BAB 4
HASIL PENELITIAN

4.1 Ringkasan Hasil Penelitian

Penelitian ini bertujuan untuk mengklasifikasikan komentar yang mampu mendeteksi komentar berpotensi *cyberbullying* pada aplikasi TikTok dengan memanfaatkan algoritma *K-Nearest Neighbor* (KNN). Proses penelitian dimulai dengan pengumpulan data melalui teknik web *scraping* menggunakan Apify.com, yang menghasilkan sebanyak 5.516 komentar dari dua video dengan tingkat interaksi tinggi. Setelah dilakukan seleksi dan pembersihan data, diperoleh 3.724 komentar yang siap diproses.

Komentar tersebut kemudian diproses melalui tahapan *pre-processing* teks, guna memastikan data dalam format yang seragam dan siap dianalisis. Setiap komentar dilabeli secara manual ke dalam dua kelas yaitu, label 0 untuk komentar non-*cyberbullying* dan label 1 untuk komentar *cyberbullying*. Data kemudian diubah menjadi bentuk numerik menggunakan metode TF-IDF. Selanjutnya, data dibagi menggunakan teknik *Stratified K-Fold Cross Validation*. Berdasarkan seluruh tahapan yang telah dilakukan, penelitian ini menunjukkan bahwa algoritma *K-Nearest Neighbor* (KNN) mampu menjalankan proses klasifikasi terhadap komentar TikTok dengan baik.

4.2 Pembahasan

Berikut adalah pembahasan secara rinci hasil yang diperoleh dari tahapan penelitian yang telah dilakukan.

4.2.1 Pengumpulan Data

Pengumpulan data dalam penelitian ini bertujuan untuk memperoleh kumpulan komentar dari platform TikTok, yang kemudian digunakan dalam proses deteksi cyberbullying dengan menerapkan algoritma K-Nearest Neighbor (KNN). Teknik utama yang digunakan dalam pengumpulan data ini adalah web scraping untuk mengekstrak komentar-komentar dari TikTok. Proses scraping dilakukan menggunakan Apify.com, yaitu sebuah layanan web scraping berbasis API yang

mendukung ekstraksi data dari berbagai situs, termasuk media sosial seperti TikTok. Pemilihan Apify sebagai *tools* pengumpulan data didasarkan pada kemampuannya untuk mengakses dan mengekstrak data secara efisien serta fleksibel, khususnya dari platform TikTok.

Data yang digunakan dalam penelitian ini diperoleh dari komentar-komentar pengguna TikTok yang dikumpulkan melalui *scraping* mandiri. Komentar tersebut diambil dari dua video milik *content creator* populer Ria Ricis (@riaricis). Proses pengambilan data dilakukan pada tanggal 14 Mei 2025. Total durasi dari dua video yang menjadi sumber data adalah sekitar 13 menit 31 detik, yang dinilai cukup untuk mendapatkan komentar dalam jumlah dan variasi yang memadai.

Melalui proses *scraping* tersebut, berhasil dikumpulkan sebanyak 5.516 komentar. Data yang diperoleh dari API Apify terdiri dari berbagai atribut yang memberikan informasi detail mengenai masing-masing komentar, dan seluruh data disimpan dalam format CSV agar mudah untuk diolah. Berikut ditampilkan cuplikan hasil dari fungsi *df.info()*. yang memberikan gambaran umum mengenai struktur dan informasi dataset.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5516 entries, 0 to 5515
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   avatarThumbnail    5516 non-null   object
 1   cid                5516 non-null   int64
 2   createTime         5516 non-null   int64
 3   createTimeISO      5516 non-null   object
 4   diggCount          5516 non-null   int64
 5   input              5516 non-null   object
 6   likedByAuthor      5516 non-null   bool
 7   pinnedByAuthor     5516 non-null   bool
 8   repliesToId        0 non-null      float64
 9   replyCommentTotal  5516 non-null   int64
 10  submittedVideoUrl  5516 non-null   object
 11  text               5516 non-null   object
 12  uid                5516 non-null   int64
 13  uniqueId           5516 non-null   object
 14  videoWebUrl        5516 non-null   object
dtypes: bool(2), float64(1), int64(5), object(7)
memory usage: 571.1+ KB
```

Gambar 4. 1 Hasil Pengumpulan Data

Gambar 4.1 menunjukkan hasil dari proses pengumpulan data yang dilakukan melalui web *scraping* pada platform TikTok. Dataset ini terdiri dari 5.516 baris data dengan 15 kolom informasi yang berbeda, anatara lain:

a. avatarThumbnail: link gambar profil pengguna.
b. cid: ID unik dari komentar.
c. createTime: waktu pembuatan komentar (*timestamp*).
d. createTimeISO: waktu pembuatan komentar dalam format ISO.
e. diggCount: jumlah likes yang diterima komentar.
f. likedByAuthor: status apakah komentar disukai oleh pemilik video.
g. pinnedByAuthor: status apakah komentar di-*pin* oleh pemilik video.
h. repliesToId: ID komentar yang dibalas (jika komentar merupakan balasan).
i. replyCommentTotal: jumlah balasan yang diterima komentar.
j. submittedVideoUrl: URL video tempat komentar dibuat.
k. text: isi komentar.
l. uid: ID pengguna yang mengomentari.
m. uniqueId: username pengguna.
n. videoWebUrl: tautan video TikTok yang menjadi sumber komentar.

Dari seluruh atribut yang tersedia dalam dataset hasil *scraping*, penelitian ini hanya menggunakan atribut *text* atau *comment* yang berisi isi komentar yang dapat dilihat pada gambar 4.2. Hal ini karena penelitian difokuskan pada isi komentar secara tekstual, yang menjadi dasar dalam menentukan apakah sebuah komentar mengandung unsur *cyberbullying* atau tidak.

index	comment
0	good job
1	up 😘
2	Wow
3	taht's great!
4	top
5	Wow
6	kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa😍 😘 😘
7	GUE KIRA ATAP TU ATAP RUMAH BJIR🤦 🤦
8	sungkem ahh sama mbak tersopanm🙏
9	udah lama nggk liat ATAP 😍
10	Klo sama atap pasti satu frekuensi dari dulu😍

Gambar 4. 2 Hasil Data *Review*

4.2.2 Pembersihan Data Awal

Pada tahapan ini langkah pertama dalam proses ini adalah memeriksa apakah terdapat nilai kosong pada kolom komentar. Pemeriksaan dilakukan menggunakan fungsi *df.isna().sum()* untuk menghitung jumlah data yang tidak memiliki isi teks.

```
# Hitung jumlah missing value di tiap kolom
df.isna().sum()
```

Hasil ini menunjukkan bahwa tidak ada nilai kosong dalam kolom komentar, sehingga seluruh data dinyatakan valid untuk digunakan pada tahap selanjutnya.

Setelah memastikan tidak ada nilai kosong, langkah berikutnya adalah memeriksa keberadaan data yang terduplikasi. Pemeriksaan ini dilakukan menggunakan fungsi *df.duplicated().sum()* digunakan untuk mengetahui jumlah data duplikat dalam dataset.

```
# Apakah terdapat duplikat
df.duplicated().sum()

np.int64(1792)
```

Hasilnya menunjukkan bahwa terdapat sebanyak 1.792 data duplikat. Untuk menjaga integritas data dan menghindari bias pada proses pelatihan model, seluruh data duplikat tersebut dihapus menggunakan fungsi *df = df.drop_duplicates()*.

```
# Menghapus data duplicate
df = df.drop_duplicates().reset_index(drop = True)

# cek apakah sudah berhasil menghapus data duplikat
df.duplicated().sum()
```

Setelah proses penghapusan dilakukan, fungsi pengecekan ulang menunjukkan nilai 0, yang menandakan bahwa seluruh duplikasi telah berhasil dihapus. Jumlah data yang tersisa setelah pembersihan adalah 3.724 baris data, dan seluruhnya siap digunakan untuk tahap analisis selanjutnya.

4.2.3 *Pre-Processing* Data

Tahapan *pre-processing* merupakan fondasi penting dalam pengolahan data teks, yang bertujuan untuk mengubah data mentah menjadi format yang lebih rapi dan terstruktur sehingga dapat digunakan secara optimal dalam analisis atau pelatihan model. Langkah ini sangat berpengaruh terhadap performa algoritma yang digunakan, karena kualitas data yang buruk dapat menghasilkan hasil yang tidak akurat. Data komentar yang diperoleh dari platform TikTok umumnya masih mengandung berbagai gangguan, seperti format penulisan yang tidak seragam, simbol atau karakter tidak relevan, serta unsur-unsur lain yang tidak diperlukan.

Oleh karena itu, perlu dilakukan proses pembersihan dan penyesuaian format melalui beberapa tahapan. Berikut ini adalah beberapa tahapan yang dilakukan dalam proses *pre-processing* data.

A. *Cleaning*

Tahap *cleaning* atau pembersihan data merupakan proses awal dalam *pre-processing* yang tidak relevan dari data teks. Pada tahap ini, dilakukan penghapusan berbagai elemen yang dianggap sebagai gangguan (*noise*) dalam analisis teks, seperti; karakter spesial dan tanda baca, angka atau simbol numerik, alamat website atau URL, sebutan pengguna (mention, diawali dengan "@"), spasi berlebih yang tidak perlu. Berikut adalah kode yang digunakan untuk melakukan proses *cleaning* pada data komentar.

```
import re
def clean_text(text):
    text = str(text)

# hapus URL
    text = re.sub(r"http\S+|www\S+|tiktok\.com\S*", "", text)
# hapus mention
    text = re.sub(r"@\w+", "", text)
# hapus simbol & angka
    text = re.sub(r"[^a-zA-Z\s]", "", text)
# hapus spasi berlebih
    text = re.sub(r"\s+", " ", text).strip()
    return text

# Terapkan fungsi 'cleaning'
df["cleaning"] = df["comment"].apply(clean_text)

# Tampilkan hasil
df.head()
```

Hasil pembersihan data teks melalui proses *cleaning* ditampilkan pada tabel 4.1 sebagai berikut.

Tabel 4. 1 Hasil *Cleaning* Data

Comment	***Cleaning***
kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa 🥹😍😍	kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa
GUE KIRA ATAP TU ATAP RUMAH BJIR 😭😭	GUE KIRA ATAP TU ATAP RUMAH BJIR
sungkem ahh sama mbak tersopanm 🦁	sungkem ahh sama mbak tersopanm
udah lama nggk liat ATAP 😍	udah lama nggk liat ATAP
Klo sama atap pasti satu frekuensi dari dulu 🥹	Klo sama atap pasti satu frekuensi dari dulu

B. *Case Folding*

Case folding adalah tahap dalam *pre-processing* teks yang bertujuan untuk menyeragamkan semua huruf menjadi format huruf kecil (*lowercase*). Berikut adalah kode yang digunakan untuk melakukan proses *case folding* pada data.

```
# Ubah huruf jadi lowercase
df["casefolding"] = df["cleaning"].str.lower()

# Tampilkan hasil
df[["cleaning", "casefolding"]].head()
```

Hasil pembersihan data teks melalui proses *case folding* ditampilkan pada tabel 4.2 sebagai berikut.

Tabel 4. 2 Hasil *Case Folding*

Cleaning	***Case Folding***
kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa	kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa
GUE KIRA ATAP TU ATAP RUMAH BJIR	gue kira atap tu atap rumah bjir
sungkem ahh sama mbak tersopanm 🦁	sungkem ahh sama mbak tersopanm
udah lama nggk liat ATAP	udah lama nggk liat atap
Klo sama atap pasti satu frekuensi dari dulu	klo sama atap pasti satu frekuensi dari dulu

C. *Tokenizing*

Pemecahan token (*tokenizing*) dilakukan dengan memisahkan teks berdasarkan spasi, sehingga setiap kata dalam kalimat dapat dianalisis secara individual. Berikut adalah kode yang digunakan untuk melakukan proses *tokenizing* pada data komentar.

```
# Memecah teks jadi list kata
df["tokenizing"] = df["casefolding"].apply(lambda x: x.split())

# Tampilkan hasil
df[["casefolding", "tokenizing"]].head()
```

Hasil pembersihan data teks melalui proses *tokenizing* ditampilkan pada tabel 4.3 sebagai berikut.

Tabel 4. 3 Hasil *Tokenizing*

Case Folding	***Tokenizing***
kembali kesetelan awal dengan ria ricis yg bgtu bnyk tmn lawan jeniss dan bu icis terliat sangat bhgiaa	['kembali', 'kesetelan', 'awal', 'dengan', 'ria', 'ricis', 'yg', 'bgtu', 'bnyk', 'tmn', 'lawan', 'jeniss', 'dan', 'bu', 'icis', 'terliat', 'sangat', 'bhgiaa']
gue kira atap tu atap rumah bjir	['gue', 'kira', 'atap', 'tu', 'atap', 'rumah', 'bjir']
sungkem ahh sama mbak tersopanm	['sungkem', 'ahh', 'sama', 'mbak', 'tersopanm']
udah lama nggk liat atap	['udah', 'lama', 'nggk', 'liat', 'atap']
klo sama atap pasti satu frekuensi dari dulu	['klo', 'sama', 'atap', 'pasti', 'satu', 'frekuensi', 'dari', 'dulu']

D. *Normalization*

Normalization merupakan proses untuk menyamakan berbagai bentuk penulisan kata yang memiliki arti sama, terutama dalam konteks bahasa informal. Dalam data teks dari media sosial seperti TikTok, sering ditemukan variasi penulisan seperti singkatan, kata tidak baku, atau istilah gaul yang perlu diseragamkan agar dapat dipahami dengan benar oleh sistem.

Sebelum melakukan proses normalisasi, dilakukan terlebih dahulu *screening* terhadap seluruh kata yang muncul dalam data. Tahap ini bertujuan untuk mengidentifikasi daftar kata unik hasil tokenisasi, yang nantinya digunakan sebagai dasar dalam menyusun kamus normalisasi dalam format CSV. Berikut adalah kode yang digunakan proses *normalization*.

```
# Ambil semua kata dari kolom tokens
semua_kata = set([token for sublist in df["tokenizing"] for
token in sublist])

# Buat DataFrame dari kata yang ditemukan
df_screening = pd.DataFrame(sorted(semua_kata),
columns=["slang"])
df_screening["formal"] = ""
```

Setelah kamus normalisasi berhasil diunggah dan disusun dalam bentuk pasangan kata tidak baku (*slang*) dan bentuk bakunya (formal), proses normalisasi dapat dilakukan dengan kode berikut.

```
# Buat dictionary dari kamus
kamus_dict = dict(zip(df_kamus['slang'], df_kamus['formal']))

# Fungsi normalisasi
def normalisasi_tokens(tokens, kamus):
    return [kamus.get(token, token) for token in tokens]

# Terapkan normalisasi
df['normalization'] = df['tokenizing'].apply(lambda x:
normalisasi_tokens(x, kamus_dict))

# Tampilkan hasil
df[['tokenizing', 'normalization']].head()
```

Hasil pembersihan data teks melalui proses *normalization* ditampilkan pada tabel 4.4 sebagai berikut.

Tabel 4. 4 Hasil *Normalization*

Tokenizing	***Normalization***
['kembali', 'kesetelan', 'awal', 'dengan', 'ria', 'ricis', 'yg', 'bgtu', 'bnyk', 'tmn', 'lawan', 'jeniss', 'dan', 'bu', 'icis', 'terliat', 'sangat', 'bhgiaa']	['kembali', 'pengaturan', 'awal', 'dengan', 'ria', 'ricis', 'yang', 'begitu', 'banyak', 'teman', 'lawan', 'jenis', 'dan', 'ibu', 'ricis', 'terlihat', 'sangat', 'bahagia']
['gue', 'kira', 'atap', 'tu', 'atap', 'rumah', 'bjir']	['saya', 'kira', 'atap', 'itu', 'atap', 'rumah', 'bjir']
['sungkem', 'ahh', 'sama', 'mbak', 'tersopanm']	['sungkem', 'ah', 'sama', 'kak', 'sopan']
['udah', 'lama', 'nggk', 'liat', 'atap']	['sudah', 'lama', 'tidak', 'lihat', 'atap']
['klo', 'sama', 'atap', 'pasti', 'satu', 'frekuensi', 'dari', 'dulu']	['jika', 'sama', 'atap', 'pasti', 'satu', 'frekuensi', 'dari', 'dulu']

E. *Stopword Removal*

Stopword Removal adalah tahap eliminasi kata-kata umum yang tidak memiliki makna penting terhadap analisis. Kata penghenti dalam bahasa Indonesia

meliputi kata-kata seperti "dan", "atau", "yang", "dengan", "untuk", "dari", "ke", "di", "pada", dan kata-kata umum lainnya.

Sebelum menerapkan fungsi penghapusan *stopword*, terlebih dahulu dilakukan proses impor pustaka yang diperlukan. Pustaka NLTK digunakan untuk mengambil daftar *stopwords* bahasa Indonesia, serta untuk menambahkan kata-kata lain yang belum termasuk dalam daftar tersebut. Proses ini dilakukan dengan menjalankan kode berikut.

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

# Ambil daftar stopwords bahasa Indonesia
stop_words = set(stopwords.words('indonesian'))

# Tambahan stopword informal/slang
custom_stopwords = {

# Gabungkan ke stopword NLTK
stop_words.update(custom_stopwords)
```

Setelah daftar *stopwords* berhasil disiapkan, proses penghapusan kata-kata umum dari data dapat dilakukan. Berikut adalah kode yang digunakan dalam proses tersebut.

```
# Fungsi stopword removal
def remove_stopwords(tokens, stopword_set):
    return [token for token in tokens if token not in
stopword_set]

# Terapkan ke kolom normalization
df["stopword"] = df["normalization"].apply(
    lambda tokens: remove_stopwords(tokens, stop_words)
)

# Tampilkan hasil
df[["normalization", "stopword"]].head()
```

Hasil pembersihan data teks melalui proses *Stopword Removal* ditampilkan pada tabel 4.5 sebagai berikut.

Tabel 4. 5 Hasil *Stopword Removal*

Normalization	Stopword Removal
['kembali', 'pengaturan', 'awal', 'dengan', 'ria', 'ricis', 'yang', 'begitu', 'banyak', 'teman', 'lawan', 'jenis', 'dan', 'ibu', 'ricis', 'terlihat', 'sangat', 'bahagia']	['pengaturan', 'teman', 'lawan', 'jenis', 'bahagia']
['saya', 'kira', 'atap', 'itu', 'atap', 'rumah', 'bjir']	['rumah']
['sungkem', 'ah', 'sama', 'kak', 'sopan']	['sungkem', 'sopan']
['sudah', 'lama', 'tidak', 'lihat', 'atap']	['lihat']
['jika', 'sama', 'atap', 'pasti', 'satu', 'frekuensi', 'dari', 'dulu']	['frekuensi']

F. *Stemming*

Stemming adalah aktivitas untuk mengubah kata berimbuhan menjadi bentuk dasar, sehingga kata-kata dengan makna serupa dapat dikelompokkan menjadi satu. Sebelum menerapkan fungsi stemming dilakukan proses instalasi library *Sastrawi*.

```
!pip install Sastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

Setelah proses instalasi library *Sastrawi* selesai, langkah selanjutnya adalah melakukan proses *stemming*. Proses ini dapat dijalankan dengan menggunakan kode berikut.

```
# Buat objek stemmer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Fungsi stemming
def stemming_to_sentence(tokens):
    if not isinstance(tokens, list):
        return ""
    return ' '.join([stemmer.stem(str(token)) for token in
tokens])

# Terapkan ke kolom stopword
df["stemming"] = df["stopword"].apply(stemming_to_sentence)

# Tampilkan hasil
df[["stopword", "stemming"]].head()
```

Hasil pembersihan data teks melalui proses *stemming* ditampilkan pada tabel 4.6 sebagai berikut.

Tabel 4. 6 Hasil *Stemming*

Stopword Removal	***Stemming***
['pengaturan', 'teman', 'lawan', 'jenis', 'bahagia']	atur teman lawan jenis bahagia
['rumah']	rumah
['sungkem', 'sopan']	sungkem sopan
['lihat']	lihat
['frekuensi']	frekuensi

Berikut pada gambar 4.3 merupakan tampilan hasil akhir dari seluruh tahapan *pre-processing* yang telah dilakukan, mulai dari *cleaning*, *case folding*, *tokenizing*, *normalization*, *stopword removal*, hingga *stemming*. Setiap tahapan ditampilkan dalam kolom terpisah untuk menunjukkan transformasi data secara bertahap terhadap komentar yang dianalisis. Hasil setelah *pre-processing*, jumlah data yang tersisa adalah 2.741 baris data.

	comment	cleaning	casefolding	tokenizing	normalization	stopword	stemming
0	kembali kesetelan awal dengan ria ricis yg bgt...	kembali kesetelan awal dengan ria ricis yg bgt...	kembali kesetelan awal dengan ria ricis yg bgt...	[kembali, kesetelan, awal, dengan, ria, ricis,...	[kembali, pengaturan, awal, dengan, ria, ricis...	[pengaturan, teman, lawan, jenis, bahagia]	atur teman lawan jenis bahagia
1	GUE KIRA ATAP TU ATAP RUMAH BJIRð□□ð□□	GUE KIRA ATAP TU ATAP RUMAH BJIR	gue kira atap tu atap rumah bjir	[gue, kira, atap, tu, atap, rumah, bjir]	[saya, kira, atap, itu, atap, rumah, bjir]	[rumah]	rumah
2	sungkem ahh sama mbak tersopanmð□□□	sungkem ahh sama mbak tersopanm	sungkem ahh sama mbak tersopanm	[sungkem, ahh, sama, mbak, tersopanm]	[sungkem, ah, sama, kak, sopan]	[sungkem, sopan]	sungkem sopan
3	udah lama nggk liat ATAP ð□□□	udah lama nggk liat ATAP	udah lama nggk liat atap	[udah, lama, nggk, liat, atap]	[sudah, lama, tidak, lihat, atap]	[lihat]	lihat
4	Klo sama atap pasti satu frekuensi dari duluð□¥º	Klo sama atap pasti satu frekuensi dari dulu	klo sama atap pasti satu frekuensi dari dulu	[klo, sama, atap, pasti, satu, frekuensi, dari...	[jika, sama, atap, pasti, satu, frekuensi, dar...	[frekuensi]	frekuensi
...	...	...	...	...	...	...	...
2736	aduhhhhhhh	aduhhhhhhh	aduhhhhhhh	[aduhhhhhhh]	[aduh]	[aduh]	aduh
2737	hai kk icis sapa aku dongð□¥°ð□□□	hai kk icis sapa aku dong	hai kk icis sapa aku dong	[hai, kk, icis, sapa, aku, dong]	[hai, kakak, ricis, sapa, saya, dong]	[hai, kakak, sapa]	hai kakak sapa
2738	cntikð□□□	cntik	cntik	[cntik]	[cantik]	[cantik]	cantik
2739	slaam dari kota pklongn icisss	slaam dari kota pklongn icisss	slaam dari kota pklongn icisss	[slaam, dari, kota, pklongn, icisss]	[salam, dari, kota, Pekalongan, ricis]	[salam, kota, Pekalongan]	salam kota kalong
2740	ke sebelah	ke sebelah	ke sebelah	[ke, sebelah]	[ke, sebelah]	[sebelah]	belah

2741 rows × 7 columns

Gambar 4. 3 Hasil *Pre-processing*

4.2.4 Pelabelan Data

Proses pelabelan manual terhadap data teks yang telah melalui seluruh tahapan *pre-processing*. Pelabelan ini menjadi langkah penting untuk membentuk dataset yang berkualitas dan siap digunakan dalam pelatihan model klasifikasi, karena label berfungsi sebagai penanda bagi model dalam memahami dan membedakan antara komentar yang bersifat *cyberbullying* dan yang mengandung unsur non-*cyberbullying*. Dalam penelitian ini, digunakan dua kelas label:

a. Label 1 diberikan jika komentar mengandung unsur *cyberbullying*.
b. Label 0 diberikan jika komentar tidak mengandung *cyberbullying*.

Tabel 4.7 berikut menampilkan hasil akhir dari proses pelabelan manual terhadap data komentar yang telah melalui tahapan *stemming*.

Tabel 4. 7 Contoh Pelabelan Manual

No	***Stemming***	**Label**
1	cakep banget sehat idola idola sehat lancar rezeki moga rezeki tular amin	0
2	cantik banget	0
3	simbol iblis jadi tren naudzubillah min dzalik	1
4	contoh popularitas pakai rusak rezeki orang	1
5	gonta ganti mulu	1
6	viral ikut campur kacau	1
7	putus rezeki orang	1
8	keren banget	0
9	tambah cantik banget senyum	0
10	senang lihat kolaborasi cocok	0

Pada gambar 4.4 berikut menampilkan sebagian dari hasil pelabelan tersebut, yang terdiri dari dua kolom utama yaitu, teks hasil *stemming* dan label klasifikasinya.

	stemming	label
0	atur teman lawan jenis bahagia	0
1	rumah	1
2	sungkem sopan	1
3	lihat	0
4	frekuensi	0
...	...	...
2736	aduh	0
2737	hai kakak sapa	0
2738	cantik	0
2739	salam kota kalong	0
2740	belah	0

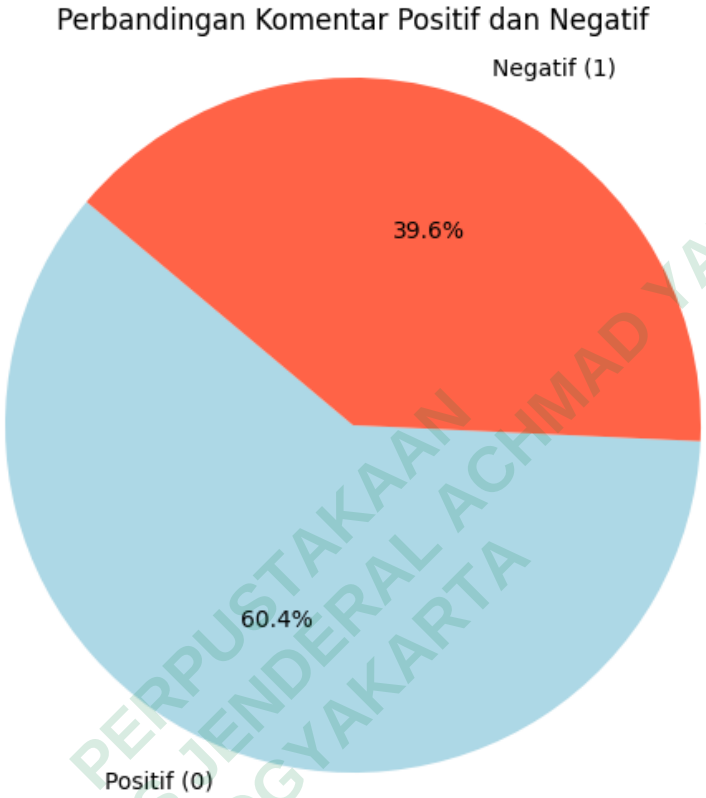
2741 rows × 2 columns

Gambar 4. 4 Hasil *Labeling*

Setelah seluruh komentar diberi label, dilakukan analisis distribusi untuk mengetahui seberapa banyak data yang tergolong dalam kategori positif dan *negative*. Hasil distribusi ditampilkan dalam bentuk diagram lingkaran (*pie chart*) agar perbandingan antar kelas lebih mudah diamati secara visual. Berikut adalah kode yang digunakan untuk menampilkan proporsi label dalam bentuk *pie chart*.

```
import matplotlib.pyplot as plt

# Hitung jumlah masing-masing label
labels_proposition = df['label'].value_counts().sort_index()

# Pie chart
plt.figure(figsize=(8, 6))
plt.pie(
    labels_proposition,
    labels=['Positif (0)', 'Negatif (1)'],
    autopct='%1.1f%%',
    startangle=140,
    colors=['lightblue', 'tomato']
)
plt.title('Perbandingan Komentar Positif dan Negatif')
# agar lingkaran tidak oval
plt.axis('equal')
plt.show()
```

Pada gambar 4.5 menunjukkan proporsi data, di mana sebanyak 60,4% komentar tidak mengandung *cyberbullying* (label 0), dan 39,6% komentar teridentifikasi mengandung *cyberbullying* (label 1).



Gambar 4. 5 Hasil Visualisasi *Labeling*

Selanjutnya dilakukan visualisasi lanjutan untuk mengidentifikasi pola kata yang sering digunakan pada kelas tersebut. Teknik *WordCloud* digunakan untuk memetakan kata-kata yang paling sering muncul dalam komentar positif. Berikut adalah kode yang digunakan untuk menampilkan kata-kata paling sering muncul dalam komentar berdasarkan kategorinya.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Positive Wordcloud
positive_com_wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(positive_com))

plt.figure(figsize=(10, 5))
plt.imshow(positive_com_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Pada gambar 4.6 menampilkan sebaran kata yang paling sering digunakan pada komentar kategori non-*cyberbullying*, dengan ukuran font yang lebih besar merepresentasikan tingkat frekuensi kata yang lebih tinggi dalam kumpulan data.

Gambar 4. 6 Hasil Visualisasi *WordCloud*

Untuk memahami lebih lanjut pola bahasa yang sering muncul dalam komentar yang tergolong *cyberbullying*, teknik *WordCloud* digunakan untuk memetakan kata-kata yang paling sering muncul dalam komentar negatif. Berikut adalah kode yang digunakan untuk menampilkan kumpulan kata berdasarkan frekuensi kemunculannya pada masing-masing kategori komentar.

```
# Negative Wordcloud
negative_com_wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(negative_com))

plt.figure(figsize=(10, 5))
plt.imshow(negative_com_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Pada gambar 4.7 menampilkan sebaran kata yang paling sering digunakan pada komentar kategori *cyberbullying*, dengan ukuran font yang lebih besar merepresentasikan tingkat frekuensi kata yang lebih tinggi dalam kumpulan data.

Gambar 4. 7 Hasil Visualisasi *WordCloud*

4.2.5 Pembobotan TF-IDF

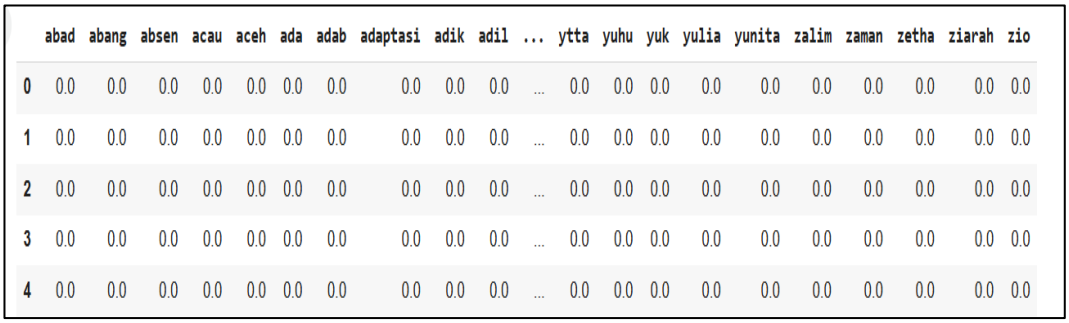
Setelah data berhasil dipisahkan, langkah berikutnya yaitu melakukan ekstraksi fitur dengan memanfaatkan metode *Term Frequency–Inverse Document Frequency* (TF-IDF). Metode ini digunakan untuk merepresentasikan teks dalam bentuk numerik sehingga dapat diproses oleh algoritma klasifikasi. TF-IDF menghitung bobot setiap kata berdasarkan dua hal yaitu, frekuensi kemunculannya dalam satu dokumen (*term frequency*) dan tingkat kelangkaannya di seluruh kumpulan dokumen (*inverse document frequency*). Tahapan ini melibatkan perhitungan:

a. DF (*Document Frequency*) untuk mengetahui seberapa banyak dokumen mengandung kata tertentu.
b. TF (*Term Frequency*) untuk mencatat frekuensi kata dalam satu komentar.
c. IDF (*Inverse Document Frequency*) untuk mengurangi bobot kata-kata yang terlalu umum.

Untuk menerapkan TF-IDF pada data komentar yang telah melalui proses *stemming*, digunakan pustaka *TfidfVectorizer* dari library *scikit-learn*. Data teks dipisahkan sebagai variabel fitur (x) dan label (y), kemudian dilakukan proses pembobotan menggunakan TF-IDF. Berikut adalah kode yang digunakan dalam proses tersebut.

```
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer

# Pisahkan objek
x = df['stemming']
y = df['label']

# Pembobotan TF-IDF
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(x)
feature_names = vectorizer.get_feature_names_out()

# Konversi TF-IDF matrix ke DataFrame
tfidf_df = pd.DataFrame(X_tfidf.toarray(),
columns=feature_names)

# Tampilkan hasilnya
tfidf_df.head()
```

Pembobotan TF-IDF menghasilkan nilai numerik untuk setiap kata. Nilai tersebut ditampilkan pada gambar 4.8 berikut.



	abad	abang	absen	acau	aceh	ada	adab	adaptasi	adik	adil	...	ytta	yuhu	yuk	yulia	yunita	zalim	zaman	zetha	ziarah	zio
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Gambar 4. 8 Hasil Perhitungan TF-IDF

Untuk memperjelas cara kerja perhitungan TF, IDF, dan TF-IDF, tabel 4.8 berikut menyajikan contoh hasil pembobotan terhadap beberapa term dari beberapa dokumen komentar.

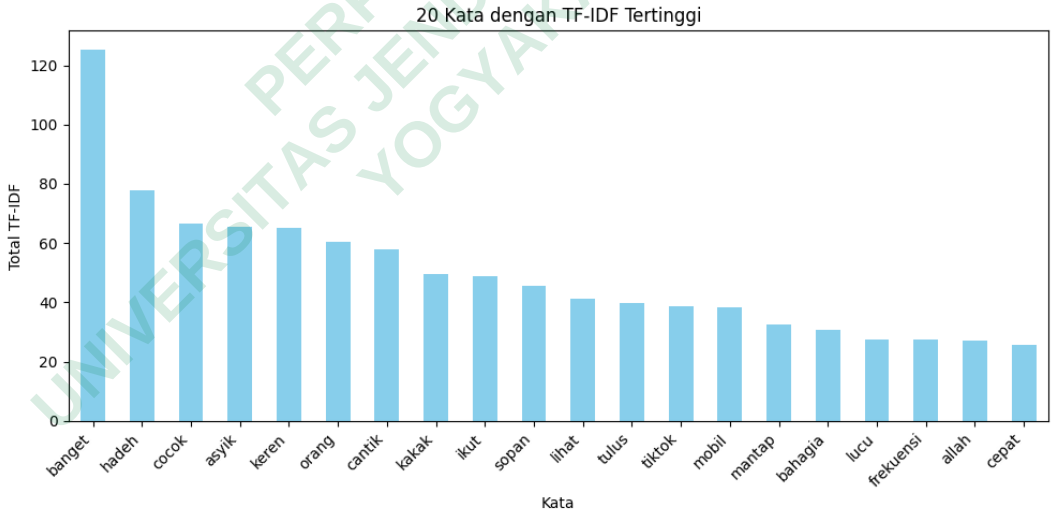
Tabel 4. 8 Hasil Perhitungan TF-IDF

No	Dokumen	Term	TF	IDF	TF-IDF
1	Doc 1	atur	0.2	1.0986	0.2197
2	Doc 1	teman	0.2	1.0986	0.2197
3	Doc 1	lawan	0.2	1.0986	0.2197
4	Doc 1	jenis	0.2	1.0986	0.2197
5	Doc 1	bahagia	0.2	1.0986	0.2197
6	Doc 2	rumah	1.0	1.0986	1.0986
7	Doc 3	sungkem	0.5	1.0986	0.5493
8	Doc 3	sopan	0.5	1.0986	0.5493

Untuk mengetahui kata-kata yang paling berpengaruh dalam dataset, dilakukan visualisasi terhadap 20 kata dengan nilai TF-IDF tertinggi secara keseluruhan. Berikut merupakan kode untuk menghasilkan visualisasi tersebut.

```
#Visualisasi kata paling penting secara keseluruhan
import matplotlib.pyplot as plt

# Hitung total TF-IDF tiap kata (abaikan kolom label)
top_words = tfidf_df.drop(columns='label',
errors='ignore').sum().sort_values(ascending=False).head(20)

# Buat grafik batang
plt.figure(figsize=(10, 5))
top_words.plot(kind='bar', color='skyblue')
plt.title("20 Kata dengan TF-IDF Tertinggi")
plt.ylabel("Total TF-IDF")
plt.xlabel("Kata")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Pada gambar 4.9 menampilkan 20 kata dengan nilai TF-IDF tertinggi. Kata-kata tersebut dianggap paling berpengaruh dalam dataset karena memiliki bobot paling besar dalam representasi dokumen.



Gambar 4. 9 Hasi Visualisasi Kata Tertinggi

4.2.6 *Spliting* Data

Proses pembagian data dalam penelitian ini dilakukan menggunakan teknik *Stratified K-Fold Cross Validation* dengan jumlah *fold* sebanyak 5. Teknik ini membagi dataset menjadi lima bagian yang memiliki proporsi label yang seimbang, sehingga distribusi kelas antara komentar yang mengandung *cyberbullying* dan

yang tidak tetap terjaga di setiap *fold*. Dataset yang digunakan terdiri dari 2.741 komentar, yang kemudian dibagi ke dalam lima bagian. Setiap *fold* akan mengambil sekitar 2.192 data untuk data latih, dan 549 data untuk data uji. Proses ini dilakukan sebanyak lima kali, dan pada setiap putaran, bagian data yang digunakan untuk pengujian akan bergiliran, sehingga seluruh data digunakan secara merata.

Untuk mengetahui kinerja model pada setiap bagian data, digunakan metode pembagian data *Stratified K-Fold Cross Validation* sebanyak 5-*fold*. Proses ini ditampilkan dalam kode berikut.

```
# Menentukan jumlah pembagian data (5-fold)
fold = 5

# Membuat objek pembagi data
cv = StratifiedKFold(n_splits=fold, shuffle=True,
random_state=2025)
```

4.2.7 Klasifikasi Model *K-Nearest Neighbor* (KNN)

Setelah proses pembagian data dilakukan, tahap selanjutnya klasifikasi dengan algoritma *K-Nearest Neighbor* (KNN). Pada tahap ini, model diuji dengan berbagai nilai k (jumlah tetangga) dari 1 hingga 10, untuk mengetahui nilai k mana yang menghasilkan akurasi terbaik. Evaluasi dilakukan menggunakan teknik *Stratified K-Fold Cross Validation* sebanyak 5-*fold*, dengan membandingkan performa model pada setiap nilai k berdasarkan akurasi rata-rata dari setiap *fold*. Berikut kode yang digunakan dalam proses klasifikasi ini.

```
# Iterasi nilai k dari 1 hingga 10
for k in range(1, 11):
    hasil_evaluasi = []

# Proses Stratified K-Fold Cross Validation
    for k_fold, (train_idx, test_idx) in
enumerate(cv.split(X_tfidf, y)):
        x_train, x_test = X_tfidf[train_idx], X_tfidf[test_idx]
        y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

# Inisialisasi dan Latih Model KNN
        knn = KNeighborClassifier(n_Neighbor=k)
        knn.fit(x_train, y_train)
        y_pred = knn.predict(x_test)

        total_data = len(y_test)
        benar = (y_test == y_pred).sum()
        salah = total_data - benar
        akurasi = round(accuracy_score(y_test, y_pred) * 100,
2)

        hasil_evaluasi.append({
            "k_fold": k_fold + 1,
            "prediksi benar": benar,
            "prediksi salah": salah,
            "data test": total_data,
            "akurasi": akurasi
        })

    semua_hasil[k] = pd.DataFrame(hasil_evaluasi)
```


Setelah proses klasifikasi dilakukan pada berbagai nilai k, selanjutnya menghitung rata-rata akurasi dari setiap nilai k untuk menentukan performa model yang paling optimal. Nilai k dengan akurasi rata-rata tertinggi kemudian dipilih sebagai parameter terbaik untuk model *K-Nearest Neighbor* (KNN). Kode berikut digunakan untuk melakukan perhitungan akurasi rata-rata dan menampilkan hasil evaluasi dari setiap *fold*.

```
# Inisialisasi nilai awal untuk menyimpan k terbaik
akurasi_tertinggi = -1
k_terbaik = -1

# Dictionary untuk menyimpan rata-rata akurasi setiap nilai k
rata2_akurasi = {}

# Untuk menghitung rata-rata akurasi dari hasil semua nilai k
for k, df in semua_hasil.items():
    avg = df['akurasi'].mean()
    rata2_akurasi[k] = avg

# Periksa apakah akurasi ini adalah yang tertinggi sejauh ini

    if avg > akurasi_tertinggi:
        akurasi_tertinggi = avg
        k_terbaik = k

# Cetak hasil evaluasi model
print(f"EVALUASI MODEL KNN DENGAN {fold} FOLD")
print(f"TOTAL DATA KESELURUHAN: {X_tfidf.shape[0]} data\n")

# Tampilkan hasil evaluasi tiap nilai k
for k, df in semua_hasil.items():
    print(f"tetangga k-{k}")
    print(df.to_string(index=False))

# Tambahkan tanda centang jika ini adalah k dengan akurasi
terbaik
    extra_info = " (✓ AKURASI TERBAIK)" if k == k_terbaik else
""
    print(f"\nAkurasi rata-rata dari {fold} fold:
{rata2_akurasi[k]:.2f}%{extra_info}\n")
```


Evaluasi performa model *K-Nearest Neighbor* (KNN) dilakukan terhadap setiap variasi nilai k, dengan menggunakan teknik validasi silang sebanyak 5-*fold*. Dari hasil tersebut, diperoleh akurasi rata-rata untuk masing-masing nilai k, yang disajikan dalam tabel 4.9 sebagai berikut.

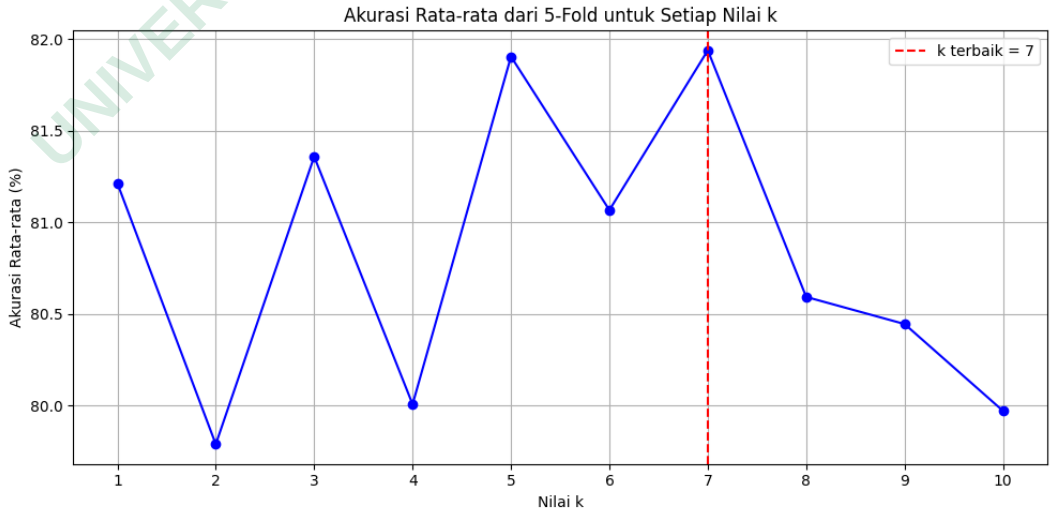
Tabel 4. 9 Hasil Rata-rata Akurasi Setiap Nilai k

Nilai k	Akurasi Rata-rata (%)
k-1	81.21%
k-2	79.79%
k-3	81.36%
k-4	80.01%
k-5	81.90%
k-6	81.07%
k-7	81.94%
k-8	80.59%
k-9	80.44%
k-10	79.97%

Agar lebih mudah memahami perbandingan performa model *K-Nearest Neighbor* (KNN) pada setiap variasi nilai k, dilakukan visualisasi dalam bentuk grafik garis. Grafik ini menampilkan akurasi rata-rata dari masing-masing nilai k, serta menunjukkan titik maksimum sebagai penanda k terbaik yang diperoleh dari evaluasi sebelumnya. Berikut merupakan kode untuk menghasilkan visualisasi tersebut.

```
# Buat list dari k dan akurasi rata-ratanya
k_values = list(rata2_akurasi.keys())
avg_accuracies = list(rata2_akurasi.values())

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(k_values, avg_accuracies, marker='o', color='blue')
plt.title("Akurasi Rata-rata dari 5-Fold untuk Setiap Nilai k")
plt.xlabel("Nilai k")
plt.ylabel("Akurasi Rata-rata (%)")
plt.xticks(k_values)  # supaya nilai k terlihat
plt.grid(True)

# Tandai k terbaik
plt.axvline(x=k_terbaik, color='red', linestyle='--', label=f'k
terbaik = {k_terbaik}')
plt.legend()

plt.tight_layout()
plt.show()
```

Gambar 4.10 berikut menampilkan grafik hubungan antara nilai k dengan rata-rata akurasi model yang diperoleh dari proses *Stratified K-Fold Cross Validation* sebanyak 5-*fold*. Setiap titik mewakili akurasi rata-rata untuk nilai k tertentu, dan garis vertikal berwarna merah menunjukkan posisi nilai k terbaik yang menghasilkan akurasi tertinggi.



Gambar 4. 10 Hasil Visualisasi Rata-rata Akurasi Nilai k

Berdasarkan grafik pada Gambar 4.10, terlihat bahwa model *K-Nearest Neighbor* (KNN) mencapai performa tertinggi pada nilai k = 7, dengan akurasi rata-rata sebesar 81,94%. Nilai ini menjadi titik puncak dari tren akurasi, sehingga dipilih sebagai parameter optimal dalam proses klasifikasi pada penelitian ini. Sementara nilai k lainnya juga menunjukkan performa yang cukup baik, namun tidak melampaui nilai akurasi yang dicapai oleh k = 7.

4.2.8 Evaluasi Model

Tahap selanjutnya adalah melakukan evaluasi menyeluruh terhadap performa model menggunakan nilai k-7 yang telah ditetapkan sebagai parameter optimal. Proses evaluasi dilakukan dengan menghasilkan *confusion matrix* dan *classification report*, termasuk *accuracy*, *precision*, *recall*, serta *f1-score*. Berikut kode yang digunakan untuk melakukan evaluasi tersebut.

```
# Inisialisasi ulang StratifiedKFold
cv = StratifiedKFold(n_splits=fold, shuffle=True,
random_state=2025)
# Simpan semua label asli dan prediksi
y_true_all = []
y_pred_all = []
# Lakukan KFold lagi dengan k terbaik
for train_idx, test_idx in cv.split(X_tfidf, y):
    x_train, x_test = X_tfidf[train_idx], X_tfidf[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    knn = KNeighborClassifier(n_Neighbor=k_terbaik)
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)

    y_true_all.extend(y_test)
    y_pred_all.extend(y_pred)
# Evaluasi keseluruhan
print(f"=== Confusion Matrix (k={k_terbaik}) ===")
print(confusion_matrix(y_true_all, y_pred_all))

print("\n=== Classification Report ===")
print(classification_report(y_true_all, y_pred_all, digits=2))

print(f"\nAkurasi Keseluruhan: {accuracy_score(y_true_all,
y_pred_all) * 100:.2f}%")
```

Pada tabel 4.10 ditampilkan ringkasan evaluasi model berdasarkan *confusion matrix*, termasuk nilai *precision*, *recall*, *f1-score* untuk tiap kelas, dan tingkat akurasi total dari model klasifikasi yang digunakan.

Tabel 4. 10 Rangkuman Perhitungan *Confusion Matrix*

Confusion Matrix				
Kelas	***Precision***	***Recall***	***F1- Score***	***Accuracy***
0 (Positif)	82%	90%	86%	81.94%
1 (Negatif)	82%	70%	0.76%	

Pada gambar 4.11 berikut menampilkan bahwa model klasifikasi *K-Nearest Neighbor* (KNN) dengan k=7 memberikan performa yang seimbang dengan akurasi sebesar 81,94%. Nilai *precision*, *recall*, dan *f1-score* pada masing-masing kelas juga menunjukkan performa yang cukup baik.

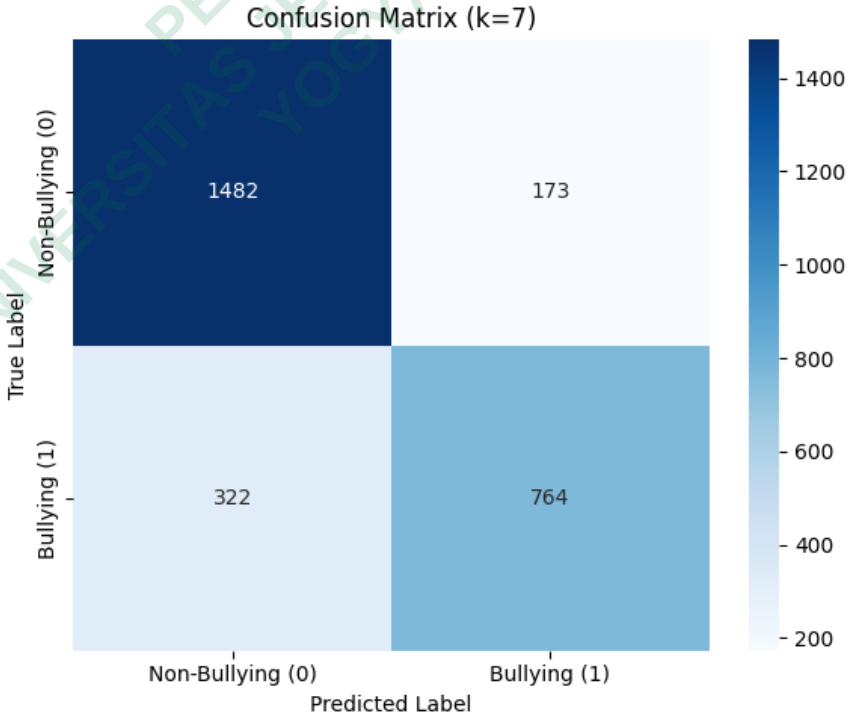
```
=== Confusion Matrix (k=7) ===
[[1482  173]
 [ 322  764]]

=== Classification Report ===
              precision    recall  f1-score   support

           0       0.82      0.90      0.86      1655
           1       0.82      0.70      0.76      1086

    accuracy                           0.82      2741
   macro avg       0.82      0.80      0.81      2741
weighted avg       0.82      0.82      0.82      2741


Akurasi Keseluruhan: 81.94%
```

Gambar 4. 11 Hasil Perhitungan *Confusion Matrix*

Untuk mempermudah interpretasi hasil klasifikasi, *confusion matrix* divisualisasikan dalam bentuk *heatmap*. Visualisasi tersebut memberikan gambaran yang lebih jelas mengenai jumlah prediksi benar dan salah pada masing-masing kelas. Berikut kode yang digunakan untuk membuat visualisasi *confusion matrix*.

```
# Hitung confusion matrix
cm = confusion_matrix(y_true_all, y_pred_all)

# Buat label kelas (0 = Non-Bullying, 1 = Bullying)
labels = ['Non-Bullying (0)', 'Bullying (1)']

# Buat heatmap dari confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=labels, yticklabels=labels)

plt.title(f'Confusion Matrix (k={k_terbaik})')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()
```

Hasil visualisasi *confusion matrix* ditunjukkan pada gambar 4.12. Warna yang lebih gelap menampilkan jumlah prediksi yang lebih tinggi, sehingga memudahkan dalam memahami seberapa baik model dalam mengklasifikasikan komentar ke dalam masing-masing kelas.



Gambar 4. 12 Hasil Visualisasi *Confusion Matrix*

Sebagai bentuk pelengkap dari visualisasi *confusion matrix* yang ditampilkan pada gambar 4.12, tabel berikut disajikan guna merepresentasikan hasil klasifikasi secara numerik.

Tabel 4. 11 Hasil *Confusion Matrix*

	Predicted Class	
Actual Class	**Positive**	**Negative**
Positive	TP = 1482	FN = 173
Negative	FP = 322	TN = 764

Berdasarkan *confusion matrix* pada tabel 4.11, terdapat:

1. True Positive sebanyak 1482 komentar non-*cyberbullying* yang diprediksi benar sebagai non-*cyberbullying*.
2. False Negative sebanyak 173 komentar non-*cyberbullying* yang salah diprediksi sebagai *cyberbullying*.
3. False Positive sebanyak 322 komentar *cyberbullying* yang salah diprediksi sebagai non-*cyberbullying*.
4. True Negative sebanyak 764 komentar *cyberbullying* yang diprediksi benar sebagai *cyberbullying*.

Tahap selanjutnya adalah melakukan evaluasi kinerja model secara menyeluruh. Evaluasi ini mencakup perhitungan terhadap sejumlah metrik penting, yaitu akurasi keseluruhan, *precision*, *recall*, serta *f1-score*. Adapun rincian hasil perhitungannya dapat dilihat pada uraian berikut.

1. Positif (Label 0)

$$Precision = \frac{TP}{TP+FP} = \frac{1482}{1482+322} = \frac{1482}{1804} = 0.82 \times 100 = 82\%$$

$$Recall = \frac{TP}{TP+FN} = \frac{1482}{1482+173} = \frac{1482}{1655} = 0.90 \times 100 = 90\%$$

$$F1\text{-}Score = \frac{2\ (PrecisionxRecall)}{Precision+Recall} = \frac{2\ (0.82 \times 0.90)}{0.82+0.90} = 0.86\ atau\ 86\%$$

2. Negatif (Label 1)

$$Precision = \frac{TN}{TN+FN} = \frac{764}{764+173} = \frac{764}{937} = 0.82 \times 100 = 82\%$$

$$Recall = \frac{TN}{TN+FP} = \frac{764}{764+322} = \frac{764}{1086} = 0.70 \times 100 = 70\%$$

$$F1\text{-}Score = \frac{2\ (PrecisionxRecall)}{Precision+Recall} = \frac{2\ (0.82 \times 0.70)}{0.82+0.70} = 0.76\ atau\ 76\%$$

3. Total Akurasi Keseluruhan

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{1482+764}{1482+764+322+173} = \frac{2246}{2741} = 0.8194 \times 100 = 81.94\%$$

Berdasarkan hasil perhitungan evaluasi, algoritma *K-Nearest Neighbor* (KNN) mencapai performa terbaik pada nilai k=7 dengan rata-rata akurasi keseluruhan sebesar 81,94%. Evaluasi lebih lanjut menunjukkan bahwa model mampu mengklasifikasikan komentar non-*cyberbullying* (label 0) dengan *precision* sebesar 82%, *recall* 90%, dan *f1-score* 86%. Sementara itu, untuk komentar *cyberbullying* (label 1), model menghasilkan *precision* 82%, *recall* 70%, dan *f1-score* 76%. Hasil ini menunjukkan bahwa algoritma *K-Nearest Neighbor* (KNN) dapat melakukan klasifikasi dengan cukup baik, meskipun masih diperlukan peningkatan khusus dalam mendeteksi komentar bermuatan negatif agar hasil klasifikasi lebih seimbang.

Jika dibandingkan dengan penelitian lain yang menggunakan platform dan algoritma serupa, terdapat hasil evaluasi yang menunjukkan akurasi hingga 90%, bahkan pendekatan menggunakan algoritma lain seperti SVM menghasilkan akurasi mendekati 88%. Perbedaan ini kemungkinan disebabkan oleh karakteristik dataset, teknik *pre-processing* yang digunakan, serta metode pelabelan yang diterapkan. Pada penelitian ini, pelabelan dilakukan secara manual terhadap komentar hasil *scraping* yang belum memiliki *ground truth* baku, sehingga memungkinkan adanya perbedaan interpretasi dalam proses anotasi.