BAB IV
HASIL DAN PEMBAHASAN

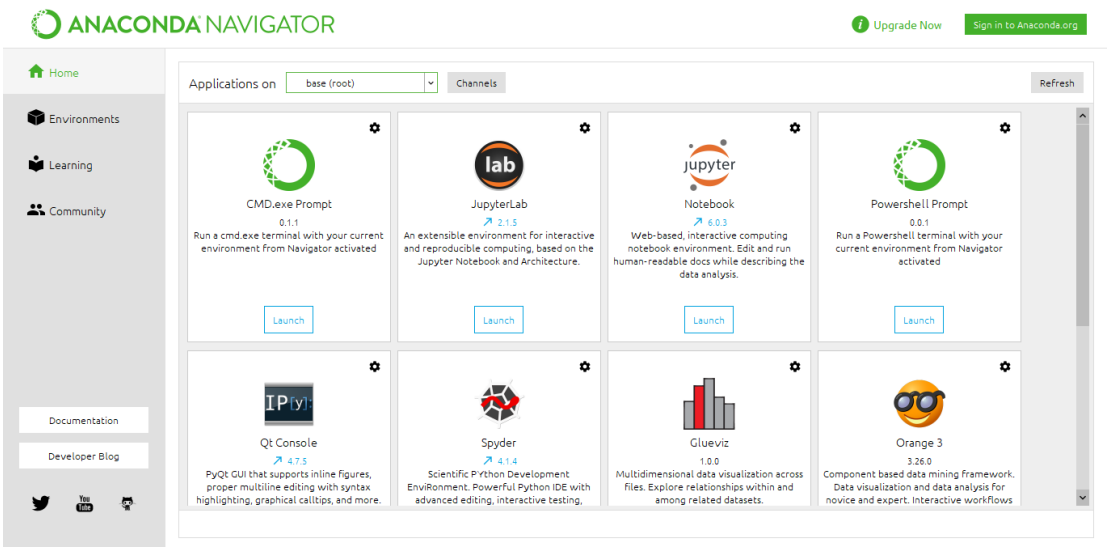
4.1 Pengambilan Data

Pengambilan data ini dilakukan dengan penggunaan dan dukungan bahasa pemrograman *Python* menggunakan kunci API Twitter. API *Key* Twitter adalah API yang memiliki perintah dan komponen untuk memfasilitasi pengumpulan data untuk program. API *Key* Twitter berisi kunci konsumen, akses token, kunci akses, dan akses token *secret* yang penting bagi perangkat lunak untuk mengakses informasi Twitter.

```
api_key = "JhCOoqSewsBkIHr1j4MpQN9t3"
api_secret_key = "76okTgjTzrhuqCgvgveQsN7GGxla8gZEdHusQp7449Tzq2YCza"
access_token = "1306832033612611589-WYNEngDG7VB4XkJ4QQQFiRM5WRiy9q"
access_token_secret = "XRyfkdC5qhk1X0HPxRnOOnCWCe9jAaxWQ5y0HtRqUALqX"
```

Gambar 4. 1 API *Key* Twitter

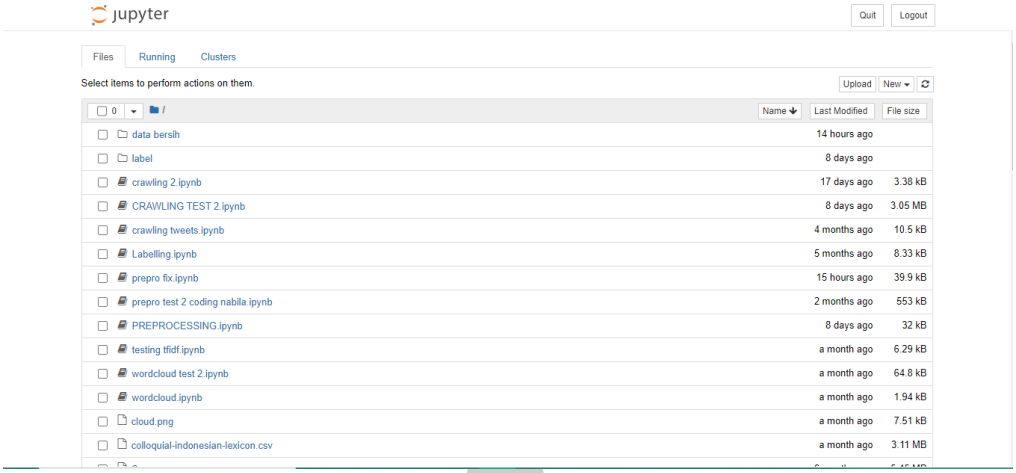
Selain menggunakan API *key* Twitter peneliti juga menggunakan *anaconda python* sebagai *tools* pendukung analisis sentimen.



Gambar 4. 2 Tampilan *Anaconda*

Anaconda digunakan oleh para peneliti karena *jupyter notebook* ada di *Anaconda python*, *jupyter notebook* merupakan aplikasi *web open source* yang dapat berjalan di

localhost computer. *jupyter notebook* bisa melakukan beberapa hal seperti menulis kode *python*, persamaan matematika, serta visualiasasi. *Jupyter notebook* merupakan project pengembangan dari Python atau i*nteractive pyhton*.



Gambar 4. 3 *Jupyter Notebook*

```
import tweepy
import json
import csv
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener

api_key = 'X2Js6dnArAy4XgoL9my1sAEjd'
api_secret = '0kx7e9nUKTiT8b0YDhyAaSOkaqgnuTsir3m6czWa4miFnCEYs6'
access_token = '1306832033612611589-DhTti0uYqhrZ6a1dV9rPAPVfXlS96F'
access_secret = 'uS2HDtVFJYglmVFXFvA9uAhX8MlWA1uP7NhUHIm3oJNbI'

auth = OAuthHandler(api_key, api_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)

#setup access API
def connectOAuth():
    auth = OAuthHandler(api_key, api_secret)
    auth.set_access_token(access_token, access_secret)

    api = tweepy.API(auth, wait_on_rate_limit=True)
    return api


    def on_error(self, status):
        print(status)
        return True
```

Gambar 4. 4 Kode Program *Library Crawling*

Pada gambar 4.4 peneliti melakukan Teknik crawling dengan memanfaatkan *key twitter* menggunakan *library* seperti *library tweepy*, *library* CSV, dan *library json* lalu mengimport *OAuthHandler*, *Stream*, dan *StreamListener* dari *library tweepy*. *Library*

tweepy merupakan *library* yang di sediakan oleh *python* untuk pengaksesan dan pengambilan data menggunakan API dari Twitter. lalu terdapat *library* CSV (*commad separed vales*) untuk membaca tipe file CSV. Setelah tu masukan *consumer keys*, *consumer secret*, *access key* dan *access secret*. Lalu *set up access* ke API menggunakan *code connectOAuth()*. Setelah itu masukan kode diatas, selanjutnya masukan kode proses seperti gambar

```
csvFile = open('vaksin20210304.csv')
csvWriter = csv.writer(csvFile)
for tweet in tweepy.Cursor(api.search,q="vaksinasi covid", count = 1000,
                           lang="id").items():
    print (tweet.created_at, tweet.text)


    csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
```

Gambar 4. 5 Kode Program Proses *Crawling*

Jalankan *source code* diatas dengan menggunakan *key words* "vaksinisasi covid". Sesudah menjalani seluruh kode nantinya akan mendapatkan file CSV yang diperoleh dalam hasil *crawling.*

waktu	tweet
3/4/2021 7:47	b'RT @drmusanordin: Kesaksamaan & keadilan akses vaksin mesti dijaga\nHentikan politik pandemik ini!\nKita sudah mual dgn kata2 polititik yg m\xe2\x80\xa6'
3/4/2021 7:47	b'Waspada Penipuan online.\nAda ribuan penipu sdh bersiap memanfaatkan program vaksinasi Covid-19.Penipuan akan terlih\xe2\x80\xa6 https://t.co/uyo36rHxSG'
3/4/2021 7:47	b'RT @drmusanordin: Kadar Prevalens COVID di-S\xe2\x80\x99gor adalah yg ke-3 tertinggi\nWalaupun bebanan kerja yang tinggi, kadar kematian diS\xe2\x80\x99gor adalah\xe2\x80\xa6'
3/4/2021 7:46	b'RT @DocMummy22: Menurut guideline terbaru untuk vaksinasi Covid 19, ibu hamil perlu berbincang dengan doktor. Manakala ibu menyusu tiada ma\xe2\x80\xa6'
3/4/2021 7:46	b'RT @drmusanordin: Jawatan Kuasa Jaminan Akses Vaksin (JKJAV) tolong-lah jangan buat kesilapan seperti KKM dengan hanya melapur nombor orang\xe2\x80\xa6'
3/4/2021 7:46	b'RT @NayDonuts: 4\xe2\x80\xa2 Mengingat ketersediaan vaksin kita banyak, pst perlu strategi jitu guna mempercepat progres vaksinasi, agar program pembe\xe2\x80\xa6'
3/4/2021 7:46	b'RT @DocMummy22: Menurut guideline terbaru untuk vaksinasi Covid 19, ibu hamil perlu berbincang dengan doktor. Manakala ibu menyusu tiada ma\xe2\x80\xa6'
3/4/2021 7:46	b'RT @drmusanordin: Jawatan Kuasa Jaminan Akses Vaksin (JKJAV) tolong-lah jangan buat kesilapan seperti KKM dengan hanya melapur nombor orang\xe2\x80\xa6'
3/4/2021 7:46	b'Haedar Nashir: Vaksinasi Bagian dari Ikhtiar Kolektif Atasi Wabah\xc2\xa0Covid-19 https://t.co/JJpsapTsY'
3/4/2021 7:46	b'RT @YRadianto: Grafik ini memperlihatkan perlawanan kita atas Pandemi Covid-19 dibanding negara\xc2\xb2 sekitar, nih..\n\nIni itungan 1 dosis, semen\xe2\x80\xa6'
3/4/2021 7:46	b'Haedar Nashir: Vaksinasi Bagian dari Ikhtiar Kolektif Atasi Wabah\xc2\xa0Covid-19 https://t.co/uuIu5GdJSk'
3/4/2021 7:46	b'VAKSINASI covid-19 untuk lanjut usia (lansia) di Kota Kupang, Nusa Tenggara Timur (NTT) sudah dimulai sejak Rabu (3\xe2\x80\xa6 https://t.co/RGShUoPWts'
3/4/2021 7:46	b'RT @YRadianto: Grafik ini memperlihatkan perlawanan kita atas Pandemi Covid-19 dibanding negara\xc2\xb2 sekitar, nih..\n\nIni itungan 1 dosis, semen\xe2\x80\xa6'
3/4/2021 7:46	b'RT @OmDennis: Bagi yg ingin daftarkan orangtuanya untuk vaksin gratis COVID-19, ini link-nya sesuai daerah bersangkutan. Daftar dulu, janga\xe2\x80\xa6'
3/4/2021 7:46	b'Personil Polsek Bunut Laksanakan Vaksinasi Covid-19 https://t.co/zjJ93fTOM9'
3/4/2021 7:46	b'Wartawan Probolinggo Terima Vaksinasi Covid-19 https://t.co/pEK6SucskN'
3/4/2021 7:45	b'Grafik ini memperlihatkan perlawanan kita atas Pandemi Covid-19 dibanding negara\xc2\xb2 sekitar, nih..\n\nIni itungan 1 dos\xe2\x80\xa6 https://t.co/XpXRqNa6oG'
3/4/2021 7:45	b'*Sangarnya Jokowi Di mata Dunia, PBB Pusing Banyak Negara Belum Vaksin, Kita Kipas-Kipas!*\n\n Ketika Presiden Jokowi\xe2\x80\xa6 https://t.co/7G88vvHVmY'
3/4/2021 7:45	b'RT @DocMummy22: Menurut guideline terbaru untuk vaksinasi Covid 19, ibu hamil perlu berbincang dengan doktor. Manakala ibu menyusu tiada ma\xe2\x80\xa6'
3/4/2021 7:45	b'Layanan Drive Thru Vaksinasi Covid-19 Tersedia, Kelompok Lansia Jadi Prioritas\n\nhttps://t.co/PppWyyJ755'
3/4/2021 7:45	b'Kompol Sugimin S.H., M.M Kapolsek Cikarang Timur menghimbau kepada Anggota yang hendak melaksanakan Vaksinasi Covid\xe2\x80\xa6 https://t.co/OukcnS4NAj'
3/4/2021 7:44	b'RT @MrsRachelIn: Oh ,, dprd dki masih ada? Waktu Jkt banjir kmn aja? \nSbg badan pengawas cuma bs minta anggaran naik & dispensasi vaksin u\xe2\x80\xa6'

Gambar 4. 6 Hasil *Crawling*

Ketika memproses pengumpulan data Twitter, peneliti menggunakan *key words* "vaksinisasi covid". Proses pengambilan data dilakukan dari bulan November 2020 pada tanggal 24/11/20 dan diambil setiap minggunya. Dikarenakan API yang dipakai merupakan API *free* dan hanya bisa mengambil tweet satu minggu dari proses

crawling. Proses pengambilan data *crawling* dilanjutkan sampai bulan Maret 2021 pada tanggal 04/03/2021 pengambilan data hanya menggunakan tweet berbahasa ndonesia. berikut hasil dari pengambilan tweet yang dijalankan selama 4 bulan

Tabel 4. 1 Hasil *Crawling*

Tanggal	**Hasil *Crawling***
24/11/20	22.716
27/11/20	8.567
09/12/20	27.383
17/02/21	17.778
04/03/21	33.577

Data yang di dapatkan sekitar 109.481 tetapi pada dataset tersebut banyak sekali tweet yang terduplikat dan tweet berbahasa inggris dan menjadikan dataset tweet yang bisa dipakai hanya sebanyak 8.000 tweets.

4.2 Text *preprocessing*

Data yang dikumpulkan dari Twitter adalah file CSV dalam bentuk teks. Oleh karena itu, data harus dikonversi menjadi data terstruktur. Sebelum memasuki model pembelajaran, data teks harus dibersihkan, yang dikenal dengan *text preprocessing*. Proses *preprocessing* teks terdiri dari banyak langkah seperti pembersihan, tokeinizing, stemming dan lain-lain. Proses ini juga baik untuk menghilangkan beberapa bagian kata yang tidak berguna. Proses ini dilakukan dengan bantuan *library* bahasa pemrograman *python*. Tahap *text preproprocessing* akan dibahas seperti berikut

4.2.1 *Case Folding*

Case Folding diperlukan untuk *preprocessing*, untuk merubah huruf kapital menjadi *lower case* (teks lebih kecil). Untuk proses *script* dapat dilihat pada gambar Gambar 4.7

```
df["lower_case"] = df["tweet"].apply(str.lower)
```

Gambar 4. 7 Kode program *case folding*

Pada gambar 4.7 proses *case folding* menggunakan *str. lower* di mana berguna untuk merubah huruf kapital supaya menjadi huruf kecil

Tabel 4. 2 Hasil *Case Folding*

Input	Hasil *case folding*
b'**Warga** yang pernah mengikuti kegiatan berkerumun akan diminta jalaini tes vaksin. https://t.co/Axwc8vKZ7s'	b'**warga** yang pernah mengikuti kegiatan berkerumun akan diminta jalaini tes vaksin. https://t.co/Axwc8vKZ7s'

4.2.2 *Cleansing*

Cleansing adalah tahap pra-pemrosesan teks yang berupaya membersihkan teks dari tab, baris baru, back-slice, link, tautan, tagar, dan URL. Gambar 4.8 menunjukkan *cleansing script*.

```
import string
import re

from nltk.tokenize import word_tokenize

def cleaning(tweet):

    #remove ascii
    tweet = tweet.encode('ascii', 'replace').decode('ascii')
    #remove angka
    tweet = re.sub('[0-9]+', '', tweet)
    # remove RT
    tweet = re.sub(r'^RT[\s]+', '', tweet)
    # remove mention, link, hashtag
    tweet = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", tweet).split())
    tweet = re.sub('@[^\s]+','',tweet)
    tweet = re.sub(r'#', '', tweet)
    #remove url
    tweet = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/]*))*', '', tweet)
    #remove tanda baca
    tweet = re.sub(r'[^\w\d\s]+', '', tweet)
    #remove whitespace
    tweet = re.sub('\s+',' ',tweet)


    return tweet
df['tweet_cleansing'] = df['lower_case'].apply(cleaning)

# menghapus tweet duplikat
df.drop_duplicates(subset ="tweet", keep = 'first', inplace = True)
```

Gambar 4. 8 Kode Program *Cleansing*

Pada Gambar 4.8, dilihat menggunakan library re atau regex. Regex adalah mecocokan teks berdasarkan pola dalam bahasa atau sebagai tools untuk

mencari atau mengganti kata dalam sebuah teks. Contohnya library regex dapat menghilangkan link dengan membuat pola seperti [A-Za-z0-9] A-Za merupakan karakter huruf dari huruf kecil maupun besar, z0-9 merupakan semua huruf digit angka. Dapat dilihat hasil dari cleansing pada Tabel 4.3

Tabel 4. 3 Hasil *Cleansing*

Input	Hasil *cleansing*
b'Warga yang pernah mengikuti kegiatan berkerumun akan diminta jalaini tes vaksin. https://t.co/Axwc8vKZ7s'	warga yang pernah mengikuti kegiatan berkerumun akan diminta jalaini tes vaksin

4.2.3 Tokenisasi

Tokenisasi adalah persiapan teks yang bertujuan untuk membagi teks menjadi token. Tokenisasi juga berguna sebegai pemisah kata simbol, atau frase dari teks.

```
def word_tokenize_wrapper(text):
    return word_tokenize(text)

df['tweet_tokens'] = df['tweet_cleansing'].apply(word_tokenize_wrapper)
```

Gambar 4. 9 Kode Program Tokenisasi

Dapat dilihat pada Gambar 4.9 merupakan kode program tokenisasi, tokenisasi menggunakan library dari NLTK yaitu nltk.tokenize yang sudah di deklarasikan pada Gambar 4.8 dan Hasil dari proses tokenisasi dapat dilihat pada Table 4.4 dibawah ini

Tabel 4. 4 Hasil Tokenisasi

Input	Hasil tokenisasi
b'Warga yang pernah mengikuti kegiatan berkerumun akan diminta jalani tes vaksin. https://t.co/Axwc8vKZ7s'	['warga', 'yang', 'pernah', 'mengikuti', 'kegiatan', 'berkerumun', 'akan', 'diminta', 'jalani', 'tes', 'vaksin']

4.2.4 Remove stopwords

Remove stopwords berguna untuk menghilangkan kata-kata yang tidak memiliki makna seperti “yang”, “dan”, “tetapi” dan sebagainya. Pada proses remove stop word menggunakan library NLTK. NLTK memiliki 758 *stopword*. Selain memanfaatkan library NLTK, kata juga dapat ditambahkan dengan menggunakan fungsi secara manual ke daftar stopword *.extend()*. *Script* untuk proses *remove stopword* dapat dilihat pada Gambar 4.10

```
from nltk.corpus import stopwords

list_stopwords = stopwords.words('indonesian')
print(len(list_stopwords))

#add manual additional stopword
list_stopwords.extend(["yg", "dg", "brt", "dgn", "ny", "b", 'klo',
                       'kalo', 'amp', 'biar', 'bikin', 'bilang',
                       'gak', 'ga', 'krn', 'nya', 'nih', 'sih',
                       'si', 'tau', 'tdk', 'tuh', 'utk', 'ya',
                       'jd', 'jgn', 'sdh', 'aja', 'n', 't',
                       'nyg', 'hehe', 'pen', 'u', 'nan', 'loh', 'rt',
                       '&amp', 'yah', "xexxa"])

list_stopwords = set(list_stopwords)

def stopwords_removal(words):
    return [word for word in words if word not in list_stopwords]

df['tweet_tokens_WSW'] = df['tweet_tokens'].apply(stopwords_removal)
```

Gambar 4. 10 Kode Program *Remove Stopword*

Hasil dari proses remove stopword dapat dilihat pada Tabel 4.5 dibawah ini

Tabel 4. 5 Hasil *remove stopwords*

Input	Hasil *stopword*
warga yang pernah mengikuti kegiatan berkerumun akan diminta jalaini tes vaksin	['warga', 'mengikuti', 'kegiatan', 'berkerumun', 'jalaini', 'tes', 'vaksin']

4.2.5 Normalisasi

Normalisasi adalah langkah *preprocessing* teks yang bertujuan untuk mengoreksi kata atau kata yang disingkat serta kata yang terdapat salah pengetikan. Terdapat 1.320 kata yang ada di dalam dataset normalisasi. Contoh perbaikan kata alay menjadi kata baku seperti berikut

7an	:Tujuan	beud	:Banget
adlah	:Adalah	bgmn	:Bagimana
adlh	:Adalah	bgs	:Bagus
aja	:Saja	bgt	:Banget
ak	:saya	blh	:Boleh

Gambar 4. 11 Contoh Normalisasi

Script untuk proses normalisasi adalah sebagai berikut dapat dilihat pada gambar 4.11

```
normalizad_word = pd.read_excel("normalisasi.xlsx")

normalizad_word_dict = {}

for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]

def normalized_term(document):
    return [normalizad_word_dict[term] if term in normalizad_word_dict else term for term in document]

df['tweet_normalisasi'] = df['tweet_tokens_WSW'].apply(normalized_term)
```

Gambar 4. 12 Kode Program Normalisasi

hasil proses normalisasi dapat dilihat pada Tabel 4.6 dibawah ini

Tabel 4. 6 Hasil Normalisasi

Input	Hasil normalisasi
['cuman', 'tolak', 'tes', 'vaksin', 'kabur', 'bawa', 'kabur', 'pasien', 'covid', 'didenda', 'rp', 'juta']	['Cuma', 'tolak', 'tes', 'vaksin', 'kabur', 'bawa', 'kabur', 'pasien', 'covid', 'didenda', 'rp', 'juta']

4.2.6 *Stemming*

Stemming merupakan fase *pre-processing* teks yang efektif untuk menghilangkan imbuhan baik dari depan maupun belakang setiap kata. Gunakan dukungan perpustakaan sastrawi selama *stemming*. Gambar 4.13 menunjukkan skrip untuk prosedur *stemming*.

```
# import Sastrawi package
import Sastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter


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

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

df['tweet_stemmed'] = df['tweet_normalisasi'].apply(lambda x: [stemmer.stem(y) for y in x])
```

Gambar 4. 13 Kode Program *Stemming*

Pada gambar 4.13 dapat dilihat kode program *stemming* harus melakukan proses pendeklarasian *library* sastrawi. Hasil dari proses stemming dapat dilihat pada Tabel 4.7

Tabel 4. 7 Hasil *Stemming*

Input	Hasil *stemming*
['warga', 'mengikuti', 'kegiatan', 'berkerumun', 'jalaini', 'tes', 'vaksin']	['warga', 'ikut', 'giat', 'kerumun', 'jalan', 'tes', 'vaksin']

4.3 Analisis sentimen

Setelah melakukan *text preprocessing* maka selanjutnya akan dilakukan pelabelan kelas sentimen. Analisis sentimen dibagi menjadi dua jenis, analisis level dokumen dan analisis kalimat. Dalam penelitian ini, analisis sentimen dilakukan pada setiap kalimat karena setiap tweet memiliki sentimen.

Pada pelabelan sentimen analisis dilakukan berdasarkan kamus *lexicon*. Pelabelan kamus *lexicon* dilakukan dengan cara melabelkan data dengan memilih setiap kata ke dalam kata positif atau kata negatif. Setelah itu kata-kata yang sudah masuk ke kategori positif dan negatif akan dikalkulasikan dan dapat diketahui apakah tweet tersebut termasuk tweet positif atau tweet negatif.

Pada proses pelebalan kelas sentimen dilakukan dengan secara otomatis menggunakan *jupyer notebook.* System skor penilaian akan dihasilkan berupa positif

atau negatif, jika nilai skor sentimen di bawah 0 (sentimen < 0) maka system akan menilai sebagai sentimen negatif sedangkan jika nilai skor sentimen diatas 0 (sentimen >= 0) maka system akan menilai sebagai sentimen positif. Dalam menentukan sebuah tweet masuk ke dalam kelas positif atau negatif harus di hitung setiap kata yang terdapat di dataset vaksin covid dengan menyamakan dataset vaksin covid dengan dataset lexicon. Dari hasil analisis didapatkan sebanyak 11.750 kata yang terdapat di dataset vaksin covid dan sebanyak 8.575 kata yang tidak terdapat di dataset *lexicon* dan hanya 1.945 kata yang terdapat di *lexicon* dengan total kata di dataset lexicon sebanyak 10.248 atau sekitar 84% tidak terdapat di dataset lexicon dan 16% kata yang terdapat di dataset lexicon . Kata yang terdapat di *lexicon* mempunyai bobot yang nantinya akan di hitung di system untuk menentukan kalimat tersebut masuk ke dalam kelas positif atau negatif dan kata yang tidak terdapat di *lexicon* tidak akan dihitung bobotnya, terdapat contoh perhitungan bobot pada Tabel 4.8

Docs 1 warga yang pernah mengikuti kegiatan berkerumun akan diminta jalani tes vaksin

Docs 2 kemenag usul beli vaksin covid buatan saudi demi kelancaran haji

Docs 3 perbandingan antara vaksin covid

Tabel 4. 8 Contoh Perhitungan Bobot *Lexicon*

Tweet Doc 1	**bobot**	**Tweet Doc 2**	**Bobot**	**Tweet Doc 3**	**bobot**
Warga	Tidak ada	Kemenag	Tidak ada	Perbandingan	3
Yang	-5	Usul	2	Antara	Tidak ada
Pernah	Tidak ada	Beli	2	Vaksin	Tidak ada
Mengikuti	3	Vaksin	Tidak ada	covid	Tidak ada
Kegiatan	3	Covid	Tidak ada		
Berkerumun	-4	Buatan	3		
Akan	Tidak ada	Saudi	Tidak ada		
dimintai	2	Demi	Tidak ada		
Jalani	Tidak ada	Kelancaran	4		
Tes	Tidak ada	Haji	Tidak ada		
vaksin	Tidak ada				
hasil	**-1**		**11**		**3**

Pada Tabel 4.8 dapat dilihat dari contoh 3 *docs* hanya ada beberapa kata saja yang terdapat di *lexicon*, seperti pada tweet docs 1 dari 11 kata hanya terdapat 5 kata yaitu yang, mengikuti, kegiatan, berkerumun, dan minta dengan kata yang masing-masing memiliki bobot yang terdapat di dataset *lexicon.* 5 kata tersebut akan di kalkulasikan jumlahnya dan di dapatkan hasil bobot sebesar -1. Untuk tweet *docs* 2 dari 10 kata hanya terdapat 4 kata di dataset *lexicon* dan di dapatkan hasil bobot sebesar 11. Dan pada tweet doc 3 dari 4 kata hanya terdapat 1 kata dan di dapatkan hasil bobot sebesar 3.

Tabel 4. 9 Hasil Labeling

Tweet	Sentimen	Label
warga yang pernah mengikuti kegiatan berkerumun akan diminta jalani tes vaksin	-1	Negatif
kemenag usul beli vaksin covid buatan saudi demi kelancaran haji	11	Positif
perbandingan antara vaksin covid	3	Positif

Setelah proses pelabelan kelas analisis sentimen, dapat di ketahui jumlah tweet untuk negatif dan positif dengan kode program seperti di Gambar 4.14 dengan menggunakan *value_counts()*

```
data['label'].value_counts()
```

Gambar 4. 14 Kode Program Jumlah Dataset

Berdasarkan kode program tersebut dapat diketahui tweet opini vaksin covid lebih banyak di bandingkan dengan tweet opini vaksin covid negatif. Untuk tweet yang

masuk ke kelas sentimen positif diperoleh sebesar 5.885 tweet dan 2.954 tweet masuk ke kelas sentiment negative.

4.4 *Feature Extraction*

Pada proses *feature extraction* menggunakan TF-IDF. TF-IDF dilakukan oleh system menggunakan library yang disediakan oleh python bernama *TfidfVectorizer().* Dataset tweet yang sudah dilakukan *text preprocessing* dan labeling selanjutnya dilakukan pembobotan kata karena dataset tetap berupa teks (kata), pada analisis klasifikasi data diharuskan terdapat angka. TF-IDF berguna untuk perhitungan pembobotan kata menjadikan dataset yang sebelumnya masih mejadi teks berubah menjadi setiap kata memiliki bobot. Pembobotan TF-IDF dilakukan setelah dataset sudah melalui proses *text preprocessing* pada tahap *stemming*. Sebagai contoh penelitian menggunakan 3 *documents,* yaitu:

Docs 1 ['warga', 'ikut', 'giat', 'kerumun', 'jalan', 'tes', 'vaksin']

Docs 2 ['kemenag', 'usul', 'beli', 'vaksin', 'covid', 'buat', 'saudi', 'lancar', 'haji']

Docs 3 ['banding', 'vaksin', 'covid']

3 *documents* diatas sudah memasuki tahap *stemming*, selanjutnya akan dilakukan perhitungan TF-IDF. TF atau frekuensi term adalah berapa kali term muncul dalam dokumen, sedangkan DF menghitung term dalam dokumen. Proses perhitungan bobot TF dan DF dapat dilihat di Tabel 4.10 di bawah ini

Tabel 4. 10 Proses Perhitungan DF

No	Kata	TF			DF	IDF =log10(3/df)
		***Doc* 1**	***Doc* 2**	***Doc* 3**		
1	Warga	1			1	0.477121
2	Ikut	1			1	0.477121
3	Giat	1			1	0.477121
4	Kerumun	1			1	0.477121
5	Jalan	1			1	0.477121

6	Test	1			1	0.477121
7	Vaksin	1	1	1	3	0
8	Kemenag		1		1	0.477121
9	Usul		1		1	0.477121
10	Beli		1		1	0.477121
12	Covid		1	1	2	0.176091
13	Buat		1		1	0.477121
14	Saudi		1		1	0.477121
15	Lancar		1		1	0.477121
16	Haji		1		1	0.477121
17	Banding			1	1	0.477121

Pada Tabel 4.10 yang pertama dilakukan ada menghitung bobot TF di setiap dokumen. Setelah selesai menghitung TF lalu selanjutnya dilakukan proses DF atau *Documen Frequency* dengan banyaknya term yang muncul di Dokumen (DF). lalu setelah itu hitung IDF (*inverse document frequency*) dengan cara menghitung hasil log D/DF. Dan didapatkan hasil dari DF. Selanjutnya di cari untuk nilai TF dapat dilihat di Tabel 4.11

Tabel 4. 11 Hasil Perhitungan TF

No	**Kata**	**TF**		
		***Doc* 1**	***Doc* 2**	***Doc* 3**
1	Warga	0.477121		
2	Ikut	0.477121		
3	Giat	0.477121		
4	Kerumun	0.477121		
5	Jalan	0.477121		
6	Test	0.477121		
7	Vaksin	0	0	0
8	Kemenag		0.477121	
9	Usul		0.477121	
10	Beli		0.477121	

12	Covid		0.352182	0.352182
13	Buat		0.477121	
14	Saudi		0.477121	
15	Lancar		0.477121	
16	Haji		0.477121	
17	Banding			0.477121

Setelah perhitungan TF dan DF selesai maka bisa dilanjutkan ke perhitungan TF-IDF dengan mengkali TF dengan DF. Perhitungan TF-IDF hanya memakai 3 contoh kata yang memiliki bobot DF 1, 2 , dan 3. Contoh hasil perhitungan dapat dilihat pada Tabel 4.12

Tabel 4. 12 Hasil Perhitungan TF-IDF

Term	**DF**	**TF**	**IDF**	**TF-IDF**
Warga	1	0.477121	0.477121	0.22764
Covid	2	0.352182	0.176091	0.25219
Vaksin	3	0	0	0

4.5 Data latih dan data uji

Pemisahan data dilakukan setelah pembobotan dengan TF-IDF. Kumpulan data harus dibagi menjadi dua, yaitu data latih dan data uji, untuk membuat model dari Naive Bayes. Data *traing* berguna untuk membuat model klasifikasi yang nantinya akan memprediksi kelas sentiment negative atau positif. dan data *testing* yang selanjutnya akan diterapkan untuk memprediksi model. Semakin banyak informasi *training* yang diberikan, semakin baik algoritma naive bayes. Data *testing* berguna untuk melihat presentasi algoritma klasifikasi apakah sudah berhasil melakukan klasifikasi dengan benar. Dalam penelitian ini dibagi menjadi rasio 80:20. Data *train* sebanyak 80% dan data *testing* 20%. Hasil jumlah dari pembagian data dapat dilihat pada Tabel 4.13

Tabel 4. 13 Jumlah Data *Train* dan Data *Testing*

Data	Hasil
Data train	7.071
Data testing	1.768

4.6 Klasifikasi Naïve bayes

Selanjutnya memprediksi klasifikasi data uji. Data latih harus mengklasifikasikan prediksi data uji. Ada kelas sentimen positif dan negatif dalam data *train*, yang memiliki sifat kata di setiap kelas. Sebuah *library* dari bahasa pemrograman Python 3 digunakan untuk mengklasifikasikan proses klasifikasi ini, yaitu *library sci-kit* untuk proses klasifikasi, serta berbagai *library* dan panda sebagai perangkat pembacaan data.

Untuk library scikit-learn dipakai untuk *TfidfVectorizer*, *multinomial NB*, *classification_report*, *accurary_score*, dan *confusion_matrix.* Pada proses *feature extraction* dan klasifikasi langkah pertama adalah menginstall *library* yang diperlukan. Selanjutnya mendeklarasi *library* yang akan digunakan.

Setelah selesai mendeklrasi *library*, proses selanjutnya proses *extraction feature* dengan mengubah dataset menjadi representasi *vector* atau merubah huruf menjadi angka memakai library *TfidfVectorizer*, setelah dilakukan pembobotan pada setiap term di dataset dilanjutkan dengan pembagian dataset, jika model naïve bayes sudah dibuat oleh data *train* maka model di uji dengan data *testing*. Metode klasifikasi data dilaksanakan dengan memakai perhitungan probabilitas per kelas. Terdapat contoh perhitungan dari *naïve bayes* untuk menentukan kelas probabilitas tweet dapat dilihat dibawah ini:

Pada contoh perhitungan ini memakai 2 data training yang sudah diketahui kelasnya dan 1 data testing yang belum di ketahui kelasnya.

Contoh data *training*:

Kelas Positif

dokumen: ['kemenag', 'usul', 'beli', 'vaksin', 'covid', 'buat', 'saudi', 'lancar', 'haji']

Kelas Negatif

dokumen: ['cuma', 'tolak', 'tes', 'vaksin', 'kabur', 'bawa', 'kabur', 'pasien', 'covid', 'denda', 'rp', 'juta']

Contoh Data *Testing* yang belum diketahui kelasnya

dokumen: ['riset', 'vaksin', 'covid', 'lomba', 'taruh', 'gengsi', 'antarnegara', 'presiden', 'rusia']

Tabel 4. 14 Contoh Menghitung *Prior*

menghitung probabilitas setiap kelas	**Prior P(c)**
P (positif)= 1/2	0.50
P (negatif)= 1/2	0.50

Pada tabel 4.14 yang pertama harus dilakukan adalah menghitung nilai *prior* dari setiap kelas yang terdapat di data *train*, pada contoh tabel 4.14 terdapat 1 tweet di kelas positif dan 1 tweet di kelas negatif dan di dapatkan nilai probabilitas masing-masing setiap kelas adalah 0.50. selanjutnya mencari untuk nilai *likelihood* dapat dilihat di Tabel 4.15 dan Tabel 4.16

Tabel 4. 15 Contoh Perhitungan *Likelihood* kelas positif

No	**Term**	**Frekuensi**	**Likelihood P(X|C)**	**Hasil**
1	kemenag	1	P(Kemenag| pos)= 1/9	0.11
2	usul	1	P(usul| pos)= 1/9	0.11
3	beli	1	P(beli| pos)= 1/9	0.11
4	vaksin	1	P(vaksin| pos)= 1/9	0.11
5	covid	1	P(covid| pos)= 1/9	0.11
6	buat	1	P(buat| pos)= 1/9	0.11
7	saudi	1	P(saudi| pos)= 1/9	0.11
8	lancar	1	P(lancar| pos)= 1/9	0.11
9	haji	1	P(haji| pos)= 1/9	0.11
	Total Kata	**9**		

Tabel 4. 16 Contoh Perhitungan *Likelihood* Kelas Negatif

No	Term	Frekuensi	Likelihood P(X|C)	Hasil
1	Cuma	1	P(Cuma|negatif) = 1/12	0.0833
2	tolak	1	P(tolak|negatif) = 1/12	0.0833
3	tes	1	P(tes|negatif) = 1/12	0.0833
4	vaksin	1	P(vaksin|negatif) = 1/12	0.0833
5	kabur	2	P(kabur|negatif) = 2/12	0.1667
6	bawa	1	P(bawa|negatif) = 1/12	0.0833
7	pasien	1	P(Pasien|negatif) = 1/12	0.0833
8	covid	1	P(covid|negatif) = 1/12	0.0833
9	denda	1	P(denda|negatif) = 1/12	0.0833
10	rp	1	P(rp|negatif) = 1/12	0.0833
11	juta	1	P(juta|negatif) = 1/12	0.0833
Total Kata		**9**		

Pada tabel 4.15 dan tabel 4.16 terdapat perhitungan untuk mencari *likelihood* setiap kata pada data *train* di kelas positif dan negatif. Untuk mencari *likelihood* harus dihitung setiap TF (*term frequency*) dari setiap kata dan di dapatkan hasilnya. Setelah itu untuk melakukan perhitungan *evidence* terlebih dahulu dihitung untuk total keseluruhan kata sebanyak 21 term.

Tabel 4. 17 Contoh Perhitungan *Evidance*

No	Term	Term Latih	Frekuensi	P(x) Evidance	
1	riset	tidak ada	0	0/21=	0
2	vaksin	ada	2	2/21 =	0.10
3	covid	ada	2	2/21 =	0.10
4	lomba	tidak ada	0	0/21=	0
5	taruh	tidak ada	0	0/21=	0
6	gengsi	tidak ada	0	0/21=	0
7	antarnegara	tidak ada	0	0/21=	0
8	presiden	tidak ada	0	0/21=	0

9	rusia	tidak ada	0	0/21=	0

Pada tabel 4.17 terdapat perhitungan untuk mencari *evidence* pada data testing. Term data testing akan di samakan dengan dengan term data *training*. Jika ada akan dihitung berapa banyak termnya. Setelah di dapatkan untuk nilai *prior, likelihood* dan *evidence* lalu dihitung menggunakan persamaan *naïve bayes* untuk setiap kelas negatif atau positif

$$P(C|X) = \frac{P(x_1|c)\ldots P(x_n|c)P(C)}{P(x)}$$

$$P(pos|data\ test) = \frac{P(Vaksin|Positif).P(Covid|Positif).P(pos)}{P(vaksin).P(covid)} = \frac{(0.11)(0.11)(0,50)}{(0.10)(0.10)} = 0.00605$$

$$P(neg|data\ test) = \frac{P(Vaksin|Positif).P(Covid|Positif).P(neg)}{P(vaksin).P(covid)} = \frac{(0.0833)(0.0833)(0,50)}{(0.10)(0.10)} = 0.00347$$

Nilai probabilitas pada data testing lebih besar di kelas positif yaitu sebesar 0.00605 dan kelas negatif 0.00347 dan data testing di asumsikan masuk ke kelas positif dikarenakan kelas positif lebih besar.

Untuk mendapatkan evaluasi model uji yang telah dikerjakan oleh data *testing* maka diperlukan metode *confusion matrix* untuk menghitung nilai akurasi dari hasil klasifikasi data. *Confusion matrix* dimanfaatkan untuk menguji hasil prediksi dari metode klasifikasi hasil akurasi dapat dilihat pada Gambar 4.15 di bawah ini

```
    accuracy                           0.73      1768
   macro avg       0.76      0.59      0.58      1768
weighted avg       0.75      0.73      0.67      1768

accuracy score:
0.7313348416289592
```

Gambar 4. 15 Hasil Akurasi

Berdasarkan Gambar 4.15 bisa didapatkan nilai akurasi dari hasil naïve bayes sebesar 0.731 atau 73,1% di mana metode perhitungannya mengacu pada hasil inilai

dari *confusion matrix* yaitu *true negative* ditambah dengan *true negative* lalu di bagi dengan seluruh data. Hasil akurasi mempengaruhi dari hasil splitting data, jika data train jumlahnya banyak maka hasil akurasi akan tinggi.

Tabel 4. 18 Hasil Confusion Matrix

Aktual	**Prediksi**	
	Negatif	**Positif**
Negatif	120	444
Positif	31	1173

Berdasarkan Tabel 4.12, 1173 data positif benar diprediksi masuk ke dalam kelas positif dan sebanyak 31 data positif yang diprediksi masuk kelas negatif serta 120 data negatif yang benar diprediksi masuk ke dalam kelas negatif dan 444 data negatif yang terprediksi masuk ke dalam sentimen positif.

Evaluasi model klasifikasi dengan *confusion matrix* menghitung selain nilai akurasi juga dapat menilai evaluasi model lainnya seperti *precision, recall,* dan *f-measure*

	precision	recall	f1-score
negatif	0.79	0.21	0.34
positif	0.73	0.97	0.83

Gambar 4. 16 Inilai Evaluasi Model

Dapat dilihat di gambar 4.16 Hasil evaluasi model dapat dilihat nilai *precision, recall,* dan *f1-score* pada kelas masing-masing dengan nilai *precision* kelas positif sebesar 79% dan kelas negatif 73%. Selanjutnya untuk *recall* kelas positif sebesar 97% dan kelas negatif 21%. Sedangkan untuk f1-score kelas positif sebesar 83% dan kelas negatif 34%

Nilai *precision* di dapatkan dengan cara nilai yang benar positif di bagi dengan data yang benar positif dijumlahkan dengan data negatif yang di prediksi masuk ke dalam kelas positif. Selanjutnya untuk *recall* sama seperti *precision*, nilai yang benar positif di bagi dengan nilai yang benar positif ditambahkan dengan data positif yang

di prediksi masuk ke dalam kelas negatif. Berikutnya untuk f1-score hasil dua kali inilai *precision* dan *recall* di bagi lagi dengan inilai *precision* dan *recall*

4.7 Visualisasi

Word cloud adalah visualisasi yang memungkinkan untuk melihat kata-kata dalam teks tertulis dengan frekuensi kata. istilah tersebut muncul di *cloud word*, semakin sering kata tersebut muncul dalam data dan maka semakin besar juga kata tersebut muncul di *word cloud.*

Tujuan dari *word cloud* ini adalah untuk mengekstrak informasi dari berbagai tweet yang tersedia, informasi mana yang dianggap relevan, dalam bentuk subjek atau sudut pandang yang sering diungkapkan oleh masyarakat tentang vaksinasi covid. Berikut kata cloud mengenai vaksin covid dilihat pada Gambar 4.16. Kata-kata ini dapat divisualisasikan oleh word cloud.

Gambar 4. 17 *Word Cloud*

Gambar 4.17 Pada terlihat *word cloud* lebih cenderung pada sentimen positif dikarenakan dapat dilihat kata seperti “uji klinis” dengan kemunculan kata sebanyak 1 kali dan terdapat juga kata yang mirip dengan uji klinis yaitu “klinis vaksin” dan “distribusi vaksin” serta “pasti vaksin” dengan kemunculan kata sebanyak 1 kali.

Untuk melihat visualisasi *word cloud* sentimen positif dan negatif dapat dilihat di Gambar 4.18 dan Gambar 4.19

Gambar 4. 18 *Word Cloud* Positif

Pada Gambar 4.18 merupakan data tweet yang masuk ke kelas positif menggunakan analisis sentimen *lexicon-based*. Dapat terlihat vaksin covid sering dibicarakan serta terdapat kata “uji klinis”, “sedia vaksin”, dan “gratis vaksin” dengan kemunculan masing-masing kata adalah 1 kali tetapi untuk kemunculan kata “vaksin” sendiri sebanyak 24 kali.

Gambar 4. 19 *Word Cloud* Negatif

Pada Gambar 4.19 merupakan data tweet yang masuk ke kelas negatif menggunakan analisis sentimen *lexicon based.* Dapat terlihat pada *word cloud* negatif perbedaannya tidak terlalu siginifikan "vaksin covid" masih menjadi topik yang sering dibicarakan, tetapi pada *word cloud* negatif terdapat juga kata seperti "tular covid", "takut", dan "efek samping" dengan kemunculan kata sebanyak 1 kali sedangkan untuk kata "vaksin" sendiri sebanyak 15 kali.