BAB 4
HASIL PENELITIAN

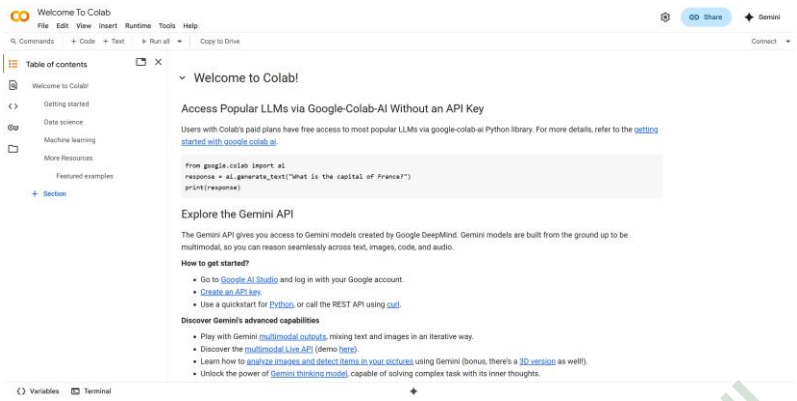
4.1 Pengambilan Data Google Play

Pengumpulan data ulasan dari Google Play dilakukan secara otomatis dengan memanfaatkan *library google-play-scraper*. Sebelum digunakan, *library* ini harus diinstal terlebih dahulu melalui perintah *pip*, seperti yang terlihat pada Gambar 4.1. Proses *scraping* menggunakan fungsi *reviews*() dengan memasukkan *app_id* dari aplikasi yang diinginkan. Data yang berhasil dikumpulkan meliputi nama pengguna, skor *rating*, waktu ulasan, dan isi komentar. Data tersebut kemudian dapat diolah menggunakan *library pandas* dan disimpan dalam format CSV. Metode ini memungkinkan pengambilan data tanpa perlu memanfaatkan API resmi dari Google.

Kode Program 4.1 Installasi *Library google-play-scraper*

```
!pip install google-play-scraper
```

Selain memanfaatkan *library google-play-scraper* untuk memperoleh data aplikasi beserta ulasannya, proses analisis sentimen juga memerlukan alat bantu tambahan, salah satunya adalah Google Colab yang berfungsi sebagai platform eksekusi kode *Python* secara daring. Google Colab dinilai efisien karena tidak memerlukan proses instalasi perangkat lunak, sehingga tidak membebani penyimpanan lokal. Platform ini cukup diakses menggunakan koneksi internet dan akun Google, serta mampu menyimpan hasil eksekusi kode secara otomatis. Selain itu, Google Colab memungkinkan penyimpanan dan berbagi skrip program melalui Google Drive. Perangkat lunak ini memiliki antarmuka yang menyerupai *Jupyter Notebook* versi *cloud* dan dapat dijalankan langsung melalui browser. Adapun tampilan dari Google Colab diperlihatkan pada Gambar 4.1.



Gambar 4.1 Tampilan *Google Collab*

Dalam penelitian ini, proses pemrograman menggunakan bahasa *Python* dijalankan melalui platform Google Colab. Pada tahap pengambilan data (*scraping*), dilakukan proses *import* terhadap data yang dibutuhkan, sebagaimana ditampilkan pada Kode Program 4.2. Data yang digunakan berasal dari ulasan pengguna terhadap aplikasi KAI Access yang diperoleh melalui Google Play, dan selanjutnya digunakan sebagai dasar dalam analisis sentimen.

Kode Program 4.2 *Import Library Scraping*

```
import pandas as pd
import numpy as np
from google_play_scraper import reviews_all, Sort
```

Kode Program 4.2 menunjukkan proses pendeklarasian beberapa *library*, yaitu *google-play-scraper* dan *pandas*. *Library google-play-scraper* berfungsi untuk mengambil informasi terkait aplikasi serta ulasan pengguna dari Google Play menggunakan bahasa pemrograman *Python*. Sementara itu, *pandas* merupakan *library Python* yang digunakan dalam pengelolaan dan manipulasi data berbentuk *DataFrame*. Kedua *library* ini tidak memerlukan proses pengunduhan tambahan karena telah tersedia secara *default* di dalam Google Colab. Adapun deklarasi kode untuk proses *scraping* data dilakukan sebagaimana diperlihatkan pada Kode Program 4.3, yang kemudian menghasilkan data seperti tampak pada gambar tersebut.

Kode Program 4.3 Proses *Scraping*

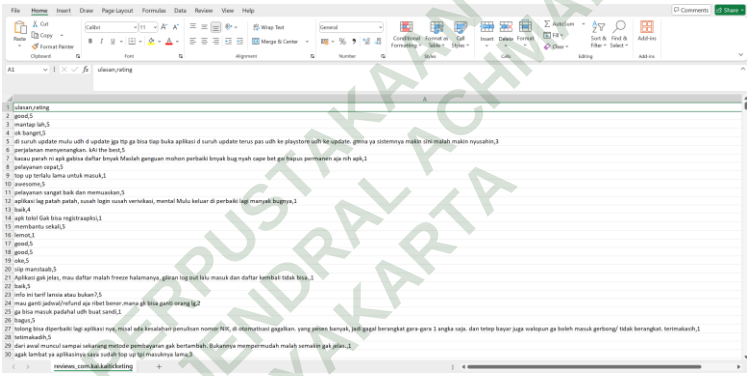
```
def scrape_google_play_reviews(app_id, lang='id', country='id'):
    """
    Mengambil semua ulasan untuk aplikasi tertentu dari Google
Play Store.
    """
    print(f"Mulai scraping ulasan untuk aplikasi: {app_id}")
    try:
        result = reviews_all(
            app_id,
            sleep_milliseconds=0,
            lang=lang,
            country=country,
            sort=Sort.NEWEST
        )

        df = pd.DataFrame(result)
        # Memilih kolom yang relevan
        df = df[['content', 'score']]
        # Mengganti nama kolom
        df.rename(columns={'content': 'ulasan', 'score':
'rating'}, inplace=True)

        # Simpan ke CSV
        nama_file = f'reviews_{app_id}.csv'
        df.to_csv(nama_file, index=False)
        print(f"Scraping selesai. Data disimpan di
'{nama_file}'")
        return df, nama_file
    except Exception as e:
        print(f"Terjadi kesalahan saat scraping: {e}")
        return None, None
```


Kode program 4.3 digunakan untuk melakukan proses *scraping* data ulasan dari Google Play Store terhadap aplikasi tertentu, dalam hal ini KAI Access, menggunakan bahasa pemrograman *Python*. Fungsi *scrape_google_play_reviews*() menerima parameter berupa *app_id* aplikasi, bahasa (*lang*), dan negara (*country*), dengan nilai default *'id'* untuk keduanya. Di dalam fungsi, digunakan metode *reviews_all* dari *library google-play-scraper* untuk mengambil seluruh ulasan berdasarkan aplikasi yang ditentukan. Hasil *scraping* kemudian dikonversi ke dalam format *DataFrame* menggunakan *library pandas*, dan hanya kolom *content* (isi ulasan) serta *score* (rating) yang dipilih untuk dianalisis lebih lanjut.

Setelah itu, kolom *content* diubah namanya menjadi ulasan, dan *score* menjadi rating agar lebih mudah dipahami. Data yang telah diproses kemudian disimpan dalam file CSV dengan format nama file *reviews_{app_id}.csv*, misalnya

reviews_com.kai.access.csv jika digunakan untuk aplikasi KAI *Access*. Proses ini dilakukan tanpa jeda waktu antar permintaan (*sleep_milliseconds*=0), sehingga pengambilan data berlangsung lebih cepat. Bila terjadi kesalahan selama proses *scraping*, sistem akan menangkap dan menampilkan pesan *error*. Fungsi ini juga mengembalikan dua nilai, yakni *Data Frame* hasil scraping dan nama file CSV yang berisi data ulasan.

Setelah menjalankan kode yang ditampilkan pada Kode Progam 4.3, dihasilkan sebuah file CSV yang berisi data mentah dari hasil *scraping*. Data tersebut belum melalui tahap pemrosesan lebih lanjut. Gambar 4.2 menampilkan tampilan awal dari hasil pengambilan data yang masih dalam bentuk asli.



Gambar 4.2 Hasil *Scraping* pada File CSV

Sebelum sampai pada tahap analisis, data terlebih dahulu diperoleh melalui proses *scraping* dari sumber Google Play, kemudian disimpan dalam format CSV. Proses ini menghasilkan dataset *berupa* kumpulan ulasan pengguna lengkap dengan nilai rating yang diberikan. File CSV tersebut kemudian dimuat ke dalam lingkungan pemrograman untuk dilakukan pengecekan struktur dan kelengkapan datanya. Gambar 4.3 menunjukkan hasil pembacaan file CSV tersebut, yang berisi 107.479 entri dengan dua kolom utama yaitu ulasan dan *rating*, tanpa adanya data kosong.

```
Index: 107479 entries, 0 to 107489
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   ulasan  107479 non-null  object
 1   rating  107479 non-null  int64 
dtypes: int64(1), object(1)
```

Gambar 4.3 Rincian Data

4.2 Pelabelan Data

Dalam proses analisis sentimen, tahap pelabelan data merupakan langkah penting yang menentukan kategori sentimen dari setiap ulasan pengguna. Pelabelan ini dilakukan secara otomatis berdasarkan nilai rating yang diberikan pengguna pada aplikasi KAI Access di Google Play Store. Sistem klasifikasi ini bertujuan untuk membagi ulasan ke dalam tiga kelas sentimen, yaitu negatif, netral, dan positif. Dengan pelabelan yang sistematis, proses klasifikasi data menjadi lebih efisien dan memudahkan dalam proses pelatihan model selanjutnya.

Kode Program 4.4 Pemberian Label Sentimen

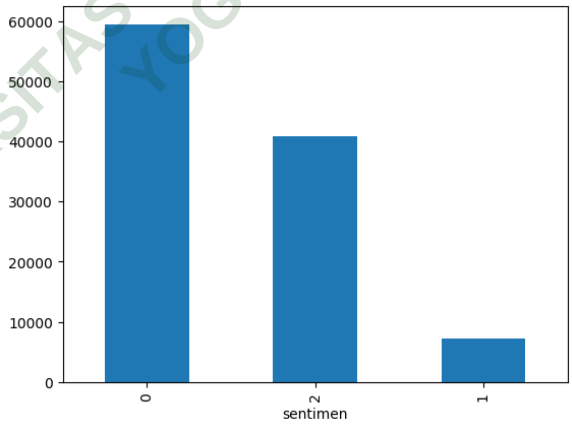
```python
def rating_to_sentiment(rating):
    if rating == 1 or rating == 2:
        return 0  # Negatif
    elif rating == 4 or rating == 5:
        return 2  # Positif
    elif rating == 3:
        return 1
    else:
        return -1

df['sentimen'] = df['rating'].apply(rating_to_sentiment)
```


Kode program 4.4 digunakan untuk mengonversi nilai rating ke dalam label sentimen. Fungsi *rating_to_sentiment*() akan mengembalikan nilai 0 jika *rating* adalah 1 atau 2, yang berarti sentimen negatif. Jika *rating* bernilai 4 atau 5, maka akan dikategorikan sebagai sentimen positif dengan label 2. Untuk *rating* 3, dikembalikan sebagai 1 yang menandakan sentimen netral. Selain itu, sistem juga memberikan nilai -1 jika ditemukan rating di luar kategori yang telah ditentukan, sebagai bentuk penanganan kondisi tidak terduga. Hasil dari fungsi ini kemudian diterapkan ke seluruh data rating melalui fungsi *apply*() pada kolom rating.

	ulasan	rating	sentimen
0	good	5	2
1	mantap lah	5	2
2	ok banget	5	2
3	di suruh update mulu udh d update jga ttp ga b...	3	1
4	perjalanan menyenangkan. kAi the best	5	2
...	...	...	...
1996	Good	5	2
1997	Keren	5	2
1998	Semoga segera di fix kan gangguan bug di app	5	2
1999	Sangat membantu	4	2
2000	TOP MARKOTOP>>>	5	2

Gambar 4.4 Hasil Pelabelan Data

Gambar 4.4 memperlihatkan hasil dari proses pelabelan data terhadap ulasan pengguna. Setiap baris ulasan yang awalnya hanya memiliki nilai rating, kini telah dilengkapi dengan label sentimen berdasarkan logika klasifikasi. Misalnya, rating 5 dikategorikan sebagai sentimen 2 (positif), sementara rating 3 dikategorikan sebagai 1 (netral). Hal ini menunjukkan bahwa proses pelabelan berhasil dilakukan secara otomatis dengan pendekatan berbasis nilai numerik.



Gambar 4. 5 Perbandingan Label Sentimen

Gambar 4.5 menampilkan distribusi jumlah ulasan berdasarkan hasil pelabelan sentimen. Terlihat bahwa kategori sentimen positif (label 2) dan negatif (label 0) mendominasi jumlah ulasan secara signifikan, sementara ulasan netral (label 1) hanya menempati sebagian kecil. Sentimen negatif merupakan kategori

terbanyak, yang menunjukkan banyaknya kritik atau keluhan dari pengguna. Meskipun begitu, jumlah sentimen positif juga cukup tinggi, menandakan bahwa sebagian besar pengguna tetap memiliki pengalaman yang baik. Visualisasi ini memberikan gambaran umum yang jelas mengenai persepsi pelanggan terhadap aplikasi, sekaligus menjadi indikator awal untuk evaluasi kualitas layanan digital KAI Access.

4.3 *Text Preprocessing*

Proses prapemrosesan data dilakukan melalui serangkaian tahapan yang dirancang untuk merapikan struktur ulasan yang tidak beraturan. Karena data yang dikumpulkan masih dalam bentuk mentah dan belum tersusun dengan baik, diperlukan langkah sistematis untuk menyeleksi serta menghilangkan unsur-unsur yang tidak diperlukan. Tujuan akhirnya adalah menghasilkan data yang bersih dan tersusun rapi sehingga layak digunakan dalam tahap klasifikasi selanjutnya. Dalam pelaksanaannya, tahapan *text preprocessing* dilakukan secara berurutan dan terstruktur agar seluruh teks telah mengalami proses pembersihan dan penyeragaman sebelum memasuki proses analisis.

4.3.1 *Cleaning*

Cleaning merupakan tahap awal dalam proses *text preprocessing* yang bertujuan untuk membersihkan data teks dari elemen-elemen yang tidak relevan. Pada tahap ini, teks ulasan dibersihkan dari berbagai komponen seperti simbol, angka, URL, tanda baca, dan spasi yang tidak perlu. Keberadaan elemen-elemen tersebut dapat mengganggu akurasi dalam proses analisis sentimen jika tidak dihilangkan. Tahap *cleaning* menjadi fondasi penting untuk menghasilkan data teks yang lebih rapi dan terstruktur. Dengan teks yang bersih, proses analisis selanjutnya dapat berjalan lebih optimal dan efisien.

Kode Program 4.5 *Import Library Cleaning*

```
import re
import string
```

Kode Program 4.5 berfungsi untuk mengimpor dua *library* penting yang digunakan dalam proses pembersihan teks. *Library re* digunakan untuk mengelola operasi pencocokan pola berbasis regular *expression*, seperti menghapus URL atau angka dari teks. Sementara itu, *string* menyediakan fungsi yang berkaitan dengan manipulasi karakter, termasuk penghapusan tanda baca.

Kode Program 4.6 Proses *Cleaning*

```
def cleaning_text(text):
    """Menghapus karakter khusus, angka, URL, dan spasi
berlebih."""
    text = re.sub(r'http\S+', '', text)  # Hapus URL
    text = re.sub(r'@[^\s]+', '', text) # Hapus mention
    text = re.sub(r'#\w+', '', text) # Hapus hashtag
    text = re.sub(r'\d+', '', text) # Hapus angka
    text = text.translate(str.maketrans('', '',
string.punctuation)) # Hapus tanda baca
    text = text.strip() # Hapus spasi di awal dan akhir
    text = re.sub(r'\s+', ' ', text) # Hapus spasi berlebih
    return text
```

Kode Program 4.6 mendefinisikan fungsi *cleaning_text*() yang dirancang untuk menghapus berbagai elemen yang tidak diperlukan dari teks ulasan. Pertama, fungsi ini menghapus URL menggunakan ekspresi reguler yang mendeteksi pola alamat web. Kemudian, *mention* seperti *@username* serta *hashtag* juga dihapus karena dianggap tidak relevan untuk analisis sentimen. Angka-angka dalam teks dihilangkan melalui pola pencarian numerik, diikuti dengan penghapusan seluruh tanda baca menggunakan fungsi dari modul string. Setelah itu, teks dibersihkan dari spasi yang berlebihan dan spasi di awal atau akhir kalimat. Hasil dari fungsi ini adalah teks yang lebih bersih, ringkas, dan siap digunakan dalam tahapan pemrosesan berikutnya. Proses pembersihan yang dilakukan melalui Kode Program 4.6 divisualisasikan pada Gambar 4.6 sebagai hasil akhir tahap *cleaning*.

```
Cleaning Text:
Input: 'top up terlalu lama untuk masuk'
Output: 'top up terlalu lama untuk masuk'
```

Gambar 4.6 Hasil Proses *Cleaning*

4.3.2 *Case Folding*

Setelah melalui tahap *cleaning*, proses dilanjutkan dengan *case folding* sebagai langkah kedua dalam *text preprocessing*. Tujuan utama dari *case folding* adalah menyeragamkan semua huruf dalam teks menjadi huruf kecil agar perbedaan kapitalisasi tidak memengaruhi analisis. Misalnya, kata "Top" dan "top" akan dianggap sama setelah melalui proses ini. Penyeragaman huruf ini penting karena dalam sistem komputasi, kata dengan huruf kapital dan tidak kapital akan dikenali sebagai entitas berbeda. Dengan menerapkan *case folding*, data menjadi lebih konsisten dan meminimalkan redundansi kata yang sebenarnya memiliki makna serupa.

Kode Program 4.7 Proses *Case Folding*

```
def case_folding(text):
    """Mengubah semua teks menjadi huruf kecil."""
    return text.lower()
```

Kode Program 4.7 menampilkan fungsi *case_folding*() yang berfungsi untuk mengubah seluruh karakter dalam teks menjadi huruf kecil. Proses ini dilakukan dengan metode *.lower*() yang merupakan fungsi bawaan pada Python untuk konversi huruf kapital ke huruf kecil. Penerapan metode ini sangat sederhana namun sangat penting untuk menghindari inkonsistensi kata yang diakibatkan oleh perbedaan kapitalisasi. Sebagai contoh, kata "Aplikasi", "APLIKASI", dan "aplikasi" akan diperlakukan sama setelah diformalkan menjadi huruf kecil. Kode Program 4.7 juga membantu mengurangi jumlah fitur unik dalam model analisis teks. Ini berguna untuk efisiensi ruang dan waktu komputasi, terutama saat membangun model klasifikasi. Hasil akhir dari proses *case folding* ditampilkan pada Gambar 4.7 yang memperlihatkan teks ulasan dalam format huruf kecil seluruhnya.

```
Case Folding:
Input: 'top up terlalu lama untuk masuk'
Output: 'top up terlalu lama untuk masuk'
```

Gambar 4.7 Hasil Proses *Case Folding*

4.3.3 *Tokenization*

Setelah teks diubah ke dalam bentuk huruf kecil melalui proses *case folding*, langkah berikutnya adalah *tokenizing*. *Tokenizing* bertujuan untuk memecah kalimat atau teks ulasan menjadi potongan-potongan kecil yang disebut token, biasanya berupa kata per kata. Proses ini penting karena algoritma pemrosesan bahasa alami bekerja lebih efektif ketika teks telah dipisah menjadi unit-unit terkecilnya. Dengan memecah teks menjadi token, sistem dapat mengenali setiap kata secara individual untuk dianalisis lebih lanjut. Tahapan ini juga mempermudah proses penghapusan *stopword* dan *stemming* pada tahap-tahap selanjutnya.

Kode Program 4.8 Proses *Tokenization*

```
def tokenize_text(text):
    """Memecah teks menjadi token (kata)."""
    return text.split()
```

Kode Program 4.8 mendefinisikan fungsi *tokenize_text*() yang digunakan untuk memecah teks menjadi token-token berdasarkan spasi antar kata. Fungsi ini menggunakan metode *.split*() dari *Python*, yang secara otomatis memisahkan kalimat setiap kali menemukan spasi. Dengan demikian, kalimat panjang dapat diuraikan menjadi daftar kata yang berdiri sendiri. Misalnya, kalimat “aplikasi ini sangat bagus” akan diubah menjadi [‘aplikasi’, ‘ini’, ‘sangat’, ‘bagus’]. Kode Program 8 sangat berguna dalam persiapan data untuk proses selanjutnya seperti *stopword removal* dan *stemming*. Token-token yang dihasilkan akan digunakan sebagai dasar dalam proses ekstraksi fitur. Hasil dari proses *tokenizing* ini divisualisasikan pada Gambar 4.8 yang menunjukkan bentuk teks setelah dipecah menjadi daftar kata.

```
Tokenizing:
Input: 'top up terlalu lama untuk masuk'
Output: ['top', 'up', 'terlalu', 'lama', 'untuk', 'masuk']
```

Gambar 4.8 Hasil Proses *Tokenizing*

4.3.4 *Stopword Removal*

Setelah teks dipecah menjadi token, langkah selanjutnya adalah menghapus *stopword*. *Stopword* merupakan kata-kata umum yang sering muncul namun tidak memberikan kontribusi signifikan terhadap analisis, seperti “yang”, “dan”, “di”, dan sebagainya. Kehadiran kata-kata ini dapat menambah noise pada data dan memperbesar dimensi fitur secara tidak perlu. Oleh karena itu, menghapus stopword menjadi langkah penting untuk menyederhanakan representasi data teks. Dengan mengurangi jumlah token yang tidak relevan, proses klasifikasi dapat dilakukan dengan lebih fokus dan efisien.

Kode Program 4.9 *Import Library Stopword Removal*

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

factory_stopword = StopWordRemoverFactory()
stopword_remover = factory_stopword.create_stop_word_remover()
```

Kode Program 4.9 digunakan untuk meng*impor* dan menginisialisasi *library Sastrawi*, yaitu pustaka yang umum digunakan dalam pemrosesan bahasa Indonesia. *StopWordRemoverFactory* merupakan bagian dari *library* tersebut yang berfungsi sebagai *generator* untuk alat penghapus *stopword*. Kode ini menghasilkan objek *stopword_remover* yang nantinya digunakan untuk menghapus kata-kata tidak penting dari teks. Dengan demikian, proses penghapusan *stopword* dapat dilakukan secara otomatis berdasarkan daftar kata yang telah ditentukan oleh pustaka *Sastrawi*.

Kode Program 4.10 Proses *Stopword Removal*

```
def remove_stopwords(tokens):
    """Menghapus kata-kata umum (stopwords) dari daftar
token."""
    # Sastrawi membutuhkan string, jadi kita gabungkan lalu
proses
    text = ' '.join(tokens)
    return stopword_remover.remove(text).split()
```

Kode Program 4.10 mendefinisikan fungsi *remove_stopwords*() untuk menghapus kata-kata umum dari daftar token. Karena pustaka *Sastrawi* hanya menerima input berupa string, maka daftar token terlebih dahulu digabungkan

menjadi satu kalimat utuh menggunakan ' '.*join*(*tokens*). Setelah teks tergabung, fungsi *stopword_remover.remove*() akan menghapus semua kata yang termasuk dalam daftar *stopword* bahasa Indonesia. Hasil dari penghapusan tersebut kemudian dipecah kembali menjadi token dengan metode *.split*(). Proses ini memungkinkan kita untuk mendapatkan daftar kata yang lebih bersih dan bermakna. Dengan mengurangi kata-kata yang tidak penting, model analisis akan lebih fokus pada fitur yang benar-benar relevan. Kode Program 4.10 sangat efektif dalam mengoptimalkan kualitas data teks sebelum digunakan dalam pelatihan model. Gambar 4.9 memperlihatkan hasil akhir dari proses *stopword removal* berupa token-token yang telah disaring dari kata-kata tidak penting.

```
Stopword Removal:
Input: ['top', 'up', 'terlalu', 'lama', 'untuk', 'masuk']
Output: ['top', 'up', 'terlalu', 'lama', 'masuk']
```

Gambar 4.9 Hasil Proses *Stopword Removal*

4.3.5 *Stemming*

Langkah terakhir dalam tahapan *text preprocessing* adalah *stemming*, yang bertujuan untuk mengubah setiap kata menjadi bentuk dasarnya. Proses ini sangat penting agar kata dengan makna serupa tetapi bentuk berbeda, seperti “berjalan”, “berjalanlah”, atau “berjalannya”, dapat dikenali sebagai satu kata dasar yaitu “jalan”. Dengan mengurangi variasi bentuk kata, proses analisis dapat dilakukan secara lebih efisien dan akurat. Stemming juga membantu mengurangi jumlah fitur dalam teks, sehingga mempercepat proses pelatihan model. Tahap ini sangat relevan dalam pengolahan bahasa Indonesia yang kaya dengan imbuhan dan variasi bentuk kata.

Kode Program 4.11 *Import Library Stemming*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory_stemmer = StemmerFactory()
stemmer = factory_stemmer.create_stemmer()
```

Kode Program 4.11 digunakan untuk mengimpor dan menginisialisasi modul *stemming* dari pustaka *Sastrawi*, yang secara khusus dirancang untuk bahasa Indonesia. *StemmerFactory* berfungsi untuk membuat objek *stemmer* yang akan

digunakan dalam proses konversi kata ke bentuk dasarnya. Dengan memanggil *create_stemmer*(), kode ini menghasilkan objek *stemmer* yang mampu melakukan *stemming* terhadap *string input*. Objek tersebut kemudian digunakan dalam fungsi pemrosesan selanjutnya.

Kode Program 4.12 Proses *Stemming*

```
def stem_text(tokens):
    """Mengubah setiap kata dalam token menjadi kata dasarnya
(stemming)."""
    # Sastrawi membutuhkan string, jadi kita gabungkan lalu
proses
    text = ' '.join(tokens)
    return stemmer.stem(text)
```

Kode Program 4.12 menjelaskan proses *stemming* dengan mendefinisikan fungsi *stem_text*() yang menerima *input* berupa token. Karena pustaka *Sastrawi* membutuhkan masukan berupa *string*, daftar token terlebih dahulu digabung menjadi satu kalimat menggunakan ' '.*join*(*tokens*). Selanjutnya, teks yang telah tergabung dikirim ke fungsi *stemmer.stem*() untuk diubah menjadi bentuk kata dasar. Setelah melalui proses *stemming*, kata-kata yang memiliki imbuhan akan direduksi ke bentuk dasar yang lebih umum. Ini sangat membantu dalam mengurangi kompleksitas data dan memastikan konsistensi dalam representasi kata. Proses ini juga penting untuk meningkatkan akurasi klasifikasi pada tahap selanjutnya. Kode Program 4.12 merupakan tahap final dalam proses pembersihan teks sebelum masuk ke proses ekstraksi fitur. Hasil *stemming* yang telah disederhanakan dapat dilihat pada Gambar 4.10 yang menampilkan bentuk kata dasar dari hasil token yang telah diproses sebelumnya.

```
Stemming:
Input: ['top', 'up', 'terlalu', 'lama', 'masuk']
Output: 'top up terlalu lama masuk'
```

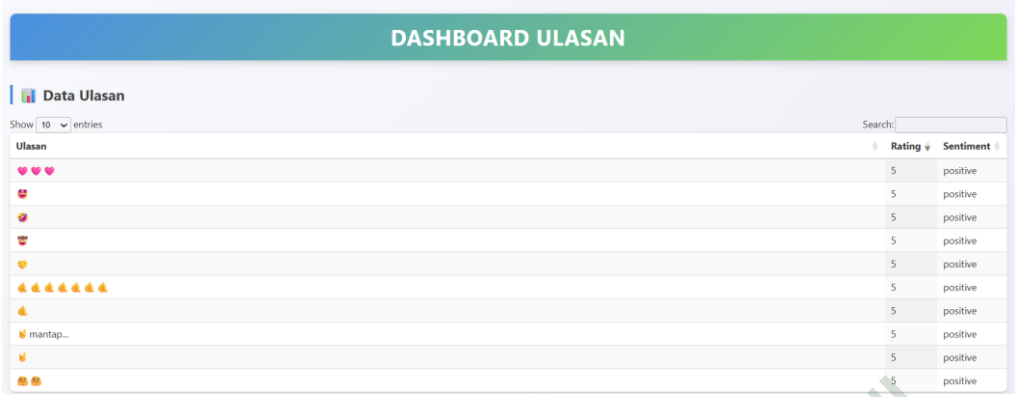
Gambar 4.10 Hasil Proses *Stemming*

4.3.6 *Dashboard*

Gambar berikut memperlihatkan *dashboard* hasil analisis sentimen ulasan pengguna aplikasi KAI Access yang diunduh dari Google Play Store. Data yang ditampilkan telah melalui tahap *preprocessing* dan klasifikasi menggunakan metode *Natural Language Processing* (NLP). Pada cuplikan ini, seluruh ulasan memiliki *rating* 1 - 5 dan dikategorikan sebagai sentimen negatif dan positif yang mengindikasikan tingginya tingkat kepuasan dan ketidakpuasan pengguna. Variasi ulasan yang mencakup teks, simbol, dan emoji menunjukkan beragam cara pelanggan menyampaikan pujian dan keluhan. Kehadiran fitur pencarian (*search*) dan pengaturan jumlah entri (*entries*) memudahkan proses eksplorasi dan analisis data secara cepat, sistematis, dan terstruktur.

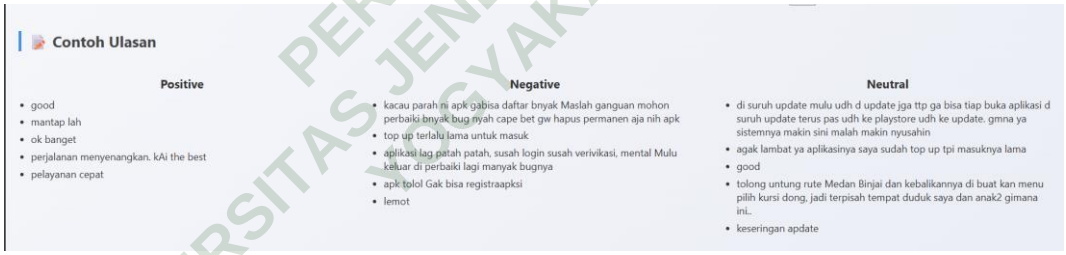
Gambar 4.11 Ulasan Pelanggan Negatif

dashboard ulasan pelanggan negatif pada aplikasi KAI Access, yang memperlihatkan hasil klasifikasi sentimen berbasis NLP dengan seluruh data pada cuplikan memiliki *rating* 1 dan berlabel negatif. Visualisasi ini memudahkan peneliti dalam mengidentifikasi pola keluhan dan menganalisis persepsi pengguna terhadap aplikasi.



Gambar 4.12 Ulasan Pelanggan Positif

Tampilan *dashboard* ulasan pelanggan positif pada aplikasi KAI Access, yang menunjukkan hasil klasifikasi sentimen berbasis NLP dengan seluruh data pada gambar memiliki *rating* 5 dan berlabel positif. Visualisasi ini mempermudah peneliti dalam mengidentifikasi bentuk apresiasi atau kepuasan pengguna, yang ditunjukkan melalui penggunaan emoji serta kata-kata singkat bernuansa positif.



Gambar 4.13 Contoh Ulasan Pelanggan

Tampilan daftar contoh ulasan pelanggan aplikasi KAI Access yang telah dikelompokkan berdasarkan hasil klasifikasi sentimen menjadi tiga kategori, yaitu positive, negative, dan neutral. Kategori *positive* memuat ulasan yang bernada apresiatif, seperti “good”, “mantap lah”, dan “pelayanan cepat”, yang menunjukkan kepuasan pengguna terhadap layanan. Kategori *negative* berisi ulasan yang memuat keluhan, kritik, atau ekspresi ketidakpuasan, seperti “lemot” dan “apk tolol Gak bisa registrasi”, yang mencerminkan pengalaman pengguna yang buruk.

Sementara itu, kategori *neutral* mencakup ulasan yang bersifat informatif atau mengandung campuran opini positif dan negatif, seperti komentar terkait

pembaruan aplikasi atau pembagian kursi penumpang. Pengelompokan ini memudahkan analisis lebih lanjut terhadap persepsi pelanggan secara terstruktur, sekaligus memberikan gambaran umum mengenai proporsi dan karakteristik tiap jenis sentimen.

Gambar 4.14 Analisis Kata (Horizontal Chart)

Visualisasi *horizontal bar chart* hasil analisis frekuensi kata pada ulasan pelanggan aplikasi KAI Access yang telah dikelompokkan berdasarkan kategori sentimen **positive**, **negative**, dan **neutral**. Pada kategori *positive*, kata yang paling sering muncul adalah “sangat”, “tiket”, dan “membantu”, yang mencerminkan penilaian positif dan pengalaman memuaskan pengguna.

Pada kategori *negative*, kata “aplikasi” dan “tiket” mendominasi, diikuti oleh kata-kata seperti “saya”, “mau”, “gak/ga”, dan “malah” yang sering digunakan dalam konteks keluhan atau ketidakpuasan. Sementara itu, pada kategori *neutral*, kata yang paling banyak muncul adalah “tiket” dan “saya”, disusul “aplikasi” dan “mau”, yang umumnya digunakan dalam ulasan bersifat deskriptif atau campuran. Analisis ini membantu mengidentifikasi kata-kata kunci yang menjadi fokus utama pengguna pada setiap kategori sentimen, sehingga memudahkan interpretasi konteks dan tema pembahasan ulasan.

4.4 Eksraksi Fitur

Pada tahap ekstraksi fitur, proses dibagi menjadi dua bagian utama, yaitu splitting data dan pembobotan TF-IDF. *Splitting data* dilakukan untuk membagi dataset menjadi data latih dan data uji dengan tujuan agar model dapat dilatih pada sebagian data dan diuji pada sisanya guna mengevaluasi performa klasifikasi. Setelah pembagian data, dilakukan proses pembobotan menggunakan metode TF-

IDF (*Term Frequency–Inverse Document Frequency*), yang berfungsi untuk mengukur tingkat kepentingan setiap kata dalam dokumen terhadap keseluruhan korpus. Dengan metode ini, sistem dapat memberikan representasi numerik pada teks sehingga memungkinkan algoritma membaca, mengenali pola, dan mengklasifikasikan sentimen dengan lebih akurat.

4.4.1 *Spliting Data*

Splitting data merupakan tahap penting dalam proses pelatihan model yang bertujuan untuk memisahkan data menjadi dua bagian, yaitu data latih (*training data*) dan data uji (*testing data*). Data latih digunakan untuk membangun model, sementara data uji berfungsi untuk mengukur performa model terhadap data yang belum pernah dilihat sebelumnya. Pemisahan ini penting agar model dapat diuji secara objektif dan tidak hanya menyesuaikan diri dengan data yang sudah dikenal. Dalam proses ini digunakan beberapa skema pembagian data dengan proporsi berbeda untuk mendapatkan gambaran performa model yang lebih komprehensif. Proporsi yang umum digunakan dalam penelitian ini adalah 70:30, 80:20, dan 90:10 antara data latih dan data uji (Bichri et al., 2024).

Kode Program 4.13 Proses *Spliting Data*

```
from sklearn.model_selection import train_test_split

# Skema split data
split_schemes = [
    {'test_size': 0.3, 'name': '70:30'},
    {'test_size': 0.2, 'name': '80:20'},
    {'test_size': 0.1, 'name': '90:10'}
]

for scheme in split_schemes:
    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=scheme['test_size'], random_state=42, stratify=y
)
```

Kode Program 4.13 menunjukkan proses pemisahan dataset menjadi data latih dan data uji menggunakan fungsi *train_test_split* dari pustaka *sklearn*. Skema pembagian data ditentukan melalui parameter *test_size*, dengan tiga konfigurasi berbeda yaitu 30%, 20%, dan 10% untuk data uji. Parameter *random_state*

digunakan agar proses pembagian data menghasilkan hasil yang konsisten setiap kali dijalankan. Sementara itu, argumen *stratify*=y memastikan bahwa distribusi kelas sentimen tetap seimbang antara data latih dan data uji. Pembagian data ini diulang sebanyak tiga kali, masing-masing menggunakan skema pembagian yang telah ditentukan. Dengan pendekatan ini, performa model dapat dibandingkan pada berbagai kondisi pembagian data yang berbeda.

4.4.2 *Pembobotan* TF-IDF

Setelah data terbagi menjadi data latih dan data uji, langkah selanjutnya adalah mengubah data teks menjadi representasi numerik melalui metode TF-IDF (*Term Frequency–Inverse Document Frequency*). Metode tersebut menghitung bobot setiap kata berdasarkan seberapa sering kata muncul dalam satu ulasan dibandingkan seluruh ulasan dalam korpus. Kata-kata yang muncul secara konsisten dalam satu ulasan namun jarang ditemukan di ulasan lain akan mendapatkan bobot yang lebih tinggi. Proses ini membantu algoritma untuk lebih fokus pada kata-kata yang benar-benar mencerminkan karakteristik dari masing-masing ulasan. Hasil pembobotan digunakan sebagai input utama dalam pelatihan model klasifikasi sentimen.

Kode Program 4.14 Proses Pembobotan TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Ekstraksi Fitur dengan TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Tiga ulasan digunakan sebagai contoh dalam simulasi perhitungan manual pembobotan TF-IDF pada penelitian ini, yaitu:

(Ulasan 1) = top up terlalu lama untuk masuk

(Ulasan 2) = terlalu sering update

(Ulasan 3) = aplikasi bagus, dan selalu update

Setelah melalui tahap *preprocessing*, masing-masing ulasan menghasilkan teks yang telah dibersihkan dan disederhanakan, yaitu sebagai berikut:

(Ulasan 1) = top up terlalu lama masuk

(Ulasan 2) = terlalu sering update

(Ulasan 3) = aplikasi bagus selalu update

Dari ketiga ulasan tersebut, diperoleh daftar lengkap kata-kata unik yang muncul. Kata-kata inilah yang selanjutnya digunakan dalam proses perhitungan bobot TF-IDF secara manual.

[‘aplikasi’, ‘bagus’, ‘lama’, ‘masuk’, ‘selalu’, ‘sering’, ‘terlalu, ‘top’, ‘up’, ‘update’]

Langkah berikut dalam pengolahan data adalah menghitung bobot kata menggunakan metode TF-IDF. Teknik ini berfungsi untuk mengonversi teks menjadi vektor angka berdasarkan tingkat kepentingan setiap kata dalam ulasan. TF-IDF terdiri dari dua komponen utama, yaitu *Term Frequency* (TF) yang mengukur seberapa sering sebuah kata muncul dalam ulasan tertentu, dan *Inverse Document Frequency* (IDF) yang menekankan bobot lebih besar pada kata yang jarang ditemukan di keseluruhan ulasan. Perhitungan dimulai dengan menentukan nilai TF untuk setiap kata dalam ulasan. Contoh kata-kata yang digunakan dalam perhitungan TF secara manual disajikan pada Tabel 4.1.

Tabel 4.1 Perhitungan TF

Term	**TF**					
	U1	**TF (U1)**	**U2**	**TF (U2)**	**U3**	**TF (U3)**
aplikasi	0	0	0	0	1	1/4 = 0.25
bagus	0	0	0	0	1	1/4 = 0.25
lama	1	1/4 = 0.25	0	0	0	0
masuk	1	1/4 = 0.25	0	0	0	0
selalu	0	0	0	0	1	1/4 = 0.25
sering	0	0	1	1/3 = 0.33	0	0
terlalu	1	1/4 = 0.25	1	1/3 = 0.33	0	0

top	1	1/4 = 0.25	0	0	0	0
up	1	1/4 = 0.25	0	0	0	0
update	0	0	1	1/3 = 0.33	1	1/4 = 0.25

Setelah mendapatkan nilai *Term Frequency* (TF) dari setiap kata dalam ulasan, tahap selanjutnya adalah menghitung *Inverse Document Frequency* (IDF). IDF digunakan untuk menilai tingkat keunikan suatu kata dibandingkan seluruh ulasan yang dianalisis. Pada penelitian ini, jumlah ulasan yang digunakan berjumlah tiga, karena terdapat tiga ulasan yang dijadikan sampel. Rincian hasil perhitungan IDF untuk masing-masing kata ditampilkan pada Tabel 4.2.

Tabel 4.2 Perhitungan IDF

Term	$\boldsymbol{n(t)}$	**IDF**
aplikasi	1	log 3/1 = 0.477
bagus	1	log 3/1 = 0.477
lama	1	log 3/1 = 0.477
masuk	1	log 3/1 = 0.477
selalu	1	log 3/1 = 0.477
sering	1	log 3/1 = 0.477
terlalu	2	log 3/2 = 0.176
top	1	log 3/1 = 0.477
up	1	log 3/1 = 0.477
update	2	log 3/2 = 0.176

Setelah memperoleh nilai TF dan IDF pada tahap sebelumnya, langkah selanjutnya adalah menghitung bobot TF-IDF untuk setiap kata. Perhitungan dilakukan dengan cara mengalikan nilai TF dan IDF dari masing-masing kata dalam ulasan. Hasil dari proses tersebut kemudian digunakan sebagai representasi numerik bobot kata. Detail hasil perhitungan TF-IDF ditampilkan dalam Tabel 4.3.

Tabel 4.3 Perhitungan TF-TDF

Term	TF-IDF U1	TF-IDF U2	TF-IDF U3
aplikasi	0	0	0.25 x 0.477 = 0.119
bagus	0	0	0.25 x 0.477 = 0.119
lama	0.25 x 0.477 = 0.119	0	0
masuk	0.25 x 0.477 = 0.119	0	0
selalu	0	0	0.25 x 0.477 = 0.119
sering	0	0.33 x 0.477 = 0.159	0
terlalu	0.25 x 0.176 = 0.044	0.33 x 0.176 = 0.059	0
top	0.25 x 0.477 = 0.119	0	0
up	0.25 x 0.477 = 0.119	0	0
update	0	0.33 x 0.176 = 0.059	0.25 x 0.477 = 0.119

Bobot TF-IDF yang telah dihitung kemudian dapat direpresentasikan dalam bentuk *array*, di mana setiap baris menggambarkan satu ulasan dan setiap kolom merepresentasikan kata yang terdapat dalam kumpulan ulasan. Representasi tersebut ditujukan untuk mempermudah interpretasi nilai bobot setiap kata dalam masing-masing ulasan, sebagaimana ditampilkan pada format berikut:

array([
  [0.000, 0.000, 0.119, 0.119, 0.000, 0.000, 0.044, 0.119, 0.119, 0.000], #U1
  [0.000, 0.000, 0.000, 0.000, 0.000, 0.159, 0.059, 0.000, 0.000, 0.059], #U2
  [0.119, 0.119, 0.000, 0.000, 0.119, 0.000, 0.000, 0.000, 0.000, 0.119] #U3
])

Setiap baris pada *array* TF-IDF yang ditampilkan merepresentasikan vektor fitur dari satu ulasan berdasarkan nilai bobot kata yang telah dihitung. *Term Frequency* (TF) mengukur intensitas kemunculan suatu kata dalam dokumen tertentu, sementara *Inverse Document Frequency* (IDF) memberikan bobot berdasarkan seberapa jarang kata tersebut ditemukan di seluruh ulasan. Kata-kata yang sering muncul di semua ulasan akan memiliki nilai IDF rendah karena dianggap kurang relevan dalam membedakan karakteristik antar ulasan.

Sebaliknya, jika suatu kata hanya muncul pada satu atau dua dokumen saja, maka nilai IDF-nya menjadi lebih tinggi karena kata tersebut dianggap lebih spesifik dan penting untuk dokumen tersebut. Melalui *array* yang terbentuk, dapat terlihat bahwa masing-masing dokumen memiliki pola bobot kata yang berbeda, yang mencerminkan kekhasan isi dari tiap ulasan.

4.5 Klasifikasi *Naive Bayes*

Setelah proses ekstraksi fitur selesai, tahap selanjutnya adalah melakukan klasifikasi menggunakan algoritma *Naive Bayes*. *Naive Bayes* bekerja berdasarkan prinsip probabilistik dan mengasumsikan bahwa fitur-fitur bersifat saling independen. Pendekatan ini dinilai efisien dan cukup akurat dalam mengklasifikasikan ulasan pengguna ke dalam kelas sentimen positif, negatif, atau netral. Model dilatih menggunakan data hasil pembobotan TF-IDF yang telah dipisahkan dalam bentuk data latih dan data uji.

Kode Program 4.15 Pelatihan Model *Naive Bayes*

```
# Pelatihan Model Naive Bayes
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# Prediksi
y_pred = model.predict(X_test_tfidf)
```

Kode Program 4.15 menunjukkan proses pelatihan dan prediksi menggunakan algoritma *Multinomial Naive Bayes*. Pertama, objek model dibuat menggunakan kelas *MultinomialNB*() dari pustaka *scikit-learn*, yang dirancang untuk menangani data representasi frekuensi seperti TF-IDF. Model kemudian dilatih dengan data *X_train_tfidf* sebagai fitur dan *y_train* sebagai label kelas menggunakan fungsi .fit*()*. Setelah pelatihan selesai, model digunakan untuk memprediksi kelas dari data uji *X_test_tfidf* melalui metode *.predict*(). Hasil prediksi disimpan dalam variabel y_pred yang nantinya akan dibandingkan dengan data aktual untuk mengevaluasi kinerja model. Kode Program 15 menjadi inti dari proses klasifikasi dalam penelitian ini karena berperan langsung dalam membangun dan menguji model sentimen. Pendekatan ini memungkinkan klasifikasi dilakukan secara otomatis terhadap data ulasan yang telah diproses sebelumnya.

4.6 Evaluasi Model

Setelah proses klasifikasi dilakukan, langkah berikutnya adalah mengevaluasi performa model untuk mengetahui seberapa baik model dalam mengenali dan mengklasifikasikan data. Evaluasi ini sangat penting untuk mengukur tingkat akurasi dan keandalan model dalam menganalisis sentimen pengguna. Salah satu metode yang digunakan adalah *Confusion Matrix*, yang menyajikan perbandingan antara hasil prediksi model dan label aktual. Selain itu, digunakan pula metrik evaluasi seperti *precision*, *recall*, *f1-score*, dan *accuracy* untuk mendapatkan gambaran menyeluruh mengenai performa model. Melalui evaluasi ini, peneliti dapat menilai kekuatan dan kelemahan model *Naive Bayes* yang telah dibangun.

Kode Program 4.16 Evaluasi Model Dengan *Confusion Matrix*

```
# Evaluasi
print(f"Classification Report (Skema {scheme['name']}):")
print(classification_report(y_test, y_pred,
target_names=['Negatif','Netral','Positif']))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred) plot_confusion_matrix(cm,
['Negatif','Netral','Positif'], f"Confusion Matrix (Skema
{scheme['name']})")
```


Kode Program 4.16 digunakan untuk mengevaluasi kinerja model klasifikasi berdasarkan hasil prediksi yang telah dihasilkan sebelumnya. Pertama, fungsi *classification_report*() menampilkan metrik evaluasi seperti *precision*, *recall*, dan *f1-score* untuk masing-masing kelas sentimen: negatif, netral, dan positif. Laporan ini membantu dalam melihat sejauh mana model mampu mengidentifikasi setiap kategori dengan tepat. Selanjutnya, fungsi *confusion_matrix*() menghasilkan data tabular yang membandingkan prediksi model terhadap nilai aktual. Hasil *Confusion Matrix* tersebut kemudian divisualisasikan menggunakan fungsi *plot_confusion_matrix*() agar lebih mudah dianalisis secara visual. Parameter *scheme['name']* digunakan untuk menunjukkan skema pembagian data (seperti 70:30, 80:20, atau 90:10) yang sedang diuji. Kode Program 4.16 secara keseluruhan memungkinkan analisis performa model dilakukan secara komprehensif berdasarkan berbagai metrik evaluasi.

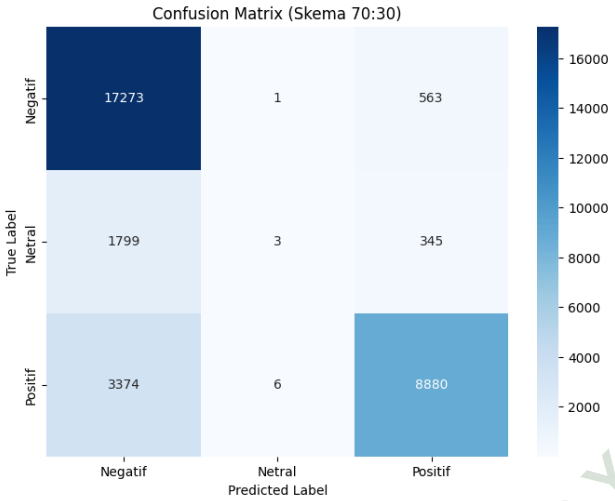
4.6.1 Skema *Split* Data 70:30

Pengujian pertama dilakukan dengan menggunakan skema pembagian data sebesar 70:30, yaitu 70% untuk data latih dan 30% untuk data uji. Dari total 107.479 data, sebanyak 75.235 data digunakan sebagai data latih dan 32.244 data sisanya digunakan sebagai data uji. Skema ini bertujuan untuk memberikan cukup data kepada model agar dapat belajar secara optimal, namun tetap menyisakan data yang cukup untuk proses evaluasi. Hasil dari evaluasi model pada skema ini disajikan pada Gambar 4.15 dan Gambar 4.16.

```
============================================================
ANALISIS DENGAN SKEMA SPLIT DATA: 70:30
============================================================
Classification Report (Skema 70:30):
              precision    recall  f1-score   support

     Negatif       0.77      0.97      0.86     17837
      Netral       0.30      0.00      0.00      2147
     Positif       0.91      0.72      0.81     12260

    accuracy                           0.81     32244
   macro avg       0.66      0.56      0.56     32244
weighted avg       0.79      0.81      0.78     32244
```

Gambar 4.15 *Classification Report* Skema *Split* Data 70:30

Gambar 4.15 menampilkan hasil evaluasi performa model berdasarkan metrik klasifikasi yang mencakup *precision*, *recall*, dan *f1-score*. Model memiliki performa sangat baik dalam mengklasifikasikan ulasan berlabel positif dan negatif, dengan nilai *f1-score* masing-masing sebesar 0.81 dan 0.86. Namun, pada kelas netral, model mengalami penurunan performa yang cukup signifikan dengan nilai *f1-score* hanya 0.00 karena *recall*-nya nol. Secara keseluruhan, akurasi model mencapai 0.81 yang menunjukkan bahwa mayoritas prediksi sesuai dengan label sebenarnya. Nilai rata-rata tertimbang (*weighted avg*) untuk precision dan recall masing-masing sebesar 0.79 dan 0.81, mencerminkan kinerja yang konsisten di kelas mayoritas.



Gambar 4.16 *Confusion Matrix* Skema *Split* Data 70:30

Gambar 4.16 memperlihatkan *Confusion Matrix* dari hasil klasifikasi dengan skema 70:30. Model mampu mengklasifikasikan 17.273 ulasan negatif dengan benar, meskipun masih terdapat 563 ulasan negatif yang diklasifikasikan sebagai positif. Untuk kelas netral, terlihat bahwa model kesulitan mengenali kategori ini, terbukti dari hanya 3 ulasan yang berhasil diklasifikasikan dengan tepat, sementara 1.799 lainnya salah diklasifikasikan sebagai negatif. Pada kelas positif, sebanyak 8.880 ulasan berhasil diklasifikasikan dengan benar, tetapi terdapat 3.374 ulasan positif yang justru dikenali sebagai negatif. Kondisi ini menunjukkan bahwa model cenderung lebih percaya diri terhadap kelas mayoritas, terutama negatif dan positif. Hal ini wajar karena distribusi data netral memang jauh lebih sedikit dibanding dua kelas lainnya. Secara umum, *Confusion Matrix* ini memperkuat temuan dari *classification report* bahwa model masih kesulitan dalam mengidentifikasi ulasan dengan sentimen netral.

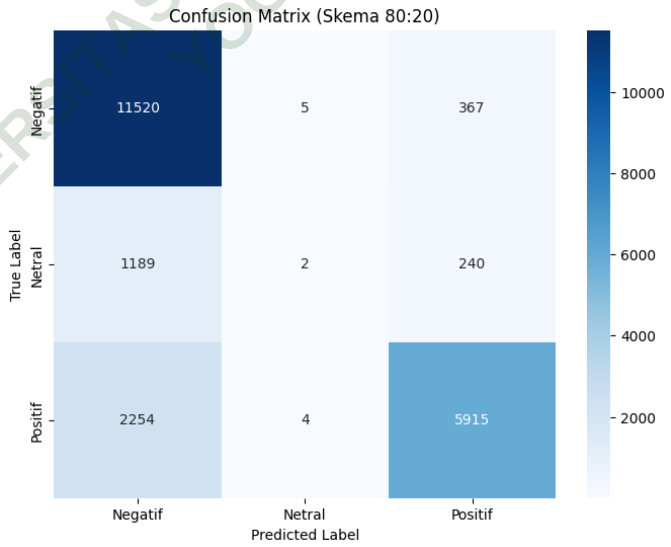
4.6.2 Skema *Split* Data 80:20

Pada skema ini, data dibagi menjadi 80% untuk pelatihan dan 20% untuk pengujian model. Dari total 107.479 data, sebanyak 85.983 data digunakan untuk melatih model, sementara 21.496 sisanya digunakan untuk menguji kinerja model. Pembagian ini memberikan model lebih banyak data untuk belajar sehingga diharapkan dapat meningkatkan ketepatan prediksi. Hasil evaluasi berdasarkan pembagian tersebut dapat dilihat pada Gambar 21 dan Gambar 22.

```
============================================================
ANALISIS DENGAN SKEMA SPLIT DATA: 80:20
============================================================
Classification Report (Skema 80:20):
              precision    recall  f1-score   support

     Negatif       0.77      0.97      0.86     11892
      Netral       0.18      0.00      0.00      1431
     Positif       0.91      0.72      0.81      8173

    accuracy                           0.81     21496
   macro avg       0.62      0.56      0.56     21496
weighted avg       0.78      0.81      0.78     21496
```

Gambar 4. 17 *Classification Report* Skema *Split* Data 80:20

Gambar 4.17 memperlihatkan hasil pengujian model pada skema 80:20 dengan menampilkan metrik evaluasi seperti *precision*, *recall*, dan *f1-score*. Model menunjukkan kinerja yang sangat baik pada kelas negatif dan positif, masing-masing dengan nilai *f1-score* sebesar 0.86 dan 0.81. Namun, untuk kelas netral, performa model sangat rendah dengan *recall* 0.00 dan *f1-score* 0.00, menandakan bahwa hampir semua ulasan netral tidak berhasil dikenali. Tingkat akurasi keseluruhan mencapai 0.81, yang berarti model mampu memprediksi dengan benar lebih dari 80% data uji. Rata-rata tertimbang menunjukkan *precision* sebesar 0.78 dan *recall* sebesar 0.81, mengindikasikan bahwa model cukup stabil dalam menangani kelas mayoritas.



Gambar 4.18 *Confusion Matrix* Skema *Split* Data 80:20

Gambar 4.18 menunjukkan *Confusion Matrix* yang menggambarkan bagaimana prediksi model tersebar pada tiap kelas. Dari total 11.892 ulasan negatif,

sebanyak 11.520 berhasil diklasifikasikan dengan benar, dan hanya sebagian kecil yang keliru sebagai positif. Untuk kelas netral, hanya 2 dari 1.431 ulasan yang dikenali dengan tepat, sedangkan sisanya sebagian besar salah diklasifikasikan sebagai negatif. Hal ini menegaskan lemahnya model dalam mengenali ulasan netral. Pada kelas positif, terdapat 5.915 prediksi yang benar dari total 8.173, dan sisanya sebagian besar keliru terklasifikasi sebagai negatif. Model tampak lebih condong mengklasifikasikan ke kelas negatif ketika ragu, seperti terlihat dari banyaknya prediksi yang masuk ke kategori tersebut. Ketidakseimbangan data antar kelas turut memengaruhi hasil ini, karena model belajar lebih banyak dari data negatif dan positif. Meski demikian, visualisasi ini menunjukkan bahwa model sudah cukup mampu membedakan sentimen utama, meskipun masih perlu perbaikan dalam menangani kategori netral.

4.6.3 Skema *Split* Data 90:10

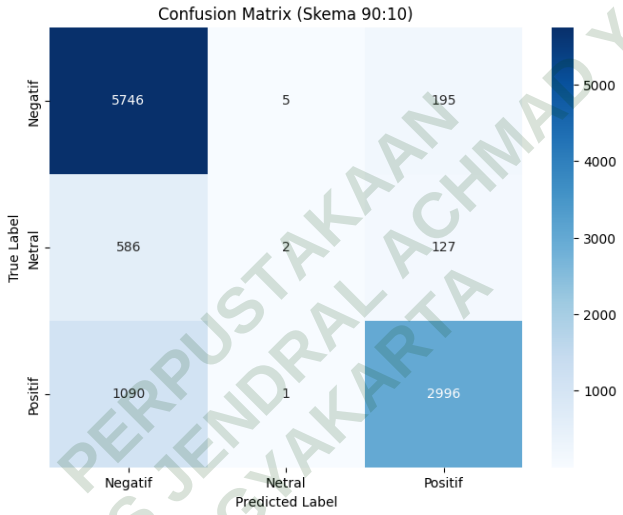
Pengujian berikutnya dilakukan dengan menggunakan skema pembagian data sebesar 90:10 yaitu 90% data digunakan sebagai data latih dan 10% sisanya sebagai data uji. Dari total 107479 data ulasan sebanyak 96731 data dialokasikan untuk pelatihan model sedangkan 10748 data lainnya digunakan untuk proses pengujian. Skema ini digunakan untuk memberikan lebih banyak data pelatihan agar model mampu mempelajari pola sentimen dengan lebih optimal.

```
============================================================
ANALISIS DENGAN SKEMA SPLIT DATA: 90:10
============================================================
Classification Report (Skema 90:10):
              precision    recall  f1-score   support

     Negatif       0.77      0.97      0.86      5946
      Netral       0.25      0.00      0.01       715
     Positif       0.90      0.73      0.81      4087

    accuracy                           0.81     10748
   macro avg       0.64      0.57      0.56     10748
weighted avg       0.79      0.81      0.78     10748
```

Gambar 4.19 *Classification Report* Skema *Split* Data 90:10

Gambar 4.19 menunjukkan hasil evaluasi kinerja model klasifikasi *Naive Bayes* dengan metrik *precision recall* dan *f1-score* pada masing-masing kelas sentimen. Sentimen negatif memperoleh nilai *precision* sebesar 0.77 *recall* sebesar

0.97 dan *f1-score* sebesar 0.86 yang menandakan bahwa model sangat baik dalam mendeteksi ulasan negatif. Sentimen positif memiliki nilai precision sebesar 0.90 *recall* sebesar 0.73 dan *f1-score* sebesar 0.81 yang menunjukkan performa yang juga cukup baik. Namun pada sentimen netral nilai *precision* hanya sebesar 0.25 dengan *recall* 0.00 dan *f1-score* sangat rendah yaitu 0.01 yang menunjukkan bahwa model kesulitan dalam mendeteksi sentimen ini. Secara keseluruhan akurasi model mencapai 0.81 dengan nilai rata-rata tertimbang *f1-score* sebesar 0.78 yang menunjukkan bahwa model cukup baik dalam klasifikasi secara umum.



Gambar 4.20 *Confusion Matrix* Skema *Split* Data 90:10

Gambar 4.20 menampilkan *Confusion Matrix* yang menggambarkan distribusi hasil prediksi model terhadap label aktual dari data uji. Pada sentimen negatif sebanyak 5746 data berhasil diklasifikasikan dengan benar sedangkan 5 data salah diklasifikasikan sebagai netral dan 195 data salah sebagai positif. Untuk sentimen netral hanya 2 data yang diklasifikasikan dengan benar sementara 586 data diklasifikasikan sebagai negatif dan 127 data sebagai positif. Kelas positif menunjukkan hasil yang cukup baik dengan 2996 data terklasifikasi dengan benar namun masih terdapat 1090 data yang diklasifikasikan sebagai negatif dan 1 data sebagai netral. Ketidakseimbangan jumlah data pada masing-masing kelas sangat memengaruhi performa model khususnya terhadap sentimen netral. Model cenderung lebih akurat dalam mengenali kelas yang memiliki jumlah data lebih

besar seperti negatif dan positif. Hal ini menunjukkan bahwa model lebih terlatih pada data yang jumlahnya dominan dalam data latih.

4.7 Interpretasi Hasil

Setelah dilakukan proses pelatihan dan pengujian model pada berbagai skema pembagian data selanjutnya dilakukan interpretasi terhadap hasil evaluasi model. Evaluasi ini dilakukan untuk mengetahui seberapa baik kinerja model dalam mengklasifikasikan ulasan pengguna ke dalam kategori sentimen negatif netral dan positif. Proses evaluasi melibatkan beberapa metrik penting yaitu *accuracy* , *precision, recall,* dan *f1-score*. Keempat metrik ini digunakan untuk menggambarkan performa model secara lebih menyeluruh baik dari segi ketepatan ketelitian maupun keseimbangan klasifikasi. Berikut adalah hasil evaluasi model *Naive Bayes* terhadap tiga skema split data yang digunakan dalam penelitian ini

Tabel 4.4 Hasil Evaluasi Model

Skema	*Accuracy*	*Precision*	*Recall*	*F1-Score*
70:30	0.81	0.66	0.56	0.56
80:20	0.81	0.62	0.56	0.56
90:10	0.81	0.64	0.57	0.56

Berdasarkan Tabel 4.4 diketahui bahwa seluruh skema menghasilkan nilai akurasi yang sama yaitu sebesar 0.81. Meskipun akurasinya konsisten terdapat perbedaan kecil pada metrik *precision*, *recall*, dan *f1-score* di antara ketiga skema. Skema 70:30 memiliki nilai *precision* tertinggi yaitu 0.66 diikuti oleh 90:10 sebesar 0.64 dan 80:20 sebesar 0.62. Ini menunjukkan bahwa skema dengan proporsi data latih 70% memiliki ketepatan klasifikasi yang sedikit lebih baik dibandingkan skema lainnya.

Pada metrik *recall* semua skema menunjukkan hasil yang serupa dengan nilai antara 0.56 hingga 0.57. Skema 90:10 memiliki *recall* tertinggi meskipun hanya berbeda tipis dari dua skema lainnya. Hal ini mengindikasikan bahwa model mampu mengenali data yang relevan secara cukup stabil pada semua skema pembagian data. Namun demikian nilai *recall* yang tergolong sedang menunjukkan

bahwa masih ada beberapa ulasan yang belum berhasil dikenali secara tepat oleh model.

F1-score pada ketiga skema bernilai sama yaitu 0.56 yang berarti keseimbangan antara *precision* dan *recall* masih belum optimal. Hal ini menunjukkan bahwa meskipun model dapat mengklasifikasikan sebagian besar ulasan dengan cukup baik namun performanya masih dapat ditingkatkan khususnya dalam menangani sentimen netral yang menjadi tantangan tersendiri. Secara keseluruhan hasil evaluasi ini memberikan gambaran bahwa model *Naive Bayes* bekerja cukup stabil namun masih memerlukan perbaikan dalam meningkatkan presisi dan sensitivitas klasifikasinya.

4.8 Visualisasi

Visualisasi data digunakan untuk melihat representasi kata-kata yang paling sering muncul dalam ulasan pengguna berdasarkan kategori sentimen. Teknik yang digunakan adalah *word cloud* yang dapat menggambarkan seberapa dominan suatu kata berdasarkan ukuran tampilannya. Semakin besar ukuran kata maka semakin sering kata tersebut muncul dalam *dataset*. *Word cloud* ini dibagi menjadi tiga bagian yaitu sentimen positif, sentimen netral, dan sentimen negatif. Visualisasi ini membantu dalam memahami fokus utama dari masing-masing sentimen yang disampaikan oleh pengguna terhadap aplikasi.

Kode Program 4.17 Proses *Generate Word Cloud*

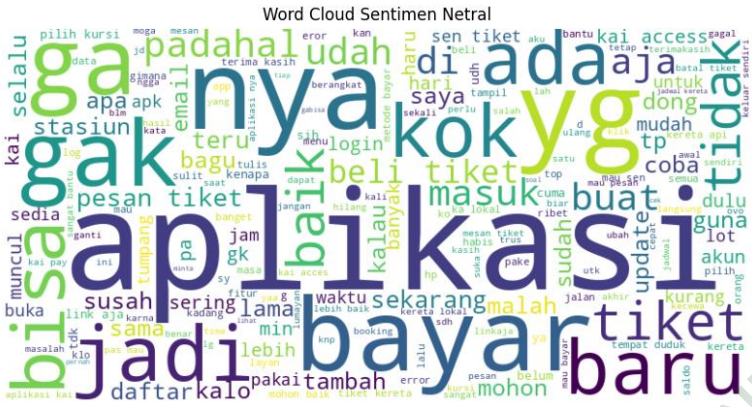
```
positif_text = df_model[df_model['sentimen'] ==
2]['processed_text']
netral_text = df_model[df_model['sentimen'] ==
1]['processed_text']
negatif_text = df_model[df_model['sentimen'] ==
0]['processed_text']

generate_wordcloud(positif_text, 'Word Cloud Sentimen Positif')
generate_wordcloud(negatif_text, 'Word Cloud Sentimen Negatif')
generate_wordcloud(netral_text, 'Word Cloud Sentimen Netral')
```

Kode Program 4.17 digunakan untuk mengelompokkan data ulasan berdasarkan label sentimen yang telah ditentukan sebelumnya. Data dibagi ke dalam tiga variabel yaitu positif, netral dan negatif yang masing-masing diambil dari kolom *processed_text*. Setelah data dikelompokkan fungsi

generate_wordcloud dijalankan untuk menghasilkan visualisasi *word cloud* dari masing-masing kategori. Fungsi ini menampilkan kumpulan kata dominan pada setiap kelompok sentimen dengan ukuran huruf yang berbeda sesuai frekuensi kemunculannya.

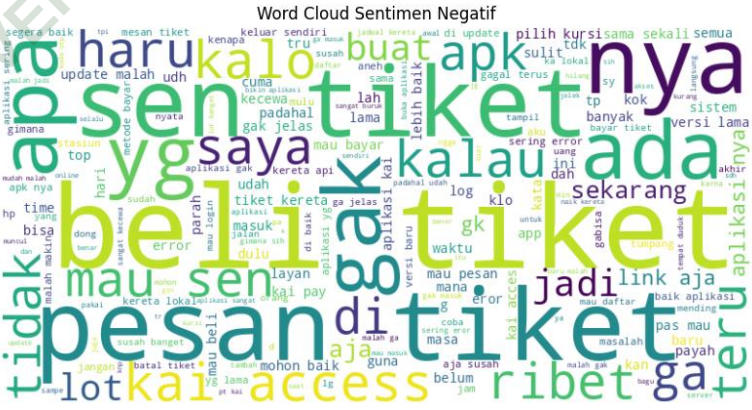
Gambar 4.21 *Word Cloud* Sentimen Positif

Gambar 4.21 menampilkan *word cloud* dari ulasan dengan sentimen positif yang berasal dari pengguna aplikasi. Kata ‘bantu’, ‘sangat’, dan ‘aplikasi’ muncul sebagai kata yang paling dominan yang menunjukkan bahwa pengguna merasa terbantu oleh keberadaan aplikasi. Selain itu, kata seperti ‘beli’, ‘tiket’, ‘pesan tiket’, dan ‘kereta’ juga tampak cukup besar yang menandakan bahwa fitur pembelian tiket menjadi aspek yang paling dihargai. Kemunculan kata ‘mantap’, ‘mudah’, ‘baik’, dan ‘keren’ memperkuat bahwa pengalaman pengguna terhadap layanan dianggap memuaskan. Kata-kata positif lainnya seperti ‘top’, ‘nyaman’, dan ‘terima kasih’ juga menunjukkan bentuk apresiasi terhadap fitur yang tersedia. Gambar 4.21 menunjukkan bahwa ulasan positif lebih banyak berkaitan dengan kemudahan akses dan keberhasilan dalam melakukan pemesanan.



Gambar 4.22 *Word Cloud* Sentimen Netral

Gambar 4.22 memperlihatkan *word cloud* dari ulasan yang tergolong dalam kategori sentimen netral. Kata ‘aplikasi’ masih menjadi yang paling menonjol diikuti oleh kata ‘bayar’, ‘jadi’, dan ‘masuk’. Hal ini menunjukkan bahwa meskipun tidak menunjukkan kepuasan atau ketidakpuasan yang jelas pengguna tetap memberikan catatan terkait fungsionalitas aplikasi. Kata seperti ‘kok’, ‘gak’, dan ‘padahal’ muncul dengan ukuran sedang yang menandakan adanya nada keheranan atau kebingungan namun tidak disertai keluhan keras. Istilah lain seperti ‘daftar’, ‘update’, dan ‘tampil’ juga menunjukkan bahwa pengguna memperhatikan aspek teknis dari aplikasi. Gambar 4.22 mencerminkan bahwa sentimen netral seringkali muncul dari situasi yang tidak terlalu memuaskan tetapi juga tidak terlalu mengganggu.



Gambar 4.23 *Word Cloud* Sentimen Negatif

Gambar 4.23 menampilkan kumpulan kata yang sering muncul dalam ulasan negatif dari pengguna aplikasi. Kata ‘pesan tiket’, ‘beli tiket’, dan ‘tidak’ menjadi

yang paling dominan yang menunjukkan bahwa masalah utama pengguna berkaitan dengan proses pemesanan tiket. Kata-kata seperti ‘ribet’, ‘tidak bisa’, ‘error’ dan ‘gagal’ menggambarkan adanya gangguan dalam penggunaan aplikasi. Selain itu munculnya kata ‘harus’ dan ‘kalo’ menandakan adanya tuntutan atau kondisi tertentu yang membuat pengguna kesulitan. Kata ‘kecewa’, ‘lama’, dan ‘susah’ memperkuat bahwa pengalaman pengguna pada kategori ini bersifat negatif dan penuh hambatan. Gambar 4.23 menunjukkan bahwa ulasan negatif cenderung berfokus pada kendala teknis dan proses yang dianggap merepotkan.