BAB III

PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan kerja magang, bidang Pengembangan e-*Government* atau Diskominfo Kota Tangerang sendiri tidak memiliki struktur pasti dan resmi, namun, secara sistem kerja dan alur komunikasi yang terjadi sehari-hari dapat digambarkan sebagai berikut.

3.1.1 Kedudukan

Gambar 3.1 Sistem Kerja Magang

Berikut deskripsi dari masing-masing peran yang terlibat pada gambar 3.1 :

1. ***Supervisor***

   Supervisor memegang peranan krusial dan tertinggi dalam manajemen proyek sekaligus pengembangan kompetensi peserta melalui program PRO-Step *career acceleration*. Bertindak sebagai mentor sekaligus pengawas, supervisor bertanggung jawab memberikan penugasan strategis, memantau hasil kerja secara periodik sesuai standar instansi, serta memberikan persetujuan atas seluruh tugas harian yang telah diselesaikan. Dalam proses

bimbingan, supervisor menerapkan pendekatan edukatif yang mendorong kemampuan berpikir kritis dan kreatif peserta magang melalui diskusi dua arah yang solutif saat menghadapi kendala teknis. Selain dukungan bimbingan, supervisor juga memegang wewenang administratif dalam proses penilaian formal dengan menginput *evaluation grade* 1 dan 2 sebagai parameter keberhasilan peserta selama menjalani masa magang.

2. **Asisten *Supervisor***

   Asisten supervisor memiliki peran strategis sebagai penghubung yang menjembatani interaksi antara peserta program PRO-Step *career acceleration* dengan supervisor melalui pendekatan bimbingan yang lebih fleksibel. Walaupun tidak memiliki otoritas dalam pemberian nilai formal seperti *evaluation grade* 1 dan 2, asisten supervisor menjadi sosok pendukung utama yang memastikan kelancaran teknis setiap penugasan, terutama saat supervisor berhalangan hadir. Dengan gaya komunikasi yang lebih santai dan terbuka, peran ini menciptakan ruang diskusi yang nyaman bagi peserta magang untuk bereksplorasi dan mengatasi hambatan pekerjaan tanpa rasa sungkan. Melalui kontribusi asisten supervisor, tercipta lingkungan kerja yang kolaboratif dan kondusif yang sangat esensial dalam mendukung akselerasi kemampuan serta perkembangan profesional peserta selama masa magang.

3. ***Intern***

   Dalam ekosistem program PRO-Step career acceleration, peserta magang atau intern memegang peran krusial dalam mendukung kelancaran operasional tim serta pencapaian target proyek melalui pelaksanaan tugas yang sesuai dengan arahan supervisor maupun asisten supervisor. Peserta diwajibkan menjunjung tinggi etika kerja dan standar kualitas sebagaimana regulasi yang telah disepakati, termasuk disiplin dalam mendokumentasikan daily task untuk ditinjau dan disetujui secara resmi oleh supervisor. Proses ini tidak hanya melatih kemampuan manajemen waktu dan tanggung jawab secara mandiri, tetapi juga mengasah keterampilan praktis serta komunikasi efektif yang relevan dengan bidang studi masing-masing. Melalui

pengalaman ini, intern mendapatkan kesempatan berharga untuk membangun jaringan profesional sekaligus memberikan kontribusi nyata dalam penyelesaian proyek-proyek strategis di instansi terkait.

3.1.2 Koordinasi

Hubungan kerja dalam program PRO-Step *career acceleration* ini berjalan melalui alur komunikasi yang sangat dinamis dan saling mendukung. Koordinasi dimulai dari supervisor sebagai penentu arah proyek yang memberikan instruksi strategis, namun tetap membuka ruang diskusi lewat pertanyaan-pertanyaan pemancing logika agar saya sebagai *intern* bisa berpikir lebih kritis. Di sisi lain, asisten supervisor berperan sebagai jembatan komunikasi yang lebih fleksibel; tempat saya berdiskusi dengan suasana yang lebih santai saat menemui kendala teknis di lapangan. Pola komunikasi ini menjadi semakin teratur karena adanya kewajiban mengisi *daily task* yang dipantau langsung sebagai bentuk pertanggungjawaban kerja. Dengan adanya sinergi yang baik antara arahan formal dari supervisor, bantuan praktis dari asisten supervisor, dan kedisiplinan saya dalam mengeksekusi tugas, setiap hambatan dalam pengerjaan proyek bisa diselesaikan dengan cepat dan efektif.

3.2 Tugas yang Dilakukan

Berisi tabel hal-hal yang penulis lakukan selama menjalankan program.

Table 3.1 Detail Pekerjaan yang Dilakukan

No.	Minggu	Proyek	Keterangan
1	1	*Business Understanding :* Pendiskusian dan penerimaan project klasifikasi sentiment laporan pengaduan dari fitur LAKSA pada aplikasi Tangerang LIVE	Pengidentifikasian permasalahan dan tujuan penelitian

No.	Minggu	Proyek	Keterangan
2	1-2	*Data understanding : import* data, informasi data	Memerhatikan data penelitian
3	3-6	Data Preparation : *data integration, data cleaning, feature engineering, text preprocessing*	Mempersiakan dan membersihkan data agar data siap di analisis
4	7-8	*Modeling* : SVM	Melakukan modeling dengan algoritma SVM beserta 3 kernel yakni RBF, polynomial dan linear
5	9-10	*Modeling* : *Random Forest*	Melakukan modeling dengan algoritma *Random Forest*
6	11-12	*Modeling* : *Naïve Bayes*	Melakukan modeling dengan algoritma *Naïve Bayes*
7	13	*Evaluation*	Evaluasi masing-masing model dengan *confusion matrix*
8	13-14	*Deployment* : *dashboard*	*Dashboard yang*

3.3 Uraian Pelaksanaan Kerja

Berdasarkan tabel 3.1, dicantumkan penjelasan detail pelaksanaan kerja secara ringkas dengan keterangan waktu dan keterangan hasil dari tahapannya. Namun, untuk tahap penjelasan yang lebih mendalam, terdapat bagian proses pelaksanaan kerja. Bagian proses pelaksanaan kerja akan menjelaskan secara rinci setiap tahapan yang dilakukan dengan menyertakan dokumen dan penjelasan.

3.3.1 Proses Pelaksanaan

Proses pelaksanaan kerja terbagi menjadi beberapa tahapan, diantaranya :

3.3.1.1 Business Understanding : Pendiskusian dan penerimaan project klasifikasi sentiment laporan pengaduan dari fitur LAKSA pada aplikasi Tangerang LIVE

Diskominfo Kota Tangerang bertugas membuat *website* maupun aplikasi bagi seluruh Instansi Dinas Pemerintahan Kota Tangerang. Salah satu aplikasi buatan Diskominfo Kota Tangerang merupakan aplikasi yang bisa disebut *superapp* yakni Tangerang LIVE. Aplikasi Tangerang LIVE menjadi *superapp* dikarenakan memiliki memiliki 12 layanan dan 34 menu. Banyaknya layanan dan menu yang tersedia bermanfaat agar masyarakat Kota Tangerang bisa mendapatkan layanan berupa adminstrasi kependudukan, laporan pengaduan masyarakat, kepengurusan pembayaran pajak daerah dan zakat. Salah satu fitur yang banyak digunakan dan memberikan banyak respon dari masyarakat merupakan fitur LAKSA (Layanan Aspirasi Kotak Saran Anda). Fitur LAKSA menghasilkan banyak sekali respon berupa pengaduan. Banyaknya pengaduan yang diterima menimbulkan asumsi internal Diskominfo Kota Tangerang seperti sebagian besar laporan pengaduan yang diterima termasuk kedalam kategori negatif. Diskominfo sendiri belum pernah melakukan berupa analisis data untuk mengklasifikasikan sentimen untuk membuktikan asumsi internal tersebut. Klasifikasi sentimen sangat penting dilakukan karena pengaduan yang masuk pada fitur LAKSA layaknya ulasan atas kinerja pemerintah Kota Tangerang dalam melaksanakan tugasnya.

Banyaknya tugas yang harus dilakukan oleh Diskominfo membuat pembuktian asumsi internal ini selalu tertunda. Dengan demikian, Diskominfo memberikan tugas analisis data untuk mengklasifikasikan sentimen ini sebagai tugas yang harus diselesaikan selama kerja magang.

Untuk membantu mengerjakan tugas projek ini, pemagang dibantu oleh *supervisor* dari mulai hal sesederhana penggunaan algoritma. *Supervisor* memberikan arahan untuk mencari jurnal yang rilis 5 tahun terakhir dan *relate* dengan klasifikasi sentimen sebanyak mungkin dan mencatat beberapa poin dari masing-masing jurnal salah satunya merupakan algoritma yang digunakan. Dari 60-70 jurnal yang dikumpulkan dan dibaca, pemagang menemukan bahwa terdapat tiga algoritma yang paling banyak digunakan, yakni *naïve bayes*, SVM, dan *random forest. Naïve bayes* digunakan karena dinilai bisa menghitung probabilitas suatu kelas sentiment berdasarkan kemunculan kata-kata dalam teks secara efisien dan terukur secara statistik. SVM sendiri banyak digunakan dikarenakan kemampuannya dalam menemukan *hyperplane* optimal dalam memisahkan kelas sentiment pada data teks berdimensi tinggi. Sedangkan *random forest* banyak digunakan karena kemampuannya dalam menggabungkan banyak *decision tree* untuk menghasilkan prediksi yang akurat. Dengan demikian tujuan dari penelitian dan algoritma yang membantu menyelesaikan penelitian telah ditetapkan.

3.3.1.2 Data understanding : import data, informasi data

Pada tahap *data understanding* dilakukan dalam beberapa hal untuk memperoleh *insight* terhadap dataset laporan pengaduan dari fitur LAKSA pada aplikasi Tangerang LIVE. Data yang diberikan merupakan data yang di *extract* oleh mitra Diskominfo secara langsung. Setelah menerima file XLSX berisikan data mentah hasil laporan pengaduan, tahap data understanding dimulai dengan *mengimport dataset* dan memasukannya kedalam *database*.

```python
import pandas as pd
from sqlalchemy import create_engine
import pymysql

db_user = 'root'
db_password = ''
db_host = 'localhost'
db_name = 'sa_pengaduan'

conn_string = f"mysql+pymysql://{db_user}:{db_password}@{db_host}/{db_name}"
engine = create_engine(conn_string)
print("Koneksi ke database berhasil!")

file_path = r"C:\Users\maura\Documents\Semester 7\MBKM 2 - Internship\pengaduan2022-2024.xlsx"

df = pd.read_excel(file_path)

df.to_sql('pengaduan2022-2024', con=engine, index=False, if_exists='replace')

print("Data dari file Excel berhasil dimasukkan ke database!")
```

Gambar 3.2 Import data

Code pada gambar 3.2 menunjukkan bahwa alur penelitian pada tahap *data understanding* dimulai dengan menyiapkan koneksi ke database MySQL lokal menggunakan *library* SQLAlchemy dan PyMySQL agar data bisa dipindahkan secara otomatis. Setelah koneksi berhasil, sistem membaca file Excel berisi data pengaduan tahun 2021-2024 yang tersimpan di direktori komputer menggunakan bantuan *library* Pandas. Data yang sudah terbaca tersebut kemudian dikonversi dan dimasukkan ke dalam tabel database bernama 'sa_pengaduan' dengan skema pembaruan otomatis jika tabel sudah ada. Proses ini diakhiri dengan munculnya notifikasi di layar yang menandakan bahwa seluruh data dari file Excel telah sukses bermigrasi ke dalam database.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10585 entries, 0 to 10584
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id_pengaduan      10585 non-null  int64
 1   id_user           10585 non-null  int64
 2   nik               10465 non-null  object
 3   nama              10585 non-null  object
 4   jenis_kelamin     10465 non-null  object
 5   tgl_pengaduan     10585 non-null  datetime64[ns]
 6   tgl_proses        10480 non-null  object
 7   tgl_selesai       10281 non-null  object
 8   status_pengaduan  10585 non-null  object
 9   jenis_aduan       10585 non-null  object
 10  kategori          10582 non-null  object
 11  isi_pengaduan     10585 non-null  object
 12  lokasi            10546 non-null  object
 13  lat               10567 non-null  object
 14  lng               10567 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(11)
memory usage: 1.2+ MB
```

Gambar 3.3 Informasi dataset

Pada gambar 3.3, dapat dilihat bahwa setelah mengimport data, dilakukan pencaritahuan mengenai informasi dataset sehingga ditemukan *insight* bahwa data terdiri dari 10.585 entri (baris) dan 15 kolom (fitur). Informasi penting selanjutnya merupakan kolom "Non-Null Count" mengungkap adanya data yang hilang (*missing values*). Terlihat bahwa kolom-kolom vital seperti isi_pengaduan (10.582 non-null), lokasi (10.546 non-null), dan tgl_proses (10.480 non-null) memiliki jumlah data yang lebih sedikit dari total entri (10.585), yang mengindikasikan bahwa penanganan nilai NaN akan menjadi langkah wajib. Selanjutnya kolom "Dtype" menyoroti adanya inkonsistensi tipe data yang serius. Meskipun tgl_pengaduan sudah benar terbaca sebagai datetime64[ns], kolom tgl_proses dan tgl_selesai—yang esensial untuk menghitung durasi respons—secara keliru dibaca sebagai tipe object (string).

3.3.1.3 Data Preparation : data integration, data cleaning, feature engineering, text preprocessing

```
# pengubahan tipe data

df['lokasi'] = df['lokasi'].astype(str)
df['isi_pengaduan'] = df['isi_pengaduan'].astype(str)
df['jenis_aduan'] = df['jenis_aduan'].astype(str)

df['tgl_pengaduan'] = pd.to_datetime(df['tgl_pengaduan'], format='%d/%m/%Y %H:%M', errors='coerce')

# Kolom 2: Format ISO (YYYY-MM-DD HH:MM:SS) - Perbaikan untuk tgl_proses
df['tgl_proses'] = pd.to_datetime(df['tgl_proses'], format='%Y-%m-%d %H:%M:%S', errors='coerce')

# Kolom 3: Format ISO (YYYY-MM-DD HH:MM:SS) - PERBAIKAN UNTUK TGL_SELESAI
df['tgl_selesai'] = pd.to_datetime(df['tgl_selesai'], format='%Y-%m-%d %H:%M:%S', errors='coerce')

```

Gambar 3.4 Pengubahan tipe data

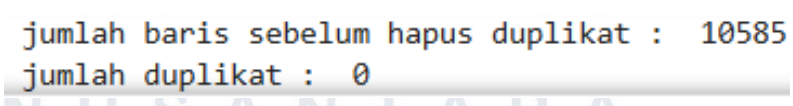
Gambar 3.4 menunjukkan *line* 3 sampai dengan *line* 5 yang merupakan *code* untuk mengubah tipe data. Ketiga *line* tersebut mengubah kolom-kolom yang berisi teks utama untuk analisis, seperti lokasi, isi_pengaduan, dan jenis_aduan menjadi tipe data *string*. Hal ini bertujuan memastikan bahwa semua data di dalamnya diperlakukan sebagai teks yang sama, sehingga proses-proses pengolahan bahasa selanjutnya dapat dijalankan tanpa menimbulkan error.

Line 7, 10, dan 12 juga menunjukkan dilakukanya proses pengubahan tipe atau format data. Sebelum diganti, kolom-kolom tersebut merupakan kolom tgl_proses dan tgl_selesai yang terbaca sebagai teks biasa (object), sehingga tidak mungkin digunakan untuk perhitungan. Maka dari itu, kolom-kolom tersebut dikonversi menjadi format tanggal datetime yang sesungguhnya. Ditemukan juga bahwa data mentah masih tidak konsisten, sehingga format tanggal yang berbeda-beda harus ditentukan secara manual agar semua data tanggal dapat dibaca dengan benar. Selanjutnya pengaturan seperti errors='coerce' juga diterapkan pada proses konversi tanggal ini. Penerapan ini memastikan jika ada data tanggal yang formatnya rusak, hilang, atau salah ketik, data tersebut akan secara otomatis diubah

menjadi nilai kosong (NaT atau Not a Time) dan tidak akan menghentikan jalannya program.

Gambar 3.5 memfilter kolom yang tidak dibutuhkan

Proses penghapusan kolom-kolom yang tidak diperlukan ditunjukkan pada Gambar 3.5. Langkah ini sangat penting untuk dilakukan karena memiliki dua tujuan utama yang berbeda. Yakni untuk melindungi privasi pelapor dan mengurangi kolom yang dinilai kurang relevan untuk penelitian. Kolom-kolom seperti id_user, nik, dan nama secara eksplisit dihapus dari dataset karena berisikan Informasi Identitas Pribadi (PII) atau data sensitif yang tidak boleh disertakan dalam analisis untuk menjaga kerahasiaan dan anonimitas penuh dari setiap individu yang laporannya digunakan. Kolom jenis_aduan juga turut dihapus karena dinilai tidak relevan untuk tujuan utama penelitian, yaitu klasifikasi sentimen. Sehingga fokus utama dari pemodelan adalah untuk memahami sentimen (positif, negatif, atau netral) yang terkandung dalam kolom isi_pengaduan tetap terjaga.



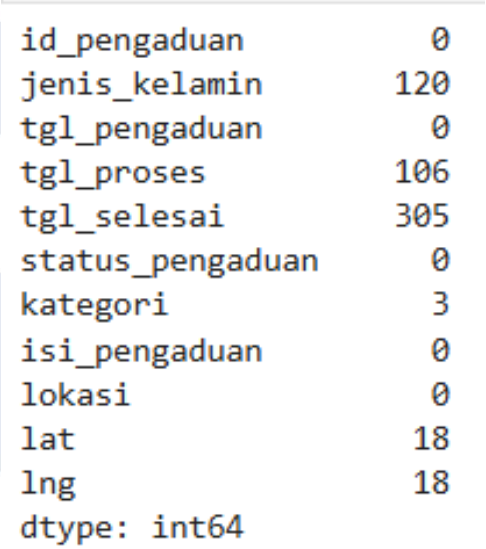
Gambar 3.6 Pengecekkan duplikasi data

Gambar 3.6 menunjukkan bahwa jumlah total baris data yang sedang diolah adalah 10.585 entri dan tidak ditemukan adanya duplikasi data. Tidak adanya duplikasi merupakan temuan yang paling krusial. Hal ini mengindikasikan bahwa setiap baris data dalam dataset mentah

yang diterima dari Diskominfo adalah unik. Proses pembersihan data dapat dilanjutkan ke tahap berikutnya, yaitu penanganan nilai yang hilang (missing values).

Gambar 3.7 Pengecekkan nilai null

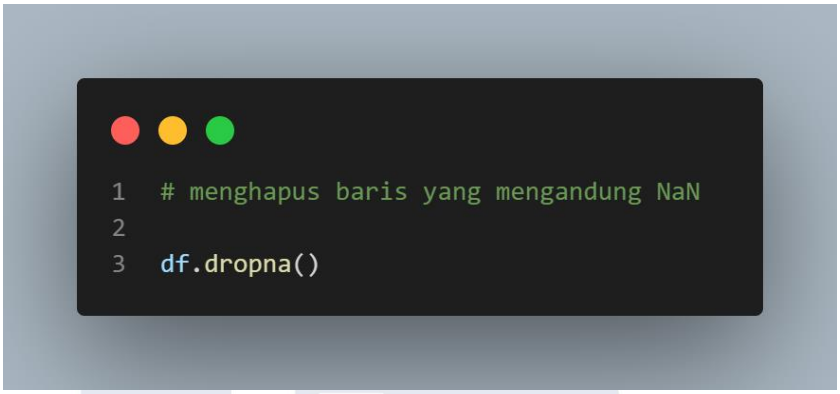
Gambar 3.7 menunjukkan "Jumlah *None* atau null : 0" (nol). Tidak ada nilai kosong yang bertipe *object None* yang tersembunyi di dalam dataset. Dengan tidak ada nilai *None*, jenis nilai hilang yang perlu ditangani adalah nilai NaN (*Not a Number*), yang keberadaannya telah teridentifikasi pada langkah df.*info*() sebelumnya.



Gambar 3.8 Pengecekkan nilai NaN

Hasil dari eksekusi *code* df.*isna*().*sum*() disajikan pada Gambar 3.8, yang memberikan wawasan diagnostik yang sangat penting untuk perencanaan pembersihan data. Pada gambar 3.8 dapat terlihat bahwa kolom inti yang akan digunakan untuk analisis sentimen, yaitu isi_pengaduan, berada dalam kondisi sempurna dengan 0 (nol) nilai

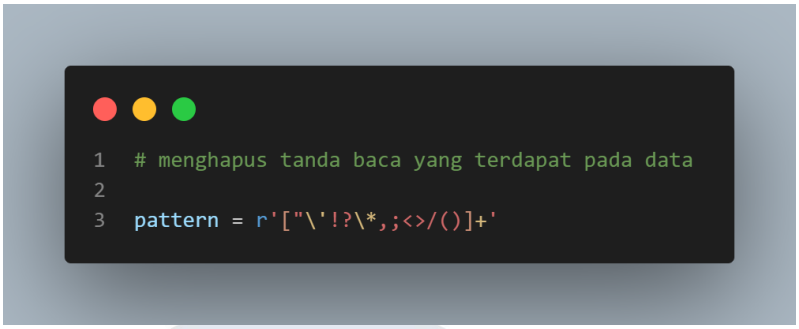
hilang. Output 4.13 menunjukkan bahwa kolom isi_pengaduan siap melalui proses analisis selanjutnya.



Gambar 3.9 Menghapus nilai NaN

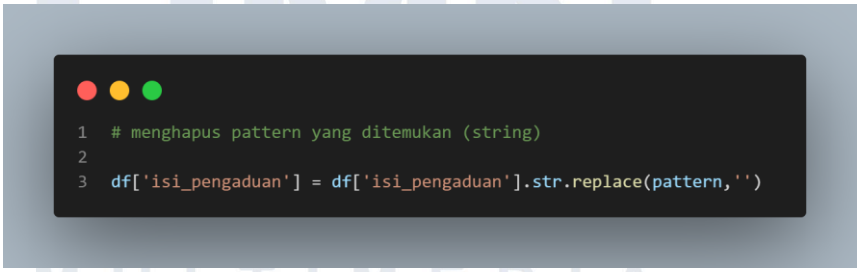
Pada Gambar 3.9, metode .*dropna*() pada *line* 3 gambar 3.9 diterapkan pada keseluruhan *Dataframe*. Langkah ini merupakan langkah penting dalam fase *Data Preparation* yang dirancang untuk memindai setiap baris data satu per satu. Jika dalam satu baris ditemukan NaN atau NaT (nilai kosong), maka keseluruhan baris tersebut akan dihapus secara permanen dari dataset.

Tahap *data preparation* selanjutnya dilakukan dengan *text preprocessing*. *Text preprocessing* merupakan salah satu tahapan terpenting karena tujuan dari dilakukannya penelitian ini merupakan menganalisis isi pengaduan yang masuk. Pembersihan teks pengaduan sangat perlu dilakukan agar pengaduan yang dianalisis tidak lagi mengandung unsur ambiguitas akibat singkatan atau istilah yang baru ditemukan.



Gambar 3.10 Pendefinisian tanda baca yang akan dihapus

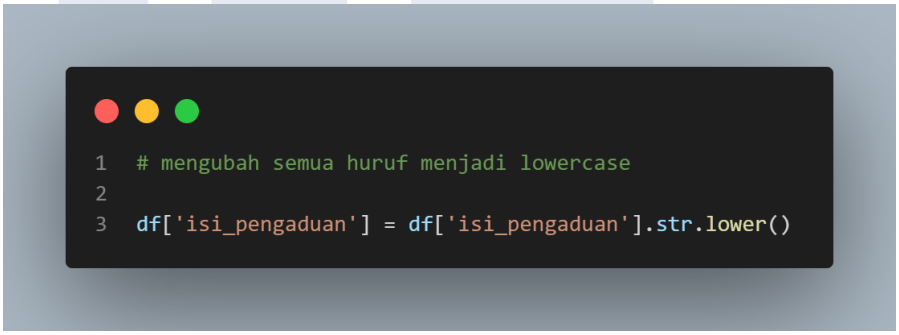
Proses pada gambar 3.10 diawali dengan pendefinisian sebuah *pattern* berbasis *Regular Expression (Regex)* yang disimpan dalam variabel pattern. Awalan r digunakan untuk menandakan *raw string*, sehingga seluruh karakter di dalam pola dibaca secara literal oleh mesin regex. Pola tersebut mencakup berbagai tanda baca seperti tanda kutip, koma, tanda seru, tanda tanya, titik, serta tanda kurung, dengan simbol + yang menunjukkan pencarian satu atau lebih kemunculan karakter secara berurutan. Karakter yang sesuai dengan pola kemudian dihapus dengan menggantinya menjadi string kosong, sehingga kolom isi_pengaduan diperbarui dengan versi teks yang telah bersih dari tanda baca.



Gambar 3.11 Eksekusi pembersihan tanda baca

Line 3 pada gambar 3.11menunjukkan kolom isi_pengaduan pada *Dataframe* diproses menggunakan fungsi *str.replace*() untuk menggantikan seluruh karakter yang sesuai dengan pola Regex yang telah didefinisikan sebelumnya. Metode ini bekerja secara vektorisasi, sehingga penghapusan tanda baca dilakukan secara serentak pada

seluruh baris data tanpa perlu iterasi manual. Parameter *pattern* berfungsi sebagai acuan karakter apa saja yang akan dihapus, sedangkan *string* kosong ('') menunjukkan bahwa karakter tersebut dihilangkan sepenuhnya dari teks. Hasil pemrosesan kemudian langsung disimpan kembali ke kolom isi_pengaduan, sehingga nilai lama digantikan dengan teks yang telah dibersihkan. Pendekatan ini meningkatkan efisiensi pemrosesan data, terutama pada dataset berukuran besar. Dengan dihilangkannya tanda baca yang tidak relevan, data teks menjadi lebih siap untuk tahap praproses lanjutan seperti tokenisasi, pembobotan fitur, dan klasifikasi sentimen.



Gambar 3.12 lowercasing kolom isi_pengaduan

Setelah proses penghapusan tanda baca selesai, tahap *text preprocessing* dilanjutkan dengan penyeragaman huruf (*case folding*). Pada tahap ini, seperti ditunjukkan pada Gambar 3.12, kolom isi_pengaduan diproses menggunakan metode .str.lower() dari Pandas untuk mengonversi seluruh karakter alfabet menjadi huruf kecil. Proses ini diterapkan pada setiap baris teks, kemudian hasilnya menggantikan kolom isi_pengaduan sebelumnya sehingga seluruh data teks berada dalam format huruf yang konsisten.

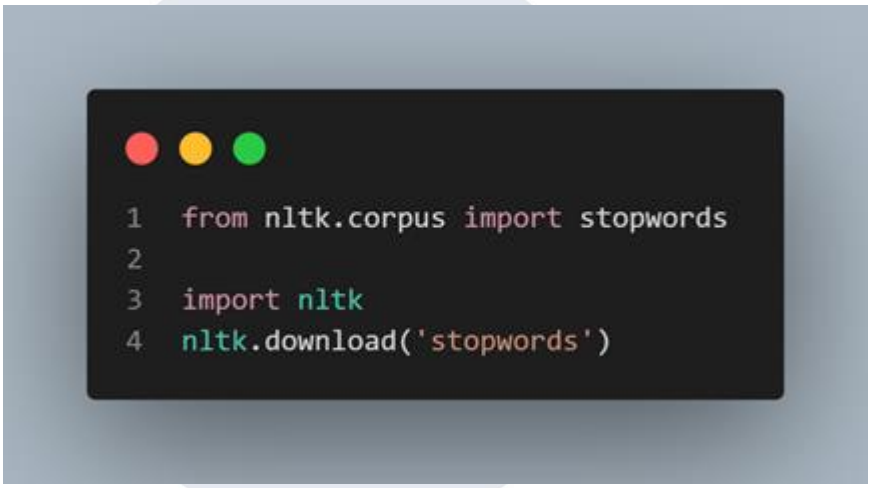
Gambar 3.13 Pustaka untuk tokenisasi

Tahap selanjutnya *text preprocessing* adalah mempersiapkan *library* pendukung untuk pemrosesan bahasa alami, yaitu NLTK (*Natural Language Toolkit*). Pada kode yang ditunjukkan pada Gambar 3.13, library NLTK diimpor terlebih dahulu agar fungsi-fungsi pemrosesan teks dapat digunakan dalam lingkungan pemrograman. Selanjutnya, perintah nltk.download('punkt_tab') digunakan untuk mengunduh *resource tokenizer* yang diperlukan oleh NLTK. *Resource* ini berperan penting dalam proses tokenisasi, yaitu pemecahan teks menjadi unit-unit yang lebih kecil seperti kata atau kalimat. Pengunduhan dilakukan satu kali dan disimpan secara lokal sehingga dapat digunakan kembali pada proses selanjutnya. Dengan tersedianya resource tokenizer ini, sistem siap melanjutkan ke tahap pemrosesan teks lanjutan seperti tokenisasi dan analisis fitur teks.

Gambar 3.14 Tokenisasi kata

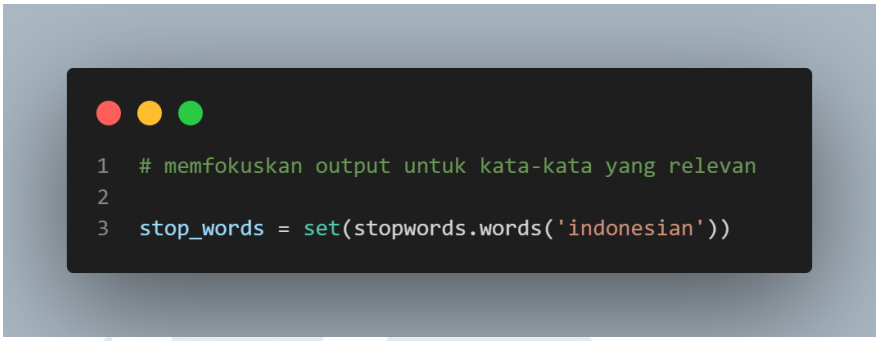
Fungsi yang digunakan pada baris pertama Gambar 3.14 diimpor dari modul nltk.tokenize. Selanjutnya, proses tokenisasi diterapkan pada seluruh kolom isi_pengaduan menggunakan metode .apply() pada

baris ketiga. Metode ini menjalankan sebuah fungsi lambda pada setiap baris teks secara berurutan. Melalui ekspresi lambda x: word_tokenize(x), setiap teks pengaduan diproses menjadi satuan kata menggunakan fungsi word_tokenize. Hasil dari tahap ini mengubah tipe data kolom isi_pengaduan dari string menjadi list yang berisi token kata individual.



Gambar 3.15 Pustaka untuk stopwords

Gambar 3.15 menunjukkan modul stopwords diimpor dari pustaka nltk.corpus untuk menyediakan akses ke kumpulan kata-kata umum yang sering muncul dalam teks. Import *library* NLTK dilakukan untuk memastikan seluruh fungsi pendukung pemrosesan bahasa alami dapat dijalankan dengan baik. Perintah *nltk.download('stopwords')* digunakan untuk mengunduh *resource stopwords* yang diperlukan apabila belum tersedia pada sistem. *Resource* ini berisi daftar *stopword* untuk berbagai bahasa, termasuk Bahasa Indonesia. Proses pengunduhan ini hanya perlu dilakukan satu kali dan hasilnya akan disimpan secara lokal. Dengan tersedianya kamus *stopwords* tersebut, sistem siap melakukan tahap penghapusan kata-kata tidak bermakna guna meningkatkan kualitas fitur teks pada proses analisis selanjutnya.



Gambar 3.16 Daftar stopwords

Setelah kamus *stopwords* dari NLTK berhasil diunduh, tahap berikutnya adalah memuat daftar stopword yang akan digunakan dalam proses pemrosesan teks. Seperti ditunjukkan pada Gambar 3.16, perintah stopwords.words('indonesian') pada baris ketiga digunakan untuk mengakses kamus stopword Bahasa Indonesia. Daftar kata umum yang dihasilkan, seperti *"yang"*, *"di"*, dan *"dan"*, kemudian dikonversi ke dalam struktur data *set* dan disimpan pada variabel stop_words untuk mendukung proses pemfilteran kata selanjutnya.

Gambar 3.17 Eksekusi stopwords

Potongan kode pada gambar 3.17 menunjukkan proses penghapusan *stopword* yang diterapkan langsung pada kolom isi_pengaduan menggunakan fungsi *apply*() dengan ekspresi lambda. Pada proses ini, setiap baris data yang telah berbentuk token dievaluasi satu per satu, kemudian dilakukan penyaringan kata berdasarkan dua kriteria utama. Kata hanya dipertahankan apabila

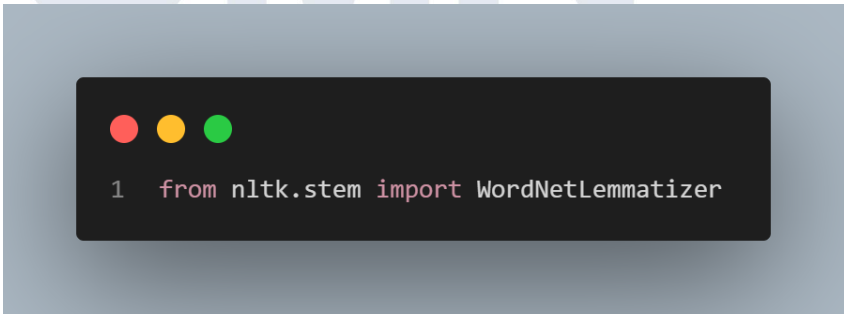
tidak termasuk ke dalam daftar stop_words serta memenuhi kondisi isalpha(), yaitu hanya terdiri dari karakter alfabet. Selain itu, penggunaan word.lower() memastikan bahwa proses pencocokan stopword bersifat konsisten tanpa dipengaruhi perbedaan huruf besar dan kecil. Hasil dari proses ini berupa daftar token yang telah bersih dari kata-kata umum dan karakter non-alfabet. Dengan demikian, kolom isi_pengaduan diperbarui dengan representasi teks yang lebih relevan dan informatif untuk tahap analisis sentimen selanjutnya.

Gambar 3.18 Customized Stopwords

Pada Gambar 3.18, didefinisikan sebuah daftar kata kustom yang disusun secara manual dan disimpan dalam variabel *customwords*. Daftar ini berisi kata-kata yang dianggap sebagai noise spesifik pada data pengaduan LAKSA, bukan *stopword* umum. Kata-kata tersebut meliputi singkatan sehari-hari, sapaan, istilah slang, serta istilah teknis yang sering muncul namun tidak memiliki kontribusi semantik terhadap analisis sentimen. Setelah daftar ini tersedia, dilakukan tahap pemfilteran lanjutan pada kolom isi_pengaduan setelah pemfilteran stopword standar. Melalui fungsi lambda pada baris ke-19, setiap token dievaluasi dan hanya dipertahankan apabila tidak termasuk dalam daftar *customwords*.

Gambar 3.19 Pustaka untuk wordnet

Dalam tahap *text preprocessing*, penggunaan teknik normalisasi kata memerlukan dukungan aset linguistik dari pustaka *Natural Language Toolkit* (NLTK). Langkah awal yang dilakukan, seperti terlihat pada Gambar 3.19, adalah memanggil fungsi nltk.download('wordnet'). Eksekusi perintah ini memastikan bahwa sistem memiliki akses ke dataset 'WordNet', sebuah referensi leksikal luas yang mengelompokkan kata ke dalam himpunan sinonim (*synsets*). Ketersediaan WordNet sangat vital karena menyediakan struktur hierarkis yang memungkinkan algoritma memahami akar kata berdasarkan konteks semantiknya, bukan sekadar pemotongan karakter secara kasar.



Gambar 3.20 Impor modul lemmatization

Setelah ketersediaan leksikal 'WordNet' sudah dipastikan, langkah selanjutnya yang dilakukan adalah menyiapkan perangkat lunak yang

akan memanfaatkan data tersebut dengan melakukan impor terhadap *class WordNetLemmatizer* dari modul nltk.stem, sebagaimana ditunjukkan pada Gambar 3.20. Tahapan impor ini berfungsi untuk memuat serta mendaftarkan definisi *class* tersebut ke dalam memori kerja (*RAM*) lingkungan Python. Langkah ini menjadi fondasi teknis yang krusial agar sebuah *object lemmatizer* sebagai instansi alat normalisasi kata yang fungsional dapat dibentuk dan diinisialisasi untuk memproses teks pada tahapan penelitian berikutnya.

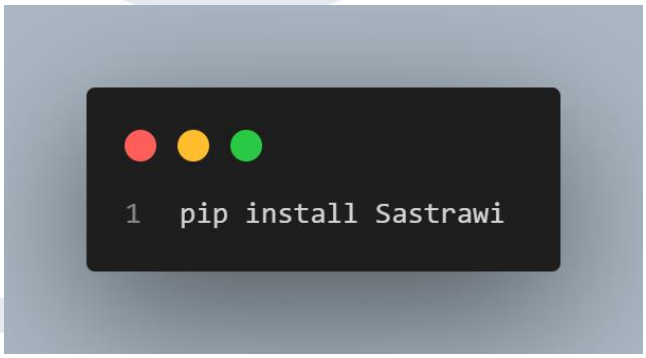
Gambar 3.21 lematization

Proses lematisasi pada Gambar 3.21 dilakukan melalui dua tahap utama. Pada tahap pertama, sebuah objek lematisasi diinisialisasi dari kelas *WordNetLemmatizer* dan disimpan dalam variabel WNL. Selanjutnya, objek tersebut diterapkan pada seluruh kolom isi_pengaduan menggunakan metode .apply(). Melalui fungsi lambda, setiap daftar token diproses dengan membangun ulang token-token baru, di mana setiap kata dilematiskan menggunakan metode WNL.lemmatize(). Hasil dari proses ini menggantikan kolom isi_pengaduan sebelumnya, sehingga seluruh token berada dalam bentuk kata dasar (lemma).

```
0        [tolong, perbaikan, peninggian, jalan, utama, ...
1        [live, taya, aplikasi, rsud, kota, buka, buka,...
2        [assalammualaikum, truk, pengangkut, sampah, s...
3           [bantuan, membersikan, sampahnya, terimakasih]
4        [izin, lapor, lampu, pju, iv, mati, khawatir, ...
                              ...
10579    [info, pembangunan, batako, jalan, jl, maulana...
10580    [tolong, ketegasan, penegak, perda, kota, tang...
10582    [kecamatan, kota, tangerang, bagun, puskeswan,...
10583    [jalur, listrik, pju, mati, penanganan, lapor,...
10584    [akibat, pelebaran, jalan, irigasi, kampung, g...
Name: isi_pengaduan, Length: 9973, dtype: object
```

Gambar 3.22 Hasil word lemmatization

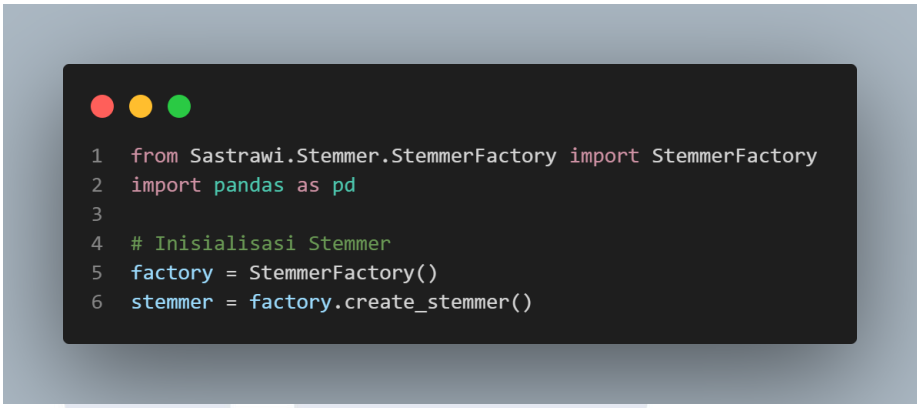
Gambar 3.22 memperlihatkan hasil transformasi kolom isi_pengaduan setelah tahap *word lemmatization*. Meskipun demikian, masih ditemukan sejumlah kata berimbuhan yang belum sepenuhnya disederhanakan ke bentuk dasar. Kondisi ini menunjukkan perlunya tahap *word stemming* sebagai proses lanjutan, dengan tujuan menghilangkan seluruh imbuhan kata sehingga diperoleh bentuk kata dasar yang lebih konsisten.



Gambar 3.23 Pustaka untuk stemming

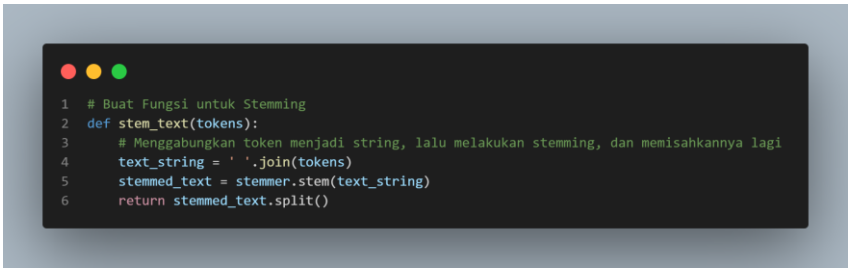
Sebagaimana divisualisasikan pada Gambar 3.23, dilakukan eksekusi perintah *pip install Sastrawi* yang berfungsi sebagai instruksi terminal kepada *package manager* standar Python, bukan sebagai baris kode pemrosesan di dalam *notebook*. Perintah ini menginstruksikan sistem melalui pip untuk membangun koneksi ke repositori resmi *Python Package Index* (PyPI) guna mengunduh serta menginstal pustaka eksternal Sastrawi ke dalam lingkungan kerja yang aktif. Integrasi

library Sastrawi ini sangat krusial karena menyediakan perangkat algoritma khusus untuk proses *stemming* bahasa Indonesia, yang memungkinkan sistem untuk mereduksi kata berimbuhan menjadi kata dasar secara akurat sesuai dengan kaidah linguistik yang berlaku.



Gambar 3.24 Inisialisasi pustaka

Sebagaimana diilustrasikan pada Gambar 3.24, tahapan *stemming* atau reduksi kata ke bentuk dasar dalam Bahasa Indonesia diawali dengan proses inisialisasi perangkat *stemmer* menggunakan pustaka Sastrawi. Langkah pertama melibatkan pemanggilan *class* utama *StemmerFactory* dari modul Sastrawi untuk bertindak sebagai kerangka pembangun, yang kemudian diinstansiasi menjadi sebuah objek *factory* pada baris kedua. Melalui objek *factory* tersebut, metode .create_stemmer() dieksekusi pada baris ketiga untuk memproduksi instansi *stemmer* yang fungsional dan siap pakai. Hasil akhir dari proses ini adalah sebuah objek pemroses teks yang disimpan dalam variabel stemmer, yang secara teknis telah siap digunakan untuk melakukan normalisasi kata berimbuhan menjadi kata dasar sesuai kaidah linguistik Indonesia pada tahapan pemrosesan data selanjutnya.



Gambar 3.25 Pembuatan fungsi stemming

Visualisasi pada Gambar 3.25 memaparkan penerapan fungsi kustom stem_text yang secara khusus dikembangkan untuk mengotomatisasi prosedur *stemming* melalui pustaka Sastrawi. Fungsi ini bekerja dengan menerima sekumpulan token teks sebagai input, menyatukannya kembali menjadi struktur kalimat, lalu mengeksekusi metode stemmer.stem() guna mereduksi setiap kata berimbuhan menjadi bentuk dasarnya sebelum akhirnya dikonversi kembali menjadi daftar token baru. Integrasi teknik *stemming* dan lemmatisasi ini merupakan langkah strategis dalam prapemrosesan data yang bertujuan untuk mencapai normalisasi morfologis sekaligus melakukan reduksi dimensi fitur agar data teks menjadi lebih terstruktur.

Dalam konteks Bahasa Indonesia, penggunaan afiks atau imbuhan sering kali menciptakan keberagaman bentuk kata dari satu akar yang sama. Contoh dari keberagaman yang dimaksud seperti pada variasi kata "lapor", "melapor", "dilaporkan", hingga "pelaporan". Apabila proses *stemming* diabaikan, algoritma *machine learning* akan mengidentifikasi setiap variasi tersebut sebagai fitur yang unik dan saling tidak berhubungan, yang pada akhirnya memicu masalah *curse of dimensionality* atau pembengkakan dimensi data secara berlebihan. Oleh sebab itu, *stemming* krusial dilakukan untuk menyatukan seluruh variansi morfologis ke dalam satu kata dasar (*root word*), sehingga model dapat lebih efektif dalam mengekstraksi esensi makna dan

meningkatkan kemampuan generalisasi dalam mengenali pola sentimen secara akurat.

```
# Terapkan fungsi stemming pada kolom 'isi_pengaduan'
df['isi_pengaduan_stemmed'] = df['isi_pengaduan'].apply(stem_text)
```

Gambar 3.26 Penerapan fungsi stemming

Maka dari itu diterapkanlah fungsi *stemming* seperti yang terlihat pada Gambar 3.26. Fungsi kustom stem_text diterapkan ke seluruh kolom isi_pengaduan menggunakan metode .apply(), di mana setiap daftar token diproses oleh stemmer Sastrawi hingga diperoleh kata dasar. Hasil normalisasi ini kemudian disimpan dalam kolom baru isi_pengaduan_stemmed, sementara kolom asli tetap dipertahankan sebagai data referensi.

```
# Konversi list menjadi tuple agar bisa di-hash dan dideteksi duplikasinya
df['isi_pengaduan_tuple'] = df['isi_pengaduan_stemmed'].apply(tuple)
```

Gambar 3.27 Penghapusan duplikasi terakhir

```
Jumlah baris sebelum menghapus duplikasi: 9973
```

Gambar 3.28 Output baris sebelum penghapusan duplikasi

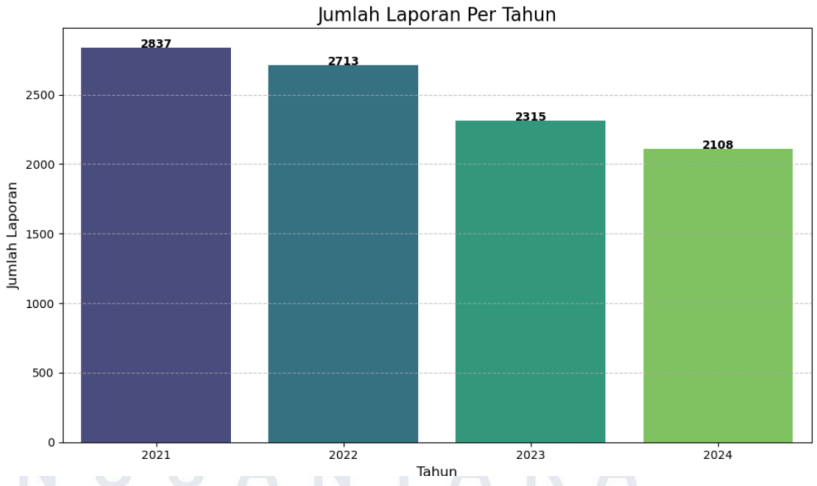
```
Jumlah baris setelah menghapus duplikasi: 9080
```

Gambar 3.29 Output setelah penghapusan duplikasi

Gambar 3.27 menunjukkan tahap penting berupa penghapusan duplikasi data untuk menetapkan jumlah awal baris yang akan dianalisis. Berdasarkan Gambar 3.28, jumlah data setelah proses data

cleaning dan text preprocessing tercatat sebanyak 9.973 baris, yang berasal dari 10.586 data mentah. Penetapan jumlah awal ini menjadi acuan untuk mengukur jumlah data yang tereliminasi akibat duplikasi, termasuk duplikasi semantik yang teridentifikasi setelah pembersihan teks. Setelah proses penghapusan duplikasi dilakukan, jumlah data yang tersisa adalah 9.080 baris, sebagaimana ditunjukkan pada Gambar 3.29.

Setelah seluruh rangkaian pembersihan serta *text preprocessing* rampung, dataset kini telah mencapai kondisi siap untuk memasuki tahap pemodelan. Namun, sebelum masuk tahap *feature engineering* dan pengembangan model, diperlukan tahap Analisis Data Eksploratif (EDA) untuk menggali karakteristik data secara lebih mendalam. Langkah awal yang krusial dalam tahap ini adalah memahami aspek demografis mendasar dari sumber data, yang dapat dimulai dengan meninjau akumulasi serta volume laporan yang masuk melalui fitur LAKSA dalam rentang waktu empat tahun terakhir.



Gambar 3.30 Jumlah laporan pengaduan yang masuk berdasarkan tahun

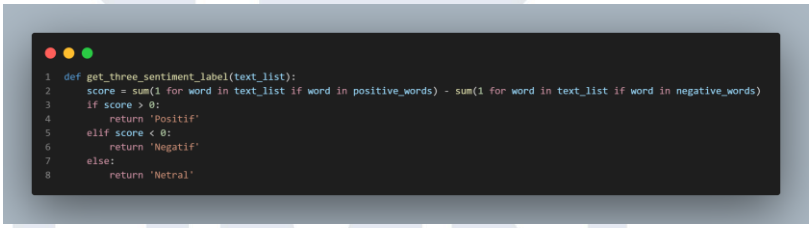
Berdasarkan gambar 3.30, dapat terlihat terjadi penurunan volume laporan pada fitur LAKSA yang terjadi secara gradual setiap tahunnya. Data menunjukkan bahwa pada tahun 2021 terdapat 2.837 laporan, namun jumlah tersebut menyusut hingga menyentuh angka

2.108 laporan pada tahun 2024. Penurunan intensitas pengaduan ini tidak diinterpretasikan sebagai melemahnya partisipasi masyarakat, melainkan sebagai indikator keberhasilan transformasi digital di internal Pemerintah Kota Tangerang. Fenomena ini didorong oleh semakin masifnya penyediaan aplikasi dan portal web khusus yang dikelola secara mandiri oleh tiap Organisasi Perangkat Daerah (OPD). Dengan adanya platform yang lebih tersegmentasi, masyarakat dapat menyampaikan keluhan secara langsung kepada instansi yang memiliki kewenangan spesifik, sehingga peran LAKSA bertransformasi menjadi portal pengaduan umum dan tidak lagi menjadi satu-satunya kanal rujukan utama. Diversifikasi saluran komunikasi ini pada akhirnya mendistribusikan beban laporan ke berbagai kanal digital lainnya, yang bertujuan untuk meningkatkan efisiensi birokrasi serta ketepatan penanganan masalah pada tingkat teknis.

Gambar 3.31 Data Labeling

Setelah data teks dinyatakan bersih dan ternormalisasi melalui proses stemming, penelitian dilanjutkan ke tahap Data Labeling sebagai bagian akhir dari fase Data Preparation. Pada tahap ini, seperti ditunjukkan pada gambar 3.31, disusun sebuah leksikon sentimen kustom yang dirancang khusus untuk domain pengaduan layanan publik LAKSA. Leksikon ini terdiri dari dua daftar utama, yaitu

positive_words dan negative_words, yang masing-masing berisi kata-kata dengan indikasi sentimen positif dan negatif yang relevan terhadap konteks pelayanan publik. Penyusunan leksikon dilakukan secara domain-specific berdasarkan kata-kata yang sering muncul dalam korpus data, mengingat beberapa istilah memiliki makna sentimen yang kuat dalam konteks pengaduan meskipun bersifat netral pada konteks lain. Leksikon ini digunakan sebagai dasar pseudo-labeling untuk secara otomatis menetapkan label sentimen Positif, Negatif, atau Netral pada seluruh 9.080 data pengaduan yang tidak memiliki label bawaan. Untuk mengantisipasi kata yang tidak tercakup dalam kamus (out-of-vocabulary), sistem secara otomatis mengklasifikasikan data tersebut ke dalam kelas Netral, yang secara metodologis sesuai karena laporan tanpa indikator sentimen kuat umumnya bersifat informatif atau administratif.



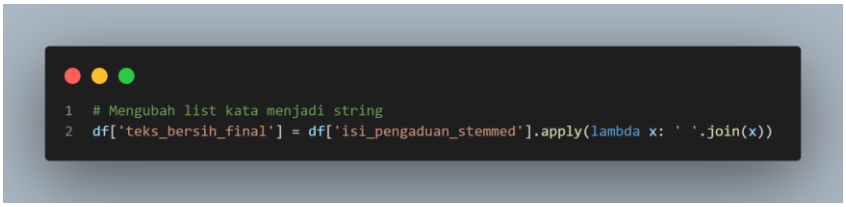
Gambar 3.32 Pendefinisian kategori klasifikasi

Setelah leksikon kustom positive_words dan negative_words disiapkan, tahap berikutnya adalah mendefinisikan fungsi logis untuk melakukan pseudo-labeling secara otomatis, sebagaimana ditunjukkan pada Gambar 3.32. Fungsi kustom get_three_sentiment_label pada *line* 1 dirancang untuk menerima masukan berupa text_list, yaitu daftar token hasil pembersihan dan stemming dari setiap pengaduan. Di dalam fungsi ini, sebuah variabel score dihitung dengan menjumlahkan kemunculan kata yang terdapat dalam positive_words sebagai nilai positif dan menguranginya dengan kemunculan kata dalam negative_words sebagai nilai negatif. Penetapan label sentimen dilakukan berdasarkan nilai akhir score, di

mana nilai lebih besar dari nol diklasifikasikan sebagai Positif, nilai kurang dari nol sebagai Negatif, dan nilai sama dengan nol sebagai Netral.

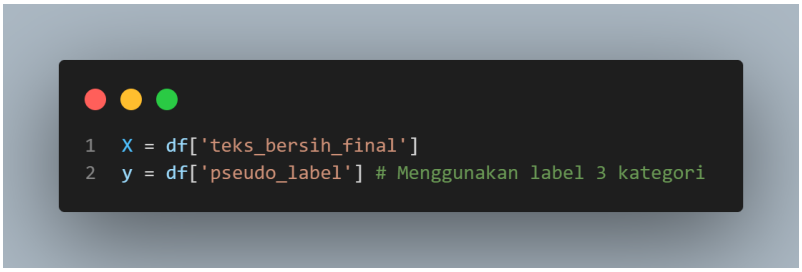
Gambar 3.33 Eksekusi Pseudo-Labeling

Sebagaimana divisualisasikan pada Gambar 3.33, dilakukan eksekusi fungsi get_three_sentiment_label yang menandai fase puncak dari seluruh rangkaian persiapan data (*Data Preparation*). Pada *line* 1, metode .apply() diimplementasikan pada setiap entri di kolom isi_pengaduan_stemmed yang mencakup 9.080 daftar token bersih guna mengkalkulasi skor polaritas melalui selisih antara total kata positif dan negatif, sehingga menghasilkan label 'Positif', 'Negatif', atau 'Netral'. Hasil pelabelan ini kemudian diintegrasikan ke dalam kolom baru bernama pseudo_label yang memegang peranan vital sebagai variabel target (*target variable*) dalam proses pelatihan (*training*) berbagai model *supervised learning*, seperti SVM, Random Forest, dan Naïve Bayes pada tahapan pemodelan. Sebagai langkah penutup di baris ketiga, perintah .value_counts() dijalankan untuk menyajikan distribusi total dari ketiga kategori tersebut, yang berfungsi sebagai validasi akhir guna meninjau proporsi data secara statistik sekaligus menjadi bukti empiris untuk mengonfirmasi atau menyangkal asumsi awal pihak Diskominfo terkait dominasi sentimen negatif dalam laporan masyarakat.



Gambar 3.34 Rekonstruksi teks menjadi kalimat

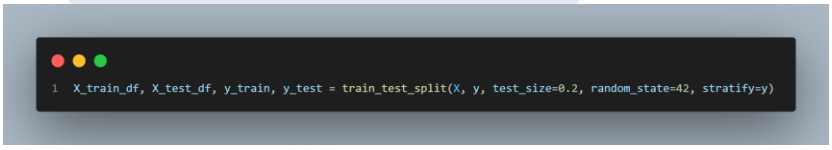
Sebagaimana divisualisasikan pada Gambar 3.34, dilakukan eksekusi tahapan final dari seluruh rangkaian prapemrosesan teks sebelum memasuki fase pemodelan. Setelah melewati proses pembersihan, tokenisasi, hingga normalisasi melalui *stemming*, data pada kolom isi_pengaduan_stemmed saat ini masih tersimpan dalam format daftar token kata individual. Mengingat perangkat *feature engineering* yang akan digunakan pada tahap berikutnya, yaitu TfidfVectorizer dari pustaka *Scikit-learn*, memerlukan masukan berupa korpus dokumen utuh dalam format *string*, maka diperlukan langkah rekonstruksi data. Melalui penerapan metode .apply() dan fungsi lambda, setiap daftar token digabungkan kembali menggunakan operasi ' '.join() menjadi satu kalimat utuh dengan spasi sebagai pemisah antar kata. Hasil rekonstruksi ini kemudian ditampung dalam kolom baru bertajuk teks_bersih_final, yang berperan sebagai *output* akhir dari fase persiapan data sekaligus menjadi input utama untuk proses ekstraksi fitur TF-IDF dan pelatihan algoritma *machine learning*.



Gambar 3.35 Pemisahan fitur (X) dan target (Y)

Sebagaimana diilustrasikan pada gambar 3.35, dilakukan tahapan transisi terakhir dari fase persiapan data menuju fase pemodelan

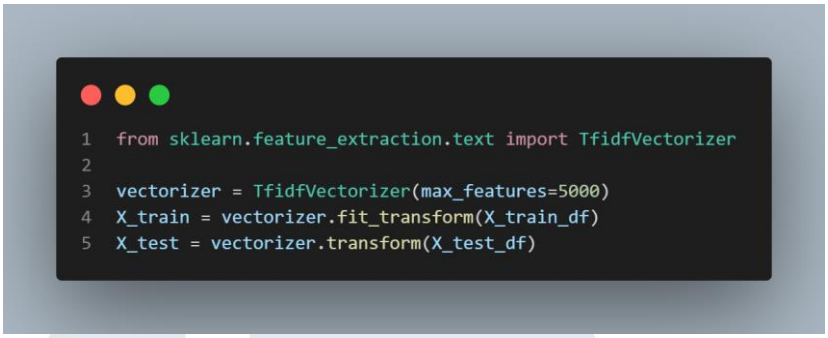
melalui pemisahan dataset df_cleaned menjadi dua komponen fundamental yang dibutuhkan oleh algoritma *machine learning*. Seluruh data teks yang telah melalui rekonstruksi pada kolom teks_bersih_final ditetapkan sebagai variabel X, yang dalam terminologi komputasi dikenal sebagai fitur atau variabel independen yang berfungsi sebagai dasar prediksi. Secara bersamaan, seluruh label sentimen pada kolom pseudo_label ditetapkan sebagai variabel y, yang berperan sebagai target atau variabel dependen yang akan diprediksi oleh model. Pemisahan ini merupakan langkah krusial karena kedua variabel tersebut nantinya akan diproses lebih lanjut menggunakan fungsi train_test_split untuk membagi data menjadi set pelatihan dan pengujian, sebelum akhirnya digunakan dalam proses induksi algoritma melalui perintah model.fit(X_train, y_train).



Gambar 3.36 Split dataset

Sebagaimana diilustrasikan pada Gambar 3.36, fase pemodelan dimulai dengan tahapan krusial berupa pembagian dataset secara proporsional. Menggunakan fungsi train_test_split dari pustaka *Scikit-learn*, variabel X (fitur teks) dan y (label sentimen) dipecah menjadi empat subset terpisah, yaitu data latih (X_train, y_train) dan data uji (X_test, y_test). Melalui pengaturan parameter test_size = 0.2, sistem mengalokasikan 80% dari total 9.080 baris data untuk proses pembelajaran model, sedangkan 20% sisanya dipreservasi khusus untuk mengevaluasi performa model terhadap data baru. Penggunaan parameter stratify=y menjadi aspek yang sangat vital dalam langkah ini guna memastikan distribusi ketiga kelas sentiment 'Positif', 'Negatif', dan 'Netral' agar tetap konsisten dan seimbang baik pada set

data latih maupun uji, sehingga hasil evaluasi model nantinya bersifat objektif dan terhindar dari bias kelas.



Gambar 3.37 Vektorisasi teks

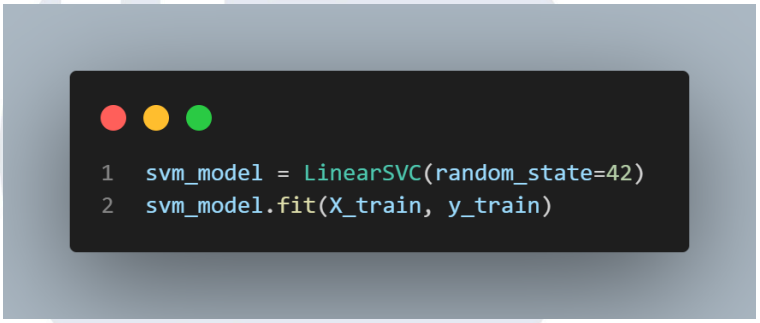
Setelah tahapan pembagian dataset menjadi set pelatihan dan pengujian selesai dilakukan, variabel X (X_train dan X_test) masih tersimpan dalam format *string* atau kalimat utuh yang tidak dapat diolah secara langsung oleh algoritma *machine learning*. Oleh karena itu, diperlukan prosedur *feature engineering* untuk mengonversi data tekstual tersebut ke dalam bentuk representasi vektor numerik yang dapat dikomputasi. Sebagaimana divisualisasikan pada Gambar 3.37, metode yang diterapkan dalam penelitian ini adalah TF-IDF (*Term Frequency-Inverse Document Frequency*) yang diintegrasikan melalui pustaka *Scikit-learn*.

Objek TfidfVectorizer diinisialisasi dengan parameter strategis max_features=5000 untuk melakukan seleksi fitur dengan cara membatasi kamus hanya pada 5.000 kata yang memiliki frekuensi kemunculan tertinggi, sehingga kata-kata langka yang dianggap sebagai *noise* dapat dieliminasi secara efektif. Selanjutnya, metode .fit_transform() diterapkan secara khusus pada X_train_df untuk mempelajari pola kosakata sekaligus menghitung bobot IDF (tingkat keunikan kata) sebelum akhirnya mengonversi teks pelatihan tersebut menjadi matriks vektor numerik. Sebagai langkah final yang krusial untuk mencegah terjadinya kebocoran data (*data leakage*), metode

.transform() digunakan pada X_test_df guna memastikan bahwa data uji ditransformasi secara konsisten menggunakan basis kamus dan pembobotan yang telah dipelajari dari data latih, tanpa melibatkan informasi baru dari set pengujian itu sendiri.

3.3.1.4 Modeling : SVM

Tahapan modeling SVM terdiri atas 2 tahapan dan digambarkan dalam code sebagai berikut



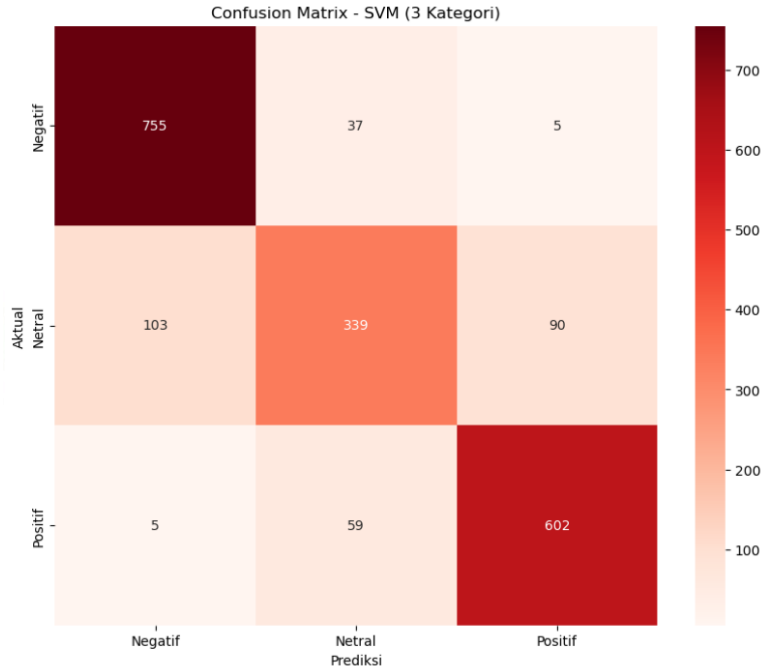
Gambar 3.38 SVM

Pada gambar 3.38 terlihat 2 tahapan modeling dengan algoritma SVM yakni

Inisialisasi. Objek model diinisialisasi menggunakan LinearSVC() untuk menguji kinerja kernel linear, yang dikenal sangat efektif dalam menangani data teks dengan dimensi tinggi. Guna menjaga aspek reproduksibilitas, parameter random_state=42 ditetapkan agar hasil eksperimen tetap konsisten dan dapat diulang kembali dengan parameter yang sama. Setelah objek svm_model tersebut siap, proses dapat dilanjutkan ke tahap pelatihan model.

Latihan. Proses pelatihan (*training*) dieksekusi melalui metode .fit(X_train, y_train), yang merupakan tahapan inti bagi model SVM untuk mempelajari data latih yang telah diverktorisasi TF-IDF beserta label sentimennya. Dalam fase ini, algoritma bekerja untuk menemukan *hyperplane* optimal yang mampu memisahkan ketiga kategori sentimen

secara akurat di dalam ruang vektor. Begitu proses ini tuntas, svm_model telah resmi "terlatih" dan memiliki kemampuan untuk melakukan prediksi pada data uji yang belum pernah diproses sebelumnya.



Gambar 3.39 Confusion Matrix SVM

```
--- Hasil Evaluasi SVM ---
Akurasi: 0.8501

Classification Report:
              precision    recall  f1-score   support

     Negatif       0.87      0.95      0.91       797
      Netral       0.78      0.64      0.70       532
     Positif       0.86      0.90      0.88       666

    accuracy                           0.85      1995
   macro avg       0.84      0.83      0.83      1995
weighted avg       0.85      0.85      0.85      1995
```

Gambar 3.40 Classification Report SVM

Berdasarkan hasil evaluasi awal pada Gambar 3.39 dan 3.40, model SVM dasar menunjukkan performa yang sedikit lebih unggul dibanding Random Forest dengan akurasi global mencapai 85,01%. Model ini sangat tangguh dalam mengenali sentimen 'Negatif' dan 'Positif' dengan masing-masing F1-Score sebesar 0,91 dan 0,88. Namun, kendala serupa kembali muncul pada kategori 'Netral', di mana model hanya mencatat recall 0,64 dan F1-Score 0,70. Matriks kebingungan menunjukkan bahwa dari 532 data netral, model gagal mengidentifikasi hampir dua ratus data di antaranya karena salah diklasifikasikan ke dalam kategori negatif maupun positif. Ketidakmampuan model dasar dalam membedakan sentimen netral secara presisi inilah yang menjadi alasan utama dilakukannya optimasi kernel untuk mencari konfigurasi SVM yang lebih optimal.

3.3.1.5 Modeling : Random Forest

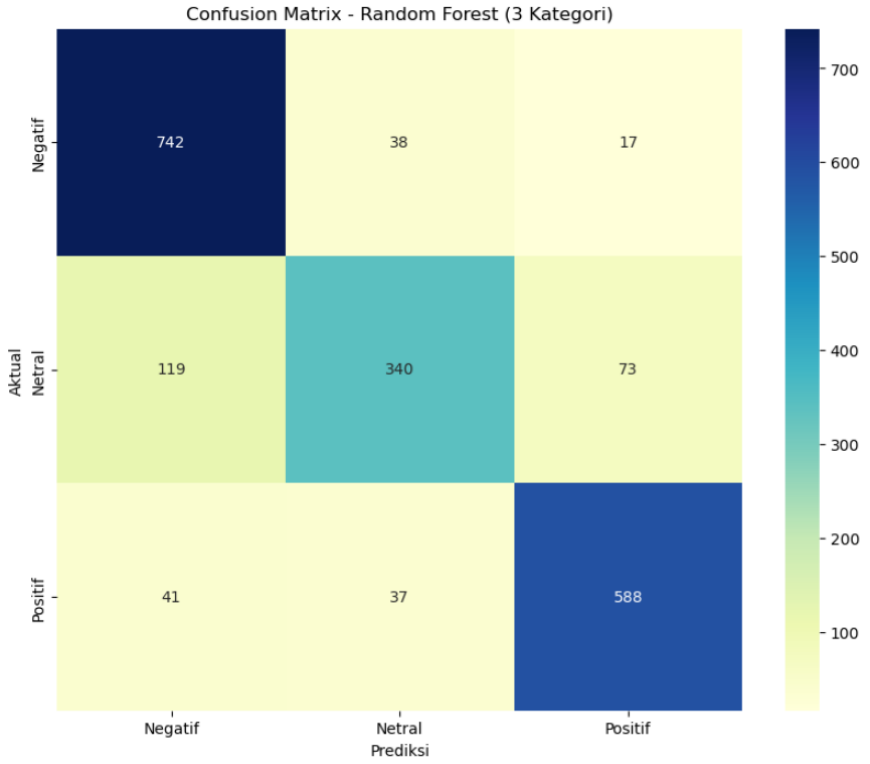
Tahapan modeling dengan algoritma *Random Forest* juga dilakukan dalam lebih dari satu tahap, sebagai berikut

Gambar 3.41 Inisialisasi dan Training Random Forest

Gambar 3.41 menunjukkan modeling *Random Forest* yang terdiri atas dua tahapan yakni inisialisasi dengan *RandomForestClassifier*() dengan menetapkan beberapa *hyperparameter* untuk mengontrol kinerja model dalam mempelajari data yang dijelaskan dalam bentuk tabel 3.1 sebagai berikut

Table 3.2 Hyperparameter Random Forest

Parameter	***value***	**Fungsi**
n_estimators	100	Menetapkan 100 sebagai jumlah pohon keputusan yang akan dibangun untuk memastikan kestabilan prediksi
class_weight	'*balanced*'	Parameter *class_weight* berfungsi memastikan kestabilan kelas agar bobot kelas tidak terpatok pada kelas mayoritas
random_state	42	42 sebagai *value* berfungsi memastikan bahwa hasil eksperimen tetap konsisten walaupun *code* dijalankan berulang kali.



Gambar 3.42 Confusion Matrix Random Forest

```
--- Hasil Evaluasi Random Forest ---
Akurasi: 0.8371

Classification Report:
              precision    recall  f1-score   support

     Negatif       0.82      0.93      0.87       797
      Netral       0.82      0.64      0.72       532
     Positif       0.87      0.88      0.88       666

    accuracy                           0.84      1995
   macro avg       0.84      0.82      0.82      1995
weighted avg       0.84      0.84      0.83      1995
```

Gambar 3.43 Classification Report Random Forest

Hasil evaluasi yang ditunjukkan pada classification report dan confusion matrix yang ditunjukkan gambar 3.42 dan 3.43 memberikan gambaran performa yang cukup kontras. Meski secara keseluruhan model mampu mencapai akurasi sebesar 83,71%, terdapat ketimpangan kinerja yang nyata di antara ketiga kelas sentimen, di mana model

sangat handal dalam mengenali sentimen 'Positif' (F1-Score 0.88) dan 'Negatif' (F1-Score 0.87) dengan tingkat keberhasilan prediksi yang tinggi. Namun, kelemahan fatal ditemukan pada kategori 'Netral' yang hanya mencatat nilai recall sebesar 0,64, karena model sering kali keliru mengategorikan laporan netral ke dalam sentimen positif atau negatif. Kegagalan visual pada confusion matrix ini menjadi catatan evaluasi utama bahwa meskipun Random Forest kuat dalam mendeteksi sentimen yang jelas, model ini masih mengalami kebingungan signifikan dalam membedakan laporan yang bersifat objektif atau netral.

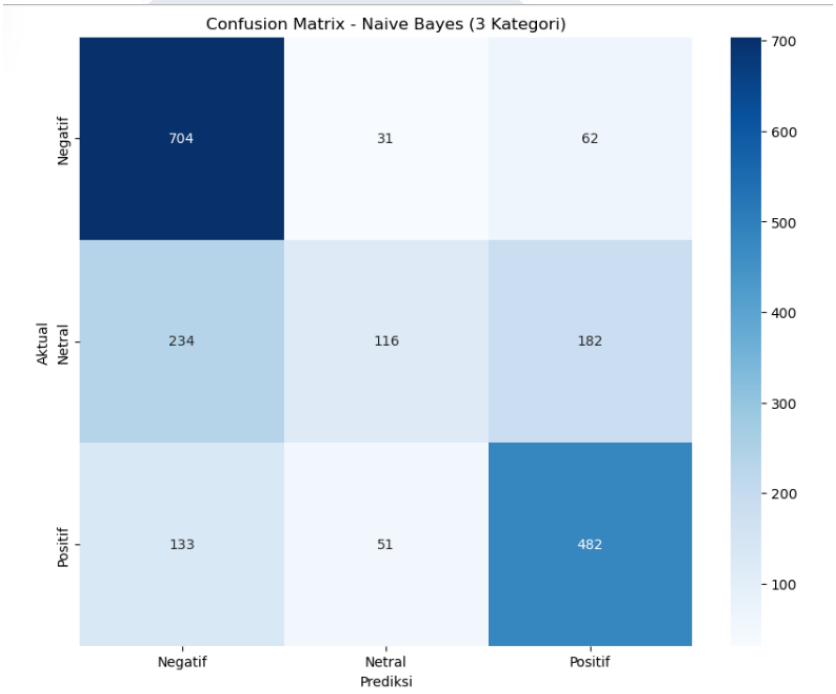
3.3.1.6 Modeling : Naïve Bayes

Sama seperti algoritma SVM dan *Random Forest*, pemodelan dengan naïve bayes juga terdiri atas 2 tahap yang gambarkan dalam *code* sebagai berikut

Gambar 3.44 Inisialisasi dan Training Naive Bayes

Code pada gambar 3.44 menunjukkan proses pemodelan dengan naïve bayes diawali dengan inisialisasi dan dilanjutkan dengan pelatihan model Naive Bayes untuk klasifikasi teks. Pada *line* 1, objek model dibuat menggunakan algoritma MultinomialNB(), yang merupakan varian Naive Bayes yang sangat populer dan efektif untuk data yang berbasis perhitungan kata (seperti hasil TF-IDF). Selanjutnya, pada *line*

kedua, perintah .fit(X_train, y_train) dijalankan untuk melatih model tersebut. Dalam proses ini, model mempelajari pola hubungan antara frekuensi kata dalam data latih (X_train) dengan label sentimen yang sesuai (y_train), sehingga model dapat memahami karakteristik bahasa yang merepresentasikan sentimen positif, negatif, atau netral untuk digunakan pada tahap prediksi.



Gambar 3.45 Confusion Matrix Naive Bayes

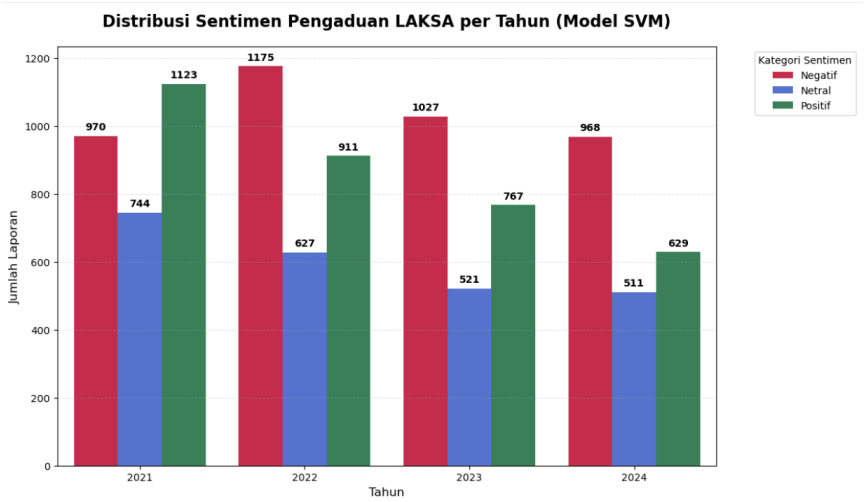
```
--- Hasil Evaluasi Naive Bayes ---
Akurasi: 0.6526

Classification Report:
              precision    recall  f1-score   support

     Negatif       0.66      0.88      0.75       797
      Netral       0.59      0.22      0.32       532
     Positif       0.66      0.72      0.69       666

    accuracy                           0.65      1995
   macro avg       0.64      0.61      0.59      1995
weighted avg       0.64      0.65      0.62      1995
```

Gambar 3.46 Classification Report

Berdasarkan hasil pengujian pada model Naïve Bayes yang disajikan dalam classification report dan confusion matrix gambar 3.45 dan 3.46, terlihat bahwa model ini memiliki performa yang paling rendah dibandingkan metode lainnya dengan akurasi global hanya sebesar 65,26%. Meskipun model masih cukup mampu mengenali sentimen 'Negatif' dengan recall 0,88, namun kualitas prediksi secara keseluruhan sangat tidak stabil karena banyaknya kesalahan klasifikasi di semua lini. Kelemahan paling mencolok terjadi pada kategori 'Netral' yang mencatatkan nilai recall sangat rendah, yakni 0,22 dengan F1-Score hanya 0,32. Visualisasi pada confusion matrix mempertegas kegagalan tersebut, di mana dari 532 data yang seharusnya 'Netral', hanya 116 data yang berhasil ditebak dengan benar, sementara sisanya justru tersebar atau salah dikira sebagai sentimen 'Negatif' (234 data) dan 'Positif' (182 data). Kondisi ini menunjukkan bahwa algoritma Naïve Bayes mengalami kesulitan besar dalam menangani ambiguitas kata pada data pengaduan LAKSA, sehingga sering kali gagal membedakan pesan yang bersifat netral dari pesan yang memiliki muatan emosional tertentu.

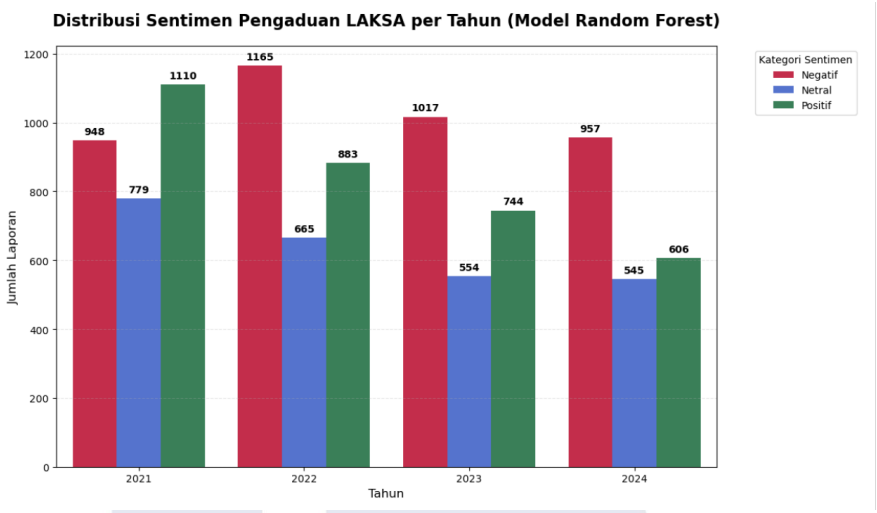
3.3.1.7 Evaluation

Berdasarkan hasil pengujian, model SVM terbukti memiliki kinerja yang lebih unggul dengan akurasi mencapai 87,37% dibandingkan model Random Forest yang berada di angka 83,71%, sementara Naïve Bayes menjadi metode dengan performa terendah di angka 65,26%. Meskipun SVM dan Random Forest menunjukkan hasil yang jauh lebih baik, keduanya masih menghadapi kendala fundamental yang sama, yaitu kesulitan dalam mengklasifikasikan kategori "Netral" secara akurat dengan *F1-Score* yang masih di bawah optimal. Hal ini disebabkan oleh tingginya tingkat ambiguitas bahasa dan *noise* tekstual pada data pengaduan, namun secara keseluruhan, SVM tetap menjadi pilihan yang paling tangguh dan stabil untuk memetakan sentimen publik dibandingkan dua algoritma lainnya



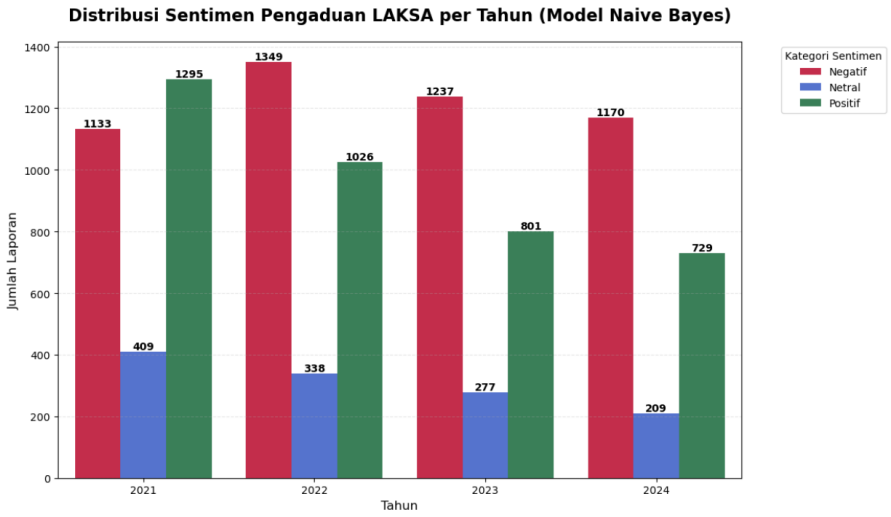
Gambar 3.47 Distribusi klasifikasi SVM

Berdasarkan visualisasi distribusi sentimen menggunakan model Support Vector Machine (SVM) pada Gambar 3.47, terlihat adanya dinamika sentimen yang cukup jelas dalam periode empat tahun. Pada tahun 2021, sentimen positif mendominasi dengan 1.123 laporan, namun pada tahun-tahun berikutnya terjadi pergeseran pola ke arah sentimen negatif, yang mencapai puncaknya pada tahun 2022 dengan 1.175 laporan. Karakteristik SVM yang memisahkan kelas melalui hyperplane optimal menghasilkan klasifikasi sentimen yang relatif tegas, meskipun kategori netral secara konsisten menunjukkan jumlah paling rendah dibandingkan sentimen positif dan negatif pada setiap tahun pengamatan.



Gambar 3.48 Distribusi Klasifikasi Random Forest

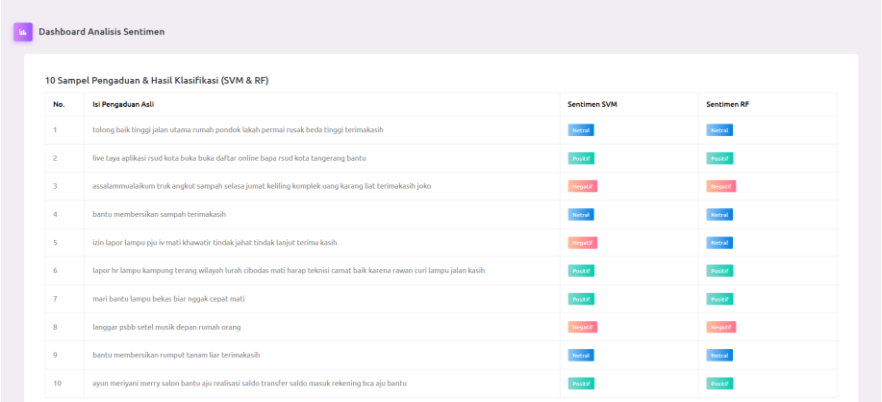
Pada model Random Forest, visualisasi pada Gambar 3.48 menunjukkan pola distribusi sentimen yang cenderung lebih konservatif dibandingkan SVM. Meskipun demikian, model ini tetap menangkap tren utama yang sama, yaitu dominasi sentimen positif pada tahun 2021 dengan 1.110 laporan serta peningkatan signifikan sentimen negatif pada tahun 2022 yang mencapai 1.165 laporan. Karakteristik ensemble learning pada Random Forest menghasilkan klasifikasi yang lebih stabil, khususnya pada kategori netral, yang mengindikasikan bahwa penggunaan banyak decision trees berkontribusi dalam mereduksi varians dan mencegah distribusi sentimen yang terlalu terpusat pada satu kelas ekstrem.



Gambar 3.49 Distribusi Klasifikasi Naive Bayes

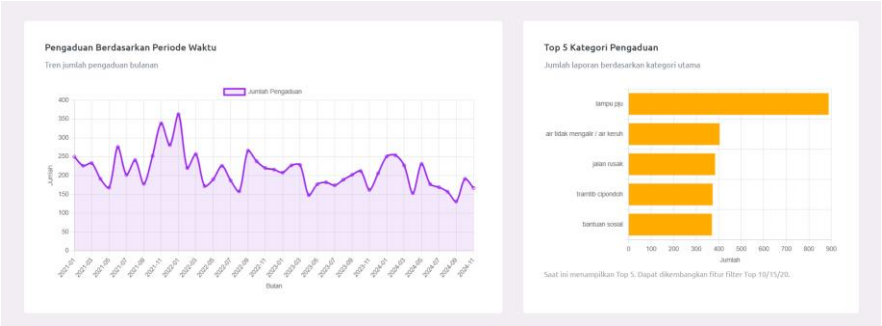
Berdasarkan visualisasi distribusi sentimen menggunakan algoritma Naive Bayes pada Gambar 3.49, terlihat pola klasifikasi yang relatif konsisten namun lebih sensitif terhadap perubahan distribusi data dibandingkan SVM dan Random Forest. Pada tahun 2021, sentimen positif menunjukkan jumlah tertinggi dengan 1.295 laporan, sementara sentimen negatif berada pada angka 1.133 laporan. Memasuki tahun 2022, sentimen negatif mengalami peningkatan signifikan dan menjadi kategori dominan dengan 1.349 laporan, diikuti oleh tren penurunan bertahap pada tahun 2023 dan 2024. Karakteristik Naive Bayes yang berbasis probabilistik dan asumsi independensi antar fitur menyebabkan model ini cenderung memberikan distribusi yang lebih tajam pada kelas mayoritas, sementara kategori netral secara konsisten memiliki jumlah paling rendah setiap tahunnya. Hal ini menunjukkan bahwa Naive Bayes cukup efektif dalam menangkap kecenderungan sentimen dominan, namun kurang fleksibel dalam menangani kelas dengan representasi fitur yang lebih ambigu.

3.3.1.8 Deployment : dashboard



Gambar 3.50 Tampilan dashboard (1)

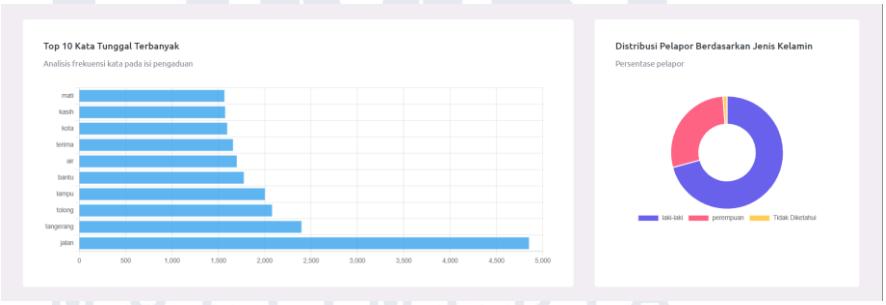
Visualisasi pada gambar 3.50 menampilkan sepuluh sampel data pengaduan beserta hasil klasifikasi sentimen menggunakan model *Support Vector Machine* (SVM) dan *Random Forest* (RF). Setiap baris merepresentasikan satu laporan pengaduan yang telah melalui proses *text preprocessing* dan pemodelan. Perbandingan hasil prediksi dari kedua model menunjukkan tingkat konsistensi yang relatif tinggi, meskipun terdapat beberapa perbedaan klasifikasi pada kasus tertentu. Perbedaan ini mencerminkan karakteristik masing-masing algoritma dalam mempelajari pola teks, di mana SVM cenderung lebih tegas dalam pemisahan kelas, sementara Random Forest lebih adaptif terhadap variasi fitur.



Gambar 3.51 Tampilan dashboard (2)

Grafik garis pada visualisasi gambar 3.51 menunjukkan jumlah pengaduan masyarakat berdasarkan periode waktu tertentu. Terlihat adanya fluktuasi jumlah laporan yang cukup dinamis, dengan beberapa periode mengalami peningkatan signifikan dan periode lain menunjukkan penurunan. Pola ini mengindikasikan bahwa intensitas pengaduan dipengaruhi oleh faktor temporal, seperti kondisi lingkungan, kebijakan publik, atau kejadian tertentu. Visualisasi ini penting untuk memahami tren pelaporan masyarakat dan dapat menjadi dasar evaluasi responsivitas layanan publik dari waktu ke waktu.

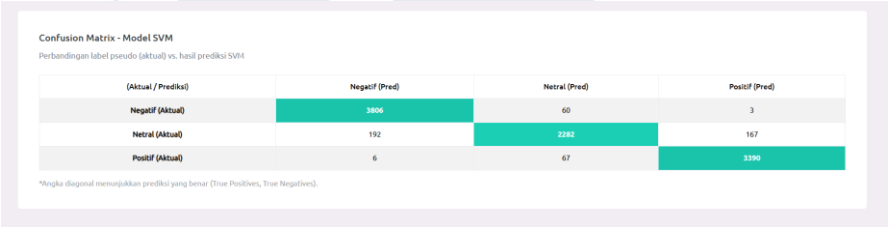
Selanjutnya diagram batang pada gambar 3.33 menggambarkan lima kategori pengaduan dengan jumlah laporan tertinggi. Terlihat bahwa beberapa kategori mendominasi secara signifikan dibandingkan kategori lainnya, yang menunjukkan area permasalahan utama yang paling sering dikeluhkan oleh masyarakat. Informasi ini memberikan gambaran prioritas isu layanan publik yang perlu mendapat perhatian lebih dari instansi terkait, serta dapat digunakan sebagai dasar pengambilan keputusan dalam perencanaan kebijakan dan alokasi sumber daya.



Gambar 3.52 Tampilan dashboard (3)

Visualisasi pada gambar 3.52 ini menampilkan sepuluh frasa yang paling banyak digunakan dalam bentuk diagram batang horizontal. Terlihat frasa “jalan” menjadi frasa paling banyak ditemukan dalam laporan. Dalam artian masalah infrastruktur menjadi permasalahan yang paling banyak dikeluhkan masyarakat.

Diagram donat pada visualisasi ini menunjukkan proporsi pelapor berdasarkan jenis kelamin. Distribusi yang relatif seimbang antara pelapor laki-laki dan perempuan mengindikasikan bahwa penggunaan aplikasi pengaduan LAKSA bersifat inklusif dan dimanfaatkan oleh berbagai kelompok masyarakat. Informasi demografis ini penting untuk memastikan bahwa sistem pengaduan publik dapat diakses secara merata dan tidak didominasi oleh kelompok tertentu saja.



Gambar 3.53 Tampilan dashboard (4)

Visualisasi *confusion matrix* pada model SVM pada gambar 3.53 menggambarkan performa klasifikasi sentimen dengan membandingkan label aktual dan label prediksi. Terlihat bahwa sebagian besar data berhasil diklasifikasikan dengan benar pada masing-masing kelas sentimen, khususnya pada kelas positif dan negatif. Kesalahan klasifikasi yang masih muncul, terutama pada kelas netral, menunjukkan adanya tumpang tindih karakteristik fitur antar kelas. Visualisasi ini memberikan gambaran yang jelas mengenai kekuatan dan keterbatasan model SVM dalam membedakan sentimen pada data pengaduan masyarakat.

3.3.2 Kendala yang Ditemukan

Terdapat beberapa kendala yang ditemukan seperti

1. **Terbatasnya akses akan data laporan pengaduan**

   Akses terhadap data laporan pengaduan sering kali menjadi kendala utama karena sifat data yang tertutup dan bersifat rahasia. Dinas

Komunikasi dan Informatika umumnya menerapkan kebijakan privasi yang ketat untuk melindungi identitas pelapor serta isi aduan yang bersifat sensitif, sehingga tidak semua informasi dapat dipublikasikan atau diakses oleh pihak eksternal, termasuk pemagang. Keterbatasan akses ini menyebabkan pemagang harus melalui proses komunikasi dan pengajuan data yang relatif panjang untuk memperoleh data yang sesuai dengan kriteria analisis. Selain itu, kebutuhan data sering kali bergantung pada bantuan staf internal untuk mengekstrak data dari basis data aplikasi Tangerang Live, yang mengakibatkan waktu pengambilan data menjadi lebih lama. Dalam beberapa kasus, data yang diberikan juga belum sepenuhnya sesuai dengan kebutuhan, misalnya terdapat kolom yang tidak diperlukan atau justru tidak menyertakan kolom yang dibutuhkan untuk keperluan analisis.

2. Waktu untuk mendapatkan data

   Proses perolehan data pengaduan juga membutuhkan waktu yang relatif lama dan berdampak pada efisiensi penelitian. Prosedur yang harus dilalui meliputi pengajuan permintaan data, penerimaan data yang belum sesuai dengan kebutuhan, hingga pengajuan permintaan ulang. Durasi tunggu yang tidak menentu ini menimbulkan jeda waktu yang cukup signifikan antara tahap perencanaan penelitian dan tahap pengolahan data. Oleh karena itu, diperlukan manajemen waktu yang lebih ketat agar jadwal penyelesaian skripsi tetap berada dalam target yang telah ditetapkan. Lamanya proses ini juga dipengaruhi oleh keterbatasan waktu staf internal, yang harus memprioritaskan tugas operasional harian, sehingga proses ekstraksi data dari sistem baru dapat dilakukan setelah pekerjaan utama mereka selesai.

3. Perbedaan istilah menghambat komunikasi

   Perbedaan pemahaman terhadap istilah atau konsep yang digunakan antara pemagang dan staf internal juga sering menjadi kendala. Perbedaan terminologi ini menyebabkan proses konsultasi maupun penyampaian perkembangan pekerjaan memerlukan waktu tambahan,

karena diperlukan penjelasan ulang hingga kedua pihak memiliki pemahaman yang sama terhadap konteks dan maksud yang dibahas. Akibatnya, komunikasi menjadi kurang efisien dan berpotensi memperlambat progres pekerjaan.

3.3.3 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala, ditemukan solusi berupa

1. **Bantuan supervisor untu mengakses data**

   Keberhasilan dalam memperoleh data laporan pengaduan tidak terlepas dari peran penting supervisor yang berperan sebagai penghubung dalam proses akses ke aplikasi terkait. Mengingat data pengaduan bersifat konfidensial, dukungan supervisor dalam memfasilitasi akses data sangat membantu dalam mengatasi berbagai kendala yang muncul. Melalui bantuan tersebut, data yang dibutuhkan untuk keperluan penelitian dapat diperoleh secara legal dan sesuai dengan prosedur yang berlaku. Selain itu, responsivitas supervisor dalam membantu proses akses basis data dan ekstraksi data secara langsung turut mempercepat penyelesaian hambatan yang dihadapi selama penelitian.

2. **Sering berkomunikasi dengan supervisor**

   Komunikasi yang intensif dan rutin dengan supervisor menjadi kunci utama dalam menjaga keberlangsungan dan progres penelitian. Melalui diskusi yang berkelanjutan, peneliti dapat segera berkonsultasi terkait kendala teknis maupun administratif yang ditemui, termasuk dalam proses pembersihan dan pengolahan data. Komunikasi yang efektif ini memastikan setiap tahapan penelitian tetap berjalan sesuai dengan rencana dan membantu peneliti dalam mengambil keputusan secara cepat ketika menghadapi hambatan selama pengumpulan data. Selain itu, pada saat pertemuan dengan supervisor, peneliti juga dapat menyampaikan kebutuhan data secara jelas dengan menyertakan batas waktu penyediaan data yang bersifat estimatif dan tidak menekan, sehingga proses koordinasi dapat berlangsung lebih efisien

3. **Memastikan memiliki pemahaman yang sama saat berdiskusi**

   Pada setiap sesi diskusi atau rapat, pemagang berupaya memastikan bahwa setiap penjelasan yang disampaikan, termasuk istilah teknis maupun konsep yang digunakan, dapat dipahami secara jelas oleh staf internal. Upaya ini dilakukan dengan menyesuaikan penggunaan bahasa, memberikan contoh konkret, serta mengonfirmasi kembali pemahaman bersama sebelum melanjutkan pembahasan ke tahap berikutnya. Pendekatan ini bertujuan untuk menyamakan persepsi sejak awal, sehingga proses diskusi dapat berjalan lebih efektif, meminimalkan miskomunikasi, dan mempercepat penyampaian informasi tanpa memerlukan penjelasan ulang yang berulang kali.