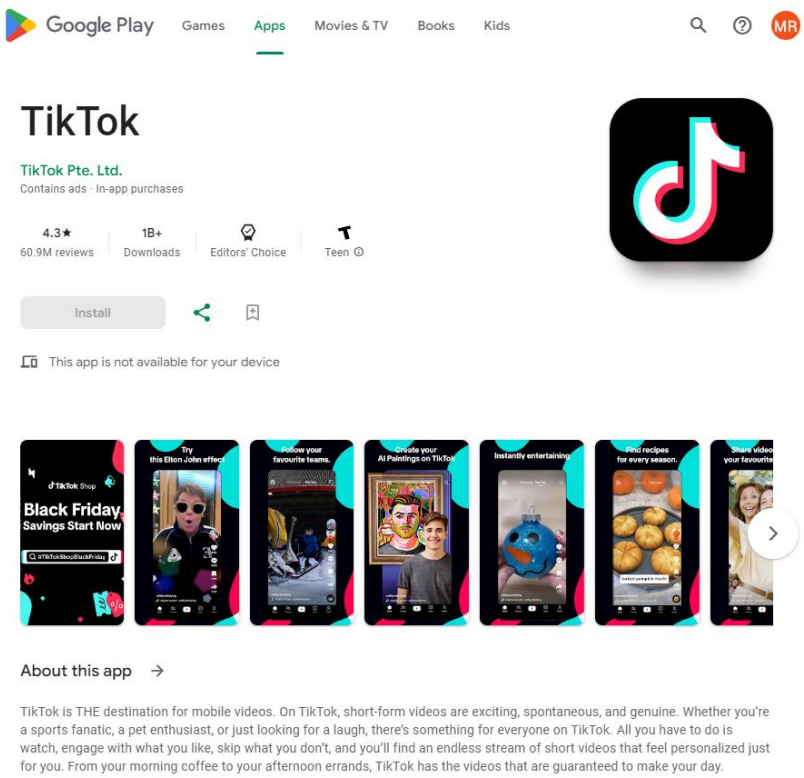
BAB IV

ANALISIS

4.1. GAMBARA UMUM APLIKASI *TIKTOK*

TikTok adalah platform media sosial untuk membuat video pendek dengan durasi maksimal 10 menit, *live streaming*, duet, dan berbagai fitur kreatif lainnya. Diluncurkan pada 2016 oleh ByteDance di bawah Zhang Yimin, *TikTok* populer di kalangan anak muda Indonesia dan berperan penting dalam membentuk budaya digital, termasuk pemasaran bisnis melalui kampanye kreatif dan kolaborasi dengan pengguna.



Gambar 4.1 Aplikasi *Tiktok* di *Google Play Store*

4.2. PENGUMPULAN DATA DAN *LABELLING*

4.2.1 Pengumpulan Data

Sumber *dataset* hanya diambil pada platform *Kaggel.com* *https://www.kaggle.com/datasets/shivkumarganesh/tiktok-google-play-store-review*, dengan keselurahan jumlah data dalam *dataset* sebanyak 460287 kemudian diambil sampel dengan menggunakan rumus *slovin*, berikut penjelasannya:

$$n = \frac{N}{1 + N(e)^2} = \frac{460287}{1 + 460287(1.5\%)^2}$$

$$= \frac{460287}{1 + 460287(0{,}000225)} = 4401.9$$

Maka dibulankan menjadi 4402. Dengan demikian jumlah sampel penelitian (n) sebanyak 4402 *sentiment.*

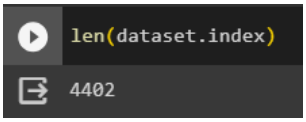
Data yang diperoleh memiliki beberapa atrbut, antara lain: *reviewId, UserNama, userImage, content, score, thumbsUpCount, reviewCreatedVersion, at, replyContent, repliedAt.* Oleh karna itu, atribut yang tidak digunakan akan dihapus, sehingga menjadi sederhana dengan 4 atribut yang digunakan yaitu:

a. *UserName*: nama user yang memberi *review tiktok* di *google play store*.
b. *Content*: merupakan isi *review* terhadap aplikasi *tiktok* di *google play store.*

c. *Score*: penilaian terhadap aplikasi *tiktok* di *google play store* dengan *score* paling kecil adalah 1, dan paling besar adalah 5, yang disimbolkan dengan bintang.

d. *At*: tanggal *username* memberi *reveiw*.

Data yang digunakan adalah *dataset* mentah yang akan melawati tahapan *preprocessing*, sehingga menjadi *dataset* yang siap digunakan.



Gambar 4.2 Proses Menghitung Jumlah Data

Pada gambar 4.2 merupakan proses perhitungan jumlah data yang akan digunakan menggunakan *code len* (…), sehingga dapat dilihat jumlah data sebanyak 4402 data.

```
dataset = pd.read_csv('/content/drive/MyDrive/claraTA/dataset_4402_.csv')
print("atribut yang tersedia : ")
atribut_yang_dipilih=['userName','content','score','at']
Df_tiktok = dataset [atribut_yang_dipilih]
Df_tiktok.to_csv('DataSentiment.csv', index=False)
```

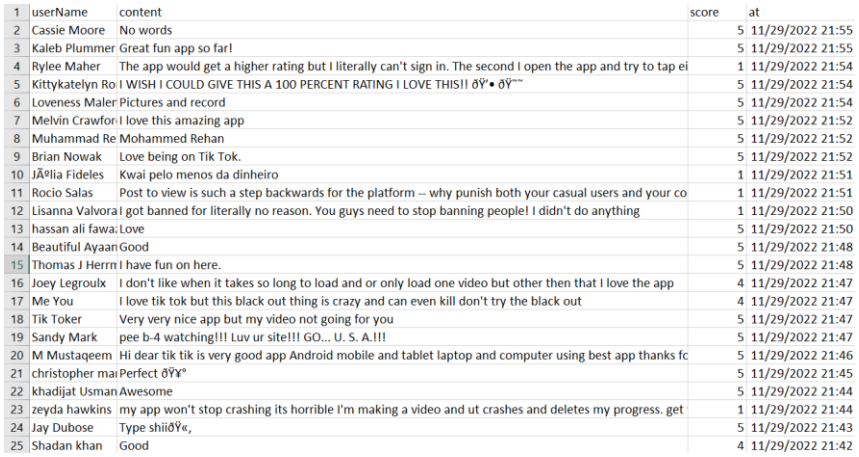
Gambar 4.3 Proses Peyderhanaan *Dataset*

Pada gambar 4.3 merupakan proses penyederhanaan *dataset*, yang dimana kita pilih terlebih dahulu atribut yang kita butuhkan, kemudian kita *save dataset* tersebut dalam bentuk *csv*, dengan nama *file DataSentiment.csv.*

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt
2	68ccaec8-	Cassie Moore	https://play-lh	No words	5	0	27.1.3	11/29/2022 21:55		
3	d84cbfd3-	Kaleb Plumme	https://play-lh	Great fun app so far!	5	0		11/29/2022 21:55		
4	96618aa1	Rylee Maher	https://play-lh	The app would get a higher rating	1	0	27.1.3	11/29/2022 21:54		
5	078c0bda-	Kittykatelyn R	https://play-lh	I WISH I COULD GIVE THIS A 100 PI	5	0		11/29/2022 21:54		
6	8e68c5cd-	Loveness Mal	https://play-lh	Pictures and record	5	0		11/29/2022 21:54		
7	08dc9129	Melvin Crawfo	https://play-lh	I love this amazing app	5	0	27.1.3	11/29/2022 21:52		
8	1b22eaaa-	Muhammad R	https://play-lh	Mohammed Rehan	5	0	27.1.3	11/29/2022 21:52		
9	f2ce64ec-!	Brian Nowak	https://play-lh	Love being on Tik Tok.	5	0	27.0.3	11/29/2022 21:52		
10	0e4af7cc-!	JÃºlia Fideles	https://play-lh	Kwai pelo menos da dinheiro	1	1	27.0.3	11/29/2022 21:51		
11	25c6b61c	Rocio Salas	https://play-lh	Post to view is such a step backwa	1	166	27.1.3	11/29/2022 21:51		
12	f97768fe-:	Lisanna Valvo	https://play-lh	I got banned for literally no reason	1	0	27.1.3	11/29/2022 21:50		
13	c45e24a9	hassan ali faw	https://play-lh	Love	5	0	27.1.3	11/29/2022 21:50		
14	102df96f-:	Beautiful Aya	https://play-lh	Good	5	0		11/29/2022 21:48		
15	79878f06-	Thomas J Her	https://play-lh	I have fun on here.	5	0	27.1.3	11/29/2022 21:48		
16	713a904d	Joey Legroulx	https://play-lh	I don't like when it takes so long to	4	0	26.1.3	11/29/2022 21:47		
17	c033301f-	Me You	https://play-lh	I love tik tok but this black out thir	4	0		11/29/2022 21:47		
18	33f9466c-	Tik Toker	https://play-lh	Very very nice app but my video no	5	0	27.0.3	11/29/2022 21:47		
19	8066bfbf-(	Sandy Mark	https://play-lh	pee b-4 watching!!! Luv ur site!!! G	5	0	27.1.3	11/29/2022 21:47		
20	968a38b1	M Mustaqeem	https://play-lh	Hi dear tik tik is very good app Anc	5	0	27.0.3	11/29/2022 21:46		
21	5b8682e2	christopher m	https://play-lh	Perfect ðŸ¥°	5	0	27.0.3	11/29/2022 21:45		
22	8ae260f0-	khadijat Usm	https://play-lh	Awesome	5	0	27.0.3	11/29/2022 21:44		
23	66b4e691	zeyda hawkin	https://play-lh	my app won't stop crashing its hor	1	0	27.1.3	11/29/2022 21:44		
24	0cf2127b-	Jay Dubose	https://play-lh	Type shiiðŸ«,	5	0		11/29/2022 21:43		
25	12a6efd7-	Shadan khan	https://play-lh	Good	4	0		11/29/2022 21:42		

Gambar 4.4 *Dataset* Sebelum Disederhanakan

Pada gambar 4.4 merupakan gambaran *dataset* yang belum disederhanakan, dengan atribut *reviewId, UserNama, userImage, content, score, thumbsUpCount, reviewCreatedVersion, at, replyContent, repliedAt.*



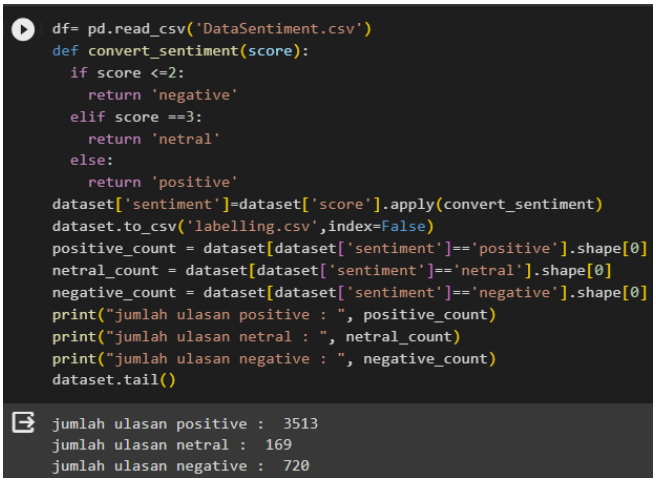
	userName	content	score	at
2	Cassie Moore	No words	5	11/29/2022 21:55
3	Kaleb Plummer	Great fun app so far!	5	11/29/2022 21:55
4	Rylee Maher	The app would get a higher rating but I literally can't sign in. The second I open the app and try to tap ei	1	11/29/2022 21:54
5	Kittykatelyn Ro	I WISH I COULD GIVE THIS A 100 PERCENT RATING I LOVE THIS!! ðŸ’• ðŸ˜˜	5	11/29/2022 21:54
6	Loveness Maler	Pictures and record	5	11/29/2022 21:54
7	Melvin Crawfor	I love this amazing app	5	11/29/2022 21:52
8	Muhammad Re	Mohammed Rehan	5	11/29/2022 21:52
9	Brian Nowak	Love being on Tik Tok.	5	11/29/2022 21:52
10	JÃºlia Fideles	Kwai pelo menos da dinheiro	1	11/29/2022 21:51
11	Rocio Salas	Post to view is such a step backwards for the platform -- why punish both your casual users and your co	1	11/29/2022 21:51
12	Lisanna Valvora	I got banned for literally no reason. You guys need to stop banning people! I didn't do anything	1	11/29/2022 21:50
13	hassan ali fawa	Love	5	11/29/2022 21:50
14	Beautiful Ayaan	Good	5	11/29/2022 21:48
15	Thomas J Herrn	I have fun on here.	5	11/29/2022 21:48
16	Joey Legroulx	I don't like when it takes so long to load and or only load one video but other then that I love the app	4	11/29/2022 21:47
17	Me You	I love tik tok but this black out thing is crazy and can even kill don't try the black out	4	11/29/2022 21:47
18	Tik Toker	Very very nice app but my video not going for you	5	11/29/2022 21:47
19	Sandy Mark	pee b-4 watching!!! Luv ur site!!! GO... U. S. A.!!!	5	11/29/2022 21:47
20	M Mustaqeem	Hi dear tik tik is very good app Android mobile and tablet laptop and computer using best app thanks fc	5	11/29/2022 21:46
21	christopher ma	Perfect ðŸ¥°	5	11/29/2022 21:45
22	khadijat Usman	Awesome	5	11/29/2022 21:44
23	zeyda hawkins	my app won't stop crashing its horrible I'm making a video and ut crashes and deletes my progress. get	1	11/29/2022 21:44
24	Jay Dubose	Type shiiðŸ«,	5	11/29/2022 21:43
25	Shadan khan	Good	4	11/29/2022 21:42

Gambar 4.5 *Dataset* Setelah Disederhanakan

Pada gambar 4.5 merupakan gambaran *dataset* yang telah disederhanakan, dengan atribut; *username, content, score, at*.

4.2.2. *Labelling Data*

Labelling data pada penelitian ini digunakan untuk memberikan label atau klasifikasi data *text*, dengan tujuan untuk mengidentifikasi apakah *text* mengandung *sentiment positive*, *negative*, dan netral. Pada penelitian ini *labelling* dilakukan berdasarkan nilai *score*. Pada penelitian ini proses *labelling* menggunakan alat bantu *google colaboratory* menggunakan bahasa pemograman *python*.



```
df= pd.read_csv('DataSentiment.csv')
def convert_sentiment(score):
  if score <=2:
    return 'negative'
  elif score ==3:
    return 'netral'
  else:
    return 'positive'
dataset['sentiment']=dataset['score'].apply(convert_sentiment)
dataset.to_csv('labelling.csv',index=False)
positive_count = dataset[dataset['sentiment']=='positive'].shape[0]
netral_count = dataset[dataset['sentiment']=='netral'].shape[0]
negative_count = dataset[dataset['sentiment']=='negative'].shape[0]
print("jumlah ulasan positive : ", positive_count)
print("jumlah ulasan netral : ", netral_count)
print("jumlah ulasan negative : ", negative_count)
dataset.tail()
```

```
jumlah ulasan positive :  3513
jumlah ulasan netral :  169
jumlah ulasan negative :  720
```

Gambar 4.6 Proses *Labelling data*

Pada gambar 4.6 merupakan proses *labelling data*, yang mulai dengan membaca *file csv* yang berisi data *sentiment*, menggunkanan fungsi *convert_sentiemnt* untuk mengkonversi skore menjadi label *sentiment*, kemudian menambahkan kolom *sentiment* kedalam *dataset*, setalah proses sebelumnya berhasil *dataset* tersebut di simpan dalam bentuk *csv* dan kemudian lakukan perhitungan jumlah ulasan masing-masing *sentiment* dan di lanjutkan menampilkan jumlah ulasan *posistive*, netral, dan *negative*, sehingga dapat kita lihat jumlah ulasan *positive* sebanyak 3513

sentiment, jumlah ulasan netra sebanyak 169 *sentiment*, dan jumlah ulasan negative sebayak 720 *sentiment*.

1	userName	content	score	sentiment	at
2	Cassie Moore	No words	5	positive	11/29/2022 21:55
3	Kaleb Plummer	Great fun app so far!	5	positive	11/29/2022 21:55
4	Rylee Maher	The app would get a higher rating but I lit	1	negative	11/29/2022 21:54
5	Kittykatelyn Romilly	I WISH I COULD GIVE THIS A 100 PERCENT	5	positive	11/29/2022 21:54
6	Loveness Malenga	Pictures and record	5	positive	11/29/2022 21:54
7	Melvin Crawford	I love this amazing app	5	positive	11/29/2022 21:52
8	Muhammad Rehan	Mohammed Rehan	5	positive	11/29/2022 21:52
9	Brian Nowak	Love being on Tik Tok.	5	positive	11/29/2022 21:52
10	JÃºlia Fideles	Kwai pelo menos da dinheiro	1	negative	11/29/2022 21:51
11	Rocio Salas	Post to view is such a step backwards for t	1	negative	11/29/2022 21:51
12	Lisanna Valvora	I got banned for literally no reason. You gu	1	negative	11/29/2022 21:50
13	hassan ali fawaz	Love	5	positive	11/29/2022 21:50
14	Beautiful Ayaan	Good	5	positive	11/29/2022 21:48
15	Thomas J Herrmann	I have fun on here.	5	positive	11/29/2022 21:48
16	Joey Legroulx	I don't like when it takes so long to load ar	4	positive	11/29/2022 21:47
17	Me You	I love tik tok but this black out thing is cra	4	positive	11/29/2022 21:47
18	Tik Toker	Very very nice app but my video not going	5	positive	11/29/2022 21:47
19	Sandy Mark	pee b-4 watching!!! Luv ur site!!! GO... U. S	5	positive	11/29/2022 21:47
20	M Mustaqeem	Hi dear tik tik is very good app Android mo	5	positive	11/29/2022 21:46

Gambar 4.7 *Dataset* Yang Telah Di-*labelling*

Pada gambar 4.7 merupakan hasil *dataset* yang telah di-*labelling* dengan label *positive, negative,* dan netral.

4.3. *PREPROCESSING*

Pada penelitian ini dilakukan *preprocessing* dengan tujuan menghasilkan *dataset* yang bersih, rapih, sehingga siap digunakan untuk melakukan *sentiment analysis*. pada penelitian ini *preprocessing* yang digunakan adalah *case folding, text cleaning, stopword removal, tokenize,* dan *stemming.*

4.3.1. *Case Folding*

Pada penelitian ini dilakukan *case folding*, yang merubah kalimat menjadi huruf kecil (*lowercase*), dengan tujuan mengatasi perbedaan huruf

besar dan kecil agar memudahkan proses penelitian *sentiment analysis* terhadap pengguna *tiktok* di *google play store*.

```
#case folding

dataset['case_folding']=dataset['content'].str.lower()
dataset.to_csv('case_folding.csv',index=False)
```

Gambar 4.8 Proses *Case Folding*

pada gambar 4.8 merupakan proses *case folding*. Proses di mulai dari menambahkab kolom *case folding* ke dalam *dataset*, kemudian menyimpan *dataset* yang telah mengalami *case folding* kedalam *file csv* baru yaitu *case folding.csv*.

Tabel 4.1 Hasil *Case Folding*

Content	***Case Folding***
Perfect	*perfect*
Awesome	*awesome*
Great fun app so far!	*great fun app so far!*
nice app. definitely keeps me from being 100% bored.	*nice app. definitely keeps me from being 100% bored.*
Pictures and record	*pictures and record*

4.3.2. *Text Cleaning*

Pada penelitian ini dilakukan *text cleaning* yang bertujuan untuk menghapus karakter yang tidak relevan dalam *sentiment analysis*, seperti menghapus emoji, angka, *html*, *url*, *hastag*, *metion*, *link*.

```
import string
import re
dataset = pd.read_csv('case_folding.csv')
dataset['case_folding']=dataset['case_folding'].astype(str)
def text_cleaning(text):
  #remove tab, new line, ans back slice
  text = text.replace('\\t'," ").replace('\\n'," ").replace('\\u'," ").replace('\\', "")
  #remove non ASCII(emoticon, chinese word, .etc)
  text = text.encode('ascii','replace').decode('ascii')
  #remove mention, link , hastag
  text = ' '.join(re.sub("([@#][A-Za-z0-9]+|(w+:\/\/\s+))"," ", text).split())
  text = re.sub(r'\d+', '', text)
  characters_to_remove = '_?,.!()/;:*#-_@="<>^[$]{}+%$&'
  for char in characters_to_remove:
      text = text.replace(char, ' ')
  character_to_remove = "'|"
  for char in character_to_remove:
      text = text.replace(char, ' ')
  #remove incomplate URL
  return text.replace("http://", " ").replace("https://"," ")
dataset['text_cleaning'] = dataset['case_folding'].apply(text_cleaning)
dataset.to_csv('text_cleaning.csv', index=False)
```

Gambar 4.9 Proses *Text Cleaning*

Pada gambar 4.9 merupakan proses *text cleaning*, yang dimulai dari mengimport beberapa *library* yang di butuhkan, kemudian membaca *file csv* hasil *case folding*, kemudian mengkonversi kolom *case folding* ke tipe *string* jika belum, kemudian menghapus karakter *non-ASCII* (emotikon, kata tionghoa, dll), kemudian menghapus *mention*, *link* dan *hastag*, di lanjtukan menghapus angka, mengahapus karakter khusus, serta menghapus *URL* yang tidak lengkap, setelah proses semua berhasil dilakukan menambah kolom *text cleaning* kedalam *dataset* dengan menerapkan fungsi *text_cleaning*, dan di lanjutkan menyimpan hasil *cleaning text* ke dalam *csv* baru dengan nama *file text cleaning.csv*.

Tabel 4.2 Hasil *Text Cleaning*

Case Folding	***Text Cleaning***
perfect	*perfect*
awesome	*awesome*
great fun app so far!	*great fun app so far*
nice app. definitely keeps me from being 100% bored.	*nice app definitely keeps me from being bored*
pictures and record	*pictures and record*

4.3.3. *Stopword Removal*

Pada penelitian ini, dilakukan *stopword removal. Stopword removal* yaitu proses menghilangkan kata-kata umum yang biasanya tidak memberikan kontribusi signifikan terhadap pemahaman teks atau informasi. Hal ini dilakukan dengan tujuan meningkatkan kualitas teks. Berikut ini merupakan proses *stopword removal*.

Gambar 4.10 Proses *Stopword Removal*

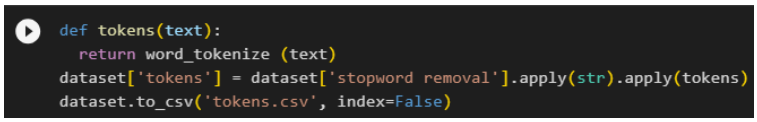
Pada gambar 4.10 merupakan proses *stopword removal*, di mulai dengan mengimport beberapa *library* yang di butuhkan, kemudian lanjutkan membaca *dataset* dari *file csv*, kemudian mengambil daftar *stopword* untuk bahasa *inggris*, lalu di lanjutkan membuat kolom baru yaitu *stopword removal* yang berisi *text* tanpa *stopword*, setelah proses tersebut selesai maka di lanjutkan untuk menyimpan *dataset* yang telah di ubah kedalam *csv* baru dengan nama *file stopword_removal.csv*.

Tabel 4.3 Hasil *Stopword Removal*

Text Cleaning	***Stopword Removal***
perfect	*perfect*
awesome	*awesome*
great fun app so far	*great fun app far*
nice app definitely keeps me from being bored	*nice app definitely keeps bored*
pictures and record	*pictures record*

4.3.4. *Tokenization*

Pada penelitian ini, dilakukan *tokenization. Tokenization* adalah proses mengkonversi *text* atau dokumen menjadi serangkaian token, yang berujuan untuk memudahkan pemrosesan yang lebih lanjut.



Gambar 4.11 Proses *Tokenization*

Pada gambar 4.11 merupakan proses *tokenization*, di mulai dari mendefenisikan fungsi kode yang menggunakan *word_tokenize* dari *nltk*, dilanjutkan menerapkan fungsi *tokens* pada kolom *stopword removal* dari *dataset*, setelah proses berhasil maka disimpan *dataset* tersebut yang telah ditokenisasi kedalam *file cvs* baru yaitu *tokens.csv*.

Tabel 4.4 Hasil *Tokenization*

Stopword Removal	***Tokenization***
perfect	*[‘perfect’]*
awesome	*[‘awesome’]*
great fun app far	*['great', 'fun', 'app', 'far']*
nice app definitely keeps bored	*['nice', 'app', 'definitely', 'keeps', 'bored']*
pictures record	*['pictures', 'record']*

4.3.5. *Stemming*

Pada penelitian ini, dilakukan *stemming. Stemming* adalah proses mengkonversi *text* atau dokumen menjadi serangkaian token dengan menghilangkan imbuhan kata sehingga menyisakan bentuk dasar, sehingga

dapat meningkatkan efisiensi dan konsistensi dalam pemrosesan *text*. Berikut merupakan proses *stemming*.

```
import pandas as pd
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
dataset = pd.read_csv('tokens.csv')
porter = PorterStemmer()
dataset['stemming text'] = dataset['tokens'].apply(lambda x: ' '.join([porter.stem(word) for word in word_tokenize(x)]))
dataset.to_csv('stemming.csv', index=False)
```

Gambar 4.12 Proses *Stemming*

Pada gambar 4.12 merupakan proses *stemming.* dimulai dengan mengimport *library* yang di perlukan, dilanjutkan dengan membaca *dataset* yang telah dikontenisasi dari *file csv*, kemudian membuat objek *PorterStemmer* untuk melakukan *stemming*, setelah itu merapkan proses *stemming* pada kolom *token* dari *dataset*, setelah proses selesai di lanjutkan menyimpan *dataset* yang telah mengalami proses *stemming* ke dalam *file* baru dengan nama *file stemming.csv*

Tabel 4.5 Hasil *Stemming*

Tokenization	***Stemming***
[‘perfect’]	*[‘perfect’]*
[‘awesome’]	*['awesom']*
['great', 'fun', 'app', 'far']	*['great ' , 'fun ' , 'app ' , 'far ']*
['nice', 'app', 'definitely', 'keeps', 'bored']	*['nice ' , 'app ' , 'definit ' , 'keep ' , 'bore ']*
['pictures', 'record']	*['pictur ' , 'record ']*

4.4. *WEIGHTING*

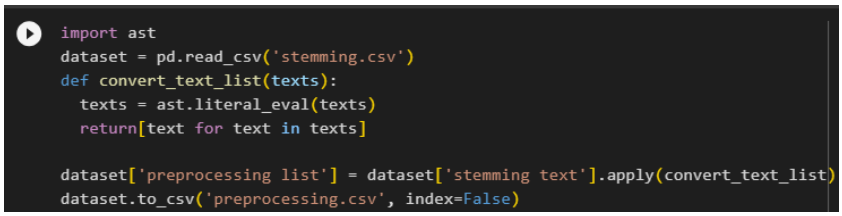
Weighting adalah proses memberikan nilai atau bobot pada suatu elemen atau fitur dalam *dataset*, pada penelitian ini *weighting* yang digunakan adalah *TF-IDF*.

4.4.1. *TF-IDF*

TF-IDF adalah singkatan dari *term frequency-inverse document frequency*, yang merupakan metode untuk memberikan bobot pada kata-kata dalam suatu koleksi dokumen. *TF-IDF* bertujuan untuk menyortir kata-kata yang memiliki nilai informatif tinggi dan membedekan dokumen satu dengan yang lain. Berikut adalah proses *TF-IDF*:

a. *Convert text to list*

Pada penelitian ini proses *convert text to list* digunakan untuk mempermudah menghitung frekuensi kata-kata (*TF*) dan bobot *TF-IDF* untuk setiap kata dalam dokumen.



Gambar 4.13 Proses *Convert text to list*

Pada gambar 4.13 merupakan proses konversi teks ke dalam *list*, dimulai dengan mengimport *library* yang di perlukan, kemudian membaca *dataset* dari *file csv* yang telah melalui proses *stemming*, selanjutnya mendefinisikan fungsi untuk mengkonversi *text* ke dalam bentuk *list*, dilanjutkan menerapkan fungsi kolom *stemming text* mengkonversi kedalam bentuk list, selanjutnya *dataset* yang telah melalui proses *text to list* kedalam *file csv* baru dengan nama *file preprocessing.csv.*

Tabel 4.6 Hasil *Convert to list*

***preprocessing* list**
['perfect ']
['awesom ']
['great ', 'fun ', 'app ', 'far ']
['nice ', 'app ', 'definit ', 'keep ', 'bore ']
['pictur ', 'record ']

b. *Trem Frequency (TF)*

trem frequency adalah ukuran yang menunjukan seberapa sering sebuah kata muncul dalam sebuah dokumen, tujuan dari *TF* adalah untuk memberikan bobot pada kata-kata berdasarkan frekuensinya didalam dokumen tersebut, berikut adalah proses *TF*:

```
def calc_TF(document):
  TF_dict = {}
  for term in document:
    if term in TF_dict:
      TF_dict[term] += 1
    else:
      TF_dict[term] = 1
  for term in TF_dict:
    TF_dict[term] = TF_dict[term]/len(document)
  return TF_dict
dataset['TF_dict'] = dataset['preprocessing list'].apply(calc_TF)
dataset.to_csv('TF_dict.csv', index=False)
```

Gambar 4.14 Proses *Trem Frequency*

Pada gambar 4.14 merupakan proses *Term Frequency (TF).* Langkah-langkahnya dimulai dengan membuat fungsi `*calc_TF*` dibuat untuk menghitung dan normalisasi *Term Frequency (TF)* pada setiap term dalam suatu dokumen. Fungsi ini diterapkan pada kolom *'preprocessing list'* dalam *dataframe 'dataset'*, dan hasilnya disimpan dalam kolom baru *'TF_dict'*. Selanjutnya, *dataframe 'dataset'* disimpan dalam *file CSV* bernama *'TF_dict.csv'*.

Tabel 4.7 Hasil *Trem Frequency*

***preprocessing* list**	***Trem Frequency***
['perfect ']	*{'perfect ': 1.0}*
['awesom ']	*{'awesom ': 1.0}*
['great ', 'fun ', 'app ', 'far ']	*{'great ': 0.25, 'fun ': 0.25, 'app ': 0.25, 'far ': 0.25}*
['nice ', 'app ', 'definit ', 'keep ', 'bore ']	*{'nice ': 0.2, 'app ': 0.2, 'definit ': 0.2, 'keep ': 0.2, 'bore ': 0.2}*
['pictur ', 'record ']	*{'pictur ': 0.5, 'record ': 0.5}*

c. *Document Frequency (DF)*

Document frequency adalah mengukur seberapa sering suatu kata tertentu muncul di seluruh *document*. *Document frequency* memberi kontribusi untuk mengevaluasi seberapa umum atau langka sebuah kata dalam *document*, semakin tinggi nilai *DF* maka semakin umum kata tersebut dalam koleksi. Berukut ini peroses *document frequency*:

```
def calc_DF(tfDict):
  count_DF= {}
  for document in tfDict:
      for term in document:
        if term in count_DF:
          count_DF[term] += 1
        else:
          count_DF[term] = 1
  return count_DF
DF = calc_DF(dataset["TF_dict"])
df_result = pd.DataFrame(list(DF.items()), columns=['Term', 'Document Frequency'])
df_result.to_csv('calc_DF.csv', index=False)
```

Gambar 4.15 Proses *Document Frequency*

Pada gambar 4.15 merupakan proses *Document Frequency*, dimuali dengan membuat fungsi *`calc_DF (tfDict)`*. Fungsi ini menerima kamus *Term Frequency (TF)* dan menghitung frekuensi dokumen untuk setiap *term*. Saat mengiterasi melalui dokumen dan *term* dalam kamus tersebut, jumlah dokumen yang mengandung

setiap *term* disimpan dalam kamus *`count_DF`*.Selanjutnya, kamus *`count_DF`* diubah menjadi dataframe menggunakan *Pandas* dan disimpan dalam *file CSV* dengan nama *'calc_DF.csv'*

Tabel 4.8 Hasil *Document Frequency*

Trem	***Document Frequency***
perfect	*19*
awesom	*72*
great	*210*
fun	*113*
app	*1196*
far	*16*
nice	*415*
definit	*8*
keep	*44*
bore	*18*
pictur	*3*
record	*6*

d. *Inverse Document Frequency (IDF)*

Inverse Document Frequency adalah nilai bobot yang digunakan untuk menimbang seberapa penting suatu *term* dalam dokumen. *IDF* memberi bobot lebih tinggi pada term yang langka dan bobot lebih rendah pada *term* yang umum dalam seluruh koleksi *document*. Berikut adalah proses *IDF*:

```
import numpy as np
n_document = len(dataset)

def calc_IDF(__n_document, __DF):
  IDF_Dict = {}
  for term in __DF:
    IDF_Dict[term] = np.log(__n_document /(__DF[term]+1))
  return IDF_Dict
IDF = calc_IDF(n_document, DF)
df_result = pd.DataFrame(list(IDF.items()), columns=['Term', 'IDF'])
df_result.to_csv('calc_IDF.csv', index=False)
```

Gambar 4.16 Proses *Inverse Document Frequency*

Pada gambar 4.15 proses *inverse document frequency*, dimulai dari ini mengimport *NumPy*. Kemudian jumlah dokumen dalam *dataset* dihitung, kemudian fungsi `*calc_IDF*` digunakan untuk menghitung dan menyimpan nilai *IDF* untuk setiap *term*. Hasilnya dikonversi ke dalam *dataframe* dan disimpan dalam *file CSV 'calc_IDF.csv'*.

Tabel 4.9 Hasil *Inverse Document Frequency*

Trem	***Inverse Document Frequency***
perfect	5.39408198853241
awesom	4.09935482093801
great	3.03795612861034
fun	3.65361581369191
app	1.30224055652843
far	5.55660091803019
nice	2.35912900182514
definit	6.19258968475018
keep	4.58315177231608
bore	5.44537528291996
pictur	7.00351990096651
record	6.44390411303109

e. *Term Frequency-Inverse Document Frequency*

TF-IDF digunakan untuk mengevaluasi dan menyorti kata-kata berdasarkan tingkat kepentingan dalam suatu *dokument*. Semakin tinggi *score TF-IDF* suatu kata dalam dokumen, semakin penting kata tersebut untuk dokumen tersebut. Berikut ini proses dari *Term Frequency-Inverse Document Frequency (TF-IDF):*

```
def calc_TF_IDF(TF):
    TF_IDF_dict = {}
    for key in TF:
        TF_IDF_dict[key] = TF[key] * IDF[key]
    return TF_IDF_dict
dataset["TF-IDF_dict"] = dataset["TF_dict"].apply(calc_TF_IDF)
df_result = pd.DataFrame(list(IDF.items()), columns=['Term', 'TF-IDF_dict'])
index =19
print('%20s' % "term", "\t", "%10s" % "TF", "\t", '%20s' % "TF-IDF\n")
```

```
for key in dataset['TF-IDF_dict'][index]:
    print('%20s' % key, "\t", '%10s' % dataset['TF_dict'][index][key], "\t", '%20s' % dataset['TF-IDF_dict'][index][key])
```

Gambar 4.17 Proses *TF-IDF*

Pada gambar 4.16 merupakan proses *TF-IDF*, Fungsi `*calc_TF_IDF*` digunakan untuk mengalikan *Term Frequency (TF)* dengan *Inverse Document Frequency (IDF)* untuk setiap *term* dalam *dataset*. Hasil perhitungan disimpan dalam kolom baru *"TF-IDF_dict"*, dan informasi tersebut juga disimpan dalam *file CSV* dengan nama *"calc_TF-IDF.csv"*.

Tabel 4.10 Hasil *TF-IDF*

Trem	***TF***	***Document Frequency***	***TF-IDF***
perfect	1.0	5.39408198853241	5.394081988532416
awesom	1.0	4.09935482093801	4.099354820938016
great	0.25	3.03795612861034	0.7594890321525851
fun	0.25	3.65361581369191	0. 9134039534229779
app	0.25	1.30224055652843	0.32556013913210846
far	0.25	5.55660091803019	1.3891502295075477
nice	0.2	2.35912900182514	0.4718258003650288
app	0.2	1.30224055652843	0.238517936500375
definit	0.2	6.19258968475018	1.2385179369500374
keep	0.2	4.58315177231608	0.9166303544632175
bore	0.2	5.44537528291996	1.089075056589935
pictur	0.5	7.00351990096651	3.501759950483258
record	0.5	6.44390411303109	3.221952056515547

f. Representasi Numerik

Representasi numerik menggunakan *TF-IDF* adalah suatu metode yang digunakan untuk mengonversi *text* atau dokumen menjadi *vector* numerik, metode ini memperhitungkan seberapa sering sebuah kata muncul dalam suatu dokumen *(TF)* dan seberapa unik kata tersebut dalam keseluruhan koleksi *document (IDF).*

```
sorted_DF = sorted(DF.items(), key = lambda kv:kv[1], reverse=True)[:50]
unique_term = [item[0] for item in sorted_DF]
def calc_TF_IDF_Vec(__TF_IDF_Dict):
  TF_IDF_vector = [0.0] * len(unique_term)
  for i, term in enumerate(unique_term):
    if term in __TF_IDF_Dict:
      TF_IDF_vector[i] = __TF_IDF_Dict[term]
    return TF_IDF_vector
dataset["TF_IDF_Vec"] = dataset["TF-IDF_dict"].apply(calc_TF_IDF_Vec)
```

Gambar 4.18 Proses Representasi Numerik

Pada gambar 4.17 merupakan proses representasi numerik dimulai dengan proses pengurutan untuk memilih 50 kata dengan frekuensi tertinggi dalam *dataset*. Selanjutnya, daftar kata unik dibuat dari kamus yang telah diurutkan. Setelah pengurutan berhasil, langkah berikutnya adalah menghitung vektor *TF-IDF* untuk setiap dokumen. Seluruh proses ini kemudian diimplementasikan pada setiap elemen dalam kolom "*TF-IDF_dict*" dari *dataset*.

Tabel 4.11 Hasil *TF-IDF*

Trem	***Vector TF-IDF***
[' perfect ']	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[' awesom ']	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,

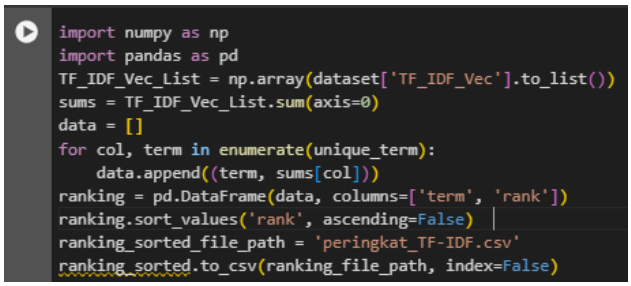
	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
['great ', 'fun ', 'app ', 'far ']	[0.32556013913210846, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
['nice ', 'app ', 'definit ', 'keep ', 'bore ']	[0.2604481113056868, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
['pictur ', 'record ']	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Pada gambar 4.11, vektor *TF-IDF* menunjukkan beberapa karakteristik kunci. Nilai 0.0 pada vektor pertama dengan *term 'perfect'* dan vektor kedua dengan *term 'awesome'* menandakan ketidakmunculan atau bobot rendah dari term tersebut dalam dokumen. Sementara vektor ketiga dan keempat menunjukkan bobot *TF-IDF* yang lebih tinggi, mencerminkan pentingnya *term* tersebut dalam dokumen dibandingkan dengan korpus secara keseluruhan.

g. Peringkat *TF-IDF*

Peringkat *TF-IDF* adalah nilai yang menggambarkan seberapa pentig suatu kata dalam suatu dokumen dalam konteks Kumpulan

dokumen yang lebih besar. Peringkat *TF-IDF* membantu menyaring dan memberi bobot pada kata-kata berdasarkan seberapa unik atau spesifik terhadap suatu dokumen.



```
import numpy as np
import pandas as pd
TF_IDF_Vec_List = np.array(dataset['TF_IDF_Vec'].to_list())
sums = TF_IDF_Vec_List.sum(axis=0)
data = []
for col, term in enumerate(unique_term):
    data.append((term, sums[col]))
ranking = pd.DataFrame(data, columns=['term', 'rank'])
ranking.sort_values('rank', ascending=False)
ranking_sorted_file_path = 'peringkat_TF-IDF.csv'
ranking_sorted.to_csv(ranking_file_path, index=False)
```

Gambar 4.19 Proses Perhitungan Peringkat *TF-IDF*

Pada gambar 4.19 merupakan proses perhitungan peringkat *TF-IDF* yang dimulai dari mengimport *numpy* dan *pandas*, di lanjutkan mengambil *column 'TF_IDF_Vec* dari *dataset* yang kemudian di konversi menjadi *array* dan setiap baris dalam *array* ini mewakili *vector TF-IDF* dari dokumen, kemudian di lanjutkan dengan menjumlahkan elemen-elemen *vector*, kemudian di gunakan '*for'* untuk iterasi melalui *('unique_term'*) ke daftar data, kemudian dilanjutkan membuat dataframe rangking dari data yang dikumpulkan sebelumnya yang memiliki 2 *column* yaitu *term* dan *rank* dan kemudian diurutkan berdasarkan nilai *rank* secara menurun. Dan hasil peringkat tersebut disimpan dalam sebuah *file csv* dengan nama *file* peringkat_*TF-IDF.csv*

Tabel 4.12 Hasil Peringkat *TF-IDF*

Term	Rank
good	747.147909000591
nice	624.521382974317

app	515.755942563035
love	401.66823302773
great	315.983967429397
amaz	258.587065329001
best	258.406385976193
tiktok	245.582781405467
awesom	214.768334217651
like	171.587179508838
wow	169.631909356298
excel	166.301320536763
fun	138.575698178852
video	135.855313532179
entertain	123.683937029819
cool	115.728030629191
pleas	110.681763392822
account	107.401758521909
tik	106.329560597232
tok	105.097240123071
enjoy	87.5954366545184
viral	82.1903953999602
thank	80.9904771248253
use	72.1596527922786
much	70.8155512905699
realli	65.7404917739402
time	58.1873332905567
make	55.9562669199407
work	55.6636700191844
ever	54.4449363802652
follow	51.4932037313624
view	47.0427858799512
help	45.7958429351812
updat	45.2221409180749
get	44.011285673461
team	41.6158715765979
go	41.3463126000457
peopl	40.759995378009
one	38.5451534613443
new	34.5719920652449
problem	31.9691670043696

watch	31.4510181482939
ban	29.5463157836575
thing	26.4955912239172
give	25.0931321289078
want	23.4524564363121
need	22.8216095054943
mani	22.6859451234025
tri	20.7302341119336
fix	15.1874435071633

h. *Token To String*

Token to string digunakan untuk mengabungkan daftar token sebagai *string* dokumen tunggal, yang kemudian digunakan untuk untuk menghitung *vector TF*, berikut ini proses *token to string*.

```
import ast
def join_text_list(texts):
  texts = ast.literal_eval(texts)
  return ' '. join([text for text in texts])
dataset['preprocessing join'] = dataset['stemming text'].apply(join_text_list)
dataset.to_csv('preprocessing_.csv', index=False)
```

Gambar 4.20 Proses *Token To String*

Pada gambar 4.20 merupakan proses token to *string*, *Code* ini digunakan untuk *preprocessing DataFrame* `*dataset*` dengan menggabungkan hasil *stemming* dari kolom *'stemming text'* menjadi satu *string* dalam kolom baru *'preprocessing join*'. Setelah itu, *DataFrame* dimodifikasi tersebut disimpan dalam *file CSV* dengan nama *'preprocessing_.csv'*.

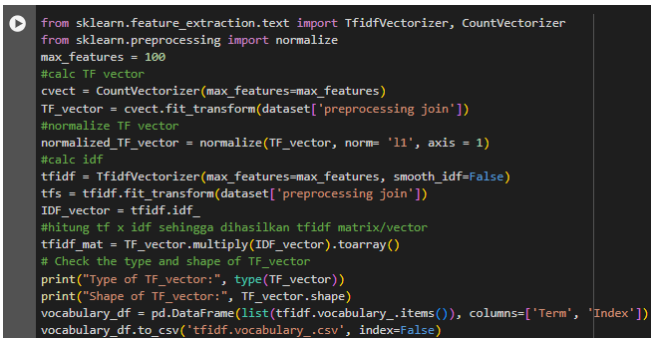
Tabel 4.13 Hasil *Token To String*

Stemming Text	***Preprocessing join***
['perfect ']	*perfect*
['awesom ']	*awesom*
['great ', 'fun ', 'app ', 'far ']	*great fun app far*

['nice ' , 'app ' , 'definit ' , 'keep ' , 'bore ']	*nice app definit keep bore*
['pictur ', 'record ']	*pictur record*

i. Ekstrasi fitur

Ekstrasi fitur adalah suatu metode yang digunakan untuk mengukur pentingnya sebuah kata dalam suatu dokumen dalam konteks kumpulan dokumen. Digunakan dalam pemrosesan teks, tujuannya memberikan nilai numerik pada setiap kata, mencerminkan tingkat kepentingan dalam seluruh kumpulan dokumen. Dokumen dengan kata-kata kunci yang spesifik dan relevan akan memiliki skor *TF-IDF* tinggi untuk kata-kata tersebut.

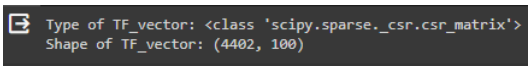


```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import normalize
max_features = 100
#calc TF vector
cvect = CountVectorizer(max_features=max_features)
TF_vector = cvect.fit_transform(dataset['preprocessing join'])
#normalize TF vector
normalized_TF_vector = normalize(TF_vector, norm= 'l1', axis = 1)
#calc idf
tfidf = TfidfVectorizer(max_features=max_features, smooth_idf=False)
tfs = tfidf.fit_transform(dataset['preprocessing join'])
IDF_vector = tfidf.idf_
#hitung tf x idf sehingga dihasilkan tfidf matrix/vector
tfidf_mat = TF_vector.multiply(IDF_vector).toarray()
# Check the type and shape of TF_vector
print("Type of TF_vector:", type(TF_vector))
print("Shape of TF_vector:", TF_vector.shape)
vocabulary_df = pd.DataFrame(list(tfidf.vocabulary_.items()), columns=['Term', 'Index'])
vocabulary_df.to_csv('tfidf.vocabulary_.csv', index=False)
```

Gambar 4.21 Proses Ekstrasi Fitur

Pada gambar 4.21 merupakan proses ekstraksi fitur dimulai dengan meng*import* modul `*TfidfVectorizer*` dan `*CountVectorizer*`. Selanjutnya, dilakukan normalisasi vektor dan penetapan jumlah fitur yang akan diekstrak. Setelah itu, vektor frekuensi *Term Frequency (TF)* dihitung, kemudian dilakukan normalisasi vektor menggunakan metode L1. Setelah semua tahapan berhasil, dilanjutkan dengan menghitung matriks *TF-IDF*. Terakhir, jenis

dari vektor *TF (`TF_vector`)* ditampilkan bersama dengan informasi mengenai jumlah baris dan kolom dari vektor tersebut.



Gambar 4.22 Informasi Ekstraksi Fitur

Tabel 4.14 Hasil Ekstraksi Fitur

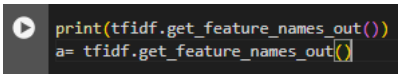
Term	Index
great	38 *Term*
fun	31 *Term*
app	5 *Term*
would	98 *Term*
get	33 *Term*
tri	83 *Term*
freez	29 *Term*
phone	60 *Term*
fix	27 *Term*
know	44 *Term*
new	55 *Term*
problem	65 *Term*
give	34 *Term*
Love	49 *Term*
Amaz	3 *Term*
Tik	79 *Term*
Tok	82 *Term*
Post	64 *Term*
View	89 *Term*
Platform	61 *Term*
User	67 *Term*
Got	37 *Term*
Ban	9 *Term*
Reason	67 *Term*
Need	54 *Term*
People	59 *Term*
Good	36 *Term*
Like	46 *Term*
Take	75 *Term*

One	57 *Term*
Video	88 *Term*
Thing	78 *Term*
Even	21 *Term*
nice	56 *Term*
Go	35 *Term*
Watch	94 *Term*
Dear	16 *Term*
Android	4 *Term*
Use	86 *Term*
Best	10 *Term*
Thank	77 *Term*
Team	76 *Term*
Awesome	6 *Term*
Make	50 *Term*
tiktok	80 *Term*
Plz	63 *Term*
Help	40 *Term*
Account	0 *Term*
Unfreez	84 *Term*
Alway	2 *Term*
Enjoy	19 *Term*
Funni	32 *Term*
Cool	14 *Term*
Lot	48 *Term*
realli	66 *Term*
Friend	30 *Term*
star	72 *Term*
Keep	43 *Term*
Everi	23 *Term*
Let	45 *Term*
Pleas	62 *Term*
Entertain	20 *Term*
follow	28 *Term*
filter	26 *Term*
Work	96 *Term*
Voic	91 *Term*
bad	8 *Term*
Live	47 *Term*

Way	95 *Term*
Much	52 *Term*
Want	92 *Term*
See	69 *Term*
Time	81 *Term*
Ever	22 *Term*
Also	1 *Term*
Say	68 *Term*
Show	70 *Term*
Comment	13 *Term*
Better	11 *Term*
Excel	25 *Term*
Mani	51 *Term*
Option	58 *Term*
Interest	42 *Term*
Warn	93 *Term*
Still	73 *Term*
Updat	85 *Term*
Viral	90 *Term*
Back	7 *Term*
Wow	99 *Term*
Easi	18 *Term*
Download	17 *Term*
Support	74 *Term*
Come	12 *Term*
World	97 *Term*
Happi	39 *Term*
everyon	24 *Term*
Sir	71 *Term*
Name	53 *Term*
Day	15 *Trem*
id	41 *Trem*

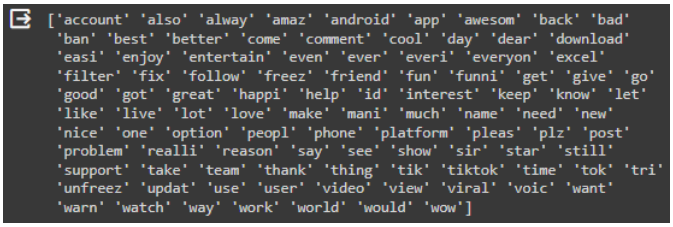
j. Daftar Kata Kunci

Daftar kata kunci yang dianggap penting berasal dari dokumen atau teks yang telah diproses menggunakan metode *TF-IDF. TF-IDF* adalah sebuah teknik yang digunakan untuk menilai signifikansi sebuah kata dalam suatu dokumen, berdasarkan seberapa sering kata tersebut muncul di dokumen tersebut dan seberapa jarang kata tersebut muncul dalam seluruh koleksi dokumen.



Gambar 4.23 Proses menampilkan daftar kata kunci

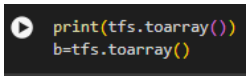
Pada gambar 4.23, terdapat proses untuk menampilkan daftar kata kunci yang dimulai dari ekstraksi kata kunci menggunakan *`tfidf.get_feature_names_out()`* dalam pemrosesan teks menggunakan *TF-IDF*. Hasilnya adalah daftar kata kunci yang diurutkan berdasarkan indeks. Proses ini kemudian dilanjutkan dengan menyimpan daftar kata kunci dalam variabel `a`.



Gambar 4.24 Daftar Kata Kunci

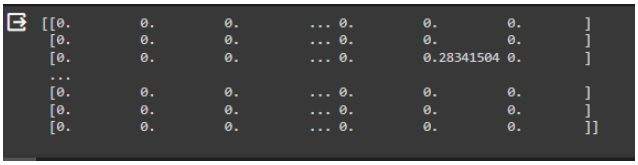
k. Mencetak hasil tranfrormasi *matrix*

transformasi *TF-IDF* adalah untuk mengekstrak informasi penting dari teks, mengidentifikasi kata kunci yang membedakan suatu dokumen dari yang lain, dan mengurangi dampak kata-kata umum yang muncul dalam banyak dokumen. Hal ini sering digunakan dalam analisis teks, pengelompokan dokumen, dan pemahaman isi dokumen secara lebih mendalam.



Gambar 4.25 proses mencetak hasil transformasi *matrix*

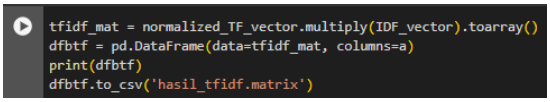
Pada gambar 4.25 merupakan proses menampilkan hasil tranformasi *matrix*, yang di mulai mencetak representasi *matrix* dari hasil transformasi *TF-IDF* ke dalam bentuk *array*, dan hasilnya disimpan untuk melanjutkan tahapan berikutnya.



Gambar 4.26 Hasil Cetak Transformasi *Matrix*.

l. Representasi *Matrix TF-IDF*

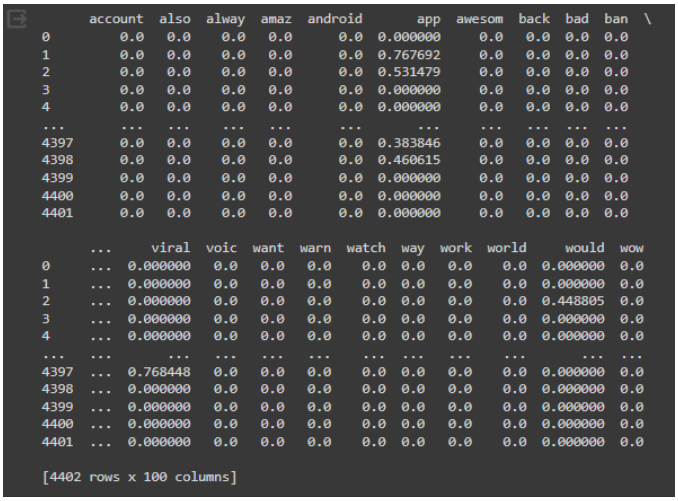
Matriks *TF-IDF* adalah representasi numerik teks dalam pemrosesan bahasa alami, memberikan bobot untuk setiap kata dalam dokumen berdasarkan *Term Frequency (TF)* dan *Inverse Document Frequency (IDF).*



```
tfidf_mat = normalized_TF_vector.multiply(IDF_vector).toarray()
dfbtf = pd.DataFrame(data=tfidf_mat, columns=a)
print(dfbtf)
dfbtf.to_csv('hasil_tfidf.matrix')
```

Gambar 4.27 Proses Representasi Matrik *TF-IDF*

Pada gambar 4.27 merupakan proeses representasi *matrix TF-IDF,* yang dimulai dengan menghitung *matrix TF-IDF* dengan mengalikan *normalized_TF_vector* dan *IDF_vector,* selanjutnya membuat *dataframe* dari matrik *TF-IDF* dengan kolom-kolom dengan variabel a, untuk memastikan berhasil atau tidaknya di lakukan penampilan *output* dan hasil *output* di simpan dalam bentuk *csv,* dengan nama *file* hasil_*tf_idf.matrix*.



	account	also	alway	amaz	android	app	awesom	back	bad	ban	\
0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.767692	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.531479	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
4397	0.0	0.0	0.0	0.0	0.0	0.383846	0.0	0.0	0.0	0.0	
4398	0.0	0.0	0.0	0.0	0.0	0.460615	0.0	0.0	0.0	0.0	
4399	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	
4400	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	
4401	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	

	...	viral	voic	want	warn	watch	way	work	world	would	wow
0	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
1	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
2	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.448805	0.0
3	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...
4397	...	0.768448	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4398	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4399	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4400	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4401	...	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0

[4402 rows x 100 columns]

Gambar 4.28 Matrik *TF-IDF*

Pada gambar 4.28 merupakan hasil matrik *tf-idf,* untuk sejumlah *term* dalam dokumen atau korpus. Setiap baris mewakili satu dokumen, dan setiap kolom mewakili suatu *term* dengan nilai *TF-IDF* yang dihitung. Nilai-nilai ini menunjukkan seberapa pentingnya setiap *term* dalam dokumen dibandingkan dengan

korpus secara keseluruhan. Misalnya, nilai 0.0 menandakan ketidakmunculan *term* dalam dokumen, sementara nilai yang lebih tinggi menunjukkan tingkat kepentingan yang lebih besar.

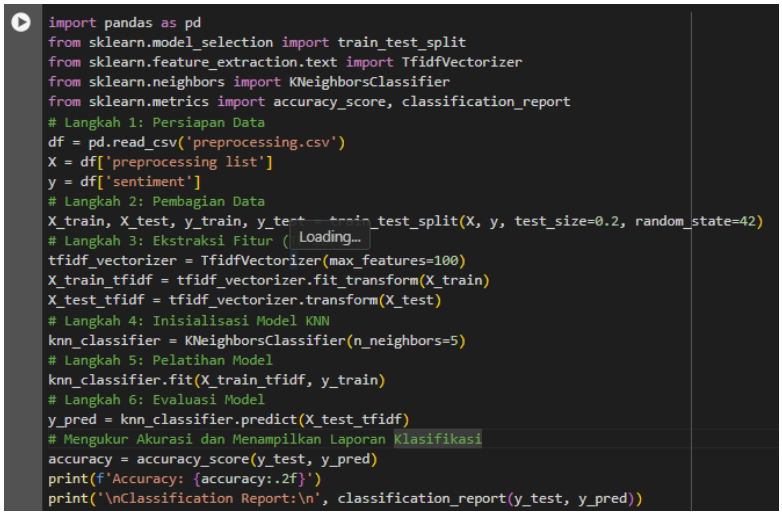
4.5 KLASIFIKASI

Kalasifikasi adalah proses pengelompokan *text* atau dokumen ke dalam kategori *sentiment* tertentu, seperti *positive, negative,* netral. Pada penelitian ini dilakukan pelabelan atau *labelling* untuk mengetahui *sentiment* dengan menggunakan *score*, untuk *sentiment positive* diukur dengan *score* 4-5, *sentiment* netral digunakan dengan *score* 3, dan untuk *sentiment negative* digunakan *score* 1-2. Pada penelitian ini klasifikasi yang digunakan adalah algoritma *k-nearest neighbor* dan *naïve bayes*.

a. *K-Nearest Neighbor*

Pada penelitian ini di lakukan beberapa proses jika menggunakan algoritma *K-Nearest Neighbor*, yaitu persipan data, representasi *text*, pembagian data, inisialisasi model, pelatihan model, serta evaluasi model, dan prediksi *sentiment* baru. Berikut ini penerapan *K-NN* dalam *sentiment analysis*:

1. Data Pelatihan 80% Dan Data Uji 20%

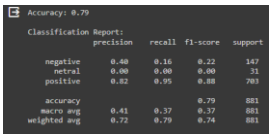


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
# Langkah 1: Persiapan Data
df = pd.read_csv('preprocessing.csv')
X = df['preprocessing list']
y = df['sentiment']
# Langkah 2: Pembagian Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Langkah 3: Ekstraksi Fitur (
tfidf_vectorizer = TfidfVectorizer(max_features=100)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
# Langkah 4: Inisialisasi Model KNN
knn_classifier = KNeighborsClassifier(n_neighbors=5)
# Langkah 5: Pelatihan Model
knn_classifier.fit(X_train_tfidf, y_train)
# Langkah 6: Evaluasi Model
y_pred = knn_classifier.predict(X_test_tfidf)
# Mengukur Akurasi dan Menampilkan Laporan Klasifikasi
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print('\nClassification Report:\n', classification_report(y_test, y_pred))
```

Loading...

Gambar 4.29 Proses Penerapan *K-NN* Dengan 80%Data Latih Dan 20% Data Uji

Pada Gambar 4.29, proses penerapan *K-NN* dengan pembagian data latih 80% dan data uji 20%, dimulai dari persiapan data, pembagian data, ekstraksi fitur *TF-IDF*, inisialisasi model *K-NN* (K=5), pelatihan model, dan evaluasi kinerja.



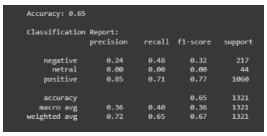
```
Accuracy: 0.79

Classification Report:
              precision    recall  f1-score   support

    negative       0.40      0.16      0.22       147
     netral        0.00      0.00      0.00        31
    positive       0.82      0.95      0.88       703

    accuracy                           0.79       881
   macro avg       0.41      0.37      0.37       881
weighted avg       0.72      0.79      0.74       881
```

Gambar 4.30 Hasil Penerapan *K-NN* Dengan 80% Data Latih Dan 20% Data Uji

Pada Gambar 4.30, penerapan *K-NN* dengan pembagian data latih 80% dan data uji 20% menunjukkan akurasi 79%. *Precision*: negatif 40%, netral 0%, positif 82%. *Recall*: negatif 16%, netral 0%, positif 95%. *F1-Score*: negatif 22%, netral 0%, positif 88%. *Support*: negatif 147, netral 31, positif 703.

Model berhasil pada 'positif', namun performa rendah pada 'negatif' dan lebih buruk pada 'netral'.

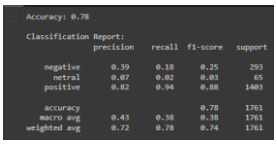
2. Data Pelatihan 70% Dan Data Uji 30%



Gambar 4.31 Hasil Penerapan *K-NN* Dengan 70% Data Latih Dan 30% Data Uji

Pada Gambar 4.31, hasil penerapan *K-NN* dengan pembagian data latih 70% dan data uji 30% menunjukkan akurasi 65%. *Precision*: negatif 24%, netral 0%, positif 85%. *Recall*: negatif 48%, netral 0%, positif 71%. *F1-Score*: negatif 32%, netral 0%, positif 77%. *Support*: negatif 217, netral 44, positif 1060. Model berhasil pada 'positif', namun performa rendah pada 'negatif' dan lebih buruk pada 'netral'.

3. Data Pelatihan 60% Dan Data Uji 40%

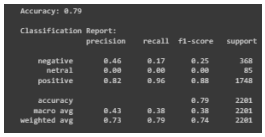


Gambar 4.32 Hasil Penerapan *K-NN* Dengan 60% Data Latih Dan 40% Data Uji

Pada Gambar 4.32, penerapan *K-NN* dengan pembagian data latih 60% dan data uji 40% mencapai akurasi 78%. *Precision*:

negatif 39%, netral 7%, positif 82%. *Recall*: negatif 18%, netral 2%, positif 94%. *F1-Score*: negatif 25%, netral 3%, positif 88%. Support: negatif 293, netral 65, positif 1403. Model berhasil pada 'positif', namun performa rendah pada 'negatif' dan lebih buruk pada 'netral'.

4. Data Pelatihan 50% Dan Data Uji 50%

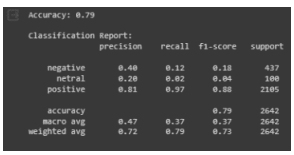


```
Accuracy: 0.79

Classification Report:
              precision    recall  f1-score   support

    negative       0.46      0.17      0.25       368
      netral       0.00      0.00      0.00        85
    positive       0.82      0.96      0.88      1748

    accuracy                           0.79      2201
   macro avg       0.43      0.38      0.38      2201
weighted avg       0.73      0.79      0.74      2201
```

Gambar 4.33 Hasil Penerapan *K-NN* Dengan 50% Data Latih Dan 50% Data Uji

Pada Gambar 4.33, hasil penerapan *K-NN* dengan pembagian data latih 50% dan data uji 50% menunjukkan akurasi 79%. *Precision*: negatif 46%, netral 0%, positif 82%. *Recall*: negatif 17%, netral 0%, positif 96%. *F1-Score*: negatif 25%, netral 0%, positif 88%. *Suppor*t: negatif 368, netral 85, positif 1748. Model berhasil pada 'positif', namun performa rendah pada 'negatif' dan lebih buruk pada 'netral'.

5. Data Pelatihan 40% Dan Data Uji 60%

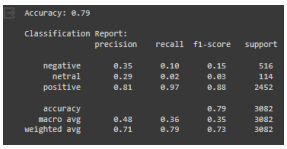


```
Accuracy: 0.79

Classification Report:
              precision    recall  f1-score   support

    negative       0.40      0.12      0.18       437
      netral       0.20      0.02      0.04       100
    positive       0.81      0.97      0.88      2105

    accuracy                           0.79      2642
   macro avg       0.47      0.37      0.37      2642
weighted avg       0.72      0.79      0.73      2642
```

Gambar 4.34 Hasil Penerapan *K-NN* Dengan 40% Data Latih Dan 60% Data Uji

Gambar 4.34 menunjukkan penerapan *K-NN* dengan data latih 40% dan data uji 60%, akurasi 79%. *Precision*: negatif 40%, netral 20%, positif 81%. *Recall*: negatif 12%, netral 2%, positif 97%. *F1-Score*: negatif 18%, netral 4%, positif 88%. *Support*: negatif 437, netral 100, positif 2105. Model berhasil untuk 'positif', tetapi performa rendah untuk 'negatif' dan lebih buruk untuk 'netral'.

6. Data Pelatihan 30% Dan Data Uji 70%

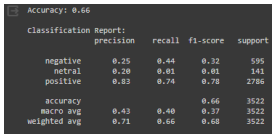


```
Accuracy: 0.79

Classification Report:
              precision    recall  f1-score   support

    negative       0.35      0.10      0.15       516
      netral       0.29      0.02      0.03       114
    positive       0.81      0.97      0.88      2452

    accuracy                           0.79      3082
   macro avg       0.48      0.36      0.35      3082
weighted avg       0.71      0.79      0.73      3082
```

Gambar 4.35 Hasil Penerapan *K-NN* Dengan 30% Data Latih Dan 70% Data Uji

Gambar 4.35 menunjukkan penerapan *K-NN* dengan data latih 30% dan data uji 70%, akurasi 79%. *Precision*: negatif 35%, netral 29%, positif 81%. *Recall*: negatif 10%, netral 2%, positif 97%. *F1-Score*: negatif 15%, netral 3%, positif 88%. *Support*: negatif 516, netral 114, positif 2452. Model berhasil untuk 'positif', tapi kinerja rendah untuk 'negatif' dan lebih buruk untuk 'netral'.

7. Data Pelatihan 20% Dan Data Uji 80%

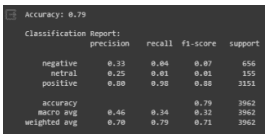


```
Accuracy: 0.66

Classification Report:
              precision    recall  f1-score   support

    negative       0.25      0.44      0.32       595
     netral        0.20      0.01      0.01       141
    positive       0.83      0.74      0.78      2786

    accuracy                           0.66      3522
   macro avg       0.43      0.40      0.37      3522
weighted avg       0.71      0.66      0.68      3522
```

Gambar 4.36 Hasil Penerapan *K-NN* Dengan 20% Data Latih Dan 80% Data Uji

Pada Gambar 4.36, *K-NN* dengan pembagian data 20% latih dan 80% uji memiliki akurasi 66%. Presisi: negatif 25%, netral 20%, positif 83%. *Recall*: negatif 44%, netral 2%, positif 74%. *F1-Score*: negatif 21%, netral 1%, positif 78%. *support*: negatif 595, netral 141, positif 2786. Model berhasil untuk 'positif', namun performa rendah untuk 'negatif' dan lebih buruk untuk 'netral'.

8. Data Pelatihan 10% Dan Data Uji 90%



```
Accuracy: 0.79

Classification Report:
              precision    recall  f1-score   support

    negative       0.33      0.04      0.07       656
     netral        0.25      0.01      0.01       155
    positive       0.80      0.98      0.88      3151

    accuracy                           0.79      3962
   macro avg       0.46      0.34      0.32      3962
weighted avg       0.70      0.79      0.71      3962
```

Gambar 4.37 Hasil Penerapan *K-NN* Dengan 10% Data Latih Dan 90% Data Uji.

Gambar 4.37 menunjukkan hasil *K-NN* dengan data latih 10% dan uji 90%. Akurasi 79%, dengan presisi negatif 33%, netral 25%, positif 80%. *Recall* negatif 17%, netral 0%, positif 96%. *F1-Score* negatif 4%, netral 1%, positif 88%. *Support*: negatif 656, netral 155, positif 3151. Model berhasil untuk 'positif',

namun performa rendah untuk 'negatif' dan lebih buruk untuk 'netral'.

Dari beberapa percobaan yang dilakukan, dimulai dari penggunaan 20% data uji hingga 90% data uji dengan metode *K-NN*, terlihat keberhasilan dalam mengklasifikasikan *sentiment* positif. Namun, secara umum, model menunjukkan kinerja yang kurang memuaskan untuk sentimen negatif dan netral.

b. *Naïve Bayes*

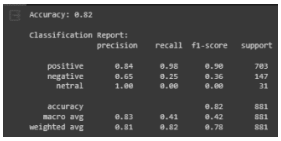
Pada penelitian ini di lakukan beberapa proses jika menggunakan algoritma *Naïve Bayes* yaitu pemilihan fitur, pelabelan data latih, perhitungan probalilitas, dan klasifikasi. Berikut ini penerapan *Naïve Bayes* dalam *sentiment analysis*

1. Data Pelatihan 80% Dan Data Uji 20%

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
# Langkah 1: Persiapan Data
df = pd.read_csv('preprocessing.csv')
X = df['preprocessing list']
y = df['sentiment']
# Langkah 2: Pembagian Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Langkah 3: Ekstraksi Fitur (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=100)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
# Langkah 4: Inisialisasi Model Naive Bayes
naive_bayes_classifier = MultinomialNB()
# Langkah 5: Pelatihan Model
naive_bayes_classifier.fit(X_train_tfidf, y_train)
# Langkah 6: Evaluasi Model
y_pred = naive_bayes_classifier.predict(X_test_tfidf)
# Mengukur Akurasi dan Menampilkan Laporan Klasifikasi
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
classification_rep = classification_report(y_test, y_pred, labels=['positive', 'negative', 'netral'], zero_division=1)
print('\nClassification Report:\n', classification_rep)
```

Gambar 4.38 Proses Penerapan *Naïve Bayes* Dengan 80%Data Latih Dan 20% Data Uji

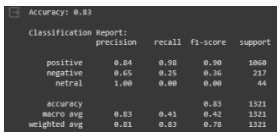
Gambar 4.38 menunjukkan proses penerapan Naïve Bayes dengan data latih 80% dan uji 20%. Melibatkan persiapan data, pembagian data, ekstraksi fitur TF-IDF, inisialisasi model Multinomial Naïve Bayes, pelatihan, evaluasi kinerja, pengukuran akurasi, dan tampilan hasil klasifikasi.



Gambar 4.39 Hasil Penerapan *Naïve Bayes* Dengan 80% Data Latih Dan 20% Data Uji.

Gambar 4.39 menunjukkan penerapan *Naïve Bayes*, data latih 80%, data uji 20%, akurasi 82%. *Precision*: *negative* 65%, netral 100%, *positive* 84%. *Recall: negative* 25%, netral 0%, *positive* 98%. *F1-Score: negative* 36%, netral 0%, *positive* 90%. *Support: negative* 147, netral 31, *positive* 703. Model berhasil pada '*positive*', namun rendah untuk '*negative*' dan lebih buruk untuk 'netral'.

2. Data Pelatihan 70% Dan Data Uji 30%

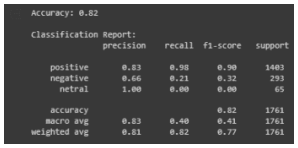


Gambar 4.40 Hasil Penerapan *Naïve Bayes* Dengan 70% Data Latih Dan 30% Data Uji

Pada Gambar 4.40, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 70% dan data uji 30% menunjukkan

akurasi 83%. *Precision negative* 65%, netral 100%, *positive* 84%. *Recall negative* 25%, netral 0%, *positive* 98%. *F1-Score negative* 36%, netral 0%, *positive* 90%. *Support: negative* 217, netral 44, *positive* 1060 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

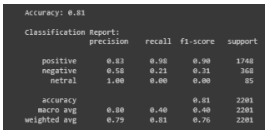
3. Data Pelatihan 60% Dan Data Uji 40%



Gambar 4.41 Hasil Penerapan Naïve Bayes Dengan 60% Data Latih Dan 40% Data Uji

Pada Gambar 4.41, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 60% dan data uji 40% menunjukkan akurasi 82%. *Precision negative* 66%, netral 100%, *positive* 83%. *Recall negative* 21%, netral 0%, *positive* 98%. *F1-Score negative* 32%, netral 0%, *positive* 90%. *Support: negative* 293, netral 65, *positive* 1403 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

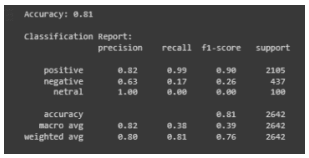
4. Data Pelatihan 50% Dan Data Uji 50%



Gambar 4.42 Hasil Penerapan *Naïve Bayes* Dengan 50% Data Latih Dan 50% Data Uji.

Pada Gambar 4.42, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 50% dan data uji 50% menunjukkan akurasi 81%. *Precision negative* 58%, netral 100%, *positive* 83%. *Recall negative* 21%, netral 0%, *positive* 98%. *F1-Score negative* 31%, netral 0%, *positive* 90%. *Support: negative* 368, netral 85, *positive* 1748 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

5. Data Pelatihan 40% Dan Data Uji 60%

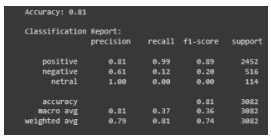


Gambar 4.43 Hasil Penerapan *Naïve Bayes* Dengan 40% Data Latih Dan 60% Data Uji.

Pada Gambar 4.43, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 40% dan data uji 60% menunjukkan akurasi 81%. *Precision negative* 63%, netral 100%, *positive* 82%. *Recall negative* 17%, netral 0%, *positive* 99%. *F1-Score*

negative 26%, netral 0%, *positive* 90%. *Support: negative* 437, netral 100, *positive* 2105 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

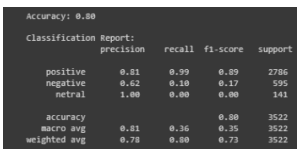
6. Data Pelatihan 30% Dan Data Uji 70%



Gambar 4.44 Hasil Penerapan *Naïve Bayes* Dengan 30% Data Latih Dan 70% Data Uji

Pada Gambar 4.44, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 30% dan data uji 70% menunjukkan akurasi 81%. *Precision negative* 61%, netral 100%, *positive* 81%. *Recall negative* 12%, netral 0%, *positive* 99%. *F1-Score negative* 20%, netral 0%, *positive 8*9%. *Support: negative* 516, netral 114, *positive* 2452 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

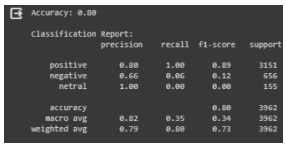
7. Data Pelatihan 20% Dan Data Uji 80%



Gambar 4.45 Hasil Penerapan *Naïve Bayes* Dengan 20% Data Latih Dan 80% Data Uji

Pada Gambar 4.45, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 20% dan data uji 80% menunjukkan akurasi 80%. *Precision negative* 62%, netral 100%, *positive* 81%. *Recall negative* 10%, netral 0%, *positive* 99%. *F1-Score negative* 17%, netral 0%, *positive 8*9%. *Support: negative* 595, netral 141, *positive* 2786 Model tampaknya berhasil untuk kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

8. Data Pelatihan 10% Dan Data Uji 90%



Gambar 4.46 Hasil Penerapan *Naïve Bayes* Dengan 10% Data Latih Dan 90% Data Uji

Pada Gambar 4.46, Hasil penerapan *Naïve Bayes* dengan pembagian data latih 10% dan data uji 90% menunjukkan akurasi 80%. *Precision negative* 60%, netral 100%, *positive* 80%. *Recall negative* 6%, netral 0%, *positive* 100%. *F1-Score negative* 12%, netral 0%, *positive 8*9%. *Support: negative* 656, netral 155, *positive* 3151 Model tampaknya berhasil untuk

kelas *'positive*', namun kinerja rendah untuk kelas '*negative'* dan lebih buruk untuk kelas 'netral'.

Dari beberapa percobaan yang dilakukan, dimulai dari penggunaan 20% data uji hingga 90% data uji dengan metode Naïve Bayes, terlihat keberhasilan dalam mengklasifikasikan sentimen positif. Namun, secara umum, model menunjukkan kinerja yang kurang memuaskan untuk sentimen negatif dan netral.

Pada penelitian ini, *K-Nearest Neighbors (KNN)* dan *Naïve Bayes* menunjukkan kinerja baik pada kelas positif, tetapi menghadapi kendala dalam memprediksi kelas negatif dan netral. Ketidakseimbangan jumlah sampel, dengan 3513 *sentimen* positif dan hanya 720 serta 169 *sentimen* negatif dan netral, menjadi penyebab utama keterbatasan kinerja pada kedua kelas tersebut. Akibatnya, kurangnya pembelajaran pada algoritma menyebabkan keterbatasan dalam mengenali pola pada kelas negatif dan netral.

4.6 *VALIDATION*

Validation adalah proses pengujian dan evaluasi kinerja model pada *dataset* yang tidak digunakan selama pelatihan model, dengan tujuan mengukur seberapa baik model dapat menggeneraliasasi dari data pelatihan yang belum pernah dilihat sebelumnya.

4.6.1 *Cofusion Matrix*

Confusion matrix adalah tabel yang digunakan untuk mengevaluasi kinerja model klasifikasi.

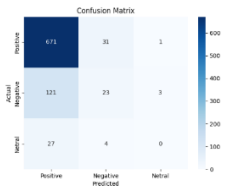
a. K-Nearst Neighbor:

1. Data Latih 80% Dan Data Uji 20%

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
conf_matrix = confusion_matrix(y_test, y_pred, labels=['positive', 'negative', 'netral'])
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Positive', 'Negative', 'Netral'], yticklabels=['Positive', 'Negative', 'Netral'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.savefig('confusion_matrix_02.png')
plt.show()
```

Gambar 4.47 Proses *Cofusion Matrik* pada Metode *K-NN* dengan 80% data latih dan 20% data uji

Pada gambar 4.47 menunjukkan proses *confusion matrix* dengan pembagian data latih 80% dan data uji 20%. Proses ini melibatkan pengimportan *library*, perhitungan *confusion matrix*, dan pembuatan *heatmap* dengan anotasi angka dan skala warna biru ('*Blues*'). Hasilnya disimpan sebagai '*confusion_matrix*_02.*png*' dan ditampilkan visual, kemudian mencatak akurasi model klasifikasi dalam bentuk presentasi.



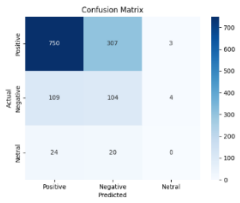
Gambar 4.48 Hasil *Cofusion Matrix* pada Metode *K-NN* Dengan 80% Data Latih Dan 20% Data Uji

Berikut ini penjelasan mengenai gambar 4.48:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 671 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 31 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 1 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 121 data.
- *True Negative (TNg)*: jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 23 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 3 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 27 data.

- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 4 data
- True Netral (TNt): Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

2. Data Latih 70% Dan Data Uji 30%



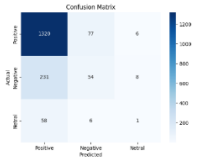
Gambar 4.49 Hasil *Cofusion Matrix* pada Metode *K-NN* Dengan 70% Data Latih Dan 30% Data Uji

Berikut ini penjelasan mengenai gambar 4.49:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 750 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 307 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 3 data.

- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 109 data.
- *True Negative (TNg)*: jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 104 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 4 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 24 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 20 data
- *True Netral (TNt)*: Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

3. Data Latih 60% Dan Data Uji 40%



Gambar 4.50 Hasil *Cofusion Matrix* pada Metode *K-NN*

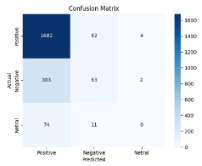
Dengan 60% Data Latih Dan 40% Data Uji

Berikut ini penjelasan mengenai gambar 4.50:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 1320 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 77 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 6 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 231 data.
- *True Negative (TNg)*: jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 54 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 8 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 58 data.

- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 6 data
- *True Netral (TNt)*: Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 1 data.

4. Data Latih 50% Dan Data Uji 50%



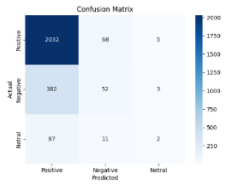
Gambar 4.51 Hasil *Cofusion Matrix* pada Metode *K-NN* Dengan 50% Data Latih Dan 50% Data Uji

Berikut ini penjelasan mengenai gambar 4.51:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 1682 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 62 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 4 data.

- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 303 data.
- *True Negative (TNg)*: jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 63 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 2 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 74 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 11 data
- *True Netral (TNt)*: Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

5. Data Latih 40% Dan Data Uji 60%



Gambar 4.52 Hasil *Cofusion Matrix* pada Metode *K-NN* Dengan 40% Data Latih Dan 60% Data Uji

Berikut ini penjelasan mengenai gambar 4.52:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 2032 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 68 data.
- *False Positive (FP)*: jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 5 data.
- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai positive, dalam kasus ini terdapat 382 data.
- *True Negative (TNg)*: jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 52 data.

- *False Negative (FNg)*: jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 3 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 87 data.
- *False Netral (FNt)*: Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 11 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 2 data.

6. Data Latih 30% Dan Data Uji 70%

Gambar 4.53 Hasil *Cofusion Matrix* pada Metode *K-NN*

Dengan 30% Data Latih Dan 70% Data Uji

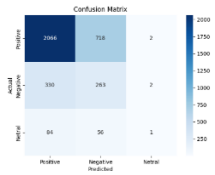
Berikut ini penjelasan mengenai gambar 4.53:

- *True Positive (TP)*: jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 2368 data.

- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 81 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 3 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 464 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 50 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 2 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 100 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 12 data

- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 2 data.

7. Data Latih 20% Dan Data Uji 80%



Gambar 4.54 Hasil *Cofusion Matrix* pada Metode *K-NN*

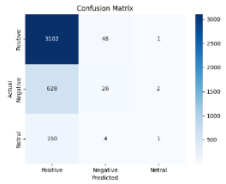
Dengan 20% Data Latih Dan 80% Data Uji

Berikut ini penjelasan mengenai gambar 4.54:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 2066 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 718 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 2 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 330 data.

- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 263 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 2 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 84 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 56 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 1 data.

8. Data Latih 10% Dan Data Uji 90%



Gambar 4.55 Hasil *Cofusion Matrix* pada Metode *K-NN* Dengan 10% Data Latih Dan 90% Data Uji

Berikut ini penjelasan mengenai gambar 4.55:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive,* dalam kasus ini terdapat 3102 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai *negative,* dalam kasus ini terdapat 48 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 1 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive,* dalam kasus ini terdapat 628 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 26 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 2 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 150 data.

- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 4 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 1 data.

Dari hasil beberapa percobaan confusion matrix dengan penggunaan data uji antara 20% hingga 90%, terlihat peningkatan akurasi analisis sentimen dengan tambahan data latih. Meskipun demikian, perlu diingat bahwa akurasi tidak selalu mencerminkan kinerja keseluruhan model. Model menunjukkan keunggulan dalam mengenali sentimen positif dengan tingginya nilai True Positive, namun mengalami kesulitan dalam mengenali sentimen negatif dan netral, terlihat dari penurunan yang signifikan pada recall dan F1-Score. Peningkatan jumlah False Positive pada sentimen negatif menunjukkan kecenderungan model untuk mengklasifikasikan ulasan negatif sebagai positif atau netral. Identifikasi sentimen netral sulit dilakukan, dengan recall dan F1-Score mencapai 0% dalam beberapa kasus. Selain memperhatikan akurasi, penting juga untuk mempertimbangkan precision, recall, dan F1-Score untuk setiap kelas sentimen. Variasi dalam pembagian data latih dan uji memberikan dampak pada kinerja model.

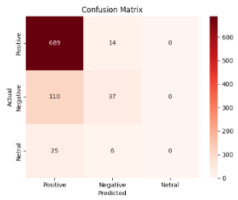
b. Naïve Bayes:

1. Data Latih 80% Dan Data Uji 20%

```
conf_matrix = confusion_matrix(y_test, y_pred, labels=['positive', 'negative', 'netral'])
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Reds', xticklabels=['Positive', 'Negative', 'Netral'], yticklabels=['Positive', 'Negative', 'Netral'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.savefig('nb_confusion_matrix_02.png')
plt.show()
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Accuracy: {accuracy:.2f}%')
```

Gambar 4.56 Proses *Cofusion Matrik* pada Metode *Naïve Bayes* dengan 80% data latih dan 20% data uji

Pada gambar 4.56 menunjukkan proses *confusion matrix* dengan pembagian data latih 80% dan data uji 20%. Dimuali menghitung *confusion matrix*, dan pembuatan *heatmap* dengan menggunakan *seaborn*, Hasilnya disimpan sebagai 'nb_*confusion_matrix*_02.*png*' dan ditampilkan visual, kemudian mencatak akurasi model klasifikasi dalam bentuk presentasi.



Gambar 4.57 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 80% Data Latih Dan 20% Data Uji

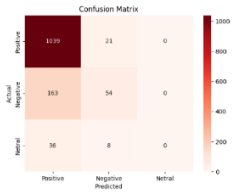
Berikut ini penjelasan mengenai gambar 4.57:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 689 data.

- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 14 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 110 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 37 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 25 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 6 data

- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

2. Data Latih 70% Dan Data Uji 30%



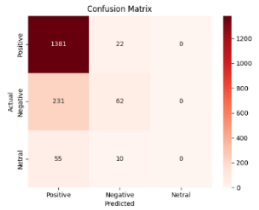
Gambar 4.58 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 70% Data Latih Dan 30% Data Uji

Berikut ini penjelasan mengenai gambar 4.58:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 1039 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 21 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 163 data.

- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 54 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 36 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 8 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

3. Data Latih 60% Dan Data Uji 40%



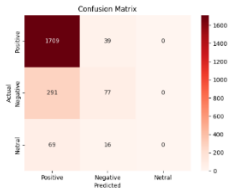
Gambar 4.59 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 60% Data Latih Dan 40% Data Uji

Berikut ini penjelasan mengenai gambar 4.59:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive,* dalam kasus ini terdapat 1381 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai *negative,* dalam kasus ini terdapat 22 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative,* tetapi telah salah diklasifikasi sebagai *positive,* dalam kasus ini terdapat 231 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative,* dalam kasus ini terdapat 62 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive,* dalam kasusu ini terdapat 55 data.

- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 10 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

4. Data Latih 50% Dan Data Uji 50%



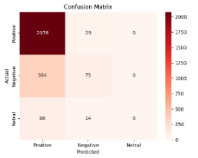
Gambar 4.60 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 50% Data Latih Dan 50% Data Uji

Berikut ini penjelasan mengenai gambar 4.60:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 1709 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 39 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.

- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 291 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 77 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 69 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 16 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

5. Data Latih 40% Dan Data Uji 60%



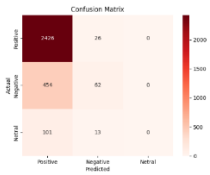
Gambar 4.61 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 40% Data Latih Dan 60% Data Uji

Berikut ini penjelasan mengenai gambar 4.61:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive,* dalam kasus ini terdapat 2076 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 29 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 364 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 73 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive*, dalam kasusu ini terdapat 86 data.

- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 14 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

6. Data Latih 30% Dan Data Uji 70%



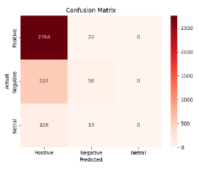
Gambar 4.62 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 30% Data Latih Dan 70% Data Uji

Berikut ini penjelasan mengenai gambar 4.62:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 2426 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebegai *negative*, dalam kasus ini terdapat 26 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebegai netral, dalam kasus ini terdapat 0 data.

- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebegai *positive*, dalam kasus ini terdapat 454 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 62 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive,* dalam kasusu ini terdapat 101 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 13 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

7. Data Latih 20% Dan Data Uji 80%



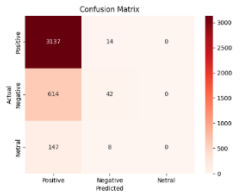
Gambar 4.63 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 20% Data Latih Dan 80% Data Uji

Berikut ini penjelasan mengenai gambar 4.63:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive,* dalam kasus ini terdapat 2764 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai *negative,* dalam kasus ini terdapat 22 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative,* tetapi telah salah diklasifikasi sebagai *positive,* dalam kasus ini terdapat 537 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative,* dalam kasus ini terdapat 58 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative,* tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive,* dalam kasusu ini terdapat 128 data.

- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 13 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

8. Data Latih 10% Dan Data Uji 90%



Gambar 4.64 Hasil *Cofusion Matrix* pada Metode *Naïve Bayes* Dengan 10% Data Latih Dan 90% Data Uji

Berikut ini penjelasan mengenai gambar 4.64:

- *True Positive (TP):* jumlah data yang sebenarnya benar-benar *positive* dan telah di klasifikasi dengan benar sebagai *positive*, dalam kasus ini terdapat 3137 data.
- *False Positive (FP):* jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai *negative*, dalam kasus ini terdapat 14 data.
- False Positive (FP): jumlah data yang sebenarnya *positive*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.

- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai *positive*, dalam kasus ini terdapat 614 data.
- *True Negative (TNg):* jumlah data yang sebenarnya benar-benar *negative* dan telah di klasifikasi dengan benar sebagai *negative*, dalam kasus ini terdapat 42 data.
- *False Negative (FNg):* jumlah data yang sebenarnya *negative*, tetapi telah salah diklasifikasi sebagai netral, dalam kasus ini terdapat 0 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi telah di klasifikasi sebagai *positive,* dalam kasusu ini terdapat 147 data.
- *False Netral (FNt):* Jumlah data yang sebenarnya netral, tetapi diklasifikasikan sebagai *negative*, dalam kasus ini terdapat 8 data
- *True Netral (TNt):* Jumlah data yang sebenarnya benar-benar netral, dan telah diklasifikasi dengan benar sebagai netral, dalam kasus ini terdapat 0 data.

Dari berbagai eksperimen menggunakan confusion matrix dengan variasi data uji antara 20% hingga 90%, terlihat bahwa model cenderung lebih baik dalam mengenali sentimen positif daripada sentimen negatif dan netral. Selain itu, perubahan proporsi antara data latih dan uji memiliki dampak langsung pada kinerja model,

dengan beberapa proporsi tertentu menunjukkan peningkatan kinerja yang signifikan.

4.6.2. *K-Fold Cross-Validation*

K-Fold Cross-Validation adalah cara umum untuk mengevaluasi seberapa baik model *machine learning* bekerja. Data dibagi menjadi beberapa kelompok (*fold*), dan model diuji sebanyak k kali. Setiap kali, satu kelompok menjadi data pengujian dan yang lainnya menjadi data pelatihan. Performa model diukur dari setiap uji, dan rata-rata hasil memberikan gambaran lebih baik tentang seberapa baik model dapat digunakan pada data yang belum pernah dilihat. Metode ini membantu mengurangi fluktuasi hasil evaluasi dan dapat mengidentifikasi masalah seperti *overfitting* atau *underfitting*. Pemilihan nilai k, pada penelitian ini digunakan k nya 5. Berikut ini *k-fold cross-validation:*

a. K-Nearst Neighbor

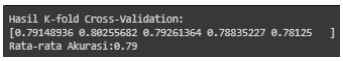
1. Data Latih 80% Dan Data Uji 20%

```
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
cv_scores = cross_val_score(knn_classifier, X_train_tfidf, y_train, cv=5)
print("Hasil K-fold Cross-Validation:")
print(cv_scores)
print(f'Rata-rata Akurasi: {cv_scores.mean():.2f}')
```

Gambar 4.65 Proses *K-Fold Cross-Validation* Pada Metode *K-NN* Dengan Data Latih 80% Dan Data Uji 20%

Pada gambar 4.65 merupakan proses *k-fold cross-validation* menggunakan data latih 80% dan data uji 20% yang dimulai dengan impor fungsi `cross_val_score` dan model *K-NN* dari

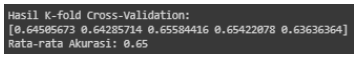
scikit-learn, perhitungan akurasi, serta menampilkan hasil dan rata-rata akurasi dari *k-fold cross-validation.*



Gambar 4.66 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Dengan Data Latih 80% Dan Data Uji 20%

Pada 4.66 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 79%.

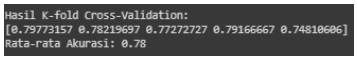
2. Data Latih 70% Dan Data Uji 30%



Gambar 4.67 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 70% Dan Data Uji 30%

Pada 4.67 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 65%.

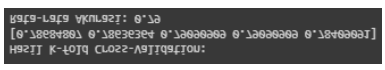
3. Data Latih 60% Dan Data Uji 40%



Gambar 4.68 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 60% Dan Data Uji 40%

Pada 4.68 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 78%.

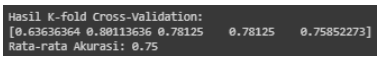
4. Data Latih 50% Dan Data Uji 50%



Gambar 4.69 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 50% Dan Data Uji 50%

Pada 4.69 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 79%.

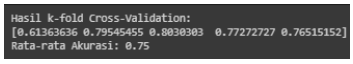
5. Data Latih 40% Dan Data Uji 60%



Gambar 4.70 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 40% Dan Data Uji 60%

Pada 4.70 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 75%.

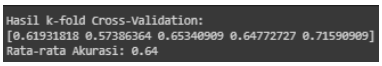
6. Data Latih 30% Dan Data Uji 70%



Gambar 4.71 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 30% Dan Data Uji 70%

Pada 4.71 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 75%.

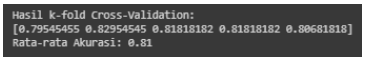
7. Data Latih 20% Dan Data Uji 80%



Gambar 4.72 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 20% Dan Data Uji 80%

Pada 4.72 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 64%,

8. Data Latih 10% Dan Data Uji 90%

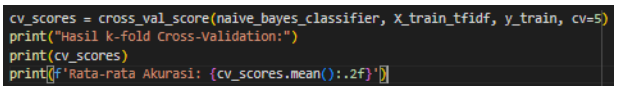


Gambar 4.73 Hasil *K-Fold Cross-Validation* Pada Metode *K-NN* Data Latih 10% Dan Data Uji 90%

Pada 4.73 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi model sekitar 81%.

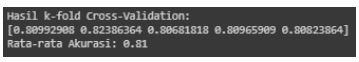
b. *Naïve Bayes*

1. Data Latih 80% Dan Data Uji 20%



Gambar 4.74 Proses *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 80% Dan Data Uji 20%

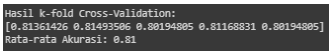
Pada gambar 4.74 merupakan proses *k-fold cross-validation* menggunakan data latih 80% dan data uji 20% yang dimulai dengan model *Naïve Bayes*, serta menampilkan hasil dan rata-rata akurasi dari *k-fold cross-validation.*



Gambar 4.75 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 80% Dan Data Uji 20%

Pada 4.75 menggambarkan hasil K-Fold *Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

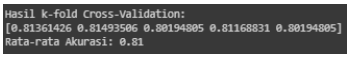
2. Data Latih 70% Dan Data Uji 30%



Gambar 4.76 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 70% Dan Data Uji 30%

Pada 4.76 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

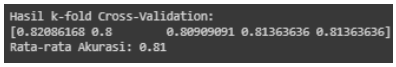
3. Data Latih 60% Dan Data Uji 40%



Gambar 4.77 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 60% Dan Data Uji 40%

Pada 4.77 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

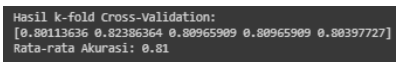
4. Data Latih 50% Dan Data Uji 50%



Gambar 4.78 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 50% Dan Data Uji 50%

Pada 4.78 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

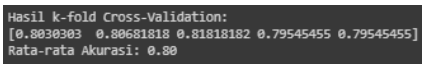
5. Data Latih 40% Dan Data Uji 60%



Gambar 4.79 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 40% Dan Data Uji 60%

Pada 4.79 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

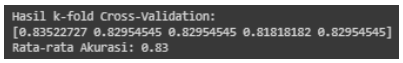
6. Data Latih 30% Dan Data Uji 70%



Gambar 4.80 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 30% Dan Data Uji 70%

Pada 4.80 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 80%.

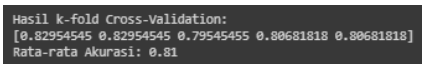
7. Data Latih 20% Dan Data Uji 80%



Gambar 4.81 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 20% Dan Data Uji 80%

Pada 4.81 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 83%.

8. Data Latih 10% Dan Data Uji 90%



Gambar 4.82 Hasil *K-Fold Cross-Validation* Pada Metode *Naïve Bayes* Data Latih 10% Dan Data Uji 90%

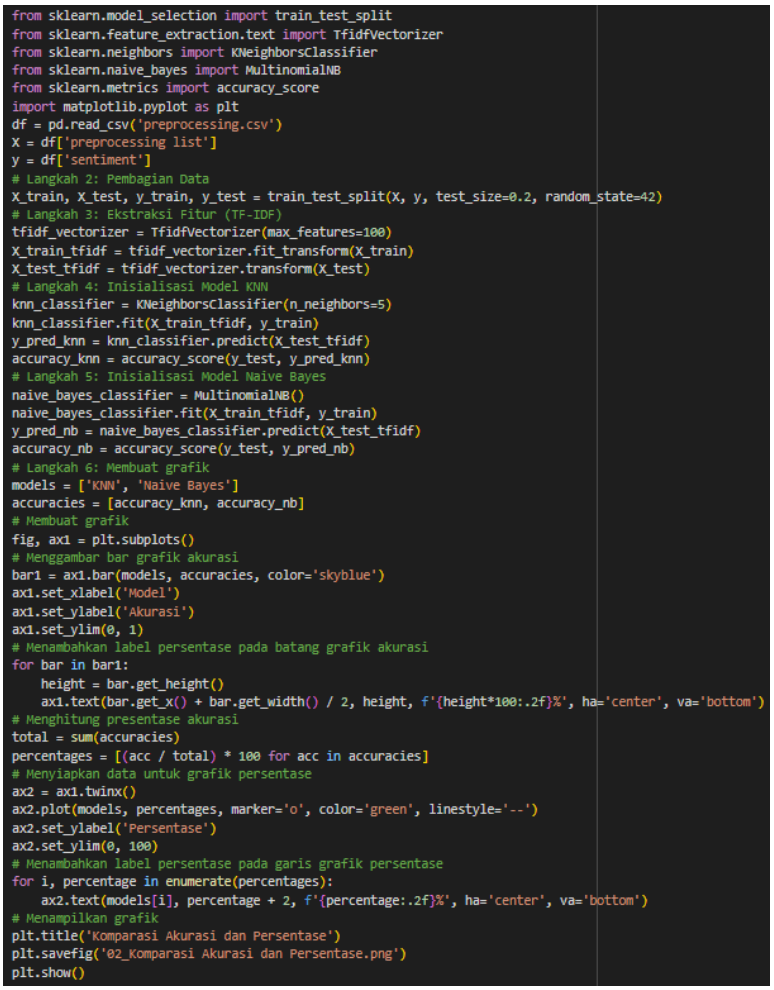
Pada 4.82 menggambarkan hasil *K-Fold Cross-Validation* menunjukkan akurasi konsisten sekitar 81%.

4.6.3. Perbandingan Metode *K-NN* dan *Naïve Bayes*

Dalam penelitian ini, dilakukan perbandingan antara metode *K-NN* dan *Naïve Bayes* dengan tujuan memperoleh pemahaman yang lebih baik

mengenai kinerja kedua metode tersebut. Berikut adalah penerapan perbandingan antara *K-NN* dan *Naïve Bayes*:

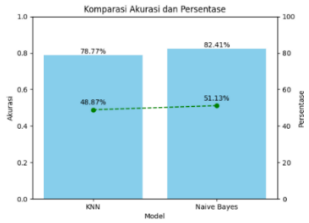
1. Data Latih 80% dan Data Uji 20%



```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
df = pd.read_csv('preprocessing.csv')
X = df['preprocessing list']
y = df['sentiment']
# Langkah 2: Pembagian Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Langkah 3: Ekstraksi Fitur (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=100)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
# Langkah 4: Inisialisasi Model KNN
knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train_tfidf, y_train)
y_pred_knn = knn_classifier.predict(X_test_tfidf)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
# Langkah 5: Inisialisasi Model Naive Bayes
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_tfidf, y_train)
y_pred_nb = naive_bayes_classifier.predict(X_test_tfidf)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
# Langkah 6: Membuat grafik
models = ['KNN', 'Naive Bayes']
accuracies = [accuracy_knn, accuracy_nb]
# Membuat grafik
fig, ax1 = plt.subplots()
# Menggambar bar grafik akurasi
bar1 = ax1.bar(models, accuracies, color='skyblue')
ax1.set_xlabel('Model')
ax1.set_ylabel('Akurasi')
ax1.set_ylim(0, 1)
# Menambahkan label persentase pada batang grafik akurasi
for bar in bar1:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width() / 2, height, f'{height*100:.2f}%', ha='center', va='bottom')
# Menghitung presentase akurasi
total = sum(accuracies)
percentages = [(acc / total) * 100 for acc in accuracies]
# Menyiapkan data untuk grafik persentase
ax2 = ax1.twinx()
ax2.plot(models, percentages, marker='o', color='green', linestyle='--')
ax2.set_ylabel('Persentase')
ax2.set_ylim(0, 100)
# Menambahkan label persentase pada garis grafik persentase
for i, percentage in enumerate(percentages):
    ax2.text(models[i], percentage + 2, f'{percentage:.2f}%', ha='center', va='bottom')
# Menampilkan grafik
plt.title('Komparasi Akurasi dan Persentase')
plt.savefig('02_Komparasi Akurasi dan Persentase.png')
plt.show()
```

Gambar 4.83 Proses Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 80% Dan Data Uji 20%

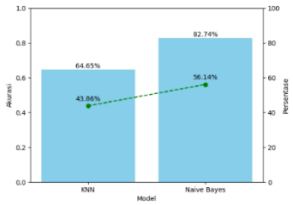
Pada gambar 4.83 merupakan proses perbandingan 2 metode yaitu *K-NN* dan *Naïve Bayes*, yang di mulai dari membagi data latih dan data uji, *Tf-Idf*, inisialisasi model *K-NN* dan *Naïve Bayes*, setelah berhasil dilanjutkan membuat grafik, dan menghitung akurasi, kemudian menampilkan hasil.



Gambar 4.84 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 80% Dan Data Uji 20%

Pada Gambar 4.84 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 78.77%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 82.41%.

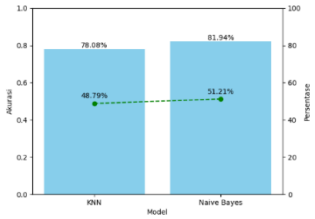
2. Data Latih 70% dan Data Uji 30%



Gambar 4.85 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 70% Dan Data Uji 30%

Pada Gambar 4.85 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 64.55%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 82.74%.

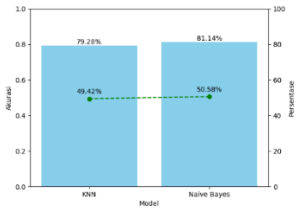
3. Data Latih 60% dan Data Uji 40%



Gambar 4.86 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 60% Dan Data Uji 40%

Pada Gambar 4.86 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 78.08%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 81.94%.

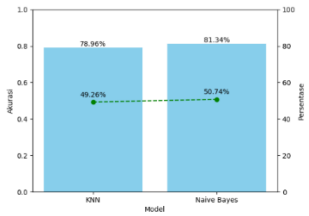
4. Data Latih 50% dan Data Uji 50%



Gambar 4.87 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 50% Dan Data Uji 50%

Pada Gambar 4.87 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 79.28%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 81.14%.

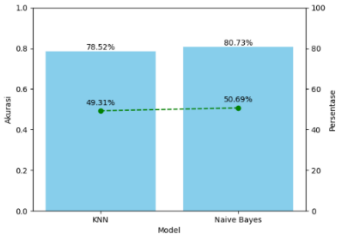
5. Data Latih 40% dan Data Uji 60%



Gambar 4.88 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 40% Dan Data Uji 60%

Pada Gambar 4.88 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 78.96%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 81.34%.

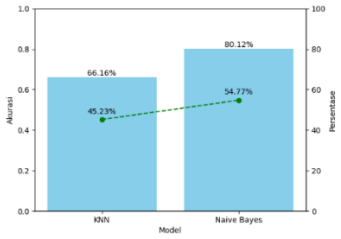
6. Data Latih 30% dan Data Uji 70%



Gambar 4.89 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 30% Dan Data Uji 70%

Pada Gambar 4.89 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 78.52%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 80.73%.

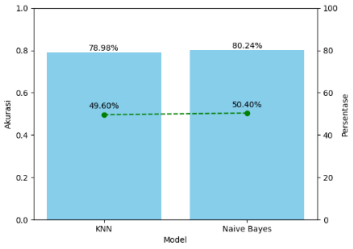
7. Data Latih 20% dan Data Uji 80%



Gambar 4.90 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 20% Dan Data Uji 80%

Pada Gambar 4.90 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma *K-NN* mencapai 66.16%, sedangkan Naïve Bayes memiliki nilai akurasi sebesar 80.12%.

8. Data Latih 10% dan Data Uji 90%



Gambar 4.91 Hasil Perbandingan Metode *K-NN* Dan *Naïve Bayes* Dengan Data Latih 10% Dan Data Uji 90%

Pada Gambar 4.91 terlihat hasil perbandingan antara metode *K-NN* dan *Naïve Bayes*, di mana nilai akurasi algoritma k-NN mencapai 78.98%, sedangkan *Naïve Bayes* memiliki nilai akurasi sebesar 80.24%.

Tabel 4.15 Hasil Perbandingan *Cofusion Matrix* Metode *K-NN* dan *Naïve Bayes*

Data Uji	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
K-NN	78.77%	64.55%	78.08%	79.28%	78.96%	78.52%	66.16%	78.98%
Naïve Bayes	82.41%	82.74%	81.94%	81.14%	81.34%	80.73%	80.12%	80.42%
Akurasi								

Setelah melakukan perbandingan dari ke dua metode *K-NN* dan *Naïve Bayes*, dapat kita lihat ke dua metode itu berkerja dengan baik. Dalam penelitian ini Metode *Naïve bayes* menunjukkan nilai akurasi yang lebih tinggi dibandingkan *K-NN*, dengan rata-rata akurasi *K-NN* sebesar 75.42% dan rata-rata akurasi *Naïve Bayes* sebesar 81.35%.

4.6.4. *Word Cloud*

Word cloud dalam analisis *sentiment* adalah representasi visual kata-kata sering muncul dalam teks, dengan ukuran atau intensitas warna mencerminkan frekuensinya. Ini membantu visualisasi frekuensi kata, mengidentifikasi kata-kata kunci, dan memberikan gambaran *sentiment*. Meski bermanfaat untuk pemahaman umum, *word cloud* tidak menyediakan analisis *sentiment* mendalam. Pada penelitian ini terbagi menjadi 3 kelas *word cloud* yaitu sebagai berikut:

a. *Word clod Positive*

```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
dataset = pd.read_csv('stemming.csv')
# Memfilter data berdasarkan sentimen
positive_text = ' '.join(dataset[dataset['sentiment'] == 'positive']['stemming text'])
# Membuat WordCloud untuk sentimen positif
wordcloud_positive = WordCloud(width=800, height=800, background_color='white', min_font_size=10).generate(positive_text)
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(wordcloud_positive)
plt.axis("off")
plt.tight_layout(pad=0)
plt.title('Word Cloud - Sentimen Positif')
plt.savefig('sentiment positive.png')
plt.show()
```

Gambar 4. 92 Proses *Word cloud Positive*

Pada gambar 4.92 merupakan proses *Word cloud* yang di mulai dari memfilter data berdasarkan *sentiment positive* dan dilanjutkan membuat *word cloud sentiment positive*.

Gambar 4. 93 Hasil *Word cloud positive*.

Pada gambar 4.93 merupakan hasil *word clod* yang Dimana dapat kita lihat pada penelitian ini untuk *word cloud positive* memiliki frekuensi yang besar pada kata *good, tiktok, love, app, nice, video.*

b. *Word clod Negative*

```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
dataset = pd.read_csv('stemming.csv')
# Memfilter data berdasarkan sentimen
negative_text = ' '.join(dataset[dataset['sentiment'] == 'negative']['stemming text'])
# Membuat WordCloud untuk sentimen negatif
wordcloud_negative = WordCloud(width=800, height=800, background_color='white', min_font_size=10).generate(negative_text)
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(wordcloud_negative)
plt.axis("off")
plt.tight_layout(pad=0)
plt.title('Word Cloud - Sentimen Negatif')
plt.savefig('sentiment negative.png')
plt.show()
```

Gambar 4. 94 Proses *Word cloud Negative*

Pada gambar 4.94 merupakan proses *Word cloud* yang di mulai dari memfilter data berdasarkan *sentiment negative* dan dilanjutkan membuat *word cloud sentiment negative.*

Gambar 4. 95 Hasil *Word cloud Negative*

Pada gambar 4.95 merupakan hasil *word clod* yang Dimana dapat kita lihat pada penelitian ini untuk *word cloud negative* memiliki frekuensi yang besar pada kata *app, tiktok, video, account, please, update, help, ban.*

c. *Word clod Netral*

```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
dataset = pd.read_csv('stemming.csv')
# Memfilter data berdasarkan sentimen
netral_text = ' '.join(dataset[dataset['sentiment'] == 'netral']['stemming text'])
# Membuat WordCloud untuk sentimen netral
wordcloud_netral = WordCloud(width=800, height=800, background_color='white', min_font_size=10).generate(netral_text)
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(wordcloud_netral)
plt.axis("off")
plt.tight_layout(pad=0)
plt.title('Word Cloud - Sentimen Netral')
plt.savefig('sentiment netral.png')
plt.show()
```

Gambar 4. 96 Proses *Word cloud Netral*

Pada gambar 4.96 merupakan proses *Word cloud* yang di mulai dari memfilter data berdasarkan *sentiment* netral dan dilanjutkan membuat *word cloud sentiment* netral.

Gambar 4. 97 Hasil *Word cloud* Netral

Pada gambar 4.97 merupakan hasil *word clod* yang Dimana dapat kita lihat pada penelitian ini untuk *word cloud* netral memiliki frekuensi yang besar pada kata *app*, *tiktok*, *video*,*please*, *android*, *good*, *people*.