BAB 3
METODE PENELITIAN

3.1 ALAT DAN BAHAN YANG DIGUNAKAN

Pada penelitian ini alat dan bahan yang digunakan adalah sebagai berikut:

3.1.1 Perangkat Keras (*Hardware*)

Perangkat keras yang dibutuhkan dan digunakan pada penelitian ini dapat dilihat pada Tabel 3.1 berikut ini:

Tabel 3. 1 Perangkat Keras.

No	Perangkat Keras	Spesifikasi
1	Laptop Acer *Aspire* A514-54G	*Processor 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz, 2995 Mhz, 2 Core(s), 4 Logical Processor(s),* Ram 8,00 GB

3.1.2 Perangkat Lunak (*Software*)

Perangkat Lunak yang dibutuhkan dan digunakan pada penelitian ini dapat dilihat pada Tabel 3.2 berikut ini:

Tabel 3. 2 Perangkat lunak.

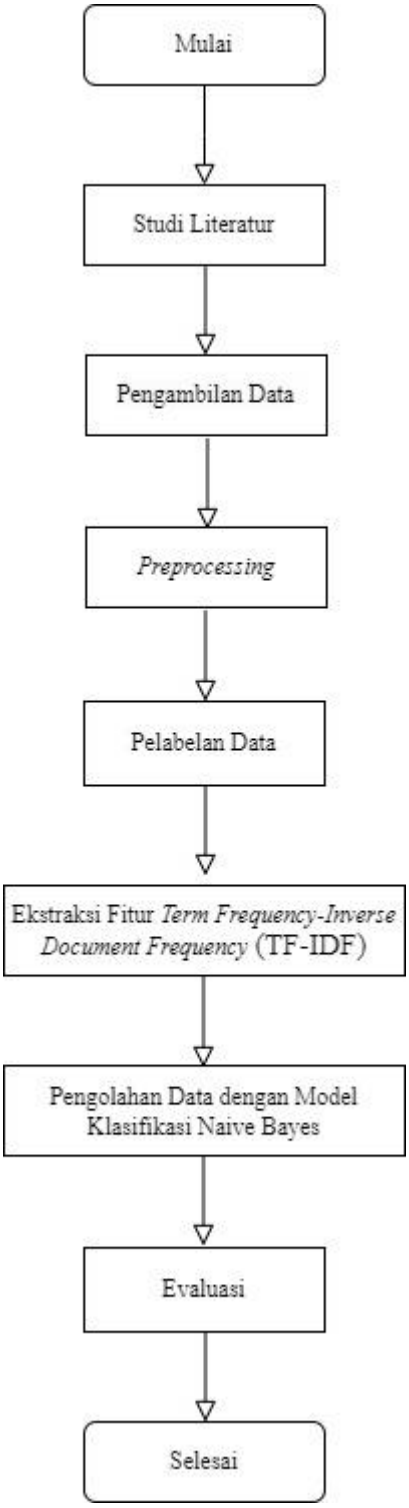
No	Perangkat Lunak
1	Sistem Operasi Windows 11
2	Twitter
3	*RapidMiner*
4	*Jupyter Notebook*
5	*Google Colab*
6	*Web browser*
7	*Microsoft Excel*
8	*Microsoft Word*

3.1.3 Bahan

Bahan Penelitian yang dibutuhkan dan digunakan pada penelitian ini adalah data opini berbahasa Indonesia yang diambil dari media sosial twitter menggunakan *RapidMiner* dengan kata kunci “Kpop” sebanyak 3.000 data tweet.

3.2 ALUR PENELITIAN

Tahapan yang dilakukan pada penelitian ini ditunjukkan dengan diagram alur penelitian pada Gambar 3.1 berikut:



Gambar 3. 1 Diagram Alur Penelitian.

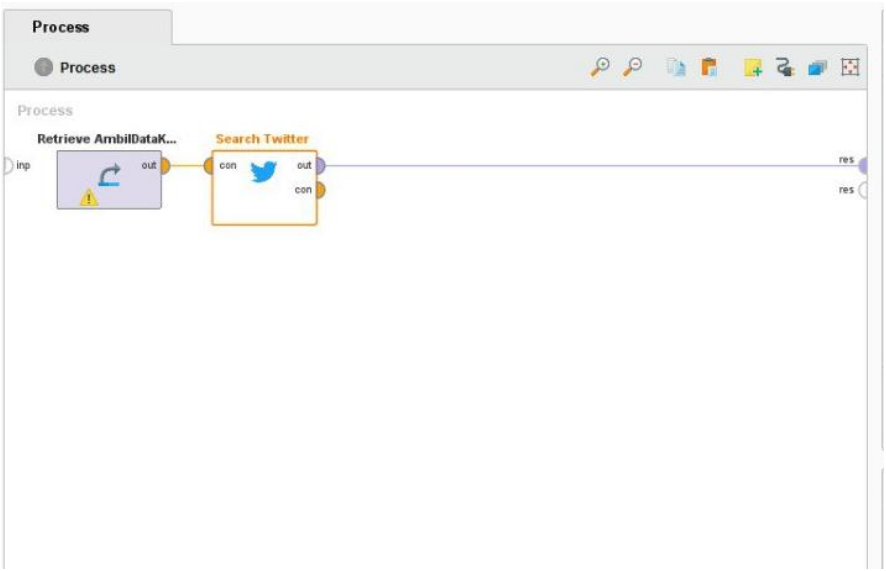
3.2.1 Studi Literatur

Pada proses ini data dikumpulkan dari berbagai referensi seperti buku, jurnal dan berbagai jenis penelitian terdahulu. Referensi di cari berdasarkan data yang sesuai tentang analisis sentimen, *Crawling* data, dan metode Naïve Bayes yang akan digunakan pada penelitian ini.

3.2.2 Pengambilan Data

Pada penelitian ini proses pertama yang dilakukan adalah pengambilan data. Pengambilan data merupakan hal yang mutlak harus diselesaikan karena seluruh proses klasifikasi akan berdasar dari data. Proses pengambilan data dilakukan dengan teknik *crawling* pada Twitter API menggunakan *tools RapidMiner*. Tahap yang dilakukan pada proses *crawling* data di *RapidMiner* adalah:

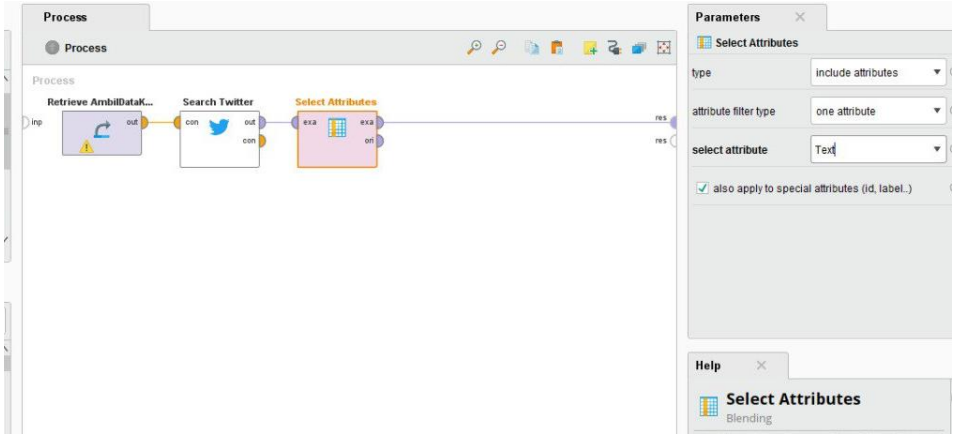
1. Menentapkan kata kunci yang tepat dengan permasalahan penelitian. *RapidMiner* sebagai *tools* akan mengekstrak data dari media sosial *Twitter* menggunakan API Twitter. Kata kunci yang digunakan adalah kpop. Data yang diambil merupakan kumpulan tweet berbahasa Indonesia dengan *result type* yaitu *recent and popular.* Proses *crawling data* dapat dilihat pada Gambar 3.2 berikut ini.



Gambar 3. 2 Rangkaian Operator pengambilan data pada *RapidMiner.*

2. Proses pembersihan *attribute* dilakukan karena hasil data yang di dapatkan pada tahap *crawling* data memiliki beberapa kolom *attribute* yang tidak di perlukan untuk tahap selanjutnya. Seperti id, waktu, *user* dan *source*.

Sedangkan yang di perlukan hanya kolom text saja. Sehingga, pada tahap ini ditambahkan *filter select attibutes.* tujuannya menghilangkan kolom yang tidak diperlukan dan hanya menyisakan kolom text saja sebagai kolom yang berisikan isi tweet yang akan dianalisis. Kemudian, data di simpan dalam bentuk csv. Proses pembersihan *attribute* dapat dilihat pada Gambar 3.3.



Gambar 3. 3 Rangkaian Operator Pembersihan *Attribute.*

Kriteria pengumpulan data untuk penelitian ini adalah sebagai berikut:

1. Data dalam bahasa Indonesia yang bersumber dari *twitter*. Metode yang digunakan yaitu *crawling data* menggunakan tools *RapidMiner* dengan memanfaatkan API *Twitter*
2. Data menyesuaikan permasalahan dengan kata kunci terkait yaitu KPop.
3. Data yang diambil dengan parameter *recent and popular*, sehingga data yang didapat adalah data-data terkini dan global sebanyak 3.000 data

Contoh hasil data *tweet* yang berhasil di ambil di tunjukkan pada Tabel 3.3 berikut ini:

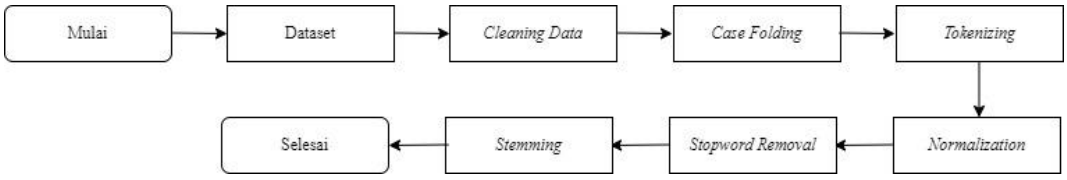
Tabel 3. 3 Sampel Hasil Pengambilan Data *Tweet.*

No	***Text***
1	RT @_AmranFans: Padah apabila menghina Kpopers ? https://t.co/2yQy3QCAFb
2	Ya allah yang delulu alias halunya model ginii ada juga toh di zona kpopers ternyataa oh ternyataa,brasa kek dejavu??? https://t.co/A31H7dQZav

No	*Text*
3	AHAGSHSHHSHSHS aku as kpopers pun kadang rasa annoyed dengan fans2 macam ni https://t.co/2LUFlvcXnz
4	@tanyakanrl Ya gpp, namanya selera. Dia tinggal nyari cewek yg bukan kpopers toh? Jadi tidak perlu ada drama saling memilih. Semua hepi.
5	@daisydraft @_AmranFans Diam la, bodoh punya kpopers. Aku pun minat kpop tpi x de benak mcm kau. Lain mcm betul

3.2.3 *Preprocessing* Data

Tahap *preprocessing data* isinya terdiri dari beberapa tahap yang terlihat pada ambar 3.4.



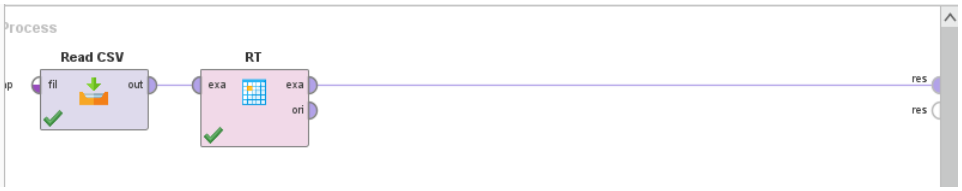
Gambar 3. 4 Diagram Preprocessing.

3.2.3.1 Cleaning Data

Pada tahap ini dilakukan proses pembersihan data yang masih memiliki banyak *noise* yang bisa mengganggu proses analisis. Proses pembersihan data dilakukan dengan beberapa tahap menggunakan operator *replace* yang sudah disediakan oleh *RapidMiner* dengan menghilangkan kata-kata sebagai berikut:

1) *Retweet* (RT)

Operator *replace* digunakan untuk menghilangkan retweet pada tweet yang sudah diambil. Retweet merupakan fitur untuk memposting ulang sebuah tweet milik orang lain atau milik sendiri dan membagikannya kembali kepada pengikut di twitter. Proses penghapusan retweet dapat dilihat pada gambar 3.5

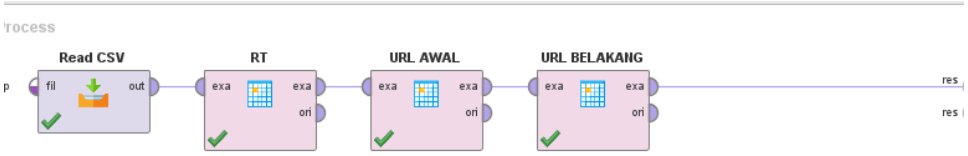


Gambar 3. 5 Rangkaian Operator Pembersihan Retweet.

Gambar 3.5 menunjukkan bahwa penggunaan operator read csv untuk membaca file data yang sudah di crawling sebelumnya yang berformat csv. Kemudian digunakan operator *replace* yang sudah di ganti nama menjadi RT untuk menghilangkan retweet.

2) *Link*

Operator *replace* Kembali digunakan untuk menghilangkan link atau URL pada tweet yang mengandung link. Link bisa berada di depan, tengah dan belakang kalimat, sehingga dibutuhkan 2 operator *replace* untuk menghilangkan link yang ada. Proses penghapusan link dapat dilihat pada Gambar 3.6.

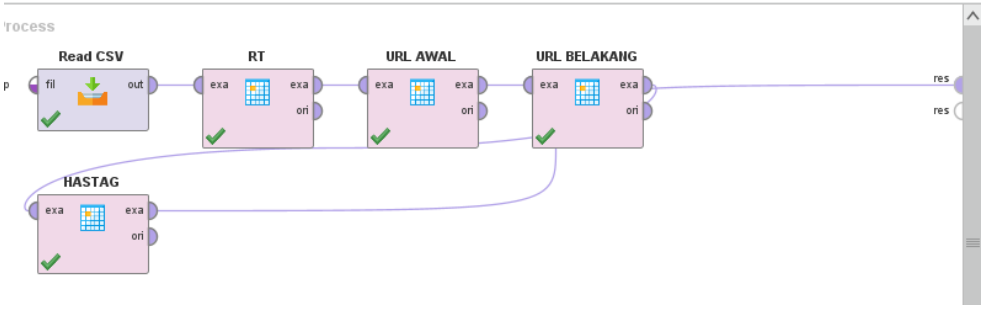


Gambar 3. 6 Rangkaian Operator Pembersihan Link.

Pada Gambar 3.6 digunakan operator *replace* pertama untuk menghilangkan link yang di depan dan tengah kalimat, selanjutnya operator *replace* berikutnya untuk menghilangkan link yang ada di belakang kalimat.

3) *Hastag* (#)

Tahap ini, operator *replace* digunakan untuk menghilangkan *Hastag* pada tweet yang sudah diambil. *Hastag* merupakan lambang '#' diikuti kata. Biasanya, *hastag* dipakai pada tweet untuk menyambungkan topik tertentu agar lebih mudah dicari. Proses penghapusan link dapat dilihat pada Gambar 3.7

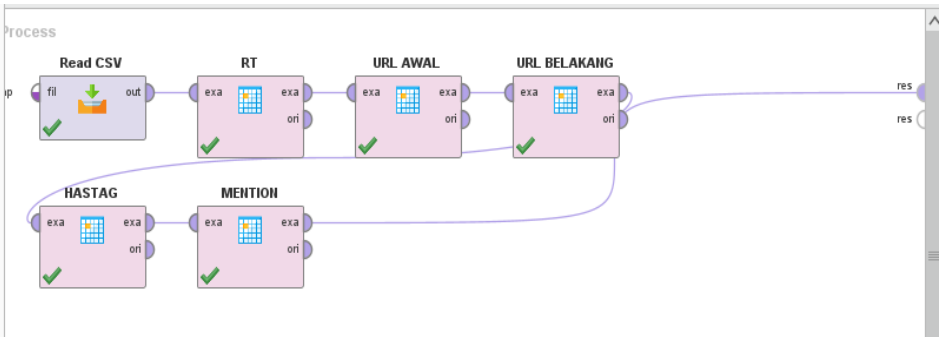


Gambar 3. 7 Rangkaian Operator Pembersihan Hastag.

pada Gambar 3.7 ditunjukkan bahwa digunakan operator *replace* untuk menghilangkan Hastag.

4) *Mention twitter* (@username)

Mention ialah penyebutan username ketika pengguna twitter menandai akun seseorang pada tweetnya. Proses penghapusan mention dapat dilihat pada gambar 3.8

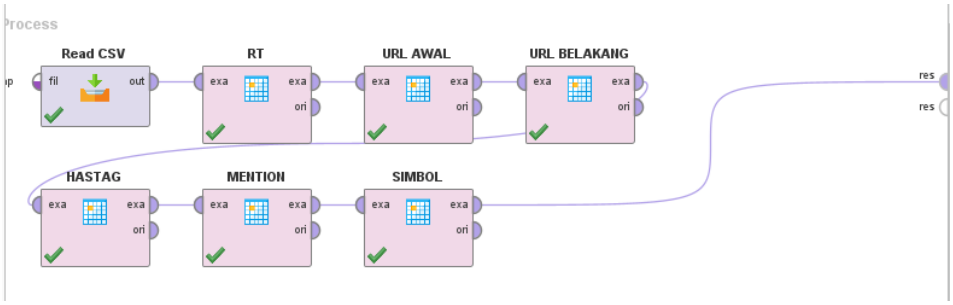


Gambar 3. 8 Rangkaian Operator Pembersihan Mention.

pada Gambar 3.8 ditunjukkan bahwa digunakan operator *replace* kembali untuk menghilangkan mention.

5) Simbol/Tanda baca

Operator *replace* Kembali digunakan untuk menghilangkan symbol atau tanda baca pada tweet yang sudah diambil. Proses penghapusan simbol dapat dilihat pada Gambar 3.9



Gambar 3. 9 Rangkaian Operator Pembersihan Simbol.

6) *Enter* dan *space* yang tidak perlu

Operator trim digunakan untuk menghilangkan spasi atau enter yang tidak perlu. Ada beberapa tweet yang memiliki spasi terlalu banyak. sehingga, operator ini berfungsi untuk menghilangkan spasi yang berlebihan tersebut. Proses penghapusan spasi dan enter ditunjukkan pada Gambar 3.10.

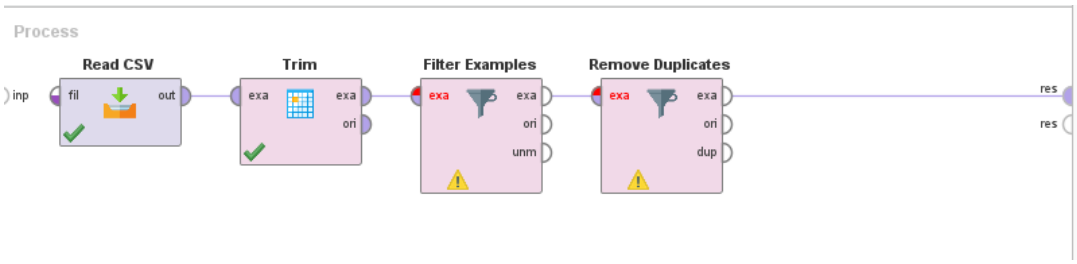
Gambar 3. 10 Rangkaian Operator Pembersihan Spasi Dan Enter tidak perlu.

Setelah melakukan proses *Cleaning*, tweet yang awalnya masih memiliki beberapa *noise* sudah bersih sehingga dapat digunakan untuk proses selanjutnya. Contoh hasil dari *cleaning* data dapat dilihat pada Tabel 3.4 dibawah ini

Tabel 3. 4 *Cleaning* Data.

No	Sebelum Melakukan *Cleaning*	Setelah Melakukan *Cleaning*
1	RT @_AmranFans: Padah apabila menghina Kpopers. ? https://t.co/2yQy3QCAFb	Padah apabila menghina kpopers

Selanjutnya, setelah membersihkan tweet dari *noise*, ada beberapa tweet yang kosong, dan memiliki isi yang sama. sehingga perlu di hilangkan dengan menggunakan *filter example* untuk menghapus *tweet* yang kosong dan *operator remove duplicate* untuk menghapus *tweet* yang sama. Proses ini dapat dilihat pada Gambar 3.11.



Gambar 3. 11 Rangkaian Operator Pembersihan Tweet Kosong dan Duplikat.

3.2.3.2 Case Folding

Proses *case folding* merupakan proses untuk mengubah huruf menjadi format yang sama. Pada penelitian ini seluruh huruf di ubah menjadi huruf kecil semua atau nonkapital. *Script* proses *case folding* dapat dilihat dibawah ini:

```
#caseFolding
def case_folding(Text):
    # konversi huruf kapital ke huruf kecil semua
```

```
    Text = Text.lower()
    return Text
df['Text'] = df['Text'].str.lower()
df
```

Berikut contoh hasil dari proses *case folding* dapat dilihat pada Tabel 3.5 berikut.

Tabel 3. 5 Contoh Hasil *Case Folding*.

No	Sebelum Melakukan *Case Folding*	Setelah Melakukan *Case Folding*
1	Padah apabila menghina Kpopers	padah apabila menghina kpopers

3.2.3.4 Tokenize

Tokenize merupakan proses penguraian deskripsi yang awalnya berupa kalimat-kalimat menjadi kata-kata. kata akan di pisahkan menjadi satu atribut tersendiri. *Script* proses Tokenize dapat dilihat dibawah ini:

```
#Tokenizing : membagi suatu kalimat menjadi perkata
def tokenization(Text):
    Text = re.split('\W+', Text)
    return Text

df['Tokenization'] = df['Text'].apply(lambda x:
tokenization(x.lower()))
df
```

Contoh hasil dari proses *tokenize* dapat dilihat pada Tabel 3.6.

Tabel 3. 6 Contoh Hasil *Tokenize*.

No	Sebelum Melakukan *Tokenize*	Setelah Melakukan *Tokenize*
1	padah apabila menghina kpopers	[‘padah’, ‘apabila’, ‘menghina’, ‘kpopers’]

3.2.3.5 Normalization

Proses *normalization* dilakukan dengan tujuan memperbaiki kata-kata yang disingkat atau salah eja yang memiliki bentuk tertentu tetapi memiliki arti yang sama. Hal ini untuk mendapatkan dokumen yang berkualitas baik. *Script* proses *normalization* ditunjukkan dibawah ini:

```
Normalization = pd.read_csv('/content/colloquial-indonesian-
lexicon.csv')
Normalization_dict = {}
```

```
for index, col in Normalization.iterrows():
  if col[0] not in Normalization_dict:
    Normalization_dict[col[0]] = col[1]

def Normalization(document):
  return [Normalization_dict[term] if term in
Normalization_dict else term for term in document]


df['Normalization'] = df['Tokenization'].apply(Normalization)
df
```

Contoh hasil dari proses *normalization* dapat dilihat pada Tabel 3.7.

Tabel 3. 7 Contoh Hasil *Normalization*.

No	Sebelum Melakukan Normalization	Setelah Melakukan *Normalization*
1	Kpopers ini otaknya emg pada dibobol semua	[‘Kpopers’, ‘ini’, ‘otaknya’, ‘emang’, ‘pada’, ‘dibobol’, ‘semua’]

3.2.3.6 Stopword removal

Proses *stopword removal* adalah proses pemilihan kata penting dan menghilangkan kata yang dianggap tidak penting. *Script* proses *Stopword removal* ditunjukkan dibawah ini:

```
#Stopword Removal
# ----------------------- get stopword from NLTK stopword ---
----------------------------
# get stopword indonesia
nltk.download('stopwords')
stopword = nltk.corpus.stopwords.words('indonesian')

def remove_stopwords(Text):
    Text = [word for word in Text if word not in stopword]
    return Text

df['Stop_Removal'] = df['Normalization'].apply(lambda x:
remove_stopwords(x))
df.head(1119)
```

Contoh hasil dari proses *normalization* dapat dilihat pada Tabel 3.8.

Tabel 3. 8 Contoh Hasil *Stopword Removal*.

No	Sebelum Melakukan *Stopword Removal*	Setelah Melakukan *Stopword Removal*
1	padah apabila menghina kpopers	[‘padah’, ‘menghina’, ‘kpopers’]

3.2.3.7 *Stemming*

Stemming ialah proses pengambilan kata dasar dengan menghilangkan imbuhan pada suatu kata. *Script* proses *Stemming* ditunjukkan dibawah ini:

```
# STEMMING
# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
    return stemmer.stem(term)

term_dict = {}

for document in df['Stop_Removal']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '

print(len(term_dict))
print("------------------------")

for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    print(term,":" ,term_dict[term])

print(term_dict)
print("------------------------")


# apply stemmed term to dataframe
def get_stemmed_term(document):
    return [term_dict[term] for term in document]

df['Tweet_Stemmed'] =
df['Stop_Removal'].swifter.apply(get_stemmed_term)
df
```

Tabel 3.9 Berikut adalah contoh hasil dari proses *Stemming*.

Tabel 3. 9 Contoh Hasil *Stemming*.

No	Sebelum Melakukan *Stemming*	Setelah Melakukan *Stemming*
1	padah apabila menghina kpopers	[‘padah’, ‘hina’, ‘kpopers’]

3.2.4 Pelabelan Data

Data yang sudah didapatkan pada proses *crawling data* dan dibersihkan dari *noise,* Kemudian dilakukan pelabelan data ulasan. Sebelum melakukan proses pelabelan, data Kembali di seleksi dengan hanya mengambil tweet yang berisikan sentimen dan menghapus tweet yang berisikan iklan atau tweet yang tidak mengandung sentimen. Tahap selanjutnya adalah *labeling* data. Proses ini akan membagi data ke dalam kelas yang sesuai. Dalam penelitian ini terdapat 3 kelas yaitu Positif, Netral, dan Negatif. Proses pelabelan pada penelitian ini dilakukan secara manual dengan dibantu oleh ahli bahasa Indonesia.

3.2.5 Pembobotan *Term Frequency-Inverse Document Frequency* (TF-IDF)

Data dari hasil proses *prepocessing* akan dibobotan dengan menggunakan metode TF-IDF. Proses ini akan di hitung berapa kali satu kata muncul dalam dokumen dan frekuensi kemunculan suatu kata pada keseluruhan dokumen. Tujuannya adalah untuk menunjukkan tingkat kepentingan sebuah kata dalam dokumen. Proses perhitungan TF-IDF ini dilakukan oleh fitur karena banyaknya kata yang akan di bobot. Sehingga tidak dapat dilakukan secara manual. Namun, contoh perhitungan manual dapat dijelaskan dibawah ini. Pada Tabel 3.10 dapat dilihat contoh dokumen yang berisi 2 buah dokumen dengan teks berbeda.

Tabel 3. 10 Contoh Dokumen.

No	*Text*
1	kamu seorang kpopers harusnya paham dong tujuan marketing yang sekarang ikutin korean wave itu karna apa
2	Kpopers ini otaknya emang pada dibobol semua

Tabel 3.11 menunjukkan hasil perhitungan nilai *Term Frequency* (TF). TF adalah frekuensi kemunculan *term*(t) pada dokumen (D) dimana setiap *term* akan membaca seluruh dokumen yang tersedia.

Tabel 3. 11 Menghitung TF.

***Term*(t)**	**D1**	**D2**
apa	1	0
dibobol	0	1
dong	1	0
emang	0	1
harusnya	1	0
ikutin	1	0
ini	0	1
itu	1	0
kamu	1	0
karna	1	0
kpopers	1	1
korean	1	0
marketing	1	0
otaknya	0	1
pada	0	1
paham	1	0
sekarang	1	0
semua	0	1
seorang	1	0
tujuan	1	0
wave	1	0
yang	1	0

Tabel 3.12 menunjukkan hasil perhitungan nilai *Document Frequency* (DF). DF merupakan hasil banyaknya dokumen dimana total kemunculan suatu *term* (t) yang dijumlahkan dari seluruh kata yang muncul dalam dokumen.

Tabel 3. 12 Menghitung DF.

***Term*(t)**	**DF**
apa	1
dibobol	1
dong	1

Term(t)	DF
emang	1
harusnya	1
ikutin	1
ini	1
itu	1
Kamu	1
karna	1
kpopers	2
korean	1
Marketing	1
Otaknya	1
Pada	1
Paham	1
sekarang	1
Semua	1
seorang	1
tujuan	1
wave	1
yang	1

Tabel 3.13 menunjukkan hasil perhitungan nilai IDF. IDF adalah perhitungan dari log keseluruhan dokumen dibagi dengan total kemunculan kata pada dokumen.

Tabel 3. 13 Menghitung IDF.

Term(t)	DF	IDF
apa	1	Log 2/1 = 0.3010
dibobol	1	Log 2/1 = 0.3010
dong	1	Log 2/1 = 0.3010
emang	1	Log 2/1 = 0.3010
harusnya	1	Log 2/1 = 0.3010
ikutin	1	Log 2/1 = 0.3010
ini	1	Log 2/1 = 0.3010

Term(t)	DF	IDF
itu	1	Log 2/1 = 0.3010
kamu	1	Log 2/1 = 0.3010
karna	1	Log 2/1 = 0.3010
kpopers	2	Log 2/2 = 0.1505
korean	1	Log 2/1 = 0.3010
marketing	1	Log 2/1 = 0.3010
otaknya	1	Log 2/1 = 0.3010
pada	1	Log 2/1 = 0.3010
paham	1	Log 2/1 = 0.3010
sekarang	1	Log 2/1 = 0.3010
semua	1	Log 2/1 = 0.3010
seorang	1	Log 2/1 = 0.3010
tujuan	1	Log 2/1 = 0.3010
wave	1	Log 2/1 = 0.3010
yang	1	Log 2/1 = 0.3010

Selanjutnya menghitung TF-IDF yang dapat dilihat pada Tabel 3.14

Tabel 3. 14 Menghitung TF-IDF.

Term(t)	TF		IDF	TF-IDF	
	D1	D2		D1	D2
apa	1	0	Log 2/1 = 0.3010	0.3010	0
dibobol	0	1	Log 2/1 = 0.3010	0	0.3010
dong	1	0	Log 2/1 = 0.3010	0.3010	0
emang	0	1	Log 2/1 = 0.3010	0	0.3010
harusnya	1	0	Log 2/1 = 0.3010	0.3010	0
ikutin	1	0	Log 2/1 = 0.3010	0.3010	0
ini	0	1	Log 2/1 = 0.3010	0	0.3010
itu	1	0	Log 2/1 = 0.3010	0.3010	0
kamu	1	0	Log 2/1 = 0.3010	0.3010	0
karna	1	0	Log 2/1 = 0.3010	0.3010	0
kpopers	1	1	Log 2/2 = 0.1505	0.07525	0,07525
korean	1	0	Log 2/1 = 0.3010	0.3010	0

Term(t)	TF		IDF	TF-IDF	
	D1	D2		D2	D2
marketing	1	0	Log 2/1 = 0.3010	0.3010	0
otaknya	0	1	Log 2/1 = 0.3010	0	0.3010
pada	0	1	Log 2/1 = 0.3010	0	0.3010
paham	1	0	Log 2/1 = 0.3010	0.3010	0
sekarang	1	0	Log 2/1 = 0.3010	0.3010	0
semua	0	1	Log 2/1 = 0.3010	0	0.3010
seorang	1	0	Log 2/1 = 0.3010	0.3010	0
tujuan	1	0	Log 2/1 = 0.3010	0.3010	0
wave	1	0	Log 2/1 = 0.3010	0.3010	0
yang	1	0	Log 2/1 = 0.3010	0.3010	0

Modul atau fungsi yang digunakan pada tahap ini adalah TFIDFVectorizer, dimana modul ini akan mengkonversi kumpulan dokumen data ulasan menjadi fitur TF-IDF. Pada proses ekstraksi fitur menggunakan TFIDFVectorizer, dapat di ambil seluruhnya atau disaring seberapa sering kata tersebut muncul, hal tersebut dapat mempengaruhi akurasi dari proses klasifikasi selanjutnya. Setelah melewati proses TF-IDF maka tiap kata pada setiap baris akan memiliki nilai bobot. Nilai bobot tentunya dipengaruhi oleh seberapa sering kata atau seberapa banyak frekuensi data itu muncul dalam dataset yang dimiliki. Pada tahap ini, penulis mencoba melakukan 5 skenario :

3.2.5.1 Skenario 1

Menggunakan TFIDFVectorizer() yaitu tanpa min_df, artinya semua kata yang ada pada dataset akan diberi bobot dan digunakan untuk proses pengolahan data dengan model prediksi algoritma Naïve Bayes. Script TFIDF skenario 1 dapat dilihat dibawah ini:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf = TfidfVectorizer()
X_train_tfidf = tfidf.fit_transform(X)
```

3.2.5.2 Skenario 2

Menggunakan TFIDFVectorizer(min_df=2), artinya kata yang akan diberi bobot dan digunakan untuk proses pengolahan data dengan model prediksi

algoritma Naïve Bayes adalah kata yang ada di data set dengan minimal kemunculan 2 kali. Script TFIDF skenario 2 dapat dilihat dibawah ini:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf = TfidfVectorizer(min_df=2)
X_train_tfidf = tfidf.fit_transform(X)
```

3.2.5.3 Skenario 3

Menggunakan TFIDFVectorizer(min_df = 3), artinya kata yang akan diberi bobot dan digunakan untuk proses pengolahan data dengan model prediksi algoritma Naïve Bayes adalah kata yang ada di data set dengan minimal kemunculan 3 kali. Script TFIDF skenario 3 dapat dilihat dibawah ini:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf = TfidfVectorizer(min_df=3)
X_train_tfidf = tfidf.fit_transform(X)
```

3.2.5.4 Skenario 4

Menggunakan TFIDFVectorizer(min_df = 4), artinya kata yang akan diberi bobot dan digunakan untuk proses pengolahan data dengan model prediksi algoritma Naïve Bayes adalah kata yang ada di data set dengan minimal kemunculan 4 kali. Script TFIDF skenario 4 dapat dilihat dibawah ini:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf = TfidfVectorizer(min_df=4)
X_train_tfidf = tfidf.fit_transform(X)
```

3.2.5.5 Skenario 5

Menggunakan TFIDFVectorizer(min_df = 5), artinya kata yang akan diberi bobot dan digunakan untuk proses pengolahan data dengan model prediksi algoritma Naïve Bayes adalah kata yang ada di data set dengan minimal kemunculan 5 kali. Script TFIDF skenario 5 dapat dilihat dibawah ini:

```
# Ekstraksi fitur menggunakan TF-IDF
tfidf = TfidfVectorizer(min_df=5)
X_train_tfidf = tfidf.fit_transform(X)
```

3.2.6 Pengolahan Data Dengan Model Prediksi Algoritma Naïve Bayes

Untuk melakukan klasifikasi sentimen, akan digunakan data hasil *preprocessing* yang sudah diberikan bobot pada proses ekstraksi fitur TF-IDF.

Naïve Bayes *Classifier* adalah algoritma yang di gunakan untuk pengujian dataset penelitian ini. Data yang sudah di siapkan sebanyak 1.119 data akan dibagi menjadi data *test* sebesar 20% dan data *training* sebesar 80%. *Script* pembagian data *training* dan data *test* dapat dilihat dibawah ini:

```
# Pembagian data menjadi data latih dan data uji
X_train, X_test, y_train, y_test =
train_test_split(X_train_tfidf, y, test_size=0.2,
random_state=30) #Tetapkan random state, agar ketika
dirunning ulang, hasilnya tidak berubah
```

Hasil pembagian data uji dan data latih dapat dilihat pada Tabel 3.115 dibawah ini.

Tabel 3. 15 Hasil Pembagian Data Uji Dan Data Latih.

	Negatif	**Netral**	**Positif**	**Total**
Training	293	227	375	895
Testing	85	39	100	224

Algoritma naïve bayes akan mempelajari pola data dan menghasilkan model klasifikasi untuk menentukan sentimen dari tweet yang belum memiliki label sentimen. *Script* pengolahan menggunakan algoritma naive bayes dapat dilihat dibawah ini:

```
# Melatih model Naive Bayes
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
# Memprediksi sentimen pada data uji
y_pred = nb_classifier.predict(X_test)
```

Pada tahap ini akan ditampilkan hasil dari 5 skenario yang dilakukan pada tahap TF-IDF sebelumnya.

3.2.7 Evaluasi

Tahap evaluasi akan menggunakan *confusion matrix* untuk mengetahui performansi dari algoritma *Naïve Bayes.* Tujuan evaluasi adalah untuk dapat melihat performa model hasil klasifikasi yang berupa nilai *accuracy* dimana berdasarkan *true* positif, *true* negatif, *true* netral, *false* positif, *false* negatif, *false* netral. Selain itu, terdapat juga nilai *precision, recall* dan F1-*Score* yang dihasilkan model untuk menganalisis senimen opini pengguna twitter terhadap KPop.