# BAB IV

# HASIL DAN PEMBAHASAN

## Lingkungan Percobaan

Agar aplikasi yang telah dikembangkan dapat berjalan dengan semestinya, dibutuhkan perangkat dengan spesidikasi 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, Mozilla Firefox

#### 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 X 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 X terdapat tiga (3) kolom yaitu: *Tweet* (*Tlatih-i*) yang berarti urutan *Tweet* 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 X 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 X berikut:

**Tabel X *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 X merupakan hasil dari proses pemisahan kata dari kolom *clean text* pada *tweet* berdasarkan pada Tabel X 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 X berikut:

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

**Tabel X 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

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 X Fitur kata dan Tabel X Vektor kosong, maka vektor kata yang dihasilkan seperti pada Tabel X berikut:

**Tabel X 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 X 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 X berikut:

**Tabel X File JSON *model* latih**

| *Key* | *Value* |
| --- | --- |
| *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 X terdapat tiga (3) *key* yaitu: *label\_list* yang berisi nilai pada kolom *Sentiment Type* dari *Tweet* ke-i pada Tabel X Sampel data latih secara berurutan, *feature\_list* yang berisi daftar fitur yang diperoleh dari Tabel X Fitur kata, *vector\_list* yang berisi nilai representasi vektor dari Tabel X Representasi vektor secara berurutan berdasarkan *Tweet* ke-i pada Tabel X Sampel data latih.

### Tahap klasifikasi *K-Nearest Neighbors*

Tahap klasifikasi menggunakan *K-Nearest Neighbors* (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 X berikut:

**Tabel X 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 jumlah fitur (*model* latih) jumlah data uji. Bedasarkan jumlah fitur pada *model* latih dan Tabel X Sampel data uji, maka vektor kosong akan terbuat seperti pada Tabel X berikut:

**Tabel X 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

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 fitur kata dalam *model* latih dengan tiap kata dalam *tweet* data uji. Maka vektor kata akan terbuat seperti pada Tabel X berikut:

**Tabel X 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 X Representasi vektor uji) dan data vektor­ pada *model* latih. Berdasarkan pada sub bab (2. 9), proses perhitungan jarak melibatkan *euclidean distance* dengan persamaan X. 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 vektor uji terhadap vektor latih adalah seperti dalam Tabel X berikut:

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

**Tabel X 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 X 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 X berikut:

**Tabel X 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. Berdasarkan hal tersebut dapat diketahui melalui *model* latih dan Tabel X data K tetangga terdekat, bahwa nilai probabilitas yang dihasilkan adalah seperti pada Tabel X berikut:

**Tabel X Nilai probabilitas data uji**

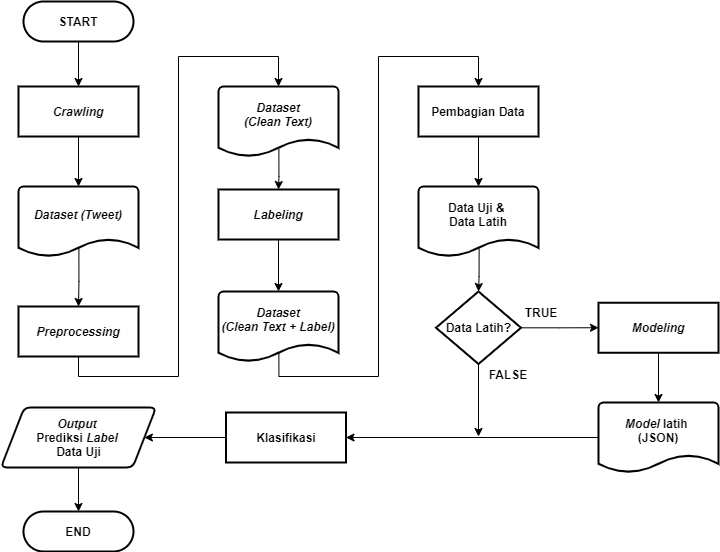
| *Tweet* (*T*uji-*i*) | *Tweet*  (*T latih-i*) | *Sentiment Type* | Probabilitas positif | Probabilitas negatif |
| --- | --- | --- | --- | --- |
| uji-1 | latih-1 | positif | 1 | 0 |
| latih-3 | positif | 1 | 0 |
| latih-5 | negatif | 0 | 1 |
| Jumlah | | | 2 (0.667) | 1 (0.333) |
|  |  |  |  |  |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

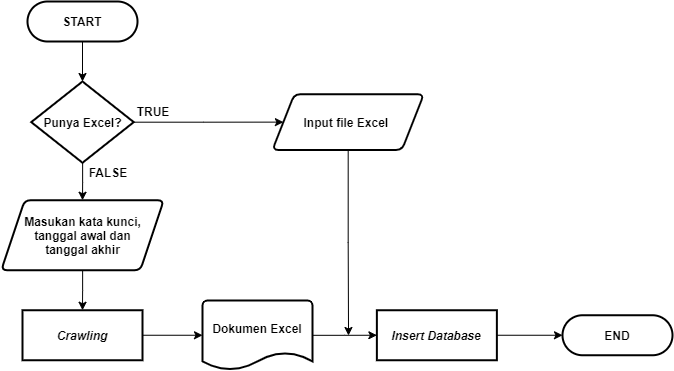


**Gambar X *Flowchart* keseluruhan sistem**

Pada Gambar X *Flowchart* keseluruhan sistem, menjelaskan proses keseluruhan 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 2:8, 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 crawling*

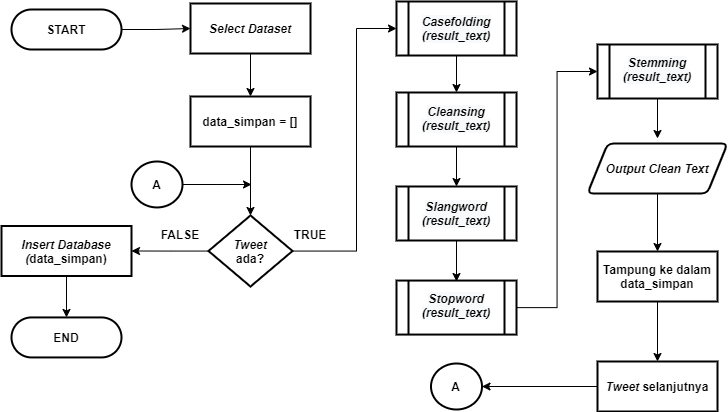
Pada *flowchart* ini, menjelaskan proses pengumpulan data atau *crawling* data tweet dimulai dari memasukkan kata kunci dan tanggal awal dan tanggal akhir, cara lainya adalah dengan fitur import file *excel*, kemudian dilakukan pencarian *tweet* bredasarkan parameter menggunakan pustaka Tweepy. Hasil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



**Gambar X *Flowchart* *crawling***

### *Flowchart preprocessing*

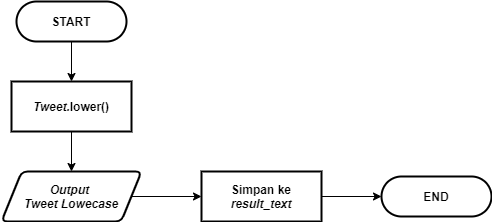
Pada *flowchart* ini, menjelaskan proses pengubahan data pengubahan data teks *tweet* menjadi terstruktur atau setara, guna mendukung proses klasifikasi agar berjalan dengan baik. Hasil proses ini berupa teks bersih, yang kemudian akan disimpan ke dalam *database*  untuk proses selanjutnya. F*lowchart*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



**Gambar X *Flowchart preprocessing***

#### Flowchart casefolding

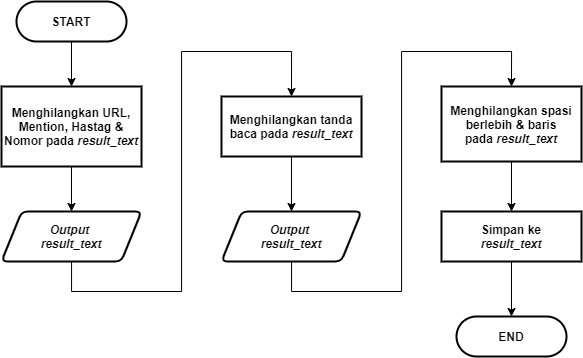
Pada *flowchart* ini dilakukan proses pengubahan teks *tweet* menjadi huruf kecil secara keseluruhan. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result\_text* untuk proses selanjutnya.



**Gambar X *Flowchart casefolding***

#### Flowchart cleansing

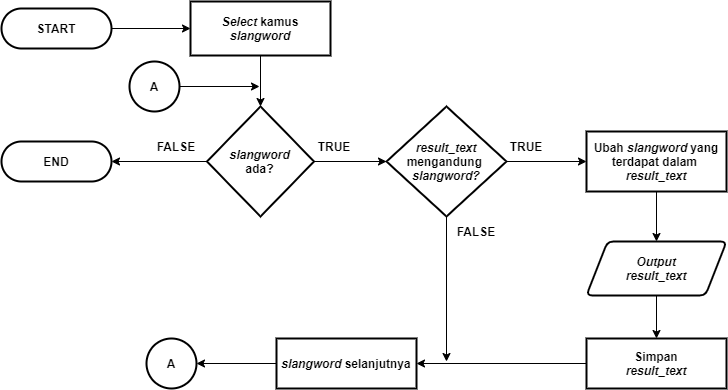
Pada *flowchart* ini dilakukan proses penyaringan dan pembuangan attribut pada teks *result*\_*text*, diantaranya penghapusan atau penghilangan URL, *mention*, *hastag*, nomor, tanda baca, dan spasi atau baris berlebih. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya.



**Gambar X *Flowchart cleansing***

#### Flowchart slangword

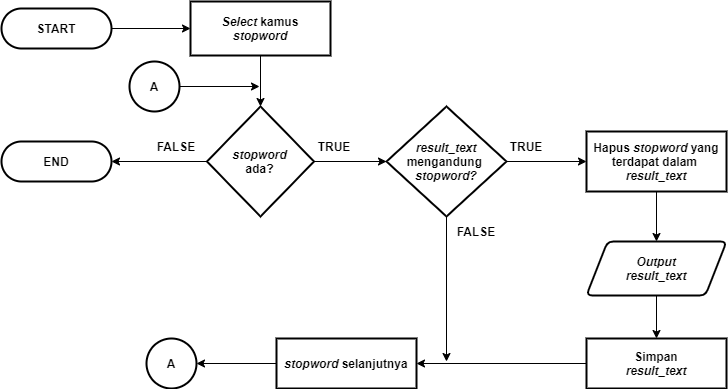
Pada *flowchart* ini dilakukan pengubahan attribut pada teks *result*\_*text* yang mengandung kata *slang* dari kamus *slangword* ke bentuk kata asli atau kata bakunya. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya.



**Gambar X *Flowchart slangword***

#### Flowchart stopword

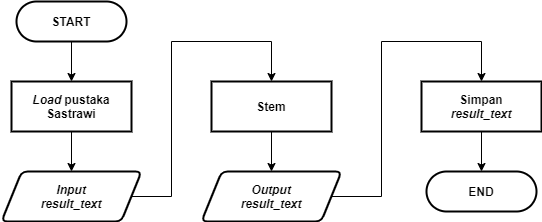
Pada *flowchart* ini dilakukan proses penghapusan attribut pada teks *result*\_*text* yang mengandung *stopword* dari kamus *stopword* karena dianggap kurang memiliki makna untuk proses klasifikasi. Hasil dari proses ini ditampung ke dalam sebuah variabel bernama *result*\_*text* untuk proses selanjutnya.



**Gambar X *Flowchart stopword***

#### Flowchart stemming

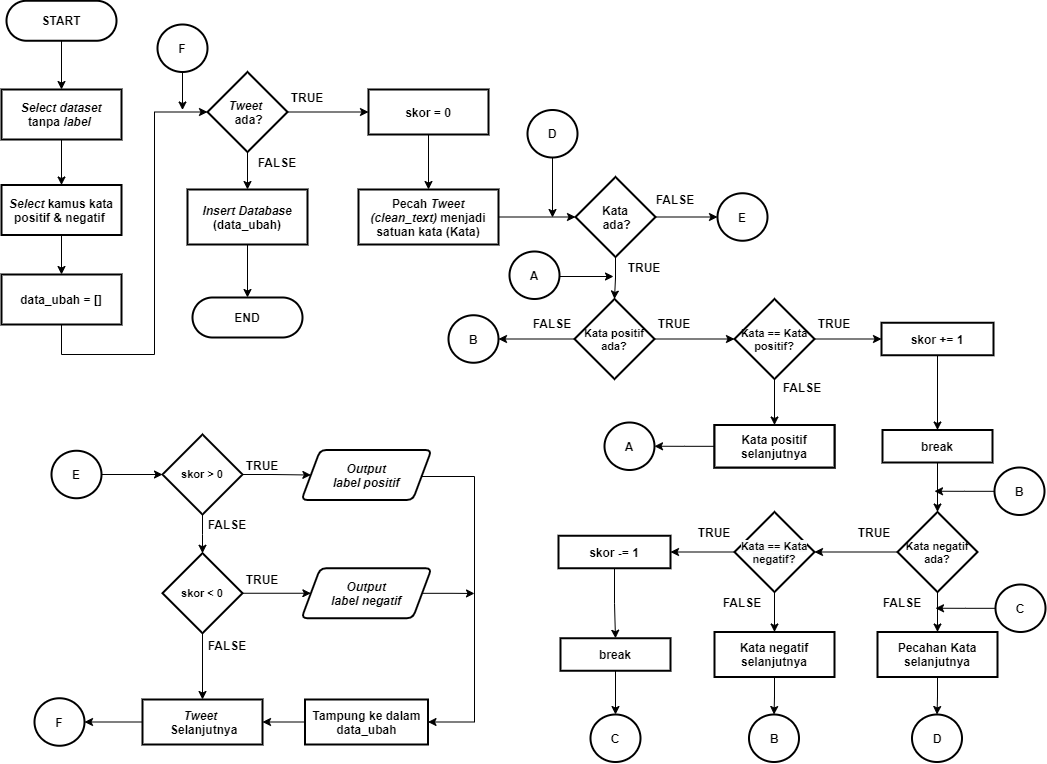
Pada *flowchart* ini dilakukan pengubahan attribut pada teks *result*\_*text* yang mengandung kataberimbuhan ke bentuk kata asalnya, menggunakan pustaka Sastrawi.



**Gambar X *Flowchart stemming***

### *Flowchart labeling*

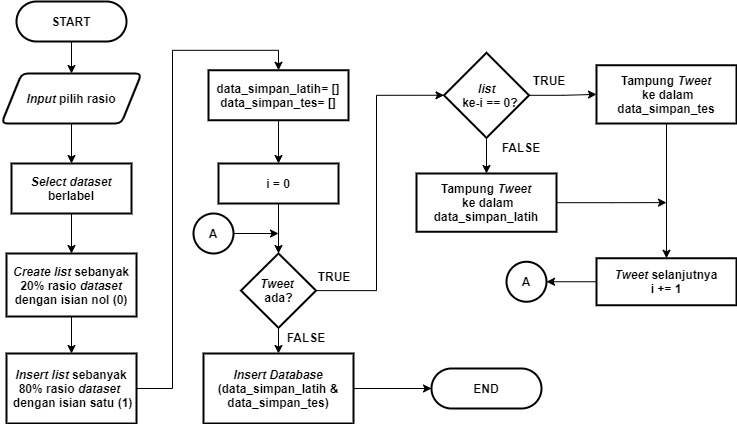
Pada *flowchart* ini, menjelaskan proses *labeling* yang dilakukan dengan cara pendekatan kamus sentimen, di mana diawali dengan perhitungan skor menggunakan kamus positif dan negatif, lalu penentuan kelas atau *label* berdasarkan nilai skor.



**Gambar X *Flowchart labeling***

### *Flowchart* pembagian data

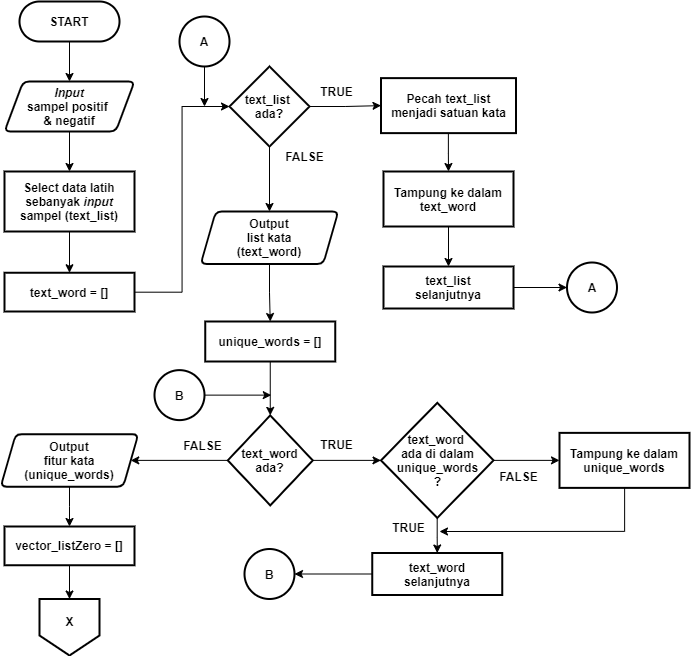
Pada *flowchart* ini, menjelaskan proses pembagian data ke dua (2) buah bagian, yaitu: data uji dan data latih, menggunakan rasio 2:8 (data uji : data latih).

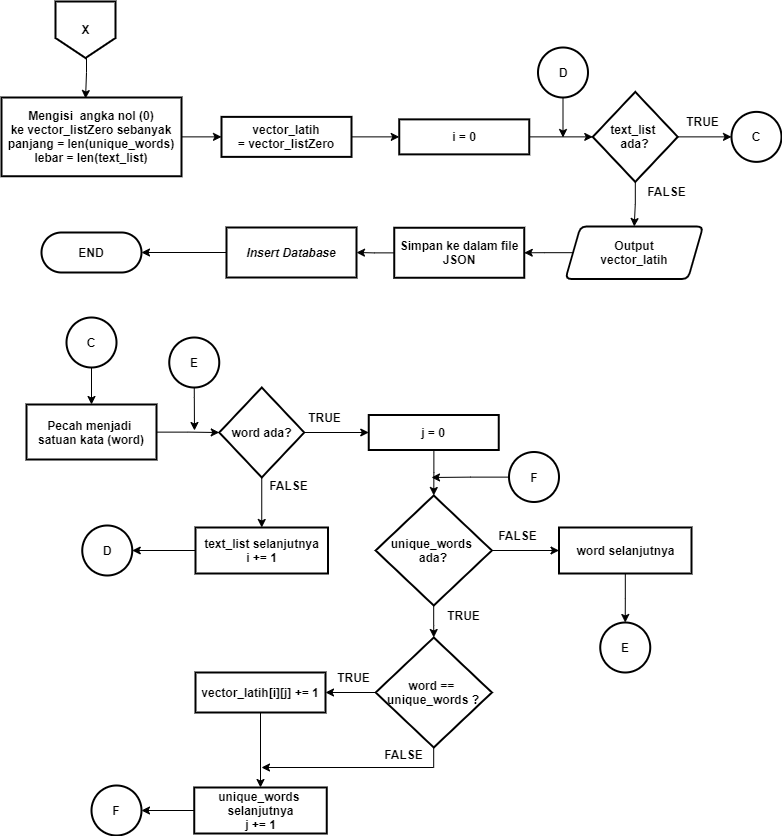


**Gambar X *Flowchart* pembagian data**

### *Flowchart* *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.

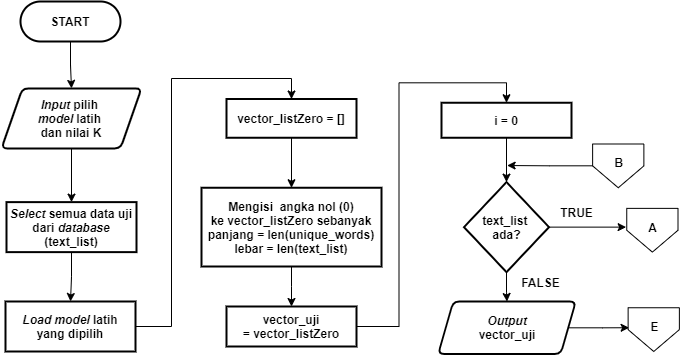


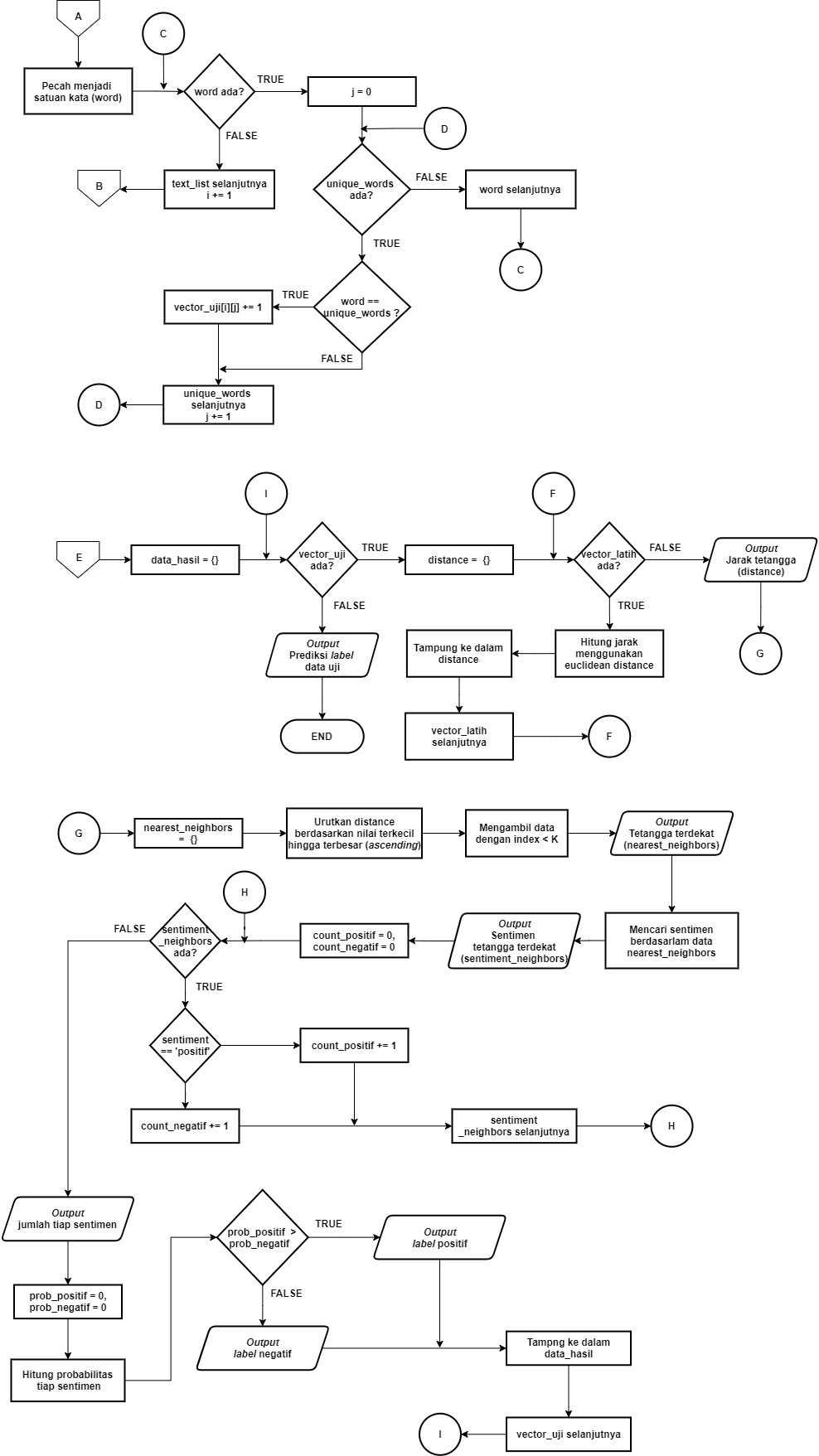


**Gambar X *Flowchart modeling***

### *Flowchart* klasifikasi

Pada *flowchart* ini, menjelaskan proses klasifikasi dengan *K-Nearest Neighbors*. Dimulai dari proses membuatan vektor uji, menghitung jarak antar data, mencari tetangga terdekat, dan menghitung nilai probabilitas.



**Gambar X *Flowchart* klasifikasi**

## 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 sistem

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 *crawling*

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 1 start  2 if (Ada File Excel)  3 Input file Excel  4 else  5 Input kata kunci, tanggal awal, dan tanggal akhir  6 Lakukan proses Crawling  7 Baca dokumen Excel  8 endif  9 Simpan ke dalam database  10 end |

### Algoritme *preprocessing*

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 casefolding

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 1 start  2 Baca Dateset (Tweet)  3 Ubah ke huruf kecil  4 Simpan result\_text  5 end |

#### Algoritme cleansing

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 1 start  2 Select result\_text  3 Hapus URL, Mention, Hastag & Nomor  4 Output result\_text  5 Hapus tanda baca  6 Output result\_text  7 Hapus spasi berlebih dan baris  8 Output result\_text  9 end |

#### Algoritme slangword

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 stopword

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 stemming

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 *labeling*

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 Simpan data\_ubah ke dalam database  41 endif  42 end |

### Algoritme pembagian data

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 1 start  2 Input pilih rasio pembagian data  3 Select dataset berlabel  4 Buat list dengan isian nol (0) sebanyak 20% dari total dataset berlabel  5 Buat list dengan isian satu (1) sebanyak 80% 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 *modeling*

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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  52 end |

### Algoritme klasifikasi

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan berdasarkan *flowchart* yang dibuat sebelumnya.

|  |
| --- |
| 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 |

## Pengujian

## 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 kamus kata

### Tampilan layar *crawling*

### Tampilan layar *preprocessing*

### Tampilan layar *labeling*

### Tampilan layar pembagian data

### Tampilan layar pengujian

### Tampilan layar visualisasi hasil