BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Peneliti mengumpulkan informasi dari komentar pengguna melalui aplikasi Segari di App Store dengan bantuan *library app store scraper* dan Google Play Store dengan *library google play scraper*. Selama proses pengumpulan data ini, peneliti menemukan 366 ulasan dari App Store dan 5373 ulasan dari Google Play Store dari data ulasan yang diambil mulai dari 28 Februari 2021 sampai dengan 31 Mei 2024. Tabel 4.1 menunjukkan daftar variabel yang digunakan dalam penelitian ini. Namun, dari kumpulan data yang dihasilkan dari ekstraksi ini, hanya satu variabel yang dianggap penting dalam penelitian ini karena relevansinya dan kemampuan untuk menganalisis sentimennya.

Tabel 4. 1 Kolom Data yang di pakai

Variabel	Deskripsi
text	Review pengguna aplikasi Segari dari platform Google Play Store dan App Store

Sedangkan untuk 5 teratas dari isi dari dataset hasil scrapping tersebut dapat dilihat pada tabel 4.2.

Tabel 4. 2 Hasil *Scrapping*

No	text
1	Jujur awal2 bagus ,,cepat,hemat,skrg janji jm 10 sampai udah jm 1 siang gg sampe2 alesan beludak padahal udah pesan dr smlm
2	ini pertama kali belanja di Segari .. wah mantaap deh .. pesanan memuaskan .. barang2 nya segar & ok smua
3	Kaya punya pintu doraemon, klik order, bayar, sampe, cepetttt benerrrrrr Segari the Best! banyakin lg stock nya yaa

…	…
5740	Enak bgt pake aplikasi segari. Ga kyk sebelah punya yg conplicated bgt

4.2 Pre-processing Data

Seperti yang disebutkan sebelumnya, data yang dikumpulkan oleh penulis masih tidak teratur. Ini termasuk penggunaan kata slang dan emotikon. Akibatnya, tahap ini sangat penting dalam pengolahan data agar siap digunakan dan diterapkan dalam pemodelan. Penulis akan melakukan proses *pre-processing* dalam lima tahapan.

4.2.1 *Case Folding*

Proses awal pra-pemrosesan data adalah mengubah huruf kapital dari setiap kata dalam dataset. Ini dilakukan dengan menggunakan metode casefoldingText pada data set yang dimiliki, dengan tujuan untuk mengubah setiap kata menjadi huruf kecil sehingga data dapat diproses dalam format yang seragam. Ini adalah perbandingan antara data asli dan hasil pengonversian huruf.

Tabel 4. 3 Hasil *Case Folding*

No.	Sebelum	sesudah
1	Jujur awal2 bagus ,,cepat,hemat,skrg janji jm 10 sampai udah jm 1 siang gg sampe2 alesan beludak padahal udah pesan dr smlm	jujur awal2 bagus ,,cepat,hemat,skrg janji jm 10 sampai udah jm 1 siang gg sampe2 alesan beludak padahal udah pesan dr smlm
2	ini pertama kali belanja di Segari .. wah mantaap deh .. pesanan memuaskan .. barang2 nya segar & ok smua	ini pertama kali belanja di segari .. wah mantaap deh .. pesanan memuaskan .. barang2 nya segar & ok smua
3	Kaya punya pintu doraemon, klik order, bayar, sampe,	kaya punya pintu doraemon, klik order, bayar, sampe,

	cepetttt benerrrrrr Segari the Best! banyakin lg stock nya yaa	cepetttt benerrrrrr segari the best! banyakin lg stock nya yaa
	…	…
5740	Enak bgt pake aplikasi segari. Ga kyk sebelah punya yg conplicated bgt	enak bgt pake aplikasi segari. ga kyk sebelah punya yg conplicated bgt

Hasil dari proses case folding tersebut juga dapat dilihat secara detail di lampiran

4.2.2 *Data Cleaning*

Menghapus karakter yang tidak relevan adalah langkah selanjutnya dalam proses pra-pemrosesan. Karakter seperti *mentions*, *hashtag*, tautan, dan tanda baca akan dihilangkan oleh penulis di sini. Hasilnya adalah data yang akurat, bersih, dan memiliki nilai yang signifikan. Ini adalah *output* dari proses pembersihan data. Dari proses ini, didapat 42 data ulasan yang kosong karena data ulasan pengguna hanya terdiri dari emotikon atau angka sehingga akan dihapus.

Tabel 4. 4 Hasil *Cleaning Data*

No.	Sebelum	sesudah
1	Jujur awal2 bagus ,,cepat,hemat,skrg janji jm 10 sampai udah jm 1 siang gg sampe2 alesan beludak padahal udah pesan dr smlm	jujur awal bagus cepat hemat skrg janji jm sampai udah jm siang gg sampe alesan beludak padahal udah pesan dr smlm
2	ini pertama kali belanja di Segari .. wah mantaap deh .. pesanan memuaskan .. barang2 nya segar & ok smua	ini pertama kali belanja di segar wah mantaap deh pesan muas barang nya segar ok smua
3	Kaya punya pintu doraemon, klik order, bayar, sampe, cepetttt benerrrrrr Segari the Best! banyakin lg stock nya yaa	kaya punya pintu doraemon klik order bayar sampe cepetttt benerrrrrr segar the best banyakin lg stock nya yaa
	…	…

5740	Enak bgt pake aplikasi segari. Ga kyk sebelah punya yg conplicated bgt	enak bgt pake aplikasi segari ga kyk sebelah punya yg conplicated bgt

Hasil dari *cleaning* data tersebut juga dapat dilihat secara detail di lampiran

4.2.3 *Stemming*

Selanjutnya, setiap kata diberi imbuhan atau diubah ke kata pokoknya. Misalnya, kata "mengikuti" diubah menjadi "ikut". Sastrawi adalah *library* Python yang diperlukan untuk menjalankan proses ini. Ini adalah *output* dari prose *stemming.*

Tabel 4. 5 Hasil *Stemming*

No.	Sebelum	sesudah
1	jujur awal bagus cepat hemat skrg janji jm sampai udah jm siang gg sampe alesan beludak padahal udah pesan dr smlm	jujur awal bagus cepat hemat skrg janji jm sampai udah jm siang gg sampe alesan beludak padahal udah pesan dr smlm
2	ini pertama kali belanja di segari wah mantaap deh pesanan memuaskan barang nya segar ok smua	ini pertama kali belanja di segari wah mantaap deh pesan muas barang nya segar ok smua
3	kaya punya pintu doraemon klik order bayar sampe cepetttt benerrrrrr segari the best banyakin lg stock nya yaa	kaya punya pintu doraemon klik order bayar sampe cepetttt benerrrrrr segari the best banyak lg stock nya yaa
…	…	…
5740	enak bgt pake aplikasi segari ga kyk sebelah punya yg conplicated bgt	enak bgt pake aplikasi segari ga kyk belah punya yg conplicated bgt

Hasil dari proses *stemming* tersebut juga dapat dilihat secara detail di lampiran

4.2.4 *Normalization*

Dalam ulasan aplikasi Segari, kata-kata singkatan dan ejaan yang salah disebabkan oleh fakta bahwa pengguna memberikan komentar mereka dalam bentuk kalimat yang tidak biasa. Oleh karena itu, untuk membuat kata-kata tersebut lebih sesuai, proses normalisasi diperlukan. Proses ini melibatkan penggunaan dataset kata-kata baku, yang membantu mengubah atau mengganti ejaan yang tidak sesuai. Dataset kata-kata baku ini berasal dari berbagai sumber, seperti https://github.com/ksnugroho/pycon-id-2020/ dan ditambahkan beberapa kata baku oleh peneliti untuk menyesuai dengan *dataset* yang digunakan, korpus kata baku yang dipakai dapat dilihat di halaman lampiran. Hasil normalisasi ditunjukkan di bawah.

Tabel 4. 6 Hasil *Normalization*

No.	Sebelum	sesudah
1	jujur awal bagus cepat hemat skrg janji jm sampai udah jm siang gg sampe alesan beludak padahal udah pesan dr smlm	jujur awal bagus cepat hemat sekarang janji jam sampai sudah jam siang tidak sampai alasan beludak padahal sudah pesan dari semalam
2	ini pertama kali belanja di segari wah mantaap deh pesan muas barang nya segar ok smua	ini pertama kali belanja di segari memukau mantap deh pesan puas barang nya segar oke semua
3	kaya punya pintu doraemon klik order bayar sampe cepetttt benerrrrrr segari the best banyak lg stock nya yaa	kayak punya pintu doraemon klik pesanan bayar sampai cepat benar segari the terbaik banyak lagi persediaan nya ya
…	…	…
5740	enak bgt pake aplikasi segari ga kyk belah punya yg conplicated bgt	enak banget pakai aplikasi segari tidak seperti belah punya yang conplicated banget

Hasil dari proses *normalization* tersebut juga dapat dilihat secara detail di lampiran.

4.2.5 Stopword

Selanjutnya, Pada tahap terakhir, akan dilakukan proses *stopword* menggunakan modul sastrawi *stopWordRemoverFactory* untuk melakukan proses filtering untuk menghilangkan kata-kata umum atau *stopword* yang tidak berguna. Ini membuat data hanya berfokus pada kata utama.

Tabel 4. 7 Hasil *Stopword*

No.	Sebelum	sesudah
1	jujur awal bagus cepat hemat sekarang janji jam sampai sudah jam siang tidak sampai alasan beludak padahal sudah pesan dari semalam	jujur bagus cepat hemat janji jam jam siang tidak alasan beludak pesan semalam
2	ini pertama kali belanja di segari memukau mantap deh pesan puas barang nya segar oke semua	kali belanja segari memukau mantap pesan puas barang segar oke
3	kayak punya pintu doraemon klik pesanan bayar sampai cepat benar segari the terbaik banyak lagi persediaan nya ya	kayak pintu klik pesanan bayar cepat segari terbaik persediaan
…	…	
5639	enak banget pakai aplikasi segari tidak seperti belah punya yang conplicated banget	enak banget pakai aplikasi segari tidak belah conplicated banget

Hasil *preprocessing* menunjukkan 62 data ulasan yang kosong karena data ulasan pengguna hanya terdiri dari kata-kata yang tidak berguna, yang dihapus oleh proses di atas. Jumlah data sisa dalam *data frame* adalah 5635. Hasil dari proses *stopword* dapat dilihat secara detail di lampiran

4.3 *Labelling*

Setelah tahap *pre-processing* selesai, data yang telah dibersihkan akan digunakan untuk proses pelabelan data menggunakan *Indonesian Sentiment Lexicon* atau *Inset Lexicon*. Lexicon ini dibuat dengan menggabungkan beberapa sumber seperti github.com/louisowen6/NLP_bahasa_resources, github.com/abhimantramb/elang/blob/master/word2vec/utils/swear-words.txt, dan https://github.com/fajri91/InSet menjadi kumpulan kamus sentimen yang menilai polaritas setiap kata.

```
import numpy as np

sencol =[]
senrow =np.array([])
nsen = 0
factory = StemmerFactory()
stemmer = factory.create_stemmer()
sentiment_list = []
# function to write the word's sentiment if it is founded
def found_word(ind,words,word,sen,sencol,sentiment,add):
    # if it is already included in the bag of words matrix, then just increase the value
    if word in sencol:
        sen[sencol.index(word)] += 1
    else:
    #if not, than add new word
        sencol.append(word)
        sen.append(1)
        add += 1
    #if there is a negation word before it, the sentiment would be the negation of it's sentiment
    if (words[ind-1] in negasi):
        sentiment += -lexicon['weight'][lexicon_word.index(word)]
    else:
        sentiment += lexicon['weight'][lexicon_word.index(word)]

    return sen,sencol,sentiment,add

# checking every words, if they are appear in the lexicon, and then calculate their sentiment if they do
for i in range(len(comment)):
    nsen = senrow.shape[0]
    words = word_tokenize(comment['data_stopword'][i])
    sentiment = 0
    add = 0
    prev = [0 for ii in range(len(words))]
    n_words = len(words)
    if len(sencol)>0:
        sen =[0 for j in range(len(sencol))]
    else:
        sen =[]

    for word in words:
        ind = words.index(word)
        # check whether they are included in the lexicon
        if word in lexicon_word :
            sen,sencol,sentiment,add= found_word(ind,words,word,sen,sencol,sentiment,add)
        else:
        # if not, then check the root word
            kata_dasar = stemmer.stem(word)
            if kata_dasar in lexicon_word:
                sen,sencol,sentiment,add= found_word(ind,words,kata_dasar,sen,sencol,sentiment,add)
        # if still negative, try to match the combination of words with the adjacent words
            elif(n_words>1):
                if ind-1>-1:
                    back_1    = words[ind-1]+' '+word
                    if (back_1 in lexicon_word):
                        sen,sencol,sentiment,add= found_word(ind,words,back_1,sen,sencol,sentiment,add)
                    elif(ind-2>-1):
                        back_2    = words[ind-2]+' '+back_1
                        if back_2 in lexicon_word:
                            sen,sencol,sentiment,add= found_word(ind,words,back_2,sen,sencol,sentiment,add)
    sentiment_list.append(sentiment)
```

Gambar 4. 1 Sintaks Implementasi Proses Labelling

Tabel 4. 8 Hasil Pelabelan Dengan InSet Lexicon

No.	ulasan	skor	kategori

1	sayur segar mantap pokok	12	Positif
2	gambar hilang	-2	Negatif
3	kali pesanan hasil puas segar langgan	14	Positif
…	…		
5639	enak banget pakai aplikasi segari belah conplicated banget	2	Positif

Sebagaimana ditunjukkan pada tabel di atas. Hasil kategori sentimen ditunjukkan dengan memberikan nilai lebih dari 0 sebagai label sentimen positif, nilai/*polarity score* kurang dari 0 sebagai label sentimen negatif. Jadi, dengan perhitungan kata, hasil perbandingan jumlah data untuk masing-masing label atau kelas dapat dilihat pada tabel di bawah. Dari proses pelabelan ini didapat juga nilai 0 sebagai label netral sebanyak 198 data ulasan yang akan dihapus karena penelitian ini berfokus pada data ulasan positif dan negatif.

Tabel 4. 9 Hasil Persebaran Data Tiap Kelas/Label

Sentimen	Skor Sentimen		Ulasan
	Minimum	Maksimum	
Positif	1	76	5043
Negatif	-28	-1	395

Dengan menggunakan *Inset Lexicon* untuk pelabelan, distribusi data untuk masing-masing kelas dapat dilihat. Komentar kelas positif memiliki mayoritas, sebanyak 5042 dari total data dan sisanya yaitu kelas negatif, sebanyak 395.

4.4 Pembobotan Kata

Setelah data diberi label dari kamus *Indonesian Sentiment Lexicon* atau *InSet Lexicon*, langkah berikutnya adalah pembobotan kata dengan TF-IDF. Dalam penelitian ini, penulis menggunakan scikit-learn dan fungsi CountVectorizer untuk mengekstrak data teks menjadi vektor numerik dan menghitung berapa kali setiap term muncul dalam dokumen, atau teks. Selanjutnya, perhitungan bobot TF-IDF akan

dilakukan dengan menggunakan perpustakaan TfidfTranformer untuk mengukur kepentingan suatu term dalam dokumen (teks). Sintaks untuk menghitung bobot TF-IDF sebagai berikut.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

# Memeriksa apakah kolom 'text' ada dalam DataFrame
if 'text' in review.columns:
    # Menghapus baris dengan nilai NaN atau kosong di kolom 'text'
    review = review.dropna(subset=['text'])
    review = review[review['text'].str.strip() != '']

    # Memeriksa apakah DataFrame tidak kosong setelah pembersihan
    if not review.empty:
        # Inisialisasi TfidfVectorizer
        tfidf_vectorizer = TfidfVectorizer()

        # Fit dan transformasi data teks di kolom 'text'
        tfidf_matrix = tfidf_vectorizer.fit_transform(review['text'])

        # Menampilkan TF-IDF per kata yang ada dalam dokumen
        feature_names = tfidf_vectorizer.get_feature_names_out()
        for i, row in enumerate(review['text']):
            print(f"TF-IDF untuk teks '{row}':")
            # Membuat dictionary untuk menyimpan nilai TF-IDF hanya untuk kata-kata yang ada dalam teks
            tfidf_dict = {word: tfidf_matrix[i, idx] for word, idx in tfidf_vectorizer.vocabulary_.items() if word in row.split()}
            for word, tfidf_value in tfidf_dict.items():
                print(f"{word}: {tfidf_value}")
            print()

            # Menampilkan kolom kategori_sentiment dari DataFrame
            kategori_sentiment = review.loc[i, 'kategori_sentiment']
            print(f"Kategori sentimen untuk teks '{row}': {kategori_sentiment}")
            print()

    else:
        print("DataFrame kosong setelah pembersihan.")
else:
    print("Kolom 'text' tidak ditemukan dalam DataFrame review.")
```

Gambar 4. 2 Sintaks Implementasi Proses Pembobotan Kata

Tabel 4. 10 Hasil dari Pembobotan Kata

No.	ulasan	Hasil Pembobotan
1	sayur segar mantap pokok	sayur: 0.45320200148766493 segar: 0.330092163355954 mantap: 0.5229748895050695 pokok: 0.6419847151496062
2	gambar hilang	gambar: 0.7008583288460836 hilang: 0.713300499710378
3	kali pesanan hasil puas segar langgan	kali: 0.36293138208756937 pesanan: 0.4107536741151008 hasil: 0.6203418339209256 puas: 0.2511765286937826 segar: 0.20662518514849887 langgan: 0.4571155482810279
…	…	

5639	enak banget pakai aplikasi segari belah conplicated banget	enak: 0.37046453032462245 banget: 0.4728474186714278 pakai: 0.2898583152326259 aplikasi: 0.229890299888645 segari: 0.1618083800200775 belah: 0.39276779518764243 conplicated: 0.5673230691179878

Dari tabel sebelumnya dapat dilihat beberapa bobot pada kata dalam suatu dokumen, dari proses ini didapat 2240 kata. Untuk melihat lebih lengkap bobot dari masing-masing kata secara keseluruhan pada dokumen dapat dilihat pada halaman lampiran.

4.5 Split Data

Sebelum model klasifikasi dibangun, proses membagi data latih dan uji akan dilakukan dengan bantuan modul train_test_split dari sklearn.model_selection, hasil latihan dan uji dapat membantu memprediksi kategori data baru. Penelitian ini menggunakan data latih dan uji yang mengandung sentimen negatif dan positif. Persentase pembagian data latih dan uji dalam dataset penelitian adalah 80 persen untuk data latih dan 20 persen untuk data uji dari 5437 data.

```
from sklearn.model_selection import train_test_split

# Membagi data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 4. 3 Sintaks Implementasi *Split Data*

Tabel 4. 11 Banyaknya Data Uji dan Data Latih

Data Uji	1088
Data Latih	4349

Tabel di atas menunjukkan banyaknya data hasil proses *split data* di mana data uji didapat sebanyak 1084 dokumen dan data latih sebanyak 4336 dokumen.

4.6 Klasifikasi dengan *Support Vector Machine* (SVM)

Proses pemodelan dengan SVM dapat dimulai setelah pembagian data latih dan data uji selesai. Untuk membuat dan melatih model *Support Vector Classifier*, model SVM akan menggunakan *library* SVC dari sklearn.svm. Ini juga akan membantu evaluasi model dengan menggunakan classification_report dan confusion_matrix dari sklearn.metrics serta akan menggunakan kernel *linear*, dari penelitian terdahulu pada bab 2 didapat bahwa kernel *linear* merupakan kernel yang menghasilkan akurasi paling tinggi. Oleh karena itu, pada penelitian ini akan memakai kernel *linear*. Di bawah, dapat dilihat kode yang digunakan untuk pemodelan.

```
# Membuat prediksi pada data uji
y_pred = svm_model_linear.predict(X_test)

# Mendapatkan koefisien dari model SVM linear
coefficients = svm_model_linear.coef_[0]
feature_names = X.columns

# Membuat DataFrame untuk koefisien dan fitur
df_coefficients = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefficients
})

# Menandai fitur positif dan negatif
df_coefficients['Sentiment'] = np.where(df_coefficients['Coefficient'] > 0, 'Positive', 'Negative')

# Menampilkan 10 fitur teratas yang paling berkontribusi untuk kelas positif dan negatif
top_positive_coefficients = df_coefficients[df_coefficients['Sentiment'] == 'Positive'].sort_values(by='Coefficient', ascending=False).head(10)
top_negative_coefficients = df_coefficients[df_coefficients['Sentiment'] == 'Negative'].sort_values(by='Coefficient').head(10)

print("Top 10 Positive Coefficients:")
print(top_positive_coefficients[['Feature', 'Coefficient']])
print("\nTop 10 Negative Coefficients:")
print(top_negative_coefficients[['Feature', 'Coefficient']])
```

Gambar 4. 4 Sintaks Klasifikasi SVM

Tabel 4. 12 Top 10 Koefisien Kata Positif

No.	Kata	Koefisien
1	segar	3.212228
2	segari	3.121597
3	promo	2.670042
4	mantap	2.086058
5	puas	1.974531
6	manfaat	1.804548
7	semoga	1.571062
8	harga	1.451550
9	produk	1.441989
10	nyata	1.433243

Tabel 4. 13 Top 10 Koefisien Kata Negatif

No.	Kata	Koefisien
1	Aplikasi	-2.784400
2	Batal	-2.706193
3	Belah	-2.026461
4	Rugi	-2.012428
5	Terlambat	-2.008502
6	Kecewa	-1.949299
7	Jelek	-1.889509
8	Kosong	-1.677537
9	Kode	-1.669345
10	salah	-1.644513

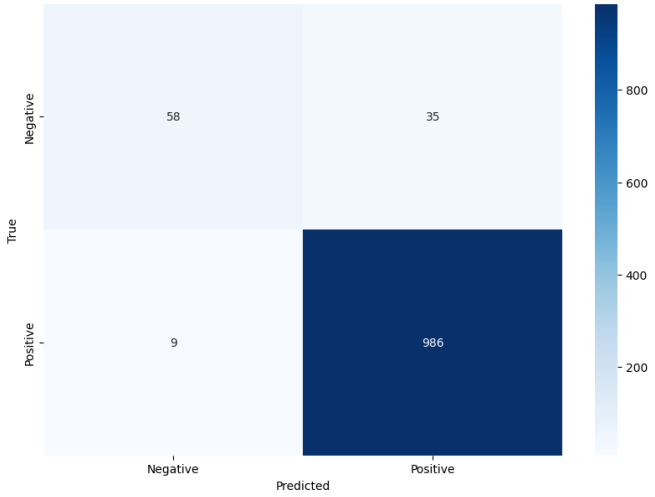
Pada tabel di atas dapat dilihat top 10 koefisien kata positif dan negatif. Untuk melihat lebih lengkap kata dan nilai koefisiennya dapat dilihat pada halaman lampiran.

4.7 Hasil Evaluasi Model

Setelah membuat model klasifikasi SVM dan menerapkannya pada dataset penelitian ini, langkah berikutnya adalah melakukan tinjauan performa model menggunakan metode *confussion matrix* untuk melihat tingkat *accuracy*, *precission*, *recall*, dan *f1-score*. Untuk melakukan ini, penulis akan menggunakan *library* sklearn.metric dengan modul classification_report dan confusion_matrix. Hasil tinjauan model dapat dilihat pada tabel di bawah.

Sebanyak 986 komentar positif diprediksi sebagai data yang benar atau TP (*True Positive*), 58 komentar negatif diprediksi sebagai data yang benar atau TN (*True Negative*), 35 komentar positif diprediksi sebagai data yang salah atau FP (*False Positive*), dan 58 komentar negatif diprediksi sebagai data yang benar atau FN (*False Negative*). Kita dapat menghitung *accuracy, precission, recall* dan *f1-score* yang didapat dari penerapan model SVM pada *dataset* ulasan aplikasi Segari melalui

penggambaran hasil *confusion matrix*. Gambar berikut menunjukkan hasil *confusion matrix* dari model SVM yang telah diterapkan.



Gambar 4. 5 Hasil *Confusion Matrix*

Perhitungan berikut dapat digunakan untuk melakukan ini.

Kelas Positif:

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{986}{986+35} = \frac{986}{1021} = 0.97$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{989}{986+9} = \frac{989}{995} = 0.99$$

$$\text{F1-Score} = 2 \text{ x } \frac{Precision\ x\ Recall}{Precision+Recall} = 2 \text{ x } \frac{0.97\ x\ 0.99}{0.97+0.99} = 0.98$$

Kelas Negatif:

$$\text{Precision} = \frac{TN}{TN+FN} = \frac{58}{58+9} = \frac{58}{67} = 0.87$$

$$\text{Recall} = \frac{TN}{TN+FP} = \frac{58}{58+35} = \frac{58}{93} = 0.62$$

$$\text{F1-Score} = 2 \text{ x } \frac{Precision\ x\ Recall}{Precision+Recall} = 2 \text{ x } \frac{0.87\ x\ 0.62}{0.87+0.62} = 0.72$$

$$\text{Akurasi} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{986+58}{986+58+35+9} \quad \frac{1044}{1088} = 0.96$$

Tabel 4. 14 Hasil Evaluasi Model

kelas	*Precission*	*Recall*	*F1-Score*	*Accuracy*
Positif	97%	99%	98%	96%
Negatif	87%	62%	72%	

Tabel di atas menunjukkan bahwa model memiliki performa yang baik. Akurasi yang tinggi memastikan bahwa sebagian besar prediksi model adalah benar, presisi yang tinggi memastikan bahwa prediksi positif model sebagian besar benar, dan *recall* yang sangat tinggi menunjukkan kemampuan model untuk menemukan hampir semua kasus positif yang sebenarnya dan kesalahan pada penentuan kasus negatif sesedikit mungkin.

4.8 *Labelling* dengan SVM

Tahapan selanjutnya, akan menggunakan model *Support Vector Machine* (SVM) yang telah kita buat sebelumnya untuk memulai tahapan *labelling*. Proses yang dilakukan oleh *Support Vector Machine* (SVM) ini dapat dilihat pada sintaks di bawah ini.

```
# Memastikan df_tfidf adalah representasi TF-IDF dari teks
X_new = df_tfidf

# Membuat prediksi pada data baru
predictions = svm_model_linear.predict(X_new)

# Menambahkan hasil prediksi sebagai kolom baru di DataFrame asli (df_labels)
df_labels['predicted_sentiment'] = predictions

# Menampilkan DataFrame dengan label yang diprediksi
print(df_labels)

# Menghitung jumlah prediksi positif dan negatif
pred_count = df_labels['predicted_sentiment'].value_counts()

# Menampilkan jumlah prediksi positif dan negatif
print("Jumlah prediksi positif:", pred_count.get(1))
print("Jumlah prediksi negatif:", pred_count.get(0))

df_labels.head()
```

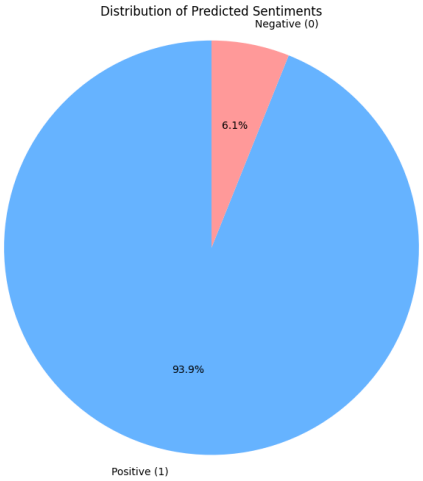
Gambar 4. 6 Sintaks Implementasi Proses *Labelling* Dengan Model yang Telah Dibuat

Tabel 4. 15 Hasil Implementasi Labelling Dengan Model yang Telah Dibuat

No	Pelabelan	Ulasan

	InSet Lexicon	SVM	
1	1	1	sayur segar mantap pokok
2	0	0	buruk tipu unduh saran
3	1	1	kali pesanan hasil puas segar langgan
…	…	…	…
5437	1	1	enak banget pakai aplikasi segari belah conplicated banget

Dengan menggunakan model *Support Vector machine* yang sudah dibuat untuk pelabelan, distribusi data untuk masing-masing kelas dapat dilihat. Komentar kelas positif dominan pada ulasan aplikasi Segari, sebanyak 93,9% atau 5107 dari total data dan sisanya yaitu kelas negatif, sebanyak 6,1% atau 330.



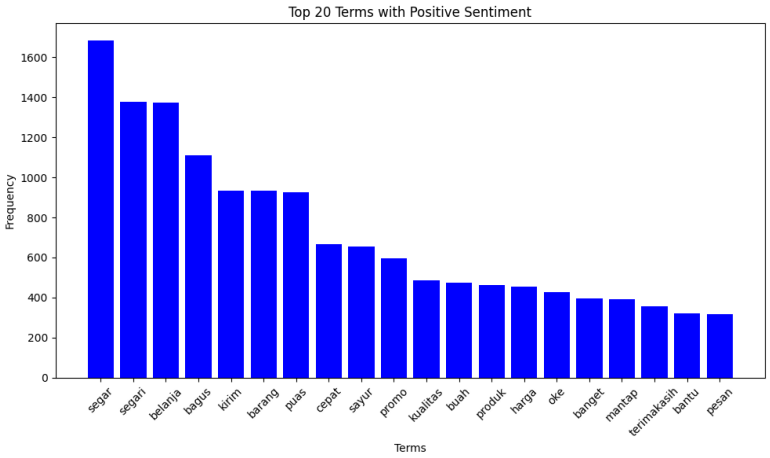
Gambar 4. 7 Persentase Dokumen Positif dan Negatif

4.9 Hasil dan Visualisasi

Hasil dari proses pengolahan data dari review aplikasi Segari memiliki visualisasi terhadap kata terbanyak dari kata positif dan negatif

4.10 Sentimen Positif

Pada kata terbanyak dari sentimen positif pada aplikasi tersebut terdapat kata “segar”, “kirim”, “barang”, “cepat”, “promo”, “kualitas”, “produk”, “harga”, dan “bantu”. Berikut merupakan hasil dari visualisasi data yang sudah diolah.



Gambar 4. 8 Top 20 Kata Dalam Dokumen Positif

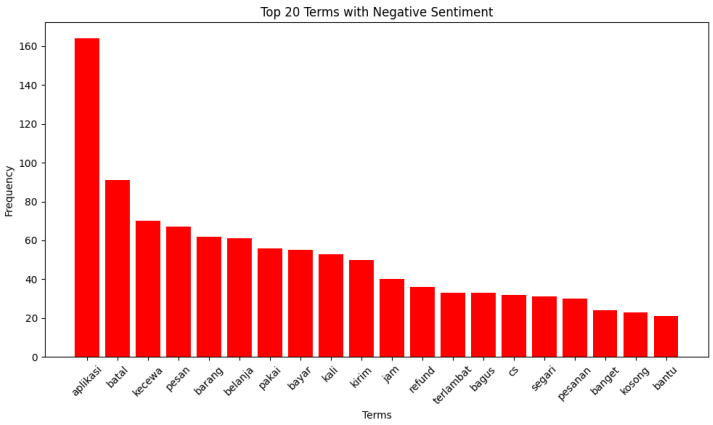
Berikut ini adalah hasil analisis data review aplikasi Segari, dengan penekanan khusus pada kata-kata terbanyak dari sentimen positif

Analisis Kata Positif Aplikasi :

- Segar : produk yang diterima konsumen biasanya segar, yang menunjukkan kualitas produk yang baik.
- Kirim : Salah satu keunggulan yang di *mention* oleh pengguna adalah pengiriman yang cepat dan efisien.
- Barang : Barang berkualitas tinggi dan sesuai dengan yang diharapkan pelanggan.
- Cepat : Pelanggan sangat menghargai kecepatan proses pemesanan dan pengiriman.
- Promo : Promosi yang menarik dan sering diberikan kepada pengguna merupakan nilai tambah yang signifikan.
- Kualitas : Salah satu keuntungan utama adalah produk yang berkualitas tinggi.
- Produk : Kepuasan dengan produk Segari.
- Harga: Harga yang kompetitif dan sesuai dengan kualitas produk.
- Bantu : Dukungan yang diberikan oleh aplikasi dan layanan pelanggan yang dianggap sangat bermanfaat oleh pengguna.

4.11 Sentimen Negatif

Pada kata terbanyak dari sentimen negatif pada aplikasi tersebut terdapat kata kata “aplikasi”, “batal”, “bayar”, “barang”, “pesan”, “kirim”, “*refund*”, “CS (*Customer Service*)”, dan “kurir”. Berikut merupakan hasil dari visualisasi data yang sudah diolah



Gambar 4. 9 Top 20 Kata Dalam Dokumen Negatif

Berikut ini adalah hasil analisis data review aplikasi Segari, dengan penekanan khusus pada kata-kata terbanyak dari sentimen negatif.

Analisis kata Negatif Aplikasi :

Ada banyak keluhan tentang masalah teknis atau masalah pengalaman pengguna dengan aplikasi Segari.

- Batal : Pesanan pelanggan sering dibatalkan karena berbagai alasan, seperti ketersediaan produk atau masalah teknis.
- Bayar : Seringkali ada masalah dengan pembayaran, mungkin karena proses transaksi yang gagal atau masalah dengan metode pembayaran.
- Barang : Ada masalah dengan produk yang diterima, baik itu terkait kualitas atau kesesuaian dengan pesanan.
- Pesan : Proses pemesanan mungkin mengalami masalah.
- Kirim dan kurir : Pengiriman produk sering menjadi masalah. Ini dapat terjadi karena keterlambatan atau masalah pengiriman yang dilakukan oleh kurir.

- *Refund* : Ketika barang dibatalkan atau dikembalikan, pelanggan sering mengalami masalah dengan prose pengembalian uang.
- CS (*Customer Service*) : Layanan pelanggan yang mungkin tidak menanggapi atau tidak berhasil menangani keluhan dan masalah pengguna dengan baik.
- Kurir : Masalah dengan kurir, seperti keterlambatan atau sikap yang tidak memuaskan.