

BAB IV

HASIL DAN PEMBAHASAN

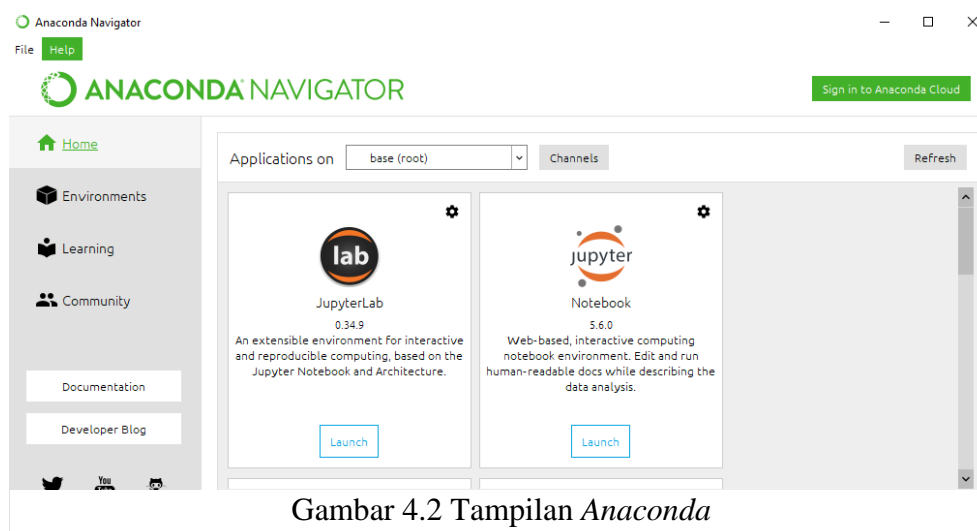
4.1 Pengambilan Data (*Crawling data*)

Pengambilan data ini atau proses *crawling* data Twitter ini menggunakan *API Key Twitter* dan proses pengambilan data Twitter dibantu dengan bahasa pemrograman *python*. *API Key Twitter* adalah *Application Programming Interface (API)* dalam *API* ini suatu layanan berisi sekumpulan perintah, fungsi, komponen dan juga protokol yang disediakan untuk mempermudah programme pada saat membangun suatu sistem perangkat lunak. *API Key Twitter* itu sendiri memiliki suatu *consumer keys*, *consumer secret*, *access key*, dan *access secret*. *Consumer keys*, *access key*, dan *access secret* tersebut digunakan untuk mengakses data Twitter yang dibutuhkan oleh programme pada Gambar 4.1 di bawah.

```
consumer_key = "I0E7xGaPsVIIyg0sq8tfw"  
consumer_secret = "frk7rjhTVHEuRErf4V2h93xZ6eSAr2myy9gH4RaU"  
access_key = "228245421-FPyC4SFufgkHDMHy7gTpEqm36mbFepYmkQ2p54xf"  
access_secret = "cYc2xsbekITAlb8RuQ9btHZxERGNofr1azKBPTnQ"
```

Gambar 4.1 *API Key Twitter*

Selain membutuhkan *API Key Twitter* untuk dapat melakukan pengambilan data Twitter disini peneliti menggunakan *tools* pendukung untuk menganalisis sentimen seperti *Anaconda* sebagai pendistribusi *Python*. Tampilan dari *anaconda* pada Gambar 4.2 di bawah.



Gambar 4.2 Tampilan *Anaconda*

Di sini peneliti menggunakan *Anaconda* sebagai *tools* karena di dalam *anaconda* sudah terdapat *Jupyter Notebook*. *Jupyter Notebook* biasa juga disebut *jupyter* ini adalah pengembangan dari *Ipython* atau *Interactive Python*. *Jupyter Notebook* ini suatu editor dalam bentuk web aplikasi yang berjalan di *localhost* komputer, adapun beberapa hal yang dapat dilakukan oleh *Jupyter Notebook* seperti menulis kode *python*, *equations*, *visualisasi* dan bisa juga sebagai *markdown editor*. Tampilan dari *Jupyter Notebook* pada Gambar 4.3 di bawah.



Gambar 4.3 Tampilan *Jupyter Notebook*

Dengan adanya *Jupyter Notebook* sekarang peneliti dapat melakukan proses pengkodean menggunakan bahasa pemrograman *Python* seperti Gambar 4.4 di bawah.

```
In [5]: import tweepy
import csv
import pandas as pd
from pandas import ExcelWriter

#Twitter API credentials

consumer_key = "I0E7xGaPsVIIyg0sq8tfw"
consumer_secret = "frk7rjhTVHEuERrf4V2h93xZ6e5Ar2myy9gH4Rau"
access_key = "228245421-FPyC45FufgkHDMHy7gTpEqm36mbFepYmkQ2p54xf"
access_secret = "cYc2xsbeKiTAlb8RuQ9btHZxERGNofr1azKBPTnQ"
```

Gambar 4.4 *Source Code* Pemanggilan *Python Library* Proses *Crawling*

Pada Gambar 4.4 di atas proses pendeklarasian *library tweepy*, *library csv*, *library pandas*. *library tweepy* adalah suatu API yang disediakan oleh pihak Twitter untuk dapat mengakses dan mengambil data-data yang ada di dalam Twitter menggunakan bahasa pemrograman *Python*. *Library Csv* (*Command Separated Values*) adalah *library* yang menyediakan layanan

baca dan menulis suatu data bertipe file csv atau excel. *Library pandas* adalah library pada Python yang berguna untuk pengolahan data. Setelah itu masukkan *consumer keys*, *consumer secret*, *access key*, dan *access secret* yang telah didapatkan pada Gambar 4.1 di atas. Setelah memasukkan kode di atas, selanjutnya memasukkan kode proses pada Gambar 4.5 di bawah.

```
In [6]: auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_key, access_secret)
        api = tweepy.API(auth,wait_on_rate_limit=True)

In [7]: csvFile = open('#andretaulanykufurnikmat.csv', 'a')
        csvWriter = csv.writer(csvFile)

In [ ]: for tweet in tweepy.Cursor(api.search,q="#andretaulanykufurnikmat",count=500,
                                   lang="id").items():
        print (tweet.created_at, tweet.text)
        csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
```

Gambar 4.5 Source Code Proses Crawling

Setelah menjalankan semua kode di atas maka akan didapat file excel yang belum diolah. Seperti Gambar 4.6 di bawah contoh file dari hasil *crawling* data.

Waktu	Tweet	label																		
05/05/2019 09:56	b'RT @lahan_poker: Andre Taulany Minta Maaf Pasca Hina Nabi, Proses Hukum Tetap Jalan! https://t.co/eq7xrexua5 #AndreTau																			
05/05/2019 09:50	b'Jaman sekarang, apa aja di politikin #SaveAndreTaulany #AndreTaulanyHinaRasulullah'																			
05/05/2019 08:16	b'RT @RiswandiDito: Dari sini kita bisa ambil pelajaran, Bercanda boleh tpi jangan berlebihan dan jangan bawa-bawa itu kedalar																			
05/05/2019 07:51	b'Andre Taulany Minta Maaf Pasca Hina Nabi, Proses Hukum Tetap Jalan! https://t.co/eq7xrexua5 https://t.co/U3D																			
05/05/2019 07:49	b'#andretaulany\n#AndreTaulanyMakinSongong\n#AndreTaulanyKufurNikmat\n#AndreTaulanyHinaRasulullah\nCoba cek ini:\n																			
05/05/2019 07:20	b'@PartaiSocmed Akun abal anal\n#AndreTaulanyHinaRasulullah'																			
05/05/2019 07:18	b'@hudlaha @PartaiSocmed Tangkap si pelawak cebong \n#AndreTaulanyKufurNikmat\n#AndreTaulanyHinaRasulullah'																			
05/05/2019 07:14	b"@umardhan @jokowi Kelakuan'a sama...apakah nasibnya akan sama??? we'll see\n#AndreTaulanyKufurNikmat\xe2\x80\xa6																			
05/05/2019 07:10	b'RT @RiswandiDito: Dari sini kita bisa ambil pelajaran, Bercanda boleh tpi jangan berlebihan dan jangan bawa-bawa itu kedalar																			
05/05/2019 07:05	b'@aburasyid13 Proses hukum tetap harus berjalan\n#AndreTaulanyHinaRasulullah\n#BoikotNetTV'																			
05/05/2019 06:32	b'RT @lahan_poker: BREAKING NEWS: Andre Taulany Minta Maaf Usai Hina Nabi, Pelapor: Proses Hukum Tetap Jalan! https://t.c																			
05/05/2019 06:21	b'RT @RiswandiDito: Dari sini kita bisa ambil pelajaran, Bercanda boleh tpi jangan berlebihan dan jangan bawa-bawa itu kedalar																			
05/05/2019 06:19	b'Saya setuju #PenjarakanAndreTaulany \n#AndreTaulanyHinaRasulullah https://t.co/4GjagcDvfV																			
05/05/2019 06:08	b'@iswan214 @prabowo siap2 nnti paspampers kewalahan\xfb\x9f\xa4\xa3\xfb\x9f\xa4\xa3 pak prabowo juga tak mau Ada jar																			

Gambar 4.6 File Excel Hasil Crawling

Pada saat proses pengambilan data Twitter, peneliti mengambil 3 sumber *Hastag* yang pada saat proses *crawling* data berada pada posisi *tranding topic* Twitter atau pada posisi pembahasan terbanyak pada tweet yaitu #andretaulanyhinarasulullah berjumlah 464 tweet pada tanggal 05/05/2019, #andretaulanykufurnikmat berjumlah 417 tweet pada tanggal 05/05/2019, #C1PlanoBabinsaAdalahKunci berjumlah 445 tweet pada tanggal 05/05/2019, anjing berjumlah 499 tweet pada tanggal 26/08/2019, babi berjumlah 398 tweet pada tanggal 01/09/2019, monyet berjumlah 277 tweet pada tanggal 01/09/2019. Semua data yang diambil berjumlah 2500 tweet.

bagian dari kalimat yang tidak berguna. Proses *preprocessing* ini dikerjakan menggunakan bantuan dari *library* pada bahasa pemrograman *Python 3*. Untuk mengerjakan proses *preprocessing* terdapat 4 tahapan proses untuk memperoleh hasil yang maksimal, sebagai berikut:

a. *Cleaning*

Pada proses *cleaning* ini berguna untuk mengurangi atau membersihkan data tweet dari kata atau kalimat yang tidak diperlukan seperti tanda baca, *unicode*, dan lain-lain. Proses *cleaning* ini terdapat 4 tahapan yang akan dilakukan oleh sistem untuk memperoleh hasil yang maksimal, seperti di bawah ini:

1. Membersihkan tanda baca
2. Membersihkan angka
3. Merubah huruf besar menjadi huruf kecil semua
4. Membersihkan kelebihan spasi

Beberapa kode program yang mengimplementasikan *cleaning* data dapat dilihat pada Gambar 4.8 di bawah.

```
def cleaning(str):
    #remove non-ascii
    str = unicodedata.normalize('NFKD', str).encode('ascii', 'ignore').decode('utf-8', 'ignore')
    str = re.sub("b'|b\"", '', str)
    #remove URLs
    str = re.sub(r'(?:i)\b(?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\\([^\s()<>]-
    #remove punctuations
    str = re.sub(r'^\w|_|', ' ', str)
    #remove digit from string
    str = re.sub("\S*\d\S*", "", str).strip()
    #remove digit or numbers
    str = re.sub(r"\b\d+\b", " ", str)
    #to lowercase
    str = str.lower()
    #Remove additional white spaces
    str = re.sub('[\s]+', ' ', str)
    return str
```

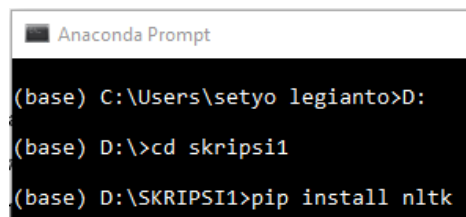
Gambar 4.8 kode program proses *cleaning*

Pada Gambar 4.8 di atas, keseluruhan dari proses *cleaning* dilakukan dengan menggunakan *regex* atau bisa juga disebut *regular expression*. *Regex* itu sendiri adalah konstruksi dalam suatu bahasa untuk mencocokkan teks berdasarkan pola tertentu, terutama untuk kasus-kasus kompleks.

b. *Remove Stopword*

Pada proses *remove stopwords* ini berguna sebagai menghapus kata *stopword* atau biasa disebut juga kata penghubung dari suatu kalimat seperti yang, dan, tetapi dan sebagainya. Dalam proses penghapusan *stopword* ini, terlebih dahulu dilakukan mendefinisikan kata-kata yang

nantinya akan terhapus ketika proses ini dijalankan. Dalam hal ini, seluruh kata-kata yang sudah di definisikan tadi disimpan di dalam sebuah file yang dengan nama *stopword_id*. File ini disimpan pada folder *corpora* yang terdapat di dalam *nlk_data*. Dalam proses penghapusan *stopword* ini dibantu dengan *library nltk* yang terdapat pada bahasa pemrograman *python3*. Dalam hal ini, peneliti melakukan proses install *library nltk* menggunakan *pip* sebagai perintah pada Gambar 4.9 di bawah.



```

Anaconda Prompt
(base) C:\Users\setyo legianto>D:
(base) D:\>cd skripsi1
(base) D:\SKRIPSI1>pip install nltk

```

Gambar 4.9 proses install *library nltk*

Setelah proses instalasi selesai, maka peneliti mendeklarasikan *library nltk* terlebih dahulu seperti pada Gambar 4.10 di bawah.

```

import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords

```

Gambar 4.10 Pendeklarasian *library nltk*

Selanjutnya proses pengimplementasian dari tahapan *remove stopwords* pada kode program Gambar 4.11 di bawah.

```

def removeStopword(str):
    stop_words = set(stopwords.words('stopwords_id'))
    word_tokens = word_tokenize(str)
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    return ' '.join(filtered_sentence)

```

Gambar 4.11 Kode Program Proses *Remove Stopword*

c. Tokenization

Pada proses *tokenization* berguna sebagai pemisah kata, simbol, frase dan entias dari suatu teks. Dalam proses ini dilakukan juga menggunakan bantuan *library nltk* pada bahasa pemrograman

python3. Adapun pengimplementasian dari kode program *tokenization* dapat dilihat pada Gambar 4.12 di bawah.

```
def word_tokenization(str):
    str = word_tokenize(str)
    return str
```

Gambar 4.12 Kode Program Proses *Tokenization*

d. *Stemming*

Pada proses *stemming* berguna sebagai penghapusan kata imbuhan dari setiap kata, baik kata imbuhan yang berada di depan kata ataupun di belakang kata. Dalam proses *stemming* dikerjakan menggunakan bantuan dari *library sastrawi* yang terdapat dalam bahasa pemrograman *python3*. Pada proses ini peneliti melakukan instalasi *library sastrawi* terlebih dahulu dengan menggunakan perintah *pip* pada Gambar 4.13 di bawah.

```
(base) D:\SKRIPSI1>pip install sastrawi
```

Gambar 4.13 Proses Instalasi *library sastrawi*

Adapun proses instalasi selesai, maka peneliti perlu mendeklarasikan *library sastrawi* terlebih dahulu pada Gambar 4.14.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

Gambar 4.14 Pendeklarasian *library sastrawi*

Selanjutnya proses pengimplementasian dari tahapan *stemming* pada kode program Gambar 4.1 di bawah.

```
def stemming(str):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return stemmer.stem(str)
```

Gambar 4.15 Kode Program Proses *Stemming*

Setelah semua proses *preprocessing* dijalankan terhadap semua data, maka hasil dari *preprocessing* disimpan menjadi suatu *file* baru yang nantinya akan dijadikan sebagai *dataset* dalam proses pengklasifikasian. Adapun hasil dari proses *preprocessing* pada Gambar 4.16 di bawah.

Tweet	cleantext	label
b'RT @lahan_poker: Andre Taulany Minta Maaf Pasca Hina Nabi, Proses Hukum Tetap Jalan! https://t.co/eq7xrexua5 #AndreTaulanyKufurNikmat #Andr\xe2\x80\xa6'	andre taulany minta maaf pasca hina nabi proses hukum tetap jalan andretaulanykufurnikmat andr	0
b'Jaman sekarang, apa aja di politikin #SaveAndreTaulany #AndreTaulanyHinaRasulullah'	jaman aja politikin saveandretaulany andretaulanyhinarasulullah	1
b'RT @RiswandiDito: Dari sini kita bisa ambil pelajaran, Bercanda boleh tpi jangan berlebihan dan jangan bawa-bawa itu kedalam keagamaan, sem\xe2\x80\xa6'	ambil ajar canda tpi lebih bawa bawa dalam agama	1
b'Andre Taulany Minta Maaf Pasca Hina Nabi, Proses Hukum Tetap Jalan! https://t.co/eq7xrexua5 \xe2\x80\xa6 https://t.co/U3DCyCb0zY '	andre taulany minta maaf pasca hina nabi proses hukum tetap jalan	0
b'#andretaulany\n#AndreTaulanyMakinSongong\n#AndreTaulanyKufurNikmat\n#AndreTaulanyHinaRasulullah\nCoba cek ini:\nAndre T\xe2\x80\xa6 https://t.co/2xcRkXGywW '	andretaulany andretaulanymaksongong andretaulanykufurnikmat andretaulanyhinarasulullah ncoba cek nandre	1
b'@PartaiSocmed Akun abal anal\n#AndreTaulanyHinaRasulullah'	akun abal anal andretaulanyhinarasulullah	1
b'@hudlaha @PartaiSocmed Tangkap si pelawak cebong \n#AndreTaulanyKufurNikmat\n#AndreTaulanyHinaRasulullah'	tangkap lawak cebong andretaulanykufurnikmat andretaulanyhinarasulullah	1

Gambar 4.16 Hasil *Preprocessing*

4.3 Ekstraksi Fitur

Pada proses ekstraksi fitur, proses pertama yang dilakukan oleh sistem setelah *tokenization* yaitu mengubah dataset menjadi suatu representasi *vector* dengan menggunakan *library* yang sudah disediakan oleh *Phyton* yang bernama *library Count Vectorizer*. Sebagai contoh penelitian menggunakan 3 komentar, diantaranya :

(Doc1)"Cowok itu bajunya bagus sekali"

(Doc2)"Mulutnya hancur banget seperti mulut anjing"

(Doc3)"Cowok itu sangat hancur"

Setelah sistem melakukan *preprocessing* terdapat 4 jumlah kata baku dari 3 kalimat di atas yaitu "Cowok", "Bagus", "Mulut", dan "Hancur".

Setelah tahapan di atas dari setiap dokumen ditampilkan mejadi sebuah *vector* dengan elemen, ketika kata tersebut terdapat di dalam dokumen maka diberikan nilai 1, jika tidak ada maka diberikan nilai 0. Sebagai contoh terdapat pada Tabel 4.1 di bawah.

Tabel 4.1 Pembuatan *Word Vector*

	Cowok	Bagus	Mulut	Hancur
Doc1	1	1	0	0

Doc2	0	0	2	1
Doc3	1	0	0	1

Dokumen yang telah diubah menjadi *word vector* selanjutnya akan dihitung menggunakan rumus *TF-IDF*, dengan menggunakan rumus ini maka akan menghasilkan *word vector* yang memiliki nilai yang sudah terbobot. *TF* atau *Term Frequency* itu sendiri adalah banyaknya frekuensi kemunculan kata dari suatu *term* dalam dokumen bersangkutan, sedangkan *IDF* atau *Inverse Document Frequency* adalah perhitungan dari bagaimana *term* disebar atau didistribusikan secara luas dalam koleksi dokumen yang bersangkutan.

Proses perhitungan bobot kata dilakukan dengan proses awal menghitung *TF* atau *Term Frequency* terlebih dahulu. Dapat dilihat contoh pada Tabel 4.2 di bawah.

Tabel 4.2 Proses Perhitungan *TF* (*Term Frequency*)

	<i>(Doc1)</i>	<i>(Doc2)</i>	<i>(Doc3)</i>
Cowok	1	0	1
Bagus	1	0	0
Mulut	0	2	0
Hancur	0	1	1

Setelah proses perhitungan bobot *TF* selesai selanjutnya dilakukan proses menentukan *DF* atau *Document Frequency* yaitu dengan banyaknya *term* (*t*) muncul dalam semua dokumen. Maka akan memperoleh hasil seperti Tabel 4.3 di bawah.

Tabel 4.3 Proses Perhitungan *DF* (*Document Frequency*)

<i>T (Term)</i>	<i>DF (Document Frequency)</i>
Cowok	2
Bagus	1
Mulut	2
Hancur	2

Kemudian setelah proses *TF* dan *DF* kemudian dilanjutkan menghitung nilai *IDF* (*Inverse Document Frequency*) dengan cara menghitung nilai dari log hasil D atau jumlah dokumen dalam contoh kasus ini ada 3 dokumen, dari 3 dokumen tersebut dibagi dengan nilai *DF*

(*Document Frequency*). Maka akan menghasilkan nilai perhitungan seperti Tabel 4.4 di bawah.

Tabel 4.4 Proses *IDF* (*Inverse Document Frequency*)

<i>T (Term)</i>	<i>DF (Document Frequency)</i>	<i>D/DF</i>	<i>IDF (Inverse Document Frequency)</i>
Cowok	2	1,5	$\log 1,5 = 0,176$
Bagus	1	3	$\log 3 = 0,477$
Mulut	2	1,5	$\log 1,5 = 0,176$
Hancur	2	1,5	$\log 1,5 = 0,176$

Setelah mendapatkan nilai *IDF* (*Inverse Document Frequency*), selanjutnya dilanjutkan dengan menghitung *TF-IDF*. Seperti pada Tabel 4.5 di bawah.

Tabel 4.5 Contoh Proses Perhitungan *TF-IDF*

<i>Q</i>	<i>TF</i>			<i>DF</i>	<i>D/DF</i>	<i>IDF</i>	<i>IDF+1</i>	<i>W = TF*(IDF+1)</i>		
	<i>Doc 1</i>	<i>Doc 2</i>	<i>Doc 3</i>					<i>Doc 1</i>	<i>Doc 2</i>	<i>Doc 3</i>
Cowok	1	0	1	2	1,5	0,176	1,176	1,176	0	1,176
Bagus	1	0	0	1	3	0,477	1,477	1,477	0	0
Mulut	0	2	0	2	1,5	0,176	1,176	0	2,352	0
Hancur	0	1	1	2	1,5	0,176	1,176	0	1,176	1,176
Nilai Bobot Dari Setiap Dokumen								2,653	3,528	2,352

Hasil dari *word vector* yang sudah mendapatkan bobot dapat dilihat pada Tabel 4.6 di bawah.

Tabel 4.6 Contoh *Word Vector* yang sudah dibobotkan

	Pantai	Bagus	Taman	Indah
(Doc1)	1,176	1,477	0	0
(Doc2)	0	0	2,352	1,176
(Doc3)	1,176	0	0	1,176

4.4 Implementasi Klasifikasi *Naïve Bayes*

Pada proses ekstraksi fitur dan proses pengklasifikasian *Naïve Bayes* yang nantinya akan di compres menjadi satu *class pipeline vectorizer => transformer => classifier*. Proses pengklasifikasian tersebut berjalan dengan bantuan *library* pada bahasa pemrograman Python3 yang mempunyai nama *library scikit-learn* untuk proses pengklasifikasian, selain itu terdapat *library numpy* dan juga *pandas* sebagai pembacaan data.

Untuk *library scikit-learn* disini yang digunakan adalah *Pipeline, CountVectorizer, Naïve Bayes, MultinomialNB, Confusion Matrix, TfidfTransformer*, dan *f1 Score*.

Untuk langkah awal pengerjaan proses ekstraksi fitur dan klasifikasi adalah dilakukan proses menginstall *library* yang diperlukan. Selanjutnya setelah semua *library* terinstall maka dilanjutkan ke proses mendeklarasi semua *library* yang akan digunakan. Adapun kode program untuk deklarasi pada Gambar 4.17 di bawah.

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.naive_bayes import MultinomialNB
4 from sklearn.svm import LinearSVC, SVC
5 from sklearn.feature_extraction.text import CountVectorizer
6 from sklearn.feature_extraction.text import TfidfTransformer
7 from sklearn.pipeline import Pipeline
8 from sklearn.model_selection import train_test_split
9 from sklearn.model_selection import cross_val_score
10 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_score, precision_score, recall_score

```

Gambar 4.17 Proses pendeklarasian *library* yang digunakan

Setelah selesai mendeklarasi *library* dilanjutkan dengan proses mengambil *dataset* yang akan dipergunakan sebagai data *training* menggunakan *library pandas*. Untuk kode program tersebut pada Gambar 4.18 di bawah.

```

1 data = pd.read_excel('AndreTaulanyHinaRasulullah_excel_label-clean.xlsx', encoding='Latin-1')
2 len(data)

```

Gambar 4.18 Proses memanggil *data set*

Selanjutnya proses pembuatan *class pipeline* yang di dalamnya terdapat 3 tahapan yaitu mengubah *dataset* yang hasil *crawling data* Twitter menjadi *representasi vector* (mengubah huruf menjadi angka) menggunakan *library CountVectorizer* dengan pembobotan menggunakan *word vector* dalam *library TfidfTransformer*, tahapan terakhir dilakukan

klasifikasi dengan menggunakan *library MultinomialNaiveBayes*. Proses dari pengimplementasian dari tiga proses pembuatan *class pipeline* pada Gambar 4.19 di bawah.

```

1 #Multinomial Naive Bayes
2 pipeline_mnb = Pipeline([
3     ('vect', CountVectorizer()),
4     ('tfidf', TfidfTransformer(use_idf=True, smooth_idf=True)),
5     ('clf', MultinomialNB(alpha=1))
6 ])
7
8 txt = data['cleantext'].values.astype('U')
9 #X_train, X_test, y_train, y_test = train_test_split(data['cleantext'], data['label'], test_size=0.33, random_state = 0)
10 X_train, X_test, y_train, y_test = train_test_split(txt, data['label'], test_size=0.33, random_state = 0)
11 pipeline_mnb.fit(X_train, y_train)

```

Gambar 4.19 Proses Pengimplementasian *Class Pipeline*

Pada proses pengklasifikasian data ini, peneliti menggunakan data tes yang diacak dari 33% atau 0,33 dari data *training*. Proses pengklasifikasian data ini dilakukan dengan menggunakan perhitungan *probabilitas* dari setiap kelas, maka peneliti baru bisa mendapatkan hasil jelas dari prediksi data yang di-*input*. Tahapan akhir setelah melakukan semua proses pengklasifikasian, maka barulah bisa menghitung dari performa dari *algoritme* yang dipergunakan.

4.5 Uji Model

Untuk mengetahui tingkatan dari performa *Algoritme Naive Bayes*, maka peneliti melakukan pengujian terhadap model. Hasil dari klasifikasi nantinya akan ditampilkan dalam bentuk *confusion matrix*. Tabel yang ditampilkan di dalam *confusion matrix* ini terdiri dari kelas *predicted* dan juga kelas *actual*. Model dari *confusion matrix* ini dapat dilihat pada Tabel 4.7.

Tabel 4.7 Model *Confusion Matrix*

		Predict Class	
		Class A	Class B
Actual class	Class A	AA	AB
	Class B	BA	BB

Untuk mengetahui nilai dari akurasi model diperoleh dari banyak jumlah data yang tepat hasil klarifikasi dibagi dengan total dari data, seperti pada Gambar 4.20 di bawah.

$$\text{Akurasi} = \frac{AA+BB}{AA+AB+BA+BB}$$

Gambar 4.20 Hasil Akurasi

Pada saat proses pengujian model maka akan mendapatkan hasil dari nilai akurasi dan *confusion matrix* 2x2 pada Gambar 4.21 di bawah.

```
Total documents classified: 2500
Accuracy: 0.7103
Confusion matrix:
[[358 96]
 [143 228]]
precision    recall  f1-score   support
0           0.71     0.79     0.75     454
1           0.70     0.61     0.66     371
avg / total     0.71     0.71     0.71     825
```

Gambar 4.21 Nilai Akurasi dan *Confusion Matrix* 2x2

Nilai akurasi yang didapatkan dari pengujian model sebesar 71.0% yang proses perhitungannya berdasarkan jumlah nilai dari diagonal *confusion matrix* dibagi dengan seluruh jumlah data. Karena jumlah pada data setiap kelas data *training* tidak seimbang, maka besarnya nilai akurasi bukanlah terpenting.

4.6 Evaluasi Model

Dalam proses evaluasi model ini dilakukan setelah uji model telah selesai dilakukan. Evaluasi model berguna sebagai menghitung performa dari metode yang dipilih. Pada proses uji model ini akan menghasilkan *confusion matrix* dengan ukuran 2x2 yang dapat dilihat pada Tabel 4.8 di bawah.

Tabel 4.8 Hasil *Confusion Matrix*

		Predict Class	
		<i>Positive</i>	<i>Negative</i>
<i>Actual class</i>	<i>Positive</i>	358	96
	<i>Negative</i>	143	228

Seperti pada Tabel 4.8 di atas, *confused matrix* matriks yang berukuran 2x2 setiap kolomnya mewakili nilai dari setiap kelas yaitu kelas *positive*, dan kelas *negative*.

Berdasarkan rumus yang terdapat dalam bab sebelumnya nilai presisi pada keseluruhan sistem bernilai sebesar **0.704** dan untuk nilai dari *recall* keseluruhan sistem berupa **0.615** sedangkan untuk nilai dari *f-1 Score* untuk pengevaluasian dalam informasi temu kembali yang dihitung mengombinasikan nilai dari *presisi* dan *recall* yaitu sebesar **0.656**. Untuk menghitung proses menghitung dari nilai *presisi*, *recall* dan *f-1 score* pada sistem ini dapat pada Gambar 4.22 di bawah.

```

8 txt = data['cleantext'].values.astype('U')
9 #X_train, X_test, y_train, y_test = train_test_split(data['cleantext'], data['label'], test_size=0.33, random_state = 0)
10 X_train, X_test, y_train, y_test = train_test_split(txt, data['label'], test_size=0.33, random_state = 0)
11 pipeline_mnb.fit(X_train, y_train)
12 predictions = pipeline_mnb.predict(X_test)
13
14 print("Accuracy: {}".format(accuracy_score(y_test, predictions)))
15 print("F1 Score: {}".format(f1_score(y_test, predictions)))
16 print("Precision score: {}".format(precision_score(y_test, predictions)))
17 print("Recall score: {}".format(recall_score(y_test, predictions)))
18 print("Confusion matrix: {}".format(confusion_matrix(y_test, predictions)))

```

Gambar 4.22 Proses Menghitung dari Nilai *Presisi*, *Recall* dan *F-1 score*

Dengan diketahuinya nilai dari *precision*, *recall*, dan *f-1 Score* dalam kinerja di keseluruhan sistem, maka dapat mengetahui kemampuan dari sistem untuk mencari ketepatan atau kebenaran dari informasi yang diminta oleh pengguna dengan hasil jawaban yang dikeluarkan oleh sistem dan memberitahu tingkat keberhasilan dari suatu sistem dalam menentukan kembali suatu informasi atau nilai *accuracy* sebesar **71%**.

Setelah proses di atas selesai, untuk performa dari metode pengklasifikasian dari setiap kelas dapat diketahui dengan *precision*, *recall*, dan *f-1 Score* di setiap kelasnya. Hasil dari *precision*, *recall*, dan *f-1 Score* memiliki ukuran penilaian sebesar 0-1. Semakin tinggi nilai maka semakin baik, dalam artian semakin mendekati angka 1 nilai dari 0 maka sistem semakin baik. Hasil dari proses pengevaluasian model keseluruhan sistem ini terdapat pada Gambar 4.23 di bawah.

```

Total documents classified: 2500
Accuracy: 0.7103
F1 Score: 0.6561
Precision score: 0.7037
Recall score: 0.6146
Confusion matrix:
[[358 96]
 [143 228]]

```

	precision	recall	f1-score	support
0	0.71	0.79	0.75	454
1	0.70	0.61	0.66	371
avg / total	0.71	0.71	0.71	825

Gambar 4.23 Hasil dari Proses Pengevaluasian Model

Hasil dari nilai *precision*, *recall*, dan *f-1 Score* di setiap kelas terdapat pada Tabel 4.9 di bawah.

Tabel 4.9 Hasil dari Nilai *Precision*, *Recall*, dan *F-1 score*

Jenis Klasifikasi	<i>Precision</i>	<i>Recall</i>	<i>F-1 Score</i>
Positif	0,71	0,79	0,75
Negatif	0,70	0,61	0,66

Dapat dilihat dari hasil evaluasi model dapat dilihat nilai *precision*, dan *recall* dari setiap kelas dapat dilihat tingkat kemampuan pemrosesan sistem dalam mencari tingkat ketepatan antara informasi yang diinginkan oleh pengguna sebagai kelas *positif* adalah “71%”, dan untuk kelas *negatif* adalah “70%”. Tingkat keberhasilan dari pemrosesan sistem dalam memperoleh kembali informasi kelas *positif* adalah “79%”, untuk kelas *negatif* adalah “61%”. Dengan nilai-nilai tersebut dapat dikatakan kinerja sistem dari keberhasilan sistem untuk menemukan kembali suatu informasi yang bernilai *positif* dan *negatif* dalam dokumen sangat rendah.

Untuk itu dilakukan proses pengujian ulang untuk menentukan hasil uji dan evaluasi yang maksimal dengan menggunakan *k-fold cross validation*. *K-fold cross validation* ini adalah metode *Cross Validation* yang digunakan melipat data sebanyak K dan mengiterasi (pengulangan) sebanyak K. Dalam penelitian ini pengujian menggunakan nilai K yaitu 5. Dalam *5 fold*, data dibagi menjadi *5 fold* berukuran kira-kira sama, sehingga sistem memiliki *5 subset* data sebagai pengevaluasian kinerja algoritme atau model. Hasil pengujian sistem menggunakan metode *5 fold cross validation* pada Gambar 4.24 di bawah.


```

Total documents classified: 2500
Accuracy: 0.7100
F1 Score: 0.7496
Precision score: 0.7668
Recall score: 0.7331
Confusion matrix:
[[138 66]
 [ 79 217]]

```

	precision	recall	f1-score	support
0	0.64	0.68	0.66	204
1	0.77	0.73	0.75	296
avg / total	0.71	0.71	0.71	500

Gambar 4.24 Hasil Pengujian 5 *K-Fold Cross Validation*

Dengan menggunakan *cross validation* dapat dilihat nilai dari *accuracy* tidak berubah dari sebelumnya yaitu sebesar **0,710** atau **71%**. Untuk dari hasil *precision*, *recall*, dan *f-1 score* mengalami perubahan hasil setiap *class* dapat dilihat pada Tabel 4.10 di bawah.

Tabel 4.10 Hasil *Precision*, *Recall*, dan *F-1 score*

Jenis Klasifikasi	<i>Precision</i>	<i>Recall</i>	<i>f-1 Score</i>
Positif	0,64	0,68	0,66
Negatif	0,77	0,73	0,75

Dari hasil evaluasi model menggunakan *fold validation* dilihat dari Tabel 4.10 di atas nilai dari *precision* dan *recall* di setiap *class* nya mengalami peningkatan dari kemampuan sistem untuk mencari ketepatan antara informasi yang pengguna minta untuk *precision* kelas *positif* **64%**, dan kelas negatif sebesar **77%**. Sedangkan dari tingkat keberhasilan sistem dalam menemukan suatu informasi kembali untuk hasil *recall* kelas *positif* **68%**, dan untuk kelas negatif **73%**. Dengan hasil dalam Tabel 4.10 di atas maka kinerja sistem tingkat keberhasilan sistem untuk menemukan kembali suatu informasi yang bernilai *positif* dan *negatif* dalam dokumen sangat rendah tetapi mengalami peningkatan dari evaluasi sebelumnya dapat dilihat dari nilai *precision* keseluruhan menjadi **76,7%**, nilai *recall* keseluruhan menjadi **73,3%**, dan juga nilai dari *f1-score* keseluruhan menjadi **74,9%**. Hasil dari nilai *K-fold validation* dapat dilihat pada Gambar 4.25 di bawah.

```

Accuracy: 0.7100
F1 Score: 0.7496
Precision score: 0.7668
Recall score: 0.7331
Confusion matrix:
[[138 66]
 [ 79 217]]

```

	precision	recall	f1-score	support
0	0.64	0.68	0.66	204
1	0.77	0.73	0.75	296
avg / total	0.71	0.71	0.71	500

Gambar 4.25 Hasil *Fold Validation*

Adapun peneliti membuat perbandingan antara hasil pengujian dari tingkat *accuracy*, *precision*, *recall*, dan *f1-score* terhadap beberapa model yang berbeda seperti pada Tabel 4.11 di bawah.

Tabel 4.11 Perbandingan Metode penelitian

Metode	Fitur Extraction	Accuracy	Precession	Recall	F1-Score
Naïve Bayes	<i>TF-IDF,</i> <i>CountVectorizer</i>	0,710	0,704	0,615	0,657
	<i>CountVectorizer</i>	0,697	0,658	0,679	0,668
	<i>Bi gram</i>	0,676	0,720	0,458	0,560
	<i>TF-IDF,</i> <i>Bi gram</i>	0,680	0,742	0,442	0,554
	<i>Tri gram</i>	0,642	0,740	0,315	0,442
	<i>TF-IDF,</i> <i>Tri gram</i>	0,651	0,798	0,299	0,435
	<i>TF-IDF,</i> <i>CountVectorizer,</i> <i>K fold 5</i>	0,710	0,766	0,733	0,749
SVM	<i>TF-IDF,</i> <i>CountVectorizer</i>	0,710	0,714	0,592	0,648
	<i>CountVectorizer</i>	0,710	0,736	0,555	0,632
	<i>Bi gram</i>	0,670	0,759	0,390	0,516
	<i>TF-IDF,</i> <i>Bi gram</i>	0,678	0,733	0,444	0,553

	<i>Tri gram</i>	0,649	0,797	0,296	0,432
	<i>TF-IDF, Tri gram</i>	0,651	0,798	0,299	0,435
	<i>TF-IDF, CountVectorizer, K fold 5</i>	0,562	0,789	0,355	0,489
Logistic Regression	<i>TF-IDF, CountVectorizer</i>	0,709	0,739	0,544	0,627
	<i>CountVectorizer</i>	0,716	0,729	0,587	0,650
	<i>Bi gram</i>	0,674	0,768	0,393	0,521
	<i>TF-IDF, Bi gram</i>	0,674	0,765	0,396	0,522
	<i>Tri gram</i>	0,650	0,797	0,297	0,432
	<i>TF-IDF, Tri gram</i>	0,650	0,797	0,297	0,432
	<i>TF-IDF, CountVectorizer, K fold 5</i>	0,548	0,769	0,337	0,469

Pada Tabel 4.11 di atas dapat dilihat hasil metode dan *fitur extraction* paling besar terdapat pada metode *Naïve Bayes* dengan menggunakan *fitur extraction TF-IDF, CountVectorizer, K fold 5* yaitu nilai *Accuracy* sebesar **0,710**, *Precession* sebesar **0,766**, *Recall* sebesar **0,733**, *F1-Score* **0,749**. Hasil metode dan *fitur extraction* paling kecil terdapat pada metode *Logistic Regression* dengan menggunakan *fitur extraction TF-IDF, CountVectorizer, K fold 5* yaitu nilai *Accuracy* sebesar **0,548**, *Precession* sebesar **0,769**, *Recall* sebesar **0,337**, *F1-Score* **0,469**.