BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini melakukan analisis sentimen menggunakan SVM dan pemodelan topik menggunakan LDA. Jumlah komentar yang berhasil terambil sebanyak 12220 data. Setelah melalui tahap preprocessing dihasilkan data bersih sebanyak 7752 data. Analisis sentimen menghasilkan nilai *accuracy* sebesar 91% dengan hasil prediksi sentimen netral sebanyak 3383 data, negatif 2276 data, positif 2093 data. Selanjutnya, pemodelan topik diterapkan untuk mengidetifikasi topik utama dari komentar yang diberikan. Topik optimal menghasilkan 13 topik utama dengan nilai coherence score sebesar 0.4853. Topik tertinggi dan terendah pada setiap sentimen terletak pada topik 3 yang terkait dengan keamanan berkendara, serta topik 2 terkait apreasiasi dan keluhan produk. Perbandingan 5 topik tertinggi yaitu pada topik 3, 5, 8, 9, dan 12. Hasil tertinggi kelima topik didominasi oleh sentimen netral, kemudian pada topik 3 dan topik 8 diikuti oleh sentimen negatif, sedangkan pada topik 5, 9, dan 12 diikuti oleh sentimen positif.

4.2 *SCRAPING DATA*

Pengumpulan data dilakukan untuk memperoleh data komentar dari para pengguna TikTok terhadap video yang berkaitan dengan ojek online motor listrik. Pengambilan data dikumpulkan melalui dua metode, yaitu:

1. Menggunakan platform *Apify TikTok Comments Scraper* dengan memasukkan URL video TikTok.
2. Menggunakan *scraping* manual menggunakan bahasa pemrograman Python dengan mengakses API TikTok menggunakan URL video TikTok.

Satu video digunakan dua cara tersebut kemudian dibandingkan untuk dipilih metode mana yang mampu mengumpulkan data komentar dalam jumlah yang lebih banyak. Pengambilan data dilakukan dari tanggal 11 Februari 2025 sampai dengan 16 April 2025 dengan jumlah 12220 data. Link URL didapatkan dengan memasukkan kata kunci seperti "ojol motor listrik", "gojek electric", dan

"pov naik ojol listrik", kemudian memilih video yang relevan dan beragam komentarnya. Tabel 4.1 merupakan daftar URL video TikTok yang dijadikan sumber data.

Tabel 4.1 URL Data

No	URL
1	https://www.tiktok.com/@optikalunett_official/video/7341688982549384454?q=ojol%20elektrik&t=1739155594849
2	https://www.tiktok.com/@evo.iz/video/7386173958967545094?q=ojol%20elektrik&t=1739155594849
3	https://www.tiktok.com/@arifardiansyahs/video/7281213963864509701?q=ojol%20elektrik&t=1739155594849
4	https://www.tiktok.com/@whosjezztin/video/7268951387629817094?q=ojol%20elektrik&t=1739155594849
5	https://www.tiktok.com/@shaguffta_/video/7269713890135543045?q=ojol%20elektrik&t=1739155594849
6	https://www.tiktok.com/@sans.dude/video/7467550967362276614?q=ojol%20elektrik&t=1739155594849
7	https://www.tiktok.com/@yangseringdicari/video/7384398830143294726?q=ojol%20elektrik&t=1739155594849
8	https://www.tiktok.com/@themaplemedia/video/7447479149700091142?q=ojol%20elektrik&t=1739155594849
9	https://www.tiktok.com/@mzojol/video/7229283378057579802?q=ojol%20elektrik&t=1739155594849
10	https://www.tiktok.com/@bisniscom/video/7262200381730589957?q=ojol%20elektrik&t=1739155594849
11	https://www.tiktok.com/@habsiii/video/7123236413633580315?q=ojol%20elektrik&t=1739257055480
12	https://www.tiktok.com/@fandi.begenk/video/7279190599918488837?q=ojol%20elektrik&t=1739257055480
13	https://www.tiktok.com/@efenerr/video/7190565147193232667?q=ojol%20motor%20listrik&t=1739257468329
14	https://www.tiktok.com/@putranainggolan1818/video/7382406632887618821?q=ojol%20motor%20listrik&t=1739257468329
15	https://www.tiktok.com/@justannabella1/video/7137898112550489370?q=ojol%20motor%20listrik&t=1739257468329
16	https://www.tiktok.com/@itsnipipipi/video/7287259545733287173?q=ojol%20motor%20listrik&t=1739257468329
17	https://www.tiktok.com/@sobatericKthohir/video/7068216689216851227?q=ojek%20online%20pakai%20motor%20listrik&t=1739258647214
18	https://www.tiktok.com/@mamahnavp4/video/7177043698700537115?q=ojek%20online%20pakai%20motor%20listrik&t=1739258647214
19	https://www.tiktok.com/@mpokjoe/video/7308981620051135749?q=ojek%20online%20pakai%20motor%20listrik&t=1739258647214
20	https://www.tiktok.com/@augustinus.as/video/7089003619915140379?q=ojek%20online%20pakai%20motor%20listrik&t=1739258647214
21	https://www.tiktok.com/@heyrizc/video/7244825401824939269
22	https://www.tiktok.com/@engkrissss/video/7467174968279698693
23	https://www.tiktok.com/@rahmanperm/video/7216980707527134490
24	https://www.tiktok.com/@onaviona__/video/7446772229779508486

No	URL
25	https://www.tiktok.com/@kopititantebet/video/7152388236013309211
26	https://www.tiktok.com/@emangpacarjungkook/video/7490394870222261522
27	https://www.tiktok.com/@amritsaraje/video/6955077356105075970?q=grab%20motor%20listrik&t=1744762991930

Scraping manual menggunakan bahasa pemrograman Python yang ditunjukkan pada Kode Program 4.1 dengan memasukkan link satu per satu. Alurnya dimulai dengan memasukkan URL video Tiktok, lalu mengambil ID video, mengakses API TikTok, mengambil informasi berupa teks komentar, nama pengguna, dan jumlah suka pada komentar.

Kode Program 4.1 *Scraping* Data

```
1. post_url =
   'https://www.tiktok.com/@amritsaraje/video/69550773561050
   75970'
2. post_id = post_url.split('/')[-1]
3. def req(post_id, curs):
4.    url =
      f'https://www.tiktok.com/api/comment/list/?aid=1988&app
      _name=tiktok_web&aweme_id={post_id}&count=50&cursor={cu
      rs}'
5.    respon = requests.get(url=url, headers=headers)
6.    raw_data = respon.json()
7. return {}
```

Langkah pertama URL video akan disimpan dalam variabel post_url, kemudian digunakan post_id untuk mengambil komentar berdasarkan ID video. Dalam def req(post_id, curs) dibentuk URL untuk mengambil daftar komentar. Variabel respon digunakan untuk mengirim permintaan yang hasil respons dari server diubah ke format JSON dan dikembalikan sebagai data mentah. Dari file JSON ini dikonversi ke format CSV untuk mempermudah proses selanjutnya.

4.3 *PREPROCESSING*

Preprocesing dilakukan untuk membersihkan data agar data siap digunakan untuk dilakukan analisis lebih lanjut. Sebelum *preprocessing* data dari masing-masing URL akan digabungkan menjadi satu file csv agar lebih mudah untuk

pengolahan datanya dan menghapus semua kolom yang hanya menyisakan kolom ‘Comment’. Berikut tahapan *preprocessing* yang dilakukan, diantaranya:

1. *Cleaning*

Tahap ini dilakukan untuk membersihkan data dari angka, emoji, simbol, tanda baca dan lainnya yang tidak memiliki arti. Tahap pembersihan ini ditunjukkan pada Kode Program 4.2.

Kode Program 4.2 *Cleaning*

```
1. def remove_emoji(text):
2.     emot=re.compile("[\U00010000-\U0010ffff]",
       flags=re.UNICODE)
3.     return emot.sub(r'', text)
4. def cleaning(text):
5.     teks = re.sub(r'http\S+|www\S+|@[\w]+|#', '', text)
6.     teks = re.sub(r'[^a-zA-Z0-9\s]', '', text)
7.     teks = re.sub(r'\s+', ' ', text).strip()
8.     teks = remove_emoji(text)
9.     return text
```

Berikut adalah tahapan pembersihan data yang dilakukan:

1. def remove_emoji(text): menghapus emoji dari teks komentar yang mencakup karakter unicode untuk emoji
2. re.sub(r'http\S+|www\S+|@[\w]+|#', '', text): menghapus link, mention, dan hastag.
3. re.sub(r'[^a-zA-Z0-9\s]', '', text): menghapus karakter seperti tanda baca dan karakter lainnya.
4. re.sub(r'\s+', ' ', text).strip(): menghapus spasi berlebih antara teks, spasi di awal dan akhir.
5. remove_emoji(text): memanggil fungsi untuk menghapus emoji

Setelah menghapus hal yang tidak penting dalam proses *cleaning* dilanjutkan dengan cek data kosong dan duplikat data yang ditunjukkan pada kode Program 4.3

Kode Program 4.3 Cek Baris Kosong dan Duplikat

```
1. bariskosong     =     df['Cleaning'].isna().sum()     +
   (df['Cleaning'].str.strip() == "").sum()
```

```
2. print(f"Jumlah baris kosong di kolom: {bariskosong}")
3. duplikat = df.duplicated().sum()
4. print(f"Jumlah baris duplikat: {duplikat}")
```

Kode tersebut berfungsi untuk menghitung berapa jumlah baris kosong pada kolom 'Cleaning'. Baris kosong dihitung dari nilai NaN maupun string kosong. Selain itu, kode ini juga menghitung duplikat pada data. Hasilnya kemudian ditampilkan untuk mengetahui baris kosong dan duplikat. Hasil cek baris kosong dan duplikat dapat dilihat pada Gambar 4.1

```
Jumlah baris kosong di kolom: 912

Jumlah baris duplikat: 718
```

Gambar 4.1 Baris Kosong dan Duplikat.

Hasilnya terdapat 912 baris kosong dan 718 baris duplikat yang harus dihapus. Hasil dari pembersihan data ditunjukkan pada Gambar 4.2.

	Comment	Cleaning
0	malah Drivernya nombok sewa molis kalo orderan...	malah Drivernya nombok sewa molis kalo orderan...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...
2	Waah, tergantung drivernya sih, harusnya kalo ...	Waah tergantung drivernya sih harusnya kalo ud...
3	iyaa tp driver sewa nya lumayan mahal.. sehari...	iyaa tp driver sewa nya lumayan mahal sehari 75 k
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb
...	...	...
12215	Kk masih sepi	Kk masih sepi
12216	gilaa keren banget	gilaa keren banget
12217	sehat2 aa	sehat2 aa
12218	hi kak	hi kak
12219	hai moga di bales	hai moga di bales

Gambar 4.2 Hasil *Cleaning*

2. *Case Folding*

Case Folding dilakukan untuk mengonversi huruf besar menjadi huruf kecil. Proses *case folding* ditunjukkan pada Kode Program 4.3.

Kode Program 4.4 *Case Folding*

```
1. def casefold(text):
2.       return text.lower()
```

Kode program tersebut mengubah setiap teks menjadi huruf kecil pada kolom 'Cleaning' menggunakan method lower(). Hasil *case folding* ditunjukkan pada Gambar 4.3.

	Comment	Cleaning	Case Folding
0	malah Drivernya nombok sewa molis kalo orderan...	malah Drivernya nombok sewa molis kalo orderan...	malah drivernya nombok sewa molis kalo orderan...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...	gue naik grab elektrik malah abis di jalan alh...
2	Waah, tergantung drivernya sih, harusnya kalo ...	Waah tergantung drivernya sih harusnya kalo ud...	waah tergantung drivernya sih harusnya kalo ud...
3	iyaa tp driver sewa nya lumayan mahal.. sehari...	iyaa tp driver sewa nya lumayan mahal sehari 75 k	iyaa tp driver sewa nya lumayan mahal sehari 75 k
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb
...	...	...	...
12215	Kk masih sepi	Kk masih sepi	kk masih sepi
12216	gilaa keren banget	gilaa keren banget	gilaa keren banget
12217	sehat2 aa🍒	sehat2 aa	sehat2 aa
12218	hi kak	hi kak	hi kak
12219	hai moga di bales	hai moga di bales	hai moga di bales

Gambar 4.3 Hasil *Case Folding*

3. Normalisasi

Normalisasi dilakukan untuk mengubah kata tidak baku, kata gaul, singkatan ke dalam bentuk baku sesuai dengan bahasa Indonesia. Untuk mengatasi kata gaul yang sering digunakan dalam sosial media, dalam normalisasi ini menggunakan kamus kata baku yang diperoleh dari repositories github analysisdatasentiment sebanyak 15186 kata dan 13 kata tambahan. Setelah proses normalisasi menggunaan kamus kata baku, dilakukan pengecekan kembali secara manual karena setiap orang memiliki cara penulisan yang berbeda. Bahkan menggunakan kamus kata baku, terkadang masih terdapat kata yang tidak berubah. Contoh kamus kata baku dapat dilihat pada Gambar 4.4.

	tidak_baku	kata_baku
0	woww	wow
1	aminn	amin
2	met	selamat
3	netaas	menetas
4	keberpa	keberapa
...	...	...
15180	pavorit	favorit
15181	nya	ya
15182	lu	kamu
15183	bang	abang
15184	kaka	kakak

Gambar 4.4 Kamus Kata Baku

(*Sumber:* *https://github.com/analysisdatasentiment/kamus_kata_baku/blob/main/kamuskatabaku.xlsx*)

Tahap normalisasi menggunakan kamus kata baku yang berbentuk excel, dimana kolom ’tidak_baku’ berisi kata slang dan kolom ’kata_baku’ berisi padanan kata bakunya. Contoh penggunaannya dapat dilihat pada Kode Program 4.5.

Kode Program 4.5 Normalisasi

```
1.  def Slangword(teks):
2.      slang = re.compile(r'\b(' + '|'.join(re.escape(k) for k
        in kamus_dict.keys()) + r')\b', flags=re.IGNORECASE)
3.      hasil = []
4.      for kalimat in teks:
5.          kalimat_normal      =      slang.sub(lambda      x:
            kamus_dict.get(x.group().lower(),      x.group()),
            kalimat)
6.          hasil.append(kalimat_normal.lower())
7.      return hasil
```

Fungsi Slangword(teks) memanfaatkan regex untuk mencocokkan kata slang lalu menggantinya dengan padanan baku yang terdapat pada kamus_dict. Hasilnya akan disimpan dalam list kosong. Selain itu, pada tahap ini dilakukan tahap menghapus komentar yang tidak berkaitan tentang ojek online motor listrik seperti yang ditunjukkan pada Gambar 4.4. Hasil normalisasi dapat dilihat pada Gambar 4.5.

	Comment	Cleaning	Case Folding
10	Siaaap	Siaaap	siaaap
12	Betul	Betul	betul
18	krl blom bisa bayar pake ovo kya gotransit	krl blom bisa bayar pake ovo kya gotransit	krl blom bisa bayar pake ovo kya gotransit
24	finally chapter🙏	finally chapter	finally chapter
57	ril🤬🤘🤘	ril	ril
...	...	...	...
10584	hai bang	hai bang	hai bang
10585	Kk masih sepi	Kk masih sepi	kk masih sepi
10587	sehat2 aa🍓	sehat2 aa	sehat2 aa
10588	hi kak	hi kak	hi kak
10589	hai moga di bales	hai moga di bales	hai moga di bales

Gambar 4.5 Komentar Tidak Penting

	Comment	Cleaning	Case Folding	Normalisasi
0	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya tambah sewa motor listrik ...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...	gue naik grab elektrik malah abis di jalan alh...	saya naik grab elektrik malah habis di jalan a...
2	Waah, tergantung pengendaranya sih, harusnya k...	Waah tergantung pengendaranya sih harusnya kal...	waah tergantung pengendaranya sih harusnya kal...	wah tergantung pengendaranya sih harusnya kala...
3	iyaa tp pengendara sewa nya lumayan mahal.. se...	iyaa tp pengendara sewa nya lumayan mahal seha...	iyaa tp pengendara sewa nya lumayan mahal seha...	iya tapi pengendara sewanya lumayan mahal seha...
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tapi yang masuk ke kita bukan 11 ribu 10400 ribu

Gambar 4.6 Hasil Normalisasi

4. *Tokenization*

Tokenization dilakukan untuk memecah kalimat menjadi sebuah pecahan-pecahan kata. Proses *tokenization* dapat dilihat pada Kode program 4.6.

Kode Program 4.6 *Tokenization*

```
1. def tokenization(text):
2.       return text.split()
```

Fungsi tokenization dilakukan menggunakan metode split() yang merupakan fungsi untuk memecah string berdasarkan spasi. Hasil *tokenization* dapat dilihat pada Gambar 4.7.

	Comment	Cleaning	Case Folding	Normalisasi	Tokenization
0	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya tambah sewa motor listrik ...	[malah, pengendaranya, tambah, sewa, motor, li...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...	gue naik grab elektrik malah abis di jalan alh...	saya naik grab elektrik malah habis di jalan a...	[saya, naik, grab, elektrik, malah, habis, di,...
2	Waah, tergantung pengendaranya sih, harusnya k...	Waah tergantung pengendaranya sih harusnya kal...	waah tergantung pengendaranya sih harusnya kal...	wah tergantung pengendaranya sih harusnya kala...	[wah, tergantung, pengendaranya, sih, harusnya...
3	iyaa tp pengendara sewa nya lumayan mahal.. se...	iyaa tp pengendara sewa nya lumayan mahal seha...	iyaa tp pengendara sewa nya lumayan mahal seha...	iya tapi pengendara sewanya lumayan mahal seha...	[iya, tapi, pengendara, sewanya, lumayan, maha...
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tapi yang masuk ke kita bukan 11 ribu 10400 ribu	[tapi, yang, masuk, ke, kita, bukan, 11, ribu,...
...	...	...	...	...	...
10555	besok beli😡	besok beli	besok beli	besok beli	[besok, beli]
10557	tungu mobi listerik	tungu mobi listerik	tungu mobi listerik	tunggu mobil listrik	[tunggu, mobil, listrik]
10558	wah jadi ingin nyoba	wah jadi ingin nyoba	wah jadi ingin nyoba	wah jadi ingin coba	[wah, jadi, ingin, coba]

Gambar 4.7 Hasil *Tokenization*

5. *Stopword Removal*

Stopword Removal dilakukan untuk menghapus sebuah kata yang tidak berguna dan tidak memiliki arti seperti dan, di, ke, dengan menggunakan *library* NLTK. Daftar kata *stopword* bahasa indonesia dapat dilihat pada Gambar 4.8.

```
ada, adalah, adanya, adapun, agak, agaknya, agar,
akan, akankah, akhir, akhiri, akhirnya, aku,
akulah, amat, amatlah, anda, andalah, antar,
antara, antaranya, apa, apaan, apabila, apakah,
apalagi, apatah, artinya, asal, asalkan, atas,
atau, ataukah, ataupun, awal, awalnya, bagai,
bagaikan, bagaimana, bagaimanakah, bagaimanapun,
bagi, bagian, bahkan, bahwa, bahwasanya, baik,
bakal, bakalan, balik, banyak, bapak, baru, bawah,
beberapa, begini, beginian, beginikah, beginilah,
begitu, begitukah, begitulah, begitupun, bekerja,
belakang, belakangan, belum, belumlah, benar,
benarkah, benarlah, berada, berakhir, berakhirlah,
berakhirnya, berapa, berapakah, berapalah,
berapapun, berarti, berawal, berbagai, berdatangan,
beri, berikan, berikut, berikutnya, berjumlah,
berkali-kali, berkata, berkehendak, berkeinginan,
berkenaan, berlainan, berlalu, berlangsung,
berlebihan, bermacam, bermacam-macam, bermaksud,
bermula, bersama, bersama-sama, bersiap, bersiap-
```

Gambar 4.8 Daftar *Stopword*

Proses *stopword* dapat dilihat pada Kode Program 4.7.

Kode Program 4.7 *Stopword*

```
1. def hapus_stopwords(text):
2.     stop_words = set(stopwords.words('indonesian'))
3.     stop_words.update(['nya', 'sih', 'eh', 'kakak'])
4.     return [word for word in text if word not in stop_words]
```

Fungsi hapus_stopwords memanfaatkan *library* NLTK untuk mengambil daftar *stopword* dalam Bahasa Indonesia dan diberi beberapa kata tambahan seperti 'nya', 'sih', 'eh', 'kakak'. Tahapan ini dilakukan dengan membuat daftar baru yang memuat kata-kata dari teks yang tidak terdapat dalam kamus *stopword*. Hasil *stopword* dapat dilihat pada Gambar 4.9.

	Comment	Cleaning	Case Folding	Normalisasi	Tokenization	Stopword
0	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya tambah sewa motor listrik ...	[malah, pengendaranya, tambah, sewa, motor, li...	[pengendaranya, sewa, motor, listrik, pesanan,...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...	gue naik grab elektrik malah abis di jalan alh...	saya naik grab elektrik malah habis di jalan a...	[saya, naik, grab, elektrik, malah, habis, di,...	[grab, elektrik, habis, jalan, suruh, turun, j...
2	Waah, tergantung pengendaranya sih, harusnya k...	Waah tergantung pengendaranya sih harusnya kal...	waah tergantung pengendaranya sih harusnya kal...	wah tergantung pengendaranya sih harusnya kala...	[wah, tergantung, pengendaranya, sih, harusnya...	[tergantung, pengendaranya, habis, ambil, tuka...
3	iyaa tp pengendara sewa nya lumayan mahal.. se...	iyaa tp pengendara sewa nya lumayan mahal seha...	iyaa tp pengendara sewa nya lumayan mahal seha...	iya tapi pengendara sewanya lumayan mahal seha...	[iya, tapi, pengendara, sewanya, lumayan, maha...	[iya, pengendara, sewanya, lumayan, mahal, seh...
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tapi yang masuk ke kita bukan 11 ribu 10400 ribu	[tapi, yang, masuk, ke, kita, bukan, 11, ribu,...	[masuk, 11, ribu, 10400, ribu]
...	...	...	...	...	...	...
10555	besok beli😩	besok beli	besok beli	besok beli	[besok, beli]	[besok, beli]
10557	tungu mobi listerik	tungu mobi listerik	tungu mobi listerik	tunggu mobil listrik	[tunggu, mobil, listrik]	[tunggu, mobil, listrik]
10558	wah jadi ingin nyoba	wah jadi ingin nyoba	wah jadi ingin nyoba	wah jadi ingin coba	[wah, jadi, ingin, coba]	[coba]

Gambar 4.9 Hasil *Stopword*

6. *Stemming*

Stemming dilakukan untuk merubah kata ke bentuk kata dasar dengan memanfaatkan *library Sastrawi*. Proses *stemming* ini dapat dilihat pada Kode program 4.8.

Kode Program 4.8 *Stemming*

```
1. def stemming_custom(token_list):
2.     result = []
3.     for word in token_list:
4.         if word in kata_dikecualikan:
5.             result.append(word)
6.         else:
7.             stemmed = stemmer.stem(word)
8.             if stemmed in koreksi_manual:
9.                 stemmed = koreksi_manual[stemmed]
10.            result.append(stemmed)
11.    return ' '.join(result)
```

Fungsi stemming_custom(token_list) menerima input berupa daftar token. Setiap kata yang masuk dalam variabel kata_dikecualikan tidak akan diubah, jika tidak maka akan kembali ke bentuk dasar berdasarkan *library Sastrawi*. Jika hasil *stemming* terdapat dalam variabel koreksi manual, maka hasilnya kan diganti dengan bentuk koresi. Hasil akhirnya digabung kembali menjadi satu kalimat menggunakan return ''.join(result). Hasil stemming dapat dilihat pada Gambar 4.10.

	Comment	Cleaning	Case Folding	Normalisasi	Tokenization	Stopword	Stemming
0	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya nombok sewa molis kalo pes...	malah pengendaranya tambah sewa motor listrik ...	[malah, pengendaranya, tambah, sewa, motor, li...	[pengendaranya, sewa, motor, listrik, pesanan,...	pengendara sewa motor listrik pesanan isi ulan...
1	Gue naik grab elektrik malah abis di jalan, al...	Gue naik grab elektrik malah abis di jalan alh...	gue naik grab elektrik malah abis di jalan alh...	saya naik grab elektrik malah habis di jalan a...	[saya, naik, grab, elektrik, malah, habis, di,...	[grab, elektrik, habis, jalan, suruh, turun, j...	grab elektrik habis jalan suruh turun jalan
2	Waah, tergantung pengendaranya sih, harusnya k...	Waah tergantung pengendaranya sih harusnya kal...	waah tergantung pengendaranya sih harusnya kal...	wah tergantung pengendaranya sih harusnya kala...	[wah, tergantung, pengendaranya, sih, harusnya...	[tergantung, pengendaranya, habis, ambil, tuka...	tergantung pengendara habis ambil tukar batera...
3	iyaa tp pengendara sewa nya lumayan mahal.. se...	iyaa tp pengendara sewa nya lumayan mahal seha...	iyaa tp pengendara sewa nya lumayan mahal seha...	iya tapi pengendara sewanya lumayan mahal seha...	[iya, tapi, pengendara, sewanya, lumayan, maha...	[iya, pengendara, sewanya, lumayan, mahal, seh...	iya pengendara sewa lumayan mahal hari 75 ribu
4	tpi yg masuk ke kita bukan 11 rb.. 10.400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tpi yg masuk ke kita bukan 11 rb 10400 rb	tapi yang masuk ke kita bukan 11 ribu 10400 ribu	[tapi, yang, masuk, ke, kita, bukan, 11, ribu,...	[masuk, 11, ribu, 10400, ribu]	masuk 11 ribu 10400 ribu
...	...	...	...	...	...	...	...
10555	besok beli	besok beli	besok beli	besok beli	[besok, beli]	[besok, beli]	besok beli

Gambar 4.10 Hasil *Stemming*

Setelah tahap *preprocessing* dilakukan, dilakukan kembali pengecekan baris kosong dan duplikat. Proses pengecekan terdapat pada Kode program 4.9.

Kode Program 4.9 Cek Baris Kosong dan Duplikat

```
1. bariskosong = df['Stemming'].isna().sum() +
   (df['Stemming'].str.strip() == "").sum()
2. print(f"Jumlah baris kosong di kolom: {bariskosong}")
3. duplikat = df.duplicated().sum()
4. print(f"Jumlah baris duplikat: {duplikat}")
```

Kode tersebut berfungsi untuk menghitung berapa jumlah baris kosong pada kolom ’Stemming’. Baris kosong dihitung dari nilai NaN maupun string kosong. Selain itu, kode ini juga menghitung duplikat pada data. Hasilnya kemudian ditampilkan untuk mengetahui baris kosong dan duplikat. Hasil cek duplikat dapat dilihat pada Gambar 4.11.

```
Jumlah baris kosong di kolom: 51
Jumlah baris duplikat: 0
```

Gambar 4.11 Hasil Cek Duplikat

Terlihat bahwa terdapat 51 baris kosong yang harus dihapus dan tidak ada duplikat untuk bisa langsung ke tahap selanjutnya yakni *labelling* data. Data hasil preprocessing menyisakan 7752 data untuk dilakukan *labelling*.

4.4 *LABELLING*

Proses *labelling* dilakukan secara otomatis menggunakan bantuan *textblob* untuk memberikan kategori teks ke dalam tiga kategori, yaitu positif, negatif, dan netral. *Textblob* merupakan proses *labelling* yang berbasis bahasa inggris, maka untuk mempermudah proses *labelling* kolom 'Stemming' akan di terjemahkan ke bahasa Inggris menggunakan bantuan rumus excel dan disimpan dalam kolom baru bernama 'TranslateStemming'. Tingkat *accuracy* yang diperoleh dari beberapa metode yang sudah dicoba dapat dilihat pada Tabel 4.2 berikut ini:

Tabel 4.2 Hasil *Accuracy*

Metode	*Accuracy* Train	*Accuracy* Test
Lexicon Based	0.92	0.8488
Vader NLTK	0.9327	0.8466
Vader Sentiment	0.9363	0.8444
Textblob	0.9508	0.9133

Labelling manual tidak digunakan karena prosesnya memakan waktu dan tenaga karena setiap baris harus dianalisis dan diberi label satu per satu. *Textblob* dipilih karena memiliki *accuracy* tertinggi dibandingkan dengan metode lainnya. Proses *labelling* menggunakan *textblob* dapat dilihat pada Kode program 4.10.

Kode Program 4.10 *Labelling*

```
1. def text_blob(text):
2.     text = str(text).lower()
3.     for kata in kata_kasar:
4.         if kata in text:
5.           return 'negatif'
6.     pola = TextBlob(text).sentiment.polarity
7.     if pola > 0:
8.         return 'positif'
9.     elif pola < 0:
10.        return 'negatif'
```

```
    else:
        return 'netral'
jumlah_per_kelas = min(1500, df["sentimen"].value_counts().min())
```

Fungsi text_blob (text) digunakan untuk memberi label pada setiap baris teks. Jika baris teks mengandung kata dari salah satu variabel kata_kasar, maka secara otomatis akan diberi label negatif. Jika tidak, maka akan dihitung *polarity* menggunakan *library textblob*. Nilai *polarity* diatas 0 diberi label positif, nilai negatif atau dibawah 0 diberi label negatif, dan nilai 0 diberi label netral. Setelah seluruh data diberi label, diambil 1500 data setiap sentimen secara acak dengan jumlah data yang seimbang. Hasil *labelling* dapat dilihat pada Gambar 4.12.

	Comment	Cleaning	Case Folding	Normalisasi	Tokenization	Stopword	Stemming	TranslateStemming	sentimen
0	gw pernah dapat pengendara yg pake motor listr...	gw pernah dapat pengendara yg pake motor listr...	gw pernah dapat pengendara yg pake motor listr...	saya pernah dapat pengendara yang pakai motor ...	['saya', 'pernah', 'dapat', 'pengendara', 'yan...	['pengendara', 'pakai', 'motor', 'listrik', 'i...	pengendara pakai motor listrik izin izin pegan...	riders using electric motorbikes permission is...	negatif
1	@adamkusumaaja kek motor mu bukan si bang??	kek motor mu bukan si bang	kek motor mu bukan si bang	seperti motor kamu bukan sih kakak	['seperti', 'motor', 'kamu', 'bukan', 'sih', '...	['motor']	motor	motor	netral
2	motor kematian motor listrik sama motor aerox ...	motor kematian motor listrik sama motor aerox ...	motor kematian motor listrik sama motor aerox ...	motor kematian motor listrik sama motor aerox ...	['motor', 'kematian', 'motor', 'listrik', 'sam...	['motor', 'kematian', 'motor', 'listrik', 'mot...	motor mati motor listrik motor aerox pegang	dead motor electric motor aerox motor hold	negatif
3	pas uda masuk gang2 dkt kost hening bgt jirrrr...	pas uda masuk gang2 dkt kost hening bgt jirrrr	pas uda masuk gang2 dkt kost hening bgt jirrrr	saat sudah masuk gang dekat kos hening banget ...	['saat', 'sudah', 'masuk', 'gang', 'dekat', 'k...	['masuk', 'gang', 'kos', 'hening', 'banget', '...	masuk gang kos hening banget anjing	entering the boarding alley the dog is very quiet	negatif
4	lewatin tikungan yang kanan kiri jalannya ketu...	lewatin tikungan yang kanan kiri jalannya ketu...	lewatin tikungan yang kanan kiri jalannya ketu...	lewatin tikungan yang kanan kiri jalannya ketu...	['lewatin', 'tikungan', 'yang', 'kanan', 'kiri...	['lewatin', 'tikungan', 'kanan', 'kiri', 'jala...	lewatin tikung kanan kiri jalan tutup lihat gi...	pass the right bend left road closed see gigih...	positif

Gambar 4.12 Hasil *Labelling*

4.5 TF-IDF

Setelah tahap *labelling* data yang digunakan untuk bangun model harus diubah menjadi representasi numerik terlebih dahulu dengan salah satu cara menggunakan TF-IDF. Terdapat lima dokumen yang digunakan sebagai contoh perhitungan manual yang dapat dilihat pada Tabel 4.3.

Tabel 4.3 Contoh Dokumen TF-IDF

Dokumen	*Comment*
D1	tahu mah baterai habis beli baterai tinggal cari stasiun motor listrik tinggal ganti
D2	motor
D3	banding motor listrik sewa iya susah beli bandingin

D4	sepeda listrik iya suara klakson takut tabrak orang
D5	ojek online motor listrik ketawa melulu motor suara

Lima dokumen tersebut akan dihitung TF terlebih dahulu. TF bertujuan untuk menghitung seberapa sering sebuah *term*/kata-kata muncul dalam dokumen. Setiap kata dalam dokumen akan dipecah menjadi *term*/kata-kata, kemudian dihitung jumlah total *term*/kata-kata dalam masing-masing dokumen. Setelah itu, nilai TF untuk setiap *term*/kata-kata dihitung dengan cara membagi jumlah kemunculan sebuah kata dengan keseluruhan jumlah kata di masing-masing dokumen/komentar. Contoh perhitungan TF dapat dilihat pada Tabel 4.4.

Tabel 4.4 Contoh Perhitungan TF

Term	**D1**	**D2**	**D3**	**D4**	**D5**
baterai	0.153846154				
banding			0.125		
bandingin			0.125		
beli	0.076923077		0.125		
cari	0.076923077				
ganti	0.076923077				
habis	0.076923077				
iya			0.125	0.125	
ketawa					0.125
klakson				0.125	
listrik	0.076923077		0.125	0.125	0.125
mah	0.076923077				
melulu					0.125
motor	0.076923077	1	0.125		0.25
ojek					0.125
online					0.125
orang				0.125	
sewa			0.125		
sepeda				0.125	
stasiun	0.076923077				
suara					0.125
susah			0.125		
tabrak				0.125	
tahu	0.076923077				
takut				0.125	

tinggal	0.153846154				

Setelah perhitungan TF, langkah selanjutnya menghitung IDF untuk setiap *term*. IDF bertujuan untuk menilai seberapa penting *term* dalam dokumen. Kata yang sering muncul dibanyak dokumen akan menghasilkan nilai IDF yang rendah. Perhitungan IDF diperoleh dengan membagi total jumlah dokumen dengan jumlah dokumen yang mengandung *term*, kemudian diambil logaritma basis 10. Contoh perhitungan IDF dapat dilihat pada Tabel 4.5.

Tabel 4.5 Contoh perhitungan IDF

Term	df(t)	IDF
baterai	1	0.69897
banding	1	0.69897
bandingin	1	0.69897
beli	2	0.39794
cari	1	0.69897
ganti	1	0.69897
habis	1	0.69897
iya	2	0.39794
ketawa	1	0.69897
klakson	1	0.69897
listrik	4	0.09691
mah	1	0.69897
melulu	1	0.69897
motor	4	0.09691
ojek	1	0.69897
online	1	0.69897
orang	1	0.69897
sewa	1	0.69897
sepeda	1	0.69897
stasiun	1	0.69897
suara	1	0.69897
susah	1	0.69897
tabrak	1	0.69897
tahu	1	0.69897
takut	1	0.69897
tinggal	1	0.69897

Setelah perghitungan TF dan ID, dilanjutkan dengan menghitung nilai TF-IDF untuk masing-masing *term* yang terdapat dalam dokumen. Nilai TF-IDF diperoleh

dengan mengalikan nilai TF dengan nilai IDF untuk masing-masing *term*. Nilai yang tinggi menunjukkan bahwa *term* tersebut memiliki tingkat kepentingan yang besar dalam dokumen tersebut dan jarang ditemukan di dokumen lain. Contoh perhitungan TF-IDF dapat dilihat pada Tabel 4.6.

Tabel 4.6 Contoh Perhitungan TF-IDF

Term	**D1**	**D2**	**D3**	**D4**	**D5**
baterai	0.107534	0	0	0	0
banding	0	0	0.087371	0	0
bandingin	0	0	0.087371	0	0
beli	0.030611	0	0.049743	0	0
cari	0.053767	0	0	0	0
ganti	0.053767	0	0	0	0
habis	0.053767	0	0	0	0
iya	0	0	0.049743	0.049743	0
ketawa	0	0	0	0	0.087371
klakson	0	0	0	0.087371	0
listrik	0.007455	0	0.012114	0.012114	0.012114
mah	0.053767	0	0	0	0
melulu	0	0	0	0	0.087371
motor	0.007455	0.09691	0.012114	0	0.024228
ojek	0	0	0	0	0.087371
online	0	0	0	0	0.087371
orang	0	0	0	0.087371	0
sewa	0	0	0.087371	0	0
sepeda	0	0	0	0.087371	0
stasiun	0.053767	0	0	0	0
suara	0	0	0	0	0.087371
susah	0	0	0.087371	0	0
tabrak	0	0	0	0.087371	0
tahu	0.053767	0	0	0	0
takut	0	0	0	0.087371	0
tinggal	0.107534	0	0	0	0

Perhitungan TF-IDF pada penelitian ini dilakukan secara otomatis menggunakan TfidfVectorizer(). Berikut proses perhitungan TF-IDF dapat dilihat pada Kode Program 4.11.

Kode Program 4.11 TF-IDF

```
1. vectorizer=TfidfVectorizer(max_features=5000, min_df=3,
   max_df=0.90, sublinear_tf=True)
2. X=vectorizer.fit_transform(df_seimbang["TranslateStemmi
   ng"])
3. y = df_seimbang["sentimen"]
```

Program tersebut mengubah data teks dari kolom ’TranslateStemming’ menjadi vektor numerik menggunakan TF-IDF melalui TfidfVectorizer(). Parameter yang digunakan antara lain, max_features menggunakan 5000 kata terpenting sebagai fitur, min_df mengabaikan kata yang mucul kurang dari 3 dokumen, max_df mengecualikan kata yang muncul lebih dari 90% dokumen, dan sublinear_tf agar frekuensi kata yang tinggi tidak terlalu mendominasi. Selanjutnya mengubah kolom ’TranslateStemming’ ke bentuk numerik yang disimpan dalam variabel X. Selanjutnya kolom sentimen disimpan dalam variabel y. Hasil TF-IDF dapat dilihat pada Gambar 4.13.

	10	100	12	125	140	15	150	18	20	2020	...	x1	yeah	yellow	yes	yesterday	yesterdays	you	young	your	youre
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.280977	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.226149	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4496	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.426095	0.0	0.0	0.0	0.0	0.0	0.0
4498	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0

Gambar 4.13 Hasil TF-IDF

4.6 KLASIFIKASI SENTIMEN SVM

Setelah melakukan ekstrasi fitur dengan TF-IDF dilanjutkan ke tahap pembangunan model analisis sentimen dengan SVM. Proses pembuatan model SVM dapat dilihat pada Kode Program 4.12.

Kode Program 4.12 Model SVM

```
1. X_train, X_test, y_train, y_test = train_test_split(X,
   y, test_size=0.2, random_state=42, stratify=y)
2. model_svm = SVC(kernel='linear')
3. model_svm.fit(X_train, y_train)
```

```
4. y_pred_train = model_svm.predict(X_train)
5. y_pred_test = model_svm.predict(X_test)
```

Penjelasan untuk Kode Program 4.12 sebagai berikut:

1. Variabel X dan y dipisahkan menjadi data pelatihan dan data pengujian dengan memanfaatkan train_test_split dengan pembagian 8:20. stratify=y memastikan bahwa distribusi kelas tetap seimbang pada data pelatihan dan data pengujian.
2. model_svm membuat model menggunakan SVC() dari library scikit-learn dengan beberapa parameter yakni, kernel='linear' yang menggunakan garis lurus (hyperplane) sebagai batas keputusan untuk memisahkan kelas.
3. model_svm.fit(X_train, y_train) digunakan untuk melatih model dengan menggunakan fit() terhadap data latih X_train, y_train
4. y_pred_train untuk menyimpan hasil prediksi pada latih untuk mengetahui seberapa baik model mengenali data.
5. y_pred_test untuk menyimpan model yang digunakan untuk memprediksi data uji.

4.7 EVALUASI MODEL SVM

Setelah pembuatan model SVM dilanjutkan dengan evaluasi model untuk mengetahui kualitas dan keakuratan model yang telah dibuat. Evaluasi model ditunjukkan pada Kode Program 4.13.

Kode Program 4.13 Evaluasi SVM

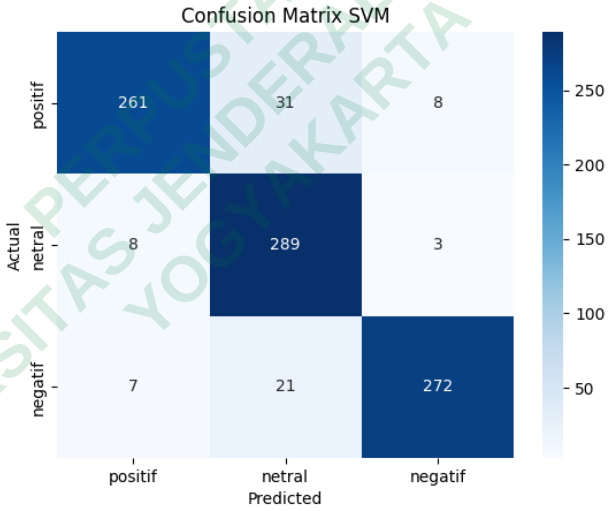
```
1. print("\nClassification
   Report:\n",classification_report(y_test, y_pred_test))
2. conf_matrix = confusion_matrix(y_test, y_pred_test,
   labels=["positif", "netral", "negatif"])
3. sns.heatmap(conf_matrix,annot=True,fmt="d",cmap="Blues"
   ,xticklabels=["positif","netral","negatif"],yticklabels
   =["positif", "netral", "negatif"])
4. plt.xlabel("Predicted")
5. plt.ylabel("Actual")
6. plt.title("Confusion Matrix SVM")
7. plt.show()
```

Kode tersebut untuk menampilkan evaluasi model klasifikasi SVM dengan 2 cara. Baris pertama digunakan untuk menampilkan classification report yang didalamnya terdapat nilai *precision*, *recall*, *F1-score* untuk masing-masing kategori sentimen. Baris selanjutnya digunakan untuk visualisasi *confusion matrix* dalam bentuk *heatmap* untuk mengetahui jumlah prediksi yang tepat dan keliru pada masing-masing kategori sentimen. Hasil evaluasi model dapat dilihat pada Gambar 4.14 dan Gambar 4.15.

Classification Report:

	precision	recall	f1-score	support
negatif	0.96	0.91	0.93	300
netral	0.85	0.96	0.90	300
positif	0.95	0.87	0.91	300
accuracy			0.91	900
macro avg	0.92	0.91	0.91	900
weighted avg	0.92	0.91	0.91	900

Gambar 4.14 *Classification Report*



Gambar 4.15 *Confusion Matrix*

Pada Gambar 4.15 menunjukkan bahwa *accuracy* pada data test yaitu 0.91 atau 91% dari total data uji berhasil diklasifikasikan dengan benar, yang menandakan bahwa model memiliki kemampuan klasifikasi yang baik. Selain itu nilai *precision*, *recall*, dan *F1-score* pada masing-masing kelas sentimen tergolong cukup tinggi dan seimbang. Gambar 4.15 terlihat bahwa sebagian data uji berhasil diklasifikasikan dengan benar, yaitu 260 komentar positif, 289 komentar netral, dan

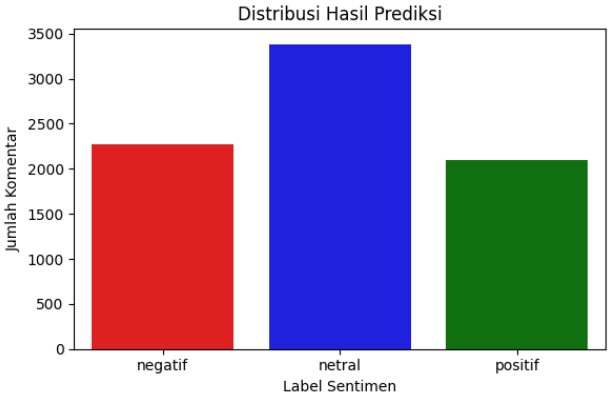
272 komentar negatif. Meskipun demikian, masih terdapat kesalahan klasifikasi, seperti komentar positif yang diprediksi sebagai netral, dll.

Setelah model SVM dibuat selanjutnya melakukan prediksi model menggunakan seluruh data yakni 7752 data. Proses prediksi model SVM ditunjukkan pada Kode program 4.14.

Kode Program 4.14 Prediksi SVM

```
1. prediksi = model_svm.predict(X)
2. df['prediksi_sentimen'] = prediksi
3. sentimen                                                    =
   df['prediksi_sentimen'].value_counts().sort_index()
4. colors = ['red', 'blue', 'green']
5. plt.figure(figsize=(6,4))
6. sns.barplot(x=sentimen.index,          y=sentimen.values,
   palette=colors)
7. plt.title('Distribusi Hasil Prediksi')
8. plt.xlabel('Label Sentimen')
9. plt.ylabel('Jumlah Komentar')
10.      plt.xticks(rotation=0)
11.      plt.tight_layout()
12.      plt.show()
```


Pada Kode Program diatas variabel prediksi digunakan untuk memanggil model SVM menggunakan model_svm untuk melalukan prediksi pada data baru pada variabel X. Variabel df['prediksi_sentimen'] digunakan untuk menyimpan hasil prediksi pada kolom baru. Selanjutnya, variabel sentimen digunakan menghitung jumlah masing-masing label sentimen dan diurutkan berdasarkan label. Visualisasi dilakukan dengan membuat grafik batang menggunakan *seaborn*, dimana sumbu x untuk kategori sentimen dan sumbu y untuk menunjukkan jumlah komentar yang diprediksi. Warna batang diatur dengan warna berbeda pada variabel colors untuk membedakan tiap label sentimen. Hasil prediksi ditampilkan pada distribusi sentimen pada Gambar 4.16.



Gambar 4.16 Distribusi Prediksi Sentimen

Berdasarkan Gambar 4.16, distribusi prediksi sentimen menunjukkan bahwa 3383 komentar tergolong dalam sentimen netral, 2276 tergolong sentimen negatif, dan 2093 tergolong sentimen positif.

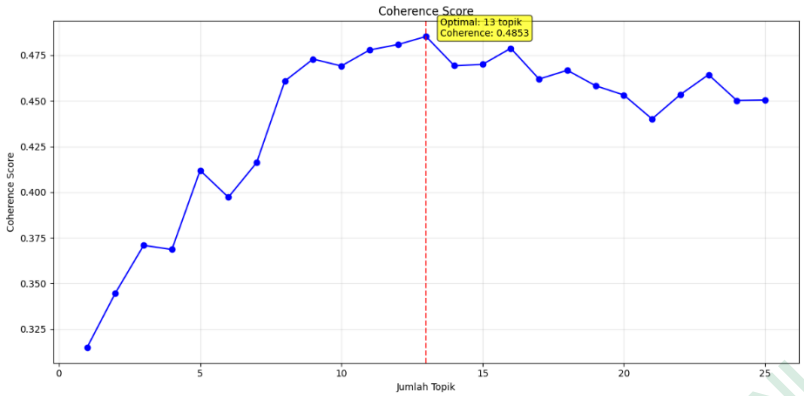
4.8 JUMLAH TOPIK LDA

Dalam pemodelan topik langkah pertama menentukan jumlah topik optimal untuk pembuatan model LDA. Pada penelitian ini penentuan jumlah topik optimal menggunakan *coherence score*. Prosesnya dilakukan dengan mencoba beberapa topik dari rentang 1 sampai 25 topik. Selanjutnya topik dengan nilai *coherence* paling tinggi dianggap yang paling optimal. Proses penentuan *coherence score* dapat dilihat pada Kode Program 4.15.

Kode Program 4.15 Topik LDA

```
1. max_coherence                                                =
   coherence_values.index(max(coherence_values))
2. optimal_topics = x[max_coherence]
```

Kode tersebut digunakan untuk menentukan jumlah topik optimal dalam pemodelan topik. Pertama, akan mencari nilai coherence tertinggi dari daftar *coherence score* yang telah dibuat pada coherence_values, lalu mengambil indeks dari nilai tertinggi tersebut. Indeks ini kemudian digunakan untuk mengambil jumlah topik tertinggi dari 1 hingga 25. Hasilnya digunakan untuk menampilkan grafik *coherence score* seperti pada Gambar 4.17.



Gambar 4.17 Grafik *Coherence Score*

Nilai *coherence score* untuk setiap topik dapat dilihat pada Tabel 4.7.

Tabel 4.7 Nilai *Coherence Score*

Topik	Coherence Score
1	0.3148847467
2	0.3446487263
3	0.3707838587
4	0.3685143207
5	0.4118912006
6	0.3971657179
7	0.4161199964
8	0.4608864033
9	0.4729092996
10	0.4691211572
11	0.4779144229
12	0.4808605241
13	0.4853033714
14	0.4692429913
15	0.4699866514
16	0.4788299523
17	0.461950526
18	0.4667634946
19	0.4583093598
20	0.4532169889
21	0.4400964474
22	0.4534116889
23	0.4643601177
24	0.4502232128
25	0.4504439129

Berdasarkan Gambar 4.17 dan Tabel 4.7 yang menunjukkan nilai *coherence score* dari jumlah topik 1 sampai 25, terlihat bahwa nilai *coherence score* cenderung mengalami peningkatan seiring dengan bertambahnya jumlah topik hingga mencapai titik tertinggi pada 13 topik dengan nilai *coherence* 0.4853. Setelah topik 13, nilai *coherence score* mulai mengalami penurunan, meskipun ada beberapa kenaikan kecil. Hal ini menunjukkan bahwa menambah jumlah topik lebih dari 13 akan beresiko menurunkan kualitas model. Dengan demikian, jumlah topik optimal yang dipilih untuk pemodelan ini adalah sebanyak 13 topik.

4.9 PEMODELAN TOPIK

Langkah selanjutnya adalah membuat model dengan menerapkan topik optimal yang sudah ditentukan, yaitu sebanyak 13 topik. Proses model LDA dapat dilihat pada Kode Program 4.16.

Kode Program 4.16. Model LDA

```
1. best_model = model_list[max_coherence]
2. model_lda = LdaModel(corpus=corpus, id2word=dictionary,
   num_topics=optimal_topics,              random_state=42,
   passes=20,                alpha='auto',     eta='auto',
   per_word_topics=True,   chunksize=100,   eval_every=None,
   minimum_probability=0.01)
```

Pada kode program tersebut nilai coherence tertinggi yaitu 13 topik akan disimpan dalam best_model. Variabel model_lda digunakan untuk melatih model dengan beberapa parameter, yakni:

1. corpus=corpus, merupakan data teks yang sudah diubah ke format BOW.
2. id2word=dictionary, kata unik dalam corpus.
3. num_topics=optimal_topics, jumlah topik dengan coherence score tertinggi.
4. random_state=42, menjaga kestabilan model.
5. passes=20, jumlah pengulangan untuk membaca corpus.
6. alpha='auto', eta='auto', model menyesuaikan distribusi topik dan kata secara otomatis.
7. per_word_topics=True, menyimpan distribusi topik untuk setiap kata.

8. chunksize=100, jumlah dokumen yang diproses sekaligus.
9. eval_every=None, tidak melakukan evaluasi coherence selama pelatihan.
10. minimum_probability=0.01, hanya menampikan topik yang memiliki probabilitas di atas 1%.

Setelah model dibangun, langkah selanjutnya adalah menampilkan daftar topik beserta kata-kata yang kerap muncul dapat dilihat pada Kode Program 4.17.

Kode Program 4.17 Daftar Topik

```
1. for idx in range(model_lda.num_topics):
2.     print(f"\nTopik {idx+1}:")
3.     print(f"   { model_lda.print_topic(idx)}")
```

Kode tersebut digunakan untuk menampilkan daftar topik yang dihasilkan oleh model LDA. Dalam perulangan, idx merepresentasikan nomor indeks dari setiap topik yang dimulai dari 1 hingga 13. Pada setiap iterasi akan menampilkan kata-kata kunci beserta bobotnya dari masing-masing topik tersebut. Kata-kata ini membantu dalam memahami isi dari masing-masing topik yang ditentukan model LDA. Hasil topik dapat dilihat pada Tabel 4.8.

Tabel 4.8 Kata kunci Topik

Topik	Kata Kunci	Analisis Topik
1	0.323*"gojek" + 0.051*"buru" + 0.051*"kencang" + 0.046*"cepat" + 0.040*"kena" + 0.033*"onlinenya" + 0.033*"mobil" + 0.027*"pulang" + 0.026*"malam" + 0.023*"cocok"	Kecepatan
2	0.249*"bagus" + 0.152*"allah" + 0.103*"aduh" + 0.085*"saking" + 0.082*"nanya" + 0.043*"pabrik" + 0.041*"keluh" + 0.030*"gesits" + 0.005*"keringat" + 0.003*"tempur"	Apresiasi dan keluhan produk.
3	0.203*"terjungkal" + 0.107*"suara" + 0.075*"gas" + 0.067*"bawa" + 0.050*"orang" + 0.045*"langsung" +	Keamanan berkendara

Topik	Kata Kunci	Analisis Topik
	0.042*"biar" + 0.039*"ngegas" + 0.027*"teman" + 0.023*"kasih"	
4	0.341*"anjing" + 0.090*"mahal" + 0.073*"hening" + 0.055*"kebut" + 0.049*"kasihan" + 0.046*"main" + 0.043*"batal" + 0.034*"masuk" + 0.029*"serius" + 0.026*"lumayan"	Harga sewa
5	0.124*"grab" + 0.073*"sewa" + 0.065*"pesanan" + 0.057*"ribu" + 0.051*"viar" + 0.039*"jam" + 0.037*"coba" + 0.023*"ganti" + 0.022*"mati" + 0.020*"asap"	Kendala operasional
6	0.283*"jalan" + 0.160*"beli" + 0.073*"tolong" + 0.056*"bensin" + 0.055*"pelan" + 0.037*"sepi" + 0.033*"tabrak" + 0.021*"kota" + 0.021*"bayangin" + 0.020*"salah"	Keselamatan jalan
7	0.158*"sumpah" + 0.093*"jaket" + 0.083*"lampu" + 0.080*"bilang" + 0.059*"merah" + 0.052*"ketawa" + 0.045*"deh" + 0.042*"dengar" + 0.038*"tahan" + 0.033*"posisi"	Reaksi pelanggan
8	0.125*"ojek" + 0.103*"pegang" + 0.086*"online" + 0.058*"duduk" + 0.053*"berasa" + 0.045*"rumah" + 0.032*"kemarin" + 0.032*"untung" + 0.027*"tidur" + 0.026*"polisi"	Kenyamanan pelanggan
9	0.165*"pakai" + 0.082*"baterai" + 0.072*"isi" + 0.061*"daya" + 0.058*"jatuh" + 0.057*"harga" + 0.037*"habis" + 0.033*"indonesia" + 0.029*"gimana" + 0.023*"turun"	Infrastruktur
10	0.281*"keren" + 0.184*"kali" + 0.060*"jakarta" + 0.060*"polusi" +	Dampak lingkungan

Topik	Kata Kunci	Analisis Topik
	0.035*"canggung" + 0.034*"ujung" + 0.031*"ringan" + 0.028*"pengalaman" + 0.027*"medan" + 0.024*"tenaga"	
11	0.100*"bikin" + 0.070*"sempit" + 0.065*"anak" + 0.062*"elektrik" + 0.059*"ramah" + 0.057*"lingkungan" + 0.055*"badan" + 0.054*"bali" + 0.040*"penumpang" + 0.027*"lambat"	Desain produk
12	0.109*"jok" + 0.092*"knalpot" + 0.078*"takut" + 0.072*"pengendara" + 0.070*"kaget" + 0.068*"suka" + 0.065*"enak" + 0.045*"lucu" + 0.041*"sepeda" + 0.039*"hujan"	Kepuasan pelanggan
13	0.494*"banget" + 0.075*"bonceng" + 0.065*"lihat" + 0.046*"jogja" + 0.033*"muat" + 0.032*"malu" + 0.020*"refleks" + 0.018*"kerja" + 0.017*"nyalip" + 0.016*"modifikasi"	Penilaian masyarakat

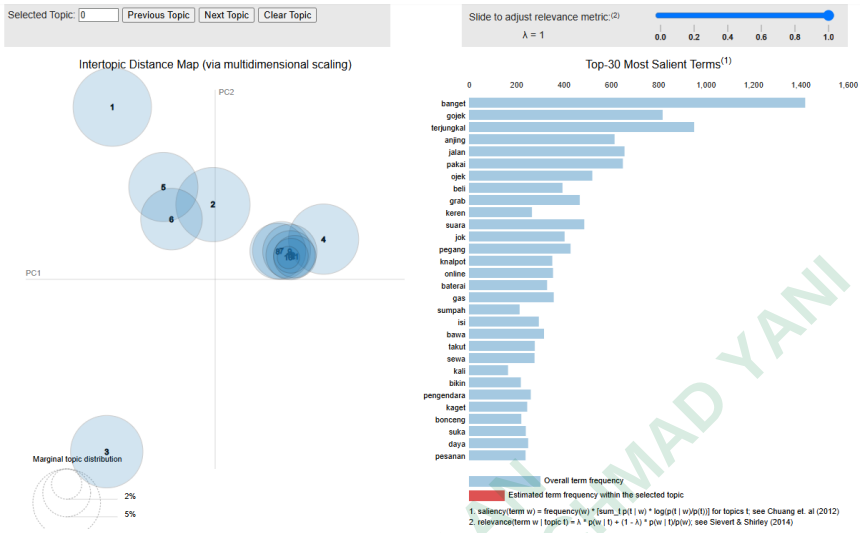
Untuk mengetahui keterkaitan antar topik digunakan visualisasi pendukung menggunakan *pyLDAvis* yang ditunjukkan pada Kode Program 4.17

Kode Program 4.18 Visualisasi Topik

```
1. pyLDAvis.enable_notebook()
2. p = gensimvis.prepare(best_model, corpus, dictionary)
3. p
```

Kode tersebut digunakan untuk membuat visualisasi dalam bentuk grafik interaktif. Kode pyLDAvis.enable_notebook() digunakan untuk mengaktifkan tampilan visualisasi langsung dalam notebook. Variabel p digunakan untuk menyimpan visualisasi dengan memasukkan model LDA, korpus data, dan kamus sebagai input. Nilai yang tampilkan oleh p adalah visualisasi yang menampilkan distribusi topik,

hubungan antar topik, dan kata kunci penting di setiap topik. Hasil Visualisasi dapat dilihat pada Gambar 4.18.



Gambar 4.18 Visualisasi *pyLDAvis*

Berdasarkan visualisasi pada Gambar 4.18 terlihat bahwa topik 1 memiliki lingkaran paling besar yang berarti sering muncul dalam dokumen, walaupun cuma sedikit kata. Topik 1, dan 3 agak terpisah dari yang lain berarti topik tersebut memiliki kata unik yang berbeda dari topik lain. Topik yang saling bertumpuk cenderung memiliki kemiripan antar kata-kata, seperti pada topik 10 dan 11. Meskipun saling bertumpuk, kata kunci yang muncul berbeda sehingga analisis dilakukan secara terpisah berdasarkan topik

Setelah hasil kata dari setiap topik ditampilkan, selanjutnya topik tersebut diimplementasikan ke masing-masing komentar. Proses ini dilakukan dengan menggabungkan topik dominan ke dalam data komentar, sehingga setiap komentar memiliki dominan topik, probabilitas topik, dan daftar kata yang merepresentasikan topik tersebut. Proses yang dilakukan dapat dilihat pada Kode Program 4.19.

Kode Program 4.19 Topik Sentimen

```
1. def topic_dominan(ldamodel, corpus, texts):
2.     topics_data = []
3.     for i, doc in enumerate(corpus):
4.         distribusi_topic = ldamodel.get_document_topics(doc,
           minimum_probability=0.0)
5.         dominan_topik = max(distribusi_topic, key=lambda x: x[1])
```

```
6.          topic_num, topic_prob = dominan_topik
7.          keyword_topik = ldamodel.show_topic(topic_num, topn=10)
8.          keyword = '', ''.join([word for word, prob in keyword_topik])
9.          topics_data.append({'Dominant_Topic': topic_num+1,
            'Topic_Probability': topic_prob, 'Topic_Keywords': keyword})
10. return pd.DataFrame(topics_data)
```

Pada kode program tersebut fungsi topic_dominan() untuk setiap dokumen pada corpus model akan menghitung distribusi topik menggunakan variabel distribusi_topic yang akan menghasilkan daftar probabilitas dokumen dengan masing-masing topik. Topik dengan probabilitas tertinggi dipilih menggunakan max() sebagai topik dominan. Sepuluh kata kunci teratas dari topik tersebut juga akan ditampilkan menggunakan variabel keyword_topik. Setelah itu akan disimpan ke dalam dataframe dan digabungkan dengan dataframe yang berisi hasil prediksi sentimen. Penggabungan dilakukan dengan mengambil kolom 'Stemming' dan 'prediksi_sentimen' pada dataframe hasil prediksi. Hasil penggabungan tersebut dapat dilihat pada Gambar 4.19.

	Stemming	prediksi_sentimen	Dominant_Topic	Topic_Probability	Topic_Keywords
0	pengendara sewa motor listrik pesanan isi ulan...	netral	5	0.254707	grab, sewa, pesanan, ribu, viar, jam, coba, ga...
1	grab elektrik habis jalan suruh turun jalan	negatif	9	0.249238	pakai, baterai, isi, daya, jatuh, harga, habis...
2	tergantung pengendara habis ambil tukar batera...	netral	9	0.200048	pakai, baterai, isi, daya, jatuh, harga, habis...
3	iya pengendara sewa lumayan mahal hari 75 ribu	negatif	5	0.214345	grab, sewa, pesanan, ribu, viar, jam, coba, ga...
4	masuk 11 ribu 10400 ribu	netral	5	0.235458	grab, sewa, pesanan, ribu, viar, jam, coba, ga...
5	setuju ayo pakai grab elektrik pengendara biar...	netral	13	0.198382	banget, bonceng, lihat, jogja, muat, malu, ref...
6	kasihan penumpang mogok lambat kantor elektrik...	negatif	11	0.382472	bikin, sempit, anak, elektrik, ramah, lingkung...
7	motor bunyi muncul	positif	3	0.290951	terjungkal, suara, gas, bawa, orang, langsung,...
8	iya klakson	netral	3	0.124371	terjungkal, suara, gas, bawa, orang, langsung,...
9	grab ragu laju kencang cepat tujuan	positif	1	0.327670	gojek, buru, kencang, cepat, kena, onlinenya, ...

Gambar 4.19 Prediksi Topik Komentar

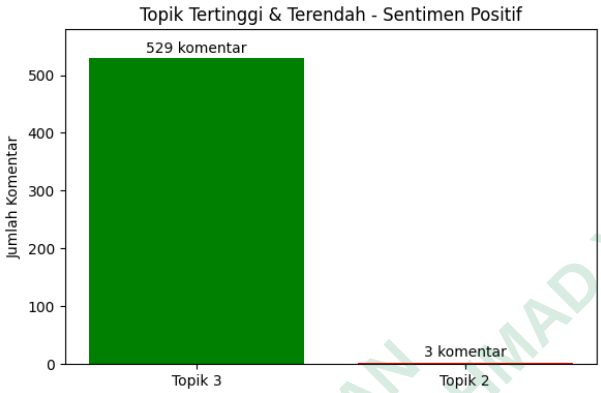
4.10 ANALISIS HASIL

Setelah topik optimal sudah ditentukan dan setiap komentar sudah memiliki topik, langkah selanjutnya adalah visualisasi dan analisis hasil pemodelan topik berdasarkan sentimen. Terdapat dua tahap analisis, yakni analisis topik setiap sentimen dan analisis persepsi setiap topik.

4.10.1 Analisis Topik Setiap Sentimen

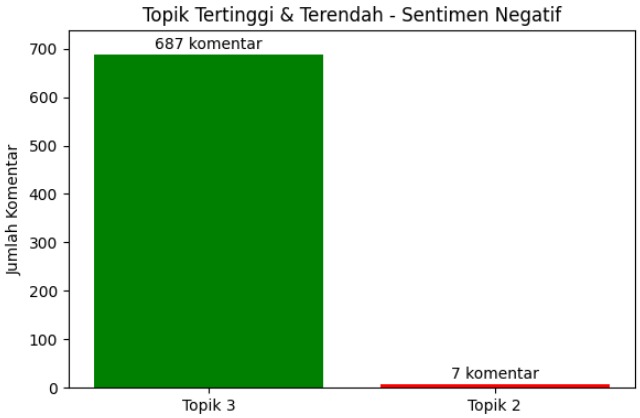
Setelah semua dokumen mendapatkan topik, selanjutnya dilakukan visualisasi untuk melihat distribusi topik berdasarkan sentimen untuk melihat topik

tertinggi pada setiap sentimen. Selanjutnya dilakukan analisis guna memahami topik-topik tertinggi dan terendah yang muncul pada setiap kategori sentimen. Berikut distribusi tertinggi dan terendah pada setiap sentimen yang dapat dilihat pada Gambar 4.20, 4.21, 4.22.



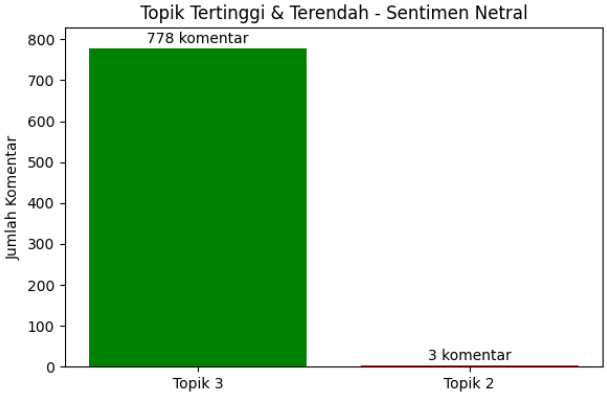
Gambar 4.20 Topik Tertinggi dan Terendah Positif

Berdasarkan Gambar 4.20, topik dengan jumlah komentar tertinggi berada di topik 3 dengan 529 komentar, sedangkan topik dengan komentar terendah berada di topik 2 dengan 3 komentar. Pada topik 3 terdapat kata kunci ”terjungkal, suara, gas” yang mengindikasikan bahwa topik ini membahas tentang keamanan dalam berkendara. Hal ini menunjukan bahwa masyarakat masih beradaptasi terhadap motor yang hening tanpa suara. Meskipun membuat beberapa orang terkejut, kondisi ini juga mencerminkan kelebihan motor listrik yang hening untuk mengurangi polusi suara, khusunya didaerah perkotaan. Sementara itu, topik 2 topik terendah terdapat kata ”bagus, gesits, tempur” yang menginterpretasikan bahwa topik tersebut membahas tentang apresiasi terhadap produk motor listrik. Hal ini menunjukkan bahwa sebagian masyarakat mendukung dan mengapresiasi penggunaan motor listrik untuk ojek online.



Gambar 4.21 Topik Tertinggi dan Terendah Negatif

Berdasarkan Gambar 4.21, topik dengan jumlah komentar tertinggi berada di topik 3 dengan 687 komentar, sedangkan topik dengan komentar terendah berada di topik 2 dengan 7 komentar. Pada topik 3 terdapat kata kunci ”terjungkal, suara, gas” yang mengindikasikan bahwa topik ini membahas tentang keamanan dalam berkendara. Hal ini menunjukan bahwa masyarakat merasa terkejut saat motor digas karena tidak ada suara, sehingga hal ini bisa berbahaya jika penumpang tidak terbiasa akan berisiko jatuh kebelakang. Sementara itu, topik 2 topik terendah terdapat kata ”aduh, keluh, keringat” yang menginterpretasikan bahwa topik tersebut membahas tentang keluhan masyarakat terhadap produk motor listrik. Hal ini menunjukkan bahwa sebagian masyarakat masih mengeluhkan pengalaman yang dirasa kurang nyaman, seperti performa motor yang dianggap terlalu pelan sehingga membuat penumpang berkeringat yang menyebabkan riasan wajah berantakan.

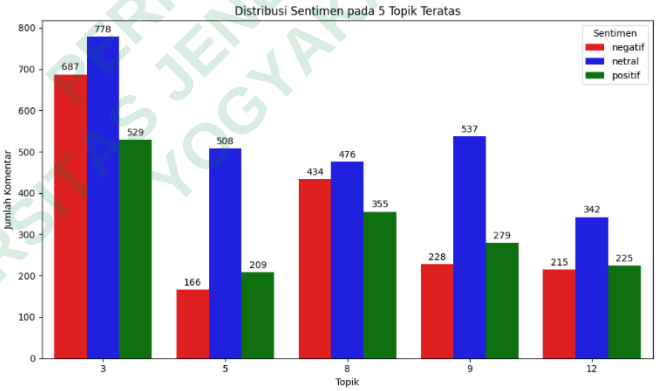


Gambar 4.22 Topik Tertinggi dan Terendah Netral

Berdasarkan Gambar 4.22, topik dengan jumlah komentar tertinggi berada di topik 3 dengan 778 komentar, sedangkan topik dengan komentar terendah berada di topik 2 dengan 3 komentar. Pada topik 3 terdapat kata kunci ”terjungkal, suara, gas” yang mengindikasikan bahwa topik ini membahas tentang keamanan dalam berkendara. Sementara itu, topik 2 topik terendah terdapat kata ”aduh, keluh, keringat” yang menginterpretasikan bahwa topik tersebut membahas tentang keluhan masyarakat terhadap produk motor listrik. Komentar dalam kedua topik tersebut bersifat netral, dimana pengguna hanya mengungkapkan sesuatu yang tidak mengandung kata positif maupun negatif. Contohnya adalah seperti komentar ”suara mesin” yang merepresentasikan topik 3 terkait suara.

4.10.2 Analisis Persepsi Setiap Topik

Setelah analisis topik setiap sentimen, selanjutnya dilakukan visualisasi analisis untuk melihat distribusi topik lima topik tertinggi berdasarkan sentimen untuk membandingkan topik pada setiap sentimen. Visualisasi lima topik tertinggi pada setiap sentimen dapat dilihat pada Gambar 4.23.



Gambar 4.23 Perbandingan Tiap Sentimen

Berdasarkan Gambar 4.23, pada topik 3 didominasi oleh sentimen netral, kemudian diikuti oleh sentimen negatif. Topik ini membahas tentang keamanan berkendara, yang menunjukkan bahwa keluhan tentang suara motor yang hening menyebabkan bahaya lebih dominan dibandingkan sisi positifnya yang dapat mengurangi polusi suara. Pada topik 5 didominasi oleh sentimen netral, kemudian diikuti oleh sentimen positif. Topik ini membahas tentang kendala operasional, meskipun topik ini membahas tentang kendala, tetapi masih ada sisi positif pada

kata kunci ”sewa, pesanan, coba” menunjukan adanya ketertarikan masyarakat untuk mencoba atau menyewa, meskipun masih menemui beberapa kendala. Pada topik 8 didominasi oleh sentimen netral, kemudian diikuti oleh sentimen negatif. Topik ini membahas tentang kenyamanan pelanggan, yang menunjukkan adanya keluhan terkait kenyamanan, seperti duduk sempit dan ketidaknyamanan saat melewati polisi tidur. Pada topik 9 didominasi oleh sentimen netral, kemudian diikuti oleh sentimen positif. Topik ini membahas tentang infrastruktur, yang menunjukkan adanya kesadaran masyarakat akan kebutuhan dukungan infrastruktur yang memadai untuk mendukung penggunaan motor listrik. Pada topik 12 didominasi oleh sentimen netral, kemudian diikuti oleh sentimen positif. topik ini berkaitan tentang kepuasan pelanggan, yang menunjukkan bahwa masyarakat menyukai motor listrik dan menilainya sebagai sesuatu yang menyenangkan.

4.11 KELEBIHAN DAN KEKURANGAN PENELITIAN

4.11.1 Kelebihan Penelitian

1. Penelitian ini mendapatkan data langsung dari komentar sosial media yang mencerminkan opini dan emosi masyarakat secara spontan dan alami yang mungkin tidak bisa didapat ketika wawancara atau kuesioner.
2. Proses *prepocessing* pada tahap normalisasi dilakukan dilakukan menggunakan kamus kata baku yang diperoleh dari github dengan sedikit tambahan kata baku. Setelah itu, dilakukan pengecekan secara manual untuk memastikan tidak ada yang kesalahan penulisan, singkatan kata, maupun yang lainnya, mengingat setiap orang memiliki gaya penulisan yang berbeda-beda.

4.11.2 Kekurangan Penelitian

1. Penelitian ini menggunakan pelabelan sentimen secara otomatis, yaitu *textblob*. Metode ini lebih cepat dan efisien untuk mengolah data dalam jumlah besar, namun textblob memiliki kekurangan dalam menangani teks berbahasa indonesia. *Texblob* terbatas dalam memahami konteks kalimat

dan struktur negasi yang menyebabkan model memberikan hasil yang bias terhadap kelas tertentu dan kurang mampu menangkap pola sentimen yang kompleks (Muhandhis et al., 2025). Dalam penelitian ini terlihat bahwa *recall* untuk sentimen positif 87% dan *precision* untuk sentimen netral 85% texblob. Hal ini menunjukkan model sedikit kesulitan dalam mengenali konteks positif dan terkadang salah dalam mengklasifikasikan netral.

2. Pemodelan topik dilakukan dengan satu model untuk seluruh sentimen yang nantinya jika kata kunci dari seluruh topik sudah ditemukan baru diimplementasikan ke seluruh komentar. Hal ini menyebabkan satu topik bisa muncul diberbagai kategori sentimen, yang menyebabkan kemunculan topik tertinggi yang serupa di setiap sentimen.