# BAB IV

# HASIL DAN PEMBAHASAN

## Lingkungan Percobaan

Agar aplikasi yang telah dikembangkan dapat berjalan dengan semestinya, dibutuhkan perangkat dengan spesifikasi tertentu, adapun dalam penelitian ini menggunakan spesifikasi perangkat diantaranya.

### Spesifikasi perangkat keras

Daftar perangkat keras yang mendukung aplikasi ini untuk berjalan dengan baik adalah sebagai berikut:

#### Processor : Intel(R) Core(TM) i3 CPU M 380 @ 2.53GHz

#### RAM : 2,00 GB

#### Harddisk : 500 GB

#### VGA : Intel(R) HD Graphics

### Spesifikasi perangkat lunak

Daftar perangkat lunak yang mendukung aplikasi ini untuk berjalan dengan baik adalah sebagai berikut:

#### Sistem Operasi : Windows 7 Professional

#### Bahasa Program Utama : Python 3.8 (32-bit)

#### IDE : Visual Studio Code v1.52.1

#### DBMS : MySQL Database

#### Browser : Google Chrome

#### Lainya : XAMPP v7.3.9, Ms. Excel 2013

## Implementasi Metode

Implementasi metode dalam penelitian ini dilakukan dengan enam (6) tahapan utama. Tahapan utama tersebut diproses secara berurutan, tahapan utama yang dimaksud antara lain: Tahap pengumpulan data, tahap *preprocessing*, tahap *labeling,* tahap pembagian data,tahap ekstraksi fitur *CountVectorizer* dan tahap klasifikasi *K-Nearest Neighbor*.

### Tahap pengumpulan data

Berdasarkan sub bab (3. 1), *dataset* penelitian bersumber dari media sosial Twitter berupa data teks kicauan (*tweet)*. *Dataset* tersebut diperoleh secara rutin melalui proses *crawling* menggunakan pustaka Tweepy, dimulai pada tanggal 1 Desember 2020 hingga 31 Desember 2020, sehingga diperoleh total *dataset* sejumlah 4.314 data (*tweet*). Kata kunci yang digunakan antara lain: ‘pembelajaran jarak jauh’, ‘belajar dari rumah’, ‘#belajaronline’, ‘#belajardarirumah’, ‘#belajardirumah’, dan ‘#kuliahonline’.

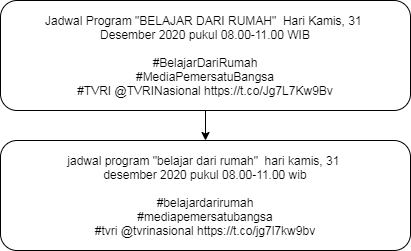
*Dataset* yang berhasil dikumpulkan melalui proses *crawling* dengan informasi antara lain: *tweet* *id*, *full*\_*text* (*tweet*), *created\_at*, *user*.*screen\_name* (*username*), akan disimpan ke dalam file *excel (*.xlsx*)*, yang kemudian akan dimasukkan ke dalam basis data (*database*) MySQL untuk tahap selanjutnya (*preprocessing*).

### Tahap *preprocessing*

Tahap *preprocessing* merupakan tahapan yang hanya dapat dilakukan setelah tersedianya satu atau lebih *dataset* pada basis data (*database*) hasil dari tahapan pengumpulan data. Berdasarkan pada sub sub bab (3. 2. 2), tahapan ini terdiri atas lima (5) proses utama antara lain*: case folding, cleansing,* mengubah *slang word,* menghapus *stop word,* dan *stemming.* Berikut penjabaran dari tahap *preprocessing:*

#### Case folding

Pada Gambar 4.1 proses *case folding* dilakukan penyetaraan teks menjadi huruf kecil secara keseluruhan, misalnya: ‘Jadwal’ akan diubah menjadi ‘jadwal’, ‘BELAJAR DARI RUMAH’ akan diubah menjadi ‘belajar dari rumah’, dan seterusnya.



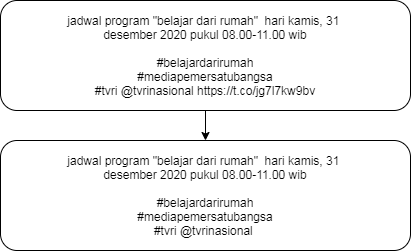
**Gambar 14.1 Proses case folding**

#### Cleansing

Pada proses *cleansing* dilakukan penyaringan dan pembuangan teks. Proses *cleansing* terdiri atas beberapa tahapan antara lain: menghapus URL, *mention* (*@mention*), *hastag* (*#hastag*), selain huruf (a-z) dan spasi berlebih.

##### Menghapus URL

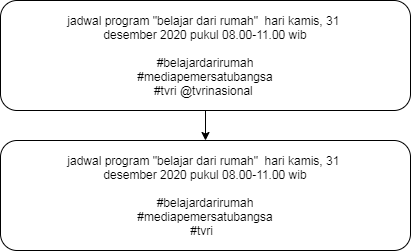
Pada Gambar 4.2 proses penghapusan URL akan menghapus semua teks yang diawali dengan ‘http’.



**Gambar 24.2 Proses menghapus URL**

##### Menghapus *mention* (*@mention*)

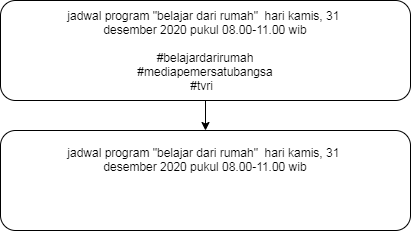
Pada Gambar 4.3 proses penghapusan *mention* (*@mention*) akan menghapus semua teks yang diawali dengan ‘@’.



**Gambar 34.3 Proses menghapus mention**

##### Menghapus *hastag* (*#hastag*)

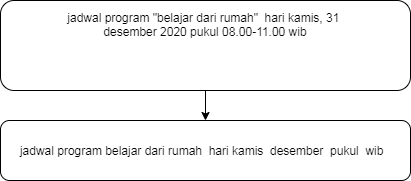
Pada Gambar 4.4 proses penghapusan tagar atau *hastag* (#hastag) akan menghapus semua teks yang diawali dengan ‘#’.



**Gambar 44.4 Proses menghapus hastag**

##### Menghapus selain huruf

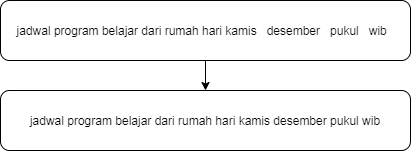
Pada Gambar 4.5 proses penghapusan selain huruf akan menghapus karakter selain huruf a-z pada teks, misalnya: ’31 desember 2020’ menjadi ‘ desember ’.



**Gambar 54.5 Proses menghapus selain huruf**

##### Menghapus spasi berlebih

Pada Gambar 4.6 proses penghapusan spasi atau (*whitespace)* berlebih akan menghapus *whitespace* berlebih pada teks, misalnya: ‘pukul wib’ menjadi ‘pukul wib’.



**Gambar 64.6 Proses menghapus spasi berlebih**

#### Merubah slang word

Pada Gambar 4.7 proses merubah *slang word* akan merubah setiap kata gaul, kata singkatan atau kata tidak baku ke bentuk bakunya, misalnya: ‘utk’ menjadi ‘untuk’, ‘yng’ menjadi ‘yang’ dan seterusnya. Proses pengubahan tersebut melibatkan kamus *slang word* yang terdapat dalam basis data (*database*).



**Gambar 74.7 Proses merubah slang word**

#### Menghapus stop word

Pada Gambar 4.8 proses menghapus *stop word* akan menghapus setiap kata yang kurang memiliki makna namun sering dijumpai dalam sebuah teks, misalnya kata: ‘untuk’ ‘yang’, dan seterusnya. Proses penghapusan tersebut melibatkan kamus *stop word* yang terdapat dalam basis data (*database*).



**Gambar 84.8 Proses menghapus stop word**

#### Stemming

Pada Gambar 4.9 proses *stemming* akan mengubah kata berimbuhan menjadi kata dasar dengan melibatkan pustaka Sastrawi, misalnya kata: ‘membantu’ menjadi ‘bantu’, ‘pembelajaran’ menjadi ‘ajar’, dan seterusnya.



**Gambar 94.9 Proses stemming**

Dalam penelitian ini, tahap *preprocessing* akan mengolah dan menghasilkan 4.314 data (*tweet*) yang lebih terstruktur atau disebut dengan *clean text,* yang kemudian akan disimpan ke dalam basis data (*database*) untuk tahap selanjutnya (*labeling*).

### Tahap *labeling*

Tahap *labeling* merupakan tahapan yang hanya dapat dilakukan setelah tersedianya satu atau lebih data *clean text* pada basis data (*database*) hasil dari tahapan *preprocessing*. Berdasarkan sub sub bab (3. 2. 3), tahap *labeling* utama dilakukan dengan cara menggunakan kamus sentimen, berdasarkan sub bab (2. 6), tahap *labeling* dengan kamus sentimen dilakukan dengan proses perhitungan skor sentimen dan pemberian kelas sentimen. Proses perhitungan skor sentimen pada sebuah *tweet* dapat dilihat pada Tabel 4.1 berikut:

**Tabel 14.1 Proses perhitungan skor**

| ***Dataset (clean text)*** | **Kata Positif** | **Kata Negatif** |
| --- | --- | --- |
| semangat teman teman ikut aktif bantu dalam proses ajar jarak jauh | semangat (1)  teman (2)  ikut (1)  aktif (1)  bantu (1)  proses (1)  ajar (1) | semangat (1)  jauh (1) |
| **Jumlah** | 8 | 2 |

Berdasarkan Tabel 4.1, menggunakan persamaan (2.1) maka dapat diperoleh perhitungan skor untuk *tweet* ‘semangat teman teman ikut aktif bantu dalam proses ajar jarak jauh’ yaitu sebagai berikut:

skor = (jumlah kata positif) – (jumlah kata negatif)

skor = 8 – 2

skor = 6

Setelah diketahui nilai skor, proses selanjutnya adalah pemberian kelas sentimen berdasarkan aturan yang telah dijelaskan pada sub sub bab (2. 6. 2) sebagai berikut:

if skor > 0:

kelas = 'positif'

elif skor < 0:

kelas = 'negatif'

else:

continue

Maka dapat disimpulkan bahwa *tweet* ‘semangat teman teman ikut aktif bantu dalam proses ajar jarak jauh’ akan mendapatkan kelas positif, karena nilai skor > 0.

Dalam penelitian ini, diperoleh 3.954 *tweet* berlabel. Sebanyak 3.733 *tweet* berlabel tersebut diperoleh menggunakan cara *labeling* kamus sentimen secara otomatis, sementara 221 *tweet* lainya diperoleh dengan *labeling* secara manual dengan melibatkan subjektifitas beberapa orang (empat orang), dengan mengabaikan *tweet* yang dinilai sebagai netral.

### Tahap pembagian data

Tahap pembagian data merupakan tahapan yang dilakukan setelah tersedianya satu atau lebih data (*tweet*)berlabel pada basis data (*database*) hasil dari tahapan *labeling*. Berdasarkan sub sub bab (3. 2. 4), *tweet* berlabel akan dibagi menjadi dua (2) bagian, antara lain data uji dan data latih. Pembagian data dilakukan menggunakan rasio yang telah ditentukan yaitu 1:9 (data uji : data latih) atau 10% data uji, 90% data latih.

Dalam penelitian ini, tahap pembagian data dilakukan terhadap 3.954 *tweet* berlabel. Dengan rasio data 1:9, maka dapat diketahui jumlah data uji yang diperoleh sebanyak 395 *tweet* berlabel, sementara jumlah data latih yang diperoleh sebanyak 3.559 *tweet* berlabel. *Tweet* berlabel tersebut kemudian akan disimpan ke dalam basis data (*database*) untuk tahap selanjutnya (ekstraksi fitur).

### Tahap ekstraksi fitur *CountVectorizer*

Tahap ekstraksi fitur menggunakan *CountVectorizer* (*modeling*) merupakan tahapan yang dilakukan setelah *tweet* melalui proses *preprocessing*, *labeling*, dan pembagian data. Tahapan ini bertujuan untuk memperoleh *model* latih atau pengetahuan melalui data latih yang ada. Berdasarkan sub sub bab (3. 2. 5), tahap ini terdapat lima (5) proses utama antara lain: seleksi data latih, pembuatan list kata, pencarian fitur kata, pembuatan vektor kosong dan membuat representasi vektor. Berikut penjabaran dari tahap *modeling*:

#### Seleksi data latih

Seleksi data latih dilakukan setelah data melalui proses *preprocessing*, *labeling*, dan pembagian data. Menggunakan teknik sampling kuota (*quota sampling*) seperti yang telah dijelaskan pada sub bab (2. 8) dan sub-sub bab (3. 2. 5), tahap pertama dalam *modeling* adalah pengambilan sampel dari populasi data latih untuk dijadikan sebagai pengetahuan berdasarkan kriteria tertentu, kriteria yang dimaksud adalah dengan menyamakan jumlah antara data berlabel positif dengan data berlabel negatif. Sampel data latih yang diambil dapat dilihat pada Tabel 4.2 berikut:

**Tabel 24.2 Sampel data latih**

| *Tweet* (*Tlatih-i*) | *Clean Text* | *Sentiment Type* |
| --- | --- | --- |
| latih-1 | ajar efektif kelas pintar semangat gratis | positif |
| latih-2 | pagi tetap semangat ajar aktivitas rabu pintar ayo simak jadwal acara | positif |
| latih-3 | pelita bangsa tengah pandemi covid bangkit semangat wujud merdeka ajar | positif |
| latih-4 | susah sulit kerja tugas bingung tanya tanya kelas pintar akibat covid | negatif |
| latih-5 | covid ajar jarak jauh sulit didik tugas banyak | negatif |
| latih-6 | pagi susah kerja lama lama ajar jarak jauh penuh drama | negatif |

Sampel data latih pada Tabel 4.2 terdapat tiga (3) kolom yaitu: *Tweet* (*Tlatih-i*) yang berarti urutan *Tweet* latih ke-i, *Clean* *Text* yang berarti teks *tweet* yang telah terstruktur setelah melalui proses *preprocessing*, dan *Sentiment Type* yang berarti jenis kategori (*label*) *tweet* yang diperoleh setelah melalui proses *labeling*.

#### Pembuatan list kata

Dari sampel data latih pada Tabel 4.2 kemudian akan dipisahkan menjadi satuan kata. Pemisahan menjadi kata dilakukan berdasarkan spasi (*whitespace*), kemudian hasilnya akan ditampung dalam sebuah wadah *list*. Hasil proses ini dapat dilihat pada Tabel 4.3 berikut:

**Tabel 34.3 List kata**

| *List* kata |
| --- |
| ['ajar', 'efektif', 'kelas', 'pintar', 'semangat', 'gratis', 'pagi', 'tetap', 'semangat', 'ajar', 'aktivitas', 'rabu', 'pintar', 'ayo', 'simak', 'jadwal', 'acara', 'pelita', 'bangsa', 'tengah', 'pandemi', 'covid', 'bangkit', 'semangat', 'wujud', 'merdeka', 'ajar', 'susah', 'sulit', 'kerja', 'tugas', 'bingung', 'tanya', 'tanya', 'kelas', 'pintar', 'akibat', 'covid', 'covid', 'ajar', 'jarak', 'jauh', 'sulit', 'didik', 'tugas', 'banyak', 'pagi', 'susah', 'kerja', 'lama', 'lama', 'ajar', 'jarak', 'jauh', 'penuh', 'drama'] |

*List* kata pada Tabel 4.3 merupakan hasil dari proses pemisahan kata dari kolom *clean text* pada *tweet* berdasarkan pada Tabel 4.2 Sampel data latih.

#### Pencarian fitur kata

Pencarian fitur kata dilakukan dengan cara melakukan pencarian dan pendataan setiap kata unik (*unique*) dengan membuang kata duplikat yang terdapat dalam *list* kata. Sehingga diperoleh *list* fitur kata seperti pada Tabel 4.4 berikut:

**Tabel 44.4 Fitur kata**

| Fiturkata |
| --- |
| ['ajar', 'efektif', 'kelas', 'pintar', 'semangat', 'gratis', 'pagi', 'tetap', 'aktivitas', 'rabu', 'ayo', 'simak', 'jadwal', 'acara', 'pelita', 'bangsa', 'tengah', 'pandemi', 'covid', 'bangkit', 'wujud', 'merdeka', 'susah', 'sulit', 'kerja', 'tugas', 'bingung', 'tanya', 'akibat', 'jarak', 'jauh', 'didik', 'banyak', 'lama', 'penuh', 'drama'] |

Fitur kata pada Tabel 4.4 merupakan hasil dari proses pencarian fitur berdasarkan pada Tabel 4.3 *list* kata.

#### Membuat vektor kosong latih

Membuat vektor kosong dimaksudkan untuk menyiapkan wadah berbentuk vektor dengan isian nilai awal yaitu angka (*integer*) nol (0). Wadah vektor tersebut dibentuk dengan panjang berdasarkan jumlah fitur kata. Berdasarkan Tabel 4.2 Sampel data latih dan Tabel 4.4 Fitur kata, maka vektor kosong yang dihasilkan seperti pada Tabel 4.5 berikut:

**Tabel 54.5 Vektor kosong latih**

| *Tweet*  *(Tlatih-i*) | Vektor Kosong |
| --- | --- |
| latih-1 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-2 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-3 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-4 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-5 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-6 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |

#### Membuat vektor kata latih

Proses membuat vektor kata dilakukan dengan pengubahan nilai pada vektor kosong berdasarkan frekuensi kemunculan fitur pada tiap kata dalam *tweet*. Nilai representasi vektor diperoleh berdasarkan jumlah kemunculan fitur dalam *tweet*. Berdasarkan pada Tabel 4.4 Fitur kata dan Tabel 4.5 Vektor kosong, maka vektor kata yang dihasilkan seperti pada Tabel 4.6 berikut:

**Tabel 64.6 Representasi vektor latih**

| *Tweet* *(Tlatih-i*) | Representasi vektor |
| --- | --- |
| latih-1 | [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-2 | [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-3 | [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| latih-4 | [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0] |
| latih-5 | [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0] |
| latih-6 | [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1] |

Representasi vektor pada Tabel 4.6 Representasi vektor latih merupakan hasil akhir dari tahap *modeling* menggunakan ekstraksi fitur *CountVectorizer*.

Sebelum beralih ke tahap selanjutnya (tahap klasifikasi), representasi vektor dan atribut lain yang dihasilkan pada tahap *modeling* akan disimpan ke dalam sebuah wadah berbentuk file dengan format JSON (*.*json). File JSON tersebut digunakan untuk menampung *model* latih seperti pada Tabel 4.7 berikut:

**Tabel 74.7 File JSON model latih**

| *key* | *value* |
| --- | --- |
| *teks\_list* | [ 'ajar efektif kelas pintar semangat gratis', 'pagi tetap semangat ajar aktivitas rabu pintar ayo simak jadwal acara', 'pelita bangsa tengah pandemi covid bangkit semangat wujud merdeka ajar', 'susah sulit kerja tugas bingung tanya tanya kelas pintar akibat covid', 'covid ajar jarak jauh sulit didik tugas banyak', 'pagi susah kerja lama lama ajar jarak jauh penuh drama' ] |
| *label\_list* | ['positif', 'positif', 'positif', 'negatif', 'negatif', 'negatif" ] |
| *feature\_list* | ['ajar', 'efektif', 'kelas', 'pintar', 'semangat', 'gratis', 'pagi', 'tetap', 'aktivitas', 'rabu', 'ayo', 'simak', 'jadwal', 'acara', 'pelita', 'bangsa', 'tengah', 'pandemi', 'covid', 'bangkit', 'wujud', 'merdeka', 'susah', 'sulit', 'kerja', 'tugas', 'bingung', 'tanya', 'akibat', 'jarak', 'jauh', 'didik', 'banyak', 'lama', 'penuh', 'drama'] |
| *vector\_list* | [  [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0],  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0],  [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1]  ] |

Pada Tabel 4.7 terdapat empat (4) *key* yaitu: *teks\_list* yang berisi nilai pada kolom *Clean Text* dari *Tweet* latih ke-i pada Tabel 4.2 Sampel data latih, *label\_list* yang berisi nilai pada kolom *Sentiment Type* dari *Tweet* ke-i pada Tabel 4.2 secara berurutan, *feature\_list* yang berisi daftar fitur yang diperoleh dari Tabel 4.4 Fitur kata, *vector\_list* yang berisi nilai representasi vektor dari Tabel 4.6 Representasi vektor secara berurutan berdasarkan *Tweet* latih ke-i pada Tabel 4.2.

Dalam t ini, dari sejumlah 3.559 data latih, hanya sebanyak 1500 data latih berlabel yang dipilih menggunakan teknik *quota sampling*, dengan ciri: 750 *tweet* latih positif dan 750 *tweet* latih negatif. *Tweet* tersebut kemudian akan dijadikan sebuah pengetahuan (*model* latih) dan disimpan ke dalam sebuah file dengan format JSON (*.*json) untuk tahap selanjutnya (klasifikasi).

### Tahap klasifikasi *K-Nearest Neighbor*

Tahap klasifikasi menggunakan *K-Nearest Neighbor* (KNN) merupakan tahapan yang dilakukan setelah tahapekstraksi fitur (*modeling*). Tahapan ini bertujuan untuk memprediksikan *label* untuk setiap data uji berdasarkan *model* latih yang dihasilkan pada tahap *modeling*. Berikut penjabaran dari tahap klasifikasi KNN:

#### Persiapan data

Proses persiapan data merupakan proses pemilihan nilai K tetangga terdekat dan *model* latih. *Model* latih yang terpilih selanjutnya akan dijadikan sebagai landasan dalam melakukan klasifikasi untuk data uji yang tersedia. Pada tahap klasifikasi dalam penulisan ini, *model* latih yang dipilih merupakan *model* latih hasil dari sub-sub bab (4. 2. 1), sementara untuk data uji akan digunakan adalah sampel data uji seperti pada Tabel 4.8 berikut:

**Tabel 84.8 Sampel data uji**

| *Tweet* *(Tuji-i*) | *Clean Text* | *Sentiment Type* |
| --- | --- | --- |
| uji-1 | semangat ikut kelas pintar ajar jarak jauh tengah pandemi | positif |
| uji-2 | susah sulit ajar jarak jauh pandemi covid covid tetap semangat | negatif |

#### Membuat representasi vektor uji

Pembuatan representasi vektor uji menggunakan pengetahuan yang bersumber dari *model* latih. Pembuatan representasi vektor uji ini terdiri atas dua (2) proses antara lain: membuat vektor kosong dan membuat vektor kata.

##### Membuat vektor kosong uji

Dalam proses ini akan dibuat wadah vektor kosong seperti yang dijelaskan pada sub-sub bab (4. 2. 1) bagian d, vektor kosong dibuat berdasarkan pada *feature\_list* (*model* latih) dan jumlah data uji. Berdasarkan jumlah fitur pada *model* latih dan Tabel 4.8 Sampel data uji, maka vektor kosong akan dibuat seperti pada Tabel 4.9 berikut:

**Tabel 94.9 Vektor kosong uji**

| *Tweet* (*T*uji-*i*) | Vektor Kosong |
| --- | --- |
| uji-1 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| uji-2 | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |

##### Membuat vektor kata uji

Dalam proses ini akan dibuat representasi vektor untuk setiap data seperti yang dijelaskan pada sub-sub bab (4. 2. 1) bagian e, vektor kata dibuat berdasarkan jumlah kemunculan *feature\_list* (*model* latih) dengan tiap kata dalam *tweet* data uji. Maka vektor kata akan dibuat seperti pada Tabel 4.10 berikut:

**Tabel 104.10 Representasi vektor uji**

| *Tweet* (*T*uji-*i*) | Representasi vektor |
| --- | --- |
| uji-1 | [1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0] |
| uji-2 | [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0] |

#### Menghitung jarak antar data

Perhitungan jarak dilakukan menggunakan data vektor uji (Tabel 4.10 Representasi vektor uji) dan data *vector\_list* pada *model* latih. Berdasarkan pada sub bab (2. 9), proses perhitungan jarak melibatkan *euclidean distance* dengan persamaan (2. 2). Berikut contoh penerapan *euclidean distance* (*d*)dalam menghitung jarak pada vektor uji-1 (*x*) dengan vektor latih-1 (*y*):

Berdasarkan contoh sebelumnya, maka hasil perhitungan untuk setiap jarak antara vektor uji dengan vektor latih adalah seperti dalam Tabel 4.11 berikut:

**Tabel 114.11 Hasil jarak euclidean distance**

| *Tweet* (*T*uji-*i*) | *Tweet* (*Tlatih-i*) | *Euclidean Distance*  ( *d(uji-i, latih-i)* ) |
| --- | --- | --- |
| uji-1 | latih-1 |  |
| latih-2 |  |
| latih-3 |  |
| latih-4 |  |
| latih-5 |  |
| latih-6 |  |
|  |  |  |
| uji-2 | latih-1 |  |
| latih-2 |  |
| latih-3 |  |
| latih-4 |  |
| latih-5 |  |
| latih-6 |  |

#### Mencari tetangga terdekat

Proses pencarian tetangga terdekat melibatkan nilai K. Nilai K dalam *K-nearest neighbor* (KNN) merupakan jumlah data ketetanggaan terdekat yang hendak diperoleh. Dalam penelitian ini nilai K yang dapat digunakan telah ditentukan, yaitu: K=3, K=5, K=7, K=9, dan K=11. Sementara pada penulisan ini nilai K yang dipilih adalah K=3.

Proses pencarian tetangga terdekat dilakukan dengan melalui dua (2) proses, antara lain: mengurutkan nilai jarak dan mengambil K data tetangga terdekat.

##### Mengurutkan nilai jarak

Dalam proses ini, nilai dari Tabel 4.11 Hasil jarak *euclidean distance* akan diurutkan secara urut menaik (*ascending*) berdasarkan jarak. Sehingga hasil urutan dapat diperoleh seperti pada Tabel 4.12 berikut:

**Tabel 124.12 Pengurutan jarak tetangga**

| Urutan | Jarak  ( *d(uji-i, latih-i)* ) | *Tweet*  (*Tuji-i, latih-i*) |
| --- | --- | --- |
| 1 | 2.449489742783178 | uji-1, latih-1 |
| 2 | 3.1622776601683795 | uji-1, latih-3 |
| 3 | 3.1622776601683795 | uji-1, latih-5 |
| 4 | 3.605551275463989 | uji-1, latih-2 |
| 5 | 3.7416573867739413 | uji-1, latih-6 |
| 6 | 4.123105625617661 | uji-1, latih-4 |
|  |  |  |
| 1 | 2.8284271247461903 | uji-2, latih-5 |
| 2 | 3.4641016151377544 | uji-2, latih-3 |
| 3 | 3.7416573867739413 | uji-2, latih-1 |
| 4 | 4.0 | uji-2, latih-6 |
| 5 | 4.123105625617661 | uji-2, latih-2 |
| 6 | 4.123105625617661 | uji-2, latih-4 |

##### Mengambil K data tetangga terdekat

Setelah melalui proses pengurutan, data dari Tabel 4.12 Pengurutan jarak tetangga akan diambil sebanyak K buah data, dengan nilai K yang telah ditentukan dan dipilih. Dengan nilai K=3, sehingga diperoleh hasil tetangga terdekat seperti Tabel 4.13 berikut:

**Tabel 134.13 Data K tetangga terdekat**

| Urutan | *Tweet*  (*Tuji-i, latih-i*) | Jarak  ( *d(uji-i, latih-i)* ) |
| --- | --- | --- |
| 1 | uji-1, latih-1 | 2.449489742783178 |
| 2 | uji-1, latih-3 | 3.1622776601683795 |
| 3 | uji-1, latih-5 | 3.1622776601683795 |
|  |  |  |
| 1 | uji-2, latih-5 | 2.8284271247461903 |
| 2 | uji-2, latih-3 | 3.4641016151377544 |
| 3 | uji-2, latih-1 | 3.7416573867739413 |

#### Menghitung nilai probabilitas

Nilai probabilitas diperoleh dengan cara melihat probabilitas *label* yang muncul pada data K tetangga terdekat. Nilai probabilitas yang dicari adalah nilai probabilitas *tweet* uji akan berlabel positif dan nilai probabilitas *tweet* uji akan berlabel negatif. Hal tersebut dapat diketahui melalui *label\_list* pada *model* latih dan Tabel 4.13 Data K tetangga terdekat, bahwa nilai probabilitas yang dihasilkan adalah seperti pada Tabel 4.14 berikut:

**Tabel 144.14 Nilai probabilitas data uji**

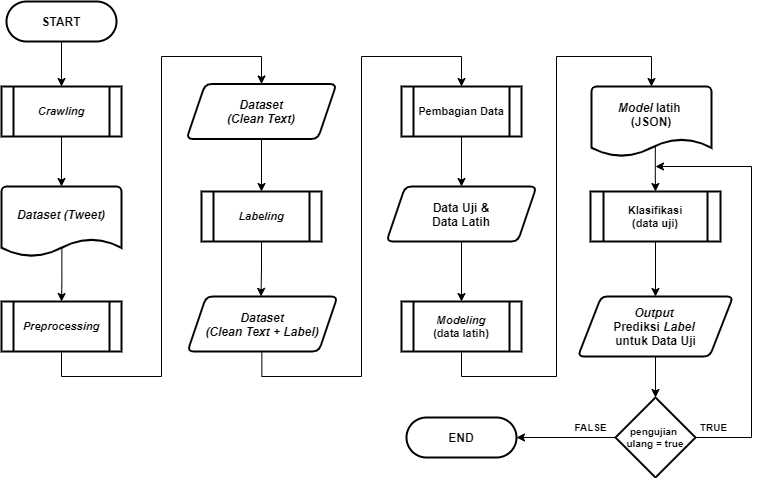
| *Tweet* (*T*uji-*i*) | *Tweet*  (*T latih-i*) | *Sentiment Type* | Probabilitas positif | Probabilitas negatif |
| --- | --- | --- | --- | --- |
| uji-1 | latih-1 | positif | 1 | 0 |
| latih-3 | positif | 1 | 0 |
| latih-5 | negatif | 0 | 1 |
| Jumlah | | | 2 (0.667) | 1 (0.333) |
|  |  |  |  |  |
| uji-2 | latih-5 | negatif | 0 | 1 |
| latih-3 | positif | 1 | 0 |
| latih-1 | positif | 1 | 0 |
| Jumlah | | | 2 (0.667) | 1 (0.333) |

Berdasarkan Tabel 4.14 Nilai probabilitas data uji, dapat diketahui dengan K=3, pada pengujian dengan *tweet* uji-1 dan *tweet* uji-2 keduanya akan sama-sama diprediksikan berlabel positif dengan nilai probabilitas yang sama yaitu 0.667 atau 66.7%.

## *Flowchart* Tahapan Metode

*Flowchart* merupakan suatu bagan atau simbol yang menggambarkan alur kerja atau urutan proses pada suatu program. Berikut adalah penjabaran *flowchart* dalam penelitian ini:

### *Flowchart* keseluruhan proses sistem

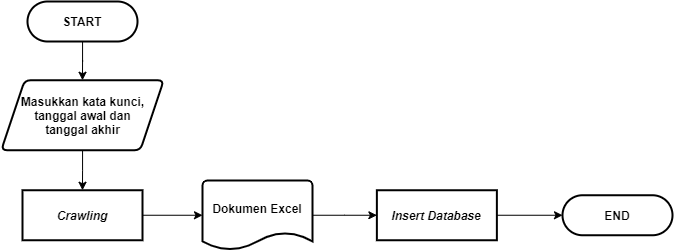


**Gambar 104.10 Flowchart keseluruhan proses sistem**

Pada Gambar 4.10 *Flowchart* keseluruhan proses sistem, menjelaskan proses-proses yang dilalui sistem yang dibuat. Dimulai dari tahap *crawling* sehingga menghasilkan *dataset* berupa *tweet*, kemudian tahap *preprocessing* untuk menghasilkan kolom *clean text*, selanjutnya tahap *labeling* untuk memberikan *label* berupa positif atau negatif, hasil tahap *labeling* akan menghasilkan kolom *label*, kemudian pembagian data untuk membagi *dataset* berlabel antara data uji dan data latih berdasarkan rasio 1:9, lalu tahap *modeling* menggunakan data latih berlabel untuk menghasilkan sebuah *model* latih, yang kemudian akan diuji beberapa kali dengan nilai variasi nilai K yang berbeda pada tahap klasifikasi. Hasil dari tahapan tersebut akan berupa *label* sentimen prediksi untuk setiap data uji.

### *Flowchart* proses *crawling*

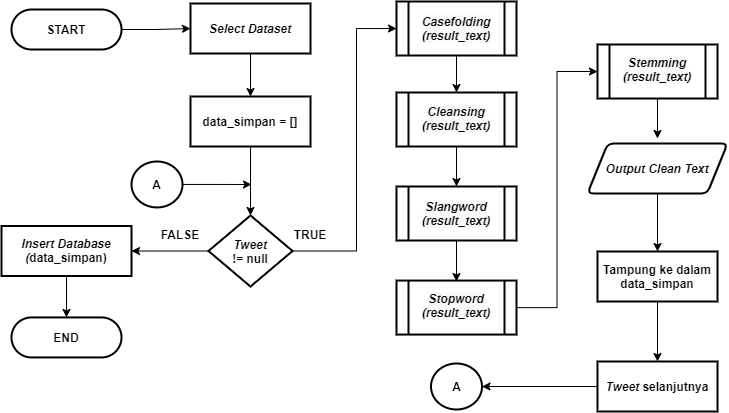
Pada *flowchart* ini, menjelaskan proses pengumpulan data atau *crawling* data tweet dimulai dari memasukkan kata kunci dan tanggal awal dan tanggal akhir, cara lainya adalah dengan fitur import file *excel*, kemudian dilakukan pencarian *tweet* berdasarkan parameter menggunakan pustaka Tweepy. Hasil pengumpulan data akan disimpan dalam bentuk file Excel (.xlsx) sebelum kemudian dimasukkan ke dalam *database*. *Flowchart* proses *crawling* dapat dilihat pada Gambar 4.11 berikut:



**Gambar 114.11 Flowchart proses crawling**

### *Flowchart* proses *preprocessing*

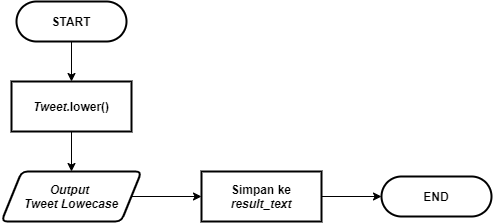
Pada *flowchart* ini, menjelaskan proses pengubahan data pengubahan data teks *tweet* menjadi terstruktur atau setara, guna mendukung proses klasifikasi agar berjalan dengan baik. Hasil proses ini berupa teks bersih, yang kemudian akan disimpan ke dalam *database*  untuk proses selanjutnya. *Flowchart* proses *preprocessing* dapat dilihat pada Gambar 4.12 berikut:



**Gambar 124.12 Flowchart proses preprocessing**

#### Flowchart proses case folding

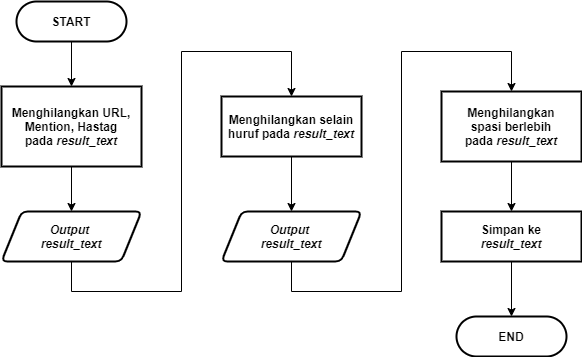
Pada *flowchart* ini dilakukan proses pengubahan teks *tweet* menjadi huruf kecil secara keseluruhan. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result\_text* untuk proses selanjutnya. *Flowchart* proses *case folding* dapat dilihat pada Gambar 4.13 berikut:



**Gambar 134.13 Flowchart proses case folding**

#### Flowchart proses cleansing

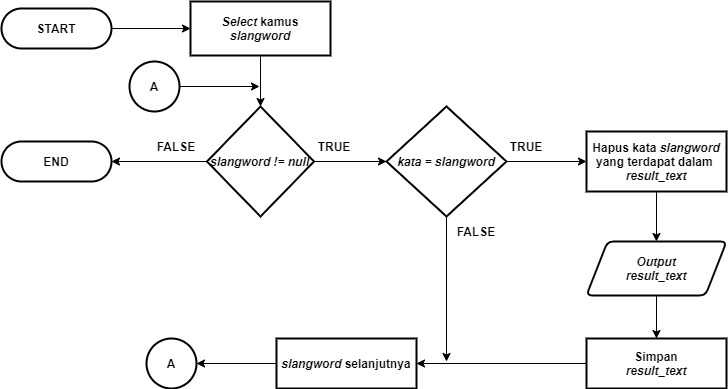
Pada *flowchart* ini dilakukan proses penyaringan dan pembuangan atribut pada teks *result*\_*text*, diantaranya penghapusan atau penghilangan URL, *mention*, *hastag*, selain huruf dan spasi atau baris berlebih. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *cleansing* dapat dilihat pada Gambar 4.14 berikut:



**Gambar 144.14 Flowchart proses cleansing**

#### Flowchart proses slang word

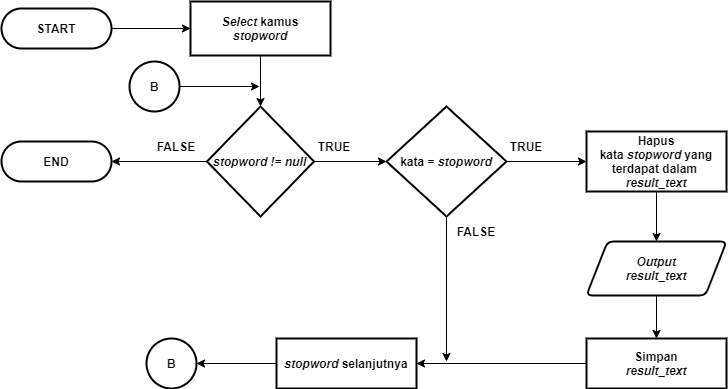
Pada *flowchart* ini dilakukan perubahan atribut pada teks *result*\_*text* yang mengandung kata *slang* dari kamus *slang word* ke bentuk kata asli atau kata bakunya. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *slang word* dapat dilihat pada Gambar 4.15 berikut:



**Gambar 154.15 Flowchart proses slang word**

#### Flowchart proses stop word

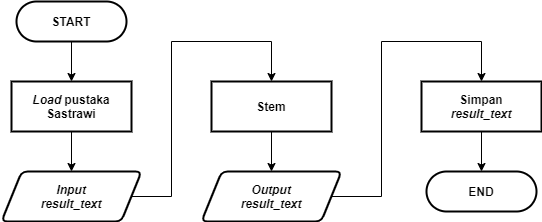
Pada *flowchart* ini dilakukan proses penghapusan atribut pada teks *result*\_*text* yang mengandung *stop word* dari kamus *stop word* karena dianggap kurang memiliki makna untuk proses klasifikasi. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *stop word* dapat dilihat pada Gambar 4.16 berikut:



**Gambar 164.16 Flowchart proses stop word**

#### Flowchart proses stemming

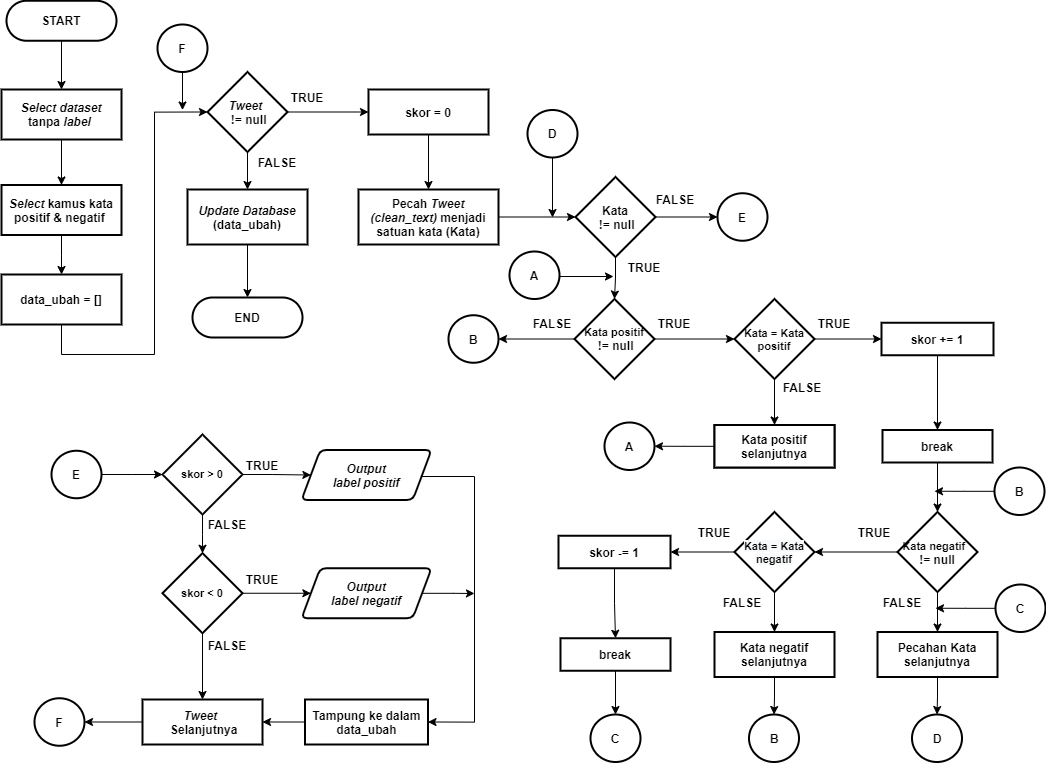
Pada *flowchart* ini dilakukan perubahan atribut pada teks *result*\_*text* yang mengandung kataberimbuhan ke bentuk kata asalnya, menggunakan pustaka Sastrawi. *Flowchart* proses *stemming* dapat dilihat pada Gambar 4.17 berikut:



**Gambar 174.17 Flowchart proses stemming**

### *Flowchart* proses *labeling*

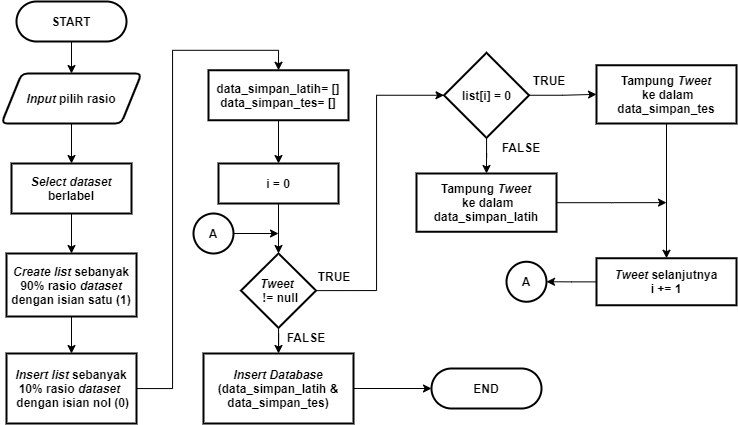
Pada *flowchart* ini, menjelaskan proses *labeling* yang dilakukan dengan cara pendekatan kamus sentimen, dimana diawali dengan perhitungan skor menggunakan kamus positif dan negatif, lalu penentuan kelas atau *label* berdasarkan nilai skor. *Flowchart* proses *labeling* dapat dilihat pada Gambar 4.18 berikut:



**Gambar 184.18 Flowchart proses labeling**

### *Flowchart* proses pembagian data

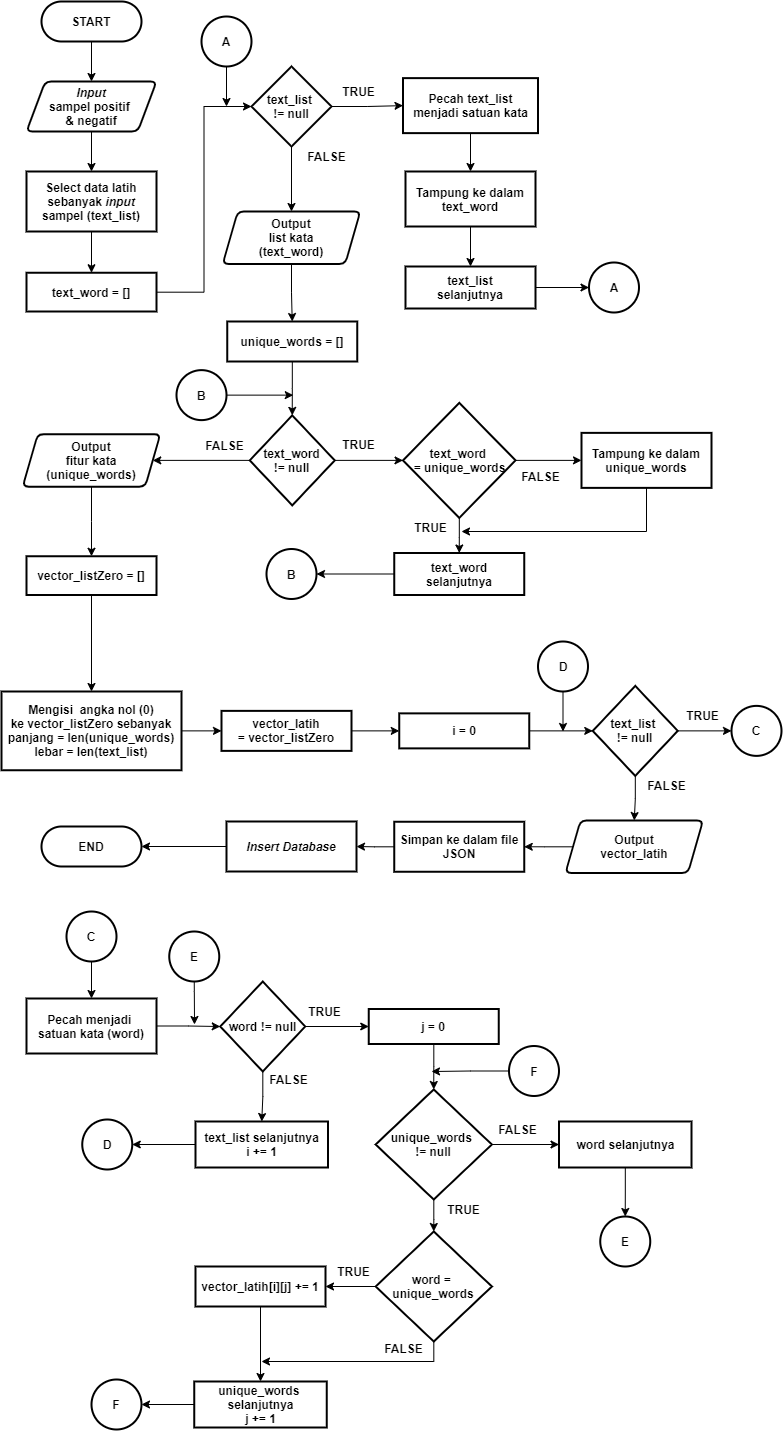
Pada *flowchart* ini, menjelaskan proses pembagian data ke dua (2) buah bagian, yaitu: data uji dan data latih, menggunakan rasio 1:9 (data uji : data latih). *Flowchart* proses pembagian datadapat dilihat pada Gambar 4.19 berikut:



**Gambar 194.19 Flowchart proses pembagian data**

### *Flowchart* proses *modeling*

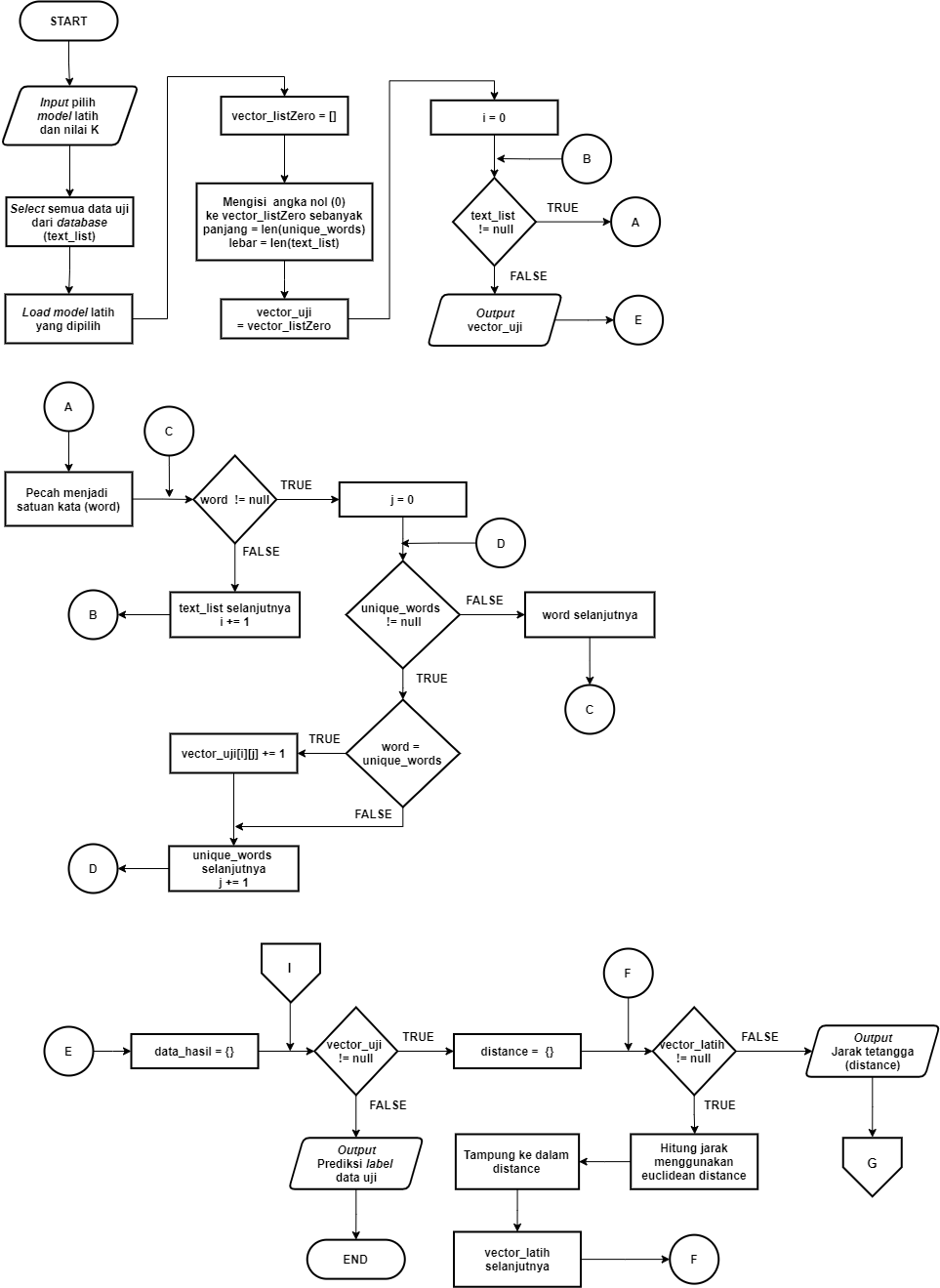
Pada *flowchart* ini, menjelaskan proses ekstraksi fitur dengan *CountVectorizer*. Dimulai dari proses seleksi data latih,pembuatan list kata, pencarian fitur kata, pembuatan vektor kosong, dan pembuatan vektor kata. Sehingga dapat diperoleh hasil berupa data *model* latih, yang kemudian disimpan ke dalam file JSON. *Flowchart* proses *modeling* dapat dilihat pada Gambar 4.20 berikut:

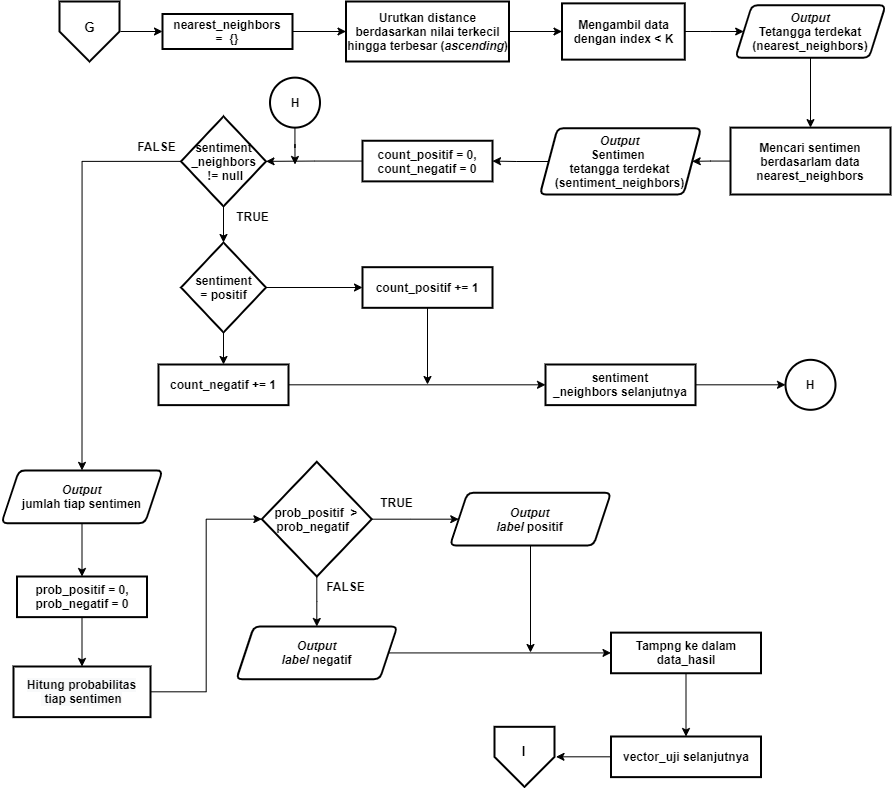


**Gambar 204.20 Flowchart proses modeling**

### *Flowchart* proses klasifikasi

Pada *flowchart* ini, menjelaskan proses klasifikasi dengan *K-Nearest Neighbor*. Dimulai dari proses pembuatan vektor uji, menghitung jarak antar data, mencari tetangga terdekat, dan menghitung nilai probabilitas. *Flowchart* proses klasifikasidapat dilihat pada Gambar 4.21 berikut:

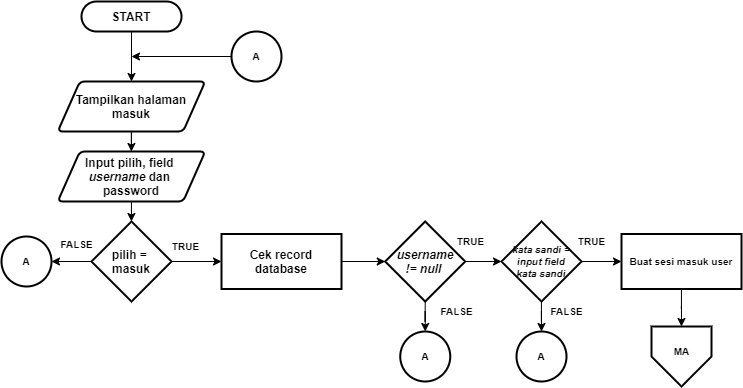




**Gambar 214.21 Flowchart proses klasifikasi**

### *Flowchart* menu masuk aplikasi

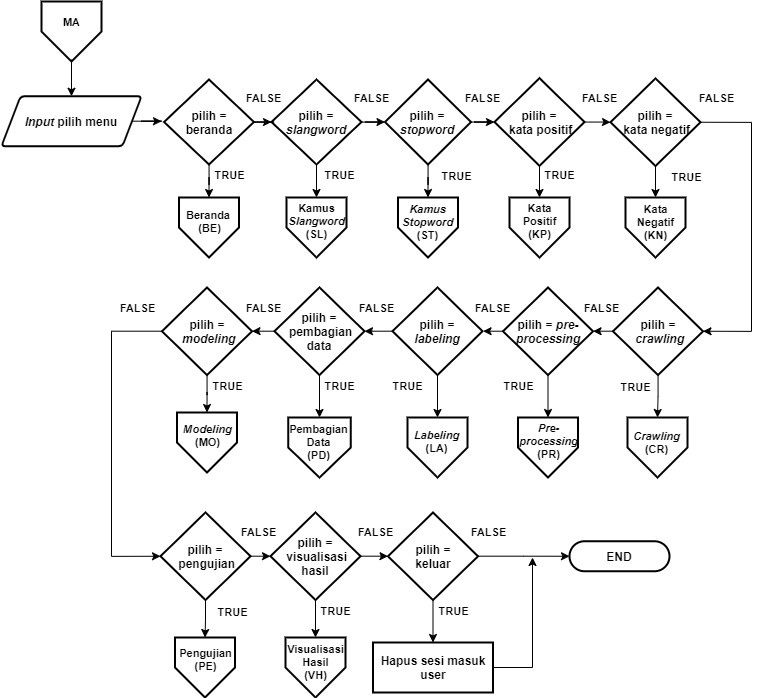
Pada *flowchart* ini, menjelaskan mengenai proses menu masuk dalam aplikasi yang dibuat. *Flowchart* menu masuk dapat dilihat pada Gambar 4.22 berikut:



**Gambar 224.22 Flowchart menu masuk aplikasi**

### *Flowchart* menu aplikasi

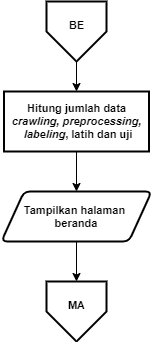
Pada *flowchart* ini, menjelaskan mengenai menu-menu dalam aplikasi yang dibuat. *Flowchart* keseluruhan menu aplikasidapat dilihat pada Gambar 4.23 berikut:



**Gambar 234.23 Flowchart menu aplikasi**

### *Flowchart* menu beranda

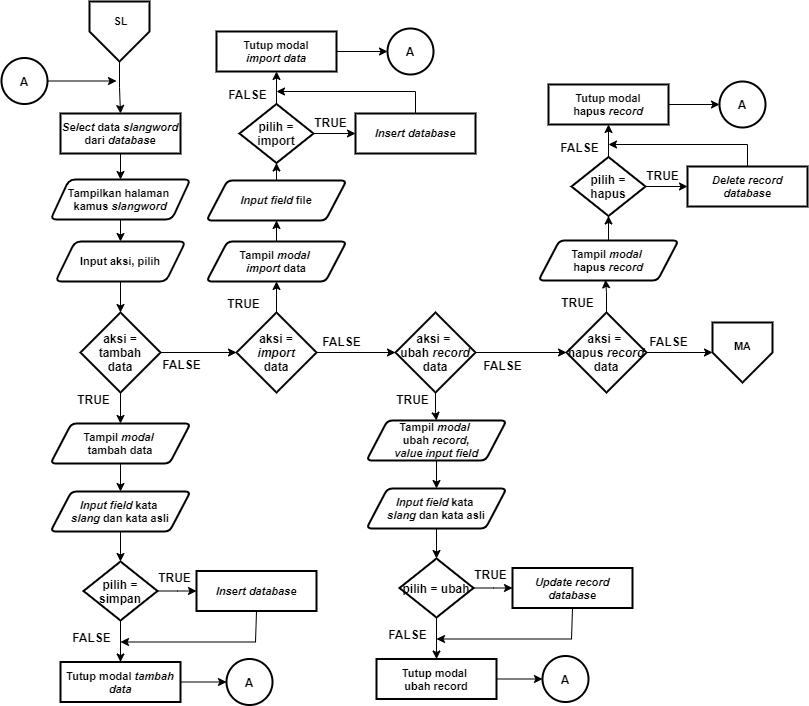
Pada *flowchart* ini, menjelaskan mengenai proses menu beranda dalam aplikasi yang dibuat. *Flowchart* menu beranda dapat dilihat pada Gambar 4.24 berikut:



**Gambar 244.24 Flowchart menu beranda**

### *Flowchart* menu *slang word*

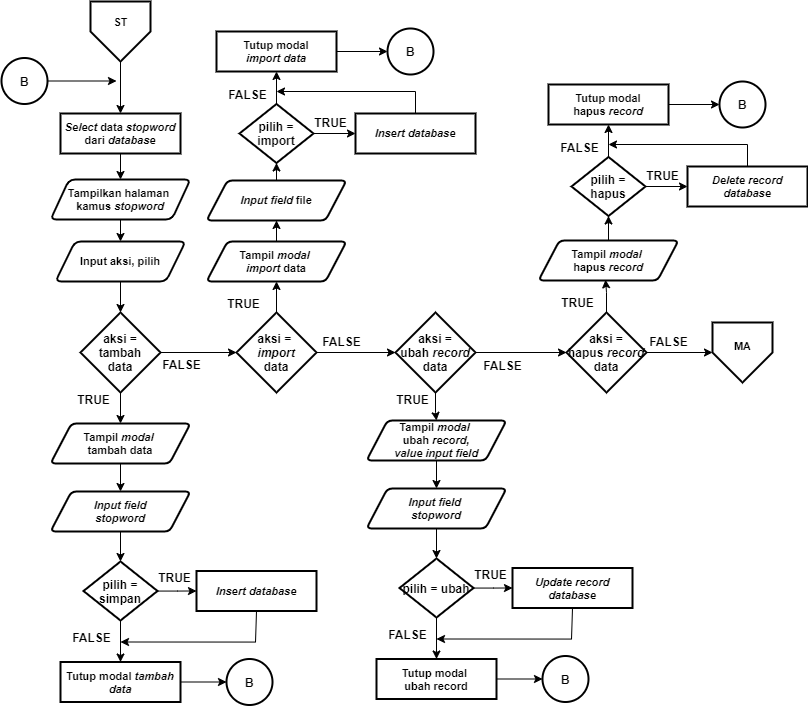
Pada *flowchart* ini, menjelaskan mengenai proses menu *slang word* dalam aplikasi yang dibuat. *Flowchart* menu *slang word* dapat dilihat pada Gambar 4.25 berikut:



**Gambar 254.25 Flowchart menu slang word**

### *Flowchart* menu *stop word*

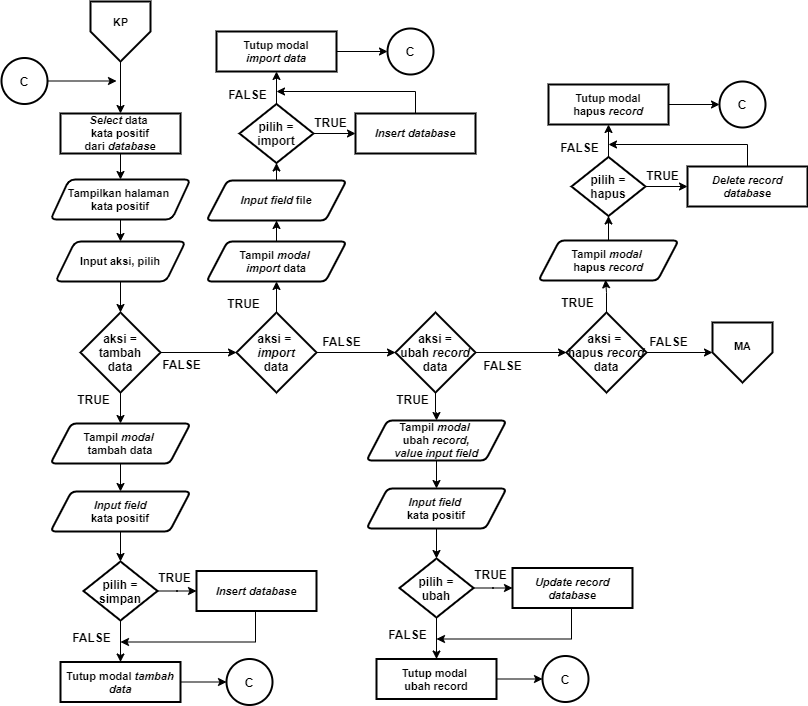
Pada *flowchart* ini, menjelaskan mengenai proses menu *stop word* dalam aplikasi yang dibuat. *Flowchart* menu *stop word* dapat dilihat pada Gambar 4.26 berikut:



**Gambar 264.26 Flowchart menu stop word**

### *Flowchart* menu kata positif

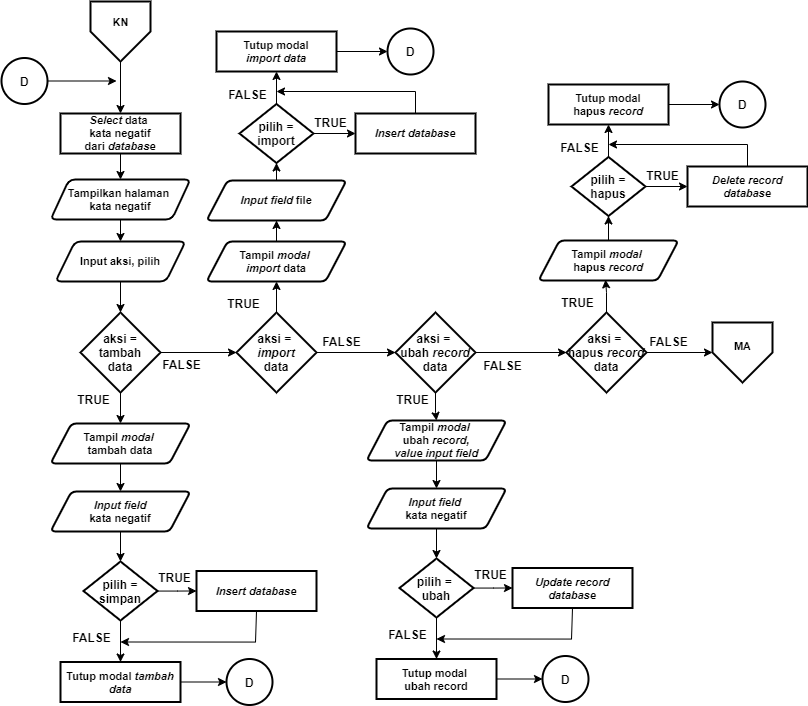
Pada *flowchart* ini, menjelaskan mengenai proses menu kata positif dalam aplikasi yang dibuat. *Flowchart* menu kata positif dapat dilihat pada Gambar 4.27 berikut:



**Gambar 274.27 Flowchart menu kata positif**

### *Flowchart* menu kata negatif

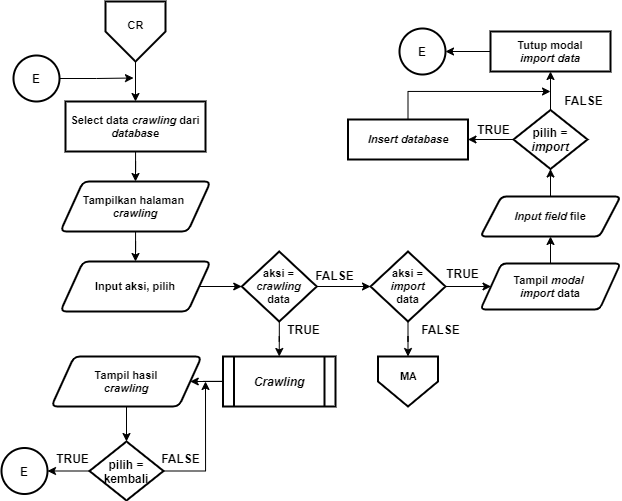
Pada *flowchart* ini, menjelaskan mengenai proses menu kata negatif dalam aplikasi yang dibuat. *Flowchart* menu kata negatif dapat dilihat pada Gambar 4.28 berikut:



**Gambar 284.28 Flowchart menu kata negatif**

### *Flowchart* menu *crawling*

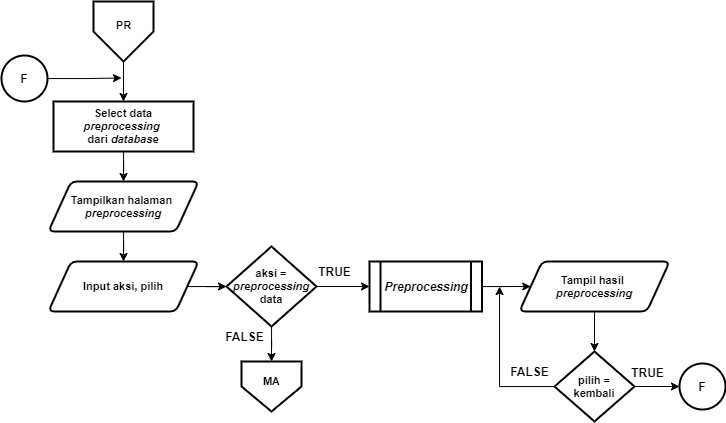
Pada *flowchart* ini, menjelaskan mengenai proses menu *crawling* dalam aplikasi yang dibuat. *Flowchart* menu *crawling* dapat dilihat pada Gambar 4.29 berikut:



**Gambar 294.29 Flowchart menu crawling**

### *Flowchart* menu *preprocessing*

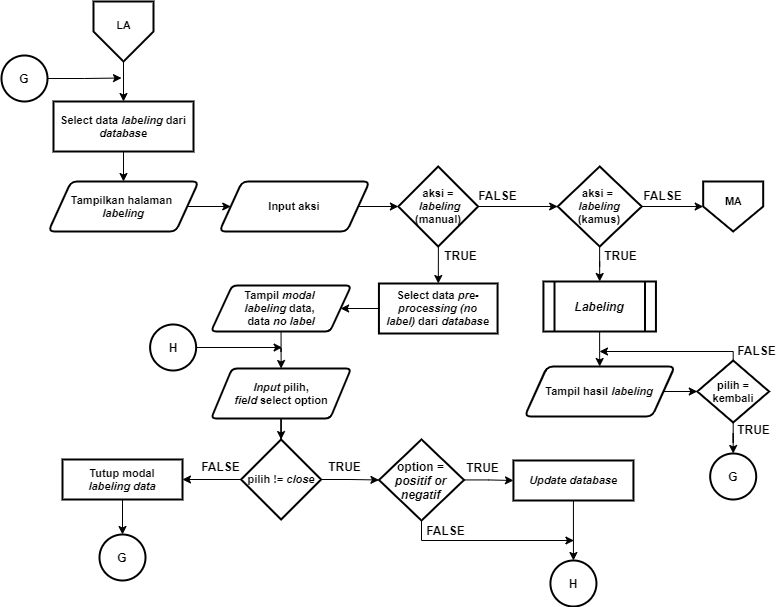
Pada *flowchart* ini, menjelaskan mengenai proses menu *preprocessing* dalam aplikasi yang dibuat. *Flowchart* menu *preprocessing* dapat dilihat pada Gambar 4.30 berikut:



**Gambar 304.30 Flowchart menu preprocessing**

### *Flowchart* menu *labeling*

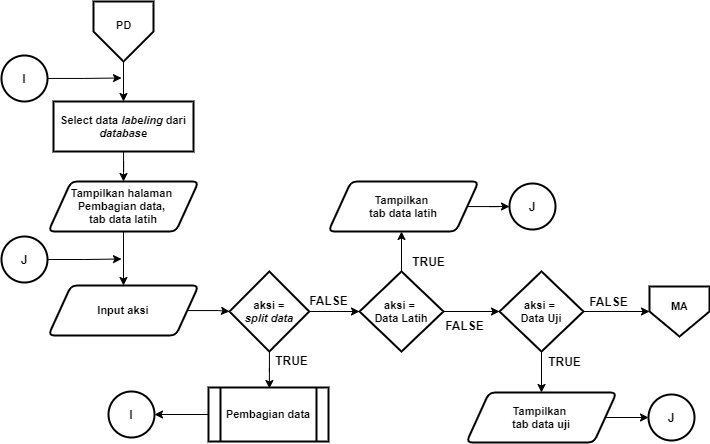
Pada *flowchart* ini, menjelaskan mengenai proses menu *labeling* dalam aplikasi yang dibuat. *Flowchart* menu *labeling* dapat dilihat pada Gambar 4.31 berikut:



**Gambar 314.31 Flowchart menu labeling**

### *Flowchart* menu pembagian data

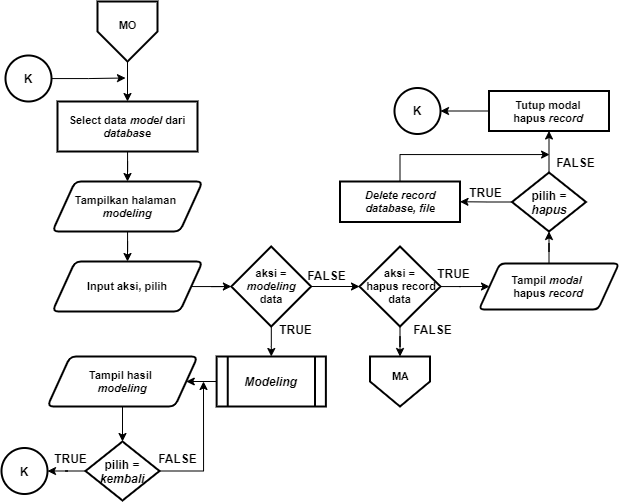
Pada *flowchart* ini, menjelaskan mengenai proses menu pembagian data dalam aplikasi yang dibuat. *Flowchart* menu pembagian data dapat dilihat pada Gambar 4.32 berikut:



**Gambar 324.32 Flowchart menu pembagian data**

### *Flowchart* menu *modeling*

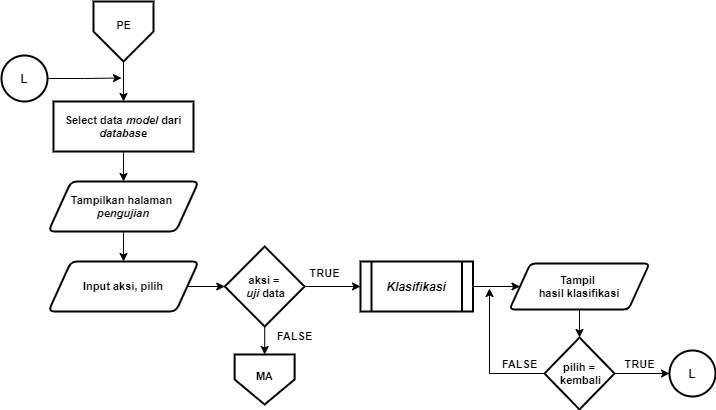
Pada *flowchart* ini, menjelaskan mengenai proses menu *modeling* dalam aplikasi yang dibuat. *Flowchart* menu *modeling* dapat dilihat pada Gambar 4.33 berikut:



**Gambar 334.33 Flowchart menu modeling**

### *Flowchart* menu pengujian

Pada *flowchart* ini, menjelaskan mengenai proses menu pengujian dalam aplikasi yang dibuat. *Flowchart* menu pengujian dapat dilihat pada Gambar 4.34 berikut:



**Gambar 344.34 Flowchart menu pengujian**

### *Flowchart* menu visualisasi hasil

Pada *flowchart* ini, menjelaskan mengenai proses menu visualisasi hasil dalam aplikasi yang dibuat. *Flowchart* menu visualisasi hasil dapat dilihat pada Gambar 4.35 berikut:



**Gambar 354.35 Flowchart menu visualisasi hasil**

## Algoritme Tahapan Metode

Algoritme merupakan suatu urutan atau tahapan proses yang dijabarkan dalam bentuk tulisan, algoritme juga merupakan representasi pengaplikasian dari suatu *flowchart*. Berikut adalah penjabaran algoritme berdasarkan pada *flowchart* yang telah dibuat sebelumnya.

### Algoritme keseluruhan proses sistem

Pada algoritme ini dijelaskan tentang proses keseluruhan sistem yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme keseluruhan proses sistem dapat dilihat pada Algoritme 4.1 berikut:

**Algoritme 4.1 Algoritme keseluruhan proses sistem**

|  |
| --- |
| 1 start  2 Lakukan proses Crawling  3 Baca Dataset Text (Tweet)  4 Lakukan proses Preprocessing  5 Baca Dataset Clean Text (Tweet)  6 Lakukan proses Labeling  7 Baca Dataset  8 Lakukan proses Pembagian Dataset  9 Baca Data Latih & Data Uji  10 Lakukan proses Modeling menggunakan data latih  11 Simpan Model ke file JSON  12 Lakukan proses Klasifikasi untuk data uji  13 Output prediksi label data uji  14 if (pengujian ulang == true)  15 Kembali ke nomor 12  16 endif  17 end |

### Algoritme proses *crawling*

Pada algoritme ini dijelaskan tentang proses pengumpulan data dengan cara *crawling* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *crawling* dapat dilihat pada Algoritme 4.2 berikut:

**Algoritme 4.2 Algoritme proses crawling**

|  |
| --- |
| 1 start  2 Input field kata kunci  3 Input field tanggal awal  4 Input field tanggal akhir  5 Crawling menggunakan pustaka Tweepy  6 Baca dokumen Excel (dataframe)  7 Simpan dataframe ke dalam database  8 end |

### Algoritme proses *preprocessing*

Pada algoritme ini dijelaskan tentang proses *preprocessing* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *preprocessing* dapat dilihat pada Algoritme 4.3 berikut:

**Algoritme 4.3 Algoritme proses preprocessing**

|  |
| --- |
| 1 start  2 Select Dateset (Tweet)  3 data\_simpan = []  4 if (Tweet != null)  5 Lakukan sub proses Casefolding  (result\_text)  6 Lakukan sub proses Cleansing (result\_text)  7 Lakukan sub proses Slangword (result\_text)  8 Lakukan sub proses Stopword (result\_text)  9 Lakukan sub proses Stemming (result\_text)  10 Simpan result\_text ke dalam list  data\_simpan  11 Tweet selanjutnya  12 Kembali ke nomor 4  13 else  14 Simpan data\_simpan ke dalam database  15 endif  16 end |

#### Algoritme proses case folding

Pada algoritme ini dijelaskan tentang proses *case folding* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *case folding* dapat dilihat pada Algoritme 4.4 berikut:

**Algoritme 4.4 Algoritme proses case folding**

|  |
| --- |
| 1 start  2 Baca Dateset (Tweet)  3 Ubah Tweet ke huruf kecil menggunakan pre-  made function lower()  4 Output result\_text  5 Simpan result\_text  6 end |

#### Algoritme proses cleansing

Pada algoritme ini dijelaskan tentang proses *cleansing* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *cleansing* dapat dilihat pada Algoritme 4.5 berikut:

**Algoritme 4.5 Algoritme proses cleansing**

|  |
| --- |
| 1 start  2 Select result\_text  3 Hapus URL  4 Hapus Mention  5 Hapus Hastag  6 Output result\_text  7 Hapus selain huruf  8 Output result\_text  9 Hapus spasi berlebih  10 Output result\_text  11 end |

#### Algoritme proses slang word

Pada algoritme ini dijelaskan tentang proses pengubahan *slang word* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *slang word* dapat dilihat pada Algoritme 4.6 berikut:

**Algoritme 4.6 Algoritme proses slang word**

|  |
| --- |
| 1 start  2 Select kamus slangword dari database  3 if (slangword != null)  4 if (teks == slangword)  5 Ubah result\_text (slangword) dengan  kata asli  6 Output result\_text  7 Simpan result\_text  8 else  9 slangword selanjutnya  10 Kembali ke nomor 3  11 endif  12 endif  13 end |

#### Algoritme proses stop word

Pada algoritme ini dijelaskan tentang proses penghilangan *stop word* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *stop word* dapat dilihat pada Algoritme 4.7 berikut:

**Algoritme 4.7 Algoritme proses stop word**

|  |
| --- |
| 1 start  2 Select kamus stopword dari database  3 if (stopword != null)  4 if (result\_text in stopword)  5 Hapus result\_text (stopword)  6 Output result\_text  7 Simpan result\_text  8 endif  9 stopword selanjutnya  10 Kembali ke nomor 3  11 endif  12 end |

#### Algoritme proses stemming

Pada algoritme ini dijelaskan tentang proses *stemming* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *stemming* dapat dilihat pada Algoritme 4.8 berikut:

**Algoritme 4.8 Algoritme proses stemming**

|  |
| --- |
| 1 start  2 Load pustaka Sastrawi  3 Inisialisasi paket StemmerFactory  4 Input result\_text sebagai parameter fungsi  stem()  5 Jalankan fungsi stem() melalui paket  StemmerFactory  6 Output result\_text  7 Simpan result\_text  8 end |

### Algoritme proses *labeling*

Pada algoritme ini dijelaskan tentang proses *labeling* kelas atau *labeling* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *labeling* dapat dilihat pada Algoritme 4.9 berikut:

**Algoritme 4.9 Algoritme proses labeling**

|  |
| --- |
| 1 start  2 Select dataset tanpa label dari database  3 Select kamus sentimen positif (Kata\_positif)  dan negatif (Kata\_negatif) dari database  4 data\_ubah = []  5 if (Tweet != null)  6 skor = 0  7 Split Tweet menjadi satuan kata (Kata)  8 if (Kata != null)  9 if (Kata\_positif != null)  10 if (Kata == Kata\_positif)  11 skor = skor + 1  12 break (Langsung menuju ke nomor 17)  13 else  14 Kata\_positif selanjutnya  15 Kembali ke nomor 9  16 endif  17 endif  18 if (Kata\_negatif != null)  19 if (Kata == Kata\_negatif)  20 skor = skor - 1  21 break (Langsung menuju ke nomor 26)  22 else  23 Kata\_negatif selanjutnya  24 Kembali ke nomor 18  25 endif  26 endif  27 Kata selanjutnya  28 Kembali ke nomor 8  29 else  30 if (skor > 0)  31 Output label positif  32 Simpan ke dalam list data\_ubah  33 else if (skor < 0)  34 Output label negatif  35 Simpan ke dalam list data\_ubah  36 else  37 continue (Langsung menuju ke nomor 38)  38 endif  39 Tweet selanjutnya  40 Kembali ke nomor 5  41 endif  42 else  43 Update database berdasarkan list data\_ubah  44 endif  45 en |

### Algoritme proses pembagian data

Pada algoritme ini dijelaskan tentang proses pembagian data yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses pembagian data dapat dilihat pada Algoritme 4.10 berikut:

**Algoritme 4.10 Algoritme proses pembagian data**

|  |
| --- |
| 1 start  2 Input pilih rasio pembagian data  3 Select dataset berlabel  4 Buat list dengan isian satu (1) sebanyak 90%  dari total dataset berlabel  5 Buat list dengan isian nol (0) sebanyak 10%  dari total dataset berlabel  6 data\_simpan\_latih = []  7 data\_simpan\_tes = []  8 i = 0  9 if (Tweet != null)  10 if(list[i] == 0)  11 Simpan Tweet dalam data\_simpan\_tes  12 else  13 Simpan Tweet le dalam data\_simpan\_latih  14 endif  15 list = list + 1  16 Kembali ke nomor 9  17 else  18 Simpan data\_simpan\_latih ke dalam database  19 Simpan data\_simpan\_tes ke dalam database  20 endif  21 end |

### Algoritme proses *modeling*

Pada algoritme ini dijelaskan tentang proses pembuatan *model* latih atau *modeling* yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses *modeling* dapat dilihat pada Algoritme 4.11 berikut:

**Algoritme 4.11 Algoritme proses modeling**

|  |
| --- |
| 1 start  2 Input jumlah sampel positif dan negatif  3 Select data latih sebanyak jumlah input sampel  (text\_list)  4 text\_word = []  5 if (text\_list != null)  6 Split text\_list menjadi satuan kata (kata)  7 Simpan kata ke dalam list text\_word  8 text\_list selanjutnya  9 Kembali ke nomor 5  10 else  11 Output text\_word  12 unique\_words = []  13 if (text\_word != null)  14 if (text\_word in unique\_words)  15 text\_word selanjutnya  16 else  17 Simpan text\_word ke dalam list  unique\_words  18 text\_word selanjutnya  19 endif  20 Kembali ke nomor 13  21 else  22 Output unique\_words  23 vector\_listZero = []  24 Isi dengan angka nol (0) ke dalam list  vector\_listZero dengan panjang =  len(unique\_words dan lebar =  len(text\_list)  25 vector\_latih = vector\_listZero  26 i = 0  27 if (text\_list[i] != null)  28 Split text\_list[i] menjadi satuan  kata (word)  29 if (word != null)  30 j = 0  31 if (unique\_words[j] != null)  32 if (word ==  unique\_words[j])  33 vector\_latih[i][j] =  vector\_latih[i][j] + 1  34 endif  35 j = j + 1  36 Kembali ke nomor 31  37 else  38 word selanjutnya  39 Kembali ke nomor 29  40 endif  41 else  42 i = i + 1  43 Kembali ke nomor 27  44 endif  45 else  46 Output vector\_latih  47 Simpan isi vector\_latih ke dalam  file JSON (.json) pada direktori  48 Simpan nama vector\_latih ke dalam  database  49 endif  50 endif  51 endif |

### Algoritme proses klasifikasi

Pada algoritme ini dijelaskan tentang proses klasifikasi yang dilakukan berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme proses klasifikasi dapat dilihat pada Algoritme 4.12 berikut:

**Algoritme 4.12 Algoritme proses klasifikasi**

|  |
| --- |
| 1 start  2 Input pilih model latih dan nilai K  3 Select data uji dari database (text\_list)  4 Load model latih yang dipilih dari direktori  5 vector\_listZero = []  6 Isi dengan angka nol (0) ke dalam list  vector\_listZero dengan panjang =  len(unique\_words) dan lebar = len(text\_list)  7 vector\_uji = vector\_listZero  8 i = 0  9 if (text\_list[i] != null)  10 Split text\_list menjadi satuan kata (word)  11 if (word != null)  12 j = 0  13 if (unique\_words[j] != null)  14 if(word == unique\_words[j])  15 vector\_uji[i][j] =  vector\_uji[i][j] + 1  16 endif  17 j = j + 1  18 Kembali ke nomor 13  19 else  20 word selanjutnya  21 Kembali ke nomor 11  22 endif  23 else  24 i = i + 1  25 Kembali ke nomor 9  26 endif  27 else  28 Output vector\_uji  29 data\_hasil = {}  30 if (vector\_uji != null)  31 distance = {}  32 if(vector\_latih != null)  33 Hitung jarak menggunakan euclidean  distance  34 Simpan hasil jarak ke dalam dict  distance  35 vector\_latih selanjutnya  36 Kembali ke nomor 32  37 else  38 Output dict distance  39 nearest\_neighbors = {}  40 Urut data secara ascending  berdasarkan jarak (distance)  41 Select data dengan index < K  42 Output nearest\_neighbors  43 Mencari jenis label berdasarkan  data nearest\_neighbors  44 Output sentiment\_neighbors  45 count\_positif = 0  46 count\_negatif = 0  47 if (sentiment\_neighbors != null)  48 if (sentimen == 'positif')  49 count\_positif += 1  50 else  51 count\_negatif += 1  52 endif  53 sentiment\_neighbors selanjutnya  54 Kembali ke nomor 47  55 else  56 Output count\_positif &  count\_negatif  57 prob\_positif = 0  58 prob\_negatif = 0  59 Hitung probabilitas positif dan  negatif  60 if (prob\_positif >  prob\_negatif)  61 Output label positif  62 else  63 Output label negatif  64 endif  65 Simpan label ke dalam list  data\_hasil  66 vector\_uji selanjutnya  67 Kembali ke nomor 30  68 endif  69 endif  70 else  71 Output Prediksi label data uji  (data\_hasil)  72 endif  73 endif  74 end |

### Algoritme menu masuk aplikasi

Pada algoritme ini dijelaskan tentang menu masuk (*login*) dalam aplikasi yang dibuat, berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme keseluruhan menu aplikasi dapat dilihat pada Algoritme 4.13 berikut:

**Algoritme 4.13 Algoritme menu masuk aplikasi**

|  |
| --- |
| 1 start  2 Tampilkan halaman masuk aplikasi  3 Input field username  4 Input field password  5 pilih = ''  6 if(pilih == 'masuk')  7 Cek record database berdasarkan inputan  8 if(record username != null)  9 if(record kata sandi == inputan kata  sandi)  10 Buat sesi masuk untuk user  11 Akses menu awal  12 else  13 Kembali ke nomor 2  14 endif  15 else  16 Kembali ke nomor 2  17 endif  18 else  19 Kembali ke nomor 2  20 endif |

### Algoritme menu aplikasi

Pada algoritme ini dijelaskan tentang menu-menu yang ada dalam aplikasi yang dibuat, berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme keseluruhan menu aplikasi dapat dilihat pada Algoritme 4.14 berikut:

**Algoritme 4.14 Algoritme menu aplikasi**

|  |
| --- |
| 1 Input pilih = ''  2 if (pilih == 'beranda')  3 Akses menu beranda  4 else (if pilih == 'slangword')  5 Akses menu kamus slangword  6 else (if pilih == 'stopword')  7 Akses menu kamus stopword  8 else (if pilih == 'kata positif')  9 Akses menu kata positif  10 else (if pilih == 'kata negatif')  11 Akses menu kata negatif  12 else (if pilih == 'crawling')  13 Akses menu crawling  14 else (if pilih == 'preprocessing')  15 Akses menu preprocessing  16 else (if pilih == 'labeling')  17 Akses menu labeling  18 else (if pilih == 'pembagian data')  19 Akses menu pembagian data  20 else (if pilih == 'modeling')  21 Akses menu modeling  22 else (if pilih == 'pengujian')  23 Akses menu pengujian  24 else (if pilih == 'visualisai hasil')  25 Akses menu visualisai hasil  26 else (if pilih == 'keluar')  27 Hapus sesi masuk user  28 endif  29 end |

### Algoritme menu beranda

Pada algoritme ini dijelaskan tentang menu beranda berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu beranda dapat dilihat pada Algoritme 4.15 berikut:

**Algoritme 4.15 Algoritme menu beranda**

|  |
| --- |
| 1 Hitung jumlah data crawling  2 Hitung jumlah data preprocessing  3 Hitung jumlah data labeling  4 Hitung jumlah data latih  5 Hitung jumlah data uji  6 Tampilkan halaman beranda  7 Akses menu awal |

### Algoritme menu *slang word*

Pada algoritme ini dijelaskan tentang menu *slang word* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *slang word* dapat dilihat pada Algoritme 4.16 berikut:

**Algoritme 4.16 Algoritme menu slang word**

|  |
| --- |
| 1 Select data slangword  2 Tampilkan halaman kamus slangword  3 aksi = ''  4 pilih = ''  5 if (aksi == 'tambah data')  6 Tampil modal tambah data  7 Input field kata slang dan kata asli  8 if (pilih == 'simpan')  9 Insert database  10 Kembali ke nomor 1  11 else  12 Tutup modal tambah data  13 Kembali ke nomor 1  14 endif  15 else if (aksi == 'import data')  16 Tampil modal import data  17 Input field file  18 if (pilih == 'import')  19 Insert database  20 Kembali ke nomor 1  21 else  22 Tutup modal import data  23 Kembali ke nomor 1  24 endif  25 else if (aksi == 'ubah record data')  26 Tampil modal ubah record data dan value  input field  27 Input field kata slang dan kata asli  28 if (pilih == 'ubah')  29 Update record database  30 Kembali ke nomor 1  31 else  32 Tutup modal ubah record data  33 Kembali ke nomor 1  34 endif  35 else if (aksi == 'hapus record data')  36 Tampil modal hapus record data  37 if (pilih == 'hapus')  38 Delete record database  39 Kembali ke nomor 1  40 else  41 Tutup modal hapus record data  42 Kembali ke nomor 1  43 endif  44 else  45 Akses menu awal  46 endif |

### Algoritme menu *stop word*

Pada algoritme ini dijelaskan tentang menu *stop word* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *stop word* dapat dilihat pada Algoritme 4.17 berikut:

**Algoritme 4.17 Algoritme menu stop word**

|  |
| --- |
| 1 Select data stopword  2 Tampilkan halaman kamus stopword  3 aksi = ''  4 pilih = ''  5 if (aksi == 'tambah data')  6 Tampil modal tambah data  7 Input field stopword  8 if (pilih == 'simpan')  9 Insert database  10 Kembali ke nomor 1  11 else  12 Tutup modal tambah data  13 Kembali ke nomor 1  14 endif  15 else if (aksi == 'import data')  16 Tampil modal import data  17 Input field file  18 if (pilih == 'import')  19 Insert database  20 Kembali ke nomor 1  21 else  22 Tutup modal import data  23 Kembali ke nomor 1  24 endif  25 else if (aksi == 'ubah record data')  26 Tampil modal ubah record data dan value  input field  27 Input field stopword  28 if (pilih == 'ubah')  29 Update record database  30 Kembali ke nomor 1  31 else  32 Tutup modal ubah record data  33 Kembali ke nomor 1  34 endif  35 else if (aksi == 'hapus record data')  36 Tampil modal hapus record data  37 if (pilih == 'hapus')  38 Delete record database  39 Kembali ke nomor 1  40 else  41 Tutup modal hapus record data  42 Kembali ke nomor 1  43 endif  44 else  45 Akses menu awal  46 endif |

### Algoritme menu kata positif

Pada algoritme ini dijelaskan tentang menu kata positif berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu kata positif dapat dilihat pada Algoritme 4.18 berikut:

**Algoritme 4.18 Algoritme menu kata positif**

|  |
| --- |
| 1 Select data kata positif  2 Tampilkan halaman kata positif  3 aksi = ''  4 pilih = ''  5 if (aksi == 'tambah data')  6 Tampil modal tambah data  7 Input field kata positif  8 if (pilih == 'simpan')  9 Insert database  10 Kembali ke nomor 1  11 else  12 Tutup modal tambah data  13 Kembali ke nomor 1  14 endif  15 else if (aksi == 'import data')  16 Tampil modal import data  17 Input field file  18 if (pilih == 'import')  19 Insert database  20 Kembali ke nomor 1  21 else  22 Tutup modal import data  23 Kembali ke nomor 1  24 endif  25 else if (aksi == 'ubah record data')  26 Tampil modal ubah record data dan value  input field  27 Input field kata positif  28 if (pilih == 'ubah')  29 Update record database  30 Kembali ke nomor 1  31 else  32 Tutup modal ubah record data  33 Kembali ke nomor 1  34 endif  35 else if (aksi == 'hapus record data')  36 Tampil modal hapus record data  37 if (pilih == 'hapus')  38 Delete record database  39 Kembali ke nomor 1  40 else  41 Tutup modal hapus record data  42 Kembali ke nomor 1  43 endif  44 else  45 Akses menu awal  46 endif |

### Algoritme menu kata negatif

Pada algoritme ini dijelaskan tentang menu kata negatif berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu kata negatif dapat dilihat pada Algoritme 4.19 berikut:

**Algoritme 4.19 Algoritme menu kata negatif**

|  |
| --- |
| 1 Select data kata negatif  2 Tampilkan halaman kata negatif  3 aksi = ''  4 pilih = ''  5 if (aksi == 'tambah data')  6 Tampil modal tambah data  7 Input field kata negatif  8 if (pilih == 'simpan')  9 Insert database  10 Kembali ke nomor 1  11 else  12 Tutup modal tambah data  13 Kembali ke nomor 1  14 endif  15 else if (aksi == 'import data')  16 Tampil modal import data  17 Input field file  18 if (pilih == 'import')  19 Insert database  20 Kembali ke nomor 1  21 else  22 Tutup modal import data  23 Kembali ke nomor 1  24 endif  25 else if (aksi == 'ubah record data')  26 Tampil modal ubah record data dan value  input field  27 Input field kata negatif  28 if (pilih == 'ubah')  29 Update record database  30 Kembali ke nomor 1  31 else  32 Tutup modal ubah record data  33 Kembali ke nomor 1  34 endif  35 else if (aksi == 'hapus record data')  36 Tampil modal hapus record data  37 if (pilih == 'hapus')  38 Delete record database  39 Kembali ke nomor 1  40 else  41 Tutup modal hapus record data  42 Kembali ke nomor 1  43 endif  44 else  45 Akses menu awal  46 endif |

### Algoritme menu *crawling*

Pada algoritme ini dijelaskan tentang menu *crawling* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *crawling* dapat dilihat pada Algoritme 4.20 berikut:

**Algoritme 4.20 Algoritme menu crawling**

|  |
| --- |
| 1 Select data crawling  2 Tampilkan halaman crawling  3 aksi = ''  4 pilih = ''  5 if (aksi == 'crawling data')  6 Akses proses crawling  7 Tampil hasil proses crawling  8 if (pilih == 'kembali')  9 Kembali ke nomor 1  10 endif  11 else if (aksi == 'import data')  12 Tampil modal import data  13 Input field file  14 if (pilih == 'import')  15 Insert database  16 Kembali ke nomor 1  17 else  18 Tutup modal import data  19 Kembali ke nomor 1  20 endif  21 else  22 Akses menu awal  23 endif |

### Algoritme menu *preprocessing*

Pada algoritme ini dijelaskan tentang menu *preprocessing* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *preprocessing* dapat dilihat pada Algoritme 4.21 berikut:

**Algoritme 4.21 Algoritme menu preprocessing**

|  |
| --- |
| 1 Select data preprocessing  2 Tampilkan halaman preprocessing  3 aksi = ''  4 pilih = ''  5 if (aksi == 'preprocessing data')  6 Akses proses preprocessing  7 Tampil hasil proses preprocessing  8 if (pilih == 'kembali')  9 Kembali ke nomor 1  10 endif  11 else  12 Akses menu awal  13 endif |

### Algoritme menu *labeling*

Pada algoritme ini dijelaskan tentang menu *labeling* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *labeling* dapat dilihat pada Algoritme 4.22 berikut:

**Algoritme 4.22 Algoritme menu labeling**

|  |
| --- |
| 1 Select data labeling  2 Tampilkan halaman labeling  3 aksi = ''  4 pilih = ''  5 if (aksi == 'labeling (manual)')  6 Select data tidak berlabel (hasil  preprocessing)  7 Tampil modal labeling data, data tidak  berlabel  8 Input select option  9 if (pilih != 'close')  10 if(option == 'positif' || option ==  'negatif')  11 Update database  12 Kembali ke nomor 8  13 endif  14 else  15 Tutup modal labeling data  16 Kembali ke nomor 1  17 endif  18 else if (aksi == 'labeling (kamus)')  19 Akses proses labeling  20 Tampil hasil proses labeling  21 if (pilih == 'kembali')  22 Kembali ke nomor 1  23 endif  24 else  25 Akses menu awal  26 endif |

### Algoritme menu pembagian data

Pada algoritme ini dijelaskan tentang menu pembagian data berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu pembagian data dapat dilihat pada Algoritme 4.23 berikut:

**Algoritme 4.23 Algoritme menu pembagian data**

|  |
| --- |
| 1 Select data hasil labeling  2 Tampilkan halaman pembagian data, tab data  latih  3 aksi = ''  4 if (aksi == 'split data')  5 Akses proses pembagian data  6 Kembali ke nomor 1  7 else if (aksi == 'data latih')  8 Tampilkan tab data latih  9 Kembali ke nomor 3  10 else if (aksi == 'data uji')  11 Tampilkan tab data uji  12 Kembali ke nomor 3  13 else  14 Akses menu awal  15 endif |

### Algoritme menu *modeling*

Pada algoritme ini dijelaskan tentang menu *modeling* berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu *modeling* dapat dilihat pada Algoritme 4.24 berikut:

**Algoritme 4.24 Algoritme menu modeling**

|  |
| --- |
| 1 Select data model  2 Tampilkan halaman modeling  3 aksi = ''  4 pilih = ''  5 if (aksi == 'modeling data')  6 Akses proses modeling  7 Tampil hasil proses modeling  8 if (pilih == 'kembali')  9 Kembali ke nomor 1  10 endif  11 else if (aksi == 'hapus record data')  12 Tampil modal hapus record data  13 if (pilih == 'hapus')  14 Delete record database, file  15 Kembali ke nomor 1  16 else  17 Tutup modal hapus record data  18 Kembali ke nomor 1  19 endif  20 else  21 Akses menu awal  22 endif |

### Algoritme menu pengujian

Pada algoritme ini dijelaskan tentang menu pengujian berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu pengujian dapat dilihat pada Algoritme 4.25 berikut:

**Algoritme 4.25 Algoritme menu pengujian**

|  |
| --- |
| 1 Select data model  2 Tampilkan halaman pengujian  3 aksi = ''  4 pilih = ''  5 if (aksi == 'uji data')  6 Akses proses klasifikasi  7 Tampil hasil proses klasifikasi  8 if (pilih == 'kembali')  9 Kembali ke nomor 1  10 endif  11 else  12 Akses menu awal  13 endif |

### Algoritme menu visualisasi hasil

Pada algoritme ini dijelaskan tentang menu visualisasi hasil berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu visualisasi hasil dapat dilihat pada Algoritme 4.26 berikut:

**Algoritme 4.26 Algoritme menu visualisasi hasil**

|  |
| --- |
| 1 Create histogram data  2 Create pie chart data  3 Create word cloud data  4 Hitung frekuensi data  5 Tampilkan halaman visualisasi hasil  6 Akses menu awal |

## Pengujian

Pengujian merupakan salah satu hal yang perlu dilakukan dalam setiap pengembangan sistem untuk mengevaluasi, menganalisa dan mengetahui tingkat akurasi atau kesamaan hasil yang telah dicapai oleh sistem yang telah dirancang. Pada penelitian ini, dilakukan pengujian dari sisi akurasi, presisi dan *recall* pada implementasi algoritme *K-nearest neighbor* (KNN)dalam memprediksikan *label* untuk data uji. Selain pada sisi akurasi, presisi dan *recall* pengujian pada penelitian ini juga menguji nilai K berdasarkan variasi yang telah ditetapkan, yaitu K=3, K=5, K-7, K=9, dan K=11. Hasil sampel prediksi oleh algoritme KNN dengan nilai K=3 dapat dilihat pada Tabel 4.15 berikut:

**Tabel 154.15 Sampel data hasil prediksi**

| No | *Tweet* | *Label*  aktual | *Label* prediksi |
| --- | --- | --- | --- |
| 1 | Selain untuk memperkuat kompetensi literasi dan numerasi, tujuan lain program BDR adalah untuk membangun ikatan emosional dalam keluarga. https://t.co/Hks8kSVhl3 | positif | positif |
| 2 | @schfess Kontra, udah capek batin belajar dari rumah, materi gak ada yang masuk dan karna gua SMK di mana harus banyak praktek. | negatif | negatif |
| 3 | Pandemi juga memaksa adik-adik kita belajar online dari rumah. Sayangnya, tidak semua dari mereka memiliki fasilitas internet dan gawai yang mendukung. Untungnya, banyak orang baik di luar sana yang tergerak membantu adik-adik agar bisa tetap bisa belajar | positif | negatif |
| ... | ... | ... | ... |
| 395 | Seri Kepribadian Muslim 3 : Bersih dan Sehat - Diskon 20% menjadi Rp.20000 Gratis ongkos kirim hingga 40rb keseluruh Indonesia. #mainanbukuanak #promo #dirumahaja #belajardirumah #buku #membaca #booklover #bookstorm SINOPSIS:Aku suka bersih. Aku suka | positif | positif |

Pada Tabel 4.15 Sampel data hasil prediksi, kolom *label* aktual merupakan data *label* yang diperoleh melalui proses *labeling*, sementara *label* prediksi merupakan data *label* hasil dari proses klasifikasi menggunakan KNN. Keseluruhan hasil prediksi data uji (395 data *tweet*) kemudian direpresentasikan ke dalam *confusion matrix*. Representasi *confusion matrix* untuk K=3 yang terbentuk dapat terlihat pada Tabel 4.16 berikut:

**Tabel 164.16 Confusion matrix pengujian K=3**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nilai Aktual | |
|  |  | positif | negatif |
| Nilai Prediksi | positif | 211 | 70 |
| negatif | 50 | 64 |

Berdasarkan Tabel 4.16 *Confusion matrix* pengujian K=3, maka perolehan nilai akurasi, presisi dan *recall* menggunakan rumus yang telah dijabarkan dalam persamaan (3. 1), persamaan (3. 2), dan persamaan (3. 3) dapat dilihat pada Tabel 4.17 berikut:

**Tabel 174.17 Nilai pengujian K=3**

| Pengujian | | |
| --- | --- | --- |
| Akurasi | = | 0.7 (70 %) |
| Presisi | = | 0.75 (75 %) |
| *Recall* | = | 0.81 (81 %) |

Pengujian di atas dilakukan secara berulang dengan variasi nilai K yang berbeda-beda. Sehingga dapat diketahui hasil pengujian secara keseluruhan adalah seperti Tabel 4.18 berikut:

**Tabel 184.18 Hasil pengujian**

|  | K=3 | K=5 | K=7 | K=9 | K=11 |
| --- | --- | --- | --- | --- | --- |
| Akurasi | 0.7 | 0.7 | 0.71 | 0.68 | 0.67 |
| Presisi | 0.75 | 0.75 | 0.74 | 0.72 | 0.72 |
| *Recall* | 0.81 | 0.84 | 0.86 | 0.84 | 0.81 |

Berdasarkan Tabel 4.18 Hasil pengujian, dapat diketahui bahwa hasil pengujian menunjukkan bahwa algoritme KNN mampu memperoleh nilai pengujian tertinggi menggunakan nilai K=7, dengan akurasi 71%, presisi 74%, dan *recall* 86%.

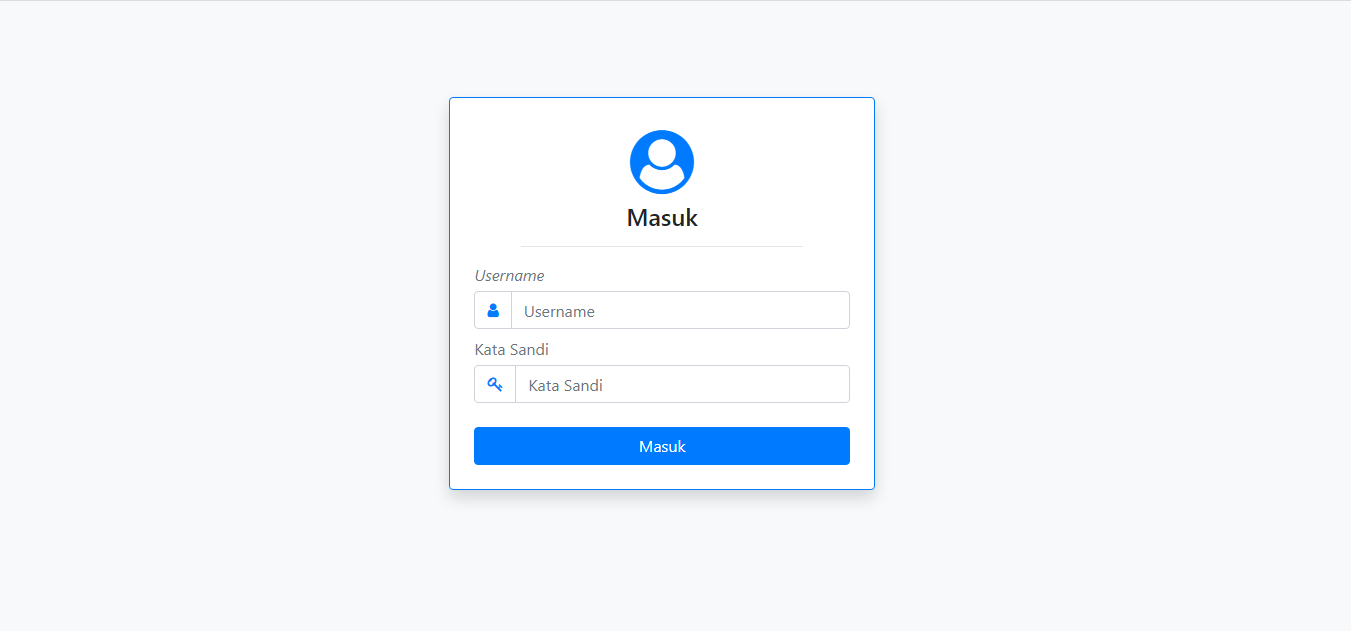
Sementara itu hasil analisis sentimen terhadap 3.954 *tweet* menunjukkan arah pandangan (sentimen) masyarakat Indonesia cenderung positif sebesar 76.56%, sementara sentimen negatif sebesar 23.44% pada periode Desember 2020.

## Tampilan Layar Aplikasi

Dalam penerapanya, penelitian ini dituangkan ke dalam bentuk program aplikasi, berikut beberapa tampilan layar dari aplikasi yang dibuat.

### Tampilan layar masuk aplikasi

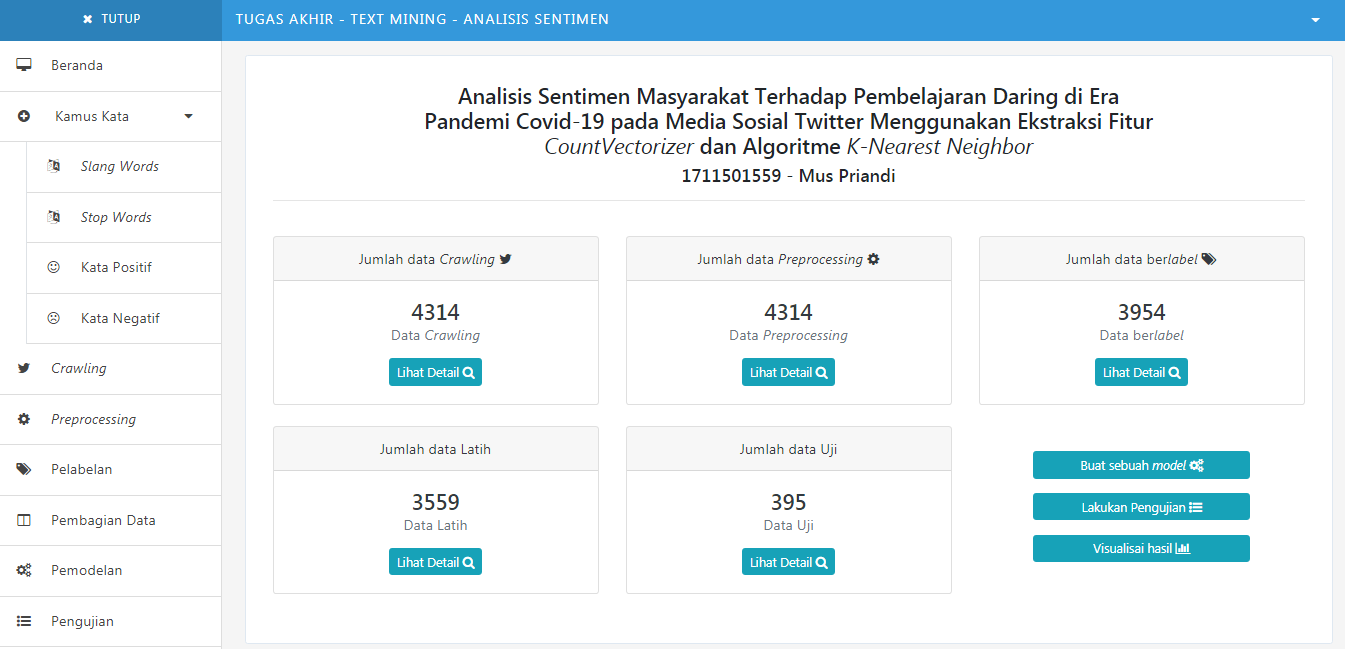
Tampilan layar masuk aplikasi dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.36 berikut:



**Gambar 36.4.36 Tampilan layar masuk aplikasi**

### Tampilan layar beranda

Tampilan layar beranda dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.37 berikut:

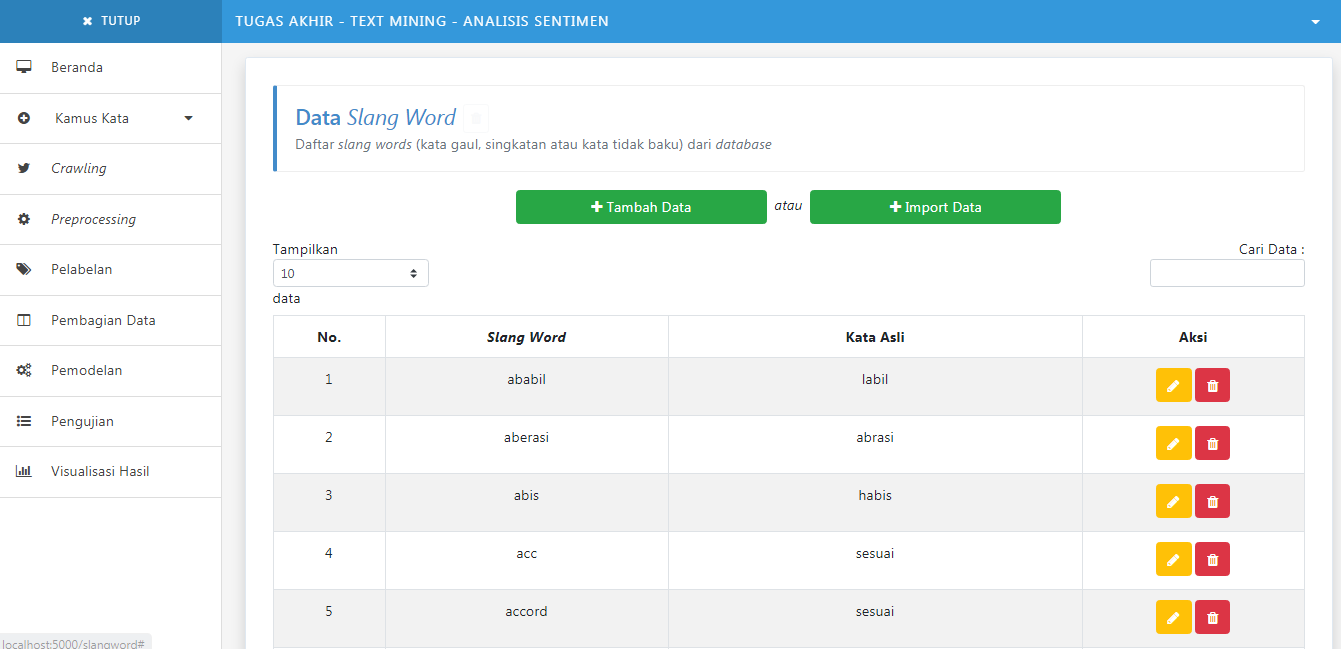


**Gambar 37.4.37 Tampilan layar beranda**

### Tampilan layar kamus kata

#### Tampilan layar kamus slang word

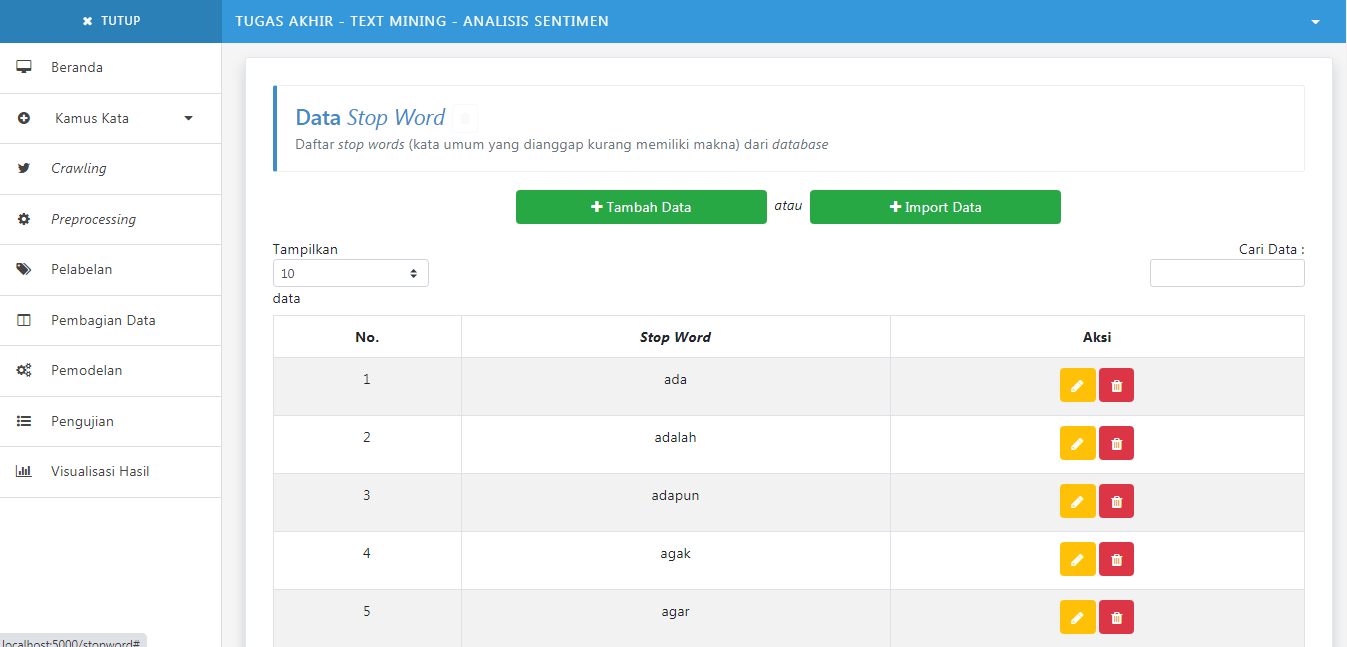
Tampilan layar kamus *slang word* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.38 berikut:



**Gambar 384.38 Tampilan layar kamus slang word**

#### Tampilan layar kamus stop word

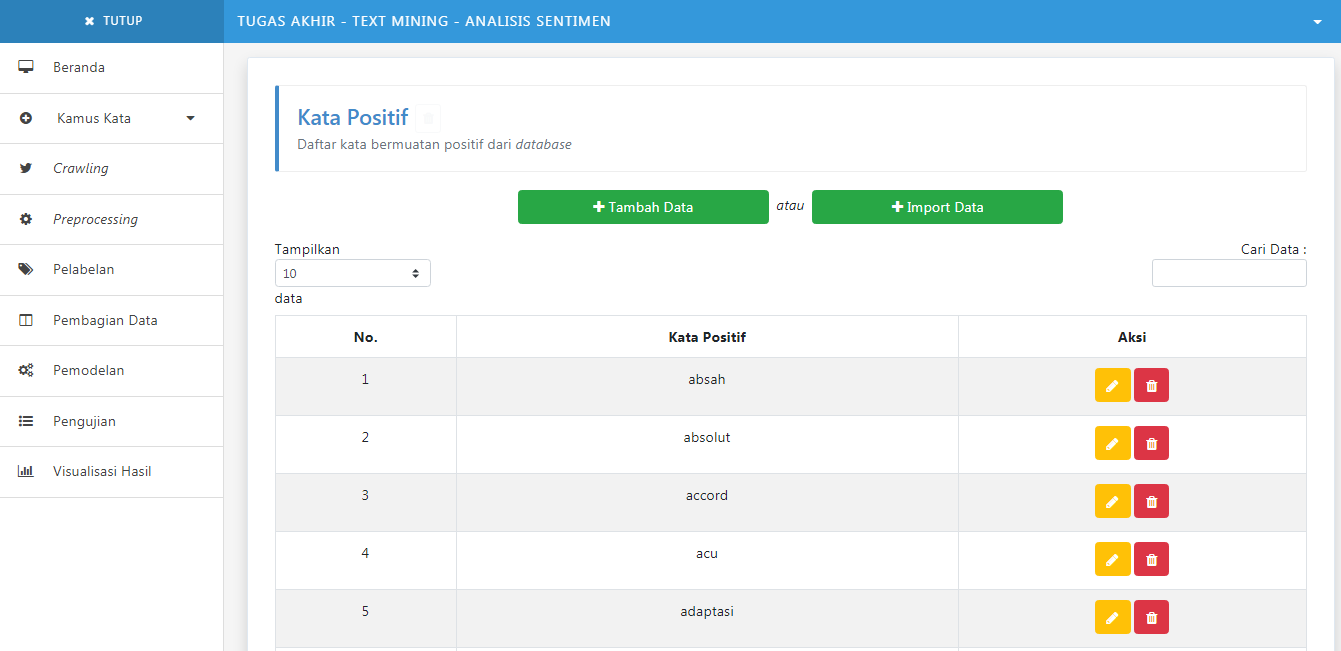
Tampilan layar kamus *stop word* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.39 berikut:



**Gambar 394.39 Tampilan layar kamus stop word**

#### Tampilan layar kamus kata positif

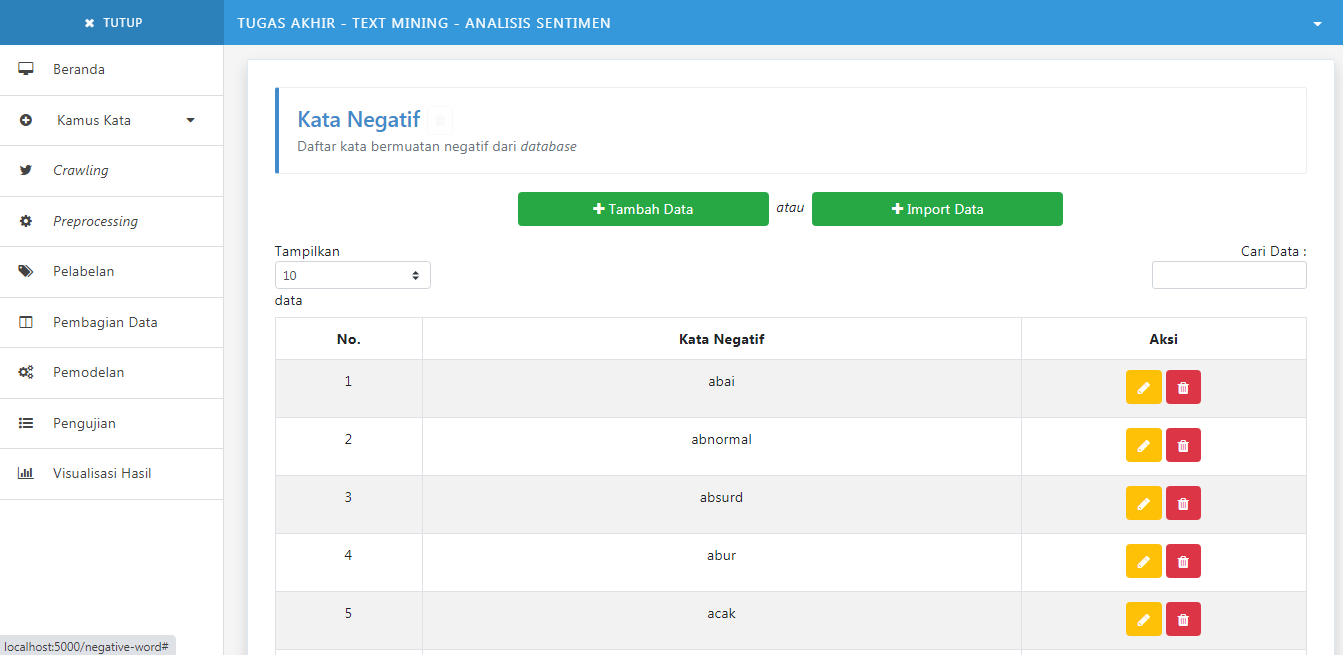
Tampilan layar kamus kata positif dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.40 berikut:



**Gambar 404.40 Tampilan layar kamus kata positif**

#### Tampilan layar kamus kata negatif

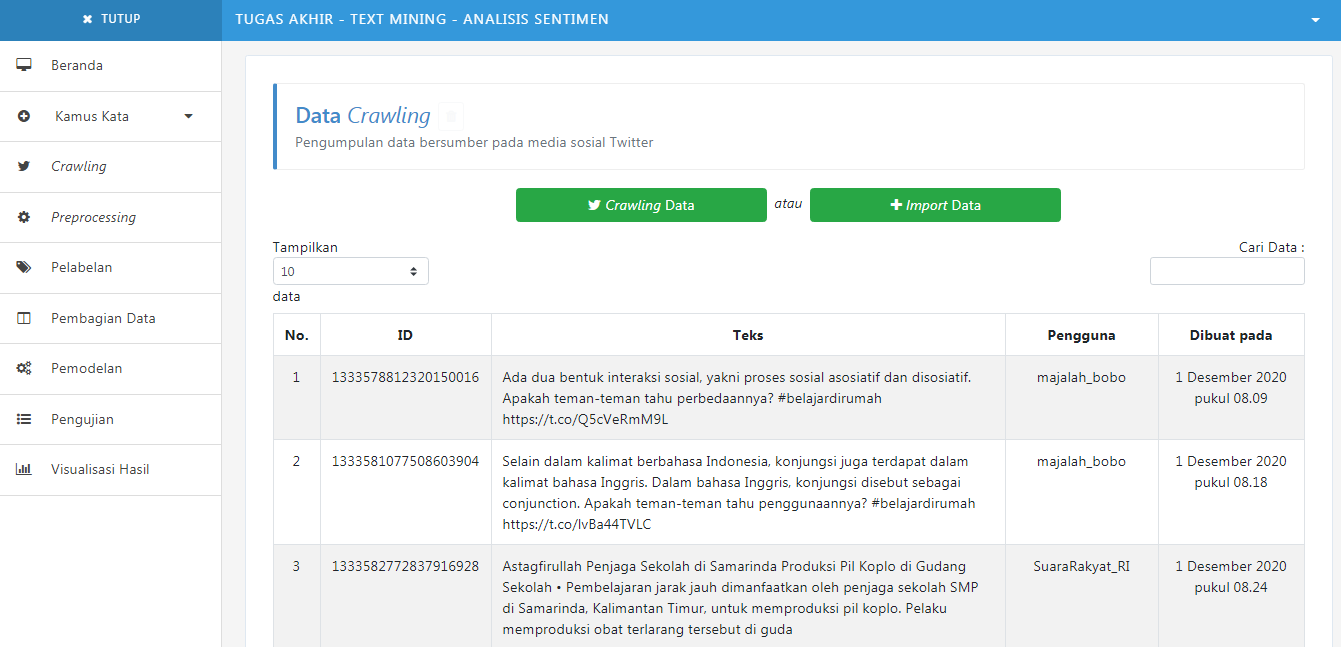
Tampilan layar kamus kata negatif dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.41 berikut:



**Gambar 414.41 Tampilan layar kamus kata negatif**

### Tampilan layar *crawling*

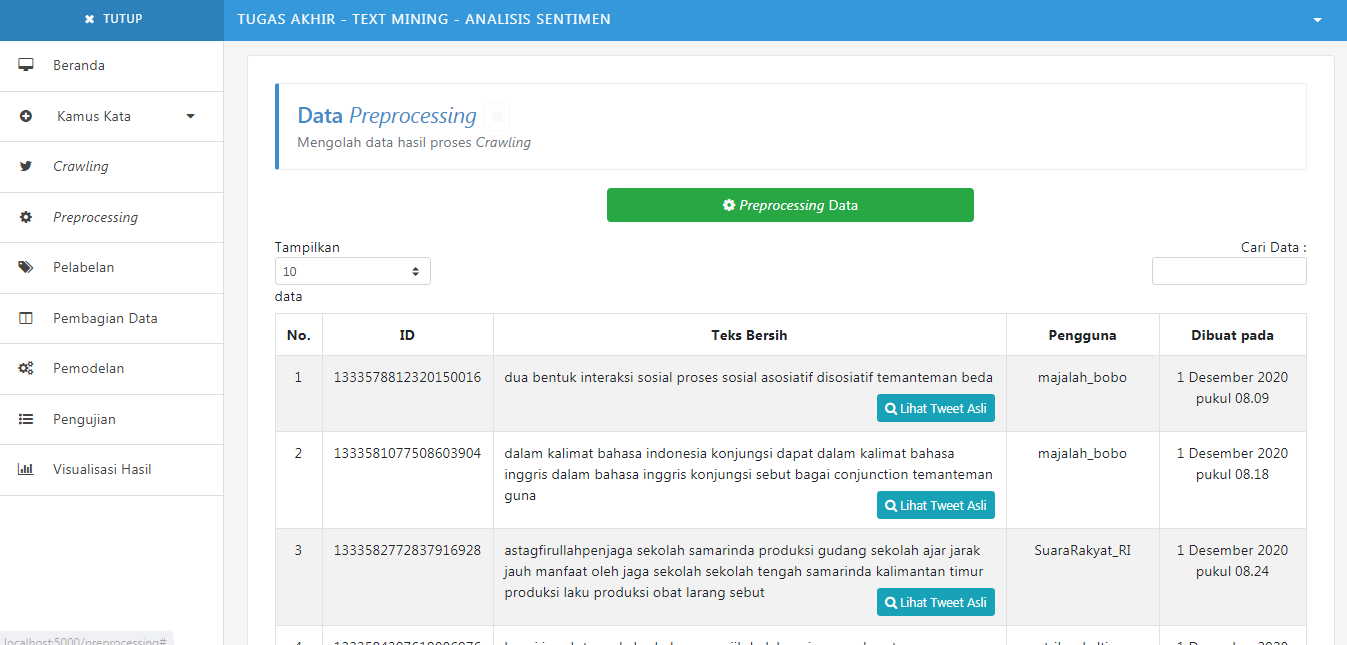
Tampilan layar *crawling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.42 berikut:



**Gambar 424.42 Tampilan layar crawling**

### Tampilan layar *preprocessing*

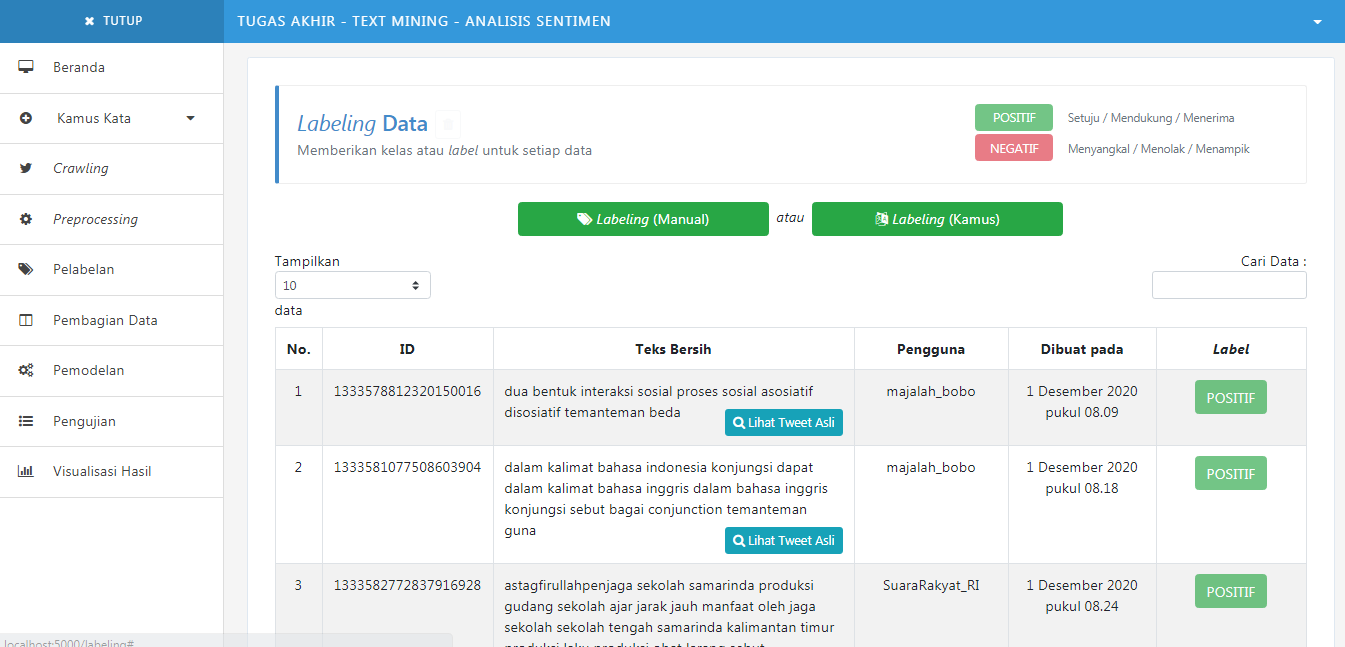
Tampilan layar *preprocessing* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.43 berikut:



**Gambar 434.43 Tampilan layar preprocessing**

### Tampilan layar *labeling*

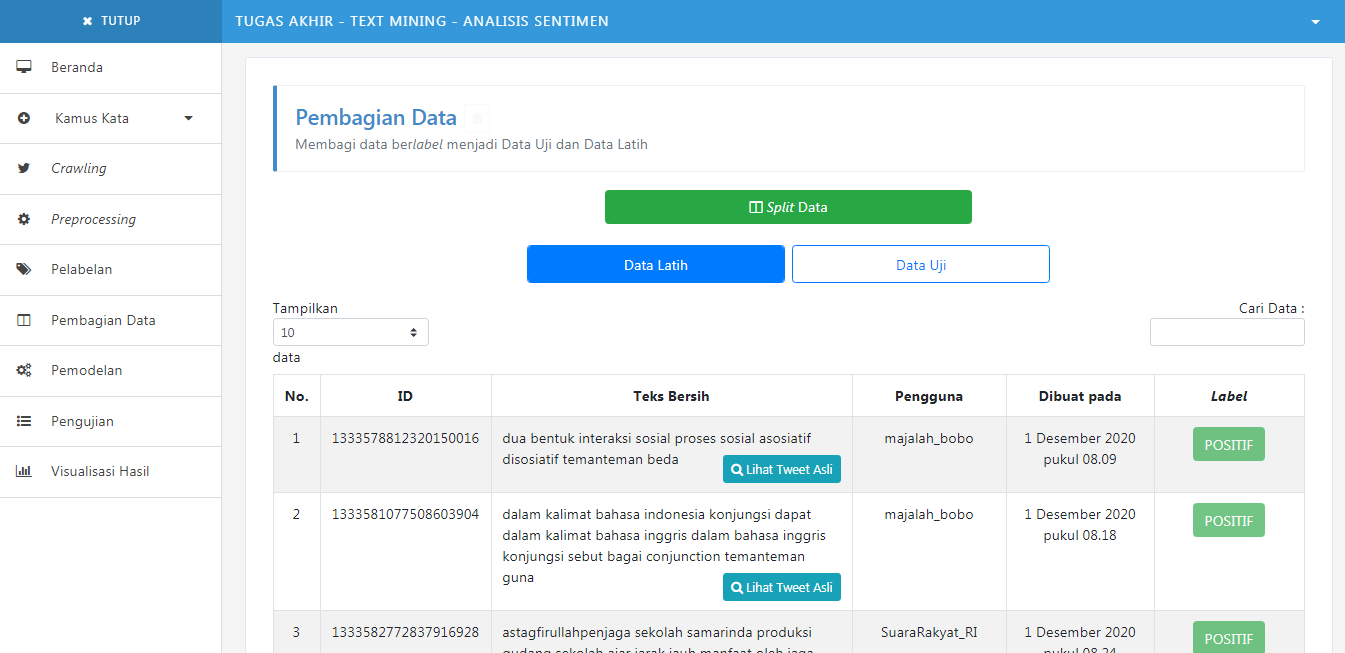
Tampilan layar *labeling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.44 berikut:



**Gambar 444.44 Tampilan layar labeling**

### Tampilan layar pembagian data

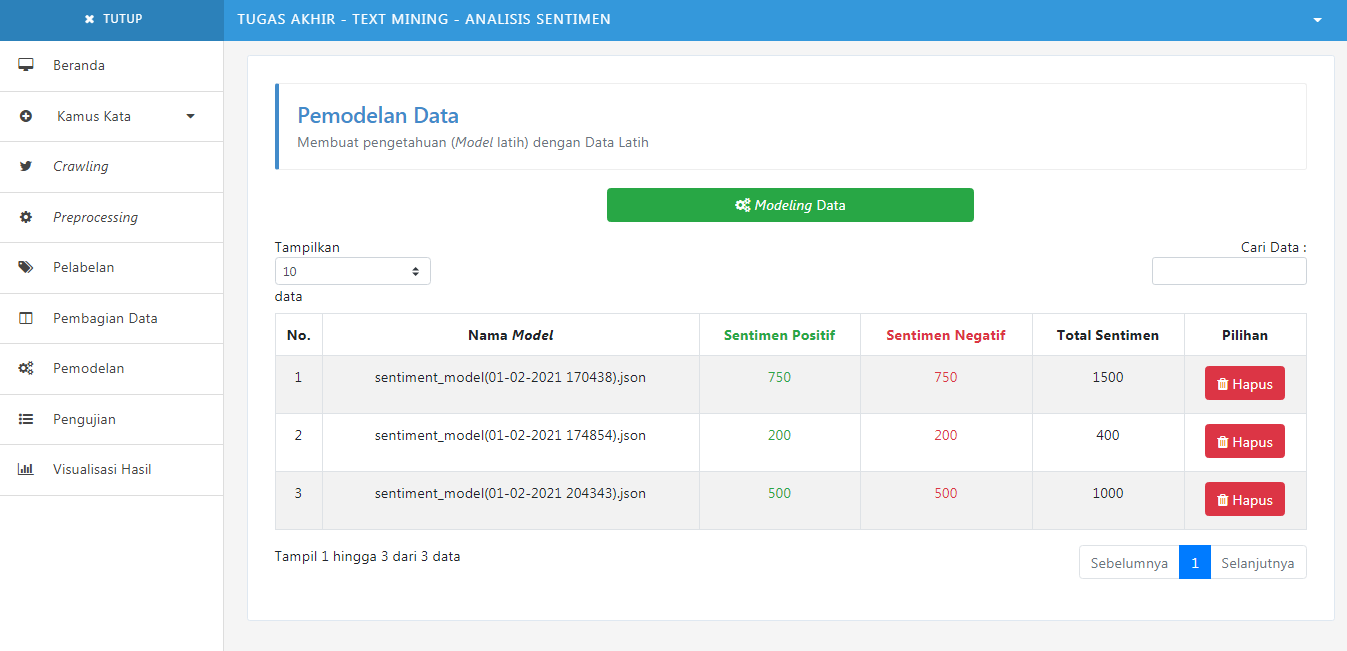
Tampilan layar pembagian data dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.45 berikut:



**Gambar 454.45 Tampilan layar pembagian data**

### Tampilan layar *modeling*

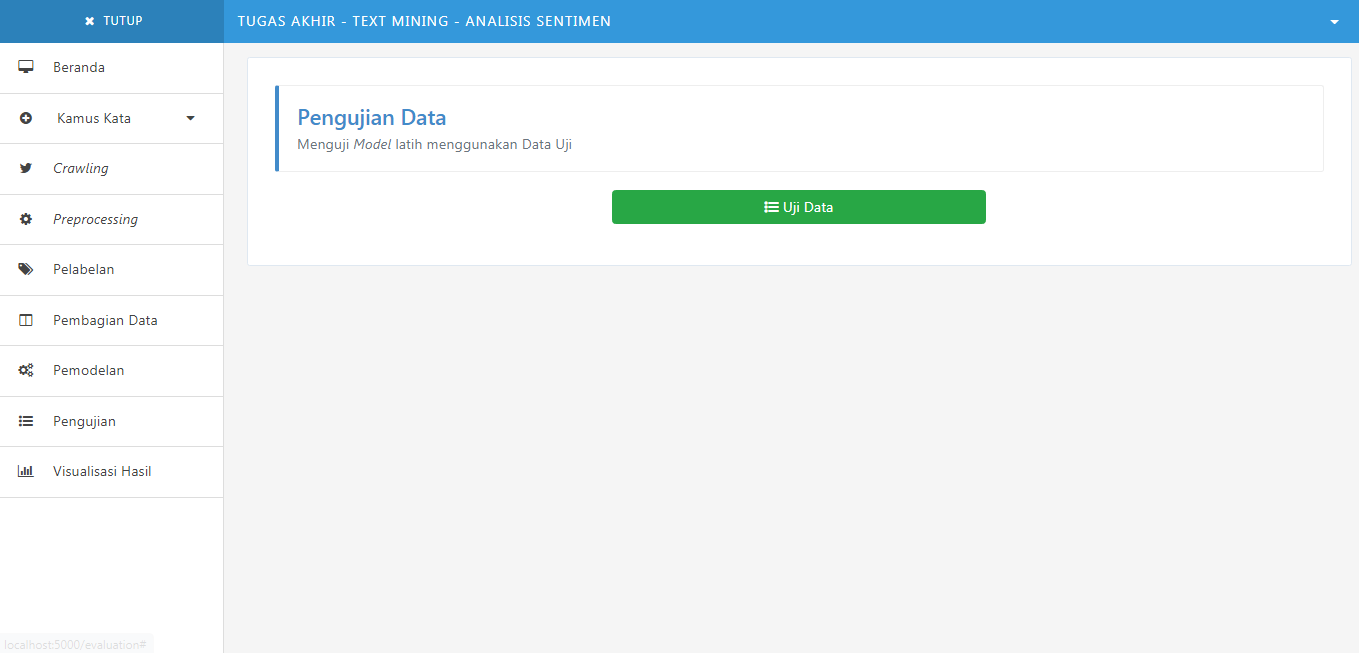
Tampilan layar *modeling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.46 berikut:



**Gambar 464.46 Tampilan layar modeling**

### Tampilan layar pengujian

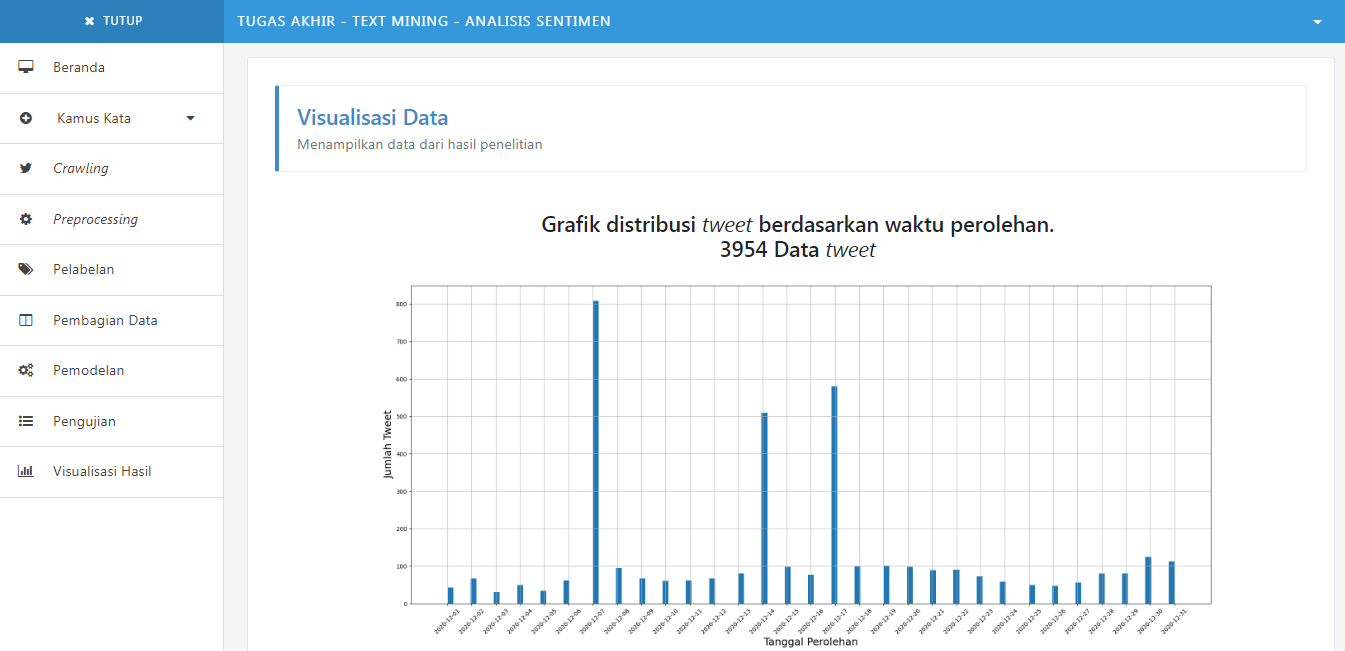
Tampilan layar pengujian dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.47 berikut:



**Gambar 474.47 Tampilan layar pengujian**

### Tampilan layar visualisasi hasil

Tampilan layar visualisasi hasil dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.48 berikut:



**Gambar 484.48 Tampilan layar visualisasi hasil**