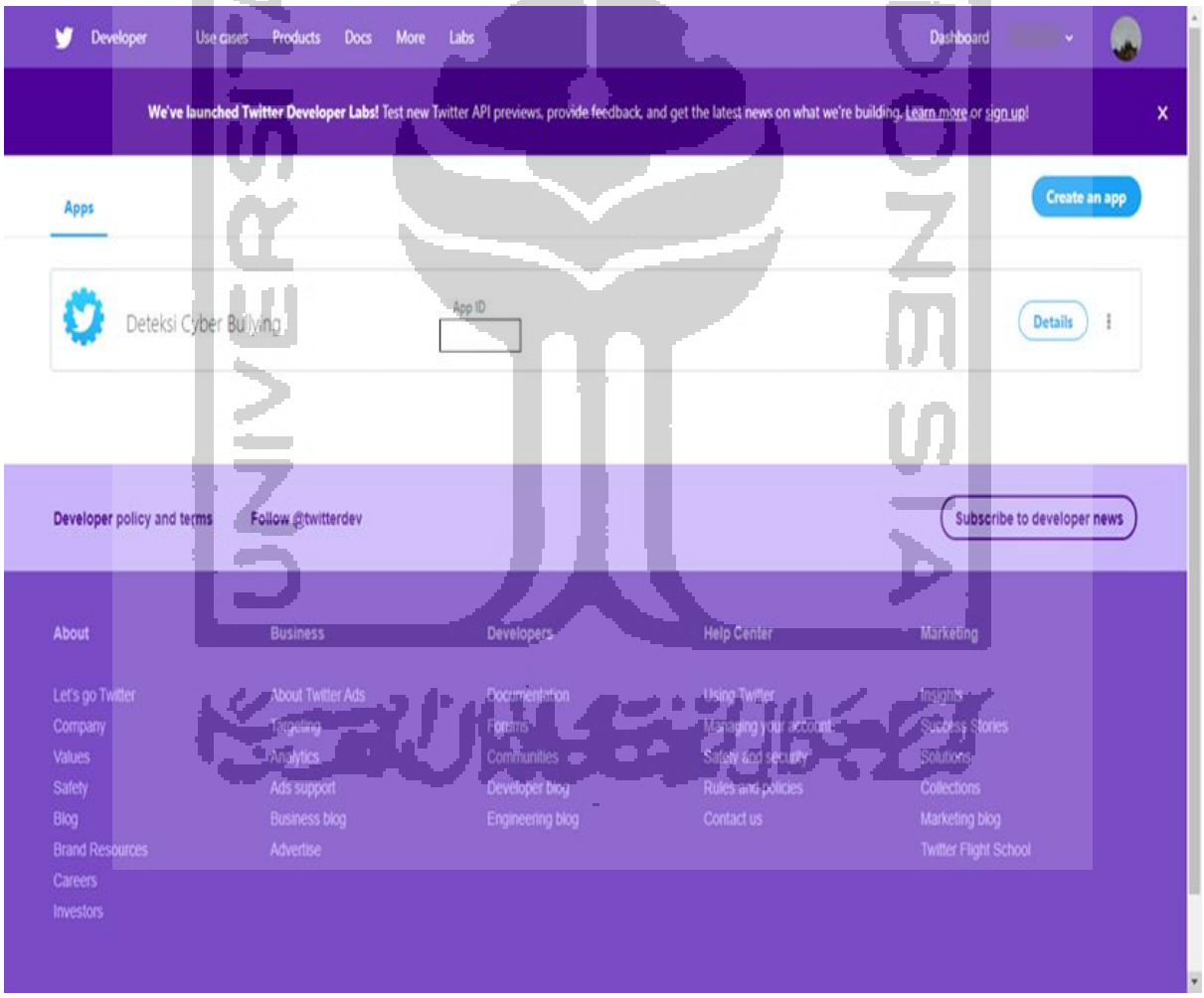
BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengambilan Data

Data cuitan dari media sosial *Twitter* diambil dengan teknik *crawling* dan tidak menggunakan aplikasi pihak ketiga ataupun ekstensi *browser*. Kode program dalam bahasa pemrograman *python* disiapkan untuk mengambil data cuitan dari media sosial *Twitter* berupa file dengan format *json*.

Sebelum melakukan *crawling data*, penulis terlebih dahulu membuat akun pada media sosial *Twitter* kemudian mengakses laman developer.twitter.com untuk mendapatkan hak akses terhadap *Twitter API*.



Gambar 4.1 Halaman developer.twitter.com

Gambar 4.1 merupakan halaman pada situs developer.twitter.com yang menampilkan daftar aplikasi yang diajukan untuk menggunakan *Twitter API*. Jika daftar aplikasi kosong,

pengguna bisa membuat pengajuan aplikasi dengan menekak tombol *create an app* kemudian disuguhkan beberapa pertanyaan sebelum diberikan akses untuk menggunakan *Twitter API* berupa *Consumer Key, Consumer Secret, Access Token,* dan *Access Secret*.

Setelah mendapat *Consumer Key, Consumer Secret, Access Token,* dan *Access Secret,* kode program dijalankan dengan menggunakan hak akses yang telah diberikan. *Crawling data* dilakukan dengan mengatur *longitude* dan *latitude* sehingga mendapatkan data cuitan yang berasal dari pengguna di Indonesia.

```
def load_api():
    ''' Function that loads the twitter API after authorizing the user. '''

    consumer_key = ''
    consumer_secret = ''
    access_token = ''
    access_secret = ''
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
    # load the twitter API via tweepy
    return tweepy.API(auth)
```

Gambar 4.2 Kode penggunaan *Twitter API*

Data yang didapatkan merupakan *file* dengan format *json*. *File* tersebut kemudian diubah formatnya menjadi *file* dengan ekstensi *.xlsx* dengan kode program yang bisa dilihat pada gambar 4.3 bawah ini:

```
import pandas

pandas.read_json("file.json", lines=True).to_excel("file.xlsx")
```

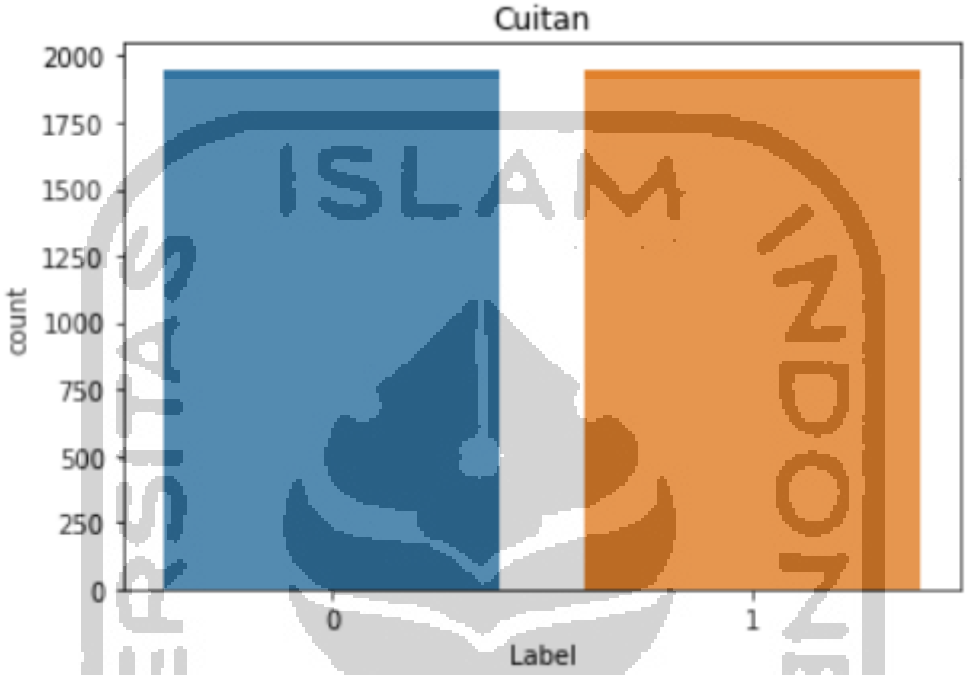
Gambar 4.3 Kode konversi *json* ke *xlsx*

Setelah melewati tahap konversi format data, didapatkan 3900 baris data cuitan. Data yang sudah berganti format ke dalam ekstensi *.xlsx* selanjutnya diberikan *labeling* secara manual oleh penulis. Data cuitan dibagi menjadi dua kelas, yaitu kelas cuitan *cyberbullying* dengan label satu (1) dan kelas cuitan *non-cyberbullying* dengan label nol (0). Contoh data yang telah dikumpulkan dapat dilihat pada tabel 4.1:

Tabel 4.1 Contoh data yang dikumpulkan

Cuitan	Label
Apakah ciri ciri orang yang so ngartis adalah "selalu membalas pesan dari siapapun"? Jawab dan jelaskan -terimakasih-	0
apakah semua orang akan sombong pada waktunya?☹	0
Asa jadi koala gendut -_-	0
@duhalupa @accmilea TOLOL WKWKWKWKEK	1
@duhmaap Mungkin mereka adalah penduduk instagram yg hijrah ke twt makanya ngartis wkwk	1
@dya_11677 @VIVAcoid jangan berkomentar kalo hanya mempertontonkan kebodohan diri sendiri !! TOLOL	1

Jumlah data cuitan yang bermakna *cyberbullying* adalah 1950 baris data, sedangkan jumlah data cuitan yang bermakna *non-cyberbullying* sebesar 1950 baris data. Perbandingan jumlah data setelah dilakukan proses *labeling* dapat dilihat pada gambar 4.4 di bawah ini:



Gambar 4.4 Perbandingan jumlah data

4.2 Preprocessing

Seperti yang dijelaskan pada bagian sebelumnya, tujuan dari tahap *preprocessing* ini adalah untuk melakukan pembersihan data cuitan dari kata, simbol, dan hal-hal lain yang kurang bermakna. Tahap *preprocessing* yang dilakukan pada penelitian ini akan dijelaskan secara berurutan

a. *Cleaning*

Tahap *cleaning* dilakukan untuk membersihkan data cuitan dari tanda baca, simbol dan sebagainya. Lebih jelas, langkah-langkah *cleaning* yang dilakukan adalah:

1. Menghilangkan URL
2. Menghapus karakter NON-ASCII
3. Menghapus angka, simbol dan tanda baca
4. Menghapus hashtag, username, dan RT
5. Penyeragaman huruf ke dalam bentuk *lowercase*

Adapun kode program yang digunakan untuk implementasi tahap ini dapat dilihat pada gambar 4.5.

```
import re
import string
import unicodedata
import nltk

def cleaning(str):

    #menghilangkan URL
    str    =    re.sub(r'(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-
z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+
|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:\'".,<>?«»“”‘’]))', '', str)

    #menghapus karakter non-ASCII
    str      =      unicodedata.normalize('NFKD',      str).encode('ascii',
'ignore').decode('utf-8', 'ignore')

    #menghapus angka, simbol, tanda baca
    str = re.sub(r'[^a-z ]*([.0-9])*\d', '', str)

    #menghapus karakter twitter
    # mention
    str = re.sub(r'(?:@[\w_]+)', ' ', str)
    # hashtag
    str = re.sub(r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)", " ", str)
    # RT/cc
    str = str = re.sub('RT', '', str)

    #casefolding
    str = str.lower()

    return str
```

Gambar 4.5 Kode tahapan *cleaning*

b. *Normalize*

Data cuitan yang didapat bukan merupakan kalimat-kalimat yang sesuai dengan penulisan yang benar. Seringkali ditemukan kata-kata ditulis dengan singkatan dan sebutan-sebutan kekinian.sebagai contoh, kata ‘yang’ disingkat menjadi ‘yg’, kata ganti ‘aku’ diubah menjadi ‘ane’ dan masih banyak lagi. Pada tahapan ini, dilakukan proses penyesuaian kata sehingga menjadi kata dengan ejaan yang baku.

Kode program yang diimplementasikan pada tahap ini seperti yang terlihat pada gambar 4.6.

```
import pandas as pd
import re
import string
import nltk


def normalize_slang_word(str):
    text_list = str.split(' ')
    slang_words_raw=pd.read_csv('D:\Kuliah\TUGAS
AKHIR\Data\slang_word_list.csv', sep=',', header=None)
    slang_word_dict = {}

    for item in slang_words_raw.values:
        slang_word_dict[item[0]] = item[1]

        for index in range(len(text_list)):
            if text_list[index] in slang_word_dict.keys():
                text_list[index] = slang_word_dict[text_list[index]]

    return ' '.join(text_list)
```

Gambar 4.6 Kode tahap *Normalize*

c. *Stemming*

Tahap ini dilakukan untuk mengubah kata-kata pada data cuitan ke dalam bentuk kata dasarnya. Kode program yang digunakan seperti yang terlihat pada gambar 4.7.

```
import re
import string
import unicodedata
import nltk
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def stemm(str):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    text = stemmer.stem(str)

    return text
```

Gambar 4.7 Kode program tahap *Stemming*

d. *Remove Stopword*

Pada bagian ini, kata-kata yang kurang bermakna dihilangkan untuk alasan efisiensi data. Daftar kata-kata yang kurang penting dibuat dalam sebuah *file* sebagai kamus dalam tahap ini. Selain menghapus kata yang kurang penting, proses ini juga melakukan pemecahan kaliam menjadi bagian-bagian penting yang dinamakan token. Kode program tahap ini dapat dilihat pada gambar 4.8.

```
import re
import string
import unicodedata
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords

def remove_stopword(str):
    stop_words            =            set(stopwords.words('D:\Kuliah\TUGAS
AKHIR\Data\stopwordID.csv'))
    word_tokens = word_tokenize(str)
    filtered_sentence = [w for w in word_tokens if not w in stop_words]

    return ' '.join(filtered_sentence)
```

Gambar 4.8 Kode program tahap *Remove Stopword*

Setelah semua data telah melalui proses *preprocessing* maka data-data cuitan sudah bisa diproses ke tahap selanjutnya. Berikut ditampilkan contoh penerapan tahap *preprocessing* pada data cuitan.

Tabel 4.2 Contoh penerapan *preprocessing*

No	Sebelum	Sesudah
1	vivanewscom: Truk Hantam Kendaraan di Cianjur, Salah Satunya Rombongan Pengantin https://t.co/wjwvdPTRBa #vivanews"'	truk hantam kendara cianjur salah satu rombong pengantin
2	MUI Kalbar: Larangan Cadar-Celana Cingkrang Khawatir Timbulkan Gejolak https://t.co/SOouK1vBlP	mui kalbar larang cadar celana ci ngkrang khawatir timbul gejolak
3	@pengguna13 @pengguna00 Biasa gaya hidup sok...ibarat org udik baru melek liat metropolitan jd jumpalitan dan gayanya gak karu karuan....	biasa gaya hidup sok orang udik melek lihat metropolitan jadi ju mpalitan dan gayanya tidak karu karuan

4.3 Ekstraksi Fitur

Setelah melalui tahap *preprocessing*, data cuitan selanjutnya melalui tahap ekstraksi fitur. Tahap ini berguna untuk mempersiapkan data cuitan, agar bisa diproses pada tahap klasifikasi menggunakan algoritma *Multinomial Naïve Bayes*.

Langkah pertama yang dilakukan pada ekstraksi fitur adalah adalah merepresentasikan data-data cuitan ke dalam bentuk vektor dengan menggunakan *library* pada bahasa pemrograman *Python* yang bernama *CountVectorizer*. Contoh penerapan *library CountVectorizer* pada data cuitan seperti terlihat pada gambar 4.9.

Gambar 4.9 *CountVectorizer* pada data cuitan

Setiap baris pada gambar 4.9 merepresentasikan baris cuitan pada *dataset*, sedangkan tiap kolomnya merepresentasikan seluruh kata pada *dataset*. Agar lebih mudah dipahami, misalkan terdapat dua cuitan yang telah dikumpulkan:

(C1) "@dudu Kamu malas sekali, pantas saja kamu miskin"

(C2) "@molawsky Jangan malas dong, biar bisa cepat lulusnya"

Setelah dilakukan *preprocessing*, didapatkan tiga kata baku, yaitu 'malas', 'miskin', dan 'lulus'. Lalu, tiap-tiap baris cuitan direpresentasikan sebagai vektor, dan kata baku yang telah didapatkan tadi menjadi elemen pada baris cuitan dengan ketentuan, apabila pada baris cuitan terdapat kata yang bersangkutan, maka diberi nilai 1, sedangkan jika tidak ada diberi nilai 0.

Tabel 4.3 *Word Vector*

	Malas	**Miskin**	**Lulus**
(C1)	1	1	0
(C2)	1	0	1

Setelah terbentuknya *Word Vector*, maka langkah selanjutnya adalah melakukan pembobotan menggunakan metode *TF-IDF* (*Term Frequency-Inverse Document Frequency*). Proses pembobotan kata diawali dengan menghitung nilai *TF* (*Term Frequency*) terlebih dahulu yang merupakan frekuensi munculnya sebuah *term* atau kata baku yang telah didapatkan pada baris cuitan.

Tabel 4.4 Nilai *TF* masing-masing *term*

	(C1)	**(C2)**
Malas	1	1
Miskin	1	0
Lulus	0	1

Setelah nilai *TF* didapatkan, langkah selanjutnya adalah menentukan *DF* yaitu jumlah baris data cuitan yang mengandung kata baku (*term*). Langkah selanjutnya setelah nilai *DF* ditemukan, adalah menghitung nilai *IDF* dengan menghitung nilai *log* dari jumlah baris cuitan dibagi dengan nilai *DF*.

Tabel 4.5 Nilai *DF*

Term	***Document Frequency***
Malas	2
Miskin	1
Lulus	1

Setelah nilai *IDF* didapatkan, langkah terakhir ialah mengali hasil *TF* dengan *IDF* yang hasilnya dapat diuraikan pada tabel di bawah ini:

Tabel 4.6 Perhitungan dengan *TF-IDF*

Kata Baku	**TF**		**DF**	**D/DF**	**IDF**	**W=TF*(IDF+1)**	
	C1	**C2**				**C1**	**C2**
Malas	1	1	2	1	0	1	1
Miskin	1	0	1	2	0.3	1.3	0
Lulus	0	1	1	2	0.3	0	1.3
nilai bobot tiap dokumen						2.3	2.3

Nilai dari *IDF* dijumlahkan dengan 1 untuk mengatasi hasil 0 pada *IDF* karena adanya kesamaan jumlah antara *D* dan *DF*.

Adapun hasil pembobotan pada dua cuitan contoh dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil pembobotan dengan *TF-IDF*

	Malas	**Miskin**	**Lulus**
(C1)	1	1.3	0
(C2)	1	0	1.3

Pada penelitian ini, pembobotan menggunakan metode *TF-IDF* dilakukan dengan bantuan *library* dari bahasa pemrograman *python* yaitu *TfidfVectorizer*.

4.4 Klasifikasi

Algoritma *Multinomial Naïve Bayes* dipilih karena memiliki performa yang bagus dengan akurasi cukup tinggi dan waktu pemrosesan data yang cukup singkat. Hal ini dibuktikan dengan menguji *dataset* dengan empat algoritma yang berbeda, yakni *Multinomial Naïve Bayes, Logistic Regression, SVM* dengan *Linear Kernel,* dan *k-NN*.

Kode program yang digunakan untuk mengukur performa masing – masing algoritma dalam memproses data dapat dilihat pada gambar 4.10

```
#Multinomial Naive Bayes
pipeline_mnb = Pipeline([
                         ('vect', CountVectorizer()),
                         ('tfidf',          TfidfTransformer(use_idf=True,
smooth_idf=True)),
                         ('clf', MultinomialNB(alpha=1))
])

#Logistic regression
pipeline_logit = Pipeline([
                          ('vect',CountVectorizer()),
                          ('tfidf',TfidfTransformer(use_idf=True,
smooth_idf=True)),
                          ('clf',LogisticRegression())
])

#Linear SVC
pipeline_lsvc = Pipeline([
                         ('vect',CountVectorizer()),
                         ('tfidf',TfidfTransformer(use_idf=True,
smooth_idf=True)),
                         ('clf',LinearSVC())
])

#Bernoulli Naive Bayes
pipeline_bnb = Pipeline([
                         ('vect',CountVectorizer()),
                         ('tfidf',TfidfTransformer(use_idf=True,
smooth_idf=True)),
                         ('clf',BernoulliNB())
])
#k-NN
pipeline_nc = Pipeline([
                           ('vect', CountVectorizer()),
                           ('tfidf',TfidfTransformer(use_idf=True,
smooth_idf=True)),
                           ('clf', KNeighborsClassifier()),
                     ])

X_train, X_test, y_train, y_test = train_test_split(X, y,   test_size=0.2,
random_state = 0)
```

Gambar 4.10 Kode program pengujian algoritma

Hasil dari pengujian keempat algoritma dapat dilihat pada tabel

Tabel 4.8 Hasil pengujian algoritma

Algoritma	***Accuracy***	***Execution Time***
Multinomial Naïve Bayes	0.8026	80.7 ms
Logistic Regression	0.8205	157 ms
Linear SVM	0.8308	100 ms
k-NN	0.7551	69.9 ms

Pada tahapan klasifikasi, dilakukan implementasi algoritma *Multinomial Naïve Bayes* terhadap data-data cuitan yang telah terbobot pada tahap ekstraksi fitur sebelumnya. Tahap klasifikasi pada pengerjaan aplikasi ini dibantu oleh *library* bahasa pemrograman *Python* yang bernama *scikit-learn*.

Sebelum dilakukan proses klasifikasi, langkah pertama yang dilakukan adalah mengakses *file* data cuitan yang sudah melalui tahap *preprocessing* sebagai data *training* dengan menggunakan *library pandas*. Kode program yang digunakan dapat dilihat pada gambar 4.11.

```
import pandas as pd

data = dataset = pd.read_excel("cleandata.xlsx")

X = data['Tweet']
y = data['Label']
```

Gambar 4.11 Kode program pemanggilan data *training*

Setelah data *training* siap untuk digunakan, langkah selanjutnya adalah mengunduh dan memanggil *library* yang dibutuhkan, seperti yang terlihat pada kode program pada gambar 4.12.

```
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer, TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, f1_score, precision_score, recall_score
import nltk
```

Gambar 4.12 Penggunaan *library*

Langkah selanjutnya ialah melakukan ekstraksi fitur dan klasifikasi sekaligus dengan satu kelas *pipeline* seperti yang terlihat pada gambar 4.13. Data *test* yang digunakan pada tahap klasifikasi ini adalah 20% dari jumlah data *training* yang dipilih secara *random by system*.

```
pipeline_mnb = Pipeline([
                          ('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer(use_idf=True,
smooth_idf=True)),
                          ('clf', MultinomialNB(alpha=1))
])

X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.2,
random_state = 0)
```

Gambar 4.13 Implementasi ekstraksi fitur dan kasifikasi

Proses klasifikasi dengan algoritma *Multinomial Naïve Bayes* adalah dengan menghitung *probabilitas* tiap-tiap kalimat cuitan terhadap kelas/labelnya.

4.5 Uji dan Evaluasi Model

Tahap selanjutnya adalah melakukan uji dan evaluasi terhadap model dengan menggunakan *Confussion Matrix*. Implementasi kode program untuk menguji performa *Multinomial Naïve Bayes* dapat dilihat pada gambar 4.14.

```
pipeline_mnb.fit(X_train, y_train)
predictions_mnb = pipeline_mnb.predict(X_test)

print("Accuracy: {0:.4f}".format(accuracy_score(y_test, predictions_mnb)))
print("Confusion matrix: {}")
print(confusion_matrix(y_test, predictions_mnb))
print(classification_report(y_test, predictions_mnb))
```

Gambar 4.14 Kode tahap pengujian model

Pengujian dilakukan dengan melakukan klasifikasi sebanyak lima kali dengan sebaran data *testing* yang bervariasi. Variasi data *testing* didapatkan dengan memberi nilai berbeda pada variabel *random_state* yang terdapat pada gambar 4.13. Pada pengujian ini, penulis menggunakan nilai *random_state* yaitu 0 sampai 4. Hasil *confussion matrix* dari tiap-tiap sebaran data dapan dilihat pada tabel-tabel berikut.

Tabel 4.9 *Confusion Matrix random_state* 0

Multinomial Naïve Bayes			
Actual Value	**Predicted Value**		
		0	1
	0	292	86
	1	47	355

Tabel 4.10 *Confusion Matrix random_state* 1

Multinomial Naïve Bayes			
Actual Value	**Predicted Value**		
		0	1
	0	277	113
	1	41	349

Tabel 4.11 *Confusion Matrix random_state* 2

Multinomial Naïve Bayes			
Actual Value	**Predicted Value**		
		0	1
	0	286	96
	1	29	369

Tabel 4.12 *Confusion Matrix random_state* 3

Multinomial Naïve Bayes			
Actual Value	**Predicted Value**		
		0	1
	0	268	93
	1	50	369

Tabel 4.13 *Confusion Matrix random_state* 4

Multinomial Naïve Bayes			
Actual Value	**Predicted Value**		
		0	1
	0	274	106
	1	47	353

Setelah nilai-nilai pada *Confussion Matrix* diketahui, langkah selanjutnya adalah menghitung nilai *precission, accuracy, recall,* dan *F1-Score* dengan menggunakan persamaan pada subbab 2.8. *Precission* adalah ketepatan nilai antara permintaan pengguna dengan respon system, *accuracy* adalah perbandingan antara informasi benar yang dijawab sistem dengan keseluruhan data, *recall* adalah ketepatan antara informasi yang sama dengan informasi yang pernah dipanggil sebelumnya, *F1-Score* adalah perbandingan rata-rata presisi dan recall yang dibobotkan. Hasil dari pengujian model dapat dilihat pada tabel 4.14 dan tabel 4.15.

Tabel 4.14 Hasil pengujian model dengan *random_state* 0 sampai 2

	random_state								
	0			1			2		
	Precision	**Recal**	**F1-Score**	**Precision**	**Recal**	**F1-Score**	**Precision**	**Recal**	**F1-Score**
non-cyberbullying (0)	0.86	0.77	0.81	0.87	0.71	0.78	0.91	0.75	0.82
cyberbullying (1)	0.8	0.88	0.84	0.76	0.89	0.82	0.79	0.93	0.86
accuracy			0.82			0.80			0.83

Tabel 4.15 Hasil pengujian model dengan *random_state* 3 sampai 4

	random_state					
	3			4		
	Precision	**Recal**	**F1-Score**	**Precision**	**Recal**	**F1-Score**
non-cyberbullying (0)	0.84	0.74	0.79	0.85	0.72	0.78
cyberbullying (1)	0.8	0.88	0.84	0.77	0.88	0.82
accuracy			0.81			0.80

Berdasarkan data-data yang terdapat pada tabel 4.14 dan tabel 4.15, diperoleh hasil akhir dari pengujian model dengan menghitung nilai rata-rata tiap variabelnya seperti yang terlihat pada tabel 4.16.

Tabel 4.16 Rata-rata nilai evaluasi model

	Hasil Pengujian		
	Precision	**Recal**	**F1-Score**
non-cyberbullying (0)	0.866	0.738	0.796
cyberbullying (1)	0.784	0.892	0.836
accuracy			0.812

Nilai *accuracy* dari model menunjukkan angka yang cukup baik yaitu 0.812. Setelah model diuji dan menunjukkan performa yang cukup baik, langkah selanjutnya yang akan dilakukan adalah pembuatan aplikasi deteksi *cyberbullying* yang akan di bahas pada subbab selanjutnya.

4.6 Perancangan Aplikasi

Setelah model berhasil dibuat dan menunjukkan performa yang cukup baik, langkah selanjutnya yang dilakukan adalah pembuatan aplikasi deteksi *cyberbullying* dengan *framework Django*. Konsep dari pemrograman menggunakan *Django* ialah *Model, View, Template*. Seperti yang sudah dijelaskan pada bagian sebelumnya, *Model* berperan untuk menghubungkan aplikasi dengan basis data, *View* berperan untuk mengimplementasikan logika pemrograman, sedangkan *Template* memiliki peran yang berkaitan dengan antarmuka sistem.

Pada bagian *Template* pembuatan antarmuka dibantu dengan *framework Bootstrap*. Antarmuka aplikasi terdiri dari sebuah *textfield* untuk menerima masukan dari pengguna berupa kalimat. Setelah pengguna memasukkan kalimat, kalimat akan dideteksi sebagai kalimat *cyberbullying* atau *non-cyberbullying* dengan menekan tombol yang ada. Kode program pada bagian *Template* bisa dilihat pada gambar-gambar di bawah ini.

```
<!DOCTYPE html>
<html>
<head>
   <title></title>
</head>
<body>

</body>
</html>
```

Gambar 4.15 Struktur *HTML*

Tag head pada *HTML* biasanya berisikan *file* eksternal untuk mengatur keindahan tampilan *website* dan juga nama *website*. Kode program pada *tag head* seperti yang terlihat pada gambar 4.16.

```
<head>
    <link rel="canonical"
    href="https://getbootstrap.com/docs/4.3/examples/cover/">

    <!-- Bootstrap core CSS -->
    <link href="{% static 'css/bootstrap.min.css'%}" rel="stylesheet"
    crossorigin="anonymous">

    <!-- Fonts -->
    <link
    href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,70
    0" rel="stylesheet">

    <!-- Icons -->
    <link href="{% static 'js/plugins/nucleo/css/nucleo.css' %}"
    rel="stylesheet" />

    <link href="{% static 'js/plugins/@fortawesome/fontawesome-
    free/css/all.min.css' %}" rel="stylesheet" />

    <!-- CSS Files -->
    <link href="{% static 'css/argon-dashboard.css' %}" rel="stylesheet" />

    <!-- Custom styles for this template -->
    <link href="{% static 'sticky-footer-navbar.css' %}" rel="stylesheet">
</head>
```

Gambar 4.16 Kode program pada *tag head*

File eksternal yang dimuat pada *tag head* berupa berkas *CSS* yang berperan untuk mengatur estetika dari antarmuka aplikasi. Berkas *CSS* yang di maksud antara lain *bootstrap.min.css*, *nucleo.css*, *all.min.css*, *argon-dashboard.css*, dan *sticky-footer-navbar.css*. Contoh pemanggilan *file* dilakukan dengan menuliskan kode *HTML* seperti "<link href="{% static 'css/bootstrap.min.css'%}" rel="stylesheet" crossorigin="anonymous">" untuk memuat berkas *bootstrap.min.css.*

Bagian selanjutnya pada *HTML* adalah *tag body*. Pada bagian ini, segala elemen pada *website* diprogramkan, mulai dari *navigation bar*, *content*, dan *footer*. Kode program pada bagian *body* dapat dilihat pada gambar 4.17. Semua kode program *HTML* ini disimpan dalam satu *file* bernama *home.html*.

```
        <nav class="navbar navbar-top navbar-expand-md navbar-dark"
         id="navbar-main">
          <div class="container-fluid">
            <!-- Brand -->
            <h1 class="h4 mb-0 text-white text-uppercase d-none d-lg-inline-
            block">Deteksi Cyberbullying</h1>
            <!-- Form -->
            <form class="navbar-search navbar-search-dark form-inline mr-3
            d-none d-md-flex ml-lg-auto">
              <div class="form-group mb-0">
              </div>
            </form>
          </div>
        </nav>
```

Gambar 4.17 *Tag body* bagian *navigation bar*

```
<div class="row">
            <div class="col-xl-8 mb-5 mb-xl-4">
              <div class="card bg-gradient-default shadow">
                <div class="card-header bg-transparent">
                    <div class="col">
                      <h6   class="text-uppercase   text-light   ls-1   mb-
                       1">KALIMAT</h6>
                    </div>
                  <div class="row align-items-center">
                    <form    class="form-group    shadow-textarea    col-md-
                     12"style="float: left;" action="hasil" method="GET">
                       <textarea      class="form-control       z-depth-1"
                       id="exampleFormControlTextarea6"          rows="7"
                       placeholder="Write       something        here..."
                       name="teks"></textarea><br>
                       <input type="submit" name="kirim">
                    </form>
                  </div>
                </div>
              </div>
            </div>
            <div class="row col-xl-4 mb-xl-4">
              <div class="card shadow">
                <div class="card-header bg-transparent">
                  <div class="row align-items-center">
                    <div class="col">
                      <h6   class="text-uppercase   text-muted   ls-1   mb-
                       1">Hasil</h6>
                      <h2 class="mb-0">Makna Kalimat</h2>
                    </div>
                  </div>
                </div>
                <div class="card-body">
                  <div>
                      {% block content %}
                      <h1 class="ml5">
                        <span class="text-wrapper">
                          <span          class="letters          letters-
                          left">{{result}}</span>
                        </span>
                      </h1>
                      {% endblock %}
                  </div> </div>      </div>          </div>         </div>
```

Gambar 4.18 *Tag body* bagian *content*

```
          <footer class="footer">
            <div class="row align-items-center justify-content-xl-between">
              <div class="col-xl-6">
                <div class="copyright text-center text-xl-left text-muted">
                  &copy; 2019 <a href="https://www.creative-tim.com"
                  class="font-weight-bold ml-1" target="_blank">Nassharih
                  Abdulloh</a>
                </div>
              </div>
            </div>
          </footer>
```

Gambar 4.19 *Tag body* bagian *footer*

Tag div dan *nav* pada gambar 4.17, 4.18, dan 4.19 dituliskan untuk memuat berkas *CSS* yang sudah dideklarasikan pada *tag head* sebelumnya. Berkas *CSS* diakses dengan menuliskan kode *class=namaKelas* dan *id=namaId*. Seperti yang terlihat pada kode baris pertama gambar 4.17.

Pada bagian *View* dituliskan kode program yang kurang lebih sama pada bagian pembuatan model. Pada aplikasi deteksi *cyberbullying*, model yang sudah dibuat sebelumnya menjadi inti dari aplikasi. Kode program pada bagian *view* dapat dilihat pada gambar.

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return render(request,'home.html')
```

Gambar 4.20 Pemanggilan *library* dan pembuatan fungsi *index*

Langkah selanjutnya adalah membuat fungsi hasil, yang akan digunakan untuk mengeksekusi model yang sudah dibuat. Adapun kode programnya dapat dilihat pada gambar 4.21.

```
def hasil(request):

   return render()
```

Gambar 4.21 Struktur fungsi hasil

Pada fungsi hasil, dituliskan kode program yang mirip dengan kode program pada pembuatan model, mulai dari pendefinisian *library* hingga proses *carry data* ke bagian *Template*. Kode-kode program yang digunakan dapat dilihat pada gambar-gambar di bawah ini.

```
    from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer, TfidfVectorizer
    from sklearn.pipeline import Pipeline
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import LinearSVC
    from sklearn.naive_bayes import BernoulliNB
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, f1_score, precision_score, recall_score
    import nltk
    import pandas as pd
```

Gambar 4.22 Pendefinisian *library*

Setelah *library* terdefinisi, langkah selanjutnya yang dilakukan ialah pemanggilan *dataset*. Data yang dijadikan *dataset* adalah data-data cuitan yang telah melalui tahap *preprocessing*. Kode program untuk memanggil *dataset* dapat dilihat pada gambar 4.21.

```
    data = dataset = pd.read_excel("cleandata.xlsx")

    X = data['Tweet']
    y = data['Label']
```

Gambar 4.23 Pemanggilan *dataset*

Tahap selanjutnya adalah penggunaan algoritma *Naïve Bayes Clasifier* seperti yang digunakan pada tahap pembuatan model.

```
    pipeline_mnb = Pipeline([
                            ('vect', CountVectorizer()),
                            ('tfidf',       TfidfTransformer(use_idf=True,
smooth_idf=True)),
                            ('clf', MultinomialNB(alpha=1))
    ])
    X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.2,
random_state = 0)

    pipeline_mnb.fit(X_train, y_train)
    predictions_mnb = pipeline_mnb.predict(X_test)
```

Gambar 4.24 Implementasi algoritma *NBC*

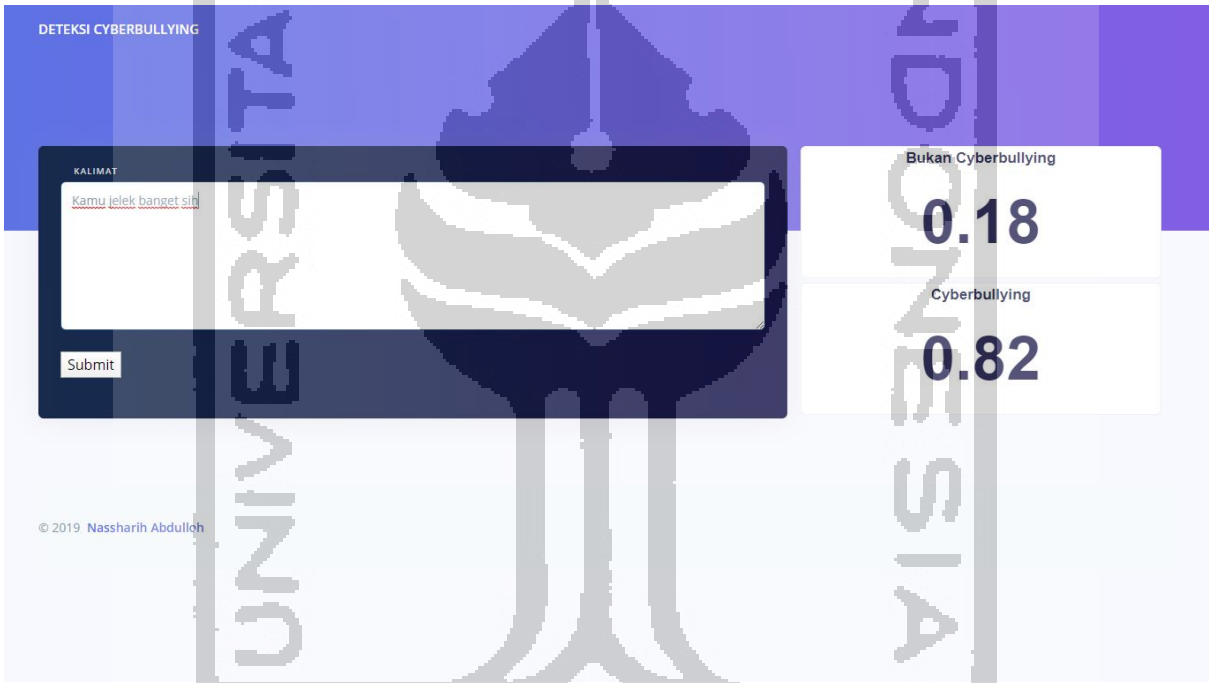
Selanjutnya ialah memproses keluaran aplikasi yang didapatkan dari masukan pengguna berupa sebuah kalimat, kalimat tersebut menjadi data *testing* untuk model, dan kemudian membawa hasil pengujian menjadi keluaran aplikasi yang akan ditampilkan pada halaman *home.html*.

Tahap akhhir dari pembuatan aplikasi ini adalah melakukan *routing* pada *fle urls.py* yang terdapat dalam *directory* Aplikasi Deteksi *Cyberbullying*. Tahap ini dilakukan untuk mengarahkan dan menghubungkan antara bagian *Template* dan *View*.

4.7 Uji dan Evaluasi Aplikasi

Tahapan terakhir dari pembuatan aplikasi ini adalah tahap pengujian dan evaluasi aplikasi. Pengujian pada aplikasi ini menggunakan metode *black box testing* dengan menguji performa aplikasi berdasarkan masukan pengguna dan keluaran aplikasi. Pengujian dengan metode ini cukup baik karena dapat menemukan celah atau kekurangan aplikasi secara langsung dengan mengoperasikan aplikasi.

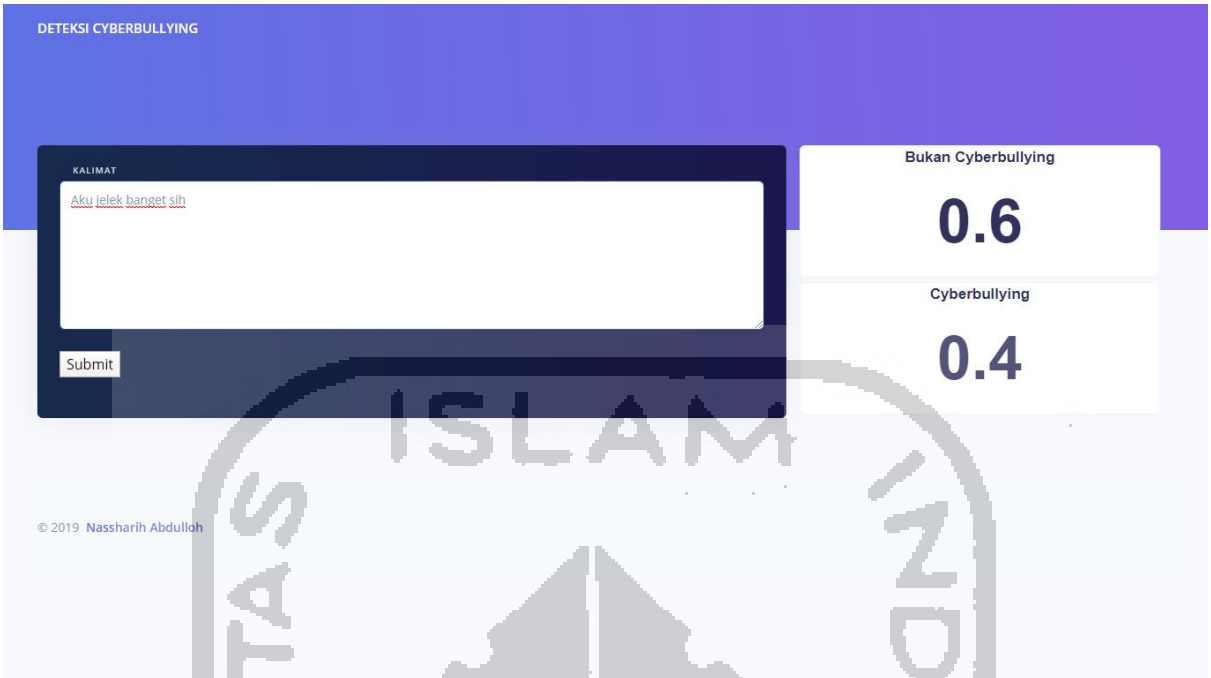
Pengujian pertama dilakukan dengan menguji aplikasi dengan kalimat masukan "Kamu jelek banget sih". Hasil pengujian dapat dilihat pada gambar 4.25.



Gambar 4.25 Pengujian aplikasi pertama

Kalimat "Kamu jelek banget sih" dikategorikan sebagai kalimat bermakna *cyberbullying* karena bersifat *offensive* dan ditujukan kepada orang lain dengan probabilitas pada kelas *cyberbullying* sebesar 0.82.

Pengujian kedua dilakukan dengan menuliskan kalimat "Aku jelek banget sih". Hasil pengujian dengan kalimat ini dapat dilihat pada gambar 4.26.



Gambar 4.26 Pengujian aplikasi kedua

Kalimat "Aku jelek banget sih" dikategorikan sebagai kalimat dengan makna *non-cyberbullying* karena mengarah kepada diri pribadi dan tidak memenuhi definisi dari *cyberbullying* dari Rahayu et al. (Rahayu, 2012).

Berikut adalah tabel yang berisi tentang daftar kalimat yang menjadi kalimat uji pada aplikasi deteksi *cyberbullying* beserta dengan hasil identifikasi maknanya.

Tabel 4.17 Pengujian aplikasi

Kalimat	Makna
Kamu jelek sekali sih	*Cyberbullying*
Aku jelek banget sih	Bukan *cyberbullying*
semua orang bilang kamu itu bodoh, gausah ngeles lagi deh	*Cyberbullying*
Kok aku masih bodoh ya?	Bukan *cyberbullying*
Jadi manusia itu gaboleh sombong ya	Bukan *cyberbullying*
Kamu tuh sombong banget	*Cyberbullying*
Apasih, kamu kampungan banget	*Cyberbullying*
Udah miskin belagu lagi, kamu tuh harus sadar jadi orang	*Cyberbullying*
Kamu sekarang gendutan, tapi jadi makin kece	Bukan *cyberbullying*
Mobil doang yang bagus, kamu mah jelek haha	Bukan *cyberbullying*

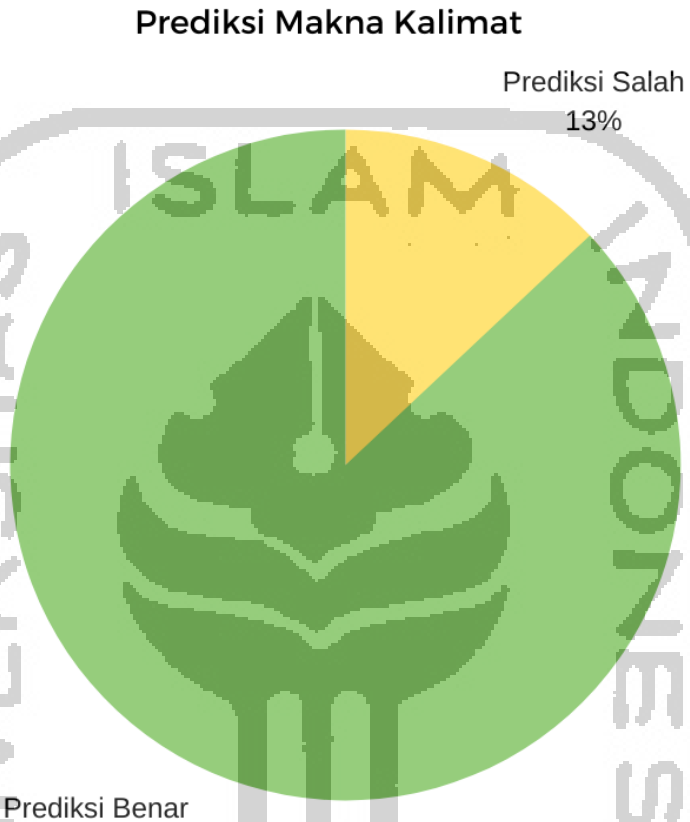
Perut kamu mulai membuncit, pantesan jomblo	Bukan *cyberbullying*
Semua orang tau kamu itu miskin, makanya jangan sosoan	*Cyberbullying*
Kok kamu baik banget sih	Bukan *cyberbullying*
Jangan sok, kamu juga miskin :)	*Cyberbullying*
Dih sombong banget, padahal muka pas pasan, duit sering minjem	*Cyberbullying*

Dari total 15 baris kalimat uji yang digunakan, dua diantaranya menghasilkan prediksi yang keliru. Kalimat dengan prediksi yang salah adalah "Mobil doang yang bagus, kamu mah jelek haha", yang seharusnya diprediksi sebagai kalimat *cyberbullying* namun aplikasi memprediksi hal sebaliknya. Kalimat dengan prediksi yang salah lainnya adalah "Perut kamu mulai membuncit, pantesan jomblo" yang juga harusnya diprediksi sebagai kalimat *cyberbullying* namun aplikasi juga memprediksi hal sebaliknya.

Kalimat	**Makna**
Kamu jelek sekali sih	*Cyberbullying*
Aku jelek banget sih	Bukan *cyberbullying*
semua orang bilang kamu itu bodoh, gausah ngeles lagi deh	*Cyberbullying*
Kok aku masih bodoh ya?	Bukan *cyberbullying*
Jadi manusia itu gaboleh sombong ya	Bukan *cyberbullying*
Kamu tuh sombong banget	*Cyberbullying*
Apasih, kamu kampungan banget	*Cyberbullying*
Udah miskin belagu lagi, kamu tuh harus sadar jadi orang	*Cyberbullying*
Kamu sekarang gendutan, tapi jadi makin kece	Bukan *cyberbullying*
Mobil doang yang bagus, kamu mah jelek haha	**Bukan *cyberbullying***
Perut kamu mulai membuncit, pantesan jomblo	**Bukan *cyberbullying***
Semua orang tau kamu itu miskin, makanya jangan sosoan	*Cyberbullying*
Kok kamu baik banget sih	Bukan *cyberbullying*
Jangan sok, kamu juga miskin :)	*Cyberbullying*
Dih sombong banget, padahal muka pas pasan, duit sering minjem	*Cyberbullying*

Gambar 4.27 Kalimat uji dengan prediksi keliru

Berdasarkan kalimat - kalimat uji yang sudah dicoba pada aplikasi, hanya terdapat 2 kalimat dari total 15 kalimat uji yang memberikan prediksi salah. Dengan hasil uji yang telah dilakukan, persentase performa aplikasi dapat dilihat pada gambar 4.28.



Gambar 4.28 Performa Aplikasi Deteksi *Cyberbullying*