**4. IMPLEMENTASI DAN PENGUJIAN**

Bab ini membahas hasil yang didapatkan dari implementasi algoritma *Support Vector Machine* dan pengujian data sesuai dengan metode analisis dan perancangan pada bab 3.

* 1. **Implementasi**

Berdasarkan hasil penelitian yang telah dilakukan pada bab sebelumnya, maka dilakukan implementasi tentang bagaimana cara melakukan langkah – langkah yang dijelaskan pada bab sebelumnya.

* 1. **Implementasi Pre-*Processing* *Data***

Tweet yang sudah diekstraksi dan di tamping kedalam text file kemudian dilakukan proses pre-*processing* untuk mengubah data tweet ke bentuk tertentu supaya memudahkan penelitian. Tweet yang telah diekstraksi dari twitter yang dilakukan pada bab sebelumnya akan diolah dengan menggunakan bahasa *python*.

* + 1. **Implementasi *Cleansing Data***

Data *tweets* yang sudah diambil seringkali mengandung banyak komponen kata, huruf atau karakter yang tidak diperlukan, sehingga diperlukan adanya penghapusan komponen tersebut supaya menghasilkan data yang lebih bersih untuk proses lebih lanjut. Secara detail, proses cleansing *tweet* yaitu :

1. Menghapus URL pada tweet
2. Menghapus hashtag
3. Menghapus RT
4. Menghapus @
5. Menghapus Double Tweet
6. Melakukan Case Folding

Penghapusan berberapa komponen diatas dilakukan dengan menggunakan bahasa *python*. Dengan bahasa *python*, mesin akan menghapus komponen yang disebutkan diatas pada setiap tweet yang ada pada text file. Potongan source code yang menjelaskan *cleansing* dapat dibawah ini:

def clean(idx, text):

text = text.strip()

try:

text = text.split('%d\_efw7912jdqw['%idx)[1]

except:

pass

text = text.split(']\_932ruo')

if len(text) == 2:

idx += 1

text = text[0]

text = re.sub(r"http\S+", '', text) (4.1)

text = re.sub(r"#\S+", '', text) (4.2)

text = re.sub(r"RT @[^\s]+:", '', text) (4.3)

text = re.sub(r"@", '', text) (4.4)

text = re.sub(r"[^0-9a-zA-Z ]+", '', text) (4.5)

text = re.sub(r"RT dan reply.\*$", '', text).lower().rstrip() (4.6)

return idx, text

Source code diatas menunjukan proses *cleansing* pada penelitian ini. Fungsi text.split berfungsi untuk memisah kata dengan parameter tertentu (berisi huruf yang ingin di pisahkan). Sedangkan fungsi re.sub() pada variable text berfungsi untuk memotong dan menggabungkan kata dengan memasukan parameter kata yang ingin dihapus. Source (4.1) berfungsi untuk menghapus url, (4.2) berfungsi untuk menghapus hastag, (4.3) berfungsi menghapus RT, (4.4) berfungsi menghapus et (@) , (4.5) berfungsi untuk menghilangkan karakter selain 0-9 dan alphabet, dan source (4.6) berfungsi untuk menghapus RT,Reply dan melakukan *case folding.*

* + 1. **Implementasi *Tokenizing***

Tahapan *Tokenizing* adalah proses yang melakukan pememecahan kalimat dalam sebuah teks menjadi kumpulan kata-kata. *Source* *code* untuk proses *tokenizing* dapat dilihat pada Source (4.7).

token = []

with open(“tokenizer.txt”) as t:

lines = t.readlines()

for line in lines: line.split(“ “)[0]

(4.7)

Proses diatas memecah kalimat pada *tweet* menjadi kata. Hal ini dilakukan supaya proses pelabelan dengan *lexicon based* dapat dilakukan. Pada proses *tokenizing* kalimat dipecah menjadi bentuk per-kata. Karakter spasi dihilangkan karena dianggap sebagai delimiter atau pemisah antar kata.

* + 1. **Implementasi *Stopword Removal***

Sesuai dengan yang dijelaskan pada bab sebelumnya, *stopword Removal* adalah proses penghapusan daftar kata yang tidak memiliki makna. Langkah ini dilakukan untuk mengurangi jumlah kata yang disimpan oleh sistem. Yang harus dilakukan adalah membuat *list stopword* dari daftar kata yang ada kemudian dilakukan penghapusan *stopword*.

Penghapusan *stopword* dilakukan dengan menghilangkan kata *stopword* pada kalimat *tweet*. Proses penghapusan *stopword* dilakukan dengan *koding* pada (4.8), kata-kata yang terdapat pada list *stopwords* dihilangkan dari kalimat data *tweet*.

remover = StopWordRemoverFactory().create\_stop\_word\_remover()

tokener = spacy.blank("id")

sw = open("stopword3.txt", 'r')

\_sw = sw.readlines()

for i, s in enumerate(\_sw):

\_sw[i] = s.strip()

remover.dictionary.add\_words(\_sw)

(4.8)

Kata yang dihapus adalah kata yang tidak mempunyai arti banyak, oleh karenanya dilakukan penghapusan. Kata yang terdapat pada *stopword* list akan otomatis dihilangkan dan membuat *tweet* lebih bersih.

* + 1. **Implementasi *Stemming***

Pada taap ini dilakukan proses *stemming*, yakni mengubah kata yang berimbuhan menjadi kata dasarnya. Proses *stemming* dilakukan dengan koding yang dapat dilihat pada (4.9). Hasil yang didapatkan dengan melakukan proses stemming adalah pengubahan data tweet menjadi kata dasar. proses *stemming* adalah sebagai berikut:

import Sastrawi.Stemmer.StemmerFactory as st

import pandas as pd

stemmer = st.StemmerFactory()

stemmer = stemmer.create\_stemmer()

df = pd.read\_csv("kpu1.csv")

result\_df.to\_csv("kpu1\_hasil.csv")

(4.9)

Kode program diatas membuat kata di dalam *variable* text akan ter-stem menjadi kata dasar. Data tweet yang telah melalui proses *stemming* kemudian siap untuk dilanjutkan ke proses selanjutnya, yaitu klasifikasi *lexicon* *base*.

* + 1. **Implementasi Klasifikasi dengan *Lexicon Based Features***

Pada tahap ini, dilakukan klasifikasi dengan *lexicon* *based* *features* untuk melakukan pelabelan terhadap *tweet*. *Tweet* akan di klasifikasikan menjadi *tweet* negatif atau positif dengan menggunakan rumus yang sama seperti pada (2.22) dan (2.23). Apabila rumus diatas menghasilkan nilai sentimen lebih besar dari 0, maka *tweet* akan digolongkan sebagai *tweet* positif. Sebaliknya, apabila nilai sentimen lebih kecil dari 0, maka *tweet* akan digolongkan sebagai *tweet* negatif. Potongan kode program yang menjelaskan proses klasifikasi *lexicon based* dapat dilihat pada (4.10)

scores = []

for ws in tqdm(words):

wss = ws.split()

sc = 0

i = 0

for w in wss:

try:

sc += kata\_kata[w]

i += 1

except:

continue

if i != 0:

scores.append(sc/i)

else:

scores.append(0.0)

(4.10)

Jika hasil sentimen netral seperti diatas, maka *tweet* akan dihapus dan tidak akan dimasukan kedalam dataset training karena penelitian ini hanya mengklasifikasikan data kedalam 2 kelas.

* 1. **Implementasi TF-IDF**

Pada tahap ini akan dilakukan perhitungan *Term Frequency – Inverse Document Frequency* (TF-IDF). TF-IDF dilakukan supaya nilai dari bobot dapat dimasukan ke perhitungan SVM, proses TF-IDF dapat dilihat pada potongan program (4.11).

vectorizer = TfidfVectorizer(min\_df=0.01, max\_df=3.0)

model = GridSearchCV(SVC(),

{'gamma':[1, 0.1, 0.001, 0.0001, 0.00001],

'kernel':['linear']},

refit=True, verbose=0)

(4.11)

* 1. **Implementasi Klasifikasi SVM**

Langkah selanjutnya adalah *training* dan *testing*. *Training* adalah proses mesin svm untuk mempelajari dan *testing* adalah proses prediksi yang dilakukan oleh mesin svm setelah proses *training*. Proses Hasil prediksi dapat mengandung bias ataupun kesalahan. Klasifikasi dengan SVM menggunakan 80% data latih (*training)* dan 20% data uji (*testing*) menggunakan library *python* sklearn dengan *kernel* *linear*.

Tahapan yang dilakukan untuk melakukan klasifikasi adalah pengumpulan data kemudian *preprocessing* data, klasifikasi *lexicon* base untuk melakukan pelabelan *tweet* dan proses pembobotan dokumen akan diklasifikasi oleh *Support Vector Machine* (SVM) untuk membuat model yang bersifat predict. Data yang digunakan di dalam penelitian ini adalah sebanyak 1000 data yang dikategorikan kelas sentimen positif dan kelas sentimen negatif oleh *lexicon* *base*. Data yang sudah dinormalisasi dan dianotasi sebelum dimasukan ke mesin SVM, data dibagi menjadi dua bagian yaitu, data latih (*training*) dan data uji (*testing*). Pengujian akan dilakukan dengan menggunakan 80% data latih dan 20% data uji. Potongan kode program untuk training data dapat dilihat pada (4.12) dan testing pada (4.13).

words\_f = vectorizer.fit\_transform(raw\_documents=words)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(words\_f, scores,

test\_size=0.2, shuffle=True)

model.fit(x\_train, y\_train)

(4.12)

model = model.best\_estimator\_

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_test)

report = classification\_report(y\_test, y\_pred, output\_dict=True)

(4.13)

## Hasil output dari klasifikasi svm berupa akurasi, presisi, recall dan f1-score.

* 1. **Pengujian**

Pada tahap ini dilakukan pengujian dengan menggunakan data latih dan data uji dengan perbandingan 80:20 [12]. Hasil dari klasifikasi menunjukan nilai akurasi sebesar 80.5%, presisi sebesar 74%, recall sebesar 77%, dan f1-score sebesar 75.4%. Perhitungan untuk menghitung akurasi, presisi dan recall, dilakukan dengan menggunakan *confusion* *matrix*. *confusion* *matrix* pada penelitian ini dapat dilihat pada tabel 4.1.

**Tabel 4.1** Implementasi *Confusion* *Matrix*

|  |  |  |
| --- | --- | --- |
| Prediction | True Values | |
| True | False |
| True | TP  62 | FP  18 |
| False | FN  21 | TN  99 |

Dengan nilai *confusion* *matrix* diatas maka, dapat dilakukan perhitungan Akurasi, Presisi dan Recall dengan perhitungan sebagai berikut:

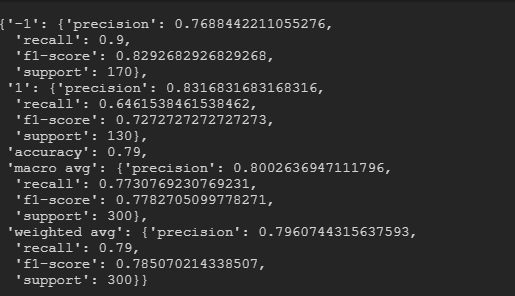
Akurasi = \* 100 = 80.5%

Presisi = \* 100 = 74%

Recall = \* 100 = 77%

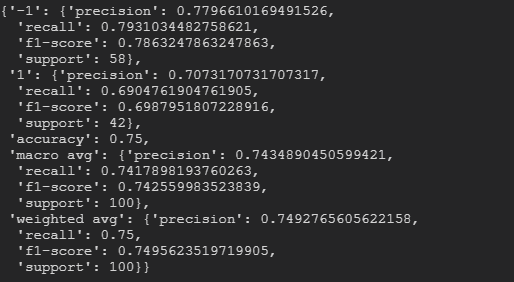
F1 Score = 2\* = 75.4%

Dengan perhitungan diatas, telah diketahui hasil dari akurasi adalah 80.5%, presisi 74%, recall 77% dan F1 Score 75.4%. digunakannya komposisi perbandingan 80:20 adalah karena 80:20 merupakan hasil dengan akurasi terbaik dibandingkan dengan komposisi 70:30 seperti pada gambar 4.1 dan 90:10 seperti pada gambar 4.2. hasil dari komposisi 80:20 dapat dilihat pada histogram pengujian dapat pada gambar 4.3.



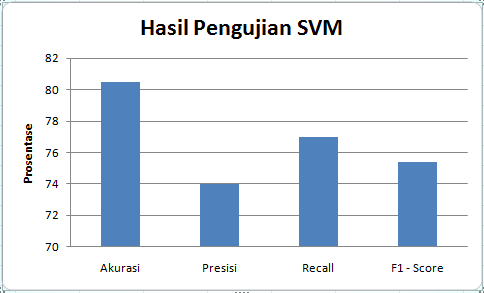
**Gambar 4.1** Komposisi 70:30

Dari gambar 4.1 dapat diambil kesimpulan bahwa akurasi yang didapat dari penelitian menggunakan komposisi 70% data training : 30% data uji menghasilkan akurasi sebesar 0.79 atau 79% dan nilai ini lebih kecil dari hasil komposisi 80 : 20.



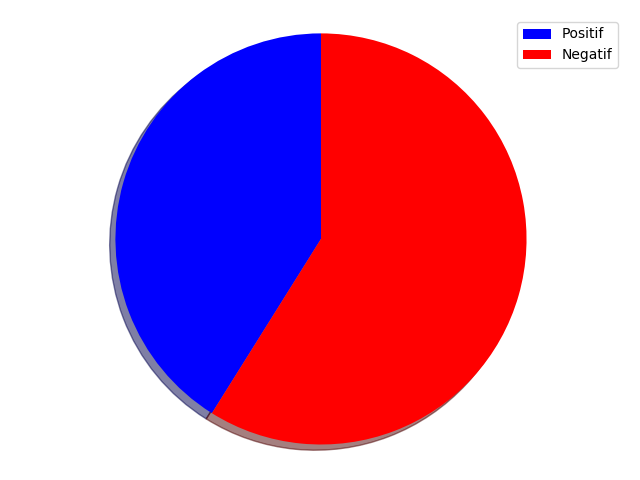
**Gambar 4.2** Komposisi 90 : 10

Dari gambar 4.2 dapat diambil kesimpulan bahwa akurasi yang didapat dari penelitian menggunakan komposisi 90% data training : 10% data uji menghasilkan akurasi sebesar 0.75 atau 75% dan nilai ini lebih kecil dari hasil komposisi 80 : 20. Karena hasil dari komposisi 80% : 20% adalah yang terbaik, maka komposisi tersebut yang digunakan di dalam penelitian ini.



**Gambar 4.3** Hasil SVM

Penelitian ini mengklasikan 1000 data tweet ke dalam 2 kelas, 589(58.9%) kedalam kelas negatif, dan 411(41.1%) ke dalam kelas negatif dengan menggunakan lexicon based features. Diagram pie untuk prosentase sentimen pada tweet dapat dilihat pada gambar 4.4.



**Gambar 4.4** Prosentase Kalimat Sentimen

* 1. **Implementasi Tampilan Website**

Pada proses ini dilakukan implementasi tampilan website sesuai dengan rancangan yang dibuat pada bab sebelumnya. Hasil dari penelitian akan ditampilkan pada sebuah website dengan alamat url [http://kpuanalysissentiment.epizy.com](http://kpuanalysissentiment.epizy.com/). Website ini berfungsi untuk menampilkan hasil penelitian dalam bentuk tabel dan diagram. Website ini dibuat dengan menggunakan framework bahasa php yaitu Codeigniter versi terbaru.

* + 1. **Halaman Utama Website**

Halaman utama website yang menampilkan hasil penelitian dapat dilihat pada gambar 4.5.



**Gambar 4.5** Halaman Awal Website

Pada gambar 4.5 menunjukan tampilan utama website sentimen analisis kpu dengan berberapa menu diantaranya, Home, *Tweet*, *Tweet* Positif, *Tweet* Negatif, *Pie* *Chart* dan *Histogram*.

* + 1. **Tampilan *Tweet***

Pada halaman *tweet* akan ditunjukan seluruh *tweet* beserta score sentimen yang digunakan dalam penelitian ini. Halaman *tweet* dapat dilihat pada gambar 4.6.



**Gambar 4.6** Halaman Tweet

* + 1. **Tampilan *Tweet* Positif dan Negatif**

Kedua halaman ini dikhususkan untuk melihat *tweet* yang bernilai sentimen positif dan tweet yang bernilai negatif. Untuk halaman *tweet* positif dapat dilihat pada gambar 4.7. dan untuk halaman *tweet* negatif dapat dilihat pada gambar 4.8.



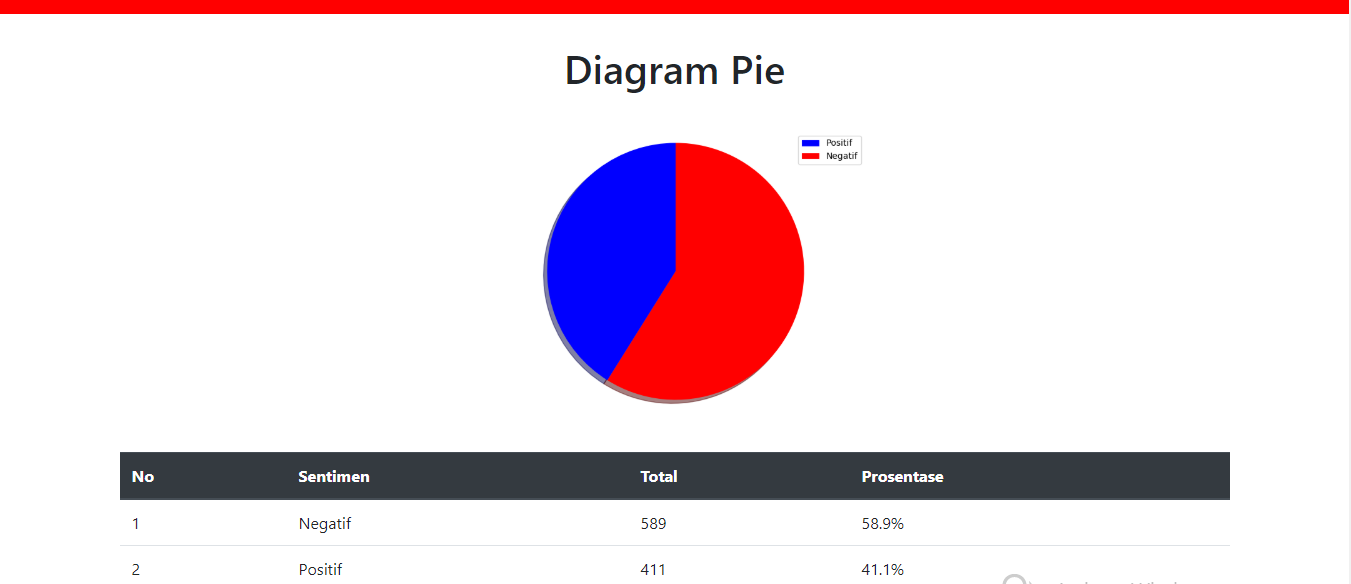
**Gambar 4.7** Halaman Tweet Positif



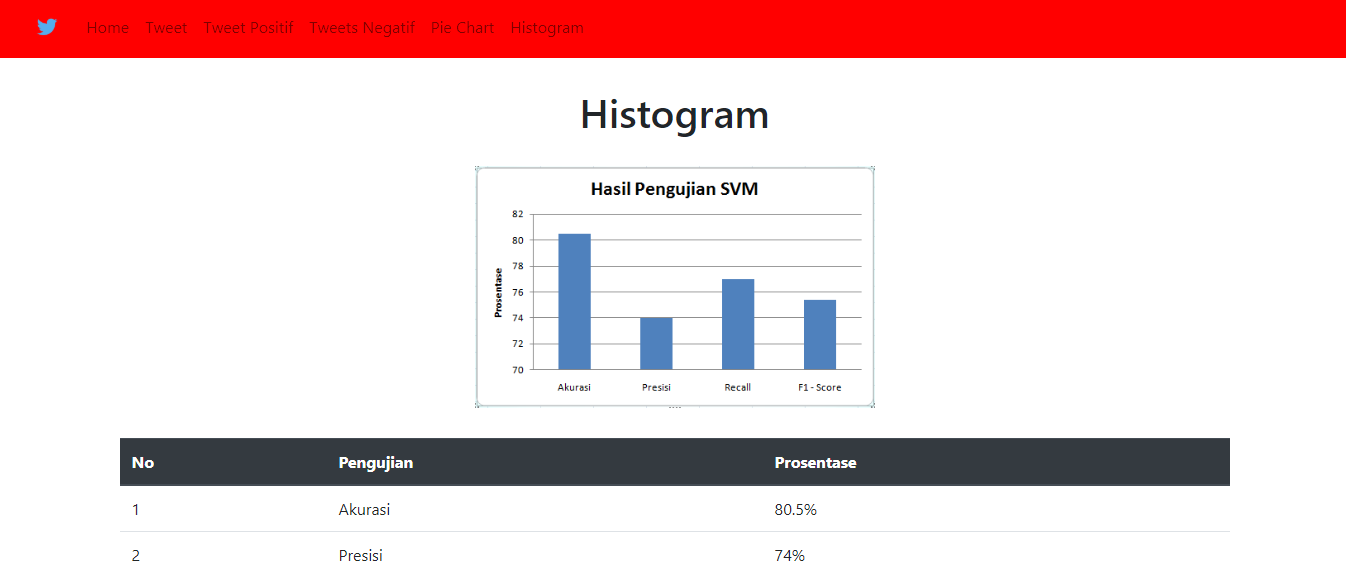
**Gambar 4.8** Halaman Tweet Negatif

* + 1. **Tampilan Diagram *Pie* dan *Histogram***

Kedua halaman ini digunakan untuk melihat diagram *pie* dan histogram. Diagram *pie* adalah hasil dari perbandingan sentimen positif dan negatif sedangkan histogram menunjukan hasil prosentase pengujian. Tampilan halaman website dapat dilihat pada gambar 4.9 untuk diagram pie dan gambar 4.10 untuk *histogram*.



**Gambar 4.9** Tampilan Halaman Pie

****

**Gambar 4.10** Tampilan Halaman Histogram