BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Pada bab ini penelitian berfokus pada perbandingan keakuratan metode NB dan KNN untuk menganalisis ulasan pada aplikasi Halodoc di *google play store*. Penelitian ini melakukan pengambilan data dengan cara *scraping* data di *google play store*. Scrapping data dari tanggal 23 April 2024.

4.2 HASIL PENELITIAN

Pada tanggal 23 April 2024, hasil penelitian di *google play store* terhadap aplikasi Halodoc menunjukkan bahwa jumlah pengguna yang tercatat sebanyak 7945 data *review*.

4.2.1 Hasil Pengumpulan Data

Dengan memanfaatkan teknik *scraping* dan bantuan library ‘scraper’, memudahkan proses pengambilan data di *google play store* . hasil dari proses tersebut telah disajikan dalam table 2 sebagai data yang telah di *scrape* dari *google play store*.

Tabel 2 Hasil Scrapping

No	Content	Score
1.	Aplikasi yang sangat membantu banyak orangâ¥¸• dokter disini sangat berpengalaman dibidangnya â•¤ï¸• dan sesuai dgn jobdesknya masingÂ². Semua jenis obat tersedia lengkap. FiturÂ²nya sangat menarik terimakasih Hallo Doc. Sekarang konsultasi dgn dokter & beli obat bisa 24 jam kapan saja dimana saja.	5
2.	Sangat disayangkan ketika pasien mau konsul ke dokter tapi telat 5 sampai 10 menit dengan alasan yang jelas tiba-tiba di cancel. padahal udah bayar dan bayarnya cukup expensive untuk harga Pelajar. Sudah kejadian selama 2 kali kayak gini terus. bahkan yang terakhir cuma telat 3 menit udah di cancel. Kita juga bayar ya di sini pakai uang bayarnya bukan pakai daun tapi kalau	1

	ada fitur kayak gini mendingan nggak usah konsultasi di halodoc ke offline aja. Mintol diperbaiki lagi atau perlu ditiadakan	
3.	Dana sudah terpotong, tibaÂ² dokternya sibuk Telfon cs infonya karna dokter sudah terbooking duluan ke yg lainnya. Harusnya jika sistem cepet2an bookingnya, pembayaran yg saya lakukan gagal atau tidak akan masuk dan saldo saya tidak terpotong. Estimasi dana kembali di 1 jam, sedangkan pemotongan saldo dalam hitungan detik.	1
4.	bagus tapi kalau saran dari saya mah kalau kita udah konsul ke salah satu dokter tolong jangan bayar lagi karena susah gitu ketika gak ada uang dan ada masalah dalam perawatan supaya gampang mencari atau ngasih tau bahwa ada kendala dan solusinya	2
5.	Dokter menyenangkan utk konsultasi, cepat tanggap dan mudah utk dimengerti, mantappp	5
6.	Solusi mudah berkonsultasi dengan dokter tanpa perlu datang ke faskes. Selain itu juga memudahkan dalam membeli alkes dan obat2an.	4
7.	Kenapa saat konsultasi,, jika resep sudah diberikan oleh dokter,, sudah ga bisa lagi dirubah? Saat klik cek harga,, baru diketahui apakah tercover asuransi atau tidaknya.. jd obat tidak bisa diganti lagi.. Mohon diperbaiki utk hal ini..	3
8.	Konsultasi dan berobat yg Aku suka dari hallo doc mudah simple	5
9.	Ini ngisi saldo nya bgimana sih? Gak ada tutorial sama sekali. Gopay saya lg bermasalah, mau top up ke halodoc kok gak bisa. Tolong TUTORIAL nya,	1
10.	Ini kenapa ya aku mau daftar kok ga bisa membalas gitu udah masukin nomor sama kode nya ini gimana ya tolong dok halodoc kok jadi begini sihhh udah ada 10x aku coba dia ngulang" Mulu, gimana nihhhh??????	1

Ketidakkonsistenan data dapat mengakibatkan hambatan dalam pemrosesan dan analisis data. Oleh karena itu, dari jumlah data sebelumnya sebanyak 7945 sekarang menjadi 7433 setelah data yang tidak konsisten dihapus untuk memastikan keberlangsungan subjektivitas dan keakuratan analisis.

4.3 HASIL PREPROCESSING DATA

Setelah mendapatkan data dari *Google Play Store* berupa *review*, langkah selanjutnya adalah proses preprocessing. Tahap ini bertujuan untuk membersihkan data. Proses preprocessing melibatkan beberapa tahap sebagai berikut :

4.3.1 Case Folding

Berikut ini adalah hasil dari proses case folding yang bertujuan untuk mengubah seluruh huruf dalam teks menjadi huruf kecil atau besar. Kode program ini digunakan untuk menyamakan bentuk huruf sehingga analisis teks dapat dilakukan dengan lebih konsisten. Kode program dapat dilihat pada gambar 4.1.

```
def case_folding(content):
    lower_word =  content.lower()
    return lower_word
dt_content['case_folding']= dt_content['content'].str.lower()

dt_content[['content','case_folding']]
```

Gambar 4.1 Kode Program Case Folding

Hasil dari proses case folding ditunjukkan pada gambar 4.2, dimana program menggunakan perintah ‘str.lower’ untuk mengubah huruf kapital menjadi huruf kecil.

	content	case_folding
0	baik untuk konsultasi	baik untuk konsultasi
1	Responnya kurang cepat. Dengan waktu yang sang...	responnya kurang cepat. dengan waktu yang sang...
2	Chat dengan dokter langsung di balas.. Dan dok...	chat dengan dokter langsung di balas.. dan dok...
3	Ini sangat membantu, dokternya juga ramah" dan...	ini sangat membantu, dokternya juga ramah" dan...
4	U terapi psikologi tidak perlu waktu tunggu, m...	u terapi psikologi tidak perlu waktu tunggu, m...
...	...	...
7758	Bagus dan cepat	bagus dan cepat
7759	Alhamdulilah bisa terbantu tanpa harus kunjung...	alhamdulilah bisa terbantu tanpa harus kunjung...
7760	Saya tes PCR di cito melalui halodoc tgl 28 de...	saya tes pcr di cito melalui halodoc tgl 28 de...
7761	dokternya baik banget ramah,enak diresapi kons...	dokternya baik banget ramah,enak diresapi kons...
7762	Suka dg aplikasi. Saran buat bisa kasih tips k...	suka dg aplikasi. saran buat bisa kasih tips k...

7763 rows × 2 columns

Gambar 4.2 Hasil Case Folding

4.3.2 Cleaning

Program *Special Removal* adalah kode atau algoritma yang digunakan untuk menghapus karakter-karakter khusus atau symbol-simbol tertentu dari teks atau data. Gambar 4.3 menampilkan kode program *special removal*.

```
def remove_tweet_special(case_folding):
    if isinstance(case_folding, str):
        text = case_folding.replace('\\t'," ").replace('\\n'," ").replace('\\',"")
        text = text.encode('ascii', 'replace').decode('ascii')
        text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
        return text.replace("http://", " ").replace("https://", " ")
    else:
        return ""
dt_content['clean']= dt_content['case_folding'].swifter.apply(remove_tweet_special)
```

Gambar 4.3 Kode Program Special Removal

1. *Number Removal* adalah proses yang digunakan untuk menghapus angka dalam teks data. Gambar 4.4 menampilkan kode program *number removal.*

```
def remove_number(clean):
    text = re.sub(r"\d+", "", clean)
    return text
dt_content['clean'] = dt_content['clean'].swifter.apply(remove_number)
```

Gambar 4.4 Kode Program Number Removal

2. *Punctuation Removal* merupakan teknik pemrosesan teks yang bertujuan untuk menghilangkan tanda baca dari teks. Teknik ini bermanfaat untuk membersihkan teks dari karakter-karakter seperti titik, koma, tanda tanya dan tanda seru. Gambar 4.5 menampilkan kode program *punctuation removal*

```
def remove_punctuation(clean):
    clean_spcl = re.compile('[/(){}\[\]\|@,;]')
    clean_symbol = re.compile('[^0-9a-z]')
    text = clean_spcl.sub('', clean)
    text = clean_symbol.sub(' ', clean)
    return text
dt_content['clean'] = dt_content['clean'].swifter.apply(remove_punctuation)
```

Gambar 4.5 Kode Program Punctuation Removal

3. *Whitespaces Removal* merupakan teknik pemrosesan teks yang bertujuan untuk menghilangkan spasi tambahan tab dan tab yang berada di awal dan

akhir kalimat. Gambar 4.6 menampilkan kode program *whitespaces removal*.

```
def remove_whitespace(clean):
    corrected = str(clean)
    corrected = re.sub(r"//t",r"\t", corrected)
    corrected = re.sub(r"( )\1+",r"\1", corrected)
    corrected = re.sub(r"(\n)\1+",r"\1", corrected)
    corrected = re.sub(r"(\r)\1+",r"\1", corrected)
    corrected = re.sub(r"(\t)\1+",r"\1", corrected)
    return corrected.strip(" ")
dt_content['clean'] = dt_content['clean'].swifter.apply(remove_whitespace)
```

Gambar 4.6 Kode Program Whitespaces Removal

4. *Single Char* merupakan teknik untuk menghapus karakter tunggal atau karakter individu dalam teks atau data. Misalnya, “@”, “$”, dan “1”. Gambar 4.7 menampilkan kode *single char*.

```
def remove_singl_char(clean):
    singl = re.sub(r"\b[a-zA-Z]\b", "",clean)
    return singl
dt_content['filtering'] = dt_content['clean'].swifter.apply(remove_singl_char)
```

Gambar 4.7 Kode Program Single Char

Proses *cleaning* data meliputi penghapusan karakter khusus, angka, tanda baca, spasi di awal dan akhir kalimat, serta karakter tunggal dalam teks. Hasil dari proses cleaning dapat dilihat pada gambar 4.8.

	case_folding	clean
0	baik untuk konsultasi	baik untuk konsultasi
1	responnya kurang cepat. dengan waktu yang sang...	responnya kurang cepat dengan waktu yang sanga...
2	chat dengan dokter langsung di balas.. dan dok...	chat dengan dokter langsung di balas dan dokte...
3	ini sangat membantu, dokternya juga ramah" dan...	ini sangat membantu dokternya juga ramah dan r...
4	u terapi psikologi tidak perlu waktu tunggu, m...	u terapi psikologi tidak perlu waktu tunggu me...
...	...	...
7758	bagus dan cepat	bagus dan cepat
7759	alhamdulilah bisa terbantu tanpa harus kunjung...	alhamdulilah bisa terbantu tanpa harus kunjung...
7760	saya tes pcr di cito melalui halodoc tgl 28 de...	saya tes pcr di cito melalui halodoc tgl des s...
7761	dokternya baik banget ramah,enak diresapi kons...	dokternya baik banget ramah enak diresapi kons...
7762	suka dg aplikasi. saran buat bisa kasih tips k...	suka dg aplikasi saran buat bisa kasih tips ke...

7763 rows × 2 columns

Gambar 4.8 Hasil Cleaning

4.3.3 Tokenizing

Tokenizing merupakan teknik pemrosesan teks yang bertujuan untuk memecahkan teks atau memisahkan kata-kata dalam teks. Gambar 4.9 menampilkan kode program *tokenizing*.

```
def tokenizing(filtering):
    tokens = word_tokenize(filtering)
    return tokens

dt_content['token'] = dt_content['filtering'].swifter.apply(tokenizing)

dt_content[['token']]
```

Gambar 4.9 Kode Program Tokenizing

Proses tokenizing bertujuan untuk membagi teks menjadi unit-unit kecil yang disebut token atau kata-kata. Tokenizing dilakukan untuk mempermudah proses selanjutnya, yaitu *stopwords*. Hasil dari *tokenizing* dapat dilihat pada gambar 4.10.

	clean	token
0	baik untuk konsultasi	[baik, untuk, konsultasi]
1	responnya kurang cepat dengan waktu yang sanga...	[responnya, kurang, cepat, dengan, waktu, yang...
2	chat dengan dokter langsung di balas dan dokte...	[chat, dengan, dokter, langsung, di, balas, da...
3	ini sangat membantu dokternya juga ramah dan r...	[ini, sangat, membantu, dokternya, juga, ramah...
4	u terapi psikologi tidak perlu waktu tunggu me...	[u, terapi, psikologi, tidak, perlu, waktu, tu...
...	...	...
7758	bagus dan cepat	[bagus, dan, cepat]
7759	alhamdulilah bisa terbantu tanpa harus kunjung...	[alhamdulilah, bisa, terbantu, tanpa, harus, k...
7760	saya tes pcr di cito melalui halodoc tgl des s...	[saya, tes, pcr, di, cito, melalui, halodoc, t...
7761	dokternya baik banget ramah enak diresapi kons...	[dokternya, baik, banget, ramah, enak, diresap...
7762	suka dg aplikasi saran buat bisa kasih tips ke...	[suka, dg, aplikasi, saran, buat, bisa, kasih,...

7763 rows × 2 columns

Gambar 4.10 Hasil Tokenizing

4.3.4 Stopwords

Stopwords merupakan proses untuk membersihkan kata-kata umum yang sering muncul dalam teks contohnya seperti “dan”, “atau”, “yang”, “di” dan sejenisnya. Gambar 4.11 menampilkan kode program *stopwords*.

```
def stopword_removal(tokens):
    if isinstance(tokens, str):
        stopwords = stopword.remove(tokens)
    else:
        stopwords = [stopword.remove(token) for token in tokens]
    return stopwords
dt_content['stopword'] = dt_content['token'].swifter.apply(stopword_removal)

dt_content[['stopword']]
```

Gambar 4.11 Kode Program Stopwords

Proses *tokenizing* bertujuan untuk memisahkan teks menjadi unit-unit kecil yang disebut token atau kata-kata. Kemudian, dilakukan proses *stopwords* yang bertujuan untuk menghapus kata-kata umum yang tidak relevan. Hasil dari *stopwords* dapat dilihat pada gambar 4.12.

	token	stopword
0	[baik, untuk, konsultasi]	[baik, , konsultasi]
1	[responnya, kurang, cepat, dengan, waktu, yang...	[responnya, kurang, cepat, , waktu, , sangat, ...
2	[chat, dengan, dokter, langsung, di, balas, da...	[chat, , dokter, langsung, , balas, , dokter, ...
3	[ini, sangat, membantu, dokternya, juga, ramah...	[, sangat, membantu, dokternya, , ramah, , res...
4	[u, terapi, psikologi, tidak, perlu, waktu, tu...	[u, terapi, psikologi, , perlu, waktu, tunggu,...
...	...	...
7758	[bagus, dan, cepat]	[bagus, , cepat]
7759	[alhamdulilah, bisa, terbantu, tanpa, harus, k...	[alhamdulilah, , terbantu, , , kunjungan, kedo...
7760	[saya, tes, pcr, di, cito, melalui, halodoc, t...	[, tes, pcr, , cito, melalui, halodoc, tgl, de...
7761	[dokternya, baik, banget, ramah, enak, diresap...	[dokternya, baik, banget, ramah, enak, diresap...
7762	[suka, dg, aplikasi, saran, buat, bisa, kasih,...	[suka, dg, aplikasi, saran, buat, , kasih, tip...

7763 rows × 2 columns

Gambar 4.12 Hasil Stopwords

4.3.5 Stemming

Stemming merupakan teknik pemrosesan teks yang bertujuan untuk mengubah kata menjadi bentuk dasarnya dengan menghilangkan imbuhan-imbuhan, sehingga kata tersebut menjadi lebih sederhana dan mudah untuk dianalisis. Contohnya, kata "berlari" akan disederhanakan menjadi "lari". Gambar 4.13 menampilkan kode program *Stemming*.

```
def stemming(stopwords):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return stemmer.stem(stopwords)
dt_content['stemmer'] = dt_content['stopword'].swifter.apply(lambda x: [stemming(token) for token in x])

dt_content[['stopword','stemmer']]
```

Gambar 4.13 Kode Program Stemming

Hasil dari stemming adalah sekumpulan kata-kata yang telah disederhanakan menjadi bentuk dasar atau akar kata. Proses stemming menghapus imbuhan atau awalan kata sehingga memungkinkan kata-kata yang memiliki bentuk yang berbeda tetapi memiliki makna yang sama. Untuk hasilnya dapat dilihat pada gambar 4.14.

	stopword	stemmer
0	[baik, , konsultasi]	[baik, , konsultasi]
1	[responnya, kurang, cepat, , waktu, , sangat, ...	[responnya, kurang, cepat, , waktu, , sangat, ...
2	[chat, , dokter, langsung, , balas, , dokter, ...	[chat, , dokter, langsung, , balas, , dokter, ...
3	[, sangat, membantu, dokternya, , ramah, , res...	[, sangat, bantu, dokter, , ramah, , responnya...
4	[u, terapi, psikologi, , perlu, waktu, tunggu,...	[u, terapi, psikologi, , perlu, waktu, tunggu,...
...	...	...
7758	[bagus, , cepat]	[bagus, , cepat]
7759	[alhamdulilah, , terbantu, , , kunjungan, kedo...	[alhamdulilah, , bantu, , , kunjung, dokter]
7760	[, tes, pcr, , cito, melalui, halodoc, tgl, de...	[, tes, pcr, , cito, lalu, halodoc, tgl, des, ...
7761	[dokternya, baik, banget, ramah, enak, diresap...	[dokter, baik, banget, ramah, enak, resap, kon...
7762	[suka, dg, aplikasi, saran, buat, , kasih, tip...	[suka, dg, aplikasi, saran, buat, , kasih, tip...

7763 rows × 2 columns

Gambar 4.14 Hasil Stemming

4.3.6 Normalization

Normalization adalah langkah mengubah kata-kata non-baku menjadi kata-kata yang umum dan memiliki makna. Tujuannya adalah untuk menghasilkan teks yang lebih seragam dan mudah diproses dalam analisis teks selanjutnya. Ini membantu memudahkan pemahaman dan pengolahan data teks secara efisien, kode program dapat dilihat pada gambar 4.15.

```
def normalizing(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    normalized_tokens = [norm_dict.get(token, token) for token in lemmatized_tokens]
    normalized_text = " ".join(normalized_tokens)
    return normalized_text

dt_content['normalizing'] = dt_content['stemmer'].swifter.apply(normalizing)

dt_content[['stemmer','normalizing']]
```

Gambar 4.15 Kode Program Normalization

Proses preprocessing yang terakhir adalah normalization. Pada proses ini dilakukan pengubahan kata-kata non baku menjadi kata-kata yang lebih umum dan memiliki makna yang jelas. Tujuannya adalah untuk menghasilkan representasi teks yang seragam dan mudah diproses dalam analisis teks berikutnya. Hasil dari normalization dapat dilihat pada gambar 4.16.

	stemmer	normalizing
0	[baik, , konsultasi]	baik konsultasi
1	[responnya, kurang, cepat, , waktu, , sangat, ...	responnya kurang cepat waktu sangat singkat
2	[chat, , dokter, langsung, , balas, , dokter, ...	chat dokter langsung balas dokter nya ramah...
3	[, sangat, bantu, dokter, , ramah, , responnya...	sangat bantu dokter ramah responnya cepat c...
4	[u, terapi, psikologi, , perlu, waktu, tunggu,...	untuk terapi psikologi perlu waktu tunggu men...
...	...	...
7758	[bagus, , cepat]	bagus cepat
7759	[alhamdulilah, , bantu, , , kunjung, dokter]	alhamdulilah bantu kunjung dokter
7760	[, tes, pcr, , cito, lalu, halodoc, tgl, des, ...	tes pcr cito lalu halodoc tanggal de tangga...
7761	[dokter, baik, banget, ramah, enak, resap, kon...	dokter baik banget ramah enak resap konsultasi
7762	[suka, dg, aplikasi, saran, buat, , kasih, tip...	suka dengan aplikasi saran buat kasih tip dr...

7763 rows × 2 columns

Gambar 4.16 Hasil Normalization

4.3.7 Menghapus Nan

Kode pada gambar 4.17 adalah kode untuk melakukan 3 langkah pembersihan data pada kolom ‘normalizing’ dari DataFrame dt_content. Kode pertama untuk menghapus baris yang memiliki nilai NaN(kosong) atau teks kosong pada kolom ‘normalizing’, kode kedua untuk menghapus spasi berlebih sehingga setiap baris di kolom tersebut hanya memiliki satu spasi antar kata, dan untuk kode ketiga menghapus baris yang memiliki nilai duplikat pada kolom ‘normalizing’ sehingga setiap baris memiliki nilai yang unik.

```
# Menghapus NaN dan teks kosong
dt_content = dt_content.dropna(subset=['normalizing'])
dt_content = dt_content[dt_content['normalizing'].str.strip() != '']

# Menghapus spasi lebih dari 1
dt_content['normalizing'] = dt_content['normalizing'].str.strip().str.replace(r'\s+', ' ', regex=True)

# Menghapus duplikat
dt_content = dt_content.drop_duplicates(subset=['normalizing'])

✓ 0.1s
```

Gambar 4.17 Kode Penghapus NaN

4.4 HASIL LABELING

Tahap pelabelan menggunakan rating pada ulasan merupakan proses yang melibatkan analisis mendalam terhadap sentimen yang terkandung dalam setiap ulasan. Dalam proses ini, rating yang diberikan oleh pengguna dijadikan salah satu faktor penting untuk mengklasifikasikan ulasan ke dalam dua kategori sentimen yaitu ada positif dan negatif. Rating yang tinggi cenderung menandakan sentimen positif, sedangkan rating rendah cenderung menunjukkan sentimen negatif. Hasil pelabelan dapat dilihat pada gambar 4.18. Dari proses, didapatkan data pelabelan sebanyak 5474 data positif dan 1958 data negatif. Untuk jumlah label dapat dilihat pada gambar 4.19.

	normalizing	sentimen	label
0	baik konsultasi	positif	1
1	responnya kurang cepat waktu sangat singkat	negatif	-1
2	chat dokter langsung balas dokter nya ramah...	positif	1
3	sangat bantu dokter ramah responnya cepat c...	positif	1
4	u terapi psikologi perlu waktu tunggu mental ...	positif	1
...	...	...	...
7427	sangat praktis beli obat alat sehat masker k...	positif	1
7428	alhamdulilah bantu kunjung dokter	positif	1
7429	te pcr cito lalu halodoc tgl de tgl jan has...	negatif	-1
7430	dokter baik banget ramah enak resap konsultasi	positif	1
7431	suka dg aplikasi saran buat kasih tip driver...	positif	1

7432 rows × 3 columns

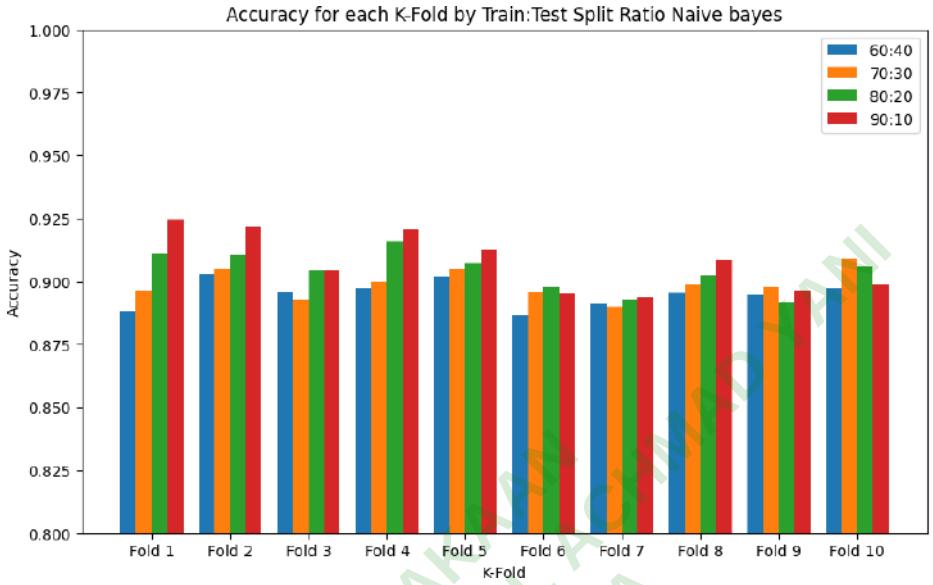
Gambar 4.18 Hasil Pelabelan

```
label
 1    5474
-1    1958
Name: count, dtype: int64
```

Gambar 4.19 Jumlah Label

4.5 K-FOLD VALIDATION

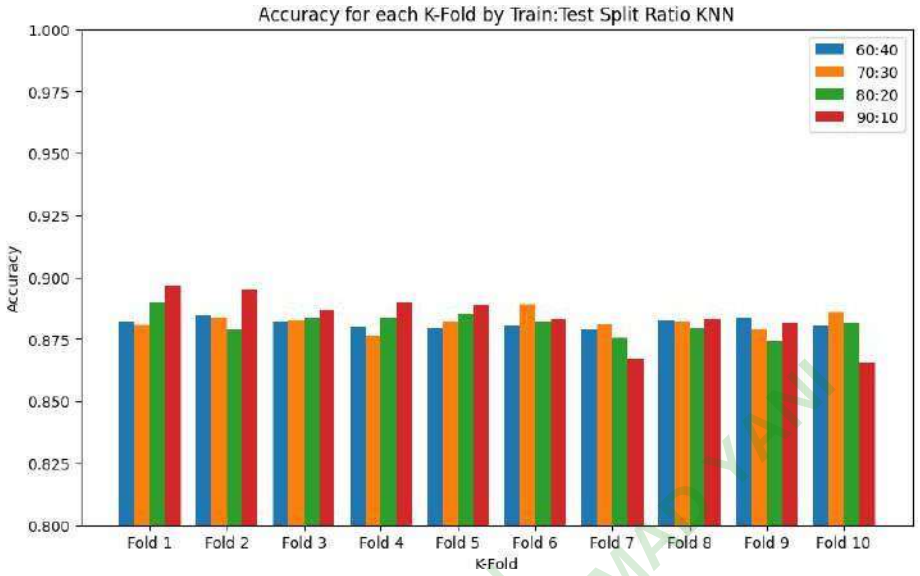
Dataset dibagi menjadi data latih dan data uji dengan menggunakan metode split data, dengan proporsi 90% untuk pelatihan dan 10% untuk pengujian untuk model NB dan KNN, melalui fungsi ‘train_test_split’ dai scikit-learn. Dengan menetapkan ‘test_size=0.1’ dan ‘random_state=30’, pembagian data dilakukan secara acak namun dapat direproduksi. Hasil pembagian ini menghasilkan sua subset, yaitu X_train dan Y_test untuk pengujian, masing-masing dengan ukuran yang sesuai untuk melatih model *maching learning* dan mengevaluasi kinerjanya. Hasil dari split data menggunakan k- fold model NB dapat dilihat pada gambar 4.20 dan keterangan bentuk tabel dapat dilihat pada gambar 4.21, rata-rata hasil split data k-fold model NB untuk perbandingan 90:10 paling tinggi mencapai 0.9077. Sedangkan untuk hasil dari split data menggunakan k- fold KNN dapat dilihat pada gambar 4.22. Pada gambar 4.3, rata-rata hasil split data k-fold model KNN untuk perbandingan 90:10 paling tinggi mencapai 0.8837.



Gambar 4.21 Split Data K-Fold Naive Bayes

	60:40	70:30	80:20	90:10
Fold 1	0.887991927346115	0.8964125560538116	0.9112306657700068	0.9247311827956989
Fold 2	0.9027917928018836	0.9049327354260089	0.9105581708137189	0.9220430107526881
Fold 3	0.8960645812310797	0.8928251121076233	0.9045057162071285	0.9045698924731183
Fold 4	0.8970736629667003	0.9	0.9159381304640215	0.9206989247311828
Fold 5	0.9021190716448032	0.9049327354260089	0.9071956960322798	0.9126344086021505
Fold 6	0.8863101244534141	0.8959641255605382	0.8977807666442502	0.8951612903225806
Fold 7	0.8913555331315169	0.8901345291479821	0.8930733019502354	0.8938172043010753
Fold 8	0.8953918600739993	0.899103139013453	0.902488231338265	0.9086021505376344
Fold 9	0.8947191389169189	0.8977578475336323	0.8917283120376597	0.896505376344086
Fold 10	0.8974100235452405	0.9089686098654709	0.9058507061197041	0.8991935483870968
Average	0.8951227716111673	0.8991031390134531	0.904034969737727	0.907795698924731

Gambar 4.20 Keterangan Split Data K-Fold Naive Bayes



Gambar 4.22 Split Data K-Fold KNN

	60:40	70:30	80:20	90:10
Fold 1	0.8819374369323916	0.8807174887892377	0.8897108271687962	0.896505376344086
Fold 2	0.8846283215607131	0.8838565022421525	0.878950907868191	0.8951612903225806
Fold 3	0.8819374369323916	0.8825112107623319	0.8836583725622058	0.8870967741935484
Fold 4	0.8799192734611504	0.8766816143497758	0.8836583725622058	0.8897849462365591
Fold 5	0.8795829128826101	0.8820627802690583	0.8850033624747814	0.8884408602150538
Fold 6	0.8805919946182308	0.889237668161435	0.8823133826496301	0.8830645161290323
Fold 7	0.87924655230407	0.8811659192825112	0.8755884330867518	0.8669354838709677
Fold 8	0.882610158089472	0.8820627802690583	0.8796234028244788	0.8830645161290323
Fold 9	0.8839556004036327	0.8789237668161435	0.8742434431741762	0.8817204301075269
Fold 10	0.8805919946182308	0.8860986547085202	0.8816408876933423	0.8655913978494624
Average	0.8815001681802894	0.8823318385650225	0.8814391392064559	0.883736559139785

Gambar 4.23 Keterangan Split Data K-Fold KNN

4.6 TF- IDF

TF-IDF (*Term Frequency – Inverse Document Frequency*) teknik yang digunakan untuk mengevaluasi kepentingan suatu dokumen dalam sebuah dokumen adalah dengan menggunakan Term Frequency (TF) yang mengukur seberapa sering kata tersebut muncul dalam dokumen. Tujuannya untuk menentukan relevensi atau kepentingan kata tersebut dalam konteks dokumen. Sementara itu, Inverse Document Frequency (IDF) mengukur seberapa penting suatu kata dengan mempertimbangkan seberapa umum atau jarang kata tersebut muncul dalam sebuah dokumen . Hasil dari perhitungan TF-IDF dapat dilihat pada gambar 4.24.

```
print(X_train_features)
✓ 0.0s
  (0, 4771)     0.1124655188752522
  (0, 2992)     0.11675308202267412
  (0, 2263)     0.10932669581562543
  (0, 3166)     0.1285578893413759
  (0, 4936)     0.16090724219457495
  (0, 984)      0.23731576494087575
  (0, 2101)     0.18115086018863663
  (0, 5304)     0.24196864542397326
  (0, 855)      0.2685279928121763
  (0, 5018)     0.2754291455937279
  (0, 5630)     0.26643320270711895
  (0, 960)      0.16696428173207
  (0, 2569)     0.23328525653624393
  (0, 5842)     0.20012476913415697
  (0, 4113)     0.14401581245429862
  (0, 5909)     0.1905125692272124
  (0, 4025)     0.12651458467298377
  (0, 5127)     0.19889259901465275
  (0, 5865)     0.2186307000790591
  (0, 4596)     0.21053342623671828
  (0, 314)      0.13930698389882035
  (0, 4253)     0.2848701508867388
  (0, 5676)     0.15031713044383135
  (0, 4646)     0.11475544890676923
  (0, 3918)     0.19521225711708354
...
  (6687, 1439)  0.17972185348406194
  (6687, 5353)  0.3674135961838261
  (6687, 498)   0.18357232331297357
  (6687, 4646)  0.3315188556301759
```

Gambar 4.24 Hasil Perhitungan TF-IDF

Pada gambar 4.24 menjelaskan pasangan (0, 4771) 0.1124655188752522 dalam representasi TF-IDF menunjukkan bahwa kata yang mungkin adalah " umum" memiliki nilai TF-IDF sebesar 0.112 dalam kalimat ini menunjukkan pentingnya kata tersebut dalam dokumen relative terhadap kumpulan dokumen lainnya, dalam konteks dokumen yang lebih besar. (0,2992) 0.11675308202267412 kata dengan indeks 2992 mungkin kata "siap" memiliki nilai TF-IDF sebesar 0.117, menyoroti keunikan kata tersebut dalam kalimat pertama. Nilai-nilai TF-IDF yang lebih tinggi menunjukkan bahwa kata-kata tersebut memiliki keunikan atau pentingnya dalam kalimat pertama dibandingkan dengan dokumen lain dalam koleksi dokumen tersebut.

4.7 NAÏVE BAYES

NB digunakan untuk memperhitungkan probabilitas setiap kata yang telah diberi bobot melalui TF-IDF dan mengidentifikasikan seberapa jauh sample data pelatihan tersebut memiliki sentimen positif. Hasil perhitungan NB dapat dilihat pada gambar 4.25.

```
Accuracy: 0.896505376344086
Precision: 0.8719417547490576
Recall (Sensitivity): 0.8418478260869565
```

Gambar 4.25 Hasil Prediksi Naive Bayes

Dalam menganalisis kinerja model NB pada dataset ini, yang terdiri dari 6688 data latih dan 744 data uji, *confusion matrix* memberikan gambaran yang jelas tentang seberapa baik model dapat mengklasifikasi data, termasuk jumlah prediksi yang benar dan salah untuk setiap kelas. Berdasarkan Confusion Matrix NB yang diberikan, terdapat 135 kasus TP (data positif yang diprediksi dengan benar), 49 kasus FN (data positif yang salah diprediksi sebagai negatif), 28 kasus FP (data negatif yang salah diprediksi sebagai positif), dan 532 kasus TN (data negatif yang diprediksi dengan benar). Dari data ini, tidak terjadi prediksi TN atau FN. Gambar. Hasil dari *confusion matrix* dapat di lihat pada gambar 4.26

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, predictions)
print(cm)
```
✓ 0.0s

```
[[135  49]
 [ 28 532]]
```

Gambar 4.26 Confusion Matrix Naive Bayes

Classification report memberikan detail yang lebih rinci tentang evaluasi matrix untuk setiap kelas, seperti *precision*, *recall*, dan *f1-score*, sementara *confusion matrix* memberikan informasi jumlah *True Positives*(TP), *True Negatives*(TN), *False positives*(FP), dan *False Negatives*(FN). Berdasarkan gambar 4.26 berhasil dalam memprediksi sentimen TN dengan baik. Untuk hasil *Classification report* dapat dilihat pada gambar 4.27.

```
from sklearn.metrics import classification_report
report = classification_report(Y_test, predictions)
print(report)
```
✓ 0.0s

	precision	recall	f1-score	support
-1	0.83	0.73	0.78	184
1	0.92	0.95	0.93	560
accuracy			0.90	744
macro avg	0.87	0.84	0.86	744
weighted avg	0.89	0.90	0.89	744

Gambar 4.27 Classification Report Naive Bayes

4.8 K-NEAREST NEIGHBORS

K-Nearest Neighbors digunakan untuk mengidentifikasikan sejauh mana k titik data terdekat dari titik data yang tidak dikenal. Hasil dari identifikasi ini dapat digunakan untuk menentukan kelas atau nilai dari titik data yang tidak dikenal tersebut, berdasarkan mayoritas kelas atau nilai dari k tetangga terdekatnya. Dengan kata lain, KNN memprediksi label atau nilai suatu titik data baru berdasarkan informasi dari tetangga yang paling mirip. Untuk menentukan nilai k gambar dapat dilihat pada gambar 4.28 dan untuk hasil perhitungan menggunakan KNN dapat dilihat pada gambar 4.29.

```python
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.neighbors import KNeighborsClassifier

# Definisikan parameter grid yang ingin dicari
param_grid = {'n_neighbors': list(range(1, 31))}  # Contoh, mencari dari 1 hingga 30 tetangga

# Initialize KNN classifier (nilai default akan diganti oleh GridSearchCV)
knn_classifier = KNeighborsClassifier()

# Initialize K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)  # Anda bisa mengatur n_splits sesuai kebutuhan

# Initialize GridSearchCV dengan KNN dan parameter grid
grid_search = GridSearchCV(estimator=knn_classifier, param_grid=param_grid, cv=kf, scoring='accuracy')

# latih mencari grid
grid_search.fit(X_train_features, Y_train)

# jumlah tetangga terbaik
best_k = grid_search.best_params_['n_neighbors']
print(f"Best number of neighbors: {best_k}")

# latih pengklasifikasia knn dengan jumlah tetangga terbaik
best_knn_classifier = KNeighborsClassifier(n_neighbors=best_k)
best_knn_classifier.fit(X_train_features, Y_train)

# buat prediksi pada set penguji dengan pengklasifikasi KNN terbaik
knn_predictions = best_knn_classifier.predict(X_test_features)
```

✓ 4m 58.5s

```
Best number of neighbors: 4
```

Gambar 4.28 Nilai k Terbaik

```
KNN Accuracy: 0.8575268817204301
KNN Precision: 0.8536957504520796
KNN Recall (Sensitivity): 0.742973602484472
```

Gambar 4.29 Hasil Prediksi KNN

Pada gambar 4.30 ini menghasilkan dan menampilkan *confusion matrix* secara visual untuk mengevaluasi kinerja model klasifikasi, mempermudah identifikasi kesalahan prediksi, dari total 744 data, model berhasil memprediksi dengan benar 120 data sebagai TP dan 536 data sebagai TN. Namun, terdapat 64 data yang seharusnya positif namun diprediksi sebagai FN, serta 24 data yang seharusnya negatif namun diprediksi sebagai FP, ini menunjukkan bahwa meskipun model memiliki kemampuan dalam mengidentifikasi kelas positif dan negatif, masih terdapat kesalahan dalam prediksi, terutama dalam mengklasifikasikan data yang seharusnya negatif. Performa model secara keseluruhan dapat dievaluasi lebih lanjut dengan menghitung matrik seperti akurasi, presisi, recall dan F1-score.

```
from sklearn.metrics import confusion_matrix
cm_knn = confusion_matrix(Y_test, knn_predictions)
print(cm_knn)
```
✓ 0.0s

```
[[120  64]
 [ 24 536]]
```

Gambar 4.30 Confusion Matrix KNN

Fungsi 'classification_report' dari Pustaka scikit-learn menghasilkan laporan klasifikasi yang merangkum kinerja model klasifikasi KNN, mencangkup *precision, recall, f1-score,* dan *support* untuk setiap kelas dalam dataset, serta menyajikan matrix evaluasi tambahan seperti *accuracy, macro avg,* dan *weighted avg.* Untuk gambar *classification report* dapat dilihat pada gambar 4.31.

```
from sklearn.metrics import classification_report
report = classification_report(Y_test, knn_predictions)
print(report)
```
✓ 0.0s

```
              precision    recall  f1-score   support

          -1       0.83      0.65      0.73       184
           1       0.89      0.96      0.92       560

    accuracy                           0.88       744
   macro avg       0.86      0.80      0.83       744
weighted avg       0.88      0.88      0.88       744
```

Gambar 4.31 Classification Report KNN

4.9 HASIL ANALISIS PERBANDINGAN

Berdasarkan hasil analisis, model NB menunjukkan performa yang lebih baik dibandingkan KNN dalam hal akurasi dan *f1-score*. *F1-score* hasil dari rata-rata antara *precision* dan *recall*, memberikan gambaran yang lebih lengkap tentang performa model. *F1-score* dihitung menggunakan rumus :

$$\text{f1} - \text{score} = 2X\frac{precision\ X\ \ recall}{precision + recall} \quad (3)$$

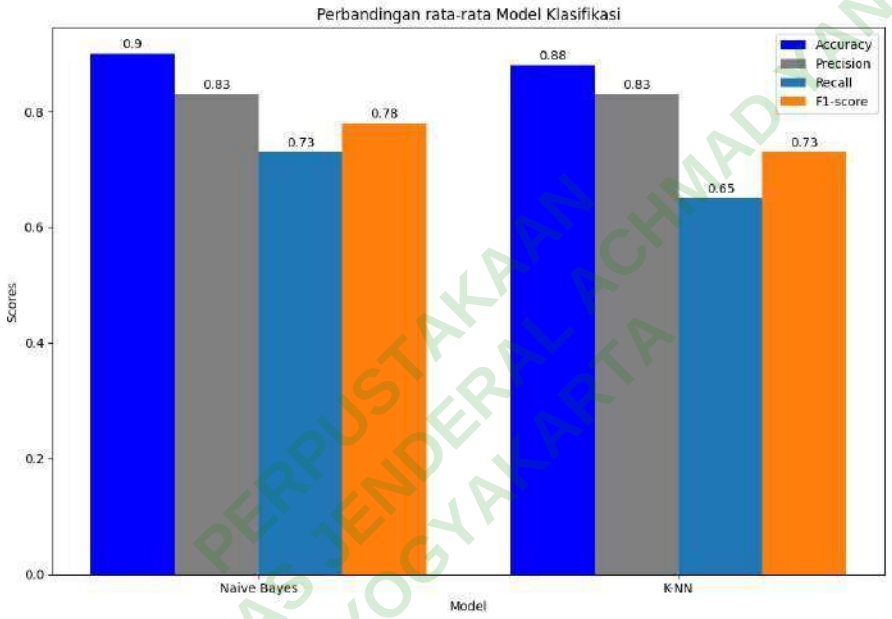
Naïve Bayes

$$\text{f1} - \text{score} = 2X\frac{0.83\ X\ \ 0.73}{0.83+0.73} = 0.78$$

KNN

$$\text{f1} - \text{score} = 2X\frac{0.83\ X\ \ 0.65}{0.83+0.65} = 0.73$$

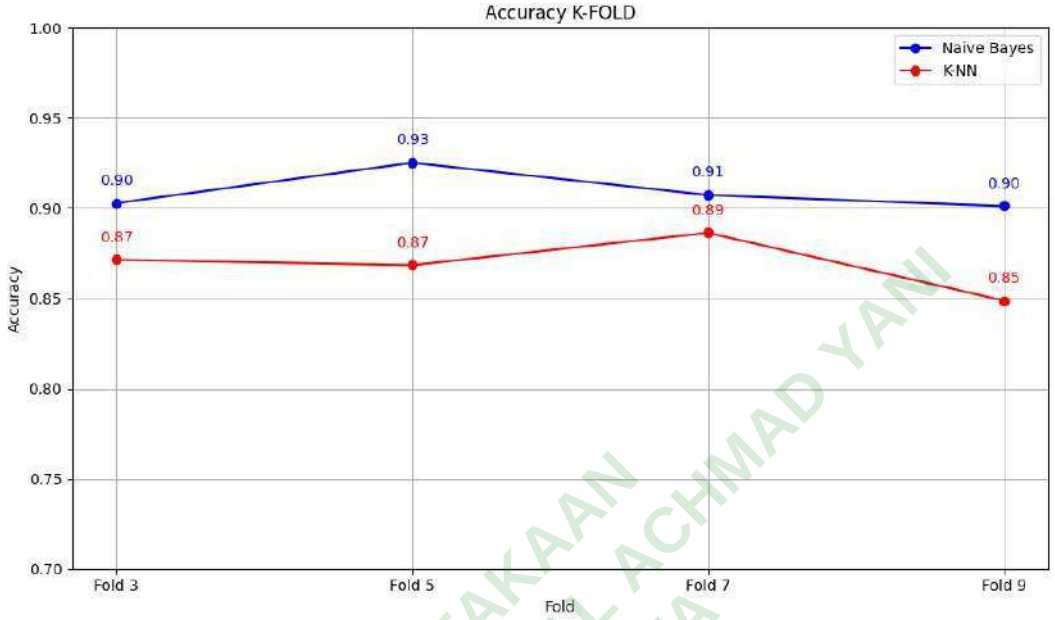
NB menghasilkan nilai akurasi sebesar 90% dan *f1-score* sebesar 78%, sedangkan KNN hanya mencapai nilai akurasi sebesar 88% dan *f1-score* sebesar 73%. Namun dalam *recall*, model NB juga lebih baik dengan nilai *recall* sebesar 73% dibandingkan model KNN yang menghasilkan nilai recall sebesar 65%, ini menunjukkan bahwa metode NB lebih unggul secara keseluruhan, termasuk dalam mendeteksi positif. Untuk hasil perbandingan model klasifikasi NB dan KNN dapat dilihat pada gambar 4.32.



Gambar 4.32 Perbandingan Klasifikasi NBdan KNN

Dengan menggunakan metode *K-fold cross validation*, membandingkan kedua model klasifikasi antara NB dan KNN dengan memvariasikan nilai k pada KNN. Hasilnya menunjukkan bahwa NB, rata-rata akurasi corss validation stabil di sekitar 0.90 hingga 0.93, dengan nilai tertinggi pada k-fold 5 dan nilai terendah k-fold 9, sementara KNN menemukan variasi dalam kinerja bergantung pada nilai k yang digunakan , melihat bahwa akurasi KNN antara 0.85 hingga 0.89, dengan nilai tertinggi di k-fold 7 dan terendah di k-fold 9. Dengan demikian dapat menyimpulkan bahwa dalam grafik ini, NB menunjukkan konsistensi dalam kinerjanya, sementara KNN membutuhkan penyesuaian yang cermat terhadap nilai

k untuk mencapai hasil yang optimal. Hasil dari perhitungan klasifikasi menggunakan *k-fold cross validation* dapat dilihat pada gambar 4.33.



Gambar 4.33 Perbandingan Klasifikasi K-Fold Corss Validation

4.10 HASIL ANALISIS SENTIMEN NAÏVE BAYES DAN KNN

Dalam visualisasi word cloud, hasilnya menggambarkan frekuensi kata-kata yang sering muncul dalam ulasan pengguna aplikasi Halodoc. Ukuran kata dalam *word cloud* semakin besar jika topik tersebut sering dibahas oleh pengguna yang memberikan ulasan. Hasil dari *word cloud* sentimen positif dapat dilihat pada gambar 4.34 dan gambar 4.35 hasil *word cloud* sentimen negatif.

Gambar 4.34 Sentimen Positif

Gambar 4.35 Sentimen Negatif

Berdasarkan *word cloud*, peneliti dapat menggunakan informasi ini untuk mengeksplorasi kata-kata yang mencerminkan sentimen positif dan negative, serta frekuensi kemunculannya. Hal ini membantu kepuasan pelanggan terhadap produk atau layanan yang disediakan.

4.11 HASIL PERBANDINGAN PENELITIAN TERDAHULU DAN TERBARU

Tabel 3 Perbandingan Hasil Penelitian Terdahulu dan Terbaru

Item	Tri Dewi Septiani	Kartarina	Ayu Wulansari
- Objek	Mobile Bangking Jenius	Memprediksi Kelulusan Mahasiswa	Aplikasi Halodoc
- Metode	Naïve Bayes Classifer dan K-Nearest Neighbors	Naïve Bayes Classifer dan K-Nearest Neighbors	Naïve Bayes Classifer dan K-Nearest Neighbors
- Akurasi	Nilai akurasi menggunakan metode Naïve Bayes menghasilkan nilai sebesar 83,06% sedangkan menggunakan metode dan K-Nearest Neighbors menghasilkan nilai sebesar 84,06%.	*K-Nearest Neighbor* memiliki kinerja lebih unggul dalam menganalisis sentimen dan menghasilkan nilai akurasi 83,06% menggunakan metode NB sedangkan menggunakan metode KNN menghasilkan nilai akurasi sebesar 84,06%.	Akurasi model Naïve Bayes mencapai tingkat akurasi sebesar 90%, sedangkan K-Nearest Neighbors mencapai 88%.