BAB 3
METODE PENELITIAN

Proses analisis ulasan pengguna dan membangun *dashboard* merupakan dasar dati penelitian ini, alat, bahan, dan teknik pengembangan sistem untuk melakukan *clustering* ulasan pengguna di *official store* Aerostreet.

3.1 BAHAN DAN ALAT PENELITIAN

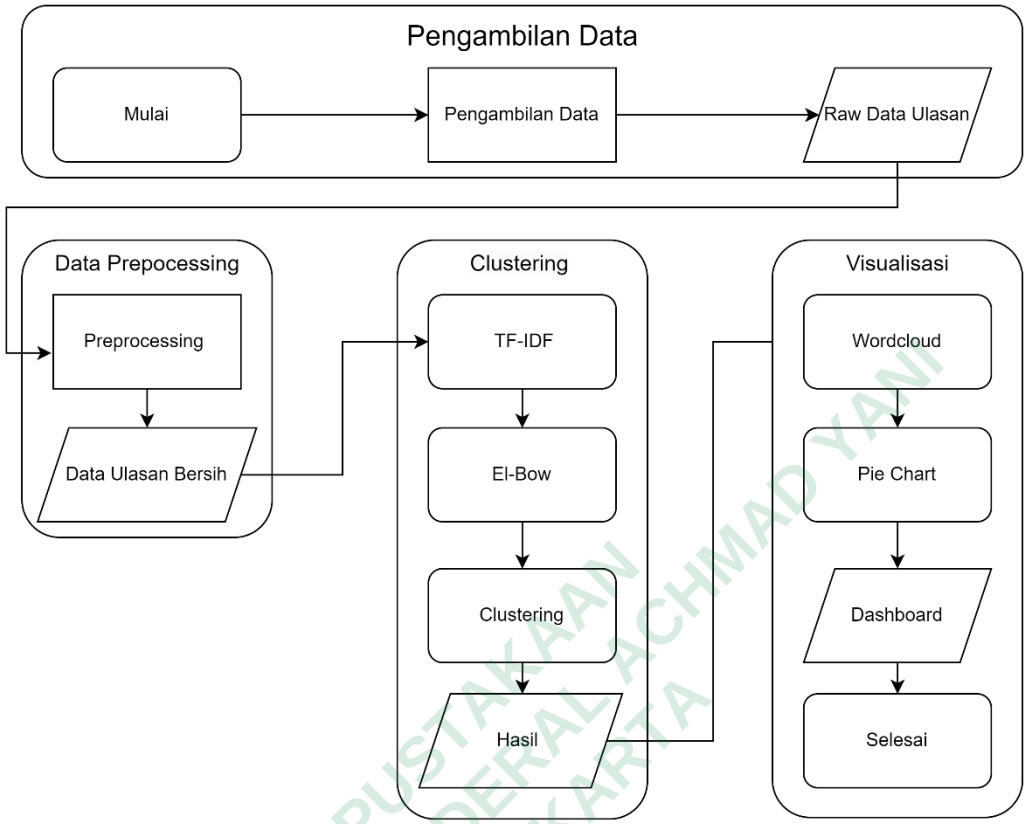
Bahan atau sumber data yang digunakan dalam penelitian ini adalah ulasan produk dari pelanggan yang memberikan komentar. Data ulasan pelanggan ini diperoleh dari *official store* Aerostreet di platform Tokopedia.

Dalam pelaksanaan penelitian ini, perangkat yang digunakan adalah komputer dengan spesifikasi yang memenuhi standar untuk menjalankan sistem operasi. Windows 11 dengan lancar, serta koneksi internet berkecepatan cukup. Selain itu ada juga perangkat lunak dan perangkat keras pendukung dalam melakukan penelitian ini adalah :

1. Sistem Operasi: Windows 11.
2. Ram 16Gb.
3. Automa 1.28.1
4. Tokopedia.
5. Visual Studio Code 1.75.1.
6. Python 3.11.

3.2 JALAN PENELITIAN

Metode yang digunakan untuk menganalisis data adalah *clustering* K-Means karena implementasainya mudah dan tingkat akurasi tinggi. Algoritma ini bertujuan untuk mengurangi variasi data dalam satu kelompok atau *cluster* sambil memaksimalkan data dalam kelompok-kelompok lainnya. Proses analisis ulasan pengguna pada Tokopedia dilakukan melalui tahapan yang melibatkan K-Means *clustering* dapat dilihat pada Gambar 3.1:



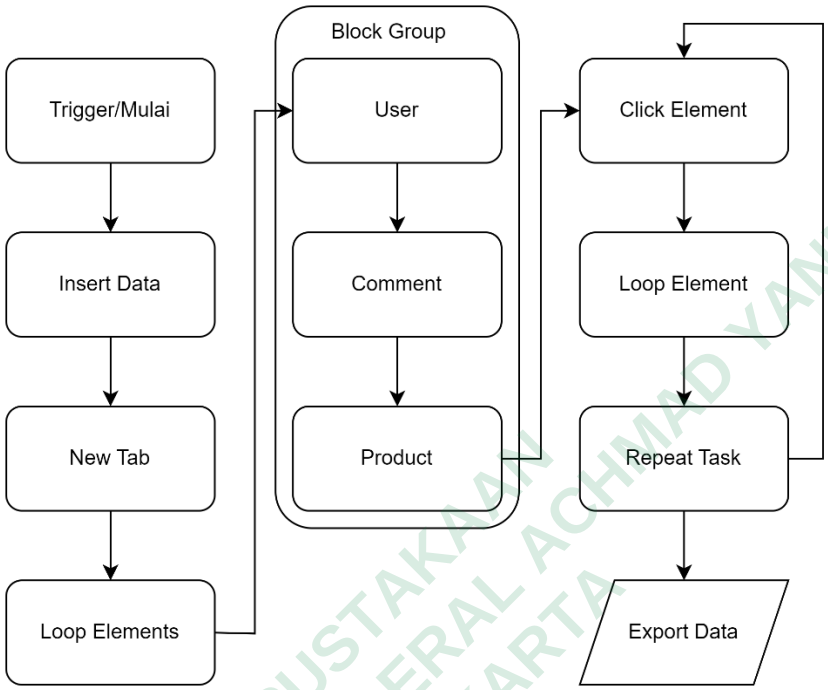
Gambar 3.1 Tahap Penelitian

Pada Gambar 3.1 Dalam rangka melakukan penelitian ini terdapat beberapa tahapan yang perlu dilalui, dalam proses analisis ulasan pelanggan di *platform* Tokopedia menggunakan metode *clustering* K-Means, di bawah ini penjelasan mengenai langkah-langkah yang ditempuh:

3.2.1 Pengambilan Data

Pengambilan data merupakan langkah mengintegrasikan data ulasan yang diperoleh dari berbagai sumber di *e-commerce* menjadi satu kesatuan utuh. Dalam penelitian ini, data diambil dari *platform* Tokopedia. Keputusan untuk memilih *official store* Aerostreet di Tokopedia sebagai objek penelitian didasarkan pada banyaknya ulasan pelanggan yang yang beragam. Pemilihan Automa sebagai alat pengambilan data dipandang tepat karena prosesnya yang sederhana tanpa perlu menulis skrip kode. Automa, sebagai ekstensi Google Chrome, memberikan solusi untuk tugas-tugas berulang, memungkinkan pengambilan data dari berbagai situs

web, langkah langkah dalam melakukan pengambilan data menggunakan Automa dapat dilihat pada Gambar 3.2:



Gambar 3.2 Tahapan Automa

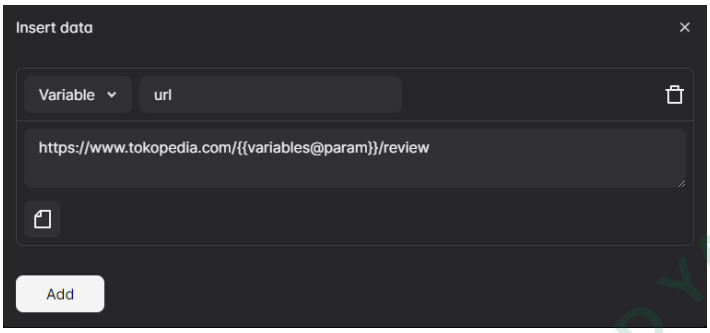
pada Gambar 3.4 menunjukan bagaimana urutan dalam melakukan pengambilan data ulasan pelanggan yang terdapat di Tokopedia dengan urutan:

3.2.1.1 Membuat Workflow.

Dalam membuat *workflow* dengan menggunakan ekstensi Automa hal yang dibutuhkan pertamakali yaitu adalah *trigger block* namun sebelum menyinggung tentang apa itu *trigger block* perlu diketahui tentang *workflow*, *workflow* merupakan serangkaian langkah menjalankan Automa secara otomatis, dalam Bahasa Indonesia *workflow* dapat diartikan sebagai "alur kerja" atau "aliran kerja". Istilah tersebut merujuk pada rangkaian langkah atau proses yang harus diikuti untuk menyelesaikan tugas atau proyek tertentu. Dalam penelitian ini, *workflow* Automa digunakan untuk mengambil data ulasan pengguna Aerostreet di Tokopedia. Selanjutnya adalah pengertian dari *trigger block*, *trigger block* adalah titik awal di mana alur kerja akan mulai dieksekusi.

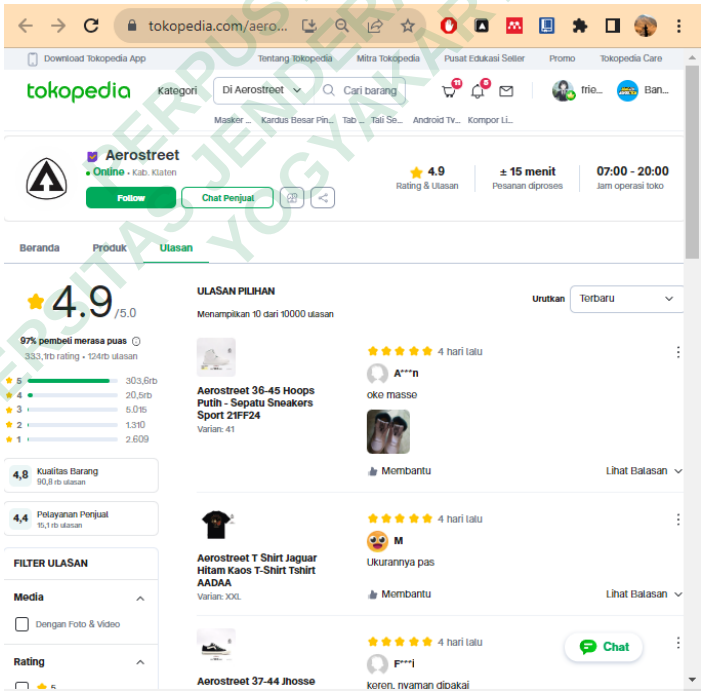
3.2.1.2 Insert Data

Setelah membuat *trigger block* adalah menentukan *link* dari situs web yang akan diambil datanya, halaman *insert* ini tampilannya apat dilihat pada Gambar 3.3:



Gambar 3.3 Insert Data Automa

Url yang sebelumnya sudah ditentukan ditunjukkan pada Gambar 3.3, dan tampilan halaman web yang mengumpulkan data ulasan pelanggan ditunjukkan pada Gambar 3.4:

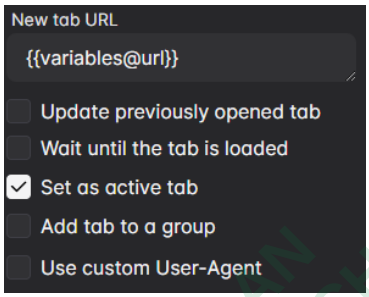


Gambar 3.4 Halaman Data Ulasan

Gambar 3.4 menunjukkan halaman yang akan digunakan dalam proses pengoperasian yang telah ditetapkan dengan menyertakan *url* halaman tersebut.

3.2.1.3 Fungsi New Tab

Kemudian proses selanjutnya adalah membuat halaman baru atau *new tab* fungsi ini membuat halaman baru berdasarkan *url* tab baru. *Url* yang valid harus menyertakan skema http:// atau https://. Dalam melakukan pengambilan data ulasan pengguna halaman baru ditentukan sebagai tab aktif sebagai dapat dilihat pada Gambar 3.5:

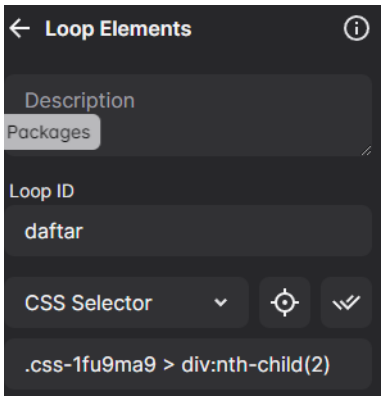


Gambar 3.5 New Tab Automa

Gambar yang tertera di atas menggambarkan bahwa setelah menjalankan proses alur kerja, akan muncul suatu halaman baru seperti yang tampak pada Gambar 3.4 sebelumnya. Halaman baru ini akan berisi *url* ulasan dari pelanggan.

3.2.1.4 Loop Elements

Loop Elementes memiliki lebih banyak fitur berikut ini fitur yang digunakan dalam pemilihan data yang nanti akan dilakukan proses perulangan dengan pilihan elemen css *selector* yang terlihat pada ilustrasi dalam Gambar 3.6:

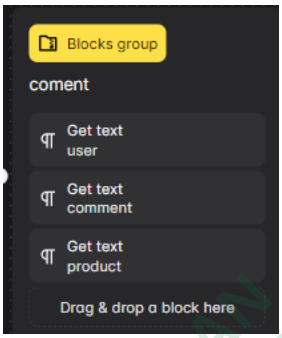


Gambar 3.6 Loop Elements

Pada Gambar 3.6 menunjukan tentang css *selector* yang sudah ditentukan dengan nama daftar.

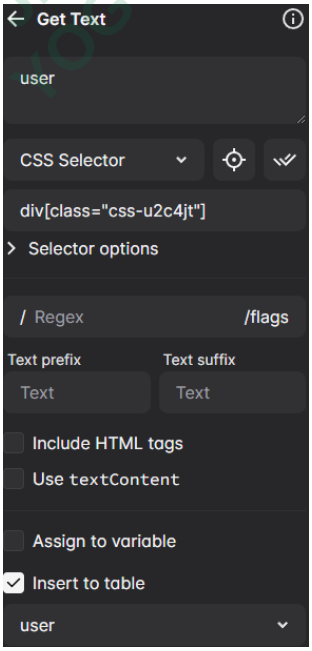
3.2.1.5 Block Group

Block Group memiliki fungsi untuk melompokkan blok lain bersama dalam satu wadah, fungsi ini membuat alur kerja terlihat lebih teratur, di dalam *block group* dijalankan secara berurutan berdasarkan urutan blok dari atas ke bawah dapat diamati pada Gambar 3.7:



Gambar 3.7 Blocks Group

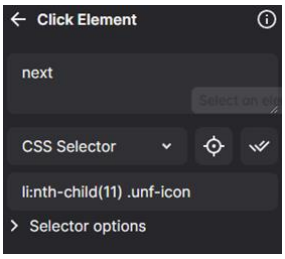
Gambar 3.7 memperlihatkan fungsi *block group*, didalamnya dapat ditentukan jenis data yang nantinya akan dikelompokan dalam tabel dengan format csv, dalam fitur *blocks group* juga menggunakan fitur *css selector* seperti pada Gambar 3.8:



Gambar 3.8 Get Text

3.2.1.6 Click Element

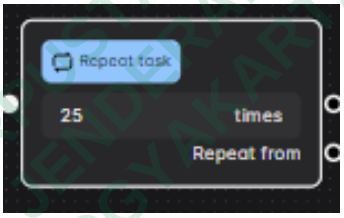
Click Element adalah fungsi klik mouse pada suatu elemen Tampilan yang dapat diakses dengan mengklik dapat ditemukan dalam Gambar. 3.9:



Gambar 3.9 Click Element

3.2.1.7 Repeat Task

Repeat Task adalah fungsi perulangan untuk menjalankan satu blok atau beberapa blok kemudian dalam fitur ini juga dapat ditentukan berapa kali pengulangan dalam input yang disediakan terdapat pada Gambar 3.10:



Gambar 3.10 Repeat Task

Pada Gambar 3.10 menunjukan perulangan sebanyak 25 kali, dalam penelitian ini perlu melakukan perulangan sebanyak 25 kali agar mendapatkan jumlah data yang Sesuai dengan standar yang telah ditetapkan sebelumnya oleh dosen pembimbing, dalam kerangka penelitian ini, jumlah minimal data ulasan yang diharapkan adalah sebanyak 4000.

3.2.1.8 Export Data

Dengan *scraping* pada halaman ulasan di Tokopedia menggunakan Automa, data ulasan pengguna di official store Aerostreet dapat diperoleh dengan waktu yang lebih cepat dan efektif kemudian, tabel 3.1 menunjukkan hasil dari Automa.:

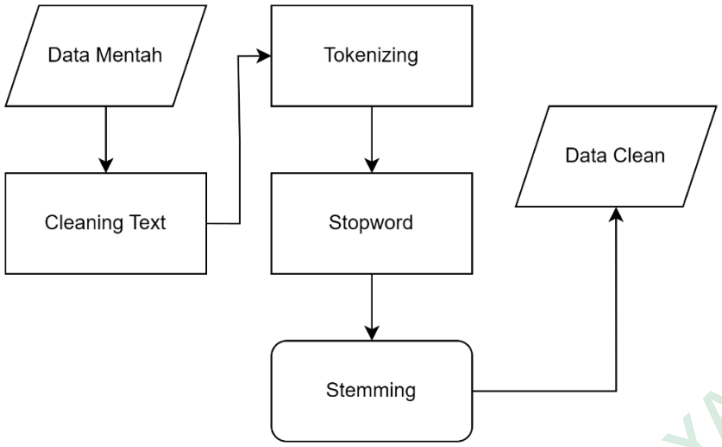
Tabel 3.1 Data Mentah

user,comment,product
Andika,"mantap nyampe nya cepet banget 2 hari an, barang nya bagus banget","Aerostreet Chinos Panjang Elvano Hitam Celana Chino EAAAA Varian: 34"
Abu,ini baru local pride,"Aerostreet 36-45 Massive Low Hitam - Sepatu Sneakers Casual 21AA30 Varian: 42"
sugiono,"kualitas g perlu diragukan,harga merakyat respon cepat","Aerostreet Jeans Slim Fit Jackson Dark Blue 1F100 Varian: 32"
Gabriella,"Uda langganan disini, baju2nya sesuai deskripsi, bahan tebal, halus n lembut, bagus deh untuk harga murah begini worth it","Aerostreet Kemeja Relaxed Fit Black Hawai Polos Gelap QAAAA Varian: L"

Data ulasan pada tabel 3.1 disimpan dalam bentuk file CSV secara otomatis oleh Automa sekitar 15.300 data ulasan namun masih banyak tabel kosong dan kata yang tidak perlu. Maka tahapan selanjutnya yaitu melakukan *preprocessing* agar data ulasan tersebut lebih terstruktur.

3.2.2 Preprocessing

Tahap permulaan dalam melakukan analisis data adalah proses *preprocessing*, dimana data dibersihkan dan maksimalkan untuk menjadi data yang berkualitas saat dianalisis. Pada tahap ini, teks tak terstruktur juga diorganisir menjadi teks terstruktur seperti yang terlihat dalam Gambar 3.11:



Gambar 3.11 Tahap Preprocessing

Dari Gambar 3.11 dapat dilihat bahwa *data preprocessing* memiliki beberapa tahapan diantaranya yaitu:

3.2.2.1 Cleaning Text

Perlu dilakukan pembersihan untuk data mentah yang diambil dari Tokopedia menggunakan Automa, seperti menghapus tanda @ pada nama pengguna (username), angka dalam ulasan, dan tanda baca seperti tanda tanya, tanda seru, dan titik. Sehingga perlu dilakukan *cleaning text* agar data yang dihasilkan menjadi lebih bersih dari kata yang tidak berguna dalam sebuah ulasan. Untuk melakukannya, data harus dibersihkan dengan menggunakan kode di bawah ini dalam pengolahan teks dengan fungsi **`cleaning_text`**:

```
def cleaning_text(data):
    data = re.sub(r"b'\@[\w]*", ' ', str(data))
    data = re.sub(r"b'[\w]*", ' ', data)
    data = re.sub(r'https\:.*$', " ", data)
    data = re.sub(r'[@][A-Za-z0-9]+', ' ', data)
    data = re.sub(r'[~^0-9]', ' ', data)
    data = re.sub(r'\@\$\w\s*', ' ', data)
    data = re.sub(r'[^\w\s]+', ' ', data)
    data = str(data).lower()
    return data
```

Untuk membersihkan dan menghapus karakter yang tidak relevan atau mengganggu dari teks, fungsi **`cleaning_text`** melakukan langkah-langkah berikut:

1. Menghilangkan kata dalam ulasan yang dimulai dengan "@", seperti username Tokopedia.
2. Menghilangkan karakter "b'" pada awal kata.
3. Menghilangkan tautan *url* dengan menggunakan pola *regex*.
4. Menghilangkan karakter "@" diikuti oleh huruf atau angka, yang mungkin merupakan tanda *tagging* atau *mention*.
5. Menghilangkan karakter "~" dan angka dari teks.
6. Menghilangkan karakter khusus seperti "@$" diikuti oleh huruf dan spasi.
7. Menghilangkan karakter yang bukan huruf atau spasi.
8. Mengubah semua karakter menjadi huruf kecil untuk konsistensi.

Fungsi **`cleaning_text`**, dapat digunakan membersihkan dan mengolah teks untuk mendapatkan teks yang bersih dan siap untuk analisis lebih lanjut. Kemudian selanjutnya melakukan ekstraksi kolom "comment" dari file data yang akan diolah dapat dilihat pada kode berikut:

```
pre = file["comment"]
```

Kemudian, dilakukan pembersihan dan pengolahan teks pada setiap komentar dalam kolom "comment" menggunakan metode *preprocessing* yang telah ditentukan sebelumnya, kemudian tahapan selanjutnya yaitu melakukan *clean data* dapat dilihat melalui kode berikut :

```
clean_data = []
for i in pre:
    clean_data.append(preprocess(i))
```

Data teks yang telah dibersihkan kemudian disusun ulang dalam bentuk **`DataFrame`** dengan kolom "comment" pada variabel **`clean_text`**:

```
clean_text = pd.DataFrame()
clean_text['comment'] = pd.Series(clean_data)
```

Untuk memastikan bahwa data teks telah dibersihkan dengan benar, kata-kata dalam setiap komentar pada sebuah ulasan digabungkan kembali menjadi satu kalimat menggunakan fungsi 'join':

```
clean_text['comment'] = [' '.join(words) for words in
clean_text['comment']]
```

Dengan rangkaian langkah-langkah di atas, data teks telah melewati tahap *preprocessing* yang menghasilkan data teks yang lebih bersih, terstruktur, dan siap

untuk proses analisis selanjutnya. Untuk melihat hasil *code* dari *cleaning text* dapat terlihat pada Tabel 3.2:

Tabel 3.2 Hasil Cleaning Text

desain sporty packaging aman warna cantik
mantap tp berat worthed sih
respon cepat banget produk bagus
model bagus ukuranny mantab
mantap enak pakai
thanks gan qualitas bagus masuk temen yg qc pabrik sepatu mantaps
cakeup sepatu empuk warna keren deh makasih
ekspektasi mantap
mantap bb kg tinggi cm pake xxl pas terima kasih aero stress
barang bagus kirim cepat

Dapat dilihat jika *text* ulasan pengguna yang sebelumnya nya dari Automa dengan banyak nya spasi kosong kini mulai terlihat rapi dan barisnya rapat, kemudian dari perbandingan data di atas teks dari data *clean* terlihat lebih bersih daripada teks yang ada di data mentah yang terdapat pada Tabel 3.1.

3.2.2.2 Tokenizing

Tokenizing adalah tahap dimana data dipecah menjadi *token-token* yang lebih kecil agar lebih mudah dalam mengakses dan memanipulasi bagian-bagian individu dari data. Untuk melakukan *tokenizing* maka diperlukan *library* python nltk yang dapat dilihat pada *code* di bawah ini:

```
from nltk import word_tokenize
import nltk
nltk.download('punkt')
```

Kode di atas adalah untuk mengimpor fungsi word_tokenize dan memastikan bahwa data **`Punkt Tokenizer Models`** dari **`NLTK`** telah diunduh. Fungsi word_tokenize kemudian dapat digunakan untuk memecah teks menjadi kata-kata atau token secara lebih akurat. Tahap selanjutnya adalah *tokenizing* berikut adalah *cod*e menggunakan fungsi *tokenzing*:

```
def tokenizing(text):
    return word_tokenize(str(text))
```

Fungsi *tokenizing* digunakan untuk mengkonversi teks menjadi kumpulan kata-kata yang disebut sebagai "token". Proses tokenisasi merujuk pada pembagian teks menjadi bagian-bagian yang lebih kecil yang disebut "token". unit-unit terkecil yang memiliki makna, yaitu kata-kata langkah-langkah dalam fungsi ini adalah:

1. Fungsi menerima input *text*, yang dapat berupa teks tunggal atau beberapa teks.
2. Fungsi mengonversi input *text* menjadi tipe data string menggunakan *str*() untuk memastikan input dalam format teks.
3. Fungsi *word_tokenize()* dari library NLTK digunakan untuk memecah teks menjadi token-token yang berupa kata-kata.
4. Fungsi mengembalikan daftar token kata-kata.

Dengan menggunakan fungsi *tokenizing*, Teks dapat mengubah menjadi token-token kata-kata yang kemudian dapat digunakan dalam analisis teks lebih lanjut, seperti penghitungan frekuensi kata atau vektorisasi teks.

3.2.2.3 Stopword Removal

Library Sastrawi digunakan dalam analisis untuk memudahkan melakukan Stopword Removal dalam bahasa Indonesia, selain melalui *Library* Sastrawi terdapat beberapa kata tambahan yang dibuat secara terpisah agar data lebih bagus saat digunakan. Untuk melakukan Stopword Removal maka bisa menggunakan *code* berikut:

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
```

Stopword Removal adalah Langkah untuk menghapus kata yang tidak memiliki arti ataupun tidak memiliki makna penting. Stopword Removal penting dilakukan karena dapat membantu meningkatkan akurasi dalam melakukan analisis teks kode berikut :

```
def stopword(text):
    factory = StopWordRemoverFactory()
    stopword = factory.create_stop_word_remover()
    text = stopword.remove(' '.join(text))
```

```
    return text.split()
```

Penjelasan mengenai fungsi ini adalah:

1. Fungsi menerima input *text*, yang seharusnya berupa token-token kata setelah tahap tokenisasi sebelumnya.
2. Fungsi menciptakan sebuah objek stopword remover menggunakan **`StopWordRemoverFactory()`** dari library Sastrawi.
3. Objek stopword remover digunakan untuk menghilangkan kata-kata stop dari token-token kata yang disatukan dengan menggunakan **`' '.join(text).`** Hasilnya adalah sebuah *string* yang telah kehilangan kata-kata stop.
4. String hasil penghapusan kata-kata stop tersebut kemudian dipecah kembali menjadi token-token kata menggunakan **`split()`** sehingga diperoleh daftar kata-kata tanpa kata-kata stop.

Dengan menggunakan fungsi *stopword*, Teks dibersihkan dari kata-kata yang tidak penting dalam analisis teks.

3.2.2.4 Stemming

Untuk melakukan *stemming* dalam kata Indonesia diperlukan *library* Sastrawi dengan menggunakan *code*:

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory=StemmerFactory()
```

Stemming adalah proses untuk menghapus afiks dan mengembalikan kata ke bentuk asalnya. Hal ini bertujuan untuk mengurangi keragaman kata. yang serupa menjadi kata inti yang identik. Berikut ini adalah uraian mengenai kode dari. fungsi *stemming*:

```
def stemming(text):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    stemmed_text = [stemmer.stem(word) for word in text]
    return stemmed_text
```

Langkah-langkah dalam fungsi dari *code* di atas adalah:

1. Fungsi menerima input *text*, yang seharusnya berupa daftar kata-kata setelah tahap penghilangan kata-kata stop sebelumnya.

2. Fungsi menciptakan sebuah objek stemmer menggunakan **`StemmerFactory()`** dari library Sastrawi.
3. Objek stemmer digunakan untuk mengubah setiap kata dalam daftar kata menjadi bentuk dasarnya (melalui proses stemming).
4. Kata-kata yang telah di-stem akan disimpan dalam daftar stemmed_text.
5. Daftar stemmed_text yang berisi kata-kata yang sudah di-stem kemudian dikembalikan sebagai hasil dari fungsi.

Dengan menggunakan fungsi stemming, kata-kata yang berupa ulasan diubah kedalam bentuk dasarnya, yang memudahkan dalam analisis teks berdasarkan akar kata.

3.2.3 Clustering K-Means

Setelah melalui tahapan *preprocessing*, langkah selanjutnya adalah melakukan *clustering* data. *Clustering* merupakan metode untuk mengelompokkan data ke dalam kategori-kategori yang dikenal sebagai *cluster*. Proses ini memisahkan populasi atau titik-titik data menjadi beberapa kelompok sehingga Data-data dalam kelompok yang sama memiliki tingkat kesamaan yang lebih tinggi daripada dengan kelompok lain. (Nur et al., 2017).

3.2.3.1 TF-IDF

TF-IDF adalah pendekatan dalam pengolahan teks yang membantu menilai pentingnya kata dalam sebuah dokumen. Di bawah ini adalah kode yang memperlihatkan cara menghitung skor TF-IDF menggunakan pustaka sklearn, yang berguna untuk mempermudah perhitungan ini:

```
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
def tfidf():
       preprocessed_data = get_non_empty_preprocessed_data()
       tfidf_result = calculate_tfidf(preprocessed_data)
       tfidf_result_filtered=tfidf_result.loc[~(tfidf_result==0).all(a
xis=1)]
       merged_tfidf = tfidf_result_filtered.mean(axis=0)
       return render_template('tfidf.html', tfidf_result=merged_tfidf)
```

Dalam kode di atas, modul *scikit-learn (sklearn)* digunakan untuk mengimplementasikan algoritma K-Means untuk melakukan pengelompokan (*clustering*) serta penerapan metode *Term Frequency-Inverse Document Frequency* (TF-IDF) dalam ekstraksi fitur pada data teks. Penjelasan lebih lanjut tentang kode ini dapat ditemukan di bawah ini.:

1. Mengimpor modul yang diperlukan:
   - from **`sklearn.cluster`** import K-Means untuk menggunakan algoritma K-Means dari **`scikit-learn`**.
   - from **`sklearn.feature_extraction.text import TfidfVectorizer`** untuk mengimpor fungsi **`TfidfVectorizer`** yang digunakan dalam ekstraksi fitur TF-IDF.
2. Mendefinisikan fungsi **`tfidf():`**
   Fungsi ini tampaknya dirancang untuk digunakan dalam aplikasi web dengan pemanggilan `render_template`.
3. Memperoleh data yang telah di-*preprocess* dan non-kosong dengan menggunakan fungsi **`get_non_empty_preprocessed_data().`** Ini mungkin mengambil data teks yang telah di-preprocess sebelumnya.
4. Menghitung nilai TF-IDF dengan menggunakan fungsi **`calculate_tfidf()`** pada data yang telah di-preprocess. Hasilnya disimpan dalam variabel **`tfidf_result`**.
5. Menghapus baris-baris yang seluruhnya berisi nilai nol (0) dalam **`tfidf_result`** dengan **tfidf_result_filtered**.
6. Menghitung nilai rata-rata (mean) dari setiap fitur dalam **`tfidf_result_filtered`** dengan **tfidf_result_filtered.mean**(axis=0) dan menyimpan hasilnya dalam variabel `merged_tfidf`.
7. Mengembalikan hasil dalam bentuk halaman HTML dengan menggunakan render_template(). Hasil TF-IDF yang telah dihitung, yaitu merged_tfidf, akan digunakan dalam halaman HTML dengan nama 'tfidf.html'. Hasil dari *code* di atas dapat dilihat pada Tabel 3.3.

Tabel 3.3 Hasil TF-IDF

TF IDF	
Kata	TF-IDF Score
aamiin	0.00045369397828467025
abang	0.0007190497853033813
abis	0.00019997485667111047
abissss	0.0021364449670582607
abisssss	0.0005738123775231137
abizzzz	0.0006850319473944953
abu	0.0016971454901483368
ad	0.00012774191329570055
adek	0.000759286438346945
adem	0.00844918963959143

3.2.3.2 Elbow

Elbow bertujuan untuk menemukan jumlah *cluster* yang optimal. Dalam implementasi kode ini, digunakan pustaka sklearn dan matplotlib untuk mempermudah perhitungan serta visualisasi data melalui *code* ini dapat dilihat sebagai:

```
def elbow():
       preprocessed_data = get_non_empty_preprocessed_data()
       tfidf_result = calculate_tfidf(preprocessed_data)
       tfidf_result_filtered = tfidf_result.loc[~(tfidf_result ==
0).all(axis=1)]
       data_for_elbow = tfidf_result_filtered.values

       data_for_elbow = data_for_elbow.reshape(-1, 1)
       max_clusters = 10
       inertias = calculate_inertia(data_for_elbow, max_clusters)

       plt.figure(figsize=(8, 5))
       plt.plot(range(1, max_clusters + 1), inertias, marker='o')
       plt.xlabel('Number of Clusters')
       plt.ylabel('Inertia')
       plt.title('Elbow Method for Optimal K')
       plt.xticks(range(1, max_clusters + 1))
```

```
        plt.grid(True)

        img_buffer = io.BytesIO()
        plt.savefig(img_buffer, format='png')
        img_buffer.seek(0)
        img_base64 =
base64.b64encode(img_buffer.getvalue()).decode('utf-8')

        return render_template('elbow.html', elbow_plot=img_base64)
```

Selanjutnya beralih kepada penjelasan *code* diatas sebagai:

1. **`preprocessed_data = get_non_empty_preprocessed_data()`**: Mengambil data yang telah di-preprocess dan tidak kosong menggunakan fungsi **`get_non_empty_preprocessed_data()`.**
2. **`tfidf_result = calculate_tfidf(preprocessed_data)`**: Menghitung nilai TF-IDF pada data yang telah di-preprocess menggunakan fungsi **`calculate_tfidf()`.** Hasil perhitungannya disimpan dalam **`tfidf_result`.**
3. **`tfidf_result_filtered = tfidf_result.loc[~(tfidf_result == 0).all(axis=1)]`**: Menghapus baris-baris yang seluruhnya berisi nilai nol (0) dari hasil TF-IDF yang dihitung sebelumnya. Ini bertujuan untuk menghindari fitur yang tidak informatif.
4. **`data_for_elbow = tfidf_result_filtered.values`**: Mengambil nilai-nilai hasil TF-IDF yang telah difilter dan menyimpannya dalam **`data_for_elbow`.**
5. **`data_for_elbow = data_for_elbow.reshape(-1, 1)`**: Mereshape data TF-IDF menjadi bentuk yang sesuai untuk analisis.
6. **`max_clusters = 10`**: Mengatur jumlah maksimal klaster yang akan diuji dalam analisis elbow.
7. **`inertias = calculate_inertia(data_for_elbow, max_clusters)`**: Menghitung nilai inertia untuk setiap jumlah klaster menggunakan fungsi **`calculate_inertia()`.**
8. Pembuatan Plot:
    - Membuat plot dengan matplotlib menggunakan **`plt.figure()`.**
    - Melakukan plotting nilai inertias terhadap jumlah klaster.
    - Memberikan label sumbu x dan y, serta judul plot.
    - Menandai nilai klaster pada sumbu x dan mengatur tampilan grid.

9. Pengolahan Gambar:
    - Membuat objek buffer untuk menyimpan gambar yang dihasilkan.
    - Menyimpan gambar dalam format PNG ke objek buffer.
    - Mengatur posisi pointer objek buffer ke awal.
    - Mengkonversi gambar dalam objek buffer menjadi format base64.
10. **`Return render_template('elbow.html', elbow_plot=img_base64)`**: Mengembalikan hasil plot dalam bentuk halaman HTML menggunakan fungsi **`render_template()`**. Hasil plot siku (elbow plot) akan ditampilkan dalam halaman HTML dengan nama 'elbow.html'.

3.2.3.3 Clustering Data

Langkah terakhir adalah melakukan *clustering*, data dikelompokkan ke dalam setiap *cluster*. Jumlah *cluster* ditentukan berdasarkan *cluster* paling optimal pada grafik Elbow. Untuk melakukan *clustering* dapat menggunakan *code*:

```
def clustering():
    preprocessed_data = get_non_empty_preprocessed_data()
    tfidf_result = calculate_tfidf(preprocessed_data)
    tfidf_result_filtered = tfidf_result.loc[~(tfidf_result == 0).all(axis=1)]
    data_for_elbow = tfidf_result_filtered.values
    n_clusters = 2
    data_for_clustering = tfidf_result_filtered.values
    cluster_labels = perform_clustering(data_for_clustering, n_clusters)
    tfidf_result_filtered['Cluster'] = cluster_labels
    merged_data = pd.concat([pd.DataFrame(preprocessed_data, columns=['Preprocessed Data']), tfidf_result_filtered], axis=1)
    return render_template('clustering.html', merged_data=merged_data)
```

hasil *code* di atas dapat dilihat pada Gambar 3.12.

Cluster	Komentar
0	bagus pas kaki
1	bagus ringan belakang rendah kalo jalan mau lepas finishing sepatu miring sedikit gk ama sepatu
1	size konsisten padahal beli celana
1	bagus
1	barang sayang sz minta tak kirim sz blm dpt kirim yg mau kasih
1	produk jg bagus deskripsi terimakasih seller pak kurir
1	mantab jiwa bahan halus nyaman recomended seller
1	keren bahan tebal
1	mantap percaya
1	mantap percaya

Gambar 3.12 Hasil Clustering

Tampak pada kolom terakhir, data telah dimasukkan ke dalam variabel. *cluster* yang sesuai. Setelah melewati tahap *clustering* maka data siap digunakan.

3.2.4 Visualization

Setelah data memasuki proses clustering dan telah dianggap baik maka tahap selanjutnya adalah mempresentasikan data yang dimiliki agar mudah dimengerti oleh orang lain. Salah satu caranya adalah menggunakan *wordcloud*. *Wordcloud* adalah representasi visual dari kata yang paling sering muncul ditampilkan dengan ukuran yang lebih besar dan lebih menonjol. Berikut merupakan *code* untuk menampilkan *wordcloud*:

```
for cluster_id in range(n_clusters):
    cluster_data = merged_data[merged_data['Cluster'] == cluster_id]
    cluster_text = ' '.join(cluster_data['Preprocessed Data'])
    wordcloud = WordCloud(width=800, height=800, background_color='white').generate(cluster_text)
    plt.figure(figsize=(8, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    wordcloud_path = os.path.join(app.config['UPLOAD_FOLDER'], f"wordcloud_cluster_{cluster_id}.png")
```

```
plt.savefig(wordcloud_path)
plt.close()
most_common_words = list(wordcloud.words_.keys())
summary = ', '.join(most_common_words[:5])  # Taking the first 5 most common words as a summary
merged_data.loc[merged_data['Cluster'] == cluster_id, 'WordCloud Summary'] = summary
```

Penjelasan *code* diatas sebagai**:**

1. Looping untuk Setiap Klaster:
    - Loop **`for cluster_id in range(n_clusters)`** digunakan untuk mengulangi langkah berikutnya untuk setiap klaster.
2. Mendapatkan Data Klaster:
    - **`cluster_data`** diisi dengan baris-baris data dari DataFrame **`merged_data`** yang termasuk dalam klaster dengan ID yang sedang di-loop.
3. Menggabungkan Teks Klaster:
    - **`cluster_text`** berisi teks yang telah digabungkan dari kolom *'Preprocessed* Data' dalam klaster yang sedang di-loop.
4. Membuat WordCloud:
    - *Wordcloud* dibuat dengan menggunakan pustaka Wordcloud..
5. Menampilkan WordCloud:
    - Menggunakan matplotlib, WordCloud ditampilkan dalam bentuk gambar dengan fungsi **`plt.imshow().`**
    - **`plt.axis('off')`** menghilangkan sumbu pada plot gambar.
6. Mengambil Kata-kata Paling Umum:
    - Mengambil kata-kata paling umum dari WordCloud dengan **`wordcloud.words_.keys()`**. Ini akan menghasilkan daftar kata-kata paling umum.
7. Menyimpan Ringkasan dalam DataFrame:
    - Ringkasan klaster dalam bentuk kata-kata paling umum ditambahkan ke kolom 'WordCloud Summary' dalam DataFrame **merged_data** yang sesuai dengan klaster yang sedang di-loop.

Semakin banyak kata yang muncul dalam wordcloud maka semakin kompleks topik yang dibahas pada *cluster* tersebut. Selain wordcloud, data juga dapat divisualisasikan dalam bentuk lain. Library matplotlib dan seaborn menyediakan banyak cara untuk memvisualisasikan data yang dimiliki, contohnya adalah pie chart dan bar chart. Berikut adalah code untuk menampilkan data dengan pie chart dan bar chart:

```
        cluster_percentage                                          =
merged_data['Cluster'].value_counts(normalize=True) * 100
        plt.figure(figsize=(8, 6))
        colors = ['green', 'purple']
        plt.pie(cluster_percentage,     labels=cluster_percentage.index,
autopct='%1.1f%%', colors=colors)
        plt.tight_layout()

        cluster_percentage_plot_path                                =
os.path.join(app.config['UPLOAD_FOLDER'], "cluster_percentage.png")
        plt.savefig(cluster_percentage_plot_path)
        plt.close()

        cluster_count = merged_data['Cluster'].value_counts()
        plt.figure(figsize=(8, 6))
        ax = cluster_count.plot(kind='bar', color='purple')
        plt.title('Cluster Count')
        plt.xlabel('Cluster')
        plt.ylabel('Count')
        plt.xticks(rotation=0)
        plt.tight_layout()

        for p in ax.patches:
            ax.annotate(str(p.get_height()), (p.get_x() + p.get_width()
/ 2., p.get_height()), ha='center', va='bottom')

        cluster_count_plot_path                                     =
os.path.join(app.config['UPLOAD_FOLDER'], "cluster_count.png")
        plt.savefig(cluster_count_plot_path)
        plt.close()

return      render_template('home.html',      merged_data=merged_data,
sentiment_plot_path=sentiment_plot_path,      cluster_ids=cluster_ids,
cluster_percentage_plot_path=cluster_percentage_plot_path,
cluster_count_plot_path=cluster_count_plot_path)
```

Dengan menampilkan data menggunakan *chart* maka dapat dengan jelas dilihat *cluster* mana yang memiliki jumlah data terbanyak. penjelasan *code* sebagai berikut :

1. Menghitung Persentase Klaster:
    - `cluster_percentage` dihitung dengan menghitung jumlah nilai dalam kolom '*cluster*' dan mengonversi menjadi persentase.
    - `normalize=True` mengubah nilai menjadi persentase, kemudian dikalikan 100.
2. Membuat Plot Pie Persentase Klaster:
    - Plot pie (lingkaran) dibuat untuk menampilkan persentase klaster menggunakan `plt.pie()`.
    - Warna untuk klaster diberikan dengan `colors`.
    - `autopct='%1.1f%%'` menambahkan persentase pada setiap bagian pie.
    - Hasil gambar pie disimpan dan ditutup menggunakan `plt.savefig()` dan **plt.close()**.
3. Membuat Plot Jumlah Klaster:
    - Jumlah klaster untuk setiap label dihitung dengan `merged_data['Cluster'].value_counts()`.
    - Plot batang (bar) dibuat dengan `cluster_count.plot(kind='bar', color='purple')`.
4. Menambahkan Label pada Plot Jumlah Klaster:
    - Angka jumlah klaster ditambahkan pada bagian atas setiap bar menggunakan `ax.annotate()`.
5. Menyimpan Plot Jumlah Klaster:
    - Plot jumlah klaster disimpan dalam file menggunakan `plt.savefig()` dan ditutup dengan `plt.close()`.
6. Mengembalikan Hasil:
    - Hasil akhir dari analisis klaster, serta hasil-hasil visualisasi dan plot yang telah disimpan, akan dikembalikan dalam bentuk halaman HTML menggunakan `render_template()`.

Ini adalah langkah akhir dalam analisis klaster dan visualisasi data yang telah diproses sebelumnya. Hasil analisis dan visualisasi ini ditampilkan dalam halaman web bernama 'home.html'.