# 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 dua (2) tahapan utama. Tahapan utama tersebut diproses secara berurutan, tahapan utama yang dimaksud antara lain: Tahapan ekstraksi fitur dan tahapan klasifikasi.

### 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. 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.

**Tabel 14.1 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 4.1 terdapat tiga (3) kolom yaitu: *Tweet* (*Tlatih-i*) yang berarti urutan *Tweet* latih ke-i, *Clean* *Text* yang berarti teks *tweet* yang telah terstuktur 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 4.1 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 4.2 berikut:

**Tabel 24.2 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 4.2 merupakan hasil dari proses pemisahan kata dari kolom *clean text* pada *tweet* berdasarkan pada Tabel 4.1 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 4.3 berikut:

**Tabel 34.3 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 4.3 merupakan hasil dari proses pencarian fitur berdasarkan pada Tabel 4.2 *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 4.1 Sampel data latih dan Tabel 4.3 Fitur kata, maka vektor kosong yang dihasilkan seperti pada Tabel 4.4 berikut:

**Tabel 44.4 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 4.3 Fitur kata dan Tabel 4.4 Vektor kosong, maka vektor kata yang dihasilkan seperti pada Tabel 4.5 berikut:

**Tabel 54.5 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 4.5 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 4.6 berikut:

**Tabel 64.6 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 4.6 terdapat empat (4) *key* yaitu: *teks\_list* yang berisi nilai pada kolom *Clean Text* dari *Tweet* latih ke-i pada Tabel 4.1 Sampel data latih, *label\_list* yang berisi nilai pada kolom *Sentiment Type* dari *Tweet* ke-i pada Tabel 4.1 secara berurutan, *feature\_list* yang berisi daftar fitur yang diperoleh dari Tabel 4.3 Fitur kata, *vector\_list* yang berisi nilai representasi vektor dari Tabel 4.5 Representasi vektor secara berurutan berdasarkan *Tweet* latih ke-i pada Tabel 4.1.

### Tahap klasifikasi *K-Nearest Neighbor*

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

#### Persiapan data

Proses persiapan data merupakan proses pemilihan file JSON yang tersedia dari hasil tahap *modeling* untuk dijadikan sebagai *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 4.7 berikut:

**Tabel 74.7 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. Bedasarkan jumlah fitur pada *model* latih dan Tabel 4.7 Sampel data uji, maka vektor kosong akan terbuat seperti pada Tabel 4.8 berikut:

**Tabel 84.8 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 terbuat seperti pada Tabel 4.9 berikut:

**Tabel 94.9 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 4.9 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 4.10 berikut:

**Tabel 104.10 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 ketetangga 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 4.10 Hasil jarak *euclidean distance* akan diurutkan secara urut menaik (*ascending*) berdasarkan jarak. Sehingga hasil urutan dapat diperoleh seperti pada Tabel 4.11 berikut:

**Tabel 114.11 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 4.11 Pengurutan jarak tetangga akan di ambil sebanyak K buah data, dengan nilai K yang telah ditentukan dan dipilih. Dengan nilai K=3, sehingga diperoleh hasil tetangga terdekat seperti Tabel 4.12 berikut:

**Tabel 124.12 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 4.12 Data K tetangga terdekat, bahwa nilai probabilitas yang dihasilkan adalah seperti pada Tabel 4.13 berikut:

**Tabel 134.13 Nilai probabilitas data uji**

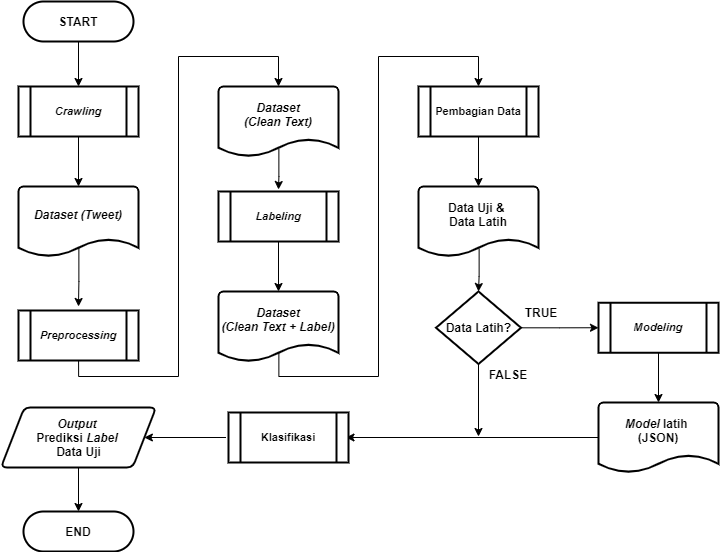
| *Tweet* (*T*uji-*i*) | *Tweet*  (*T latih-i*) | *Sentiment Type* | Probabilitas positif | Probabilitas negatif |
| --- | --- | --- | --- | --- |
| uji-1 | latih-1 | positif | 1 | 0 |
| latih-3 | positif | 1 | 0 |
| latih-5 | negatif | 0 | 1 |
| Jumlah | | | 2 (0.667) | 1 (0.333) |
|  |  |  |  |  |
| 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 4.13 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

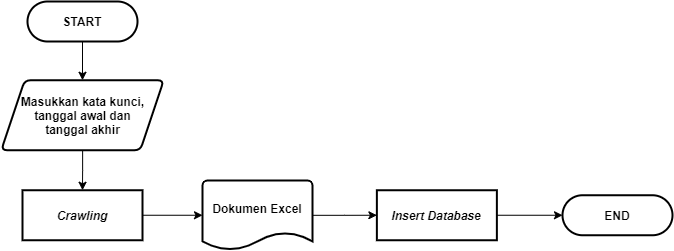


**Gambar 14.1 Flowchart keseluruhan proses sistem**

Pada Gambar 4.1 *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, lalu tahap *modeling* menggunakan *model* latih untuk menghasilkan sebuah *model* latih yang kemudian akan diuji pada tahap klasifikasi menggunakan data uji yang tersedia. Hasil dari tahapan tersebut akan berupa *label* sentimen prediksi untuk setiap data uji.

### *Flowchart* proses *crawling*

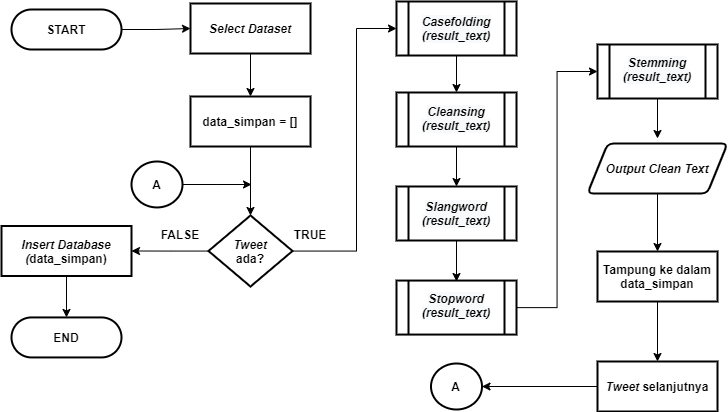
Pada *flowchart* ini, menjelaskan proses pengumpulan data atau *crawling* data tweet dimulai dari memasukkan kata kunci dan tanggal awal dan tanggal akhir, cara lainya adalah dengan fitur import file *excel*, kemudian dilakukan pencarian *tweet* bredasarkan parameter menggunakan pustaka Tweepy. Hasil pengumpulan data akan disimpan dalam bentuk file Excel (.xlsx) sebelum kemudian dimasukkan ke dalam *database*. *Flowchart* proses *crawling* dapat dilihat pada Gambar 4.2 berikut:



**Gambar 24.2 Flowchart proses crawling**

### *Flowchart* proses *preprocessing*

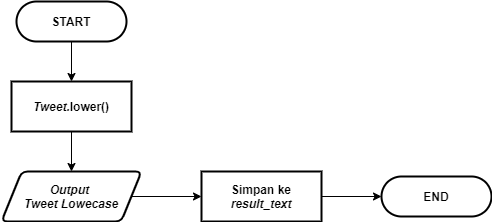
Pada *flowchart* ini, menjelaskan proses pengubahan data pengubahan data teks *tweet* menjadi terstruktur atau setara, guna mendukung proses klasifikasi agar berjalan dengan baik. Hasil proses ini berupa teks bersih, yang kemudian akan disimpan ke dalam *database*  untuk proses selanjutnya. *Flowchart* proses *preprocessing* dapat dilihat pada Gambar 4.3 berikut:



**Gambar 34.3 Flowchart proses preprocessing**

#### Flowchart proses casefolding

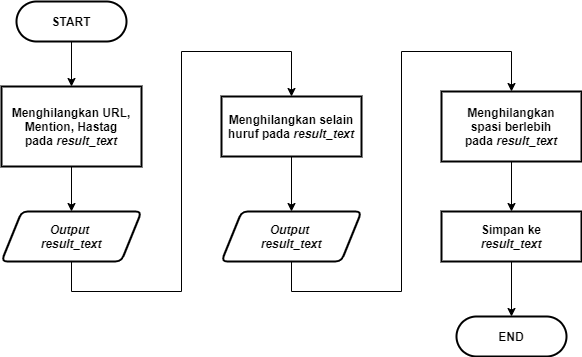
Pada *flowchart* ini dilakukan proses pengubahan teks *tweet* menjadi huruf kecil secara keseluruhan. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result\_text* untuk proses selanjutnya. *Flowchart* proses *casefolding* dapat dilihat pada Gambar 4.4 berikut:



**Gambar 44.4 Flowchart proses casefolding**

#### Flowchart proses cleansing

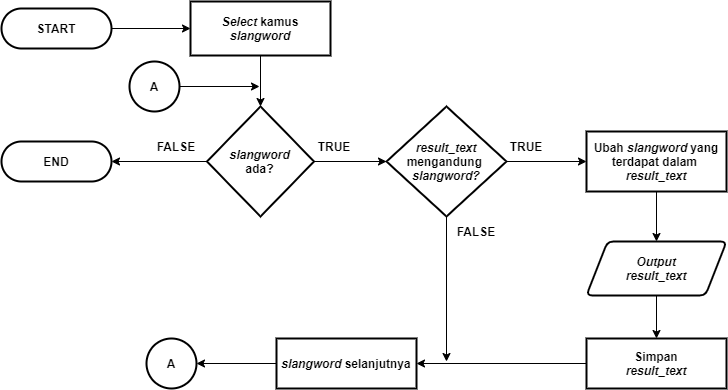
Pada *flowchart* ini dilakukan proses penyaringan dan pembuangan attribut pada teks *result*\_*text*, diantaranya penghapusan atau penghilangan URL, *mention*, *hastag*, selain huruf dan spasi atau baris berlebih. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *cleansing* dapat dilihat pada Gambar 4.5 berikut:



**Gambar 54.5 Flowchart proses cleansing**

#### Flowchart proses slangword

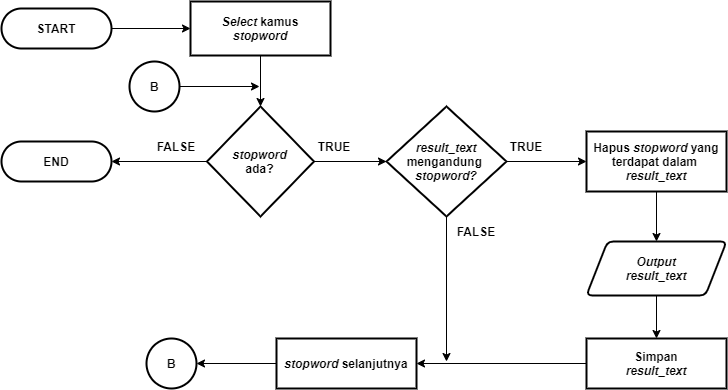
Pada *flowchart* ini dilakukan pengubahan attribut pada teks *result*\_*text* yang mengandung kata *slang* dari kamus *slangword* ke bentuk kata asli atau kata bakunya. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *slangword* dapat dilihat pada Gambar 4.6 berikut:



**Gambar 64.6 Flowchart proses slangword**

#### Flowchart proses stopword

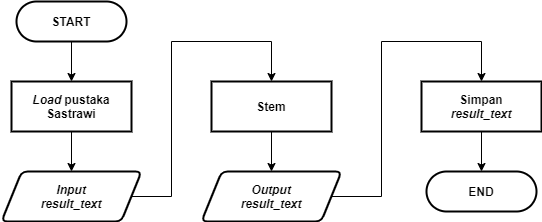
Pada *flowchart* ini dilakukan proses penghapusan attribut pada teks *result*\_*text* yang mengandung *stopword* dari kamus *stopword* karena dianggap kurang memiliki makna untuk proses klasifikasi. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya. *Flowchart* proses *stopword* dapat dilihat pada Gambar 4.7 berikut:



**Gambar 74.7 Flowchart proses stopword**

#### Flowchart proses stemming

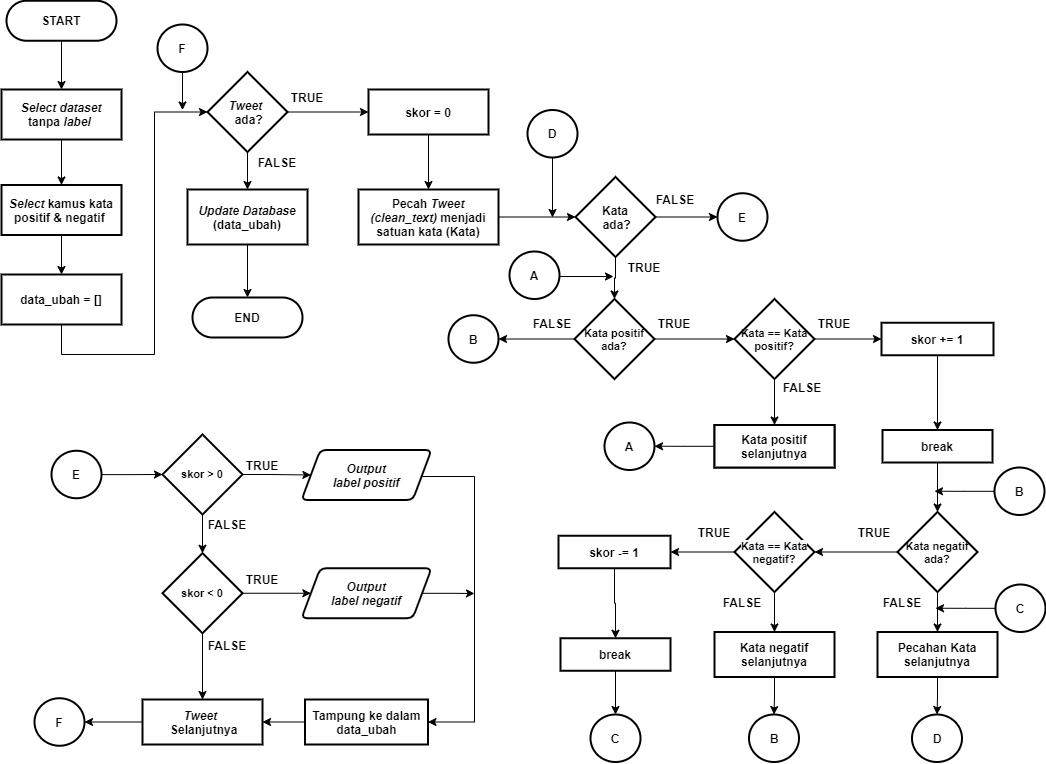
Pada *flowchart* ini dilakukan pengubahan attribut pada teks *result*\_*text* yang mengandung kataberimbuhan ke bentuk kata asalnya, menggunakan pustaka Sastrawi. *Flowchart* proses *stemming* dapat dilihat pada Gambar 4.8 berikut:



**Gambar 84.8 Flowchart proses stemming**

### *Flowchart* proses *labeling*

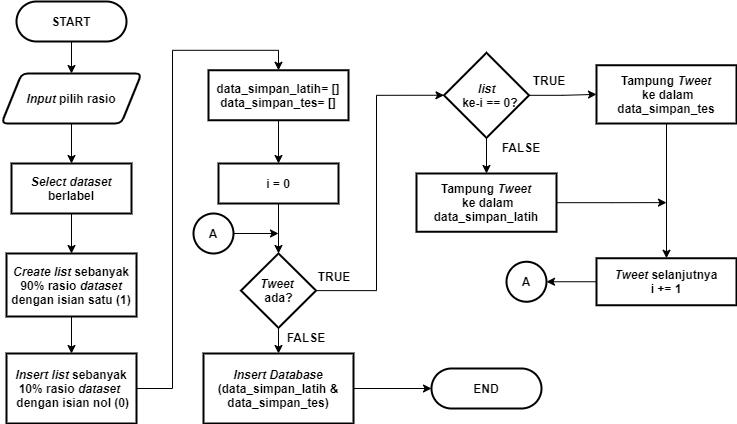
Pada *flowchart* ini, menjelaskan proses *labeling* yang dilakukan dengan cara pendekatan kamus sentimen, di mana diawali dengan perhitungan skor menggunakan kamus positif dan negatif, lalu penentuan kelas atau *label* berdasarkan nilai skor. *Flowchart* proses *labeling* dapat dilihat pada Gambar 4.9 berikut:



**Gambar 94.9 Flowchart proses labeling**

### *Flowchart* proses pembagian data

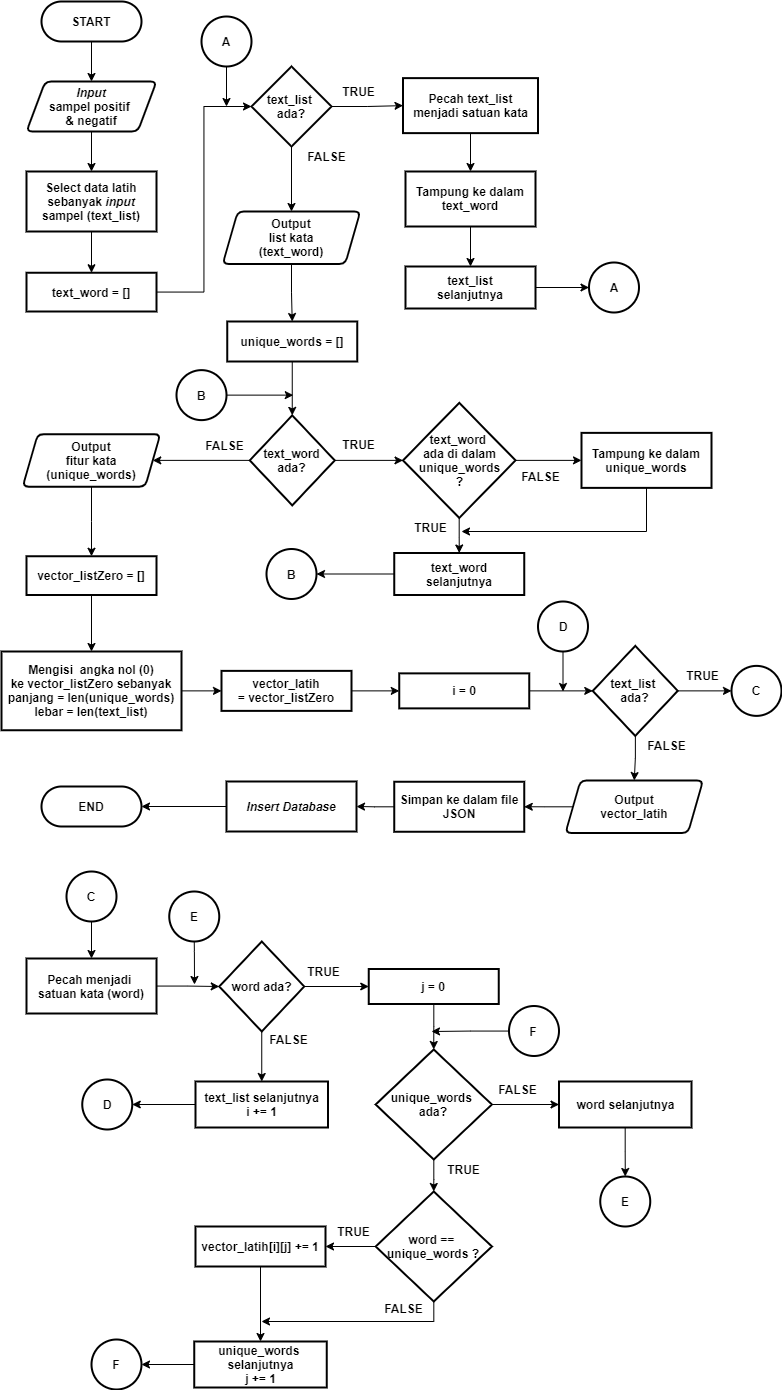
Pada *flowchart* ini, menjelaskan proses pembagian data ke dua (2) buah bagian, yaitu: data uji dan data latih, menggunakan rasio 1:9 (data uji : data latih). *Flowchart* proses pembagian datadapat dilihat pada Gambar 4.10 berikut:



**Gambar 104.10 Flowchart proses pembagian data**

### *Flowchart* proses *modeling*

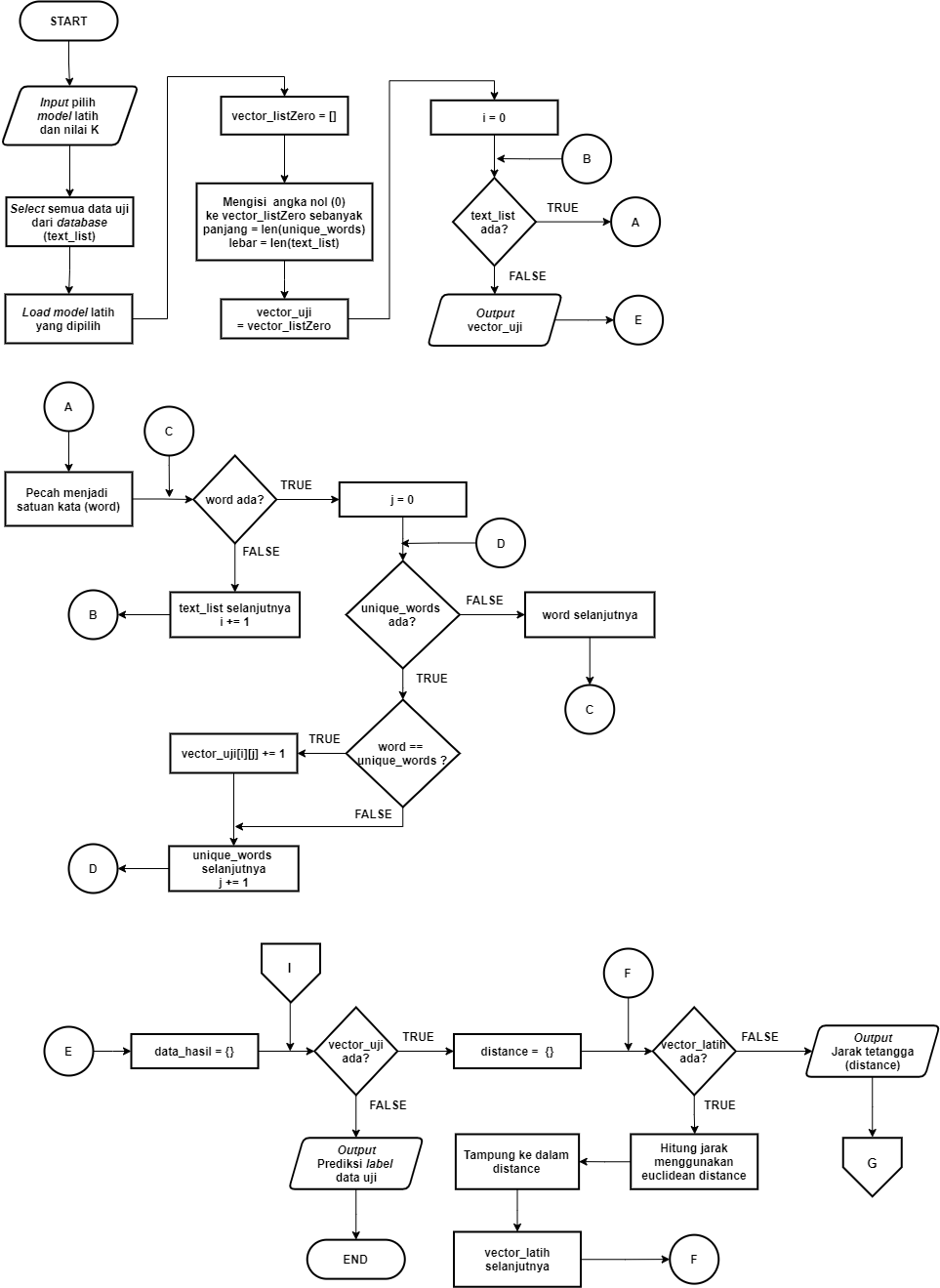
Pada *flowchart* ini, menjelaskan proses ekstraksi fitur dengan *CountVectorizer*. Dimulai dari proses seleksi data latih,pembuatan list kata, pencarian fitur kata, pembuatan vektor kosong, dan pembuatan vektor kata. Sehingga didapatkan hasil berupa *model* latih dengan format JSON. *Flowchart* proses *modeling* dapat dilihat pada Gambar 4.11 berikut:

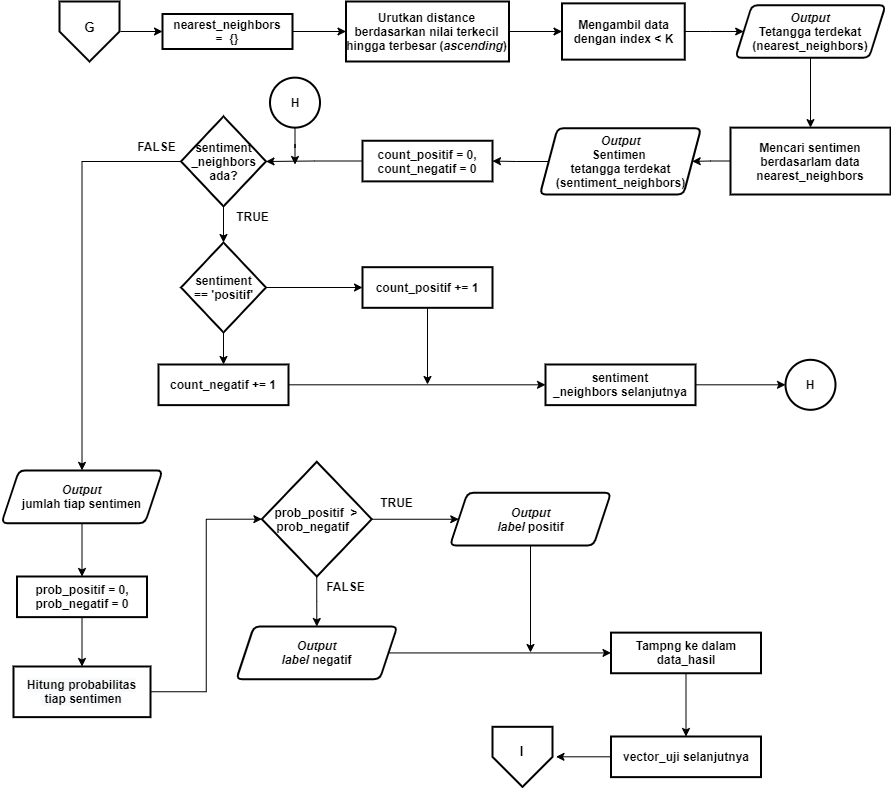


**Gambar 114.11 Flowchart proses modeling**

### *Flowchart* proses klasifikasi

Pada *flowchart* ini, menjelaskan proses klasifikasi dengan *K-Nearest Neighbor*. Dimulai dari proses membuatan vektor uji, menghitung jarak antar data, mencari tetangga terdekat, dan menghitung nilai probabilitas. *Flowchart* proses klasifikasidapat dilihat pada Gambar 4.12 berikut:

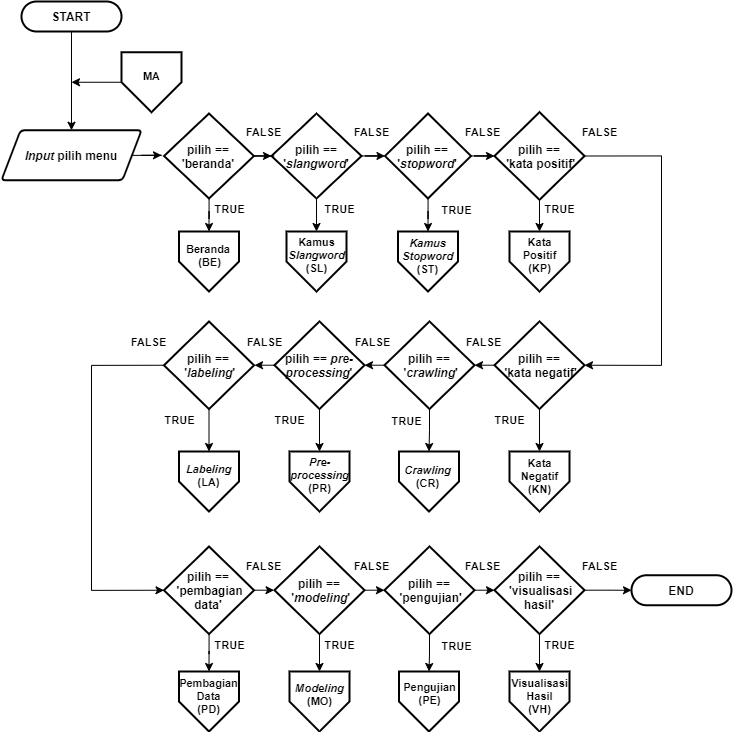




**Gambar 124.12 Flowchart proses klasifikasi**

### *Flowchart* keseluruhan menu aplikasi

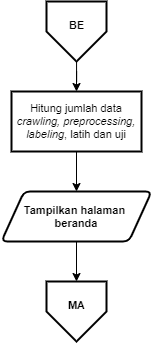
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya aplikasi pertampilan layar. *Flowchart* keseluruhan menu aplikasidapat dilihat pada Gambar 4.13 berikut:



**Gambar 134.13 Flowchart keseluruhan menu aplikasi**

### *Flowchart* menu beranda

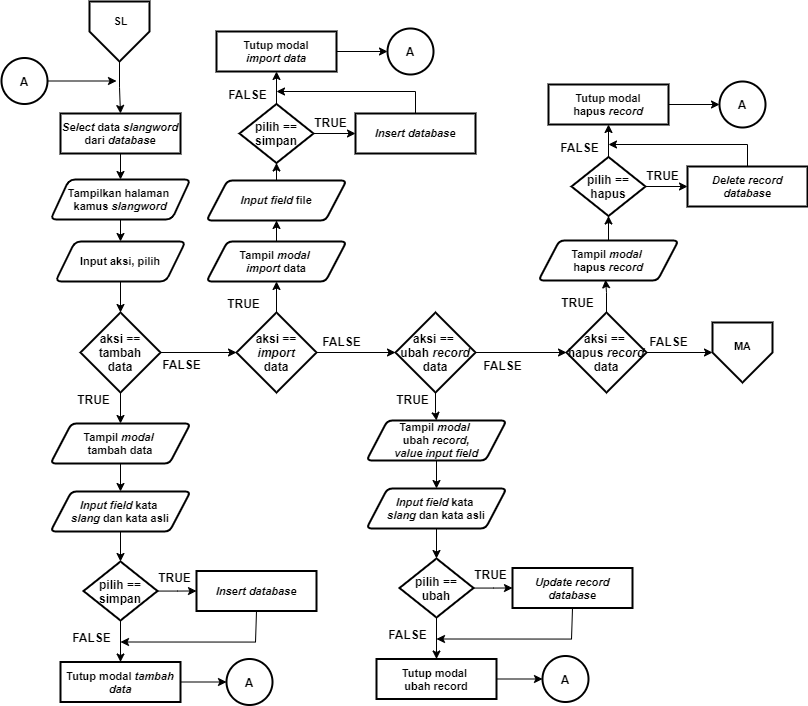
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu beranda. *Flowchart* menu beranda dapat dilihat pada Gambar 4.14 berikut:



**Gambar 144.14 Flowchart menu beranda**

### *Flowchart* menu *slangword*

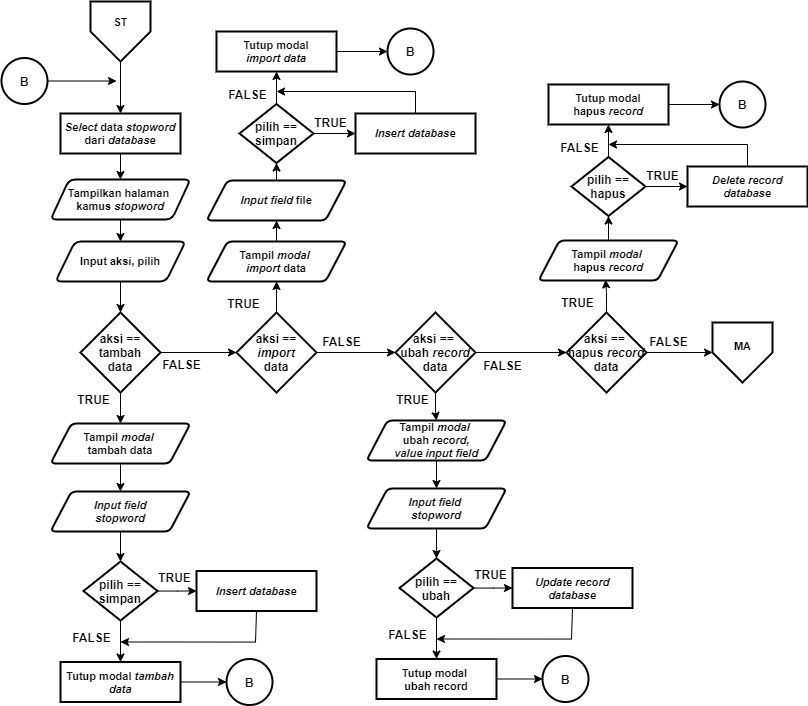
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *slangword*. *Flowchart* menu *slangword* dapat dilihat pada Gambar 4.15 berikut:



**Gambar 154.15 Flowchart menu slangword**

### *Flowchart* menu *stopword*

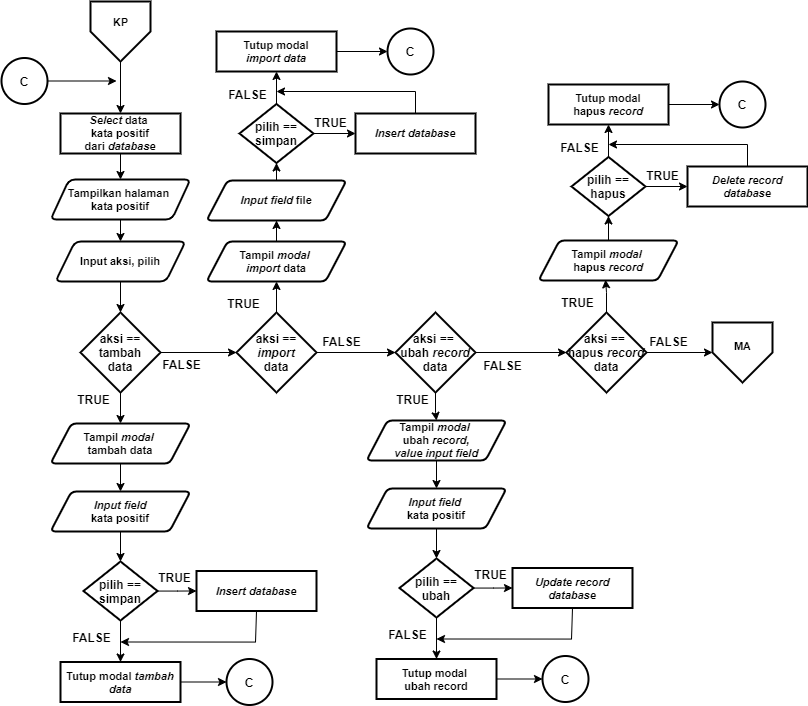
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *stopword*. *Flowchart* menu *stopword* dapat dilihat pada Gambar 4.16 berikut:



**Gambar 164.16 Flowchart menu stopword**

### *Flowchart* menu kata positif

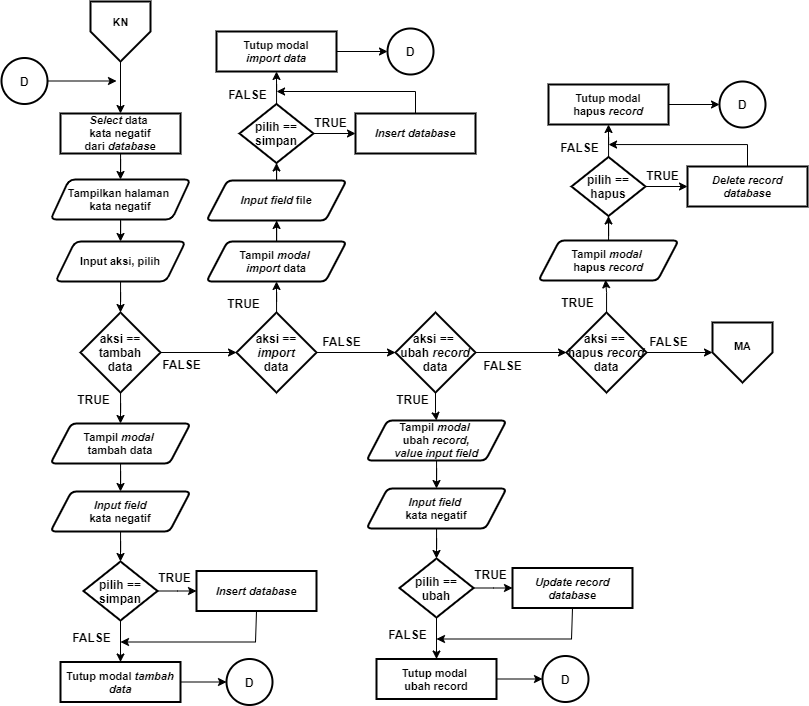
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu kata positif. *Flowchart* menu kata positif dapat dilihat pada Gambar 4.17 berikut:



**Gambar 174.17 Flowchart menu kata positif**

### *Flowchart* menu kata negatif

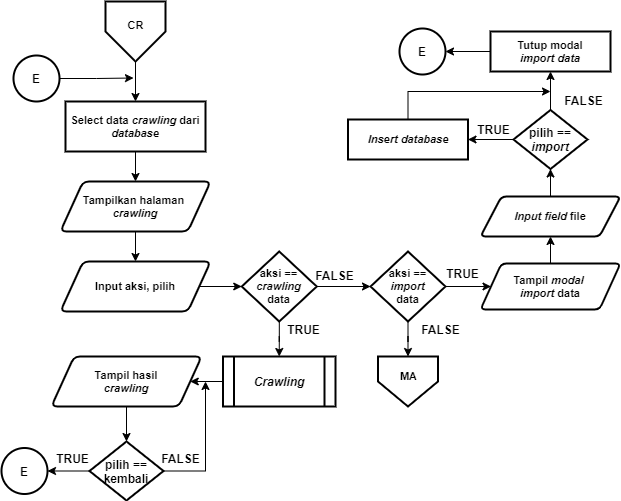
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu kata negatif. *Flowchart* menu kata negatif dapat dilihat pada Gambar 4.18 berikut:



**Gambar 184.18 Flowchart menu kata negatif**

### *Flowchart* menu *crawling*

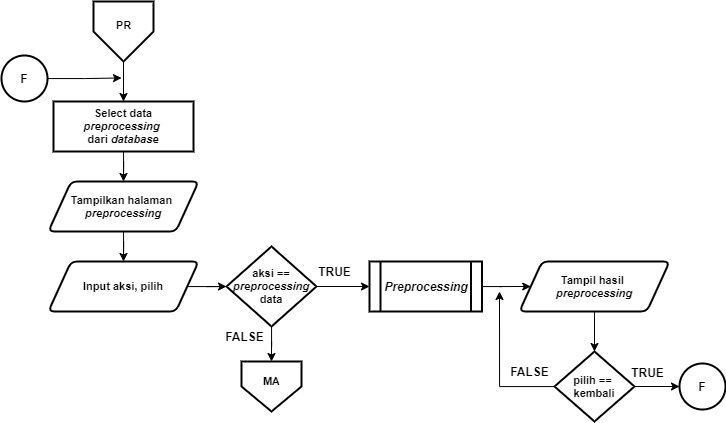
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *crawling*. *Flowchart* menu *crawling* dapat dilihat pada Gambar 4.19 berikut:



**Gambar 194.19 Flowchart menu crawling**

### *Flowchart* menu *preprocessing*

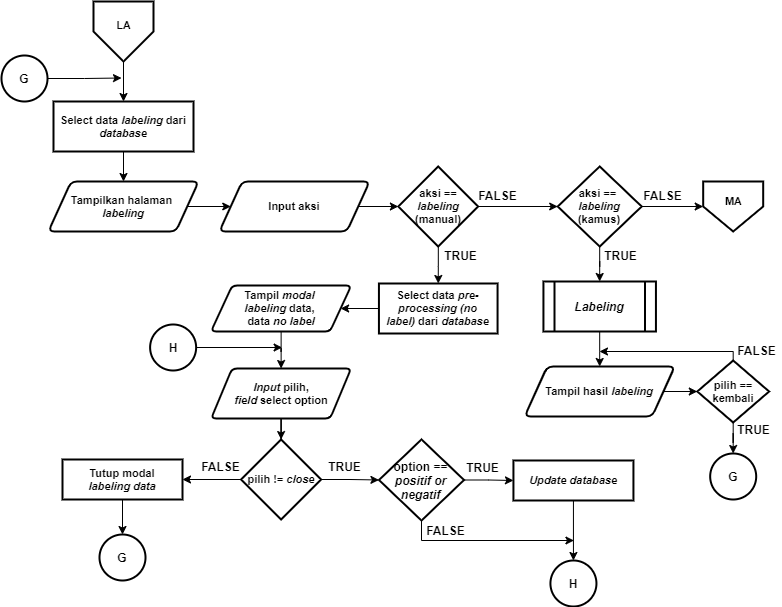
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *preprocessing*. *Flowchart* menu *preprocessing* dapat dilihat pada Gambar 4.20 berikut:



**Gambar 204.20 Flowchart menu preprocessing**

### *Flowchart* menu *labeling*

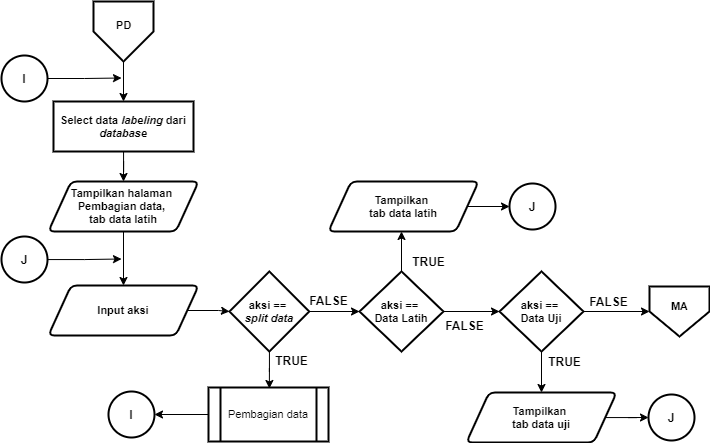
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *labeling*. *Flowchart* menu *labeling* dapat dilihat pada Gambar 4.21 berikut:



**Gambar 214.21 Flowchart menu labeling**

### *Flowchart* menu pembagian data

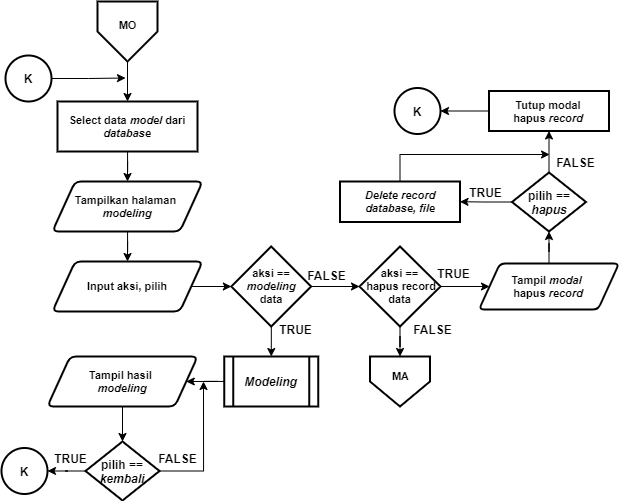
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu pembagian data. *Flowchart* menu pembagian data dapat dilihat pada Gambar 4.22 berikut:



**Gambar 224.22 Flowchart menu pembagian data**

### *Flowchart* menu *modeling*

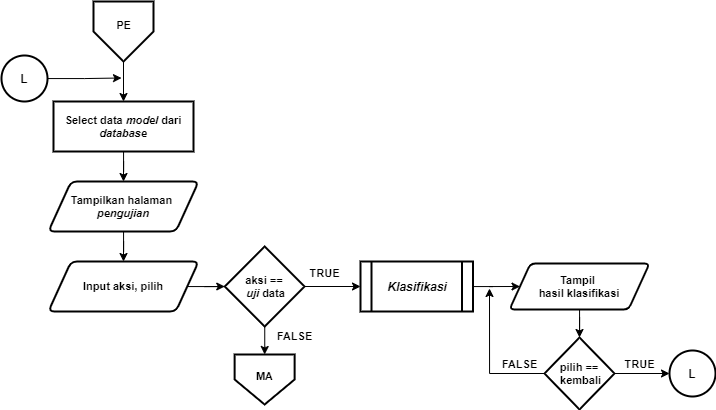
Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu *modeling*. *Flowchart* menu *modeling* dapat dilihat pada Gambar 4.23 berikut:



**Gambar 234.23 Flowchart menu modeling**

### *Flowchart* menu pengujian

Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu pengujian. *Flowchart* menu pengujian dapat dilihat pada Gambar 4.24 berikut:



**Gambar 244.24 Flowchart menu pengujian**

### *Flowchart* menu visualisasi hasil

Pada *flowchart* ini, menjelaskan mengenai proses berjalannya menu visualisasi hasil. *Flowchart* menu visualisasi hasil dapat dilihat pada Gambar 4.25 berikut:



**Gambar 254.25 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 14.1 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 if (Data Latih)  11 Lakukan proses Modeling  12 Simpan Model ke file JSON  13 endif  14 Lakukan proses Klasifikasi  15 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 24.2 Algoritme proses crawling**

|  |
| --- |
| 1 start  2 Input kata kunci, tanggal awal, dan tanggal  akhir  3 Lakukan proses Crawling  4 Baca dokumen Excel  5 Simpan ke dalam database  6 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 34.3 Algoritme proses preprocessing**

|  |
| --- |
| 1 start  2 Select Dateset (Tweet)  3 data\_simpan = []  4 if (Tweet)  5 Lakukan proses Casefolding (result\_text)  6 Lakukan proses Cleansing (result\_text)  7 Lakukan proses Slangword (result\_text)  8 Lakukan proses Stopword (result\_text)  9 Lakukan 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 casefolding

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

**Algoritme 44.4 Algoritme proses casefolding**

|  |
| --- |
| 1 start  2 Baca Dateset (Tweet)  3 Ubah ke huruf kecil  4 Simpan result\_text  5 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 54.5 Algoritme proses cleansing**

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

#### Algoritme proses slangword

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

**Algoritme 64.6 Algoritme proses slangword**

|  |
| --- |
| 1 start  2 Select kamus slangword dari database  3 if (slangword)  4 if (result\_text in 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 stopword

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

**Algoritme 74.7 Algoritme proses stopword**

|  |
| --- |
| 1 start  2 Select kamus stopword dari database  3 if (stopword)  4 if (result\_text in stopword)  5 Hapus result\_text (stopword) dengan  kata asli  6 Output result\_text  7 Simpan result\_text  8 else  9 stopword selanjutnya  10 Kembali ke nomor 3  11 endif  12 endif  13 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 84.8 Algoritme proses stemming**

|  |
| --- |
| 1 start  2 Load pustaka Sastrawi  3 Input result\_text ke pustaka Sastrawi  4 Lakukan fungsi stem()  5 Output result\_text  6 Simpan result\_text  7 end |

### Algoritme proses *labeling*

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

**Algoritme 94.9 Algoritme proses labeling**

|  |
| --- |
| 1 start  2 Select dataset tanpa label dari database  3 Select kamus sentimen positif dan negatif dari  database  4 data\_ubah = []  5 if (Tweet)  6 skor = 0  7 Split Tweet menjadi satuan kata (Kata)  8 if (Kata)  9 if (Kata\_positif)  10 if (Kata == Kata\_positif)  11 skor = skor + 1  12 break (Langsung ke nomor 18)  13 else  14 Kata\_positif selanjutnya  15 Kembali ke nomor 9  16 endif  17 endif  18 if (Kata\_negatif)  19 if (Kata == Kata\_negatif)  20 skor = skor + 1  21 break (Langsung ke nomor 27)  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 else  33 Output label negatif  34 endif  35 Simpan ke dalam list data\_ubah  36 Tweet selanjutnya  37 Kembali ke nomor 5  38 endif  39 else  40 Update database data\_ubah  41 endif  42 end |

### 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 104.10 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)  10 if(list[i] == 0)  11 Simpan Tweet ke list data\_simpan\_tes  12 else  13 Simpan Tweet ke list 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 114.11 Algoritme proses modeling**

|  |
| --- |
| 1 start  2 Input sampel positif dan negatif  3 Select data latih sebanyak input sampel  4 text\_word = []  5 if (text\_list)  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)  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])  28 Split text\_list[i] menjadi satuan  kata (word)  29 if (word)  30 j = 0  31 if (unique\_words[j])  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 ke dalam file JSON (.json)  48 Simpan 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 124.12 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 penyimpanan  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])  10 Split text\_list menjadi satuan kata (word)  11 if (word)  12 j = 0  13 if (unique\_words[j])  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)  31 distance = {}  32 if(vector\_latih)  33 Hitung jarak dengan euclidean  distance  34 Simpan hasil jarak ke dalam dict  distance  35 vector\_latih selanjutnya  36 Kembali ke nomor 32  37 else  38 Output distance  39 nearest\_neighbors = {}  40 Urut data menjadi 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)  48 if (sentimen == 'positif')  49 count\_positif = 0  50 else  51 count\_negatif = 0  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 per data uji  72 endif  73 endif  74 end |

### Algoritme keseluruhan menu aplikasi

Pada algoritme ini dijelaskan tentang keseluruhan menu yang ada berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme keseluruhan menu aplikasi dapat dilihat pada Algoritme 4.13 berikut:

**Algoritme 134.13 Algoritme keseluruhan menu aplikasi**

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

### Algoritme menu beranda

Pada algoritme ini dijelaskan tentang menu beranda berdasarkan *flowchart* yang dibuat sebelumnya. Algoritme menu beranda dapat dilihat pada Algoritme 4.14 berikut:

**Algoritme 144.14 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 *slangword*

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

**Algoritme 154.15 Algoritme menu slangword**

|  |
| --- |
| 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 == 'simpan')  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 *stopword*

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

**Algoritme 164.16 Algoritme menu beranda**

|  |
| --- |
| 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 == 'simpan')  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 4.17 berikut:

**Algoritme 174.17 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 == 'simpan')  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 4.18 berikut:

**Algoritme 184.18 Algoritme menu beranda**

|  |
| --- |
| 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 == 'simpan')  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 4.19 berikut:

**Algoritme 194.19 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 4.20 berikut:

**Algoritme 204.20 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 4.21 berikut:

**Algoritme 214.21 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 4.22 berikut:

**Algoritme 224.22 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 4.23 berikut:

**Algoritme 234.23 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 4.24 berikut:

**Algoritme 244.24 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 4.25 berikut:

**Algoritme 254.25 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 mengklasfikasikan atau prediksi *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 4.14 berikut:

**Tabel 144.14 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 4.14, 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 4.15 berikut:

**Tabel 154.15 Confusion matrix pengujian K=3**

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

Berdasarkan Tabel 4.15, 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 4.16 berikut:

**Tabel 164.16 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 4.17 berikut:

**Tabel 174.17 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 4.17, 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%.

## Tampilan Layar Aplikasi

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

### Tampilan layar beranda

Tampilan layar beranda dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.26 berikut:

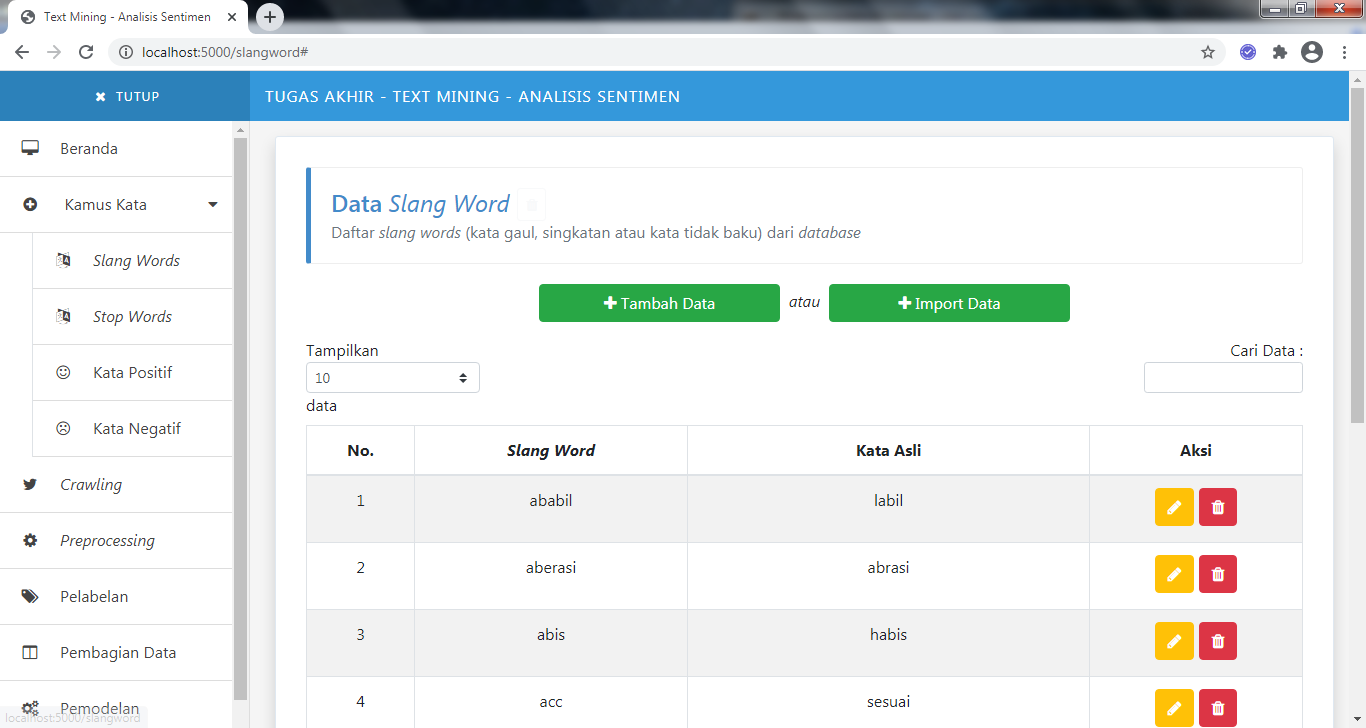


**Gambar 26.4.26 Tampilan layar beranda**

### Tampilan layar kamus kata

#### Tampilan layar kamus slangword

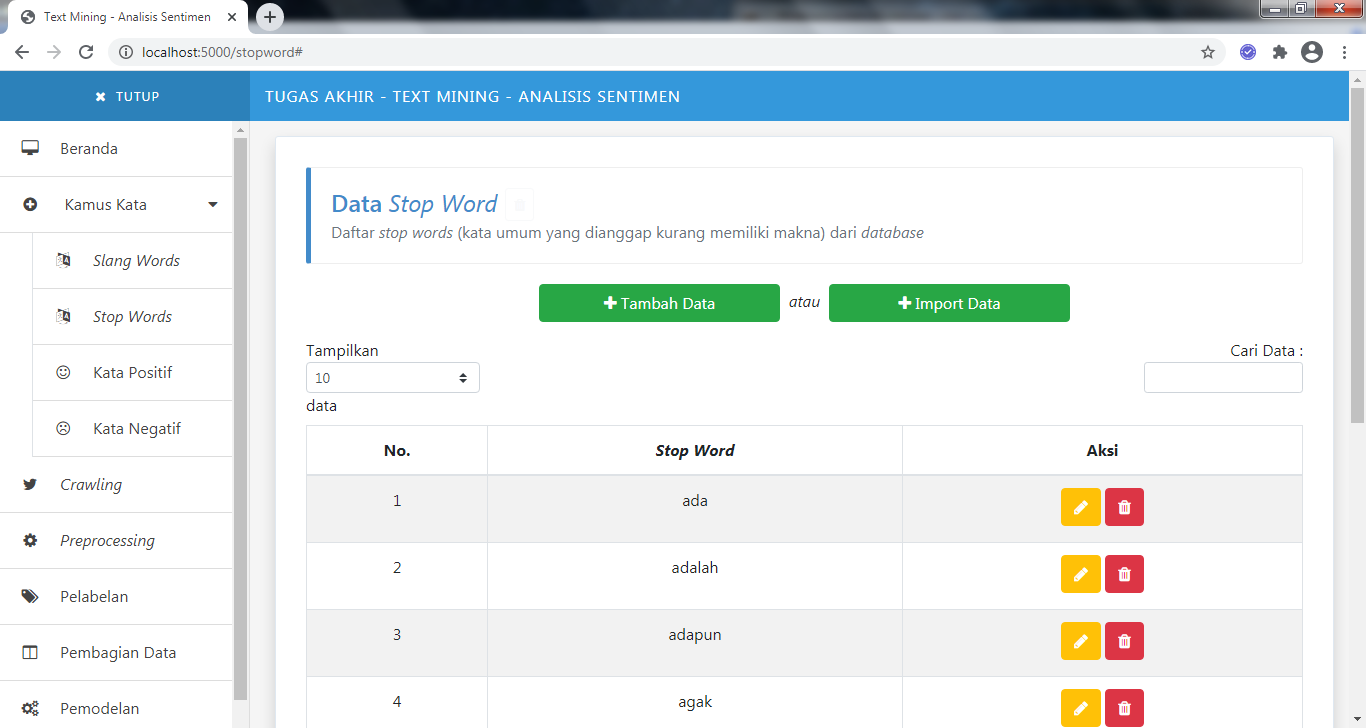
Tampilan layar kamus *slangword* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.27 berikut:



**Gambar 274.27 Tampilan layar kamus slangword**

#### Tampilan layar kamus stopword

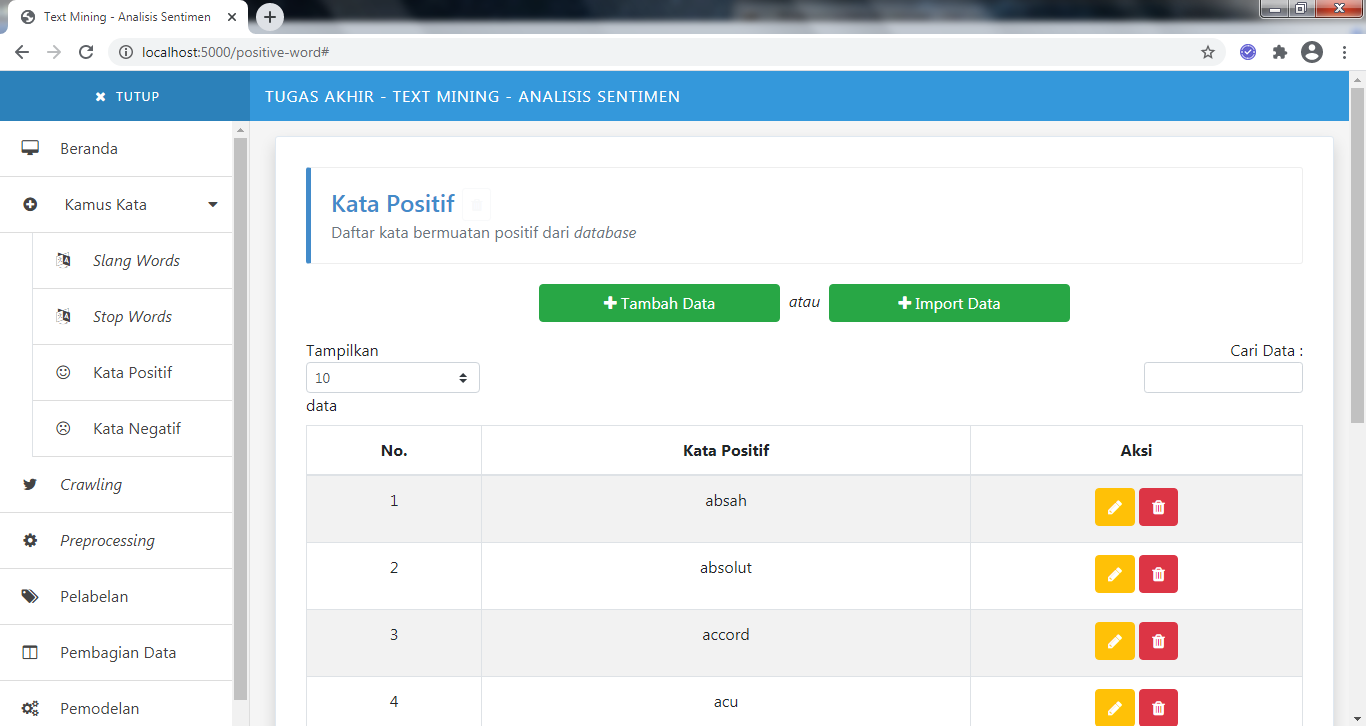
Tampilan layar kamus *stopword* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.28 berikut:



**Gambar 284.28 Tampilan layar kamus stopword**

#### Tampilan layar kamus kata positif

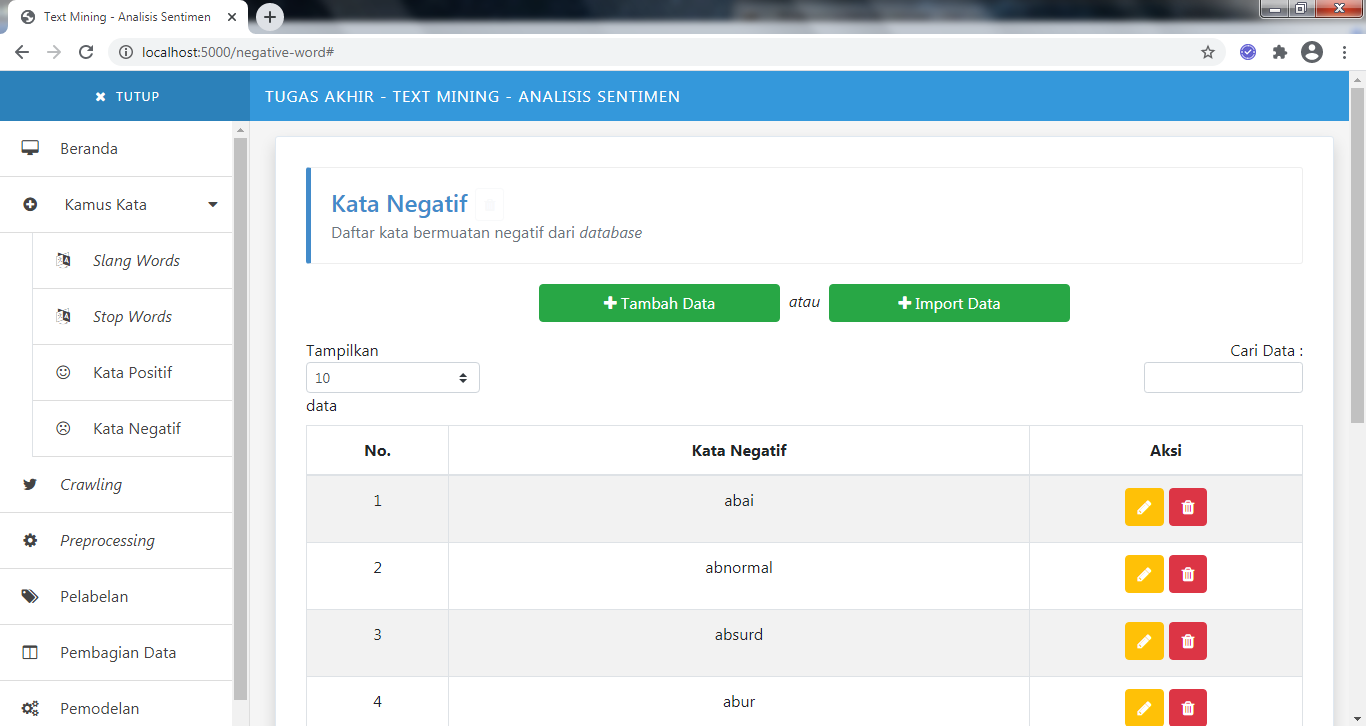
Tampilan layar kamus kata positif dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.29 berikut:



**Gambar 294.29 Tampilan layar kamus kata positif**

#### Tampilan layar kamus kata negatif

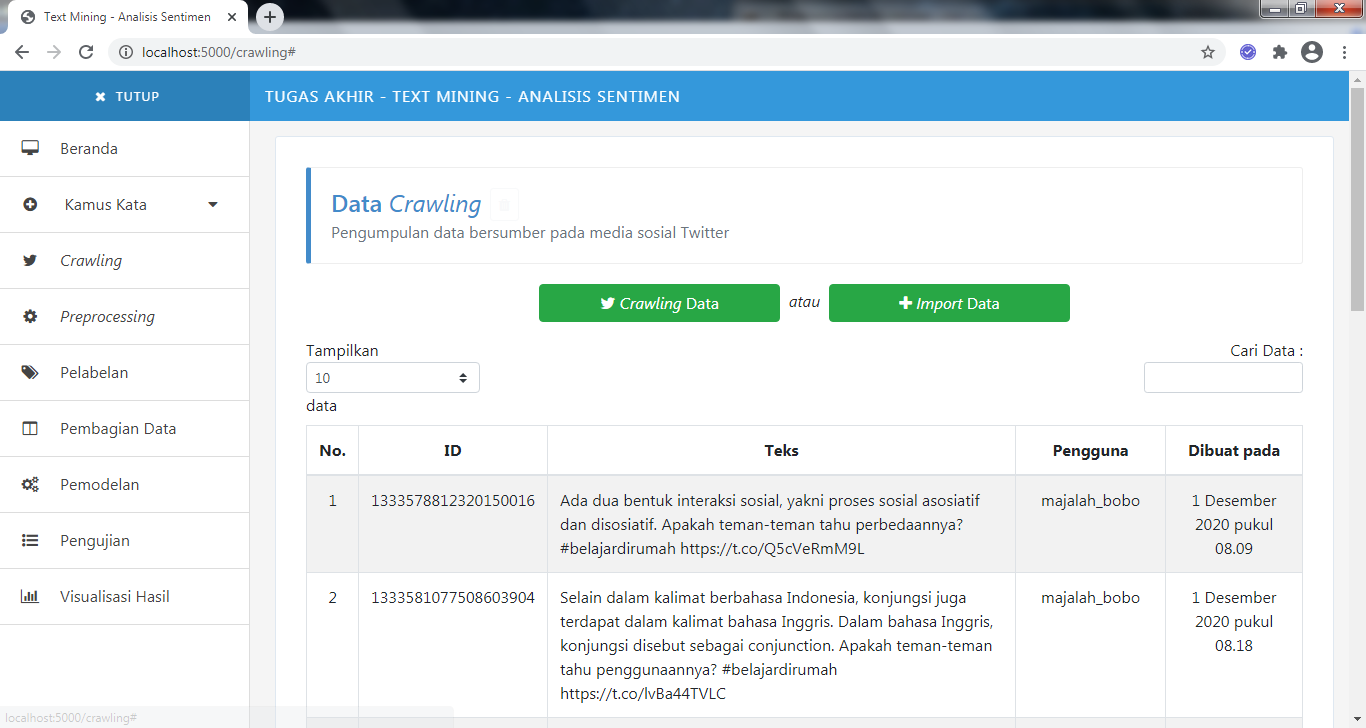
Tampilan layar kamus kata negatif dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.30 berikut:



**Gambar 304.30 Tampilan layar kamus kata negatif**

### Tampilan layar *crawling*

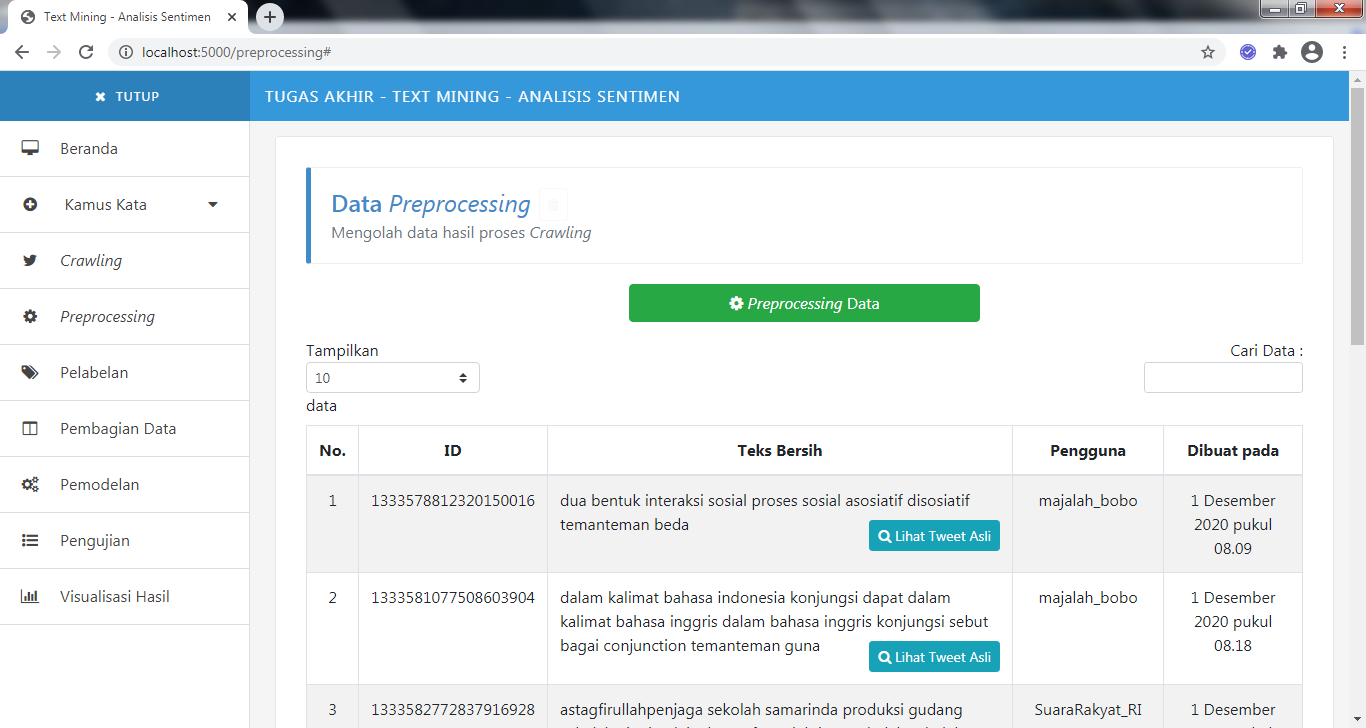
Tampilan layar *crawling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.31 berikut:



**Gambar 314.31 Tampilan layar crawling**

### Tampilan layar *preprocessing*

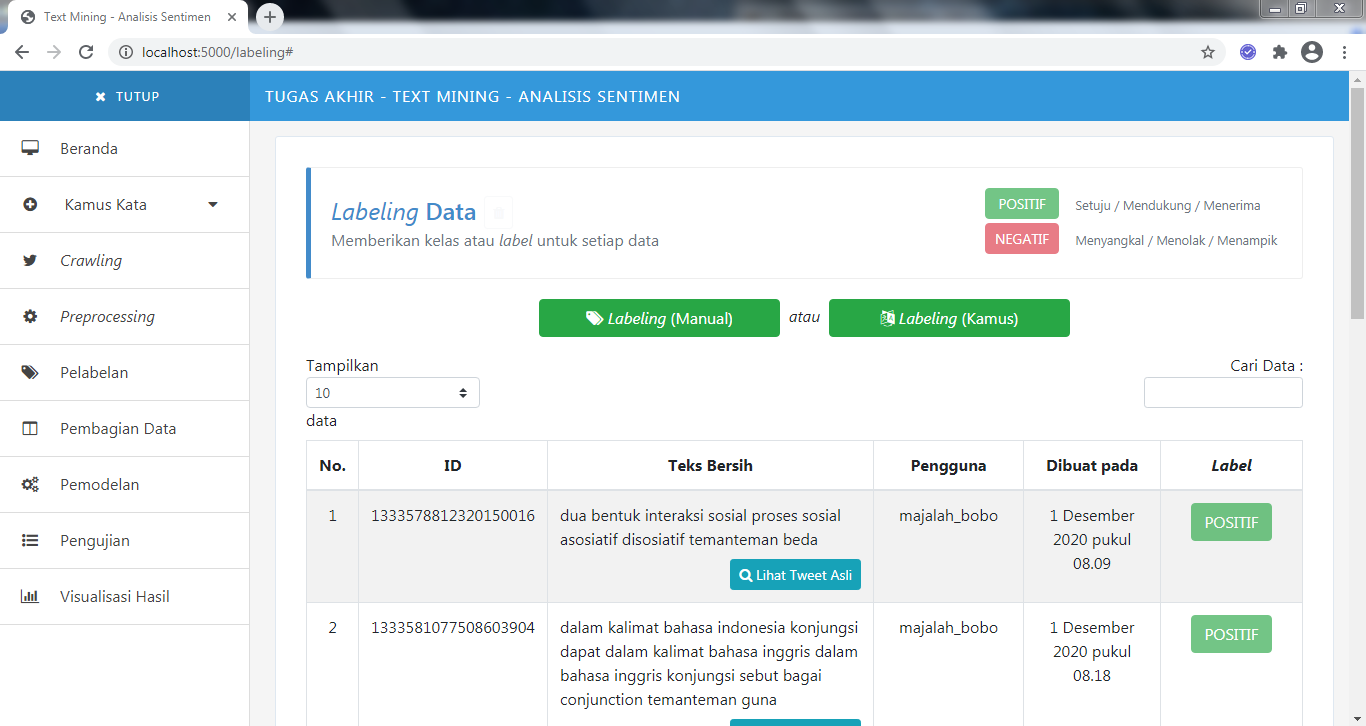
Tampilan layar *preprocessing* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.32 berikut:



**Gambar 324.32 Tampilan layar preprocessing**

### Tampilan layar *labeling*

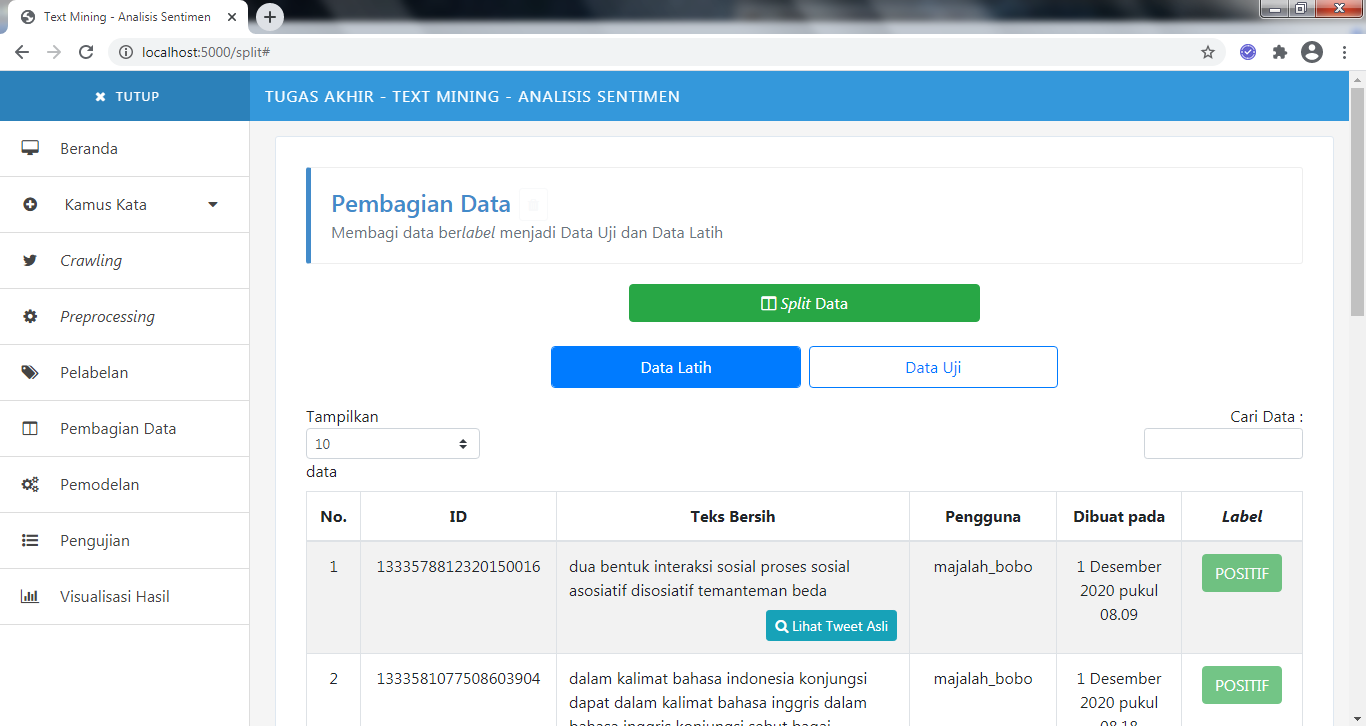
Tampilan layar *labeling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.33 berikut:



**Gambar 334.33 Tampilan layar labeling**

### Tampilan layar pembagian data

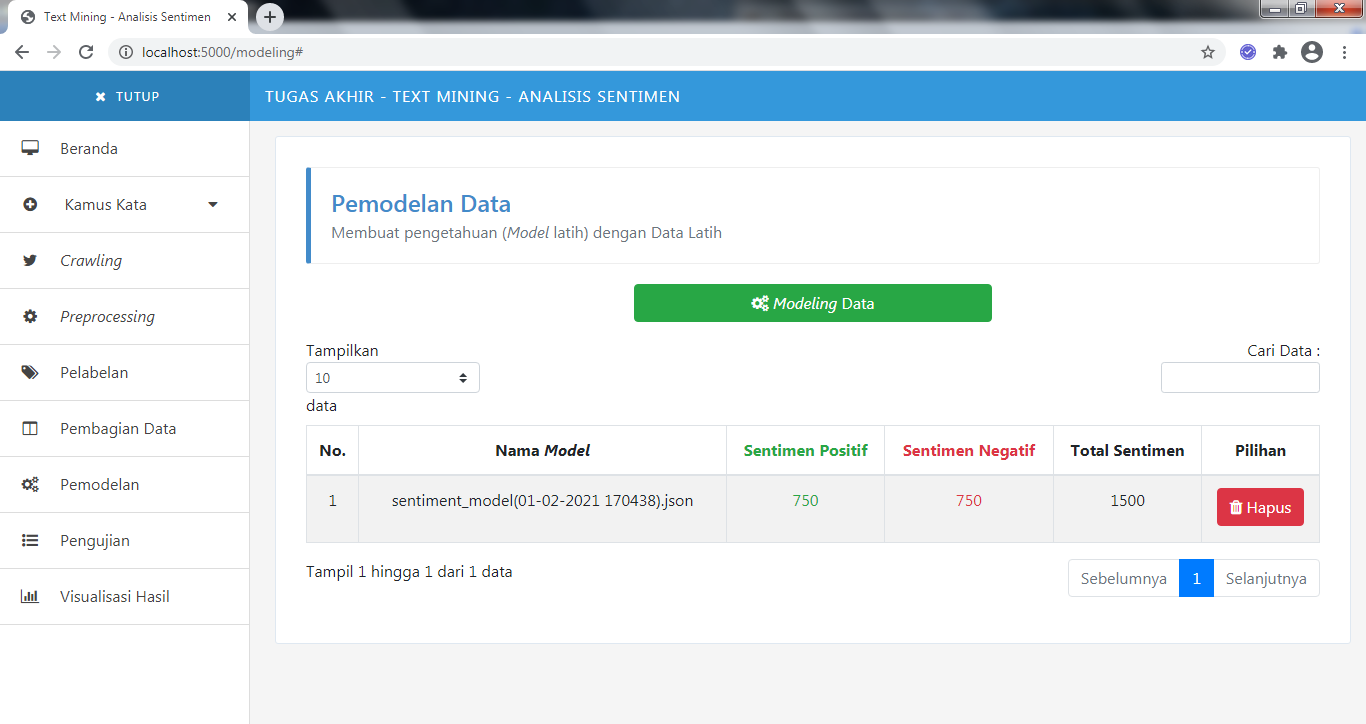
Tampilan layar pembagian data dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.34 berikut:



**Gambar 344.34 Tampilan layar pembagian data**

### Tampilan layar *modeling*

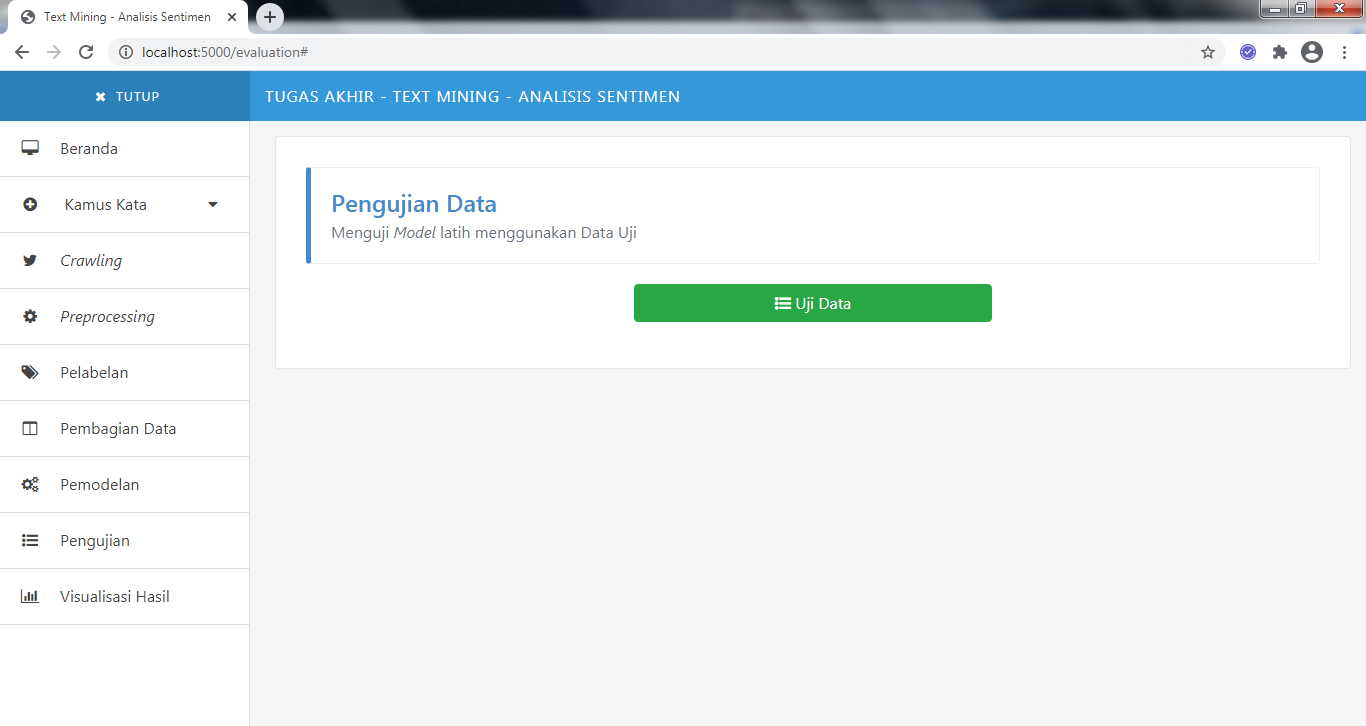
Tampilan layar *modeling* dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.35 berikut:



**Gambar 354.35 Tampilan layar modeling**

### Tampilan layar pengujian

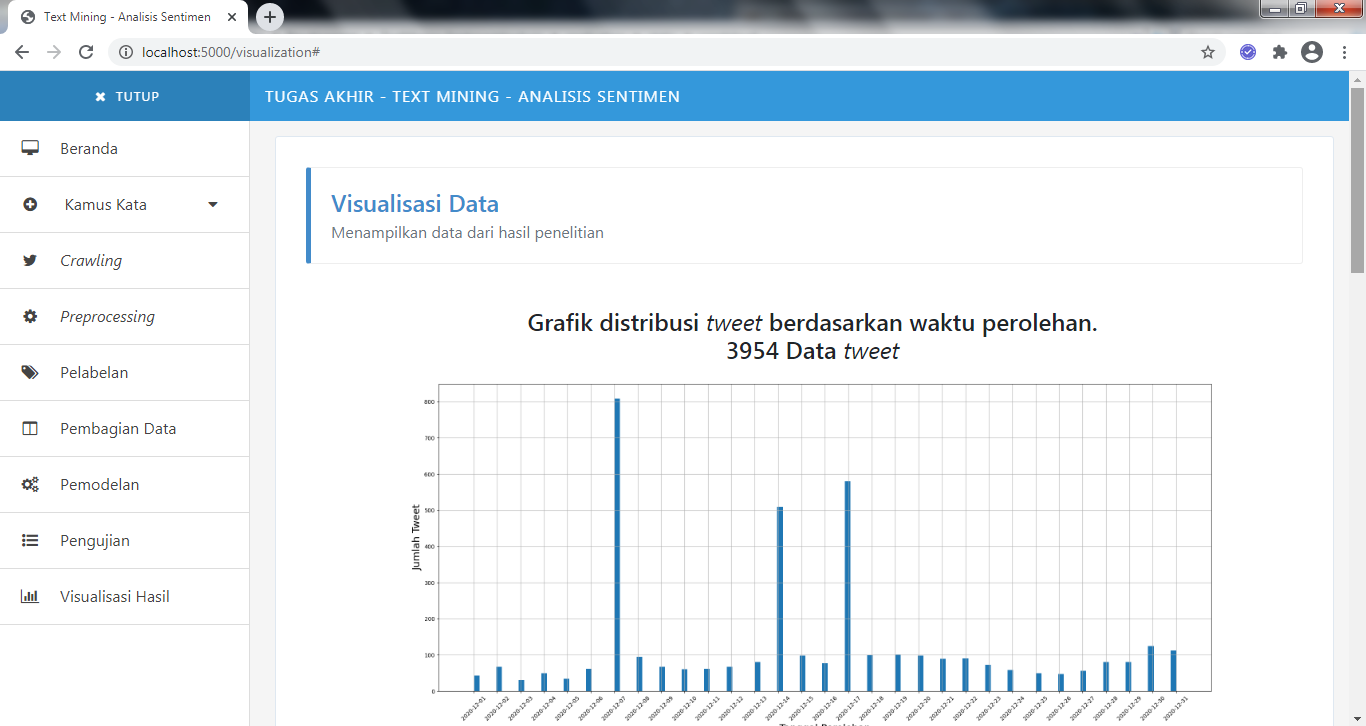
Tampilan layar pengujian dari program aplikasi yang dibuat dapat dilihat pada Gambar 4.36 berikut:



**Gambar 364.36 Tampilan layar pengujian**

### Tampilan layar visualisasi hasil

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



**Gambar 374.37 Tampilan layar visualisasi hasil**