BAB IV
HASIL DAN PEMBAHASAN

4.1. Hasil

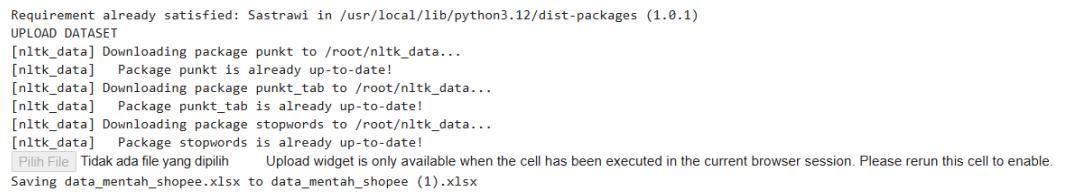
4.1.1. *Install* dan *Import Library*

Pada bagian pertama *script* dilakukan proses instalasi dan pemanggilan berbagai *Library* yang dibutuhkan dalam penelitian analisis sentimen. *Library* yang digunakan antara lain Sastrawi, pandas, string, nltk, dan *scikit-learn*. *Library* Sastrawi digunakan untuk proses *Stemming* bahasa Indonesia sehingga kata yang memiliki imbuhan dapat dikembalikan ke bentuk dasar. *Library* pandas digunakan untuk membaca dan mengelola dataset dalam bentuk tabel atau *DataFrame*. Selain itu digunakan juga *Library* string untuk membantu proses pembersihan teks dari tanda baca. *Library* nltk digunakan untuk proses pengolahan bahasa alami seperti *Tokenisasi* dan *Stopword removal*.

```
# ====================================
# 1. INSTALL DAN IMPORT LIBRARY
# ====================================
!pip install Sastrawi

import pandas as pd
import string
import nltk

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

from google.colab import files

# download resource nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
```

Pada gambar di atas merupakan tahap instalasi dan *Import Library* yang digunakan dalam penelitian analisis sentimen terhadap ulasan aplikasi *Shopee*. Pada tahap ini sistem terlebih dahulu melakukan instalasi *Library* Sastrawi menggunakan perintah!pip *Install* Sastrawi. *Library* ini digunakan untuk melakukan proses *Stemming* pada teks berbahasa Indonesia. Selain itu beberapa *Library* lain juga di*Import* seperti pandas, string, dan nltk yang berfungsi untuk melakukan pengolahan data teks. *Library* pandas digunakan untuk membaca dan mengelola dataset dalam bentuk tabel. Sementara itu *Library* nltk digunakan untuk melakukan berbagai proses pengolahan bahasa alami seperti *Tokenisasi* dan *Stopword removal*.

Pada gambar di atas juga terlihat proses *Import* beberapa modul dari *Library scikit-learn* yang digunakan untuk membangun model *machine learning*. Modul yang digunakan antara lain train_test_split, *TfIDFVectorizer*, dan MultinomialNB. Modul train_test_split digunakan untuk membagi dataset menjadi data training dan data testing. Modul *TfIDFVectorizer* digunakan untuk melakukan pembobotan kata menggunakan metode *TF-IDF*. Setelah dilakukan proses Install dan Import data, maka kemudian diperoleh hasil yaitusebagai berikut.



Gambar 4. 1. Hasil Install dan Import Library

Pada gambar di atas merupakan tahap instalasi dan *Import Library* yang digunakan dalam proses analisis sentimen terhadap ulasan pengguna aplikasi *Shopee*. Pada tahap ini dilakukan instalasi *Library* Sastrawi yang digunakan untuk

proses *Stemming* bahasa Indonesia. Selain itu dilakukan juga *Import* berbagai *Library* seperti pandas, string, nltk, serta beberapa modul dari *scikit-learn* yang digunakan dalam proses *machine learning*. *Library* pandas digunakan untuk membaca dan mengelola dataset, sedangkan nltk digunakan untuk berbagai proses pengolahan teks seperti *Tokenisasi* dan *Stopword removal*. Modul dari *scikit-learn* digunakan untuk pembobotan kata menggunakan *TF-IDF*, pembagian data, serta proses klasifikasi menggunakan metode *Naive Bayes*. Tahap ini merupakan langkah awal agar seluruh proses analisis dapat dijalankan dengan baik.

Berdasarkan hasil yang diperoleh terlihat bahwa *Library* Sastrawi telah berhasil terinstal pada sistem dengan status *Requirement already satisfied*. Hal ini menunjukkan bahwa *Library* tersebut telah tersedia pada lingkungan *Google Colab* sehingga dapat langsung digunakan tanpa perlu instalasi ulang. Selain itu resource dari *Library* NLTK seperti *punkt*, *punkt_tab*, dan *Stopwords* juga berhasil diunduh dengan status *already up-to-date*. Kondisi ini menandakan bahwa semua kebutuhan *Library* untuk proses pengolahan teks telah siap digunakan. Dengan tersedianya seluruh *Library* tersebut, maka proses *preprocessing* teks dapat berjalan dengan optimal.

Keberhasilan instalasi dan pemanggilan *Library* menjadi dasar penting dalam proses analisis sentimen. Tanpa *Library* yang sesuai, proses pengolahan data teks dan pembangunan model *machine learning* tidak dapat dilakukan. Dengan tersedianya seluruh *Library* yang diperlukan, maka tahapan penelitian selanjutnya dapat dijalankan dengan lancar. Selain itu penggunaan *Library* yang tepat juga membantu mempermudah proses implementasi metode yang digunakan. Oleh

karena itu tahap ini memastikan bahwa sistem telah siap untuk melakukan proses analisis data ulasan pengguna aplikasi *Shopee*.

4.1.2. *Upload Dataset*

Pada tahap kedua dilakukan proses pengunggahan dataset ke dalam *Google Colab*. Dataset yang digunakan dalam penelitian ini merupakan data ulasan aplikasi *Shopee* yang sebelumnya diperoleh melalui proses *scraping* dari *Google Play Store*. *File* dataset tersebut kemudian diunggah menggunakan fungsi *files.upload()* yang memungkinkan pengguna memilih *file* dari perangkat komputer. Setelah *file* berhasil diunggah, data tersebut dibaca menggunakan fungsi *pd.read_excel()* sehingga dapat diolah dalam bentuk *DataFrame* menggunakan *Library* pandas. *DataFrame* ini memudahkan proses manipulasi data, analisis, serta pengolahan teks pada tahap selanjutnya.

```
# ====================================
# 2. UPLOAD DATASET
# ====================================
print("UPLOAD DATASET")
uploaded = files.upload()

df = pd.read_excel('data_mentah_shopee.xlsx')

print("\nDATASET AWAL")
print(df.head())

print("\nJUMLAH DATA :", len(df))
```

Pada gambar di atas merupakan tahap pengunggahan dataset ke dalam lingkungan pemrograman *Google Colab*. Dataset yang digunakan dalam penelitian ini merupakan data ulasan pengguna aplikasi *Shopee* yang diperoleh dari *Google Play Store* melalui proses *scraping* data. Dataset tersebut kemudian diunggah

menggunakan fungsi *files.upload()* sehingga *file* dapat dipilih langsung dari perangkat pengguna. Setelah proses upload selesai, dataset dibaca menggunakan fungsi *pd.read_excel()* dari *Library* pandas. Fungsi ini memungkinkan sistem membaca *file* berformat Excel dan mengubahnya menjadi *DataFrame*. *DataFrame* merupakan struktur data berbentuk tabel yang memudahkan proses pengolahan dan analisis data.

Pada gambar di atas juga terlihat bahwa sistem menampilkan beberapa baris awal dataset menggunakan fungsi *head()*. Tujuan dari langkah ini adalah untuk memastikan bahwa dataset telah berhasil dimuat dengan benar. Data yang ditampilkan biasanya berisi kolom content yang berisi teks ulasan dan kolom score yang berisi nilai rating dari pengguna. Informasi ini sangat penting karena akan digunakan dalam proses analisis sentimen. Selain itu *script* juga menampilkan jumlah total data menggunakan fungsi *len(df)*. Informasi jumlah data ini berguna untuk mengetahui ukuran dataset yang akan dianalisis. Setelah dilakukan upload dataset, maka diperoleh hasil upload dataset yaitu sebagai berikut.

```
DATASET AWAL
                                                content  score
0  setiap pengajuan, kerna barang gak sesuai past...        1
1  shopee Indonesia sangat bagus & banyak sekali ...        5
2  Shopee emang paling terbaik buat pesan2 barang...        5
3                                             mantap        1
4          bagus sih emang paling top markotop deh👍 👍        5

JUMLAH DATA : 1000
```

Gambar 4. 2. Hasil Upload Dataset

Pada gambar di atas merupakan tahap pengunggahan dataset ke dalam lingkungan *Google Colab*. Dataset yang digunakan dalam penelitian ini merupakan data ulasan pengguna aplikasi *Shopee* yang diperoleh dari *Google Play Store*

melalui proses *scraping* data. Data tersebut disimpan dalam bentuk *file* Excel dengan nama data_mentah_*Shopee*.xlsx. *File* tersebut kemudian diunggah menggunakan fitur *files.upload()* sehingga dapat dibaca oleh sistem. Setelah proses upload selesai, dataset dibaca menggunakan fungsi *pd.read_excel()* dari *Library* pandas. Tahap ini bertujuan untuk memuat dataset ke dalam sistem agar dapat diproses lebih lanjut.

Berdasarkan hasil yang ditampilkan terlihat bahwa dataset berhasil dimuat ke dalam sistem dengan menampilkan beberapa baris pertama data. Data tersebut terdiri dari kolom content yang berisi teks ulasan pengguna dan kolom score yang berisi nilai rating yang diberikan oleh pengguna. Contoh ulasan yang ditampilkan menunjukkan variasi opini pengguna terhadap aplikasi *Shopee*. Beberapa ulasan memiliki rating tinggi yang menunjukkan kepuasan pengguna, sementara ulasan lainnya memiliki rating rendah yang menunjukkan ketidakpuasan. Selain itu sistem juga menampilkan jumlah total data yang digunakan yaitu sebanyak 1000 data ulasan.

Jumlah data tersebut menunjukkan bahwa dataset yang digunakan cukup besar untuk melakukan analisis sentimen. Dengan jumlah data yang cukup banyak, model klasifikasi dapat mempelajari pola sentimen secara lebih baik. Data yang beragam juga membantu meningkatkan kualitas analisis yang dilakukan.

4.1.3. Konversi Score Menjadi Sentimen

Pada tahap ini dilakukan proses konversi nilai rating menjadi kategori sentimen. Data ulasan dari *Google Play Store* biasanya memiliki nilai rating dalam bentuk angka antara 1 hingga 5. Nilai tersebut kemudian diubah menjadi kategori

sentimen yaitu positif, negatif, dan netral. Proses ini dilakukan dengan menggunakan fungsi *convert_sentiment()* yang mengelompokkan nilai rating berdasarkan kriteria tertentu. Jika nilai rating lebih besar atau sama dengan 4 maka dikategorikan sebagai sentimen positif. Jika nilai rating kurang dari atau sama dengan 2 maka dikategorikan sebagai sentimen negatif. Sedangkan nilai rating 3 dianggap sebagai sentimen netral.

```
# ====================================
# 3. KONVERSI SCORE MENJADI SENTIMEN
# ====================================
def convert_sentiment(score):

    if score >= 4:
        return "positif"

    elif score <= 2:
        return "negatif"

    else:
        return "netral"

df['sentimen'] = df['score'].apply(convert_sentiment)

# buang netral
df = df[df['sentimen'] != 'netral']

print("\nDATASET SETELAH KONVERSI SENTIMEN")
print(df[['content','score','sentimen']].head())
```

Pada gambar di atas merupakan proses konversi nilai rating menjadi kategori sentimen. Data ulasan yang diperoleh dari *Google Play Store* biasanya memiliki nilai rating dalam bentuk angka dari 1 hingga 5. Nilai tersebut kemudian diubah menjadi kategori sentimen yaitu positif, negatif, dan netral. Proses konversi ini dilakukan dengan menggunakan fungsi *convert_sentiment()* yang dibuat dalam

script. Jika nilai rating lebih besar atau sama dengan 4 maka ulasan dikategorikan sebagai sentimen positif. Jika nilai rating kurang dari atau sama dengan 2 maka ulasan dikategorikan sebagai sentimen negatif. Sedangkan nilai rating 3 dikategorikan sebagai sentimen netral.

Pada gambar di atas setelah proses konversi dilakukan, sistem menambahkan kolom baru yaitu sentimen pada dataset. Kolom ini berisi label sentimen yang dihasilkan dari proses konversi rating. Label sentimen ini nantinya akan digunakan sebagai target dalam proses klasifikasi. Dengan adanya label ini, model *machine learning* dapat mempelajari hubungan antara kata-kata dalam ulasan dengan kategori sentimen. *Script* kemudian menampilkan contoh beberapa data yang telah memiliki label sentimen. Hal ini dilakukan untuk memastikan bahwa proses konversi telah berjalan dengan baik. Setelah dilakukan konversi score menjadi sentiment yaitu sebagai berikut.

DATASET SETELAH KONVERSI SENTIMEN

	content	score	sentimen
0	setiap pengajuan, kerna barang gak sesuai past...	1	negatif
1	shopee Indonesia sangat bagus & banyak sekali ...	5	positif
2	Shopee emang paling terbaik buat pesan2 barang...	5	positif
3	mantap	1	negatif
4	bagus sih emang paling top markotop deh👍 👍	5	positif

Gambar 4. 3. Konversi Score Menjadi Sentimen

Pada gambar di atas merupakan tahap konversi nilai rating menjadi kategori sentimen. Nilai rating yang diberikan pengguna pada *Google Play Store* berada pada rentang angka 1 hingga 5. Pada penelitian ini nilai rating tersebut dikonversi menjadi kategori sentimen yaitu positif, negatif, dan netral. Rating dengan nilai 4 dan 5 dikategorikan sebagai sentimen positif, rating 1 dan 2 sebagai sentimen

negatif, sedangkan rating 3 dikategorikan sebagai sentimen netral. Proses konversi ini dilakukan menggunakan fungsi *convert_sentiment()* dalam *script*. Tahap ini bertujuan untuk menghasilkan label sentimen yang akan digunakan dalam proses klasifikasi.

Berdasarkan hasil yang ditampilkan terlihat bahwa setiap data ulasan kini memiliki kolom tambahan yaitu sentimen. Kolom ini menunjukkan kategori sentimen dari setiap ulasan berdasarkan nilai rating yang diberikan pengguna. Contoh data menunjukkan bahwa ulasan dengan rating rendah dikategorikan sebagai sentimen negatif, sedangkan ulasan dengan rating tinggi dikategorikan sebagai sentimen positif. Hasil ini menunjukkan bahwa proses konversi sentimen telah berjalan dengan baik. Data yang telah memiliki label sentimen ini kemudian dapat digunakan dalam proses analisis selanjutnya.

Pada tahap ini juga dilakukan proses penghapusan data dengan sentimen netral. Hal ini dilakukan agar proses klasifikasi hanya berfokus pada dua kategori utama yaitu positif dan negatif. Dengan menghapus data netral, model klasifikasi dapat bekerja lebih fokus dalam membedakan dua kelas sentimen. Selain itu langkah ini juga membantu meningkatkan performa model klasifikasi. Dataset yang tersisa kemudian akan diproses pada tahap pembersihan data. Tahap ini menjadi dasar penting dalam proses analisis sentimen.

4.1.4. *Cleaning text*

Tahap *Cleaning text* merupakan proses awal dalam pembersihan data teks. Pada tahap ini dilakukan proses penghapusan berbagai karakter yang tidak diperlukan dalam analisis teks. Misalnya tanda baca, simbol, dan karakter khusus

yang tidak memiliki makna dalam analisis sentimen. Selain itu seluruh huruf pada teks juga diubah menjadi huruf kecil *(lowercase)* menggunakan metode *case folding*. Tujuan dari proses ini adalah untuk menghindari perbedaan antara huruf besar dan huruf kecil yang sebenarnya memiliki makna yang sama. Dengan demikian kata seperti “Bagus” dan “bagus” akan dianggap sebagai kata yang sama dalam proses analisis.

```
# ====================================
# 4. CLEANING TEXT
# ====================================
def cleaning(text):

    text = str(text).lower()
    text = text.translate(str.maketrans('', '', string.punctuation))

    return text

df['cleaning'] = df['content'].apply(cleaning)

print("\nHASIL CLEANING")
print(df[['content','cleaning']].head())
```

Pada gambar di atas merupakan tahap *Cleaning text* yang merupakan bagian dari proses pembersihan data teks. Pada tahap ini dilakukan penghapusan berbagai karakter yang tidak diperlukan dalam analisis teks seperti tanda baca dan simbol. Selain itu seluruh huruf dalam teks juga diubah menjadi huruf kecil menggunakan metode *case folding*. Tujuan dari proses ini adalah untuk menyamakan format penulisan kata sehingga tidak terjadi perbedaan antara huruf besar dan huruf kecil. Misalnya kata “Bagus” dan “bagus” akan dianggap sebagai kata yang sama dalam analisis. Proses *Cleaning* dilakukan menggunakan fungsi *Cleaning*() yang telah dibuat dalam *script*.

Pada gambar di atas fungsi *Cleaning* bekerja dengan menggunakan metode lower() untuk mengubah huruf menjadi huruf kecil. Selain itu digunakan juga fungsi *translate()* untuk menghapus tanda baca dari teks. Dengan menghapus tanda baca, teks menjadi lebih sederhana dan mudah diproses. Proses ini sangat penting karena tanda baca biasanya tidak memberikan informasi penting dalam analisis sentimen. *Script* kemudian menerapkan fungsi *Cleaning* pada seluruh data ulasan menggunakan metode *apply()*. Hasilnya disimpan dalam kolom baru bernama *Cleaning*. Setelah dilakukan proses *Cleaning text*, maka diperoleh hasil sebagai berikut.

```
HASIL CLEANING
                                                content  \
0  setiap pengajuan, kerna barang gak sesuai past...
1  shopee Indonesia sangat bagus & banyak sekali ...
2  Shopee emang paling terbaik buat pesan2 barang...
3                                              mantap
4          bagus sih emang paling top markotop deh👍 👍

                                               cleaning
0  setiap pengajuan kerna barang gak sesuai pasti...
1  shopee indonesia sangat bagus  banyak sekali m...
2  shopee emang paling terbaik buat pesan2 barang...
3                                              mantap
4          bagus sih emang paling top markotop deh👍 👍
```

Gambar 4. 4. Hasil Cleaning

Pada gambar di atas merupakan tahap *Cleaning text* yang bertujuan untuk membersihkan teks dari karakter yang tidak diperlukan. Proses ini meliputi penghapusan tanda baca, simbol, serta perubahan huruf menjadi huruf kecil menggunakan metode *case folding*. Pembersihan teks dilakukan agar data menjadi lebih terstruktur dan mudah diproses pada tahap selanjutnya. Fungsi *Cleaning*()

digunakan untuk melakukan proses ini pada setiap data ulasan. Dengan proses ini teks yang sebelumnya memiliki berbagai simbol dan tanda baca akan menjadi lebih sederhana. Tahap ini merupakan bagian penting dari *preprocessing* data.

Berdasarkan hasil yang ditampilkan terlihat bahwa teks ulasan telah mengalami perubahan setelah proses *Cleaning* dilakukan. Teks yang sebelumnya mengandung tanda baca dan huruf kapital kini telah berubah menjadi teks sederhana dengan huruf kecil. Misalnya simbol seperti & atau tanda baca lainnya telah dihapus dari teks. Perubahan ini membuat teks menjadi lebih bersih dan mudah dianalisis. Selain itu kata-kata yang sebelumnya memiliki format berbeda kini menjadi seragam. Hal ini membantu proses analisis kata pada tahap selanjutnya.

Proses *Cleaning* membantu mengurangi noise dalam data teks. Noise merupakan karakter atau informasi yang tidak memiliki makna penting dalam analisis sentimen. Dengan menghilangkan noise tersebut, model *machine learning* dapat lebih fokus pada kata-kata yang memiliki makna penting. Hal ini juga membantu meningkatkan kualitas fitur yang akan digunakan dalam proses klasifikasi. Oleh karena itu tahap *Cleaning text* merupakan langkah penting dalam proses *preprocessing* data teks.

4.1.5. *Tokenisasi*

Tokenisasi merupakan proses memecah kalimat menjadi kata-kata tunggal yang disebut sebagai token. Proses ini dilakukan menggunakan fungsi *word_tokenize()* dari *Library* NLTK. Tujuan dari *Tokenisasi* adalah untuk mengubah teks yang awalnya berupa kalimat panjang menjadi daftar kata yang

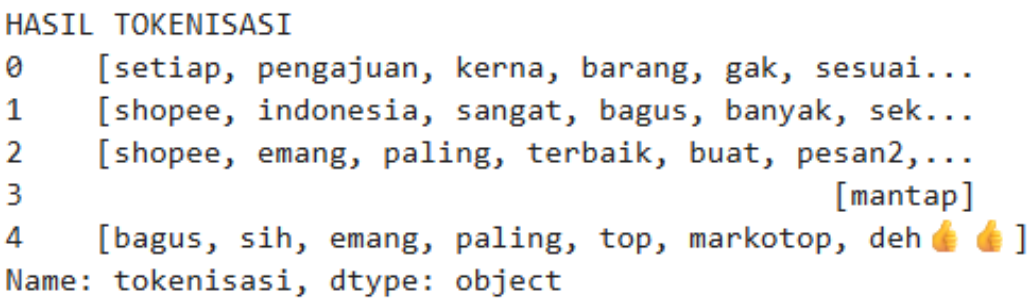
lebih mudah dianalisis. Dengan memecah kalimat menjadi kata, sistem dapat mengetahui kata apa saja yang muncul dalam suatu ulasan. Informasi ini sangat penting dalam proses analisis teks dan pembobotan kata. *Tokenisasi* juga membantu dalam proses penghapusan *Stopword* dan *Stemming* pada tahap berikutnya.

```
# ====================================
# 5. TOKENISASI
# ====================================
df['tokenisasi'] = df['cleaning'].apply(word_tokenize)

print("\nHASIL TOKENISASI")
print(df['tokenisasi'].head())
```

Pada gambar di atas merupakan tahap *Tokenisasi* yang dilakukan setelah proses *Cleaning text* selesai. *Tokenisasi* adalah proses memecah teks atau kalimat menjadi bagian-bagian kecil berupa kata yang disebut sebagai token. Proses ini dilakukan menggunakan fungsi word_tokenize() dari *Library* NLTK. Dengan *Tokenisasi*, kalimat yang sebelumnya berbentuk teks panjang dapat diubah menjadi daftar kata yang lebih mudah dianalisis. Proses ini sangat penting dalam pengolahan bahasa alami karena banyak metode analisis teks yang bekerja berdasarkan kata. Dengan memecah kalimat menjadi token, sistem dapat mengidentifikasi kata-kata yang muncul dalam setiap ulasan pengguna.

Pada gambar di atas setiap teks yang telah melalui proses *Cleaning* kemudian diproses menggunakan fungsi *Tokenisasi*. Hasil dari proses ini berupa daftar kata yang tersimpan dalam bentuk list atau array. Misalnya kalimat “aplikasi ini sangat bagus” akan berubah menjadi daftar kata seperti [aplikasi, ini, sangat, bagus]. Bentuk daftar kata ini memudahkan proses analisis selanjutnya seperti penghapusan *Stopword* dan *Stemming*. *Tokenisasi* juga membantu dalam menghitung frekuensi

kemunculan kata dalam suatu dokumen. Informasi frekuensi ini nantinya digunakan dalam proses pembobotan kata menggunakan metode *TF-IDF*. Setelah dilakukan tokenisasi, maka diperoleh hasil tokenisasi yaitu sebagai berikut.



Gambar 4. 5. Hasil Tokenisasi

Pada gambar di atas merupakan tahap *Tokenisasi* dalam proses *preprocessing* data teks. *Tokenisasi* merupakan proses pemecahan teks menjadi bagian-bagian kecil berupa kata atau token. Proses ini dilakukan menggunakan fungsi word_tokenize() dari *Library* NLTK. Dengan *Tokenisasi*, kalimat ulasan yang sebelumnya berbentuk teks utuh akan dipecah menjadi daftar kata yang lebih mudah dianalisis. Tahap ini bertujuan untuk mempermudah proses pengolahan teks pada tahap selanjutnya seperti *Stopword removal* dan *Stemming*. *Tokenisasi* merupakan langkah penting dalam analisis teks karena membantu sistem mengenali setiap kata yang terdapat dalam ulasan pengguna.

Berdasarkan hasil yang ditampilkan terlihat bahwa setiap ulasan telah diubah menjadi kumpulan kata dalam bentuk list. Misalnya sebuah kalimat ulasan yang sebelumnya berbentuk teks panjang kini dipecah menjadi beberapa kata seperti *Shopee*, bagus, manfaat, dan sebagainya. Hasil *Tokenisasi* ini menunjukkan bahwa sistem berhasil memisahkan setiap kata yang terdapat dalam teks ulasan. Proses ini

memungkinkan sistem untuk menganalisis setiap kata secara terpisah. Dengan demikian setiap kata dapat diproses lebih lanjut pada tahap berikutnya.

Tokenisasi membantu mempersiapkan data teks agar lebih mudah diproses oleh sistem analisis sentimen. Dengan memecah kalimat menjadi kata-kata, sistem dapat mengidentifikasi kata yang memiliki makna sentimen tertentu. Selain itu proses ini juga membantu dalam menghapus kata yang tidak penting pada tahap berikutnya. Hasil *Tokenisasi* ini kemudian digunakan sebagai input dalam proses *Stopword removal*. Dengan demikian tahap *Tokenisasi* menjadi dasar penting dalam proses analisis teks.

4.1.6. *Stopword removal*

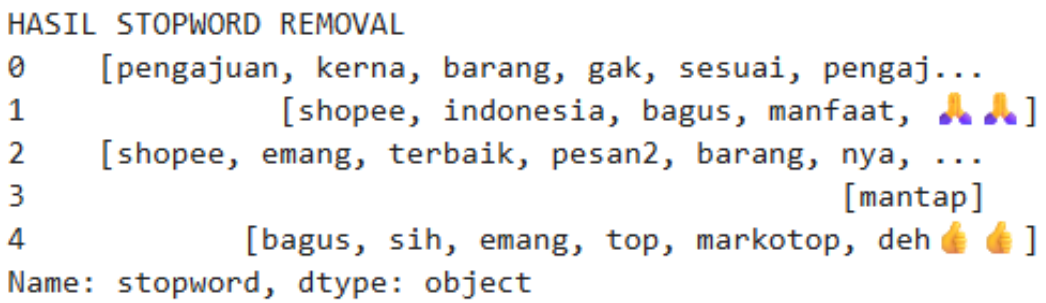
Stopword removal merupakan proses menghapus kata-kata umum yang tidak memiliki makna penting dalam analisis teks. Contoh kata *Stopword* dalam bahasa Indonesia antara lain “dan”, “yang”, “di”, “ke”, dan sebagainya. Kata-kata tersebut sering muncul dalam kalimat tetapi tidak memberikan informasi penting mengenai sentimen suatu ulasan. Oleh karena itu kata tersebut biasanya dihapus agar analisis lebih fokus pada kata yang memiliki makna penting. Proses ini dilakukan menggunakan daftar *Stopword* dari *Library* NLTK untuk bahasa Indonesia. Dengan menghapus *Stopword*, jumlah kata yang dianalisis menjadi lebih sedikit namun lebih relevan.

```
# ====================================
# 6. STOPWORD REMOVAL
# ====================================
stop_words = set(stopwords.words('indonesian'))

df['stopword'] = df['tokenisasi'].apply(
    lambda words: [word for word in words if word not in stop_words]
)

print("\nHASIL STOPWORD REMOVAL")
print(df['stopword'].head())
```

Pada gambar di atas merupakan tahap *Stopword removal* yang bertujuan untuk menghapus kata-kata umum yang tidak memiliki makna penting dalam analisis teks. *Stopword* biasanya berupa kata yang sering muncul dalam kalimat namun tidak memberikan informasi khusus mengenai sentimen suatu ulasan. Contoh kata *Stopword* dalam bahasa Indonesia antara lain “dan”, “yang”, “di”, “ke”, dan “dari”. Kata-kata tersebut biasanya tidak berpengaruh terhadap penentuan sentimen positif maupun negatif. Oleh karena itu kata tersebut dihapus agar analisis lebih fokus pada kata-kata yang memiliki makna penting. Proses ini dilakukan menggunakan daftar *Stopword* dari *Library* NLTK.

Pada gambar di atas proses *Stopword removal* dilakukan dengan membuat variabel stop_words yang berisi kumpulan kata *Stopword* bahasa Indonesia. Setelah itu setiap daftar kata hasil *Tokenisasi* diperiksa satu per satu. Kata yang termasuk dalam daftar *Stopword* akan dihapus dari teks. Proses ini dilakukan menggunakan fungsi lambda yang diterapkan pada seluruh data menggunakan metode *apply()*. Hasil dari proses ini adalah daftar kata yang telah dibersihkan dari kata-kata umum yang tidak penting. Data yang dihasilkan kemudian disimpan dalam kolom baru

bernama *Stopword*. Setelah dilakukan proses *Stopword removal*, maka diperoleh hasil sebagai berikut.



Gambar 4. 6. Stopword Removal

Pada gambar di atas merupakan tahap *Stopword removal* yang bertujuan untuk menghapus kata-kata yang tidak memiliki makna penting dalam analisis sentimen. *Stopword* merupakan kata-kata umum yang sering muncul dalam teks namun tidak memberikan kontribusi besar terhadap makna kalimat, seperti kata dan, yang, di, dan sebagainya. Proses ini dilakukan menggunakan daftar *Stopword* bahasa Indonesia yang tersedia pada *Library* NLTK. Kata-kata yang termasuk dalam daftar *Stopword* akan dihapus dari hasil *Tokenisasi*. Tahap ini bertujuan untuk menyisakan kata-kata yang lebih relevan dalam proses analisis sentimen. Dengan demikian kualitas fitur yang digunakan dalam proses klasifikasi dapat menjadi lebih baik.

Berdasarkan hasil yang ditampilkan terlihat bahwa beberapa kata yang sebelumnya muncul pada tahap *Tokenisasi* telah dihapus dari teks. Kata-kata umum seperti kata penghubung atau kata yang tidak memiliki makna sentimen telah dihilangkan dari daftar kata. Hasil ini menunjukkan bahwa proses *Stopword removal* berhasil mengurangi kata-kata yang tidak penting dalam data teks. Dengan berkurangnya kata yang tidak relevan, teks menjadi lebih fokus pada kata yang

memiliki makna utama. Hal ini membantu meningkatkan kualitas analisis pada tahap selanjutnya.

Penghapusan *Stopword* membantu mengurangi jumlah kata yang tidak relevan dalam dataset. Dengan jumlah kata yang lebih sedikit namun lebih bermakna, sistem dapat bekerja dengan lebih efisien. Selain itu proses ini juga membantu meningkatkan akurasi model klasifikasi karena fitur yang digunakan lebih representatif. Kata-kata yang tersisa setelah proses *Stopword removal* akan diproses lebih lanjut pada tahap *Stemming*. Oleh karena itu tahap ini menjadi bagian penting dalam proses *preprocessing* data teks.

4.1.7. *Stemming*

Stemming merupakan proses mengubah kata menjadi bentuk dasar dengan menghilangkan imbuhan seperti awalan, akhiran, maupun sisipan. Dalam bahasa Indonesia banyak kata yang memiliki imbuhan, misalnya “membeli”, “dibeli”, atau “pembelian”. Kata-kata tersebut sebenarnya memiliki makna dasar yang sama yaitu “beli”. Oleh karena itu proses *Stemming* dilakukan agar semua variasi kata tersebut dianggap sebagai kata yang sama. Pada *script* ini proses *Stemming* dilakukan menggunakan *Library* Sastrawi yang khusus dirancang untuk bahasa Indonesia. *Library* ini mampu mengubah kata berimbuhan menjadi kata dasar secara otomatis.

```
# ====================================
# 7. STEMMING
# ====================================
factory = StemmerFactory()
stemmer = factory.create_stemmer()

df['stemming'] = df['stopword'].apply(
    lambda words: [stemmer.stem(word) for word in words]
)

print("\nHASIL STEMMING")
print(df['stemming'].head())


# gabungkan kembali kata
df['text_bersih'] = df['stemming'].apply(lambda x: " ".join(x))

print("\nTEKS SETELAH PREPROCESSING")
print(df[['content','text_bersih']].head())
```

Pada gambar di atas merupakan tahap *Stemming* yang bertujuan untuk mengubah kata menjadi bentuk dasar. Dalam bahasa Indonesia banyak kata yang memiliki imbuhan seperti awalan, akhiran, maupun sisipan. Misalnya kata "membeli", "pembelian", dan "dibeli" sebenarnya memiliki kata dasar yang sama yaitu "beli". Proses *Stemming* dilakukan agar semua variasi kata tersebut dianggap sebagai kata yang sama dalam analisis. Hal ini membantu menyederhanakan bentuk kata dalam dataset. Dengan demikian jumlah variasi kata yang dianalisis menjadi lebih sedikit.

Pada gambar di atas proses *Stemming* dilakukan menggunakan *Library* Sastrawi yang khusus dirancang untuk bahasa Indonesia. *Library* ini memiliki algoritma yang mampu menghapus imbuhan dari kata sehingga menghasilkan bentuk dasar kata. *Script* membuat objek stemmer menggunakan StemmerFactory()

dan kemudian menerapkan fungsi *Stemming* pada setiap kata. Proses ini dilakukan menggunakan metode *apply()* pada kolom *Stopword*. Setiap kata dalam daftar akan diproses satu per satu oleh stemmer. Hasilnya berupa daftar kata yang telah diubah menjadi bentuk dasar. Setelah dilakukan proses steaming, maka terdapat hasil dari steaming yaitu sebagaai berikut.

```
HASIL STEMMING
0    [aju, kerna, barang, gak, sesuai, aju, tolak, ...
1                [shopee, indonesia, bagus, manfaat, ]
2    [shopee, emang, baik, pesan2, barang, nya, ses...
3                                              [mantap]
4              [bagus, sih, emang, top, markotop, deh]
Name: stemming, dtype: object
```

Gambar 4. 7. Hasil Stemming

Pada gambar di atas merupakan tahap *Stemming* yang bertujuan untuk mengubah kata menjadi bentuk dasar. *Stemming* dilakukan untuk menghilangkan imbuhan seperti awalan, akhiran, atau sisipan dalam suatu kata. Proses ini dilakukan menggunakan *Library* Sastrawi yang dirancang khusus untuk *Stemming* bahasa Indonesia. Dengan proses ini kata-kata seperti pengajuan, mengajukan, atau diajukan dapat diubah menjadi bentuk dasar yaitu aju. Tahap ini bertujuan untuk menyederhanakan kata-kata yang memiliki makna yang sama. Dengan demikian sistem dapat mengenali kata yang memiliki arti serupa sebagai satu bentuk kata.

Berdasarkan hasil yang ditampilkan terlihat bahwa beberapa kata mengalami perubahan setelah proses *Stemming* dilakukan. Kata yang sebelumnya memiliki imbuhan kini berubah menjadi bentuk dasar yang lebih sederhana. Misalnya kata pengajuan berubah menjadi aju, sedangkan kata terbaik berubah menjadi baik. Perubahan ini menunjukkan bahwa proses *Stemming* berhasil mengurangi variasi

kata dalam dataset. Dengan demikian kata yang memiliki makna yang sama dapat diperlakukan sebagai satu kata yang sama oleh sistem.

Proses *Stemming* membantu mengurangi kompleksitas data teks. Dengan mengubah kata ke bentuk dasar, jumlah variasi kata dalam dataset menjadi lebih sedikit. Hal ini membantu meningkatkan efisiensi proses analisis dan pembelajaran model *machine learning*. Selain itu *Stemming* juga membantu meningkatkan kualitas fitur yang digunakan dalam proses klasifikasi. Hasil *Stemming* ini kemudian digabungkan kembali menjadi teks bersih yang akan digunakan pada tahap berikutnya.

4.1.8. *TF-IDF*

TF-IDF merupakan metode yang digunakan untuk mengubah teks menjadi data numerik sehingga dapat diproses oleh algoritma *machine learning*. Metode ini menghitung bobot setiap kata berdasarkan frekuensi kemunculannya dalam dokumen serta tingkat kemunculannya di seluruh dokumen. Kata yang sering muncul dalam satu dokumen tetapi jarang muncul di dokumen lain akan memiliki bobot yang lebih tinggi. Sebaliknya kata yang muncul di hampir semua dokumen akan memiliki bobot yang lebih rendah. Dengan cara ini *TF-IDF* mampu menyoroti kata-kata yang paling penting dalam suatu dokumen. Pada *script* ini *TF-IDF* diterapkan pada kolom *text*_bersih.

```
# ====================================
# 8. TF-IDF
# ====================================
tfidf = TfidfVectorizer()

X = tfidf.fit_transform(df['text_bersih'])

y = df['sentimen']

print("\nUKURAN MATRIX TF-IDF :", X.shape)

tfidf_df = pd.DataFrame(X.toarray(), columns=tfidf.get_feature_names_out())

print("\nCONTOH DATA TF-IDF")
print(tfidf_df.head())
```

Pada gambar di atas merupakan proses pembobotan kata menggunakan metode *Term Frequency* – Inverse *Document Frequency* (*TF-IDF*). Metode ini digunakan untuk mengubah data teks menjadi data numerik sehingga dapat diproses oleh algoritma *machine learning*. *TF-IDF* bekerja dengan menghitung tingkat kepentingan suatu kata dalam sebuah dokumen dibandingkan dengan seluruh dokumen dalam dataset. Kata yang sering muncul dalam suatu dokumen tetapi jarang muncul di dokumen lain akan memiliki bobot yang tinggi. Sebaliknya kata yang sering muncul di banyak dokumen akan memiliki bobot yang lebih rendah. Dengan cara ini *TF-IDF* mampu menyoroti kata-kata yang paling relevan dalam setiap dokumen.

Pada gambar di atas proses *TF-IDF* dilakukan menggunakan *TfIDFVectorizer*() dari *Library scikit-learn*. Fungsi ini akan mengubah teks bersih menjadi sebuah matriks numerik yang berisi bobot setiap kata. Matriks ini memiliki ukuran berdasarkan jumlah dokumen dan jumlah fitur kata yang dihasilkan. *Script* kemudian menyimpan hasil *TF-IDF* ke dalam variabel X sebagai fitur input. Sedangkan variabel y digunakan untuk menyimpan label sentimen dari setiap

ulasan. Dengan cara ini dataset siap digunakan dalam proses *machine learning*. Setelah dilakukan proses *TF-IDF*, maka diperoleh hasil *TF-IDF* yaitu sebagai berikut.

```
CONTOH DATA TF-IDF
    10  100  1000  109  10k  10rb   11   12  12000  12jt  ...  yahapa  yahh  \
0  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  0.0   0.0  0.0  0.0   0.0  0.0  0.0    0.0   0.0  ...     0.0   0.0
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
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
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

   yakin  yata  yatemanteman  yatidak   yg  ygngutang  you  zosss
0    0.0   0.0           0.0      0.0  0.0        0.0  0.0    0.0
1    0.0   0.0           0.0      0.0  0.0        0.0  0.0    0.0
2    0.0   0.0           0.0      0.0  0.0        0.0  0.0    0.0
3    0.0   0.0           0.0      0.0  0.0        0.0  0.0    0.0
4    0.0   0.0           0.0      0.0  0.0        0.0  0.0    0.0

[5 rows x 1981 columns]
```

Gambar 4. 8. Hasil TF-IDF

Pada gambar di atas merupakan tahap pembobotan kata menggunakan metode *TF-IDF* (*Term Frequency* – Inverse *Document Frequency*). Metode ini digunakan untuk mengubah data teks menjadi bentuk numerik yang dapat diproses oleh algoritma *machine learning*. *TF-IDF* bekerja dengan memberikan bobot pada setiap kata berdasarkan seberapa sering kata tersebut muncul dalam suatu dokumen dan seberapa jarang kata tersebut muncul pada seluruh dokumen. Dengan metode ini kata yang sering muncul namun hanya pada dokumen tertentu akan memiliki bobot yang lebih tinggi. Proses ini dilakukan menggunakan *TfIDFVectorizer* dari *Library scikit-learn*. Hasil dari proses ini berupa matriks numerik yang mewakili setiap kata dalam dataset.

Berdasarkan hasil yang ditampilkan terlihat bahwa ukuran matriks *TF-IDF* adalah (969, 1981). Hal ini menunjukkan bahwa terdapat 969 data ulasan yang telah

diproses dan 1981 fitur kata yang dihasilkan dari proses pembobotan. Setiap baris dalam matriks mewakili satu dokumen ulasan, sedangkan setiap kolom mewakili satu kata unik dalam dataset. Nilai pada matriks tersebut menunjukkan bobot pentingnya suatu kata dalam dokumen tertentu. Semakin besar nilai *TF-IDF* maka semakin penting kata tersebut dalam dokumen tersebut.

Proses pembobotan *TF-IDF* sangat penting dalam analisis sentimen berbasis *machine learning*. Dengan metode ini data teks yang awalnya tidak dapat diproses oleh algoritma komputer dapat diubah menjadi data numerik. Representasi numerik ini memungkinkan model *machine learning* untuk mempelajari pola kata dalam dataset. Selain itu *TF-IDF* juga membantu mengurangi pengaruh kata yang terlalu sering muncul namun tidak memiliki makna penting. Hasil dari proses ini kemudian digunakan sebagai input pada tahap pembagian data training dan testing.

4.1.9. Split Data

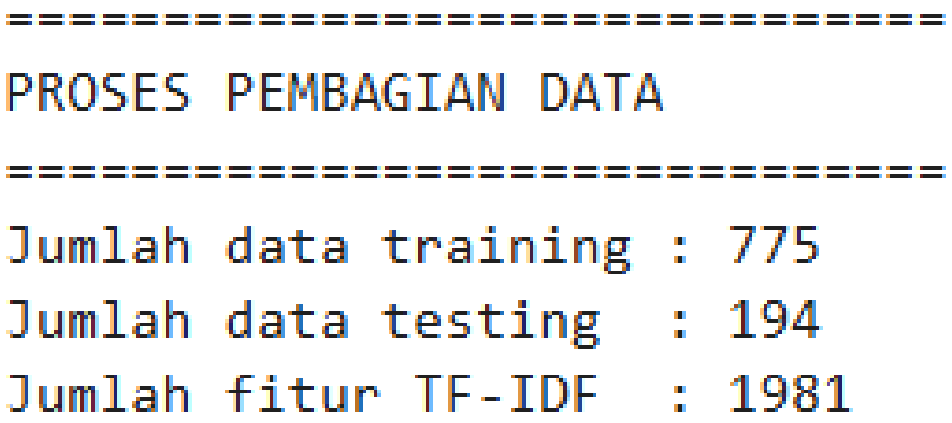
Tahap split data merupakan proses membagi dataset menjadi data training dan data testing. Pembagian ini dilakukan menggunakan fungsi *train_test_split()* dari *Library scikit-learn*. Data training digunakan untuk melatih model klasifikasi agar dapat mempelajari pola dalam data. Sedangkan data testing digunakan untuk menguji kemampuan model dalam memprediksi data baru. Dalam *script* ini digunakan parameter test_size = 0.2 yang berarti 20% data digunakan sebagai data testing. Sedangkan 80% data lainnya digunakan sebagai data training.

```
# ====================================
# 9. SPLIT DATA
# ====================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

print("\n==============================")
print("PROSES PEMBAGIAN DATA")
print("==============================")

print("Jumlah data training :", X_train.shape[0])
print("Jumlah data testing  :", X_test.shape[0])
print("Jumlah fitur TF-IDF  :", X_train.shape[1])
```

Pada gambar di atas merupakan tahap pembagian data menjadi data training dan data testing. Pembagian data ini bertujuan untuk melatih dan menguji model *machine learning*. Data training digunakan untuk melatih model agar dapat mempelajari pola dalam dataset. Sedangkan data testing digunakan untuk menguji kemampuan model dalam memprediksi data baru yang belum pernah dilihat sebelumnya. Pembagian data ini dilakukan menggunakan fungsi *train_test_split()* dari *Library scikit-learn*. Parameter test_size = 0.2 menunjukkan bahwa 20% data digunakan sebagai data testing.

Pada gambar di atas setelah proses pembagian dilakukan, sistem akan menampilkan jumlah data yang masuk ke dalam data training dan data testing. Informasi ini penting untuk memastikan bahwa pembagian data telah dilakukan dengan benar. Selain itu *script* juga menampilkan jumlah fitur yang dihasilkan dari proses *TF-IDF*. Fitur ini merupakan kata-kata penting yang digunakan dalam proses klasifikasi sentimen. Dengan mengetahui jumlah fitur, peneliti dapat memahami

kompleksitas data yang dianalisis. Informasi ini juga berguna untuk proses analisis lebih lanjut. Setelah dilakukan proses split data, kemudian diperoleh hasil split data yaitu sebagai berikut.



Gambar 4. 9. Hasil Split Data (Pembagian Data)

Pada gambar di atas merupakan tahap pembagian data menjadi data training dan data testing. Proses ini dilakukan menggunakan fungsi *train_test_split()* dari *Library scikit-learn*. Dataset dibagi dengan rasio 80% untuk data training dan 20% untuk data testing. Data training digunakan untuk melatih model klasifikasi agar dapat mempelajari pola dari data yang tersedia. Sementara itu data testing digunakan untuk menguji kemampuan model dalam melakukan klasifikasi terhadap data yang belum pernah dilihat sebelumnya. Pembagian data ini bertujuan untuk mengevaluasi kinerja model secara objektif.

Berdasarkan hasil yang diperoleh terlihat bahwa jumlah data training sebanyak 775 data dan data testing sebanyak 194 data. Selain itu jumlah fitur yang digunakan dalam proses klasifikasi adalah sebanyak 1981 fitur kata yang berasal dari hasil pembobotan *TF-IDF*. Data training yang lebih besar memungkinkan

model untuk mempelajari pola data dengan lebih baik. Sementara data testing digunakan sebagai data evaluasi untuk mengukur performa model. Pembagian data ini dilakukan secara acak dengan parameter random_state=42 agar hasil pembagian dapat direproduksi.

Pembagian data merupakan tahap penting dalam proses *machine learning*. Dengan memisahkan data training dan data testing, model dapat diuji secara lebih objektif. Jika seluruh data digunakan untuk training, maka model tidak dapat diuji secara valid. Oleh karena itu penggunaan data testing membantu mengukur kemampuan model dalam memprediksi data baru. Tahap ini menjadi dasar sebelum proses pelatihan model *Naive Bayes* dilakukan.

4.1.10. Training Model *Naive Bayes*

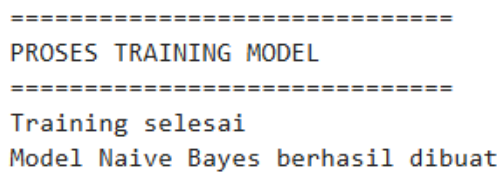
Pada tahap ini dilakukan proses pelatihan model klasifikasi menggunakan metode *Naive Bayes*. *Naive Bayes* merupakan algoritma klasifikasi berbasis probabilitas yang sering digunakan dalam analisis teks dan analisis sentimen. Algoritma ini bekerja dengan menghitung probabilitas kemunculan suatu kata dalam setiap kategori kelas. Kemudian model menentukan kelas dengan probabilitas tertinggi sebagai hasil prediksi. Pada *script* ini digunakan algoritma Multinomial *Naive Bayes* yang cocok untuk data teks dengan fitur berupa frekuensi kata. Proses pelatihan dilakukan menggunakan fungsi model.fit(X_train, y_train).

```
# ====================================
# 10. TRAINING MODEL NAIVE BAYES
# ====================================
print("\n==============================")
print("PROSES TRAINING MODEL")
print("==============================")

model = MultinomialNB()

model.fit(X_train, y_train)

print("Training selesai")
print("Model Naive Bayes berhasil dibuat")
```

Pada gambar di atas merupakan proses training model klasifikasi menggunakan metode *Naive Bayes*. *Naive Bayes* merupakan algoritma klasifikasi berbasis probabilitas yang sering digunakan dalam analisis teks. Algoritma ini bekerja dengan menghitung kemungkinan suatu dokumen termasuk dalam kategori tertentu berdasarkan kata-kata yang muncul di dalamnya. Metode ini disebut "naive" karena mengasumsikan bahwa setiap fitur atau kata bersifat independen satu sama lain. Meskipun asumsi ini sederhana, metode *Naive Bayes* sering memberikan hasil yang cukup baik dalam analisis sentimen. Oleh karena itu metode ini banyak digunakan dalam penelitian *text* mining.

Pada gambar di atas model *Naive Bayes* dibuat menggunakan kelas MultinomialNB() dari *Library scikit-learn*. Model ini sangat cocok digunakan untuk data teks yang direpresentasikan dalam bentuk frekuensi kata atau bobot *TF-IDF*. Proses training dilakukan menggunakan fungsi fit() dengan memasukkan data training dan label sentimen. Pada tahap ini model akan mempelajari hubungan antara kata-kata dalam ulasan dengan kategori sentimen yang diberikan. Model

kemudian menyimpan informasi probabilitas yang akan digunakan untuk proses prediksi. Setelah dilakukan proses training model *Naïve Bayes*, maka diperoleh hasil sebagai berikut.



Gambar 4. 10. Hasil Training Model

Pada gambar di atas merupakan tahap pelatihan model klasifikasi menggunakan algoritma *Naive Bayes*. Algoritma yang digunakan dalam penelitian ini adalah Multinomial *Naive Bayes* yang umum digunakan dalam klasifikasi teks. Model ini bekerja dengan menghitung probabilitas kemunculan kata pada setiap kategori sentimen. Dengan menggunakan data training, model akan mempelajari pola kata yang sering muncul pada sentimen positif maupun negatif. Proses pelatihan dilakukan menggunakan fungsi fit() dari model *Naive Bayes*. Tahap ini bertujuan untuk membangun model klasifikasi yang dapat digunakan dalam proses prediksi.

Berdasarkan hasil yang ditampilkan terlihat bahwa proses training telah berhasil dilakukan dengan pesan "Training selesai" dan "Model *Naive Bayes* berhasil dibuat". Hal ini menunjukkan bahwa model telah berhasil mempelajari pola dari data training yang digunakan. Proses pelatihan ini menggunakan data training sebanyak 775 data ulasan. Model kemudian menyimpan informasi mengenai probabilitas kata yang berkaitan dengan masing-masing kategori sentimen. Informasi ini akan digunakan pada tahap prediksi data testing.

Pelatihan model merupakan tahap inti dalam proses klasifikasi. Pada tahap ini model belajar mengenali hubungan antara kata dalam ulasan dengan kategori sentimen yang sesuai. Semakin baik kualitas data training yang digunakan, maka semakin baik pula model dalam mempelajari pola sentimen. Model yang telah dilatih kemudian siap digunakan untuk melakukan prediksi terhadap data testing. Tahap berikutnya adalah proses prediksi menggunakan model yang telah dibangun.

4.1.11. Prediksi

Setelah model berhasil dilatih, tahap berikutnya adalah melakukan prediksi terhadap data testing. Proses ini dilakukan menggunakan fungsi model.predict(X_test). Fungsi tersebut akan menghasilkan prediksi sentimen berdasarkan pola yang telah dipelajari pada data training. Hasil prediksi ini disimpan dalam variabel y_pred. Prediksi tersebut berisi label sentimen yang diperkirakan oleh model untuk setiap data testing. Dengan cara ini model dapat digunakan untuk mengklasifikasikan ulasan pengguna secara otomatis.

```
# ====================================
# 11. PREDIKSI
# ====================================
print("\n==============================")
print("PROSES PREDIKSI DATA TEST")
print("==============================")

y_pred = model.predict(X_test)

print("Contoh hasil prediksi:")
print(y_pred[:10])
```

Pada gambar di atas merupakan tahap prediksi sentimen terhadap data testing menggunakan model yang telah dilatih. Proses prediksi dilakukan menggunakan fungsi predict() dari model *Naive Bayes*. Fungsi ini akan menghasilkan prediksi

label sentimen untuk setiap data yang terdapat dalam data testing. Hasil prediksi ini kemudian disimpan dalam variabel y_pred. Variabel ini berisi label sentimen yang diperkirakan oleh model berdasarkan pola yang telah dipelajari sebelumnya. Dengan cara ini model dapat digunakan untuk mengklasifikasikan ulasan pengguna secara otomatis.

Pada gambar di atas *script* juga menampilkan beberapa contoh hasil prediksi untuk melihat bagaimana model bekerja. Contoh hasil prediksi biasanya ditampilkan dalam bentuk beberapa label sentimen pertama dari data testing. Hal ini membantu peneliti memahami hasil klasifikasi yang dihasilkan oleh model. Jika model bekerja dengan baik, maka hasil prediksi akan mendekati label sentimen yang sebenarnya. Proses prediksi ini merupakan langkah penting sebelum melakukan evaluasi model. Setelah dilakukan proses prediksi, maka diperoleh hasil nya sebagai berikut.

```
==============================
PROSES PREDIKSI DATA TEST
==============================
Contoh hasil prediksi:
['positif' 'positif' 'positif' 'positif' 'positif' 'positif' 'positif'
 'positif' 'positif' 'positif']
```

Gambar 4. 11. Hasil Prediksi Data

Pada gambar di atas merupakan tahap prediksi sentimen pada data testing menggunakan model *Naive Bayes* yang telah dilatih sebelumnya. Proses prediksi dilakukan menggunakan fungsi predict() dari model yang telah dibuat. Data testing yang sebelumnya dipisahkan dari dataset digunakan sebagai input dalam proses prediksi ini. Tujuan dari tahap ini adalah untuk mengetahui bagaimana model

mengklasifikasikan data baru yang belum pernah dilihat sebelumnya. Hasil prediksi berupa label sentimen seperti positif atau negatif. Tahap ini menjadi langkah awal sebelum proses evaluasi model dilakukan.

Berdasarkan hasil yang ditampilkan terlihat bahwa sebagian besar data testing diprediksi sebagai sentimen positif. Hal ini dapat dilihat dari contoh hasil prediksi yang ditampilkan dimana beberapa data pertama diklasifikasikan sebagai positif. Hasil ini menunjukkan bahwa model telah berhasil melakukan proses klasifikasi terhadap data testing. Prediksi ini didasarkan pada pola kata yang telah dipelajari oleh model selama proses training. Dengan demikian model dapat menentukan kategori sentimen dari setiap ulasan pengguna.

Proses prediksi merupakan tahap penting untuk melihat kemampuan model dalam mengklasifikasikan data baru. Jika model mampu menghasilkan prediksi yang mendekati label sebenarnya, maka model tersebut dapat dikatakan memiliki performa yang baik. Hasil prediksi ini kemudian akan dibandingkan dengan label sebenarnya pada data testing. Perbandingan tersebut digunakan dalam proses evaluasi model. Tahap evaluasi ini bertujuan untuk mengukur tingkat akurasi dan performa model secara keseluruhan.

4.1.12. Evaluasi Model

Tahap terakhir dalam *script* ini adalah evaluasi model klasifikasi. Evaluasi dilakukan untuk mengukur seberapa baik model dalam memprediksi sentimen ulasan pengguna. Pada *script* ini digunakan beberapa metrik evaluasi yaitu akurasi, precision, recall, dan F1-score. Akurasi menunjukkan persentase prediksi yang benar dibandingkan dengan seluruh data yang diuji. Precision menunjukkan

seberapa banyak prediksi positif yang benar-benar positif. Recall menunjukkan kemampuan model dalam menemukan semua data positif yang sebenarnya. Sedangkan F1-score merupakan kombinasi dari precision dan recall.

```
# ====================================
# 12. EVALUASI MODEL
# ====================================
print("\n==============================")
print("EVALUASI MODEL")
print("==============================")

print("\nAKURASI MODEL")
print(accuracy_score(y_test, y_pred))

print("\nCLASSIFICATION REPORT")
print(classification_report(y_test, y_pred))
```

Pada gambar di atas merupakan tahap prediksi sentimen terhadap data testing menggunakan model yang telah dilatih. Proses prediksi dilakukan menggunakan fungsi predict() dari model *Naive Bayes*. Fungsi ini akan menghasilkan prediksi label sentimen untuk setiap data yang terdapat dalam data testing. Hasil prediksi ini kemudian disimpan dalam variabel y_pred. Variabel ini berisi label sentimen yang diperkirakan oleh model berdasarkan pola yang telah dipelajari sebelumnya. Dengan cara ini model dapat digunakan untuk mengklasifikasikan ulasan pengguna secara otomatis. Setelah dilakukan evaluasi model, maka diperoleh hasil sebagai berikut.

```
==============================
EVALUASI MODEL
==============================

AKURASI MODEL
0.8402061855670103

CLASSIFICATION REPORT
              precision    recall  f1-score   support

     negatif       0.88      0.52      0.65        56
     positif       0.83      0.97      0.90       138

    accuracy                           0.84       194
   macro avg       0.86      0.74      0.77       194
weighted avg       0.85      0.84      0.83       194
```

Gambar 4. 12. Hasil Evaluasi Model

Pada gambar di atas merupakan tahap evaluasi model klasifikasi untuk mengetahui kinerja model *Naive Bayes* yang telah dibuat. Evaluasi dilakukan menggunakan beberapa metrik yaitu accuracy, precision, recall, dan f1-score. Metrik tersebut dihitung menggunakan fungsi accuracy_score() dan classification_report() dari *Library scikit-learn*. Accuracy digunakan untuk mengukur tingkat ketepatan model dalam melakukan klasifikasi. Sementara itu precision, recall, dan f1-score digunakan untuk mengevaluasi performa model pada masing-masing kelas sentimen. Tahap ini bertujuan untuk mengetahui seberapa baik model dalam mengklasifikasikan data ulasan pengguna.

Berdasarkan hasil evaluasi yang diperoleh terlihat bahwa model memiliki nilai akurasi sebesar 0.84 atau 84%. Hal ini menunjukkan bahwa model mampu mengklasifikasikan sekitar 84% data testing dengan benar. Pada kelas negatif diperoleh nilai precision 0.88, recall 0.52, dan f1-score 0.65. Sedangkan pada kelas

positif diperoleh nilai precision 0.83, recall 0.97, dan f1-score 0.90. Hasil ini menunjukkan bahwa model memiliki performa yang sangat baik dalam mengenali sentimen positif. Namun kemampuan model dalam mengenali sentimen negatif masih lebih rendah dibandingkan sentimen positif.

Secara keseluruhan hasil evaluasi menunjukkan bahwa metode *Naive Bayes* mampu melakukan klasifikasi sentimen terhadap ulasan pengguna aplikasi *Shopee* dengan cukup baik. Nilai akurasi yang mencapai 84% menunjukkan bahwa model memiliki tingkat keakuratan yang tinggi dalam mengklasifikasikan data ulasan. Meskipun demikian terdapat perbedaan performa antara kelas positif dan negatif yang dapat disebabkan oleh ketidakseimbangan jumlah data pada masing-masing kelas. Hal ini menunjukkan bahwa model cenderung lebih mudah mengenali sentimen positif dibandingkan negatif. Namun secara umum model yang dihasilkan sudah cukup baik untuk digunakan dalam analisis sentimen terhadap ulasan pengguna aplikasi *Shopee*.