BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Hasil penelitian analisis sentimen dengan kata kunci "Telkomsel" menggunakan metode Naive Bayes Classification untuk menganalisis data dari *tweet* berbahasa Indonesia baik positif, negatif maupun netral. Data Twitter yang diambil selama periode 20 April-30 Juli 2022 dengan jumlah 13,239 data *tweet* dan *retweet.* Data yang digunakan selama *training* sebanyak 600 data dengan 200 data berlabel positif, 200 data berlabel negatif dan 200 data berlabel netral. Pada proses *testing* menggunakan sebanyak 336 data dengan 112 data berlabel positif, 112 data berlabel negatif dan 112 data berlabel netral. Analisis sentimen media sosial Twitter dibangun untuk mengolah data *tweet* menggunakan bahasa pemrograman Python dengan metode Naive Bayes Classification untuk mendapatkan hasil perhitungan sentimen menggunakan *framework* Flask. Penelitian ini akan memberikan informasi tentang analisis sentimen pengguna Telkomsel di media sosial Twitter tentang Telkomsel. .

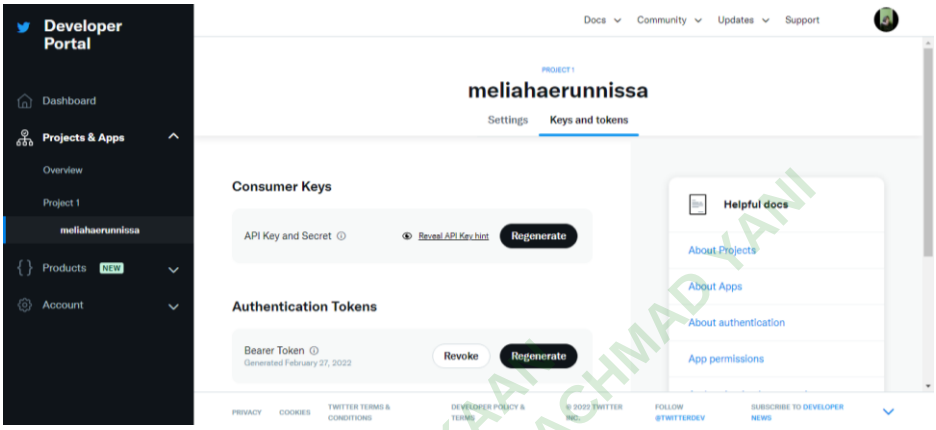
4.2 PENGAMBILAN DATA TWITTER

Pengambilan data dilakukan di media sosial Twitter untuk mendapatkan data *tweet* yang berkaitan dengan kata kunci "Telkomsel". Data *tweet* yang diambil mulai dari tanggal 20 Mei – 30 Juli 2022 dengan 13.239 data *tweet, re-tweet* dan komentar. Proses pengambilan data *tweet* menggunakan API Twitter dilakukan di Anaconda Prompt dan dieksekusi di Jupyter Notebook. Data *tweet* yang dapat diambil berformat file *Comma Separated Values* (CSV) sehingga dapat dibuka di Microsoft Excel.

4.2.1 Autentifikasi API Twitter

Autentikasi API Twitter adalah kerangka kerja untuk secara aman dan ringkas mencapai akses terbatas dengan mengisi data pada *developer portal*

Twitter. *Access token* digunakan untuk mengambil data *tweet* sesuai kebutuhan yang ditetapkan oleh Twitter, setelah mendapatkan *access token*, *access token secret*, *consumer key*, *consumer secret* API berguna untuk mendapatkan data *tweet* untuk pemrosesan data. Tampilan autentifikasi API Twitter ditunjukkan pada Gambar 3.8.



Gambar 3.8 Tampilan Autentifikasi API Twitter

4.2.2 Pengumpulan Data Twitter

Pengumpulan data Twitter dilakukan dengan bahasa pemrograman Python. Penelitian ini menggunakan *library tweepy* dengan melakukan akses pada API Twitter untuk mendapatkan data *tweet* berdasarkan kata kunci "Telkomsel". Tampilan import *library* ditunjukkan pada Gambar 3.9.

```
1 import tweepy
2 import csv
3 import sys
4 import re
5 import pandas as pd
6 import numpy as np
7 import string
8 import datetime
9 import time
```

Gambar 3.9 Library Pengambilan Data Twitter

Library Python yang digunakan untuk proses pengumpulan data *tweet* kemudian disimpan sebagai file *Comma Separated Values* (CSV). Dalam pengambilan data *tweet* menggunakan API Twitter, data *tweet* yang dihasilkan

memiliki variasi dalam berbagai bentuk, termasuk *hashtag*, tautan URL, angka, username sehingga diperlukan proses *preprocessing* untuk mendapatkan data *tweet* yang diinginkan. *Pandas* memiliki manipulasi dengan pembersihan data dan fungsi *datetime* menjalankan operasi terkait waktu. Contoh data *tweet* yang sudah diambil ditunjukkan pada Tabel 3.1.

Tabel 3.1 Data Tweet

No	Data Tweet
1	b'@scorpeeoss aku pk Telkomsel 25-29k perbulan udh sama Disney+'
2	b'TELKOMSEL MAHAL BGT ANJIR KUOTANYA NANGISSSSSA https://t.co/3FSOI9HPiu'
3	b'Telkomsel knp siii'
4	b'woi @Telkomsel kuota msh ada 10 gb knp lemot yh ini hp w yg lemot ap jaringan nya'
5	b'min @Telkomsel tolong dong ini udah hampir sebulanan gini kok jaringan ku lemot terus ya :('
6	b'@Meitryparamita1 Halo Kak Mei. Terkait paket internet infoin nomor Hp nya yuk Kak melalui DM agar Sulis cek dan\xe2\x80\xa6 https://t.co/n4aHNoJE1J'
7	b'@Telkomsel Nah kan bener dilempar lempar'
8	b'Semoga besok sinyal telkomsel gak bermasalah :\xe2\x80\x99)'
9	b'Ada apa sih telkomsel dan indihome hr ini??? Kok jelek bgt'
10	b'@Felixie04 coba lgsg dtg ke gerai telkomsel aja kayanya bs kalau mau di aktivasi lg'

Tabel 3.1 menunjukan hasil pengambilan data *tweet* yang belum terstruktur dan memiliki variasi bentuk, sehingga diperlukan beberapa *preprocessing* karena masih banyak variasi bentuk yang perlu dihilangkan karena tidak digunakan dalam pembuatan analisis sentimen agar menjadi data yang terstruktur.

4.2.3 *Preprocessing*

Preprocessing dilakukan untuk proses pengolahan data yang berfungsi memperbaiki data *tweet* yang belum terstruktur dengan melakukan tahapan-

tahapan agar menjadi data yang tersetruktur. Contoh data *tweet* yang sudah dilakukan proses *preprocessing* ditunjukkan pada Tabel 3.2.

Tabel 3.2 Data *Tweet* Hasil *Preprocessing*

No	Data Tweet
1	my telkomsel ganggu kah
2	kenapa tidak masuk langgan disney lagi
3	app my telkomsel error mau panjang paket gimana ini
4	siap kak putra kami akan cek dm dari kakak tunggu konfirmasi lanjut di dm ya kak makasih gundi
5	memang lagi ganggu apa gimana sih kok tiba tiba sinyal hilang
6	baik kak jika belum kakak sudah pernah kirim data yang minta belum yuk siapa kami di dm
7	aku dapat bulan khusus disney di telkomsel
8	mati lampu mati juga lah telkomsel
9	sih telkomsel mahal banget kouta nya boros pula
10	maaf banget ya kak jeje jadi tidak nyaman kena keluh tidak bisa akses apps mytelkomsel yuk infoin nomor

Tabel 3.2 menunjukan hasil data *tweet* yang sudah dilakukan proses *preprocessing*. Data *tweet* tersebut menjadi lebih terstruktur agar dapat digunakan untuk melakukan perhitungan ditahap selanjutnya. Berikut merupakan tahapan-tahapan dalam proses *preprocessing* data:

a. Case Folding

Case Folding merupakan tahapan yang dilakukan untuk mengubah setiap huruf pada data *tweet* menjadi *lowercase* atau huruf kecil.

b. Number Removal

Number removal merupakan tahapan yang dilakukan untuk membersihkan data dari karakter angka.

c. Punctuation Removal

Punctuation removal merupakan tahapan yang dilakukan untuk penghapusan karakter pada data teks.

d. Whitespaces Removal

Whitespaces removal merupakan langkah yang dilakukan untuk menghilangkan spasi diawal dan akhir kalimat.

e. Tokenize

Tokenizing merupakan tahapan yang dilakukan untuk pemecahan atau pemisahan kalimat dalam suatu teks yang disebut token.

f. Stopword Removal

Stopword removal merupakan tahapan yang dilakukan untuk penghapusan kata yang memiliki informasi rendah pada teks.

g. Normalisasi

Normalization merupakan tahapan yang dilakukan membakukan kata-kata dengan salah eja atau penggunaan bahasa yang tidak biasa. Contoh normalisasi dapat dilihat pada Tabel 3.3.

Tabel 3.3 Data Normalisasi

Before	After
dtg	datang
lgsg	langsung
pk	pakai
jgn	jangan
cmn	hanya
tuku	beli
kmren	kemarin
gbs	tidak bisa
dipake	dipakai

h. Stemming

Stemming merupakan tahapan yang dilakukan untuk penghilangan infleksi pada kata menjadi bentuk dasarnya.

4.2.4 Pelabelan manual

Pelabelan manual merupakan proses memberikan label positif, negatif dan netral terhadap data *tweet* yang sudah dilakukan proses *preprocessing*. Proses pelabelan manual dilakukan di Microsoft Excel secara manual agar data *tweet* dapat dianalisis. Data *tweet* yang sudah diberi label didapatkan 600 data *tweet* yang digunakan untuk *training* dengan data berlabel positif 200 data *tweet*, berlabel netral 200 data *tweet* dan berlabel negatif 200 data *tweet*. Contoh data *tweet* yang sudah dilakukan pelabelan manual ditunjukkan pada Tabel 3.4.

Tabel 3.4 Tabel Pelabelan Manual

No	Tweet	Kelas	Label
1	ngebut banget asli si paling ngebut sudah pokok	Positif	1
2	ada cashback nampol sampai dari mytelkomsel yang paling bisa bagi bagi promo ntransaksi minimal rp ribu di mytelkomsel	Positif	1
3	hai kak maaf ya untuk keluh akses internet lambat yuk infoin nomor handphone tanggal jadi loka	Netral	0
4	tenang kak gilang kalau yang kakak maksud kendala akses aplikasi mytelkomsel yuk infoin nomor handphone tanggal	Netral	0
5	jelek banget kalau mati lampu sinyal ikut ngilang	Negatif	-1
6	telkomsel memang asa boros banget apa gimana sih pakai bentar doang sudah habis banyak	Negatif	-1
7	hai kak maaf ya jadi tidak nyaman inara akan bantu kendala kakak yuk dm nomor handphone kakak lokasi detail	Netral	0
8	harga naik disney hotstar yang murah banget kalau pakai telkomsel cuma rb bulan	Positif	1
9	telkomsel legal gift dari mytelkomsel proses cepat langgan disney bulan	Positif	1

No	Tweet	Kelas	Label
10	usaha donk buat telkomsel jangan kartu baru mati yang lama juga bisa di bantu agar lebih byk yang guna kayu telkomsel dan puas	Negatif	-1

Tabel 3.4 menunjukan hasil pelabelan manual bahwa data *tweet* dengan kelas positif diberi label 1, kelas negatif diberi label -1, dan kelas netral diberi label 0. Data pelabelan ini digunakan untuk memberikan nilai sentimen yang dihitung akurasinya.

4.2.5 *Training*

Training data merupakan proses yang dilakukan untuk menghasilkan model klasifikasi yang secara otomatis digunakan untuk proses klasifikasi, proses *training* menggunakan metode Naive Bayes Classification dengan fungsi ekstraksi TF-IDF pada data teks. Contoh perhitungan TF-IDF secara manual ditunjukkan pada Tabel 3.5.

Tabel 3.5 Data *Tweet* TF-IDF

Dokumen (d)	Tweet
d1	harga naik disney hotstar yang murah banget kalau pakai telkomsel cuma rb bulan
d2	maaf banget ya kak sane jadi tidak nyaman kena keluh tidak bisa akses apps mytelkomsel yuk infoin nomor handphone
d3	jaring sangat buruk sangat lambat
d4	coba langsung datang ke gerai telkomsel saja kaya bisa kalau mau di aktivasi lg
d5	parah banget memang sekarang enggak kalau jaring tidak dimaksimalin harga paket jangan mahal dong

Tabel 3.5 menjelaskan data *tweet* akan digunakan sebagai contoh perhitungan TF-IDF, perhitungan TF-IDF dilakukan secara manual menggunakan 5 data *tweet* yaitu d1, d2, d3, d4, d5 dan beberapa komponen seperti t yaitu kata, d yaitu kalimat atau data *tweet* untuk mengetahui berapa banyak data *tweet* dimana suatu kata muncul. Perhitungan TF-IDF ditunjukkan pada Tabel 3.6.

Tabel 3.6 Perhitungan *Term Frequency* (TF)

term/kata	d1	d2	d3	d4	d5	df
akses		1				1
apps		1				1
banget	1	1			1	3
bisa		1		1		2
bulan	1					1
buruk			1			1
coba				1		1
cuma	1					1
datang				1		1
diaktivasi				1		1
dimaksimalkan					1	1
disney	1					1
dong					1	1
gerai				1		1
handphone		1				1
harga	1				1	2
hotstar	1					1
infoin		1				1
jangan					1	1
jaring			1		1	2
kak	1					1
kalau	1			1	1	3
kaya				1		1
ke				1		1
keluh		1				1
kena		1				1
lagi				1		1
lambat			1			1
langsung				1		1

term/kata	d1	d2	d3	d4	d5	df
maaf	1					1
mahal					1	1
mau				1		1
memang					1	1
murah	1					1
mytelkomsel		1				1
naik	1					1
nomor		1				1
nyaman		1				1
pakai	1					1
paket					1	1
parah					1	1
ribu	1					1
saja				1		1
sane	1					1
sangat			2			1
sekarang					1	1
telkomsel	1			1		2
tidak		2			2	2
ya	1					1
yang	1					1
yuk		1				1

Tabel 3.6 menjelaskan perhitungan *term frequency* (TF) sehingga setiap kata dari kalimat yang terdapat dalam dokumen akan dilakukan perhitungan *invers document frequency* (IDF). Proses perhitungan tersebut menghitung sebuah kata yang terdapat pada dokumen dengan N banyaknya dokumen menggunakan komponen *term* atau kata, *document frequency* (DF) dan *invers document frequency* (IDF). Perhitungan IDF ditunjukkan pada Tabel 3.7.

Tabel 3.7 Perhitungan *Invers Document Frequency* (IDF)

term/kata	Df	Idf	idf(N=4)	idf(N=1000
akses	1	1	0,602059991	3
apps	1	1	0,602059991	3
banget	3	0,333333333	0,124938737	2,522878745
bisa	2	0,5	0,301029996	2,698970004
bulan	1	1	0,602059991	3
buruk	1	1	0,602059991	3
coba	1	1	0,602059991	3
cuma	1	1	0,602059991	3
datang	1	1	0,602059991	3
diaktivasi	1	1	0,602059991	3
dimaksimalkan	1	1	0,602059991	3
disney	1	1	0,602059991	3
dong	1	1	0,602059991	3
gerai	1	1	0,602059991	3
handphone	1	1	0,602059991	3
harga	2	0,5	0,301029996	2,698970004
hotstar	1	1	0,602059991	3
infoin	1	1	0,602059991	3
jangan	1	1	0,602059991	3
jaring	2	0,5	0,301029996	2,698970004
kak	1	1	0,602059991	3
kalau	3	0,333333333	0,124938737	2,522878745
kaya	1	1	0,602059991	3
ke	1	1	0,602059991	3
keluh	1	1	0,602059991	3
kena	1	1	0,602059991	3
lagi	1	1	0,602059991	3
lambat	1	1	0,602059991	3
langsung	1	1	0,602059991	3

term/kata	Df	Idf	idf(N=4)	idf(N=1000
maaf	1	1	0,602059991	3
mahal	1	1	0,602059991	3
mau	1	1	0,602059991	3
memang	1	1	0,602059991	3
murah	1	1	0,602059991	3
mytelkomsel	1	1	0,602059991	3
naik	1	1	0,602059991	3
nomor	1	1	0,602059991	3
nyaman	1	1	0,602059991	3
pakai	1	1	0,602059991	3
paket	1	1	0,602059991	3
parah	1	1	0,602059991	3
ribu	1	1	0,602059991	3
saja	1	1	0,602059991	3
sane	1	1	0,602059991	3
sangat	1	1	0,602059991	3
sekarang	1	1	0,602059991	3
telkomsel	2	0,5	0,301029996	2,698970004
tidak	2	0,5	0,301029996	2,698970004
ya	1	1	0,602059991	3
yang	1	1	0,602059991	3
yuk	1	1	0,602059991	3

Tabel 3.7 menjelaskan perhitungan *invers document frequency* (IDF) dengan rumus $idf = log\frac{(N)}{(df)}$ untuk mengurangi bobot setiap *term* atau kata dalam kalimat yang terdapat dalam dokumen untuk mengetahui banyaknya *term* atau kata yang muncul. Perhitungan TF-IDF ditunjukkan pada Tabel 3.8.

Tabel 3.8 Perhitungan (TF-IDF)

term/kata	d1	d2	d3	d4	d5
akses		0,60206			
apps		0,60206			
Banget	0,124939	0,124939			0,124939
Bisa		0,30103		0,30103	
Bulan	0,60206				
Buruk			0,60206		
Coba				0,60206	
Cuma	0,60206				
Datang				0,60206	
Diaktivasi				0,60206	
Dimaksimalkan					0,60206
Disney	0,60206				
Dong					0,60206
Gerai				0,60206	
Handphone		0,60206			
Harga	0,30103				0,30103
Hotstar	0,60206				
Infoin		0,60206			
Jangan					0,60206
Jaring			0,30103		0,30103
Kak	0,60206				
Kalau	0,124939			0,124939	0,124939
Kaya				0,60206	
Ke				0,60206	
Keluh		0,60206			
Kena		0,60206			
Lagi				0,60206	
Lambat			0,60206		
Langsung				0,60206	

term/kata	d1	d2	d3	d4	d5
Maaf	0,60206				
Mahal					0,60206
Mau				0,60206	
Memang					0,60206
Murah	0,60206				
Mytelkomsel		0,60206			
Naik	0,60206				
Nomor		0,60206			
Nyaman		0,60206			
Pakai	0,60206				
Paket					0,60206
Parah					0,60206
Ribu	0,60206				
Saja				0,60206	
Sane	0,60206				
Sangat			0,60206		
Sekarang					0,60206
Telkomsel	0,30103			0,30103	
Tidak		0,30103			0,30103
Ya	0,60206				
Yang	0,60206				
Yuk		0,60206			

Tabel 3.8 menjelaskan perhitungan *term frequency-invers document frequency* (TF-IDF) dengan perhitungan secara manual.

4.2.6 *Testing*

Testing merupakan proses yang digunakan untuk memprediksi kelas, label berdasarkan data *training* yang sudah dibangun untuk mengetahui tingkat keakuratan pemodelan.

4.2.7 Hasil Model Klasifikasi

Melakukan evaluasi model klasifikasi data *training* menggunakan perhitungan *confusion matrix* untuk menemukan nilai aktual dan nilai prediksi berdasarkan akurasi yang diberikan oleh aplikasi. Penelitian ini menggunakan *multiclass confusion matrix* karena memiliki 3 kelas sentimen, yaitu positif, negatif dan netral. Data *training* yang akan dihitung adalah 600 data *tweet* yang diberi label secara manual sebagai positif, negatif dan netral. *Multiclass confusion matrix* memiliki istilah yang mewakili hasil dari proses klasifikasi diantaranya *true positive* (TPost), *true negative* (TNeg), *true netral* (TNet), *false positive* (FPost) dan *false negative* (FNeg), *false netral* (FNet). Hasil perhitungan *multiclass confusion matrix* pada data *training* ditunjukkan pada Tabel 3.9.

Tabel 3.9 Hasil Confusion Matrix Data *Training*

Kelas Aktual	Kelas Prediksi		
	Positif	Negatif	Netral
Positif	30	4	7
Negatif	1	37	5
Netral	2	3	31

Tabel 3.9 menjelaskan bahwa hasil perhitungan menggunakan *confusion matrix* didapatkan nilai berupa TPos = 30, FPostNeg = 4, FPostNet = 7, FNegPost = 1, TNeg = 37, FNegNet = 5 dan FNetPos = 2, FNegNet = 3, TNet = 31 maka dilanjutkan perhitungan klasifikasi. Hasil klasifikasi data *training* dapat dilihat pada Tabel 3.10.

Tabel 3.10 Hasil Klasifikasi Data *Training*

Accuracy	0,82
F1-Score	0,82
Precission	0,83
Recall	0,82

4.2.8 Hasil Evaluasi Klasifikasi

Hasil evaluasi klasifikasi untuk data *testing* dilakukan dengan menggunakan perhitungan *confusion matrix* untuk menghitung nilai akurasi untuk menentukan perbedaan antara data *training* dan *testing*. Data *testing* akan dihitung sebanyak 336 data *tweet* yang telah diberi kelas positif, negatif dan netral masing-masing 112 label data *tweet*. Hasil perhitungan klasifikasi pada data *testing* ditunjukkan pada Tabel 3.11.

Tabel 3.11 Hasil Klasifikasi Data *Testing*

Accuracy	0,84
F1-Score	0,84
Precission	0,84
Recall	0,83

Tabel 3.11 menunjukan hasil perhitungan klasifikasi memiliki akurasi yang cukup baik yaitu 0,84, selama pengujian menggunakan *confusion matrix* untuk mengetahui nilai aktual dan prediksi dalam data *testing*. Perhitungan *confusion matrix* pada data *training* dapat dilihat pada Tabel 3.12.

Tabel 3.12 Hasil *Confusion Matrix* Data *Testing*

Kelas Aktual	Kelas Prediksi		
	Positif	Negatif	Netral
Positif	19	3	1
Negatif	1	22	3
Netral	3	0	16

Tabel 3.12 bahwa hasil perhitungan menggunakan *confusion matrix* didapatkan nilai berupa TPos = 19, FPostNeg = 3, FPostNet = 1, FNegPost = 1, TNeg = 22, FNegNet = 3 dan FNetPost = 3, FNetNeg = 0, TNet = 16, kemudian

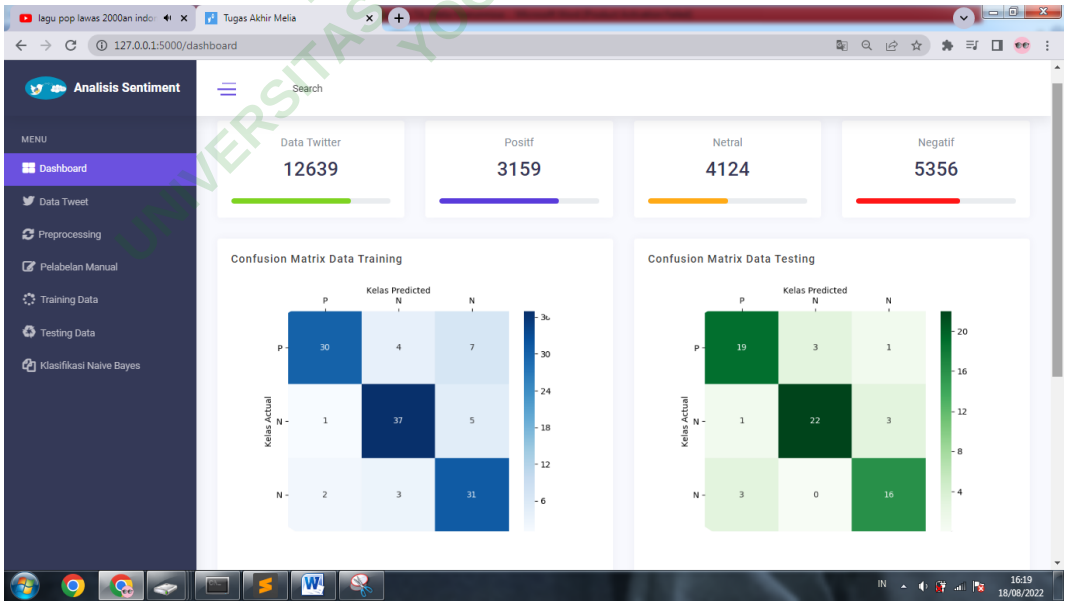
hasil perbandingan data *training* sebanyak 600 data *tweet* dengan data *testing* sebanyak 336 data *tweet* memiliki pengujian yang hampir sama.

4.3 IMPLEMENTASI DISAIN INTERFACE

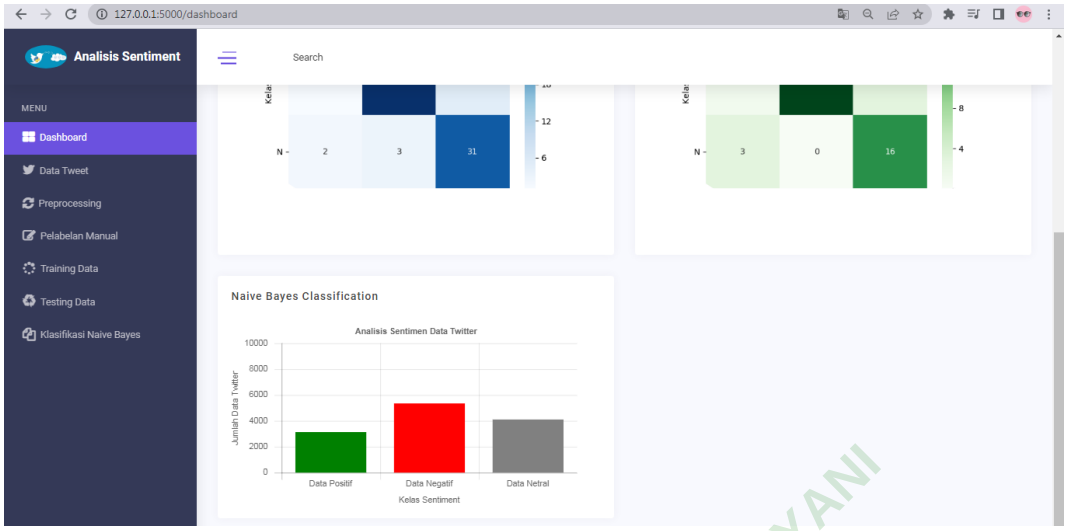
Implementasi *desain interface* merupakan implementasi desain tampilan sistem. Setiap desain yang telah dilakukan sebelumnya di realisasikan menggunakan bahasa pemrograman Python. Dalam Analisis sentimen kepuasan pelanggan perusahaan telekomunikasi seluler Telkomsel di Twitter ini yang digunakan adalah Python dengan memanfaatkan *framework* Flask. Dibawah ini adalah beberapa halaman dan contoh kode program yang termasuk dalam sistem analisis sentimen kepuasan pelanggan perusahaan telekomunikasi seluler Telkomsel di Twitter.

4.3.1 Halaman Dashboard Analisis Sentimen

Halaman dashboard memuat beberapa informasi berdasarkan hasil dari perhitungan *confusion matrix* data *training*, *testing* dan hasil data *tweet* yang didapat seperti banyaknya data Twitter berdasarkan komentar positif, netral, dan negatif. Implementasi halaman dashboard ditunjukkan pada Gambar 4.1.



Gambar 4.1 Implementasi Halaman Dashboard Confusion Matrix



Gambar 4.2 Implementasi Halaman Dashboard Grafik

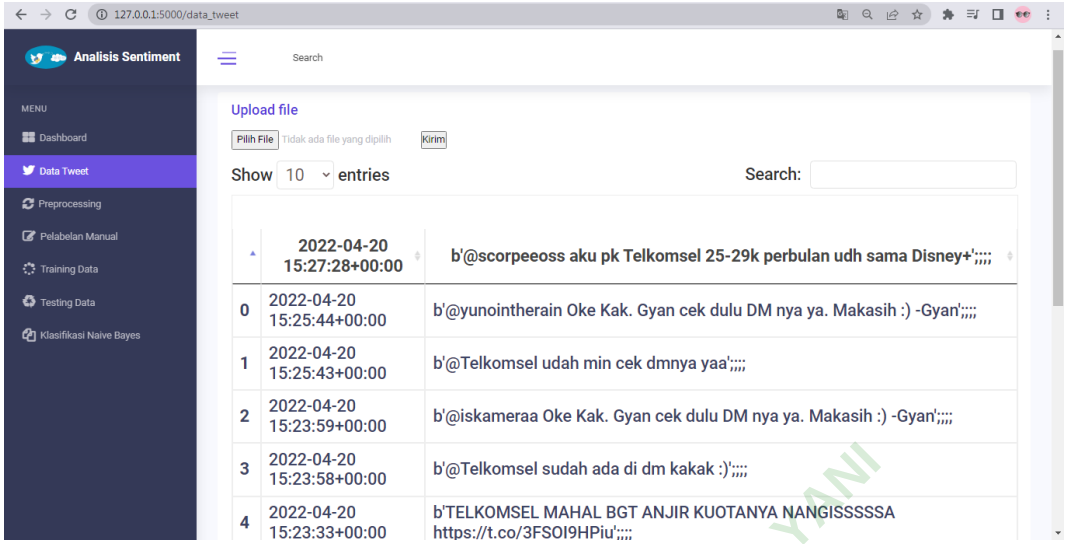
Berikut ini potongan kode pada *controller* yang berfungsi untuk menampilkan informasi dalam bentuk grafik.

```
#<------------ DASHBOARD ------------>
@application.route('/dashboard')
def dashboard():
    data_testing = pd.read_excel('TA-
Melia/static/uploads/hasil_data_keseluruhan.xlsx')
    data_testing.dropna()

    positif = data_testing.kelas.value_counts().Positif
    netral = data_testing.kelas.value_counts().Netral
    negatif = data_testing.kelas.value_counts().Negatif
    total = positif + netral + negatif

    return render_template('dashboard.html', total=total,
positif=positif, netral=netral, negatif=negatif)
```

Fungsi diatas digunakan untuk menampilkan hasil dari data *tweet* menggunakan `value_counts()` untuk mengetahui total dari data *tweet* dan menampilkan total sentiment positif, netral, dan negatif dengan membuat variabel.

4.3.2 Halaman Data Twitter

Halaman data Twitter digunakan untuk melakukan upload file dan menampilkan file CSV yang sudah terupload ke dalam directory agar memudahkan proses *preprocessing*, *training*, *testing* dan *klasifikasi*. Implementasi halaman data Twitter ditunjukkan pada Gambar 4.3.



Gambar 4.3 Implementasi Halaman Data Tweet

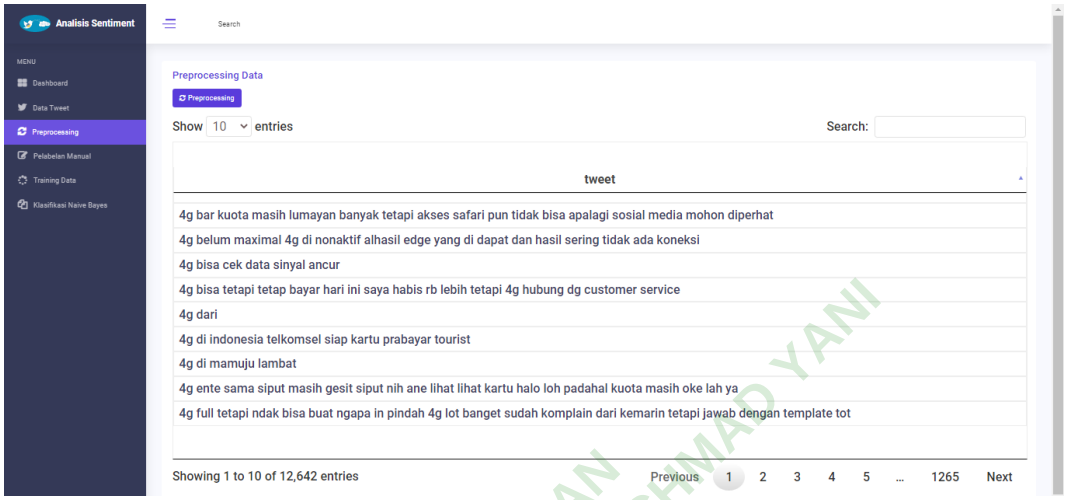
Berikut ini potongan kode pada *controller* yang berfungsi untuk melakukan upload file.

```
#<------------ UPLOAD FILE ------------>
@application.route('/data_tweet', methods=['GET', 'POST'])
def data_tweet():
    data = []
    data = pd.DataFrame()
    if request.method == "POST":
        uploaded_file = request.files['filename']
        file = os.path.join(application.config['UPLOAD_FOLDER'],
uploaded_file.filename)
        uploaded_file.save(file)
        try:
            data = pd.read_excel(file)
            data.drop(['Unnamed: 1','Unnamed: 2','Unnamed: 3',
'Unnamed: 4', 'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12'], axis=1, inplace=True)
            data.append(data)
        except:
            data = pd.read_csv(file,  on_bad_lines='skip')
            data.dropna()
            data.append(data)

    return render_template('data_tweet.html',
data=[data.to_html(justify='center', classes=['table-striped',
'table-bordered', 'dt-responsive', 'table-style'],
table_id='example')])
```

4.3.3 Halaman *Preprocessing* Data

Halaman *preprocessing* berfungsi untuk menampilkan dan melakukan proses pengolahan data teks yang sudah tersedia. *Preprocessing* berfungsi untuk

memperbaiki data *tweet* dengan melakukan tahapan-tahapan agar menjadi data yang tersetruktur. Implementasi halaman *preprocessing* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Implementasi Halaman *Preprocessing* Data

Berikut ini potongan kode pada *controller* yang berfungsi untuk melakukan *preprocessing* data.

```
#<------------ PREPROCESSING DATA ------------->
@application.route('/preprocessing', methods=['GET', 'POST'])
def preprocessing():
    if request.method == 'POST':
        data = pd.read_csv('TA-
Melia/static/uploads/hasil_15April.csv', names=['tanggal',
'tweet'], encoding='latin-1')
        data.dropna()
        data.drop(['tanggal'], axis=1, inplace=True)

        data['tweet'] = data['tweet'].map(lambda x: lower(x))
        data['tweet'] = data['tweet'].map(lambda x:
remove_punctuation(x))
        data['tweet'] = data['tweet'].map(lambda x:
normalized_term(x))
        data['tweet'] = data['tweet'].map(lambda x:
remove_stopwords(x))
        data['tweet'] = data['tweet'].map(lambda x: stem_text(x))
        data['tweet'] = data['tweet'].map(lambda x: tokenize(x))

        data = data.drop_duplicates(subset=['tweet'], keep=False)
        data.to_excel('TA-
Melia/static/uploads/hasil_preprocessing.xlsx')

    data_preprocessing = pd.read_excel('TA-
Melia/static/uploads/hasil_preprocessing.xlsx')
    data_preprocessing.dropna()
```

```
    data_preprocessing.drop(['Unnamed: 0'], axis=1, inplace=True)

    return render_template('preprocessing.html',
data_preprocessing=[data_preprocessing.to_html(index=False,
justify='center', classes=['table-striped', 'table-bordered', 'dt-
responsive', 'table-style'], table_id='example')])
```

Fungsi `.map` diatas digunakan untuk mengaplikasikan suatu fungsi pada semua anggota array yang terdapat pada variabel data *tweet*, `lambda` digunakan untuk melakukan operasi tanpa harus mendefinisikan fungsinya, memiliki beberapa argumen 1 ekspresi.

4.3.4 Halaman Pelabelan Manual

Halaman pelabelan manual digunakan untuk menampilkan hasil *Accuracy*, *Precission*, *Recall* dan *F1-score* berdasarkan data *tweet* yang sudah diberi kelas, label terhadap kalimat yang ada pada data *tweet* secara manual. Proses pelabelan manual dilakukan menggunakan Microsoft Excel. Implementasi halaman pelabelan manual ditunjukkan pada Gambar 4.5.

Gambar 4.5 Implementasi Halaman Pelabelan Manual

Berikut ini potongan kode pada *controller* yang berfungsi untuk menampilkan data *tweet* yang sudah dilakukan pelabelan.

```
#<------------ PELABELAN MANUAL ------------->
@application.route('/pelabelan_manual', methods=['GET', 'POST'])
def pelabelan_manual():
```

```
    data_pelabelan = pd.read_excel('TA-
Melia/static/uploads/data_training.xlsx')
    data_pelabelan.dropna()
    data_pelabelan.drop(['Unnamed: 1', 'Unnamed: 2','Unnamed: 3',
'Unnamed: 4', 'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12'], axis=1, inplace=True)
    data_pelabelan.drop_duplicates(['tweet'])
```

Fungsi diatas berfungsi untuk menampilkan data *tweet* yang sudah dilakukan pelabelan secara manual.

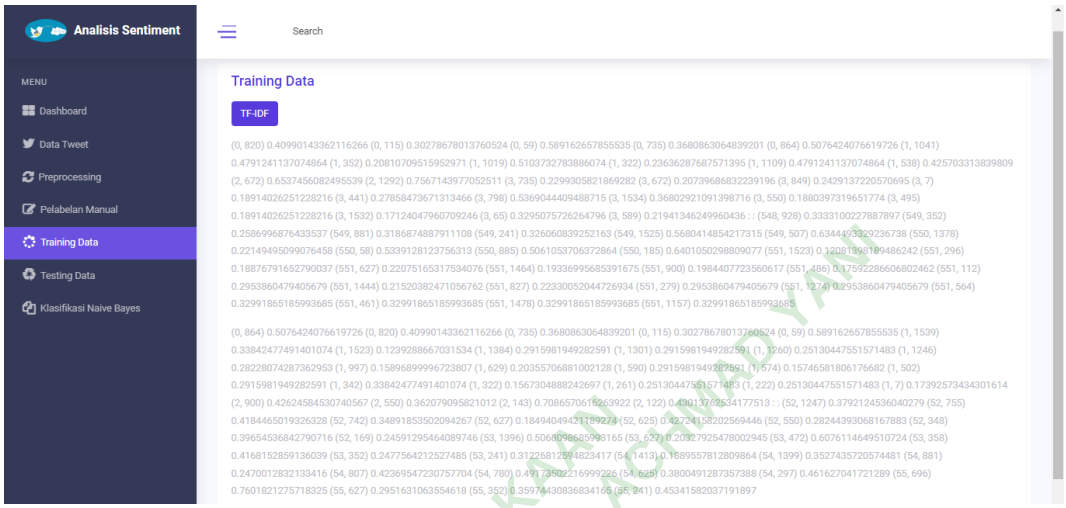
Berikut ini potongan kode pada *controller* yang berfungsi untuk melakukan perhitungan akurasi pada data *tweet* yang sudah dilakukan pelabelan secara manual.

```
#<------------ AKURASI TRAINING ------------->
    X = data_pelabelan['tweet']
    Y = data_pelabelan['label']
    X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.2,random_state=42)

    vect = TfidfVectorizer(analyzer =
"word",min_df=0.0004,max_df=0.115, ngram_range=(1,3))
    vect.fit(X_train)
    X_train_dtm = vect.transform(X_train)
    X_test_dtm = vect.transform(X_test)

    nbmodel = MultinomialNB(alpha=0.1)
    nbmodel = nbmodel.fit(X_train_dtm,Y_train)
    Y_pred = nbmodel.predict(X_test_dtm)

    akurasi = accuracy_score(Y_test, Y_pred) * 100
    f1score = f1_score(Y_test, Y_pred, average='weighted') * 100
    presision = precision_score(Y_test, Y_pred,
average='weighted') * 100
    recall = recall_score(Y_test, Y_pred, average='weighted') *
100

    return render_template('pelabelan_manual.html',
data_pelabelan=[data_pelabelan.to_html(index=False,
justify='center', classes=['table-striped', 'table-bordered',
'table-hover', 'table-condensed', 'dt-responsive', 'table-style'],
table_id='example')], accuracy_score=akurasi, f1_score=f1score,
precision_score=presision, recall_score=recall)
```

4.3.5 Halaman *Training* Data

Halaman *training* data digunakan untuk membuat model klasifikasi yang digunakan untuk mengklasifikasi sentimen secara otomatis. Langkah-langkah

dalam *training* data menggunakan metode Naive Bayes Classification dan *library* Multinomial Naive Bayes dengan fitur ekstraksi menggunakan *term frequency-invers document frequency* (TF-IDF) pada data teks. Implementasi halaman *training* data ditunjukkan pada Gambar 4.6.



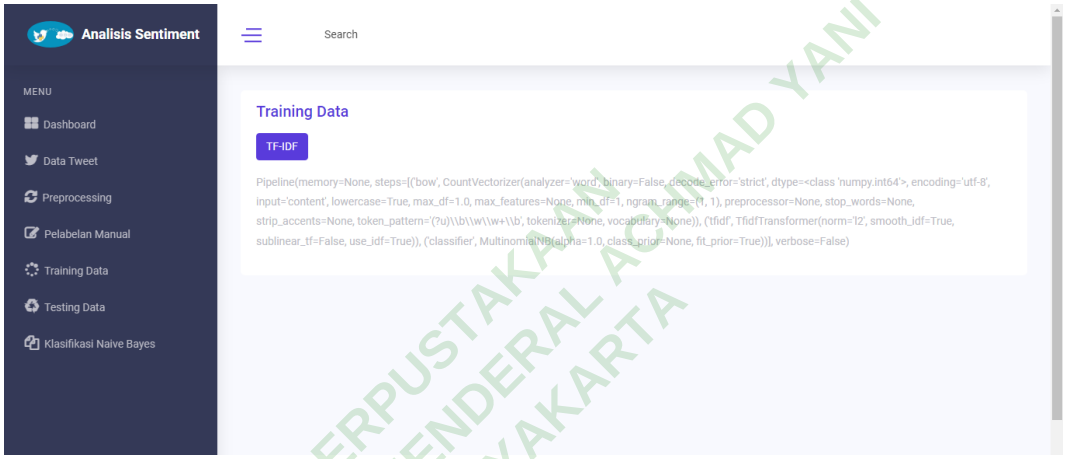
Gambar 4.6 Implementasi Halaman Training Data

Berikut ini potongan kode pada *controller* yang berfungsi untuk melakukan perhitungan secara otomatis dengan pembobotan pada kata yang ada pada data *training*.

```
#<------------ TRAINING DATA ------------->
@application.route('/training', methods=['GET', 'POST'])
def training():
    data_testing = pd.read_excel('TA-
Melia/static/uploads/data_training.xlsx')
    data_testing.dropna()
    data_testing.drop(['Unnamed: 3', 'Unnamed: 4', 'Unnamed: 5',
'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12'], axis=1, inplace=True)

    X = data_testing['tweet']
    y = data_testing['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=13)

    vectorizer = TfidfVectorizer()

    X_train = vectorizer.fit_transform(data_testing['tweet'])
    X_test = vectorizer.transform(X_test)
```

```
    return render_template('testing_tfidf.html', y=y, X=X,
X_train=X_train, X_test=X_test, y_train=y_train, y_test=y_test)
```

Setelah melakukan perhitungan *term frequency-invers document frequency* (TF-IDF) selanjutnya dilakukan pembuatan model klasifikasi yang dibuat dalam sebuah fungsi untuk memudahkan dalam pemanggilan. Tombol Button TF-IDF berfungsi untuk melakukan pembuatan model yang akan disimpan dalam bentuk file *.pickle* yang diberi nama variabel pipeline. Implementasi halaman pembuatan model klasifikasi dapat dilihat pada Gambar 4.7.



Gambar 4.7 Pembuatan Model Klasifikasi

Berikut ini potongan kode pada *controller* yang berfungsi untuk pembuatan model klasifikasi yang disimpan dalam bentuk file *.pickle*.

```
#<------------ FILE PICKLE ------------->
@application.route('/testing_tfidf', methods=['GET', 'POST'])
def testing_tfidf():
    data_testing = pd.read_excel('TA-
Melia/static/uploads/data_training.xlsx')
    data_testing.dropna()
    data_testing.drop(['Unnamed: 3', 'Unnamed: 4', 'Unnamed: 5',
'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12'], axis=1, inplace=True)

    bow_transformer = CountVectorizer().fit(data_testing['tweet'])
    messages_bow =
bow_transformer.transform(data_testing['tweet'])

    tfidf_transformer = TfidfTransformer().fit(messages_bow)

    messages_tfidf = tfidf_transformer.transform(messages_bow)
```

```
    pipeline = Pipeline([
        ('bow', CountVectorizer()),
        ('tfidf', TfidfTransformer()),
        ('classifier', MultinomialNB())
        ])

    X_train = np.asarray(data_testing['tweet'])
    pipeline = pipeline.fit(X_train,
np.asarray(data_testing['label']))

    file_data = pickle.dump(pipeline, open('TA-
Melia/static/uploads/latihan_ta.pickle', 'wb'))

    return render_template('testing_tfidf.html',
pipeline=pipeline)
```

4.3.6 Halaman *Testing* Data

Halaman *testing* digunakan untuk menampilkan hasil *Accuracy*, *Precission*, *Recall* dan *F1-score* berdasarkan data *tweet* yang sudah diprediksi kelas dan labelnya secara otomatis, untuk menentukan seberapa akurat model yang dibangun pada *training* data. Implementasi halaman *testing* data ditunjukkan pada Gambar 4.8.

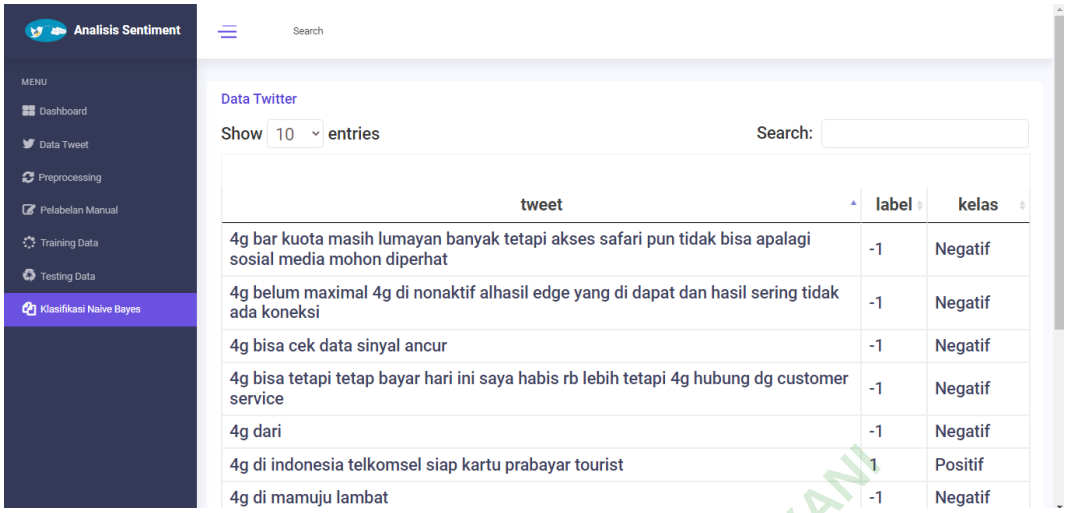
Gambar 4.8 Implementasi Halaman *Testing*

Berikut ini potongan kode pada *controller* yang berfungsi untuk menampilkan data *tweet* yang sudah diprediksi untuk mengetahui tingkat keakuratan pemodelan yang sudah dibangun pada *training*.

```
#<------------ TESTING ------------->
@application.route('/testing', methods=['GET', 'POST'])
def testing():
    import pickle
    vect = pickle.load(open('TA-
Melia/static/uploads/latihan_ta.pickle', 'rb'))

    testing = pd.read_excel('TA-
Melia/static/uploads/testing.xlsx')
    testing.dropna()
    testing.drop(['Unnamed: 0','Unnamed: 3', 'Unnamed: 4',
'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12'], axis=1, inplace=True)

    testing = testing['tweet'].fillna(' ')
    prediction = vect.predict(testing)

    result = []

    for i in range(len(prediction)):
        if(prediction[i] == 1):
            sentiment = 'Positif'
        elif(prediction[i]==0):
            sentiment = 'Netral'
        else:
            sentiment = 'Negatif'

        result.append({'tweet':testing[i],'label':prediction[i],
'kelas':sentiment })

    testing = pd.DataFrame(result)
    testing = testing.dropna()
    testing.to_excel('TA-Melia/static/uploads/hasil_testing.xlsx')
```


4.3.7 Halaman Klasifikasi Naive Bayes Classification

Halaman klasifikasi Naive Bayes Classification digunakan untuk menampilkan data *tweet* yang sudah diprediksi kelas dan labelnya secara otomatis berdasarkan model yang sudah dibangun pada tahap *training*. Implementasi halaman klasifikasi Naive Bayes Classification ditunjukkan pada Gambar 4.9.



Gambar 4.9 Implementasi Halaman Klasifikasi Naive Bayes Classification

Berikut ini potongan kode pada *controller* yang berfungsi untuk menampilkan data *tweet* yang sudah diprediksi.

```
#<------------ KLASIFIKASI NAIVE BAYES ------------->
@application.route('/klasifikasi', methods=['GET', 'POST'])
def klasifikasi():
    import pickle
    vect = pickle.load(open('TA-
Melia/static/uploads/latihan_ta.pickle', 'rb'))

    klasifikasi = pd.read_excel('TA-
Melia/static/uploads/data_tweet_keseluruhan.xlsx')
    klasifikasi.dropna()
    klasifikasi.drop(['Unnamed: 0','Unnamed: 3', 'Unnamed: 4',
'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7',
        'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10','Unnamed:
11','Unnamed: 12','Unnamed: 13'], axis=1, inplace=True)

    klasifikasi = klasifikasi['tweet'].fillna(' ')
    prediction = vect.predict(klasifikasi)

    result = []

    for i in range(len(prediction)):
        if(prediction[i] == 1):
            sentiment = 'Positif'
        elif(prediction[i]==0):
            sentiment = 'Netral'
        else:
            sentiment = 'Negatif'


result.append({'tweet':klasifikasi[i],'label':prediction[i],
'kelas':sentiment })
```

```
    klasifikasi = pd.DataFrame(result)
    klasifikasi = klasifikasi.dropna()
    klasifikasi.to_excel('TA-
Melia/static/uploads/hasil_data_keseluruhan.xlsx')

    return render_template('klasifikasi_naive_bayes.html',
klasifikasi=[klasifikasi.to_html(index=False, justify='center',
classes=['table-striped', 'table-bordered', 'table-hover', 'table-
condensed', 'dt-responsive', 'table-style'], table_id='example')])
```

4.4 PEMBAHASAN

Analisis sentimen berdasarkan kata kunci "Telkomsel" menggunakan data *tweet* sebanyak 13.239 data untuk dilakukan analisis. Data *tweet* sebanyak 600 data diberi kelas dan label secara manual dengan rata-rata akurasi yang cukup bagus sebesar 81,7% dan data *tweet* sebanyak 336 data yang sudah diprediksi secara otomatis berdasarkan model yang sudah dibangun pada *training* dengan rata-rata akurasi 84%. Pada pengujian Data *tweet* untuk dilakukan prediksi yang sudah dibangun pada *training* sebanyak 12.639 data, hasil prediksi didapatkan data *tweet* berlabel positif sebanyak 3.159 data, data *tweet* berlabel netral sebanyak 4.124 data dan data *tweet* berlabel negatif sebanyak 5.356 data.

Analisis sentimen kepuasan pelanggan perusahaan telekomunikasi seluler di Twitter dibangun menggunakan bahasa pemrograman Python dengan *framework* Flask, pada sistem analisis sentimen memiliki beberapa menu, yaitu menu dashboard, menu data *tweet*, menu *preprocessing*, menu pelabelan manual, menu *training* data, menu *testing* dan menu klasifikasi Naive Bayes Classification. Pada menu dashboard berfungsi untuk memberikan informasi berdasarkan hasil dari proses klasifikasi Naive Bayes Classification, menu data *tweet* digunakan untuk melakukan upload file dan menampilkan file yang sudah terupload didalam directory, menu *preprocessing* digunakan untuk melakukan proses pengolahan data teks dengan memerlukan beberapa tahapan-tahapan, menu pelabelan manual berfungsi untuk menampilkan hasil akurasi dari data *tweet* yang sudah diberi kelas atau label berdasarkan kalimat secara manual, menu *training* digunakan untuk membangun model klasifikasi sentimen secara otomatis dengan fungsi ekstraksi menggunakan *term frequency-invers document frequency* (TF-IDF) pada data teks, menu *testing* berfungsi untuk menampilkan hasil akurasi

berdasarkan data *tweet* yang diprediksi secara otomatis untuk menentukan seberapa akurat model yang dibangun pada *training*, menu klasifikasi Naive Bayes Classification berfungsi untuk menampilkan data *tweet* yang sudah diprediksi secara otomatis.

Grafik yang digunakan dalam sistem analisis sentimen ini adalah grafik bentuk *Bar*. Grafik tersebut secara otomatis menunjukan jumlah data *tweet* yang diprediksi berdasarkan kelas positif, negatif dan netral. Sistem analisis sentimen ini tentunya masih terdapat banyak kekurangan seperti dalam pemanggilan file belum bisa dilakukan pemanggilan secara otomatis.