BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini melakukan analisis sentimen dan pemodelan topik terhadap data umpan balik pengguna Bima *Mobile* pada *Play Store*. Jumlah data umpan balik yang berhasil terambil sebanyak 4294 data. Setelah melalui tahapan *preprocessing* yang terdiri dari *data cleaning, case folding, tokenizing, normalization, stopword removal,* dan *stemming* dihasilkan data bersih sebanyak 4242 data. Setelah diubah ke vektor numerik, sentimen diterapkan menggunakan model SVM, dengan akurasi sebesar 95%. Hasil analisis sentimen menunjukkan bahwa sentimen negatif lebih mendominasi dengan jumlah 2374 umpan balik, sedangkan 1868 umpan balik memberikan sentimen positif. Selanjutnya, pemodelan topik diterapkan pada masing-masing kategori sentimen untuk mengidentifikasi topik utama dari umpan balik yang diberikan. Pada kategori positif diperoleh jumlah topik optimal sebanyak 4 topik, sementara pada kategori negatif diperoleh jumlah topik optimal sebanyak 7 topik. Pada kategori negatif pengguna mengeluhkan terkait kesulitan akses, gangguan teknis dan *error* saat penggunaan, masalah registrasi dan verifikasi nomor, penambahan fitur, munculnya bug setelah update, kendala *login*, dan performa aplikasi yang lambat. Sedangkan pada kategori positif menunjukkan bahwa aplikasi Bima *Mobile* telah berhasil membangun kepercayaan dan kepuasan di beberapa kalangan penggunanya, baik dari sisi kemudahan, kebermanfaatan untuk transaksi, maupun peningkatan berkelanjutan.

4.2 HASIL PENGAMBILAN DATA *(SCRAPING)*

Pengambilan data dilakukan untuk memperoleh data umpan balik dari pengguna aplikasi Bima *Mobile*. Tahap ini memerlukan bantuan *Google Colab* yang telah terinstall *Google-Play-Scraper* yang merupakan alat untuk melakukan *scraping* data dari *Play Store*. Scraping dilakukan dengan menginputkan id dari aplikasi Bima *Mobile* dan mengambil data seluruh umpan balik dari urutan yang terbaru. Data yang terambil diantaranya *review id, username, user image, content,*

score, thumbsup count, review created version, at, reply content, replied at, dan *app version*. Pengambilan data dilakukan pada tanggal 7 Maret 2025 dengan jumlah 4294 data yang kemudian disimpan dalam format csv. Gambar 4.1 merupakan hasil pengambilan data yang sudah dikonversi ke dalam bentuk *DataFrame*.

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt	appVersion
0	da51cc6f-1b6b-4ce7-a602-75475b08b2af	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Tambahin fitur ewallet, topup gopay, ovo, dana...	4	0	2.2.1	########	NaN	NaN	2.2.1
1	af055ff4-81ef-4650-9d39-c937ad944e46	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Mantap.. Mudah digunakan..	5	0	2.2.1	########	NaN	NaN	2.2.1
2	ce1fa128-0996-4417-b448-960a95f35940	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Tambahkan transaksi ke E-wallet, auto bintang 5	5	0	2.2.1	########	NaN	NaN	2.2.1
3	fa8e334b-65ac-44d8-bd3a-14d6f7f78329	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Sering error,,	2	0	2.2.1	########	NaN	NaN	2.2.1
4	36b587bc-17b4-494e-9b7f-eb04ada50dd1	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Baik	5	0	2.2.1	########	NaN	NaN	2.2.1
...	...	...	...	...	...	...	...	...	...	...	...
4278	6afd7fd3-b818-41bf-addd-14b524c3109d	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Aplikasi keren, kekinian, cocok buat kawula mu...	5	7	1.0.0	########	Bank Jateng akan terus melakukan pengembangan ...	########	1.0.0
4279	092b91c6-8dfb-41ea-a326-7589671479fc	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	аЎ'*	5	0	1.0.0	########	Bank Jateng akan terus melakukan pengembangan ...	########	1.0.0
4280	80cc4dfd-62fe-44b9-8652-fef8c9088494	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Pada akhirnya kemudahan di depan mata #BIMASEM...	5	1	1.0.0	########	Bank Jateng akan terus melakukan pengembangan ...	########	1.0.0
4281	2381d73e-3e1b-424f-97de-e4b4d0ad6bd3	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Sangat mudah dan terjangkau menggunakan Bima M...	5	1	1.0.0	########	Bank Jateng akan terus melakukan pengembangan ...	########	1.0.0
4282	cb86fe82-a9c2-47c6-9740-317dd69e4a69	Pengguna Google	https://play-lh.googleusercontent.com/EGemoI2N...	Keren banget aplikasinya i love Bank Jateng аЎ'_	5	1	1.0.0	########	Bank Jateng akan terus melakukan pengembangan ...	########	1.0.0

4283 rows × 11 columns

Gambar 4.1 Hasil Pengambilan Data

Gambar 4.1 menunjukkan bahwa data yang diambil masih banyak komponen yang perlu dihilangkan karena tidak semua data digunakan dalam proses pelatihan model. Data tersebut masih memiliki banyak karakter yang tidak bisa diartikan atau diterjemahkan ke dalam makna yang relevan, sehingga perlu dilakukan tahapan *preprocessing* agar menghasilkan data yang sesuai.

4.3 *PREPROCESSING* DATA

Preprocessing data adalah tahapan pengolahan data untuk mempersiapkan data yang siap digunakan untuk analisis atau pelatihan model. Sebelum masuk ke dalam tahapan *preprocessing* perlu dilakukan pengecekan data, sehingga jika terdapat data duplikat maupun data kosong perlu dilakukan penghapusan terlebih dahulu. Data yang digunakan dalam penelitian ini hanya data umpan balik yang ada dalam kolom ‘content’. Selanjutnya, *preprocessing* data terdiri dari beberapa tahapan, diantaranya:

1. Data *Cleaning*

Tahapan yang pertama yaitu proses membersihkan data dari karakter yang tidak sesuai dengan menggunakan *library regular expression (re)*. Proses dalam data *cleaning* ditunjukkan pada Kode Program 4.1.

Kode Program 4.1 Data *Cleaning*

```
1. def cleanText(text):
2.     text = re.sub(r'@[A-Za-z0-9]+', '', text)
3.     text = re.sub(r'#[A-Za-z0-9]+', '', text)
4.     text = re.sub(r"http\S+", '', text)
5.     text = re.sub(r'[0-9]+', '', text)
6.     text = re.sub(r'[^\w\s]', '', text)
7.     text = text.replace('\n', ' ')
8.     text = re.sub(r'[^\x00-\x7F]+', '', text)
9.     text = text.strip(' ')
```

Berikut tahapan-tahapan dari fungsi ‘cleanText’:

a. re.sub(r'@[A-Za-z0-9]+', '', text) : Menghapus semua *mention*, misalnya @username.
b. re.sub(r'#[A-Za-z0-9]+', '', text) : Menghapus hastag atau tagar.
c. re.sub(r"http\S+", '', text) : Menghapus link atau url yang diawali http.
d. re.sub(r'[0-9]+', '', text) : Menghapus angka yang ada dalam teks.
e. re.sub(r'[^\w\s]', '', text) : Menghapus semua tanda baca atau disebut *punctuation removal*.
f. re.sub(r'[^\x00-\x7F]+', '', text) : Menghapus semua karakter non-ASCII.
g. text.replace('\n', ' ') : Mengganti *newline* dengan spasi biasa.
h. text.strip(' ') : Menghapus spasi yang ada di awal dan akhir teks.

Setelah melakukan tahapan tersebut teks yang sudah bersih akan dikembalikan dengan perintah ‘return’. Hasil data *cleaning* ditunjukkan pada Gambar 4.2.

content	result_clean
Tambahin fitur ewallet, topup gopay, ovo, dana...	Tambahin fitur ewallet topup gopay ovo dana sh...
Mantap.. Mudah digunakan..	Mantap Mudah digunakan
Tambahkan transaksi ke E-wallet, auto bintang ...	Tambahkan transaksi ke Ewallet auto bintang lima
Sering error,,,	Sering error
Baik	Baik
Membantu transaksiku	Membantu transaksiku
Login mbanking susahnya.. cuma sampe logo bima...	Login mbanking susahnya cuma sampe logo bima saja
Bagus	Bagus
Udh sehari aplikasi ga bisa dibuka ya min, pad...	Udh sehari aplikasi ga bisa dibuka ya min pada...
Data saya bocor maaf saya kasih bintang satu d...	Data saya bocor maaf saya kasih bintang satu d...

Gambar 4.2 Hasil Data *Cleaning*

2. *Case Folding*

Tahapan kedua yaitu *case folding* yang merupakan proses mengubah huruf besar atau kapital pada teks menjadi huruf kecil. Proses *case folding* ditunjukkan pada Kode Program 4.2.

Kode Program 4.2 *Case Folding*

```
1. def casefoldingText(text):
2.   text = text.lower()
3.   return text
```

Proses mengubah huruf kapital menjadi huruf kecil pada fungsi 'casefoldingText' dilakukan dengan menggunakan perintah lower(). Hasil case folding ditunjukkan pada Gambar 4.3.

result_clean	result_casefolding
Tambahin fitur ewallet topup gopay ovo dana sh...	tambahin fitur ewallet topup gopay ovo dana sh...
Mantap Mudah digunakan	mantap mudah digunakan
Tambahkan transaksi ke Ewallet auto bintang lima	tambahkan transaksi ke ewallet auto bintang lima
Sering error	sering error
Baik	baik
Membantu transaksiku	membantu transaksiku
Login mbanking susahnya cuma sampe logo bima saja	login mbanking susahnya cuma sampe logo bima saja
Bagus	bagus
Udh sehari aplikasi ga bisa dibuka ya min pada...	udh sehari aplikasi ga bisa dibuka ya min pada...
Data saya bocor maaf saya kasih bintang satu d...	data saya bocor maaf saya kasih bintang satu d...

Gambar 4.3 Hasil *Case Folding*

3. *Tokenizing*

Tahapan selanjutnya yaitu *tokenizing* yang merupakan proses memecah teks menjadi kata-kata (token). Proses *tokenizing* ditunjukkan dalam Kode Program 4.3.

Kode Program 4.3 *Tokenizing*

```
1. def tokenizingText(text):
2.     text = word_tokenize(text)
3.     return text
```

Proses tokenisasi dilakukan dengan menggunakan fungsi 'word_tokenize' dari NLTK. Teks dipecah menjadi daftar kata (token) berdasarkan spasi dan tanda baca. Hasil *tokenizing* ditunjukkan pada Gambar 4.4.

result_casefolding	result_tokenizing
tambahin fitur ewallet topup gopay ovo dana sh...	[tambahin, fitur, ewallet, topup, gopay, ovo, ...
mantap mudah digunakan	[mantap, mudah, digunakan]
tambahkan transaksi ke ewallet auto bintang lima	[tambahkan, transaksi, ke, ewallet, auto, bint...
sering error	[sering, error]
baik	[baik]
membantu transaksiku	[membantu, transaksiku]
login mbanking susahnya cuma sampe logo bima saja	[login, mbanking, susahnya, cuma, sampe, logo,...
bagus	[bagus]
udh sehari aplikasi ga bisa dibuka ya min pada...	[udh, sehari, aplikasi, ga, bisa, dibuka, ya, ...
data saya bocor maaf saya kasih bintang satu d...	[data, saya, bocor, maaf, saya, kasih, bintang...

Gambar 4.4 Hasil *Tokenizing*

4. *Normalization*

Tahapan selanjutnya yaitu *normalization* yang bertujuan untuk mengubah kata tidak baku ke bentuk yang standar supaya mengurangi variasi kata yang memiliki arti mirip atau sama. Kata tidak baku yang biasanya digunakan dalam percakapan sehari-hari seperti bahasa gaul atau singkatan disebut dengan *slangword*. Daftar *slangword* yang digunakan pada penelitian ini berasal dari github sebanyak 15081 kata dan *slangword* tambahan sebanyak 42 kata. Contoh daftar *slangword* yang digunakan pada penelitian ini dapat dilihat pada Gambar 4.5.

slang	formal	slang	formal
gmn	bagaimana	therchakitty	tersakiti
gilakkk	gila	mauuuuu	mau
gtu2	begitu-gitu	kaliii	kali
sehh	sih	bangat	banget
kakk	kak	bkin	bikin
ajah	saja	dongg	dong
cpet	cepat	yokkk	yuk
bnget	banget	gini	begini
aq	aku	yu	yuk

Gambar 4.5 Contoh Daftar *Slangword*

Sumber: (https://github.com/luthfilkhairi/kamus/blob/master/Slangword-indonesian.xlsm)

Pada proses *normalization* ini perlu menggabungkan daftar *slangword* dari github dan *slangword* tambahan untuk kata yang tidak ada di daftar tersebut. Selanjutnya, yaitu mengganti kata-kata slang dalam sebuah teks umpan balik dengan bentuk bakunya berdasarkan daftar *slangword* yang telah dibuat. Proses *normalization* teks umpan balik ditunjukkan pada Kode Program 4.4.

Kode Program 4.4 *Normalization*

```
1. def fix_slangwords(text):
2.     fixed_words = [slangwords_dict[word.lower()] if
       word.lower() in slangwords_dict else word for word
       in text]
3.     return fixed_words
```

Fungsi ‘fix_slangwords’ digunakan untuk mengoreksi kata slang menggunakan kamus ‘slangword_dict’. Fungsi akan memproses setiap kata dalam ‘text’, apakah kata tersebut merupakan slang, jika slang makan akan diganti dengan kata baku yang benar, jika bukan maka kata asli akan tetap digunakan. Hasil dari proses *normalization* ini adalah list baru ‘fixed_word’ yang berisi kata-kata yang sudah dikoreksi, seperti yang ditunjukkan pada Gambar 4.6.

result_tokenizing	result_slangwords
[tambahin, fitur, ewallet, topup, gopay, ovo, ...	[tambahkan, fitur, ewallet, topup, gopay, ovo,...
[mantap, mudah, digunakan]	[mantap, mudah, digunakan]
[tambahkan, transaksi, ke, ewallet, auto, bint...	[tambahkan, transaksi, ke, ewallet, auto, bint...
[sering, error]	[sering, error]
[baik]	[baik]
[membantu, transaksiku]	[membantu, transaksiku]
[login, mbanking, susahnya, cuma, sampe, logo,...	[login, mbanking, susahnya, cuma, sampai, logo...
[bagus]	[bagus]
[udh, sehari, aplikasi, ga, bisa, dibuka, ya, ...	[sudah, sehari, aplikasi, tidak, bisa, dibuka,...
[data, saya, bocor, maaf, saya, kasih, bintang...	[data, saya, bocor, maaf, saya, kasih, bintang...

Gambar 4.6 Hasil *Normalization*

5. *Stopword Removal*

Tahapan *preprocessing* selanjutnya yaitu *stopword removal*, yang merupakan langkah untuk menghilangkan kata-kata umum atau tidak penting

seperti “dan”, “yang”, “di”, dan sebagainya yang ada dalam *library Sastrawi*. Daftar kata yang termasuk dalam *stopword* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar *Stopword*

Daftar *Stopword*
['yang', 'untuk', 'pada', 'ke', 'para', 'namun', 'menurut', 'antara', 'dia', 'dua', 'ia', 'seperti', 'jika', 'sehingga', 'kembali', 'dan', 'tidak', 'ini', 'karena', 'kepada', 'oleh', 'saat', 'harus', 'sementara', 'setelah', 'belum', 'kami', 'sekitar', 'bagi', 'serta', 'di', 'dari', 'telah', 'sebagai', 'masih', 'hal', 'ketika', 'adalah', 'itu', 'dalam', 'bisa', 'bahwa', 'atau', 'hanya', 'kita', 'dengan', 'akan', 'juga', 'ada', 'mereka', 'sudah', 'saya', 'terhadap', 'secara', 'agar', 'lain', 'anda', 'begitu', 'mengapa', 'kenapa', 'yaitu', 'yakni', 'daripada', 'itulah', 'lagi', 'maka', 'tentang', 'demi', 'dimana', 'kemana', 'pula', 'sambil', 'sebelum', 'sesudah', 'supaya', 'guna', 'kah', 'pun', 'sampai', 'sedangkan', 'selagi', 'sementara', 'tetapi', 'apakah', 'kecuali', 'sebab', 'selain', 'seolah', 'seraya', 'seterusnya', 'tanpa', 'agak', 'boleh', 'dapat', 'dsb', 'dst', 'dll', 'dahulu', 'dulunya', 'anu', 'demikian', 'tapi', 'ingin', 'juga', 'nggak', 'mari', 'nanti', 'melainkan', 'oh', 'ok', 'seharusnya', 'sebetulnya', 'setiap', 'setidaknya', 'sesuatu', 'pasti', 'saja', 'toh', 'ya', 'walau', 'tolong', 'tentu', 'amat', 'apalagi', 'bagaimanapun']

Proses *stopword removal* ditunjukkan pada Kode Program 4.5.

Kode Program 4.5 *Stopword Removal*

```
1. def stopwordText(text):
2.     factory = StopWordRemoverFactory()
3.     stopwords = factory.get_stop_words()
4.     keep_words = {'tidak', 'belum', 'nggak'}
5.     stopwords = [word for word in stopwords if word not in
       keep_words]
6.     filtered = [word for word in text if word not in
       stopwords]
7.     return filtered
```

Dalam daftar *stopword* yang ditunjukkan pada Tabel 4.1, terdapat beberapa kata yang perlu dipertahankan agar tidak terhapus saat proses *stopword removal*, dikarenakan kata tersebut dinilai penting sehingga berpengaruh dalam analisis sentimen. Daftar kata yang tidak boleh dihapus disimpan dalam variabel ‘keep_words’. Berikut tahapan-tahapan dalam fungsi ‘stopwordText’:

a. factory = StopWordRemoverFactory() : Membuat objek ‘factory’ dari Sastrawi untuk mengambil daftar kata *stopword* bahasa Indonesia.

b. stopwords = factory.get_stop_words() : Mengambil daftar stopword default dari *library Sastrawi* ke dalam variabel ‘stopwords’.
c. Membuat set ‘keep_words’ yang berisi daftar kata penting yang tidak boleh dihapus, meskipun terdapat di dalam daftar *stopword*.
d. stopwords = [word for word in stopwords if word not in keep_words] : Membuat ulang list ‘stopwords’, hanya berisi kata-kata *stopword* yang tidak masuk ‘keep_words’.
e. filtered = [word for word in text if word not in stopwords] : Membuat list baru ‘filtered’, isinya hanya kata-kata dari ‘text’ yang bukan *stopword*.
f. Selanjutnya list kata-kata yang sudah difilter akan dikembalikan dengan perintah ‘return’.

Hasil *stopword removal* ditunjukkan pada Gambar 4.7.

result_slangwords	result_stopword
[tambahkan, fitur, ewallet, topup, gopay, ovo,...	[tambahkan, fitur, ewallet, topup, gopay, ovo,...
[mantap, mudah, digunakan]	[mantap, mudah, digunakan]
[tambahkan, transaksi, ke, ewallet, auto, bint...	[tambahkan, transaksi, ewallet, auto, bintang,...
[sering, error]	[sering, error]
[baik]	[baik]
[membantu, transaksiku]	[membantu, transaksiku]
[login, mbanking, susahnya, cuma, sampai, logo...	[login, mbanking, susahnya, cuma, logo, bima]
[bagus]	[bagus]
[sudah, sehari, aplikasi, tidak, bisa, dibuka,...	[sehari, aplikasi, tidak, bisa, dibuka, min, p...
[data, saya, bocor, maaf, saya, kasih, bintang...	[data, bocor, maaf, kasih, bintang, satu, dulu...

Gambar 4.7 Hasil *Stopword Removal*

6. *Stemming*

Tahapan *preprocessing* yang terakhir yaitu melakukan *stemming*. *Stemming* dilakukan untuk mengubah kata menjadi bentuk dasarnya. Proses *stemming* ditunjukkan pada Kode Program 4.6.

Kode Program 4.6 *Stemming*

```
1. def stemmingText(text):
2.     factory = StemmerFactory()
3.     stemmer = factory.create_stemmer()
4.     exception_words = {"setuju","gini","tujuan",
       "pemasukan","pengeluaran","dituju","perbaiki"}
5.     stem_correction = {"layan":"layanan","lingkung":
       "lingkungan", "pasuk":"masuk","muas":"puas","moga":
       "semoga","kece":"cek","ecek":"cek","langgan":
       "pelanggan","laku":"lakukan","alam":"pengalaman",
       "terang":"keterangan","kait":"terkait","ira":"kira",
       "alas":"alasan","lot":"lemot","dar": "sekedar", "asa":
       "rasa", "bagai": "sebagai"}
6.     stemmed_words = []
7.     for word in text:
8.         if word in exception_words:
9.             stemmed_words.append(word)
10.           else:
11.               stemmed = stemmer.stem(word)
12.               corrected = stem_correction.get(stemmed,
                  stemmed)
13.               stemmed_words.append(corrected)
14.    stemmed_text = ' '.join(stemmed_words
15.    return stemmed_text
```

Hasil dari *stemming* belum tentu semuanya tepat, sehingga diperlukan kamus koreksi manual yang disimpan dalam variabel ‘stem_correction’ supaya hasil akhir teks tetap menghasilkan kata yang bermakna sesuai dengan aslinya. Kamus manual tersebut berisi hasil *stemming* dan kata perbaikannya. Berikut tahapan-tahapan dalam fungsi ‘stemming Text’:

a. factory = StemmerFactory() : Membuat objek StemmerFactory dari *library Sastrawi*.
b. stemmer = factory.create_stemmer() : Membuat objek *stemmer* yang bisa digunakan untuk mengubah kata menjadi bentuk dasarnya.
c. Membuat set ‘exception_words’ yang berisi kata yang tidak perlu dilakukan *stemming*.

d. Membuat kamus koreksi manual ‘stem_correction’ untuk memperbaiki hasil *stemming* yang kurang tepat.

e. stemmed_words = [] : Membuat list kosong untuk menyimpan kata-kata setelah proses stemming.

f. for word in text:

```
if word in exception_words:
    stemmed_words.append(word)
else:
    stemmed = stemmer.stem(word)
    corrected = stem_correction.get(stemmed, stemmed)
    stemmed_words.append(corrected)
```

Melakukan *looping*, jika kata tersebut terdapat di ‘exception_words’, maka langsung ditambahkan ke ‘stemmed_words’ tanpa diubah. Jika tidak, lakukan stemming ‘stemmer.stem(word)’, kemudian cek apakah hasilnya perlu dikoreksi melalui ‘stem_correction’. Selanjutnya tambahkan kata hasil koreksi ke ‘stemmed_words’.

g. stemmed_text = ' '.join(stemmed_words) : Semua kata yang terdapat di list ‘stemmed_words’ digabung menjadi satu string dengan perintah join().

h. Teks yang telah di *stemming*, dikembalikan dengan perintah ‘return’.

Hasil *stemming* teks umpan balik ditunjukkan pada Gambar 4.8.

result_stopword	result_stemming
[tambahkan, fitur, ewallet, topup, gopay, ovo,...	tambah fitur ewallet topup gopay ovo dana shop...
[mantap, mudah, digunakan]	mantap mudah guna
[tambahkan, transaksi, ewallet, auto, bintang,...	tambah transaksi ewallet auto bintang lima
[sering, error]	sering error
[baik]	baik
[membantu, transaksiku]	bantu transaksi
[login, mbanking, susahnya, cuma, logo, bima]	login mbanking susah cuma logo bima
[bagus]	bagus
[sehari, aplikasi, tidak, bisa, dibuka, min, p...	hari aplikasi tidak bisa buka min padahal sist...
[data, bocor, maaf, kasih, bintang, satu, dulu...	data bocor maaf kasih bintang satu dulu tingka...

Gambar 4.8 Hasil *Stemming*

Setelah dilakukan berbagai tahapan dalam *preprocessing*, perlu dicek kembali terkait hasil akhir teks umpan balik yang telah di-*preprocessing* untuk mengetahui ada tidaknya data kosong. Proses pengecekan kembali data setelah di preprocessing ditunjukkan pada kode Program 4.7.

Kode Program 4.7 *Re-check* Hasil *Preprocessing*

```
1. empty_column = data_fix.eq('').sum()
2. print(f"Jumlah data kosong: \n{empty_column}")
```

Proses pengecekan ini menggunakan perintah eq (' ') untuk memeriksa setiap kolom di 'data_fix' apakah sama dengan string kosong (' '). Selanjutnya, perintah sum() digunakan untuk mendapatkan jumlah data kosong per kolom. Hasil *re-check preprocessing* ditunjukkan pada Gambar 4.9.

```
Jumlah data kosong:
content             0
result_clean        0
result_casefolding  0
result_tokenizing   0
result_slangwords   0
result_stopword     0
result_stemming    41
dtype: int64
```

Gambar 4.9 Hasil *Re-check Preprocessing*

Dari Gambar 4.9 dapat dilihat bahwa pada kolom 'result_stemming' terdapat 41 data kosong, yang mana data kosong tersebut perlu dihapus karena tidak dapat digunakan untuk proses selanjutnya. Proses penghapusan data kosong setelah *preprocessing* ditunjukkan pada Kode Program 4.8.

Kode Program 4.8 Hapus Data Kosong Setelah *Preprocessing*

```
1. data_fix = data_fix[~data_fix.eq('').any(axis=1)]
2. data_fix
```

Hasil akhir data setelah dilakukan *preprocessing* dan penghapusan data kosong ditunjukkan pada Gambar 4.10.

	content	result_clean	result_casefolding	result_tokenizing	result_slangwords	result_stopword	result_stemming
0	Tambahin fitur ewallet, topup gopay, ovo, dana...	Tambahin fitur ewallet topup gopay ovo dana sh...	tambahin fitur ewallet topup gopay ovo dana sh...	[tambahin, fitur, ewallet, topup, gopay, ovo, ...	[tambahkan, fitur, ewallet, topup, gopay, ovo,...	[tambahkan, fitur, ewallet, topup, gopay, ovo,...	tambah fitur ewallet topup gopay ovo dana shop...
1	Mantap.. Mudah digunakan..	Mantap Mudah digunakan	mantap mudah digunakan	[mantap, mudah, digunakan]	[mantap, mudah, digunakan]	[mantap, mudah, digunakan]	mantap mudah guna
2	Tambahkan transaksi ke E-wallet, auto bintang 5	Tambahkan transaksi ke Ewallet auto bintang	tambahkan transaksi ke ewallet auto bintang	[tambahkan, transaksi, ke, ewallet, auto, bint...	[tambahkan, transaksi, ke, ewallet, auto, bint...	[tambahkan, transaksi, ewallet, auto, bintang]	tambah transaksi ewallet auto bintang
3	Sering error,,,	Sering error	sering error	[sering, error]	[sering, error]	[sering, error]	sering error
4	Baik	Baik	baik	[baik]	[baik]	[baik]	baik
...	...	...	...	...	...	...	...
4277	Tampilannya mudah digunakan,, ditambah fitur2n...	Tampilannya mudah digunakan ditambah fiturnya ya	tampilannya mudah digunakan ditambah fiturnya ya	[tampilannya, mudah, digunakan, ditambah, fitu...	[tampilannya, mudah, digunakan, ditambah, fitu...	[tampilannya, mudah, digunakan, ditambah, fitu...	tampil mudah guna tambah fiturnya
4278	Aplikasi keren, kekinian, cocok buat kawula mu...	Aplikasi keren kekinian cocok buat kawula muda...	aplikasi keren kekinian cocok buat kawula muda...	[aplikasi, keren, kekinian, cocok, buat, kawul...	[aplikasi, keren, kekinian, cocok, buat, kawul...	[aplikasi, keren, kekinian, cocok, buat, kawul...	aplikasi keren kini cocok buat kawula muda bis...
4280	Pada akhirnya kemudahan di depan mata #BIMASEM...	Pada akhirnya kemudahan di depan mata	pada akhirnya kemudahan di depan mata	[pada, akhirnya, kemudahan, di, depan, mata]	[pada, akhirnya, kemudahan, di, depan, mata]	[akhirnya, kemudahan, depan, mata]	akhir mudah depan mata
4281	Sangat mudah dan terjangkau menggunakan Bima M...	Sangat mudah dan terjangkau menggunakan Bima M...	sangat mudah dan terjangkau menggunakan bima m...	[sangat, mudah, dan, terjangkau, menggunakan, ...	[sangat, mudah, dan, terjangkau, menggunakan, ...	[sangat, mudah, terjangkau, menggunakan, bima,...	sangat mudah jangkau guna bima mobile
4282	Keren banget aplikasinya i love Bank Jateng ðŸ˜	Keren banget aplikasinya i love Bank Jateng	keren banget aplikasinya i love bank jateng	[keren, banget, aplikasinya, i, love, bank, ja...	[keren, banget, aplikasinya, i, love, bank, ja...	[keren, banget, aplikasinya, i, love, bank, ja...	keren banget aplikasi i love bank jateng

4242 rows × 7 columns

Gambar 4.10 Hasil Akhir *Preprocessing* Data

Pada Gambar 4.10 dapat dilihat bahwa jumlah data akhir yang akan digunakan ke tahapan selanjutnya yaitu *labelling*, sejumlah **4242** data.

4.4 *LABELLING*

Proses *labelling* atau memberikan label dilakukan secara manual terhadap teks umpan balik supaya dapat digunakan untuk melatih model SVM. *Labelling* secara manual dilakukan dengan cara membaca satu persatu data umpan balik, kemudian dikategorikan ke dalam kelas positif atau negatif. Berdasarkan data yang telah diberi label, dihasilkan **1902** data dengan label positif dan **2340** data dengan label negatif. Contoh data yang dilabeli secara manual dapat dilihat pada Tabel 4.2

Tabel 4.2 Contoh *Labelling* Manual

No	Result_Stemming	Label
1.	tambah fitur ewallet topup gopay ovo dana shopeepay dan lain lain tolong biar lebih modern	Negatif
2.	mantap mudah guna	Positif

3.	tambah transaksi ewallet auto bintang lima	Negatif
4.	sering error	Negatif
5.	baik	Positif
6.	bantu transaksi	Positif
7.	login mbanking susah cuma logo bima	Negatif
8.	bagus	Positif
9.	sehari aplikasi tidak bisa buka min padahal sistem android aman uninstall ulang kali restart tetap tidak bisa buka padahal semalem bisa cardless terus solusi bagaimana min	Negatif
10.	data bocor maaf kasih bintang satu dulu tingkat disiplin karyawan jaga data nasabah tempat kerja sistem suka potong saldo tiba awal bulan kali jadi bulan padahal data jaga pin password reset waduh tidak rekomendasi deh pakai banget	Negatif

Hasil *labelling* ditunjukkan pada Gambar 4.11.

content	result_clean	result_casefolding	result_tokenizing	result_slangwords	result_stopword	result_stemming	label
Tambahin fitur ewallet, topup gopay, ovo, dana...	Tambahin fitur ewallet topup gopay ovo dana sh...	tambahin fitur ewallet topup gopay ovo dana sh...	['tambahin', 'fitur', 'ewallet', 'topup', 'gop...	['tambahkan', 'fitur', 'ewallet', 'topup', 'go...	['tambahkan', 'fitur', 'ewallet', 'topup', 'go...	tambah fitur ewallet topup gopay ovo dana shop...	Negatif
Mantap.. Mudah digunakan..	Mantap Mudah digunakan	mantap mudah digunakan	['mantap', 'mudah', 'digunakan']	['mantap', 'mudah', 'digunakan']	['mantap', 'mudah', 'digunakan']	mantap mudah guna	Positif
Tambahkan transaksi ke E-wallet, auto bintang ...	Tambahkan transaksi ke Ewallet auto bintang lima	tambahkan transaksi ke ewallet auto bintang lima	['tambahkan', 'transaksi', 'ke', 'ewallet', 'a...	['tambahkan', 'transaksi', 'ke', 'ewallet', 'a...	['tambahkan', 'transaksi', 'ewallet', 'auto', ...	tambah transaksi ewallet auto bintang lima	Negatif
Sering error,,,	Sering error	sering error	['sering', 'error']	['sering', 'error']	['sering', 'error']	sering error	Negatif
Baik	Baik	baik	['baik']	['baik']	['baik']	baik	Positif
Membantu transaksiku	Membantu transaksiku	membantu transaksiku	['membantu', 'transaksiku']	['membantu', 'transaksiku']	['membantu', 'transaksiku']	bantu transaksi	Positif
Login mbanking susahnya.. cuma sampe logo bima...	Login mbanking susahnya cuma sampe logo bima saja	login mbanking susahnya cuma sampe logo bima saja	['login', 'mbanking', 'susahnya', 'cuma', 'sam...	['login', 'mbanking', 'susahnya', 'cuma', 'sam...	['login', 'mbanking', 'susahnya', 'cuma', 'log...	login mbanking susah cuma logo bima	Negatif
Bagus	Bagus	bagus	['bagus']	['bagus']	['bagus']	bagus	Positif
Udh sehari aplikasi ga bisa dibuka ya min, pad...	Udh sehari aplikasi ga bisa dibuka ya min pada...	udh sehari aplikasi ga bisa dibuka ya min pada...	['udh', 'sehari', 'aplikasi', 'ga', 'bisa', 'd...	['sudah', 'sehari', 'aplikasi', 'tidak', 'bisa...	['sehari', 'aplikasi', 'tidak', 'bisa', 'dibuk...	hari aplikasi tidak bisa buka min padahal sist...	Negatif
Data saya bocor maaf saya kasih bintang satu d...	Data saya bocor maaf saya kasih bintang satu d...	data saya bocor maaf saya kasih bintang satu d...	['data', 'saya', 'bocor', 'maaf', 'saya', 'kas...	['data', 'saya', 'bocor', 'maaf', 'saya', 'kas...	['data', 'bocor', 'maaf', 'kasih', 'bintang', ...	data bocor maaf kasih bintang satu dulu tingka...	Negatif

Gambar 4.11 Hasil *Labelling*

4.5 TF-IDF

Setelah dilakukan *labelling*, data bisa digunakan untuk proses *training*. Namun, teks utuh tidak dapat langsung dipakai untuk melatih model, sehingga perlu dilakukan konversi ke vektor numerik, salah satunya dengan menggunakan TF-

IDF. Perhitungan TF-IDF bisa dilakukan secara manual, contoh dokumen yang akan dihitung TF-IDFnya ditunjukkan pada Tabel 4.3.

Tabel 4.3 Dokumen TF-IDF

Dokumen (d)	Kalimat
d1	tambah fitur ewallet topup gopay ovo dana shopeepay dan lain lain tolong biar lebih modern
d2	mantap mudah guna
d3	tambah transaksi ewallet auto bintang
d4	sering error
d5	baik

Terdapat 5 dokumen yang akan digunakan sebagai contoh perhitungan TF-IDF secara manual, yaitu d1, d2, d3, d4, dan d5. Langkah pertama yaitu melakukan perhitungan TF, dengan menggunakan komponen t yaitu *term* (kata) dan d yaitu dokumen. TF adalah perhitungan untuk mengukur seberapa sering sebuah kata muncul dalm sebuah dokumen. Contoh perhitungan TF ditunjukkan pada Tabel 4.4.

Tabel 4.4 Contoh Perhitungan TF

Term	d1	d2	d3	d4	d5
auto	0	0	0.301	0	0
baik	0	0	0	0	0.301
biar	0.301	0	0	0	0
bintang	0	0	0.301	0	0
dan	0.301	0	0	0	0
dana	0.301	0	0	0	0
error	0	0	0	0.301	0
ewallet	0.301	0	0.301	0	0
fitur	0.301	0	0	0	0
gopay	0.301	0	0	0	0
guna	0	0.301	0	0	0
lain	0.301	0	0	0	0
lebih	0.301	0	0	0	0

Term	d1	d2	d3	d4	d5
mantap	0	0.301	0	0	0
modern	0.301	0	0	0	0
mudah	0	0.301	0	0	0
ovo	0.301	0	0	0	0
sering	0	0	0	0.301	0
shopeepay	0.301	0	0	0	0
tambah	0.301	0	0.301	0	0
tolong	0.301	0	0	0	0
topup	0.301	0	0	0	0
transaksi	0	0	0.301	0	0

Selanjutnya yaitu menghitung IDF untuk mengukur seberapa penting sebuah kata dalam kumpulan dokumen. Komponen dalam perhitungan IDF diantaranya, *term* (kata), df atau jumlah dokumen yang mengandung *term* tersebut, serta N atau banyaknya dokumen. Contoh perhitungan IDF ditunjukkan pada Tabel 4.5.

Tabel 4.5 Contoh Perhitungan IDF

Term	df	IDF
auto	1	0.699
baik	1	0.699
biar	1	0.699
bintang	1	0.699
dan	1	0.699
dana	1	0.699
error	1	0.699
ewallet	2	0.398
fitur	1	0.699
gopay	1	0.699
guna	1	0.699
lain	1	0.699
lebih	1	0.699

Term	df	IDF
mantap	1	0.699
modern	1	0.699
mudah	1	0.699
ovo	1	0.699
sering	1	0.699
shopeepay	1	0.699
tambah	2	0.398
tolong	1	0.699
topup	1	0.699
transaksi	1	0.699

Langkah selanjutnya yaitu menghitung TF-IDF dengan cara mengalikan hasil TF dan IDF yang telah dilakukan. Contoh perhitungan TF-IDF ditunjukkan pada Tabel 4.6.

Tabel 4.6 Contoh Perhitungan TF-IDF

Term	d1	d2	d3	d4	d5
auto	0	0	0.210	0	0
baik	0	0	0	0	0.210
biar	0.210	0	0	0	0
bintang	0	0	0.210	0	0
dan	0.210	0	0	0	0
dana	0.210	0	0	0	0
error	0	0	0	0.210	0
ewallet	0.120	0	0.120	0	0
fitur	0.210	0	0	0	0
gopay	0.210	0	0	0	0
guna	0	0.210	0	0	0
lain	0.210	0	0	0	0
lebih	0.210	0	0	0	0
mantap	0	0.210	0	0	0

Term	d1	d2	d3	d4	d5
modern	0.210	0	0	0	0
mudah	0	0.210	0	0	0
ovo	0.210	0	0	0	0
sering	0	0	0	0.210	0
shopeepay	0.210	0	0	0	0
tambah	0.120	0	0.120	0	0
tolong	0.210	0	0	0	0
topup	0.210	0	0	0	0
transaksi	0	0	0.210	0	0

Perhitungan TF-IDF pada penelitian ini dilakukan secara otomatis menggunakan library Python yaitu dengan mengimpor *TfidfVectorizer* dari *library sklearn*. Proses perhitungan TF-IDF ditunjukkan pada Kode Program 4.9.

Kode Program 4.9 Perhitungan TF-IDF

```
1. X = df['result_stemming']
2. y = df['label']
3. tfidf = TfidfVectorizer()
4. X_tfidf = tfidf.fit_transform(X)
5. features_df = pd.DataFrame(X_tfidf.toarray(),
   columns=tfidf.get_feature_names_out())
```

Berikut tahapan dalam ektraksi fitur menggunakan TF-IDF:

a. X = df['result_stemming'] : X merupakan data teks yang akan dihitung TF-IDFnya yang merupakan kolom ‘result_stemming’ dari DataFrame ‘df’.
b. y = df['label'] : merupakan target atau label prediksi yang diambil dari kolom ‘label’ pada DataFrame ‘df’.
c. tfidf = TfidfVectorizer() : Membuat objek dari kelas *TfidfVectorizer* yang digunakan untuk menghitung TF-IDF.
d. X_tfidf = tfidf.fit_transform(X) : Membuat matriks yang berisi hasil dari TF-IDF, ‘fit_transform(X)’ digunakan untuk mempelajari seluruh kata unik dari data X kemudian mengubahnya menjadi angka berdasarkan nilai TF-IDF.

e. features_df=pd.DataFrame(X_tfidf.toarray(),columns=tfidf.get_feature_names _out()) : Mengubah hasil matriks TF-IDF menjadi array dengan menggunakan perintah toarray(), lalu mengambil semua kata yang sudah diidentifikasi dari teks yang kemudian diubah bentuknya kedalam DataFrame.

Hasil dari perhitungan TF-IDF ditunjukkan pada Gambar 4.12.

	aplikasi	bagus	baik	bank	bantu	bima	bisa	buka	error	jateng	...	padahal	pakai	sangat	sering	terus	tidak	tolong	transaksi	transfer	update
0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.181064	0.000000	0.0	0.0
1	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
2	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.239394	0.0	0.0
3	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.694666	0.000000	...	0.0	0.0	0.000000	0.719332	0.0	0.0	0.000000	0.000000	0.0	0.0
4	0.000000	0.0	1.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4237	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
4238	0.154733	0.0	0.0	0.000000	0.0	0.104833	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
4239	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
4240	0.000000	0.0	0.0	0.000000	0.0	0.357392	0.0	0.0	0.000000	0.000000	...	0.0	0.0	0.329191	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0
4241	0.223270	0.0	0.0	0.274374	0.0	0.000000	0.0	0.0	0.000000	0.308549	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0	0.0

4242 rows × 29 columns

Gambar 4.12 Hasil Perhitungan TF-IDF

4.6 ANALISIS SENTIMEN DENGAN SVM

Setelah melakukan ekstraksi fitur dengan TF-IDF, dilanjutkan dengan pelatihan model analisis sentimen dengan SVM. Proses pelatihan model sentimen dengan menggunakan algoritma SVM ditunjukkan pada Kode Program 4.10.

Kode Program 4.10 Model Sentimen dengan SVM

```
1. X_train, X_test, y_train, y_test = train_test_split
   (X_tfidf, y, test_size=0.2, random_state=42)
2. svm_model = SVC(kernel='linear')
3. svm_model.fit(X_train, y_train)
4. y_pred_train_svm = svm_model.predict(X_train)
5. y_pred_test_svm = svm_model.predict(X_test)
```

Berikut tahapan dalam pelatihan model sentimen dengan SVM:

a. X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42) : Membagi data menjadi data latih (*training*) dan data uji (*testing*), dengan fitur inputnya hasil dari TF-IDF pada data teks ('X_tfidf') dan y berupa label (target) dari data. 'test_size = 0.2' menunjukkan bahwa 20% data dijadikan data *testing*, 80% untuk data *training*. Kemudian, 'random_state = 42' digunakan supaya pembagian data selalu konsisten saat dijalankan ulang (*reproducible*).

b. svm_model = SVC(kernel='linear') : Membuat objek model SVM untuk klasifikasi dengan menggunakan kernel linear, yang artinya model akan mencari garis lurus *(hyperplane)* yang memisahkan kelas-kelas di ruang fitur.
c. svm_model.fit(X_train, y_train) : Melatih model SVM dengan penggunakan perintah fit() yang mana model akan mempelajari hubungan antara fitur (X_train) dan label (y_train).
d. y_pred_train_svm = svm_model.predict(X_train) : Melakukan prediksi pada data latih untuk melihat seberapa baik model belajar dari data pelatihan.
e. y_pred_test_svm = svm_model.predict(X_test) : Melakukan prediksi pada data uji untuk mengevaluasi performa model terhadap data yang belum pernah dilihat sebelumnya.

Selanjutnya adalah tahapan evaluasi untuk mengetahui performa atau tingkat keakuratan dari model yang telah dilatih. Proses evaluasi model dilakukan pada kedua data yaitu data *training* dan data *testing*, yang ditunjukkan pada Kode Program 4.11 dan Kode Program 4.12.

Kode Program 4.11 Evaluasi Model SVM pada Data *Training*

```
1. accuracy_train = accuracy_score(y_train, y_pred_train_svm)
2. print("\nClassification Report (Train):")
3. print(classification_report(y_train, y_pred_train_svm))
4. print('SVM - accuracy_train:', accuracy_train)
```

Kode Program 4.12 Evaluasi Model SVM pada Data *Testing*

```
1. accuracy_test = accuracy_score(y_test, y_pred_test_svm)
2. print("\nClassification Report (Test):")
3. print(classification_report(y_test, y_pred_test_svm))
4. print('SVM - accuracy_test:', accuracy_test)
```

Berikut tahapan evaluasi model yang dilakukan pada data *training* dan data *testing*:

a. Menghitung akurasi model pada data *training* dan data *testing* dengan menggunakan perintah ‘accuracy_score’ yang menghitung rasio prediksi benar terhadap total prediksi. ‘y_train’ dan ‘y_test’ merupakan label sebenarnya, sedangkan ‘y_pred_train_svm’ dan ‘y_pred_test_svm’ merupakan hasil prediksi model.

b. Kemudian, mencetak laporan evaluasi *(classification report)* dengan menggunakan perintah 'classification_report'. Metrik yang dihasilkan dari evaluasi ini diantaranya *precision, recall, f1-score,* dan *support*.

Hasil evaluasi model pada data *training* dalam bentuk *classification report* ditunjukkan pada Gambar 4.13.

```
Classification Report (Train):
              precision    recall  f1-score   support

     Negatif       0.96      0.98      0.97      1857
     Positif       0.97      0.96      0.96      1536

    accuracy                           0.97      3393
   macro avg       0.97      0.97      0.97      3393
weighted avg       0.97      0.97      0.97      3393

SVM - accuracy_train: 0.9675803124078987
```

Gambar 4.13 *Classification Report* Data *Training*

Berdasarkan Gambar 4.13, diperlihatkan bahwa akurasi model pada data *training* yaitu 96,75% yang berarti sekitar 97 dari setiap 100 sampel pelatihan diprediksi benar oleh model SVM. Hal tersebut didukung oleh hasil *precision* dan *recall* yang nilainya tidak jauh berbeda, menunjukkan bahwa model memiliki keseimbangan yang baik dalam menangani kesalahan jenis *false positive* dan *false negative*. Jika nilai *precision* dan *recall* seimbang, maka *F-1 score* juga tinggi yang menunjukkan performa modelnya bagus.

Hasil evaluasi model pada data *testing* dalam bentuk *classification report* ditunjukkan pada Gambar 4.14.

```
Classification Report (Test):
              precision    recall  f1-score   support

     Negatif       0.94      0.97      0.96       483
     Positif       0.95      0.92      0.94       366

    accuracy                           0.95       849
   macro avg       0.95      0.95      0.95       849
weighted avg       0.95      0.95      0.95       849

SVM - accuracy_test: 0.9481743227326266
```

Gambar 4.14 *Classification Report* Data *Testing*

Berdasarkan Gambar 4.14, diperlihatkan bahwa akurasi model pada data *testing* yaitu sebesar 94, 81% yang berarti sekitar 95 dari setiap 100 sampel data uji berhasil

diprediksi dengan benar oleh model SVM. Hasil tersebut menunjukkan bahwa model memiliki performa yang sangat baik dalam melakukan klasifikasi terhadap dua kelas, yaitu Negatif dan Positif. Hal ini didukung oleh hasil evaluasi lain seperti *precision, recall,* dan *f1-score*, yang semuanya memiliki nilai yang tinggi dan seimbang, yang menunjukkan model memiliki kinerja baik dengan kesalahan prediksi yang relatif rendah.

Langkah selanjutnya yaitu melakukan prediksi menggunakan model SVM yang telah dibuat untuk mengetahui sentimen dari umpan balik pengguna Bima Mobile. Proses prediksi data umpan balik pengguna Bima Mobile ditunjukkan pada Kode Program 4.13.

Kode Program 4.13 Prediksi Data dengan Model SVM

```
1. predict_data = svm_model.predict(X_tfidf)
2. df['label_prediksi'] = predict_data
3. predict_label_counts = df['label_prediksi'].value_counts()
4. print(predict_label_counts)
```

Berikut tahapan dalam melakukan prediksi dengan menggunakan model SVM:

a. predict_data = svm_model.predict(X_tfidf) : Menggunakan model SV (‘svm_model’) untuk melakukan prediksi terhadap data fitur ‘X_tfidf’ yang merupakan data input yang telah diolah dengan TF-IDF. Prediksi dilakukan dengan menggunakan perintah predict(), kemudian ‘predict_data’ akan berisi array atau daftar label hasil prediksi dari model.
b. df['label_prediksi'] = predict_data : Hasil prediksi ditambahkan ke kolom baru dengan nama ‘label_prediksi’ ke dalam DataFrame ‘df’.
c. predict_label_counts = df['label_prediksi'].value_counts() : Menghitung jumlah dari masing-masing label hasil prediksi dengan menggunakan perintah value_counts().
d. print(predict_label_counts) : Digunakan untuk menampilkan hasil jumlah masing-masing label prediksi.

Rincian hasil prediksi dari model SVM yang telah dibuat ditunjukkan pada Gambar 4.15.

```
label_prediksi
Negatif    2374
Positif    1868
Name: count, dtype: int64
```

Gambar 4.15 Hasil Prediksi Sentimen

Berdasarkan Gambar 4.15, distribusi sentimen pada data prediksi menunjukkan bahwa 1868 umpan balik memberikan sentimen positif, sedangkan 2374 umpan balik memberikan sentimen negatif. Sebelum masuk ke tahapan pemodelan topik, pisahkan terlebih dahulu data dari masing-masing kelas. Proses pemisahan data kategori positif dan negatif ditunjukkan pada Kode Program 4.14.

Kode Program 4.14 Pemisahan Data Positif dan Negatif

```
1. positif_df = df[df['label_prediksi'] == 'Positif']
2. negatif_df = df[df['label_prediksi'] == 'Negatif']
```

Hasil pemisahan data dapat dilihat pada Gambar 4.16 dan 4.17.

content	result_clean	result_casefolding	result_tokenizing	result_slangwords	result_stopword	result_stemming	label	label_prediksi
Mantap.. Mudah digunakan..	Mantap Mudah digunakan	mantap mudah digunakan	['mantap', 'mudah', 'digunakan']	['mantap', 'mudah', 'digunakan']	['mantap', 'mudah', 'digunakan']	mantap mudah guna	Positif	Positif
Baik	Baik	baik	['baik']	['baik']	['baik']	baik	Positif	Positif
Membantu transaksiku	Membantu transaksiku	membantu transaksiku	['membantu', 'transaksiku']	['membantu', 'transaksiku']	['membantu, 'transaksiku']	bantu transaksi	Positif	Positif
Bagus	Bagus	bagus	['bagus']	['bagus']	['bagus']	bagus	Positif	Positif
sangat bagus	sangat bagus	sangat bagus	['sangat', 'bagus']	['sangat', 'bagus']	['sangat', 'bagus']	sangat bagus	Positif	Positif

Gambar 4.16 Hasil Data Positif

content	result_clean	result_casefolding	result_tokenizing	result_slangwords	result_stopword	result_stemming	label	label_prediksi
Tambahin fitur ewallet, topup gopay, ovo, dana...	Tambahin fitur ewallet topup gopay ovo dana sh...	tambahin fitur ewallet topup gopay ovo dana sh...	['tambahin', 'fitur', 'ewallet', 'topup', 'gop...	['tambahkan', 'fitur', 'ewallet', 'topup', 'go...	['tambahkan', 'fitur', 'ewallet', 'topup', 'go...	tambah fitur ewallet topup gopay ovo dana shop...	Negatif	Negatif
Tambahkan transaksi ke E-wallet, auto bintang ...	Tambahkan transaksi ke Ewallet auto bintang lima	tambahkan transaksi ke ewallet auto bintang lima	['tambahkan', 'transaksi', 'ke', 'ewallet', 'a...	['tambahkan, 'transaksi, 'ke', 'ewallet, 'a...	['tambahkan', 'transaksi', 'ewallet', 'auto', ...	tambah transaksi ewallet auto bintang lima	Negatif	Negatif
Sering error,,	Sering error	sering error	['sering', 'error']	['sering, 'error']	['sering', 'error']	sering error	Negatif	Negatif
Login mbenking susehnya... cuma sampai logo bima...	Login mbenking susehnya cuma sampai logo bima saja	login mbenking susehnya cuma sampai logo bima saja	['login', 'mbenking', 'susehnya', 'cuma', 'sam...	['login', 'mbenking', 'susehnya', 'cuma', 'sam...	['login', 'mbenking', 'susehnya', 'cuma', 'log...	login mbenking suseh cuma logo bima	Negatif	Negatif
Udh sehari aplikasi ga bisa dibuka ya min, pad...	Udh sehari aplikasi ga bisa dibuka ya min pada...	udh sehari aplikasi ga bisa dibuka ya min pada...	['udh, sehari', 'aplikasi', 'ga', 'bisa', 'd...	['sudah, sehari, 'aplikasi', 'tidak', 'bisa...	['sehari', 'aplikasi', 'tidak', 'bisa, dibuk...	hari aplikasi tidak bisa buka min padahal sist...	Negatif	Negatif

Gambar 4.17 Hasil Data Negatif

4.7 PEMODELAN TOPIK DENGAN NMF

Pemodelan topik dilakukan pada masing-masing kelas hasil sentimen dari data umpan balik yang ada. Tahap pemodelan topik diawali dengan menentukan jumlah topik optimal untuk pembuatan model NMF. Pada penelitian ini penentuan jumlah topik optimal dilakukan dengan menggunakan *coherence score*. Penghitungan *coherence score* dilakukan pada masing-masing kategori sentimen, yaitu data negatif dan data positif.

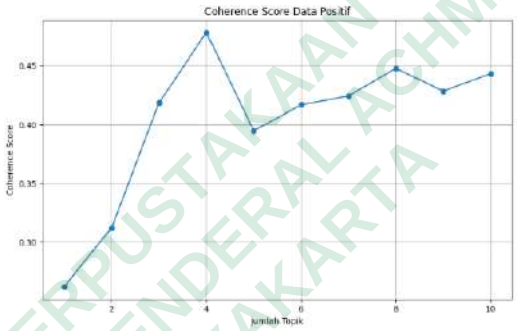
1. *Coherence Score* Data Positif

Proses penentuan jumlah topik optimal pada kategori data positif, ditunjukkan pada Kode Program 4.15.

Kode Program 4.15 *Coherence Score* Data Positif

```
1. optimal_pos = optimize_nmf_topics(positif_df)
```

Kode tersebut digunakan untuk memanggil fungsi 'optimize_nmf_topics' dengan inputnya yaitu kategori data positif dan menghasilkan *output* berupa *coherence score* dari masing-masing jumlah topik serta menentukan jumlah topik optimalnya. Grafik hasil *coherence score* dari kategori data positif dapat dilihat pada Gambar 4.18.



Gambar 4.18 Grafik *Coherence Score* Data Positif

Coherence score dari masing-masing jumlah topik, ditunjukkan pada Tabel 4.7.

Tabel 4.7 Hasil *Coherence Score* Positif

Jumlah Topik	*Coherence Score*
1	0,2623
2	0,3122
3	0,4182
4	**0,4780**
5	0,3947
6	0,4165
7	0,4241
8	0,4476
9	0,4282
10	0,4431

Berdasarkan Gambar 4.18 dan Tabel 4.7, dapat disimpulkan bahwa performa model topik meningkat seiring dengan bertambahnya jumlah topik hingga titik tertentu. Dimulai dari 1 topik dengan *coherence score* sebesar 0,2623, skor tersebut terus meningkat dan mencapai nilai tertinggi sebesar 0,4780 pada 4 topik. Peningkatan ini menunjukkan bahwa penambahan jumlah topik hingga 4 membantu model dalam mengelompokkan kata-kata menjadi topik-topik yang lebih koheren dan bermakna secara semantik. Namun, setelah melewati titik tersebut (topik ke-5 hingga ke-10), *coherence score* mulai menurun atau mengalami fluktuasi, menandakan bahwa jumlah topik yang berlebihan dapat mengurangi konsistensi dan kualitas topik yang terbentuk. Oleh karena itu, jumlah topik yang optimal untuk data positif adalah sebanyak **4 topik**, sebagaimana tercermin dari *coherence score* tertingginya.

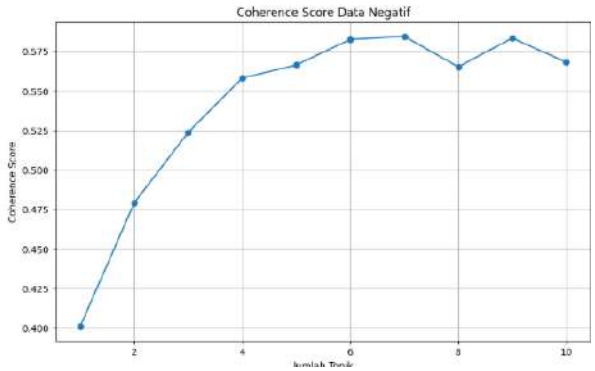
2. *Coherence Score* Data Negatif

Proses penentuan jumlah topik optimal pada kategori data negatif, ditunjukkan pada Kode Program 4.16.

Kode Program 4.16 *Coherence Score* Data Negatif

```
1. optimal_neg = optimize_nmf_topics(negatif_df)
```

Kode tersebut digunakan untuk memanggil fungsi ‘optimize_nmf_topics’ dengan inputannya yaitu kategori data negatif dan menghasilkan output *berupa coherence score* dari masing-masing jumlah topik serta menentukan jumlah topik optimalnya. Grafik hasil *coherence score* dari kategori data negatif dapat dilihat pada Gambar 4.19.



Gambar 4.19 Grafik *Coherence Score* Data Negatif

Coherence score dari masing-masing jumlah topik, ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil *Coherence Score* Data Negatif

Jumlah Topik	***Coherence Score***
1	0,4010
2	0,4791
3	0,5237
4	0,5580
5	0,5662
6	0,5824
7	**0,5844**
8	0,5651
9	0,5833
10	0,5680

Berdasarkan Gambar 4.19 dan Tabel 4.8, dapat disimpulkan bahwa performa model topik meningkat seiring dengan bertambahnya jumlah topik hingga titik tertentu. Dimulai dari 1 topik dengan *coherence score* sebesar 0,4010, skor tersebut terus meningkat hingga mencapai nilai tertinggi sebesar 0,5844 pada 7 topik. Peningkatan ini menunjukkan bahwa penambahan jumlah topik hingga 7 membantu model dalam mengelompokkan kata-kata menjadi topik-topik yang lebih koheren dan bermakna secara semantik. Namun, setelah melewati titik tersebut (topik ke-8 hingga ke-10), *coherence score* mulai menurun, menandakan bahwa jumlah topik yang berlebihan menyebabkan konsistensi topik berkurang. Oleh karena itu, jumlah topik yang optimal untuk data negatif adalah sebanyak **7 topik**, tercermin dari nilai *coherence score* tertingginya.

Setelah menentukan jumlah topik optimal dari masing-masing kategori data, tahap selanjutnya yaitu menerapkan model NMF untuk mengetahui topik utama dari setiap kategori data. Pada penelitian ini, penerapan model NMF diawali dengan membuat fungsi terlebih dahulu supaya dapat dipanggil untuk kedua kategori data. Fungsi untuk penerapan model NMF ditunjukkan pada Kode Program 4.17.

Kode Program 4.17 Model NMF

```
1. def nmf_topic_modeling(data, n_topics, n_words=10):
2.     vectorizer = TfidfVectorizer()
3.     X = vectorizer.fit_transform(data['result_stemming'])
4.     nmf = NMF(n_components=n_topics, random_state=42,
             init='nndsvd')
5.     W = nmf.fit_transform(X)
6.     H = nmf.components_
7.     words = vectorizer.get_feature_names_out()
8.     topics = []
9.     topic_words = []
10.       for topic_idx, topic in enumerate(H):
11.           top_words = [words[i] for i in
              topic.argsort()[:-n_words - 1:-1]]
12.           topics.append({'Topic': topic_idx+1, 'Words': ',
              '.join(top_words)})
13.           topic_words.append(top_words)
```

Berikut tahapan penerapan model NMF pada fungsi ‘nmf_topic_modeling’():

a. Fungsi tersebut memiliki 3 parameter yaitu ‘data’ (hasil stemming), ‘n_topics’ (jumlah topik), ‘n_words’ (jumlah kata teratas yang ditampilkan untuk setiap topik).

b. vectorizer = TfidfVectorizer()

   X = vectorizer.fit_transform(data['result_stemming'])

   Membuat representasi teks dalam bentuk TF-IDF dari data *stemming*.

c. nmf = NMF(n_components=n_topics, random_state=42, init='nndsvd')

   W = nmf.fit_transform(X)

   H = nmf.components_

   Melakukan pemodelan topik NMF dengan jumlah topik ‘n_topics’, menggunakan ‘random_state’ supaya hasilnya konsisten, serta init=’nndvsd’ untuk mempercepat dan menstabilkan pelatihan. ‘W’ merupakan matriks dokumen-topik, sedangkan ‘H’ merupakan matriks topik-kata.

d. words = vectorizer.get_feature_names_out()

   topics = []

   topic_words = []

   for topic_idx, topic in enumerate(H):

```
    top_words = [words[i] for i in topic.argsort()[:-n_words - 1:-1]]
    topics.append({'Topic': topic_idx+1, 'Words': ', '.join(top_words)})
```

Menghasilkan semua kata dari TF-IDF, lalu iterasi tiap topik (H) untuk mendapatkan kata-kata yang paling penting. Kemudian urutkan nilai kata dalam topik dengan perintah argsort() dan ambil. Tambahkan urutan topik dan kata-kata utamanya ke list 'topics'.

Penggunaan fungsi 'nmf_topic_modeling' pada masing-masing kategori data ditunjukkan pada Kode Program 4.18 dan 4.19.

Kode Program 4.18 Model NMF Data Positif

```
1. nmf_topic_modeling(positif_df, n_topics=optimal_pos)
```

Kode Program 4.19 Model NMF Data Negatif

```
1. nmf_topic_modeling(negatif_df, n_topics=optimal_neg)
```

Kode tersebut digunakan untuk memanggil fungsi 'nmf_topic_modeling' yang diterapkan pada data negatif dan data positif, dengan jumlah topiknya sama dengan hasil dari 'optimal_neg' dan 'optimal_pos'. Berdasarkan hasil *coherence score* yang telah dilakukan sebelumnya, jumlah topik yang digunakan pada data negatif sebanyak 7 topik, sedangkan jumlah topik pada data positif sebanyak 4 topik. Dari hasil pemodelan topik menggunakan NMF, diperoleh sejumlah topik yang masing-masing direpresentasikan oleh sekumpulan kata kunci teratas. Hasil kata kunci teratas di setiap topik dari masing-masing kategori data ditunjukkan pada Tabel 4.9 dan Tabel 4.10.

Tabel 4.9 Hasil Pemodelan Topik Data Positif

Topik	Kata Kunci
Topik 1	bagus, aplikasi, sekali, banget, mudah, layanan, sangat, transaksi, cepat, manfaat
Topik 2	bantu, sangat, transaksi, mudah, aplikasi, baik, sekali, manfaat, terimakasih, bank
Topik 3	mantap, aplikasi, betul, bank, jateng, mudah, makin, transaksi, lebih, jiwa

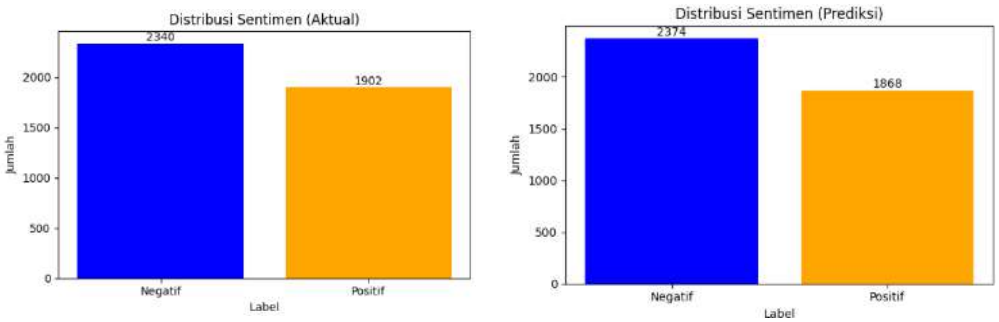
Topik	Kata Kunci
Topik 4	oke, aplikasi, banget, bima, bank, jateng, makin, mudah, transaksi, mobile

Tabel 4.10 Hasil Pemodelan Topik Data Negatif

Topik	Kata Kunci
Topik 1	bisa, tidak, buka, aplikasi, bima, kok, guna, hari, masuk, server
Topik 2	error, sering, terus, aplikasi, server, trouble, transfer, perbaiki, jadi, banget
Topik 3	nomor, daftar, hp, masuk, tidak, padahal, valid, mau, benar, atm
Topik 4	transfer, bank, tambah, fitur, tolong, lain, pakai, transaksi, rekening, lebih
Topik 5	update, malah, bisa, baru, versi, tidak, kok, minta, suruh, bagaimana
Topik 6	susah, login, banget, mau, gagal, terus, aplikasi, respon, service, server
Topik 7	lemot, banget, aplikasi, sekarang, sangat, jelek, parah, sekali, lama, perbaiki

4.8 VISUALISASI DAN ANALISIS HASIL

Tahap terakhir yaitu menyajikan visualisasi dan analisis terhadap hasil yang diperoleh dari proses analisis sentimen dan pemodelan topik. Visualisasi tersebut digunakan untuk memberikan gambaran lebih jelas mengenai distribusi sentimen serta topik-topik yang dihasilkan. Selanjutnya, dilakukan analisis guna memahami topik-topik yang muncul pada masing-masing kategori sentimen. Distribusi sentimen dapat divisualisasikan ke dalam berbagai diagram, salah satunya diagram batang *(bar chart)*. *Bar chart* digunakan untuk menampilkan jumlah data pada maing-masing kategori, sehingga memudahkan dalam melihat perbandingan secara kuantitatif. Perbandingan distribusi sentimen pada data aktual dan prediksi ditunjukkan pada Gambar 4.20.



Gambar 4.20 Hasil Distribusi Sentimen

Berdasarkan Gambar 4.20, dapat dilihat bahwa pada data aktual terdapat 2340 (55%) sentimen negatif dan 1902 (45%) sentimen positif, sedangkan dari hasil prediksi didapatkan 2374 sentimen negatif (56%) dan 1868 sentimen positif(44%). Dari hasil tersebut dapat diketahui bahwa sentimen negatif lebih mendominasi, yang menunjukkan pengguna Bima *Mobile* cenderung memberikan umpan balik yang bersifat kurang puas atau mengungkapkan keluhan terhadap aplikasi tersebut. Dominasi sentimen negatif ini dapat menjadi indikator adanya aspek-aspek dalam aplikasi yang perlu mendapatkan perhatian dan perbaikan lebih lanjut oleh pihak pengembang. Selain itu, hasil prediksi sentimen dengan data aktualnya hanya memiliki sedikit perbedaan, yang menunjukkan bahwa model SVM yang digunakan memiliki performa klasifikasi yang cukup baik. Model ini mampu mengenali pola-pola dalam data teks umpan balik secara efektif, sehingga mampu mengelompokkan sentimen dengan tingkat akurasi yang cukup tinggi.

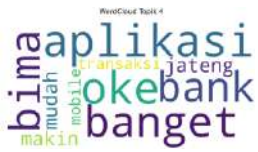
Setelah data dikategorikan ke dalam kategori sentimen positif dan negatif, dilakukan pemodelan topik dengan model NMF. Visualisasi dari hasil pemodelan topik dapat dilakukan dengan menggunakan *WordCloud*. *WordCloud* akan menampilkan kata-kata kunci dengan ukuran yang bervariasi, di mana semakin besar ukuran suatu kata, maka semakin tinggi frekuensi kemunculannya dalam topik tersebut. Visualisasi ini sangat membantu dalam memberikan gambaran tentang hal yang sering dibahas pengguna, sehingga dapat dijadikan dasar untuk analisis lebih lanjut maupun pengambilan keputusan.

1. Analisis Topik Positif

Hasil pemodelan topik dengan menggunakan teknik perhitungan *coherence score* ‘c_v’ pada kategori data positif menghasilkan jumlah topik optimal sebanyak

4 topik dengan 10 kata kunci yang ditampilkan untuk membentuk topik utama. Identifikasi dan analisis topik pada kategori data positif ditunjukkan pada Tabel 4.11.

Tabel 4.11 Identifikasi dan Analisis Topik Positif

No Topik	*WordCloud*	Topik	Jumlah Umpan Balik
Topik 1		Topik 1 berfokus pada **kepuasan pengguna**. Pengguna menilai aplikasinya bagus, cepat, memberikan kemudahan dan manfaat layanan transaksi. Hal tersebut menjadi indikator bahwa Bima Mobile memberikan pengalaman yang memuaskan.	332
Topik 2		Topik 2 berfokus pada **kemudahan penggunaan**. Pengguna merasa sangat terbantu, yang menandakan bahwa aplikasi berfungsi sebagai solusi praktis dalam kehidupan sehari-hari, terutama untuk transaksi.	792
Topik 3		Topik 3 berfokus pada **kepuasan pengguna terhadap pembaruan aplikasi**. Topik ini menunjukkan bahwa pengembang aplikasi telah berhasil membangun hubungan positif dengan pengguna melalui proses *update* yang dilakukan.	418
Topik 4		Topik 4 hampir mirip dengan topik 3, berisi ungkapan **kepuasan peningkatan performa dan fungsionalitas aplikasi**. Pengalaman pengguna aplikasi mobile dari Bank Jateng terus meningkat, aplikasi makin mudah digunakan dan performanya semakin baik.	330

Terdapat beberapa umpan balik positif yang diberikan pengguna menunjukkan bahwa aplikasi Bima *Mobile* telah berhasil membangun kepercayaan dan kepuasan di beberapa kalangan penggunanya, baik dari sisi kemudahan, kebermanfaatan untuk transaksi, maupun peningkatan berkelanjutan. Kemudahan penggunaan menjadi aspek yang paling banyak disebutkan dalam umpan balik positif. Hal tersebut menunjukkan bahwa aplikasi layak untuk terus ditingkatkan serta dipromosikan sebagai solusi perbankan digital yang handal dan terpercaya, dengan melakukan perbaikan pada aspek yang masih sering dikeluhkan pengguna.

2. Analisis Topik Negatif

Hasil pemodelan topik dengan menggunakan teknik perhitungan *coherence score* 'c_v' pada kategori data negatif menghasilkan jumlah topik optimal sebanyak 7 topik dengan 10 kata kunci yang ditampilkan untuk membentuk topik utama. Identifikasi dan analisis topik pada kategori data negatif ditunjukkan pada Tabel 4.12.

Tabel 4.122 Identifikasi dan Analisis Topik Data Negatif

No Topik	*WordCloud*	Topik	Jumlah Umpan Balik
Topik 1		Topik 1 berfokus pada **akses aplikasi**, yang menggambarkan bahwa pengguna sering mengalami kendala saat mencoba membuka atau masuk ke aplikasi seperti *stuck* di logo Bima *Mobile*, dan lainnya. Permasalahan lain yaitu berkaitan dengan *server* atau *downtime* yang berkepanjangan. Dari hal tersebut, maka perlu adanya peningkatan stabilitas *server* dan respon cepat terhadap gangguan layanan.	414
Topik 2		Topik 2 berfokus pada **gangguan teknis dan *error***. Topik ini mengarah pada frekuensi error dan gangguan	277

No Topik	*WordCloud*	Topik	Jumlah Umpan Balik
		teknis lainnya, terutama saat menggunakan fitur tertentu seperti transfer atau aplikasi secara umum. Berdasarkan permasalahan tersebut menunjukkan pentingnya peningkatan serta pengujian fitur secara berkala untuk memastikan performa aplikasi tetap optimal dalam berbagai kondisi penggunaan.	
Topik 3		Topik 3 berfokus pada **registrasi dan verifikasi nomor**. Topik ini membahas masalah saat mendaftar atau verifikasi nomor hp, terutama kendala masuk maupun validasi data. Hal tersebut menunjukkan perlunya perbaikan dalam proses registrasi, seperti dalam hal integrasi dengan sistem otentikasi atau basis data nomor telepon.	453
Topik 4		Topik 4 berfokus pada **penambahan fitur**. Pengguna menginginkan adanya penambahan fitur untuk melakukan transfer maupun transaksi lainnya. Hal ini menunjukkan bahwa aplikasi mungkin belum sepenuhnya memenuhi kebutuhan transaksi perbankan pengguna, sehingga menjadi peluang untuk pengembangan fitur baru agar lebih kompetitif.	705
Topik 5		Topik 5 berfokus pada **masalah setelah pembaruan *(update)***. Pengguna mengeluhkan bahwa setelah melakukan *update*, muncul beberapa masalah pada	191

No Topik	*WordCloud*	Topik	Jumlah Umpan Balik
		aplikasi. Hal itu mencerminkan adanya penurunan kualitas atau bug baru pasca-*update*, sehingga perlu dilakukan pengujian yang lebih menyeluruh sebelum rilis pembaruan agar tidak mengganggu fungsionalitas yang sudah ada.	
Topik 6		Topik 6 berfokus pada **kendala login**. Topik ini berkaitan dengan kesulitan *login* ke aplikasi, baik karena respon *service* gagal atau faktor lainnya. Hal tersebut menunjukkan adanya masalah sistemik atau bug yang menyebabkan proses *login* gagal. Masalah ini perlu diperhatikan karena login adalah titik awal pengalaman pengguna, dan jika terganggu, bisa menyebabkan penurunan kepercayaan terhadap aplikasi.	224
Topik 7		Topik 7 berfokus pada **performa aplikasi**. Topik ini menyoroti keluhan terhadap performa aplikasi yang lambat atau lemot. Dari permasalah tersebut, maka diperlukan optimasi perforam aplikasi, termasuk pengelolaan memori, efisiensi *backend*, dan *update* yang memperbaiki lag.	106

Secara keseluruhan, permasalahan yang dikeluhkan pengguna aplikasi Bima *Mobile* diantaranya kesulitan akses, gangguan teknis dan *error* saat penggunaan, masalah registrasi dan verifikasi nomor, penambahan fitur, munculnya *bug* setelah *update*, kendala *login*, dan performa aplikasi yang lambat. Penambahan fitur menjadi hal utama yang sering dikeluhkan. Sebagian besar pengguna menginginkan

fitur yang lebih lengkap dan kompetitif untuk memenuhi kebutuhan pengguna dalam transaksi modern. Selain memberikan pemahaman mengenai sentimen dan tren topik dari aplikasi Bima *Mobile*, hasil penelitian ini juga menunjukkan bahwa data umpan balik pengguna pada *Play Store* dapat menjadi sumber informasi yang menggambarkan pengalaman pengguna secara lebih menyeluruh jika dibandingkan dengan metode lain seperti kuesioner maupun observasi. Selain itu, kelebihan dari penelitian ini terletak pada keberhasilan penerapan metode SVM dan NMF, ditunjukkan melalui performa model yang baik serta hasil topik yang mudah diinterpretasikan sehingga mempermudah dalam memahami tren topik yang muncul dalam umpan balik pengguna.

Adapun kekurangan dari penelitian ini terletak pada proses *labelling* yang masih dilakukan secara manual oleh manusia, sehingga sangat bergantung pada sudut pandang pribadi. Hal ini dapat menimbulkan subjektivitas, potensi inkonsistensi, serta ketergantungan pada tingkat ketelitian individu dalam memahami konteks setiap umpan balik pengguna. Selain itu, proses manual ini juga memerlukan waktu yang cukup lama, terutama ketika *volume* data yang dianalisis besar. Kekurangan lainnya adalah penggunaan data umpan balik yang mencakup seluruh periode tanpa mempertimbangkan waktu setelah pembaruan aplikasi. Akibatnya, penelitian ini belum mampu mengidentifikasi secara spesifik persepsi pengguna setelah dilakukan pembaruan, sehingga perubahan pada aspek-aspek yang sebelumnya dikeluhkan pengguna tidak dapat dipastikan telah diperbaiki atau masih tetap menjadi permasalahan yang berulang.