
25.print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

Gambar 4. 68 Kode Pemrograman Model Prediksi Sentimen

Penjelasan dari kode ini adalah sebagai berikut, baris kode ke-1 mengharuskan pengguna untuk memasukkan teks *tweet*, yang tersimpan dalam variabel b. Baris kode ke-2 mengaplikasikan fungsi pembersihan teks pada variabel b. Baris kode ke-3 mengonversikan teks yang telah bersih ke bentuk huruf kecil, langkah ini sering digunakan untuk memastikan

keseragaman data untuk analisis teks. Baris kode ke-4 melakukan tokenisasi terhadap teks yang sebelumnya diubah ke huruf kecil menjadi kata atau token. Baris kode ke-5 mengaplikasikan normalisasi teks pada token sebelumnya dengan tujuan untuk mengubah bentuk kata menjadi baku. Baris kode ke-6 menghapus *stopwords* yang ada pada data *tweet*. Baris kode ke-7 mengaplikasikan proses *stemming* pada token menggunakan fungsi stemmer_kata. Baris kode ke-8 mengonversikan kembali token yang telah di-*stemming* ke dalam bentuk *string* tunggal, yang dipisahkan oleh spasi. Baris kode ke-9 menggunakan sebuah vectorizer yang sebelumnya telah dilatih untuk mengubah teks yang telah di-*preprocessing* sebelumnya ke dalam bentuk vektor angka. Baris kode ke-10 mengonversikan bentuk *sparse matrix* pada teks yang telah divektorisasi ke dalam *dense NumPy array*.

Baris kode ke-13 mendefinisikan sebuah *mapping dictionary* untuk mengasosiasikan prediksi numerik dengan label sentimennya. Baris kode ke-16 hingga 17 melakukan prediksi menggunakan dua model, yaitu *Naïve Bayes* dan SVM pada data *tweet* b yang telah diproses. Baris kode ke-20 hingga 21 memetakan prediksi sentimen numerik pada labelnya menggunakan *dictionary* sentiment_mapping. Baris kode ke-24 hingga 25 mencetak label sentimen hasil dari prediksi menggunakan metode *Naïve Bayes* dan SVM.

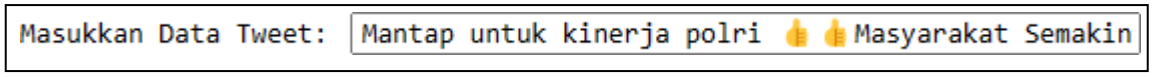
Implementasi dari proses prediksi terhadap suatu *tweet* dimulai dari input data *tweet*, lalu model akan bekerja melakukan pemrosesan terhadap teks dan melakukan ekstraksi fitur. Setelah dilakukan ekstraksi fitur, maka akan keluar prediksi sentimen dari *tweet* yang telah dimasukkan ke dalam model. *Tweet* yang dijadikan sebagai data uji untuk memberikan contoh untuk masing-masing sentimen pada data yang benar maupun data yang salah ditunjukkan pada Tabel 4. 12 untuk data *tweet* yang benar dalam masing-masing labelnya.

Tabel 4. 12 Data *Tweet* yang Benar untuk Kedua Sentimen

Tweet	Sentimen
Mantap untuk kinerja polri 👍 👍 Masyarakat Semakin Percaya https://t.co/hLEiaIqTi8	Positif
RT @humasreswaropen: "Apel Pagi Awal Kinerja Polri Yang Presisi." #DivisiHumasPolri #PoldaPapua #PolresWaropen #PolriPresisi https://t.co/o…	Netral

@txtdrberseragam Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg bikin lo… https://t.co/fBviwlEdVE	Negatif

Proses input data *tweet* bersentimen positif yaitu "Mantap untuk kinerja polri 👍 👍 Masyarakat Semakin Percaya https://t.co/hLEiaIqTi8" ditunjukkan pada Gambar 4. 69. Untuk hasil prediksinya ditunjukkan pada Gambar 4. 70 yang menunjukkan data yang benar diklasifikasi oleh *Naïve Bayes* dan SVM untuk sentimen positif. Data latih yang diterapkan untuk melakukan prediksi sentimen ini adalah data yang telah melalui proses *oversampling* dengan SMOTE yang tujuan untuk menyeimbangkan jumlah sentimen pada masing-masing kelas data.



Gambar 4. 69 Input Data Kedalam Model Prediksi

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Mantap untuk kinerja polri 👍 👍Masyarakat Semakin Percaya https://t.co/hLEiaIqTi8
Prediksi Sentimen Tweet Metode Naive Bayes :  POSITIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 70 Hasil Prediksi Model dengan Data yang Benar pada Sentimen Positif

Ditunjukkan pula contoh data yang benar diklasifikasi oleh *Naïve Bayes* dan SVM untuk memprediksi label sentimen netral dan negatif pada Gambar 4. 71 dan 4. 72. Untuk data *tweet* bersentimen netral adalah "@humasreswaropen: "Apel Pagi Awal Kinerja Polri Yang Presisi."#DivisiHumasPolri #PoldaPapua #PolresWaropen #PolriPresisi https://t.co/o…". Pada Gambar 4. 71, ditunjukkan bahwa *tweet* ini mendapatkan hasil prediksi dari *Naïve Bayes* dan SVM secara tepat sesuai labelnya, yaitu netral.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: RT @humasreswaropen: "Apel Pagi Awal Kinerja Polri Yang Presisi." #DivisiHumasPolri #PoldaPapua #PolresWar
open #PolriPresisi https://t.co/o…
Prediksi Sentimen Tweet Metode Naive Bayes :  NETRAL
Prediksi Sentimen Tweet Metode SVM :  NETRAL
```

Gambar 4. 71 Hasil Prediksi Model dengan Data yang Benar pada Sentimen Netral

Pada Gambar 4. 72 ditunjukkan hasil prediksi dari model *Naïve Bayes* dan SVM untuk menguji data yang benar pada sentimen negatif. *Tweet* tersebut adalah "@txtdrberseragam Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg bikin lo… https://t.co/fBviwlEdVE". Proses prediksi data yang benar ketika dilakukan oleh kedua model *classifier* menghasilkan prediksi sentimen yang sesuai dengan *tweet* yang memiliki label negatif. Kedua model sama-sama mengategorikan *tweet* yang diinput ke dalam label yang tepat, yaitu *tweet* negatif.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: @txtdrberseragam Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg
bikin lo… https://t.co/fBviwlEdVE
Prediksi Sentimen Tweet Metode Naive Bayes :  NEGATIF
Prediksi Sentimen Tweet Metode SVM :  NEGATIF
```

Gambar 4. 72 Hasil Prediksi Model dengan Data yang Benar pada Sentimen Negatif

Setelah model prediksi diuji dengan data yang benar pada masing-masing sentimen, selanjutnya dilakukan pengujian dengan memasukkan data ke dalam model prediksi untuk mengetahui kesalahan kedua *classifier* dalam mengklasifikasi suatu data *tweet*. Berikut ditunjukkan *tweet* apa saja yang menjadi data masukan dalam menguji kesalahan model SVM pada Tabel 4. 13

Tabel 4. 13 Data *Tweet* yang Benar dalam Masing-Masing Label pada SVM

Tweet	Sentimen
RT @lownllyzie: setelah kejadian bom di Polsek Astana Anyar Bandung,	Positif

polri berhasil mengamankan 24 teroris yangditangkap di wilayah Jawa b…	
Audit Kinerja Polsek Pangkalan Balai Oleh Tim ltwasum Mabes Polri di polres Banyuasin Tahun 2023 #Kapolri #KapoldaSumsel #KapolresBanyuasin #PolriPresisi2023 #PolriAktif2023 #rahmadwibowo #KapoldaSumsel #poldasumsel	Netral
@DivHumas_Polri liatlah hasil kinerja kelen ini wak. Tak respon cepat entah karena rakyat jelata bukan anak seleb atau pejabat, mungkin pas lapor sipelapor tak bawak syarat juga kali bah.	Negatif

Proses pertama dilakukan untuk menguji performa model SVM dalam melakukan klasifikasi terhadap *tweet* bersentimen positif. Gambar 4. 73 menunjukkan kesalahan model SVM dalam mengklasifikasi tweet "RT @lownllyzie: setelah kejadian bom di Polsek Astana Anyar Bandung, polri berhasil mengamankan 24 teroris yangditangkap di wilayah Jawa b…". *Tweet* ini memiliki sentimen positif, namun model SVM mengklasifikasikannya ke dalam sentimen netral.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: RT @lownllyzie: setelah kejadian bom di Polsek Astana Anyar Bandung, polri berhasil mengamankan 24 teroris
yang ditangkap di wilayah Jawa b…
Prediksi Sentimen Tweet Metode Naive Bayes :  NETRAL
Prediksi Sentimen Tweet Metode SVM :  NETRAL
```

Gambar 4. 73 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh SVM pada Sentimen Positif

Proses kedua dilakukan untuk melakukan pengujian terhadap performa model SVM dalam melakukan klasifikasi terhadap *tweet* bersentimen netral. Gambar 4. 74 menunjukkan kesalahan model SVM dalam mengklasifikasi tweet "Audit Kinerja Polsek Pangkalan Balai Oleh Tim ltwasum Mabes Polri di polres Banyuasin Tahun 2023 #Kapolri #KapoldaSumsel #KapolresBanyuasin #PolriPresisi2023 #PolriAktif2023 #rahmadwibowo #KapoldaSumsel #poldasumsel.". *Tweet* ini memiliki sentimen netral, namun model SVM mengklasifikasikannya ke dalam sentimen positif.

```
In [28]: b = input('Masukkan Data Tweet: ')
         b = text_clean(b,"(?:@|http?://|https?://|www)\S+")
         b = b.lower()
         b = tweet_tokenizer.tokenize(b)
         b = normalized_term(b)
         b = stopwords(b)
         b = [stemmer_kata(word) for word in b]
         b = [' '.join(b)]
         b = tfidf_vectorizer.transform(b)
         b = b.toarray()

         # Define the mapping dictionary
         sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

         # Perform predictions
         naive_bayes_prediction = naive_bayes_model_imb.predict(b)
         svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

         # Map numerical predictions to sentiment labels
         naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
         svm_sentiment = sentiment_mapping[svm_prediction[0]]

         # Print the results
         print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
         print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Audit Kinerja Polsek Pangkalan Balai Oleh Tim ltwasum Mabes Polri di polres Banyuasin Tahun 2023  #Kapolri
#KapoldaSumsel #KapolresBanyuasin #PolriPresisi2023 #PolriAktif2023  #rahmadwibowo #KapoldaSumsel  #poldasumsel
Prediksi Sentimen Tweet Metode Naive Bayes :  NETRAL
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 74 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh SVM pada Sentimen Netral

Proses terakhir dilakukan dengan tujuan untuk menguji performa model SVM dalam melakukan klasifikasi terhadap *tweet* bersentimen negatif. Gambar 4. 75 menunjukkan kesalahan model SVM dalam mengklasifikasi tweet dengan sentimen negatif. *Tweet* "@DivHumas_Polri liatlah hasil kinerja kelen ini wak. Tak respon cepat entah karena rakyat

jelata bukan anak seleb atau pejabat, mungkin pas lapor sipelapor tak bawak syarat juga kali bah" dilabeli sebagai positif oleh model SVM.

```
b = input('Masukkan Data Tweet: ')
b = text_clean(b,"(?:@|http?://|https?://|www)\S+")
b = b.lower()
b = tweet_tokenizer.tokenize(b)
b = normalized_term(b)
b = stopwords(b)
b = [stemmer_kata(word) for word in b]
b = [' '.join(b)]
b = tfidf_vectorizer.transform(b)
b = b.toarray()

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: @DivHumas_Polri  liatlah hasil kinerja kelen ini wak. Tak respon cepat entah karena rakyat jelata bukan an
ak seleb atau pejabat, mungkin pas lapor sipelapor tak bawak syarat juga kali bah.
Prediksi Sentimen Tweet Metode Naive Bayes :  NEGATIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 75 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh SVM pada Sentimen Negatif

Setelah model prediksi diuji dengan data *tweet* pada masing-masing kelas sentimen untuk mengetahui kesalahan proses klasifikasi yang dilakukan oleh model prediksi, selanjutnya dilakukan pengujian dengan memasukkan data ke dalam model prediksi. Hal ini bertujuan untuk mengetahui kesalahan *classifier Naïve Bayes* dalam mengklasifikasi suatu data *tweet*. Berikut ditunjukkan *tweet* apa saja yang menjadi data masukan dalam menguji kesalahan model *Naïve Bayes* pada Tabel 4. 14.

Tabel 4. 14 Data *Tweet* yang Benar dalam Masing-Masing Label pada *Naïve Bayes*

Tweet	Sentimen
@siapkawalpolri @ListyoSigitP apresiasi sekali sama kinerja polri	Positif
Betul, dg memahami & bertindak ssi aturan saja (contoh di jalan raya) itu sudah termasuk apresiasi thd kinerja Polri sih menurut saya. TNI juga begitu, kerja sesuai jobnya ya sudah biasa, apalagi yg kerjanya ga boleh diperlihatkan, Bagus ga dipuji Salah dicaci Hilang ga dicari	Netral

kasus Kamaruddin Simanjuntak ini salah satu contoh betapa payahnya kinerja Polri akhir-akhir ini. penetapan tersangka entah apa dasarnya. dugaan saya motifnya bukan penegakan hukum tapi asfshbdhdbdhbdjdnd *sinyal ilang*	Negatif

Proses pertama dilakukan untuk menguji performa model *Naïve Bayes* dalam melakukan klasifikasi terhadap *tweet* bersentimen positif. Gambar 4. 76 menunjukkan kesalahan model *Naïve Bayes* dalam mengklasifikasi tweet "@siapkawalpolri @ListyoSigitP apresiasi sekali sama kinerja polri". *Tweet* ini memiliki sentimen positif, namun model *Naïve Bayes* mengklasifikasikannya ke dalam sentimen negatif.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: @siapkawalpolri @ListyoSigitP apresiasi sekali sama kinerja polri
Prediksi Sentimen Tweet Metode Naive Bayes :  NEGATIF
Prediksi Sentimen Tweet Metode SVM :  NEGATIF
```

Gambar 4. 76 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh *Naïve Bayes* pada Sentimen Positif

Proses kedua dilakukan untuk menguji performa model *Naïve Bayes* dalam melakukan klasifikasi terhadap *tweet* bersentimen netral. Gambar 4. 77 menunjukkan kesalahan model *Naïve Bayes* dalam mengklasifikasi *tweet* "Betul, dg memahami & bertindak ssi aturan saja (contoh di jalan raya) itu sudah termasuk apresiasi thd kinerja Polri sih menurut saya. TNI juga begitu, kerja sesuai jobnya ya sudah biasa, apalagi yg kerjanya ga boleh diperlihatkan, Bagus ga dipuji Salah dicaci Hilang ga dicari". *Tweet* ini memiliki sentimen netral, namun model *Naïve Bayes* mengklasifikasikannya ke dalam sentimen negatif.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Betul, dg memahami & bertindak ssi aturan saja (contoh di jalan raya) itu sudah termasuk apresiasi thd kin
erja Polri sih menurut saya.  TNI juga begitu, kerja sesuai jobnya ya sudah biasa, apalagi yg kerjanya ga boleh diperlihatkan,
Bagus ga dipuji Salah dicaci Hilang ga dicari
Prediksi Sentimen Tweet Metode Naive Bayes :  NEGATIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 77 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh *Naïve Bayes* pada Sentimen Netral

Proses terakhir yang dilakukan untuk menguji performa model *Naïve Bayes* dalam melakukan klasifikasi terhadap *tweet* bersentimen negatif. Gambar 4. 78 menunjukkan kesalahan model *Naïve Bayes* dalam mengklasifikasi *tweet* "kasus Kamaruddin Simanjuntak ini salah satu contoh betapa payahnya kinerja Polri akhir-akhir ini. penetapan tersangka entah apa dasarnya. dugaan saya motifnya bukan penegakan hukum tapi asfshbdhdbdhbdjdnd *sinyal ilang*". *Tweet* ini memiliki sentimen negatif, namun model *Naïve Bayes* mengklasifikasikannya ke dalam sentimen positif.

```
In [27]: b = input('Masukkan Data Tweet: ')
         b = text_clean(b,"(?:@|http?://|https?://|www)\S+")
         b = b.lower()
         b = tweet_tokenizer.tokenize(b)
         b = normalized_term(b)
         b = stopwords(b)
         b = [stemmer_kata(word) for word in b]
         b = [' '.join(b)]
         b = tfidf_vectorizer.transform(b)
         b = b.toarray()

         # Define the mapping dictionary
         sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

         # Perform predictions
         naive_bayes_prediction = naive_bayes_model_imb.predict(b)
         svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

         # Map numerical predictions to sentiment labels
         naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
         svm_sentiment = sentiment_mapping[svm_prediction[0]]

         # Print the results
         print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
         print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: kasus Kamaruddin Simanjuntak ini salah satu contoh betapa payahnya kinerja Polri akhir-akhir ini. penetapa
n tersangka entah apa dasarnya. dugaan saya motifnya bukan penegakan hukum tapi asfshbdhdbdhbdjdnd *sinyal ilang*
Prediksi Sentimen Tweet Metode Naive Bayes :  POSITIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 78 Hasil Prediksi Model dengan Data yang Salah Diklasifikasi oleh *Naïve Bayes* pada Sentimen Negatif

Berdasarkan hasil prediksi terhadap data *tweet* yang dimasukkan ke dalam model prediksi sentimen, didapatkan hasil berupa model *Naïve Bayes* dan *Support Vector Machine* yang berhasil melakukan klasifikasi *tweet* dengan tepat. Hal ini didukung dengan hasil pengujian kedua model yang memiliki akurasi 91,4% untuk *Naïve Bayes* dan 91,1% untuk model *Support Vector Machine* pada saat diterapkan SMOTE. Ketika tidak diterapkan SMOTE, model memiliki nilai akurasi yang lebih rendah, yaitu 90,2% untuk *Naïve Bayes,* dan 90,9% untuk SVM Proses pelabelan juga berkontribusi untuk menyumbangkan hasil bahwa sentimen mengenai Institusi Kepolisian Republik Indonesia (POLRI) di media sosial Twitter pada rentang waktu September – Desember 2022 yaitu **POSITIF**. Alasannya adalah karena dominannya label positif yang terdapat pada *dataset*, yaitu berjumlah 2313 *tweet,* jika dibandingkan dengan *tweet* netral yang berjumlah 337 *tweet* netral, dan *tweet* negatif yang berjumlah 164. Algoritma *Naïve Bayes* baik digunakan saat menyesuaikan pada tipe *Naïve Bayes* yang digunakan, seperti pada klasifikasi teks pada penelitian ini menggunakan Multinomial *Naïve Bayes*. Algoritma SVM baik digunakan pada saat dataset jumlahnya tidak terlalu besar, karena mengakibatkan waktu *training* data yang akan menjadi lama.

4.11 Analisis Perbandingan Hasil Data *Balanced* dan *Imbalanced*

Proses pelabelan yang dilakukan dalam penelitian ini menghasilkan klasifikasi data yang bisa dinilai tidak seimbang. Sangat banyaknya *tweet* yang mendapatkan label positif dan jumlahnya sangat terpaut jauh jika dibandingkan dengan *tweet* netral dan *tweet* negatif sehingga butuh untuk diterapkan SMOTE untuk menyeimbangkan data. Pada pemrosesan yang terdapat dalam keseluruhan laporan penelitian ini, data yang digunakan adalah data yang sudah diseimbangkan, sehingga performa model mengalami peningkatan. Data *tweet* yang dijadikan sebagai data masukan dalam model prediksi ini merupakan data uji.

Berikut ditunjukkan *tweet* apa saja yang dijadikan sebagai bahan uji dan label sentimennya dalam Tabel 4. 15. *Tweet* pertama dimasukkan untuk mengetahui performa model dalam menguji data *tweet* dengan sentimen netral. *Tweet* kedua untuk menguji data *tweet* dengan sentimen negatif, dan *tweet* ketiga untuk menunjukkan sentimen positif.

Tabel 4. 15 Daftar *Tweet* dari Data Uji untuk Dijadikan sebagai Data Masukan

Tweet	Sentimen
Halo Sobat Jurnalis.. Lomba ini diselenggarakan untuk para Jurnalis/ Wartawan yg akan berlangsung	Netral

mulai 5 s/d 18 O… https://t.co/AIwkXgB3zR	
Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg bikin lomba tulisan tentang apresiasi kinerja polri. Wkwk beraaaak 🤢	Negatif
Jakarta. Sebanyak 86,1% responden menyatakan puas atas kinerja polri memberantas tindak pidana perdagangan orang (TPPO) melalui satuan tugas khusus (satgasus) bentukan Kapolri Jenderal Polisi Drs. Listyo Sigit Prabowo.	Positif

Pada *tweet* pertama bersentimen netral yang dimasukkan ke dalam model prediksi, kedua jenis model data latih, yaitu *balanced* data dan *imbalanced* data, menghasilkan prediksi yang sama dalam dua *classifier* yang diuji. *Classifier Naïve Bayes* pada kedua jenis model data sama-sama mengategorikan *tweet* ke dalam sentimen netral, dan classifier SVM pada kedua jenis model data juga mengategorikan *tweet* tersebut ke dalam sentimen netral. Perbandingan kedua jenis data ditunjukkan pada Gambar 4. 79 dan 4. 80.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model.predict(a)
svm_prediction = svm_model.predict(a)  # Assuming a_dense is defined and computed

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping.get(naive_bayes_prediction[0])
svm_sentiment = sentiment_mapping.get(svm_prediction[0])

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Halo Sobat Jurnalis.. Lomba ini diselenggarakan untuk para Jurnalis/ Wartawan yg akan berlangsung mulai 5
s/d 18 O… https://t.co/AIwkXgB3zR
Prediksi Sentimen Tweet Metode Naive Bayes :  NETRAL
Prediksi Sentimen Tweet Metode SVM :  NETRAL
```

Gambar 4. 79 Analisis *Tweet* netral tanpa *Imbalanced Data Handling*

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Halo Sobat Jurnalis.. Lomba ini diselenggarakan untuk para Jurnalis/ Wartawan yg akan berlangsung mulai 5
s/d 18 O… https://t.co/AIwkXgB3zR
Prediksi Sentimen Tweet Metode Naive Bayes :  NETRAL
Prediksi Sentimen Tweet Metode SVM :  NETRAL
```

Gambar 4. 80 Analisis *Tweet* netral dengan *Imbalanced Data Handling*

Selanjutnya dilakukan perbandingan dengan memasukkan *tweet* negatif pada kedua jenis data. Model prediksi yang tidak mengalami *Imbalanced Data Handling* mengategorikan *tweet* ini ke dalam kelas sentimen positif untuk *classifier Naïve Bayes* dan kelas negatif untuk sentimen SVM. Sementara pada model yang mengalami *Imbalanced Data Handling* kedua *classifier* sama-sama mengategorikan *tweet* ini ke dalam sentimen negatif. Perbandingan keduanya dapat dilihat pada Gambar 4. 81 dan 4. 82.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model.predict(a)
svm_prediction = svm_model.predict(a)  # Assuming a_dense is defined and computed

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping.get(naive_bayes_prediction[0])
svm_sentiment = sentiment_mapping.get(svm_prediction[0])

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg bikin lomba tuli
san tentang apresiasi kinerja polri. Wkwk beraaaak
Prediksi Sentimen Tweet Metode Naive Bayes :  POSITIF
Prediksi Sentimen Tweet Metode SVM :  NEGATIF
```

Gambar 4. 81 Analisis *Tweet* negatif tanpa *Imbalanced Data Handling*

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Polisi ini dimana2 nyari panggung ya, udh ngebunuh 100+ ada aja sensasinya. Yg terbaru yg bikin lomba tuli
san tentang apresiasi kinerja polri. Wkwk beraaaak
Prediksi Sentimen Tweet Metode Naive Bayes :  NEGATIF
Prediksi Sentimen Tweet Metode SVM :  NEGATIF
```

Gambar 4. 82 Analisis *Tweet* negatif dengan *Imbalanced Data Handling*

Perbandingan terakhir terhadap performa kedua model prediksi dilakukan dengan memasukkan *tweet* positif pada kedua jenis data. Model prediksi yang tidak mengalami *Imbalanced Data Handling* mengategorikan *tweet* ini ke dalam kelas sentimen positif untuk kedua jenis *classifier*. Model dengan penerapan *Imbalanced Data Handling* juga menghasilkan pengkategorian yang sama untuk kedua *classifier*, yaitu ke dalam kelas positif. Perbandingan keduanya dapat dilihat pada Gambar 4. 83 dan 4. 84.

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model_imb.predict(b)
svm_prediction = svm_model_imb.predict(b)  # Assuming a_dense is defined and computed

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping[naive_bayes_prediction[0]]
svm_sentiment = sentiment_mapping[svm_prediction[0]]

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Jakarta. Sebanyak 86,1% responden menyatakan puas atas kinerja polri memberantas tindak pidana perdagangan
orang (TPPO) melalui satuan tugas khusus (satgasus) bentukan Kapolri Jenderal Polisi Drs. Listyo Sigit Prabowo.
Prediksi Sentimen Tweet Metode Naive Bayes :  POSITIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 83 Analisis *Tweet* Positif tanpa *Imbalanced Data Handling*

```
# Perform predictions
naive_bayes_prediction = naive_bayes_model.predict(a)
svm_prediction = svm_model.predict(a)  # Assuming a_dense is defined and computed

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Define the mapping dictionary
sentiment_mapping = {1: 'POSITIF', 2: 'NETRAL', 3: 'NEGATIF'}

# Map numerical predictions to sentiment labels
naive_bayes_sentiment = sentiment_mapping.get(naive_bayes_prediction[0])
svm_sentiment = sentiment_mapping.get(svm_prediction[0])

# Print the results
print("Prediksi Sentimen Tweet Metode Naive Bayes : ", naive_bayes_sentiment)
print("Prediksi Sentimen Tweet Metode SVM : ", svm_sentiment)
```

```
Masukkan Data Tweet: Jakarta. Sebanyak 86,1% responden menyatakan puas atas kinerja polri memberantas tindak pidana perdagangan
orang (TPPO) melalui satuan tugas khusus (satgasus) bentukan Kapolri Jenderal Polisi Drs. Listyo Sigit Prabowo.
Prediksi Sentimen Tweet Metode Naive Bayes :  POSITIF
Prediksi Sentimen Tweet Metode SVM :  POSITIF
```

Gambar 4. 84 Analisis *Tweet* Positif dengan *Imbalanced Data Handling*

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan mengenai analisis sentimen kinerja Kepolisian Republik Indonesia (POLRI), maka dapat dilakukan penarikan kesimpulan, yakni algoritma *Naïve Bayes Classifier* dan *Support Vector Machine* dapat mengategorikan *tweet* dengan baik sesuai dengan kelasnya. Hal ini ditunjukkan dengan berhasilnya algoritma ini memprediksi sentimen pada data *tweet* baru yang dimasukkan dengan tepat. Namun, masih terdapat kesalahan pada proses klasifikasi sehingga menghasilkan prediksi sentimen yang salah. Skor akurasi yang diperoleh kedua algoritma ini masing-masing 91,4% untuk *Naïve Bayes Classifier* dan 91,1% untuk *Support Vector Machine*, hal ini dapat terjadi karena dilakukannya penyeimbangan terhadap data latih dengan metode SMOTE. Waktu prediksi yang dibutuhkan oleh *Naïve Bayes Classifier* lebih sedikit dibandingkan dengan *Support Vector Machine* yaitu hanya 0.00 detik. Sedangkan *Support Vector Machine* membutuhkan waktu sekitar 0.06 detik, sehingga dapat dinyatakan bahwa metode *Naïve bayes* memiliki performa yang lebih baik dalam melakukan klasifikasi. Didasari oleh pengelompokkan data *tweet* mengenai kinerja POLRI di media sosial Twitter menurut label datanya, ditemukan bahwa sentimen mengenai kinerja POLRI adalah positif. Kata yang paling sering muncul dalam keseluruhan data yang dikumpulkan dari media sosial Twitter adalah polri, kerja, apresiasi, usut, tuntas, maksimal, kerja.

5.2 Saran

Pada penelitian ini, masih terdapat banyak sekali keterbatasan dan kekurangan, maka dari itu penulis memberikan beberapa saran agar dapat disempurnakan pada penelitian berikutnya:

a. Menambahkan fitur klasifikasi teks yang lebih spesifik seperti komentar dengan kelas kritik, saran, pengancaman, dan lain-lain.
b. Penambahan data *training* dengan tujuan untuk meningkatkan kinerja model yang dihasilkan.
c. Menambahkan penggunaan metode klasifikasi selain *Naïve Bayes Classifier* dan *Support Vector Machine* agar dapat dilakukan perbandingan dengan lebih beragam untuk dapat ditentukan algoritma yang memiliki kinerja lebih baik di antara metode lainnya.

DAFTAR PUSTAKA

Al-Aidaroos, K. M., Abu Bakar, A., & Othman, Z. (2010). Naïve Bayes variants in classification learning. *Proceedings - 2010 International Conference on Information Retrieval and Knowledge Management: Exploring the Invisible World, CAMP'10*, 276–281. https://doi.org/10.1109/INFRKM.2010.5466902

Ali, W. P., Sibaroni, Y., & Si, S. (2019). Analisis Sentimen Masyarakat Terhadap Kinerja Presiden Indonesia Dalam Aspek Ekonomi , Kesehatan , dan Pembangunan Berdasarkan Opini dari Twitter. *E-Proceeding of Engineering*, *6*(2), 8637–8649.

Amirul Haj, A. S., Amrizal, V., & Arini, A. (2020). Analisis Sentimen Kinerja KPU Pemilu 2019 Menggunakan Algoritma K-Means Dengan Algoritma Confix Stripping Stemmer. *Journal of Innovation Information Technology and Application (JINITA)*, *2*(01), 9–18. https://doi.org/10.35970/jinita.v2i01.119

Buntoro, G. A. (2017). Analisis Sentimen Calon Gubernur DKI Jakarta 2017 Di Twitter. *Integer Journal*, *2*(1), 32–41. https://t.co/jrvaMsgBdH

Chapman, C., & Stolee, K. T. (2016). Exploring regular expression usage and context in python. *ISSTA 2016 - Proceedings of the 25th International Symposium on Software Testing and Analysis*, 282–293. https://doi.org/10.1145/2931037.2931073

Chou, J. S., Cheng, M. Y., Wu, Y. W., & Pham, A. D. (2014). Optimizing parameters of support vector *machine* using fast messy genetic algorithm for dispute classification. *Expert Systems with Applications*, *41*(8), 3955–3964. https://doi.org/10.1016/j.eswa.2013.12.035

Darwis, D., Siskawati, N., & Abidin, Z. (2021). Penerapan Algoritma Naive Bayes Untuk Analisis Sentimen Review Data Twitter Bmkg Nasional. *Jurnal Tekno Kompak*, *15*(1), 131. https://doi.org/10.33365/jtk.v15i1.744

Dermawan, R., Herdiansyah, M. I., Teknik, F., Komputer, I., & Darma, U. B. (n.d.). *Analisis sentimen komentar kinerja kegiatan kepala daerah kota lubuk linggau*. 1027–1040.

Duei Putri, D., Nama, G. F., & Sulistiono, W. E. (2022). Analisis Sentimen Kinerja Dewan Perwakilan Rakyat (DPR) Pada Twitter Menggunakan Metode Naive Bayes Classifier. *Jurnal Informatika Dan Teknik Elektro Terapan*, *10*(1), 34–40. https://doi.org/10.23960/jitet.v10i1.2262

Farmadiansyah, A. Z., Hidayatullah, A. F., & Rahma, F. (2021). Deteksi Surel Spam dan Non Spam Bahasa Indonesia. *Automata*. https://journal.uii.ac.id/AUTOMATA/article/view/19514

Febriyanti, A. (2020). *Analisis Sentimen Persepsi Pengguna JNE Menggunakan Algoritma Naïve Bayes Classifier*. *16522259*. https://dspace.uii.ac.id/handle/123456789/24681%0Ahttps://dspace.uii.ac.id/bitstream/handle/123456789/24681/16522259 Adelia Febriyanti.pdf?sequence=1&isAllowed=y

Giovani, A. P., Ardiansyah, A., Haryanti, T., Kurniawati, L., & Gata, W. (2020). Analisis Sentimen Aplikasi Ruang Guru Di Twitter Menggunakan Algoritma Klasifikasi. *Jurnal Teknoinfo*, *14*(2), 115. https://doi.org/10.33365/jti.v14i2.679

Harijiatno, S. D. (2019). Analisis Sentimen Pada Twitter Menggunakan Multinominal Naive Bayes Skripsi. *Αγαη*, *8*(5), 55.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Imron, A. (2019). Analisis Sentimen Terhadap Tempat Wisata di Kabupaten Rembang Menggunakan Metode Naive Bayes Classifier. *Teknik Informatika*, 10–13. https://dspace.uii.ac.id/handle/123456789/14268

Irfan, M. (2022). Named Entity Recognition Untuk Data Review Tempat Wisata Dengan Metode "Bidirectional Encoder Representations from Transformers." *Universitas Islam Indonesia*.

Istiani, N., & Islamy, A. (2020). Fikih Media Sosial Di Indonesia. *Asy Syar'Iyyah: Jurnal Ilmu Syari'Ah Dan Perbankan Islam*, *5*(2), 202–225. https://doi.org/10.32923/asy.v5i2.1586

Kowalski, G. (2011). Information Retrieval Architecture and Algorithms. In *Antimicrobial agents and chemotherapy* (Vol. 58, Issue 12). Springer US. https://doi.org/10.1007/978-1-4419-7716-8

Kristiyanti, D. A. (2015). Analisis Sentimen Review Produk Kosmetik Menggunakan Algoritma Support Vector *Machine* Dan Particle Swarm Optimization Sebagai. *Seminar Nasional Inovasi & Tren (SNIT) 2015 "Peluang Dan Tantangan Indonesia Dalam Menyikapi Afta 2015,"* 134–141. http://lppm.bsi.ac.id/SNIT2015/BidangA/A22-134-141_2015-SNIT-Dinar Ajeng Kristiyanti_ ALGORITMA SUPPORT VECTOR.pdf

Librarian, A. (2017). *High quality stemmer library for Indonesian Language (Bahasa)*. GitHub. https://github.com/sastrawi/sastrawi

Liu, B. (2012). *Sentiment Analysis and Opinion mining*. https://doi.org/10.1007/978-3-031-

02145-9

Liu, J., Tian, Z., Liu, P., Jiang, J., & Li, Z. (2016). An approach of semantic web service classification based on naive bayes. *Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016*, 356–362. https://doi.org/10.1109/SCC.2016.53

Mckinney, W. (2018). *Python for Data Analysis (Second Edition)*.

Mulia, R. K. (2021). *Text mining Analysis Dan Sentiment Analysis Dengan Menggunakan Metode Naive Bayes Classifier*.

Muslehatin, W., Ibnu, M., & Mustakim. (2017). PENERAPAN NAÏVE BAYES CLASSIFICATION UNTUK KLASIFIKASI TINGKATKEMUNGKINAN OBESITAS MAHASISWA SISTEM INFORMASI UIN SUSKA RIAU | Muslehatin | Seminar Nasional Teknologi Informasi Komunikasi dan Industri. *... Komunikasi Dan Industri*, 250–256. http://ejournal.uin-suska.ac.id/index.php/SNTIKI/article/view/3276/2158

Normawati, D., & Prayogi, S. A. (2021). Implementasi Naïve Bayes Classifier Dan Confusion Matrix Pada Analisis Sentimen Berbasis Teks Pada Twitter. *Jurnal Sains Komputer & Informatika (J-SAKTI*, *5*(2), 697–711. http://ejurnal.tunasbangsa.ac.id/index.php/jsakti/article/view/369

Nugroho, A. S. (2007). Pengantar Support Vector *Machine* ∗. *Jurnal Data Mining, Jakarta*, 3.

Nugroho, D. G., Chrisnanto, Y. H., & Wahana, A. (2015). *Analisis Sentimen Pada Jasa Ojek Online ... (Nugroho dkk.)*. 156–161.

Nurhopipah, A., Ceasar, Y., & Priadana, A. (2021). Improving *Machine* Leaming Accuracy using Data Augmentation in Recruitment Recommendation Process. *3rd 2021 East Indonesia Conference on Computer and Information Technology, EIConCIT 2021*, 203–208. https://doi.org/10.1109/EIConCIT50028.2021.9431908

Permana, A. Y., & Makmun, M. (2020). Analisis Sentimen pada Teks Opini Penilaian Kinerja Dosen dengan Pendekatan Algoritma KNN. *Jurnal Ilmiah Komputasi*, *19*(1), 39–50. https://doi.org/10.32409/jikstik.19.1.154

Prakoso, R. (2017). *analisis-sentimen/kamus at master · ramaprakoso/analisis-sentimen · GitHub*. https://github.com/ramaprakoso/analisis-sentimen/tree/master/kamus

Purbohastuti, A. W. (2017). Efektivitas Media Sosial Sebagai Media Promosi. *Tirtayasa Ekonomika*, *12*(2), 212. https://doi.org/10.35448/jte.v12i2.4456

Purwarianti, A., Andhika, A., Wicaksono, A. F., Afif, I., & Ferdian, F. (2016). InaNLP: Indonesia *Natural Language Processing* toolkit, case study: Complaint tweet classification. *4th IGNITE Conference and 2016 International Conference on Advanced*

Informatics: Concepts, Theory and Application, ICAICTA 2016, 5–9. https://doi.org/10.1109/ICAICTA.2016.7803103

QORITA, A. K. (2022). *Analisis Sentimen Berbasis Aspek Pada Ulasan Tempat Wisata Diy*. https://dspace.uii.ac.id/handle/123456789/39202%0Ahttps://dspace.uii.ac.id/bitstream/handle/123456789/39202/18523214.pdf?sequence=1

Qudsi, D. H., Lubis, J. H., Syaliman, K. U., & Najwa, N. F. (2021). Analisis Sentimen Pada Data Saran Mahasiswa Terhadap Kinerja Departemen Di Perguruan Tinggi Menggunakan Sentiment Analysis in the Student ' S Reviews of College Department Performance Using. *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIIK)*, *8*(5), 1067–1076. https://doi.org/10.25126/jtiik.202184842

Rachman, F. P. (2021). Perbandingan Model Deep Learning untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Languange Processing. *Jurnal Teknologi Dan Manajemen Informatika*, *7*(2), 113–121. https://doi.org/10.26905/jtmi.v7i2.6506

Ratnawati, F. (2018). Implementasi Algoritma Naive Bayes Terhadap Analisis Sentimen Opini Film Pada Twitter. *INOVTEK Polbeng - Seri Informatika*, *3*(1), 50. https://doi.org/10.35314/isi.v3i1.335

Rosid, M. A., Fitrani, A. S., Astutik, I. R. I., Mulloh, N. I., & Gozali, H. A. (2020). Improving Text Preprocessing for Student Complaint Document Classification Using Sastrawi. *IOP Conference Series: Materials Science and Engineering*, *874*(1). https://doi.org/10.1088/1757-899X/874/1/012017

Saravanan, R., & Sujatha, P. (2019). A State of Art Techniques on *Machine Learning* Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, *Iciccs*, 945–949. https://doi.org/10.1109/ICCONS.2018.8663155

Septian, J. A., Fachrudin, T. M., & Nugroho, A. (2019). Analisis Sentimen Pengguna Twitter Terhadap Polemik Persepakbolaan Indonesia Menggunakan Pembobotan TF-IDF dan K-Nearest Neighbor. *Journal of Intelligent System and Computation*, *1*(1), 43–49. https://doi.org/10.52985/insyst.v1i1.36

Siringoringo, R. (2018). Klasifikasi data tidak Seimbang menggunakan algoritma SMOTE dan k-nearest neighbor. *Jurnal ISD*, *3*(1), 44–49.

Sorgente, Antonio, Vettigli, Giuseppe, & Mele, Francesco. (2022). An Italian corpus for aspect based sentiment analysis of movie reviews. *Proceedings of the First Italian Conference on Computational Linguistics CLiC-It 2014 and of the Fourth International Workshop*

EVALITA 2014 9-11 December 2014, Pisa, 349–353. https://doi.org/10.12871/clicit2014167

Tala, F. Z. (2003). A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. *M.Sc. Thesis, Appendix D*, *pp*, 39–46.

Villavicencio, C., Macrohon, J. J., Inbaraj, X. A., Jeng, J. H., & Hsieh, J. G. (2021). Twitter sentiment analysis towards covid-19 vaccines in the Philippines using naïve bayes. *Information (Switzerland)*, *12*(5). https://doi.org/10.3390/info12050204

Vitandy, S. W. U., Supianto, A. A., & Bachtiar, F. A. (2019). Analisis Sentimen Evaluasi Kinerja Dosen menggunakan Term Frequency- Inverse Document Frequency dan Naïve Bayes Classifier. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(6), 6080–6088. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5645