

**LAPORAN FINAL PROJEK**  
**“SISTEM IDENTIFIKASI SENTIMEN DATA TEKS MENGGUNAKAN**  
**MULTINOMIAL NAIVE BAYES”**  
**PENGANTAR DAN PEMROSESAN DATA MULTIMEDIA**



**Disusun oleh:**

|                              |            |
|------------------------------|------------|
| Ni Made Julia Budiantari     | 2108561004 |
| Kameliya Putri               | 2108561019 |
| I Gede Rizki Heriana Prayoga | 2108561014 |

**Dosen Pengampu:**

Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si., M.Eng

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**UNIVERSITAS UDAYANA**  
**BADUNG**

## DAFTAR ISI

|   |    |
|---|----|
| <b>DAFTAR ISI</b> .....                       | ii |
| <b>BAB I PENDAHULUAN</b> .....                | 1  |
| 1.1 Latar belakang.....                       | 1  |
| 1.2 Problem Komputasi.....                    | 2  |
| 1.3 Tujuan .....                              | 3  |
| 1.4 Manfaat .....                             | 3  |
| <b>BAB II ISI</b> .....                       | 4  |
| 2.1 Manual Aplikasi.....                      | 4  |
| 2.1.1 Fitur Sistem .....                      | 4  |
| 2.1.2 Antar Muka Aplikasi.....                | 9  |
| 2.2 Source Code Modul .....                   | 11 |
| 2.2.1 Lib.py .....                            | 11 |
| 2.2.2 Main.py .....                           | 11 |
| 2.2.3 Model.py .....                          | 12 |
| 2.2.4 Preprocecing.py.....                    | 13 |
| 2.2.5 Preprocessing.ipynb.....                | 13 |
| 2.2.6 MNB modeling use 10 percent.ipynb ..... | 15 |
| <b>BAB III PENUTUP</b> .....                  | 18 |
| 3.1 Kesimpulan .....                          | 18 |
| 3.2 Saran.....                                | 19 |
| <b>DAFTAR PUSTAKA</b> .....                   | 20 |

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar belakang**

Dalam era digital yang semakin maju, banyak informasi yang tersedia dalam bentuk teks di berbagai platform seperti media sosial, forum online, blog, dan artikel berita. Data teks ini mengandung berbagai sentimen atau perasaan pengguna seperti positif, negatif, atau netral. Oleh karena itu, penting untuk mengembangkan sistem yang dapat mengidentifikasi sentimen dari data teks secara otomatis. Metode klasifikasi yang banyak digunakan untuk identifikasi sentimen adalah Naive Bayes. Naive Bayes adalah sebuah metode yang didasarkan pada teorema Bayes dan mengasumsikan independensi kondisional antara fitur-fitur data[1].

Salah satu variasi dari Naive Bayes adalah Multinomial Naive Bayes, yang dirancang khusus untuk mengklasifikasikan data dengan fitur-fitur diskrit seperti data teks. Multinomial Naive Bayes telah terbukti efektif dalam banyak aplikasi pengklasifikasian teks, termasuk analisis sentimen [1]. Metode ini mampu menangani masalah skala besar dan menghasilkan hasil yang cepat dan akurat [2]. Kelebihan lain dari Multinomial Naive Bayes adalah kemampuannya dalam menghadapi ketidakseimbangan kelas, di mana jumlah data yang dikategorikan sebagai sentimen positif, negatif, atau netral dapat berbeda-beda[3]. Salah satu aspek penting dalam pengklasifikasian teks adalah pembobotan Term Frequency (TF). Pembobotan TF digunakan untuk menghitung frekuensi kemunculan kata-kata dalam sebuah dokumen teks, yang kemudian metode pembobotan ini memberikan nilai yang relatif untuk setiap kata berdasarkan seberapa sering kata tersebut muncul dalam teks[4].

Dalam laporan ini, kami akan menyajikan pengembangan dan evaluasi sistem identifikasi sentimen data teks menggunakan metode Multinomial Naive Bayes. Laporan ini akan menjelaskan langkah-langkah yang kami ambil dalam mempersiapkan data, melakukan pemrosesan teks, dan melatih model klasifikasi. Kami juga akan membahas pengujian dan evaluasi model yang kami lakukan

menggunakan metrik yang relevan seperti akurasi, presisi, recall, dan F1-score. Dengan menggunakan sistem identifikasi sentimen data teks yang handal, berbagai industri dan organisasi dapat memanfaatkannya untuk memahami sentimen pengguna terhadap produk atau layanan mereka. Misalnya, perusahaan dapat menggunakan sistem ini untuk memantau umpan balik pelanggan, memperbaiki aspek yang negatif, dan meningkatkan pengalaman pelanggan secara keseluruhan. Dengan laporan ini, kami berharap dapat memberikan kontribusi dalam pengembangan dan pemahaman lebih lanjut tentang metode identifikasi sentimen menggunakan Multinomial Naive Bayes.

## **1.2 Problem Komputasi**

Membangun sistem aplikasi untuk mengidentifikasi sentimen dari sebuah/beberapa ulasan. Terdapat dua sentimen yaitu sentimen positif dan sentimen negatif. Dataset merupakan data teks ulasan. Semua data dibagi menjadi 2 bagian: 80% untuk dataset training dan 20% untuk dataset testing. Data text ulasan terbagi menjadi 2 label sentimen positif (1) dan negatif (0). Adapun beberapa tahapan utama proses komputasi untuk menghasilkan model klasifikasi antara lain:

### **A. Tahapan preprocessing untuk data teks:**

- 1) Feature Extraction untuk memperoleh nilai-nilai bobot TF atau TF IDF dari data ulasan. Tahapan feature extraction: tokenization, lower case converting, stop word removal, stemming, dan pembobotan (TF atau TFIDF). Untuk metode Multinomial Naive Bayes pembobotan kata menggunakan formula TF, sedangkan untuk metode yang lain menggunakan TF IDF. Total jumlah fitur dari setiap data text tergantung dari panjang/ jumlah vocabulary setelah preprocessing (jumlah term indeks) dari koleksi ulasan yang digunakan.
- 2) Feature Selection dengan menggunakan formula Chi-Square (lihat penjelasan di URL terkait), gunakan beberapa variasi jumlah fitur yang dipertahankan (10% atau 20% atau 30%, dsb).

**B. Tahapan Training:**

Training dilaksanakan untuk menghasilkan model klasifikasi yang terbaik.

Metode Naive Bayes yang digunakan adalah Multinomial Naive Bayes.

**C. Tahapan Testing:**

Ukuran evaluasi yang digunakan adalah akurasi, precision, recall, an F1-Score.

**D. Tahapan Deployment:**

Model yang dihasilkan di deploy ke sistem aplikasi berbasis web dengan fitur utama adalah user menginput satu atau beberapa data dan outputnya adalah hasil sentimen atau identifikasi emosi.

### **1.3 Tujuan**

Adapun tujuan dari laporan yang dibuat ini adalah

- a. Mengevaluasi dan mengukur performa model Multinomial Naive Bayes dalam mengidentifikasi sentimen data teks.
- b. Menyajikan interpretasi hasil dan temuan dari sistem identifikasi sentimen data teks menggunakan Multinomial Naive Bayes.

### **1.4 Manfaat**

Manfaat dari laporan ini adalah dapat meningkatkan pemahaman tentang penggunaan Multinomial Naive Bayes dalam identifikasi sentimen data teks. Dengan mempelajari metode ini, pembaca akan mendapatkan wawasan tentang algoritma klasifikasi yang efektif untuk memproses data teks dan mengenali sentimen yang terkandung di dalamnya.

## BAB II

### ISI

## 2.1 Manual Aplikasi

### 2.1.1 Fitur Sistem

#### 2.1.1.1 Tampilan program

Fitur-fitur yang disediakan dari aplikasi pada program adalah sebagai berikut:

#### 1. Melakukan *preprocessing* teks

Fitur ini berfungsi untuk melakukan *preprocessing* terhadap teks *review*. Dalam melakukan teks *preprocessing* melibatkan beberapa tahapan yang dimana dimulai dari tahap tokenisasi. Dalam tahap tokenisasi juga dilakukan proses *cleaning* yang merupakan tahap pembersihan teks dari karakter-karakter yang bukan alphabet seperti tanda baca, emoticon, link url dan lain sebagainya. Hasil dari *cleaning* maka teks ulasan tersebut dijadikan token-token atau dipisahkan menjadi per kata. Selanjutnya dilakukan *stopword removal* untuk menghapus kata-kata yang tidak penting dalam dokumen seperti kata “yang, ke, dan, di” dan lain sebagainya. Dalam mengubah kata gaul menjadi kata baku digunakanlah proses normalisasi dan terakhir yaitu tahap *stemming* yaitu mengubah kata menjadi kata dasarnya. Berikut merupakan gambar output dari fitur hasil *preprocessing*.

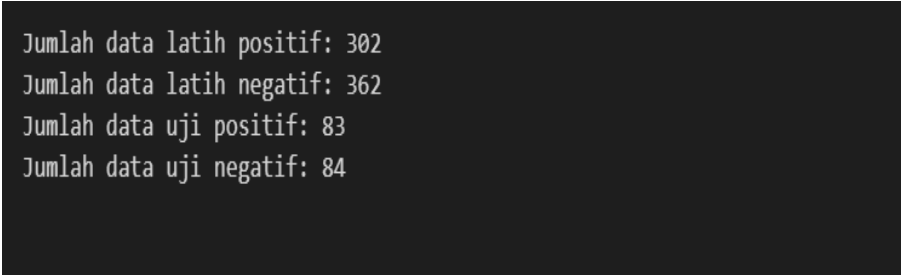
| Label | Reviews | Reviews tokens                                    | Reviews Stopword  | reviews normalized                                 | stemming   |
|-------|---------|---|---|--|--|
| 0     | 1       | kemeja nya bagus bgt a mau ngis knpa ga dri dl... | [kemeja, nya, bagus, bgt, a, mau, ngis, knpa, ...]        | [kemeja, bagus, banget, a, ngis, knpa, tidak, ...] | [kemeja, bagus, banget, a, ngis, knpa, tidak, ...] |
| 1     | 0       | jahitnya sih rapi cuman ada benang yang ikut ...  | [jahitnya, sih, rapi, cuman, ada, benang, yan...]         | [jahitnya, rapi, cuman, benang, jahit, jelek]      | [jahitnya, rapi, cuman, benang, jahit, jelek]      |
| 2     | 0       | sesuai harga agak tipis tapi masih oke kok war... | [sesuai, harga, agak, tipis, tapi, warnanya, abu, kal...] | [sesuai, harga, tipis, oke, warnanya, abu, kal...] | [sesuai, harga, tipis, oke, warna, abu, kalau...]  |
| 3     | 1       | wah gila sih sebgus itu se worth it se lembut...  | [wah, gila, sih, sebgus, itu, se, worth, it, ...]         | [gila, sebgus, worth, it, lembut, bajunya, ki...]  | [gila, bagus, worth, it, lembut, baju, kirain...]  |
| 4     | 0       | kain nya bagus halus tapi kok di bukak kotor y... | [kain, nya, bagus, halus, tapi, kok, di, bukak...]        | [kain, bagus, halus, bukak, kotor, ya, warna, ...] | [kain, bagus, halus, buka, kotor, ya, warna, p...] |
| ...   | ...     | ...   | ...   | ...  | ...  |
| 826   | 1       | terima kasih barang sudah sampai sesuai ukuran... | [terima, kasih, barang, sudah, sampai, sesuai...]         | [terima, kasih, barang, sesuai, ukuran, sesuai...] | [terima, kasih, barang, sesuai, ukur, sesuai, ...] |
| 827   | 1       | mantap realpict bangt tapi pengemasan nya cuma... | [mantap, realpict, bangt, tapi, pengemasan, ny...]        | [mantap, realpict, bangt, pengemasan, cuman, p...] | [mantap, realpict, bangt, emas, cuman, plastik...] |
| 828   | 1       | suka bgt sama tasnya ga kayak tas local keren ... | [suka, bgt, sama, tasnya, ga, kayak, tas, loca...]        | [suka, banget, tasnya, tidak, seperti, tas, lo...] | [suka, banget, tas, tidak, seperti, tas, local...] |
| 829   | 1       | kualitas produk sangat baik produk original ha... | [kualitas, produk, sangat, baik, produk, origi...]        | [kualitas, produk, produk, original, harga, pr...] | [kualitas, produk, produk, original, harga, pr...] |
| 830   | 1       | barang udah sampai dg selamat mantul banget da... | [barang, udah, sampai, dg, selamat, mantul, bange...]     | [barang, sudah, dengan, selamat, mantul, bange...] | [barang, sudah, dengan, selamat, mantul, bange...] |

831 rows x 6 columns

Gambar 1. Output fitur *preprocessing*

## 2. Memisahkan data latih dan data uji

Fitur ini berfungsi untuk membagi data uji dan data latih sebelum data di *training*. Dalam membagi data ini dilakukan pembagian yang seimbang antara data positif dan negatif untuk masing-masing data uji dan data latih untuk mendapatkan akurasi yang baik. Berikut merupakan gambar output dari fitur pembagian data.



```
Jumlah data latih positif: 302  
Jumlah data latih negatif: 362  
Jumlah data uji positif: 83  
Jumlah data uji negatif: 84
```

Gambar 2. Output fitur pembagian data latih dan data uji

## 3. Melakukan ekstraksi fitur TF (*Term Frequency*)

Fitur ini berfungsi untuk melakukan ekstraksi fitur TF (*Term Frequency*) yang merupakan metode pembobotan yang digunakan untuk mengukur pentingnya suatu kata dalam sebuah dokumen atau korpus. Metode ini menggunakan term atau kata-kata yang telah melalui proses preprocessing sebagai inputnya. Dengan menggunakan TF-IDF, kita dapat menentukan bobot atau nilai penting dari setiap kata dalam dokumen berdasarkan frekuensi kemunculannya dalam dokumen tersebut dan dalam keseluruhan korpus. Ekstraksi fitur digunakan untuk mengungkapkan informasi yang berpotensi dan mewakili kata-kata sebagai vektor fitur. Vektor ini kemudian digunakan sebagai input untuk metode klasifikasi pada tahap selanjutnya. Berikut merupakan gambar output dari fitur ekstraksi fitur TF (*Term Frequency*).

```

TF vector pada train set:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Normalized TF vector pada train set:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

```

Gambar 3. Output fitur TF (*Term Frequency*)

#### 4. Melakukan seleksi fitur dengan *Chi-Square*

Fitur ini digunakan untuk melakukan seleksi fitur yang merupakan proses memilih subset fitur yang paling relevan atau informatif dari suatu dataset untuk digunakan dalam pembuatan model atau analisis data. Dalam program ini digunakan *Chi-Square* sebagai seleksi fitur. Seleksi fitur *Chi-square* adalah salah satu metode seleksi fitur yang umum digunakan dalam analisis data untuk mengidentifikasi fitur-fitur yang paling relevan dalam klasifikasi atau analisis kategorikal. Metode ini didasarkan pada uji statistik *Chi-square* yang digunakan untuk mengukur hubungan antara dua variabel kategorikal. Dalam seleksi fitur ini dapat menggunakan beberapa presentase yang dapat dipertahankan seperti 10%, 20%, 30% dan lain sebagainya. Berikut merupakan gambar output dari fitur seleksi fitur *Chi-Square*.

```

Hasil seleksi fitur dengan chi-square pada train set:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
Hasil seleksi fitur dengan chi-square pada test set:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

```

Gambar 4. Output fitur *Chi-Square*



##### 5. Melakukan klasifikasi dengan MNB

Fitur ini digunakan untuk melakukan klasifikasi teks ulasan ke dalam dua kelas yaitu positif negatif. Dalam program ini menggunakan algoritma *Multinomial Naive Bayes*. Multinomial Naive Bayes adalah algoritma klasifikasi yang berbasis pada teorema Bayes dengan asumsi bahwa fitur-fitur yang digunakan adalah variabel acak dengan distribusi multinomial. Berikut merupakan gambar output dari fitur klasifikasi dengan algoritma MNB.

```
Jumlah data uji positif : 83
Jumlah data uji negatif : 84
Jumlah prediksi benar   : 155
Jumlah prediksi salah   : 12
Jumlah prediksi positif : 85
Jumlah prediksi negatif : 82
Akurasi: 0.9281437125748503
Precision: 0.9176470588235294
Recall: 0.9397590361445783
F1-score: 0.9285714285714286
```

Gambar 5. Output fitur klasifikasi dengan MNB

##### 6. Melakukan perhitungan dengan tabel *confusion matrix*

Fitur ini digunakan untuk melakukan perhitungan tabel *confusion matrix*. Confusion matrix adalah sebuah tabel yang umumnya digunakan untuk memvisualisasikan kinerja model klasifikasi pada sejumlah data uji di mana nilai sebenarnya sudah diketahui. Berikut gambar output tabel *confusion matrix*. Dalam gambar dibawah ini diketahui *true* positif berjumlah 77, *false* positif sebanyak 7, *true* negatif sebanyak 78 dan *false* negatif sebanyak 5.

```
Tabel Cofusion Matrix:
[[77  7]
 [ 5 78]]
```

Gambar 6. Output fitur tabel *confusion matrix*

## 7. Melakukan hasil evaluasi

Fitur ini digunakan untuk melakukan evaluasi terhadap model yang dibangun. Evaluasi dilakukan untuk mengukur performa dari model yang telah dibangun. Performa ini dapat diukur menggunakan *tabel confusion matrix*. Berikut merupakan gambar output fitur dari hasil evaluasi. Dalam gambar dibawah diketahui hasil akurasi yang dihasilkan sebesar 93% yang dimana untuk akurasi masing-masing kelas positif dan negatif sebesar 93%.

| Classification Report: |           |        |          |         |
|------------------------|-----------|--------|----------|---------|
|                        | precision | recall | f1-score | support |
| negatif                | 0.94      | 0.92   | 0.93     | 84      |
| positif                | 0.92      | 0.94   | 0.93     | 83      |
| accuracy               |           |        | 0.93     | 167     |
| macro avg              | 0.93      | 0.93   | 0.93     | 167     |
| weighted avg           | 0.93      | 0.93   | 0.93     | 167     |

Gambar 7. Output fitur hasil evaluasi

### 2.1.1.2 Hasil program dan pembahasan

Dalam program klasifikasi yang menggunakan Multinomial Naive Bayes dengan tuning hyper-parameter pada variasi fitur yang digunakan dalam seleksi fitur Chi-Square, terdapat beberapa hasil yang menarik. Pertama, dilakukan variasi fitur Chi-Square dengan persentase yang berbeda, yaitu 10%, 20%, 30%, 40%, dan 50%. Kemudian, performa model dikuantifikasi menggunakan metrik akurasi. Hasil yang diperoleh adalah sebagai berikut:

Tabel 1. *Tuning hyper-parameter*

| Variasi Fitur | 10%    | 20%    | 30%    | 40%    | 50%    |
|---------------|--------|--------|--------|--------|--------|
| Akurasi       | 91,61% | 92,81% | 92,81% | 92,81% | 92,81% |

Dengan menggunakan variasi fitur 10%, model menghasilkan akurasi sebesar 91,61%. Sedangkan dengan menggunakan variasi fitur 20%, 30%, 40%, dan 50%, model menghasilkan akurasi sebesar 92,81%.

Dari hasil tersebut, dapat disimpulkan bahwa model dengan variasi fitur sebesar 30% memiliki kinerja terbaik dengan akurasi 92,81%. Oleh karena itu, model ini dipilih sebagai "best model" untuk sistem aplikasi ini. Seleksi fitur Chi-Square memainkan peran penting dalam meningkatkan kinerja model klasifikasi. Dengan memilih fitur-fitur yang paling informatif atau relevan, model dapat fokus pada informasi yang penting dan mengurangi dimensi data. Hal ini dapat mengurangi overfitting, meningkatkan akurasi, dan meningkatkan pemahaman terhadap dataset. Dalam konteks ini, dengan menggunakan seleksi fitur Chi-Square, variasi fitur sebesar 30% memberikan keseimbangan yang baik antara jumlah fitur yang digunakan dan kinerja model. Meskipun variasi fitur lainnya memiliki akurasi yang sedikit lebih tinggi, mungkin menghasilkan model yang lebih kompleks dan memerlukan sumber daya komputasi yang lebih besar. Hal ini menunjukkan bahwa penggunaan seleksi fitur Chi-Square dengan persentase yang tepat dapat meningkatkan kinerja model klasifikasi.

### 2.1.2 Antar Muka Aplikasi

Berikut adalah tampilan antarmuka pengguna beserta penjelasannya.



Gambar 8. Tampilan awal program.

Gambar ini menampilkan tampilan awal dari suatu program. Tampilan ini biasanya muncul setelah pengguna berhasil membuka atau mengakses program tersebut. Tampilan awal ini dirancang untuk memberikan pengguna informasi dan kontrol dasar yang diperlukan untuk memulai menggunakan program dengan

efektif. Pada tampilan ini terdapat field untuk menginput *text* yang ingin diketahui analisis sentimennya dan bottom prediksi kalimat untuk mengecek kalimatnya.



Gambar 9. Tampilan awal program.

Gambar ini menampilkan tampilan awal dari suatu program. Tampilan ini biasanya muncul setelah pengguna berhasil membuka atau mengakses program tersebut. Tampilan awal ini dirancang untuk memberikan pengguna informasi dan kontrol dasar yang diperlukan untuk memulai menggunakan program dengan efektif. Pada tampilan ini terdapat field untuk menginput *text* yang ingin diketahui analisis sentimennya dan bottom prediksi kalimat untuk mengecek kalimatnya.



Gambar 10. Tampilan program dengan output ulasan negatif.

Gambar ini menampilkan review yang menghasilkan sentimen negatif dengan inputan “udah jelek, kain kasar, isi robek lg” yang menandakan ulasan negatif.

## 2.2 Source Code Modul

Berikut adalah penjelasan setiap modul kodingan yang digunakan.

### 2.2.1 Lib.py

| Potongan Code   | Penjelasan  |
|---|---|
| <pre>import streamlit as st import pandas as pd import string import re  import nltk from nltk.tokenize import word_tokenize from nltk.probability import FreqDist from nltk.corpus import stopwords  from Sastrawi.Stemmer.StemmerFactory import StemmerFactory import swifter  import ast  from sklearn.feature_extraction.text import CountVectorizer from sklearn.preprocessing import normalize from sklearn.model_selection import train_test_split</pre> | <p>Mengimport Library. Langkah awal sebelum mulai membuat program adalah mengimport library yang akan diperlukan. Beberapa library yang digunakan adalah streamlit, pandas (pd), string, re, nltk, nltk.tokenize, nltk.probability, Sastrawi.Stemmer.StemmerFactory, swifter, ast, sklearn.feature_extraction.text.CountVectorizer, sklearn.preprocessing, sklearn.model_selection.train_test_split, sklearn.feature_selection.SelectPercentile, sklearn.feature_selection.chi2, sklearn.naive_bayes.MultinomialNB, joblib, pickle.</p> |

### 2.2.2 Main.py

|  |  |
|--|--|
| <pre>from lib import * from model import * from preprocessing import *  def main():     st.title("Final Projek PPDM")     st.title(         ":green[Review Ulasan(Sentiment Analysis Expression)]:blue[dengan Python]")     st.header('Menerapkan Metode Multinomial Naive Bayes And Chi - Square')     teks = st.text_input('Input text')     hasil = preprocess_text(teks)     kalimat_normalisasi = normalized_term(hasil)     def convert_to_sentence(word_list):         sentence = ' '.join(word_list)         return sentence     kalimat = convert_to_sentence(kalimat_normalisasi)     if st.button('Prediksi Kalimat'):         prediction = predict_text(kalimat)  if __name__ == '__main__':    main()</pre> | <p>Alur Eksekusi Program Utama. Fungsi main() digunakan untuk mengatur tampilan judul aplikasi Streamlit dan sebagai tempat untuk menuliskan kode program utama. Dapat juga menambahkan kode program lainnya di dalam fungsi main(), seperti memuat data, melakukan pemrosesan teks, membangun model, dan menampilkan hasilnya menggunakan Streamlit</p> |
|--|--|

### 2.2.3 Model.py

```
from lib import *

def join_text_list(texts):
    texts = ast.literal_eval(texts)
    return ' '.join([text for text in texts])

def predict_text(input_text):
    reviews_data = pd.read_excel(
        "Hasil_Preprocessing.xlsx", usecols=["Label", "stemming"])
    reviews_data.columns = ["label", "reviews"]
    reviews_data["reviews_join"] = reviews_data["reviews"].apply(
        join_text_list)

    label = reviews_data["label"]
    text = reviews_data["reviews_join"]

    train_data, test_data, train_labels, test_labels = train_test_split(
        text, label, test_size=0.2, random_state=42)

    positive_count = (train_labels == 1).sum()
    negative_count = (train_labels == 0).sum()
    total_count = len(train_labels)
    positive_ratio = positive_count / total_count
    negative_ratio = negative_count / total_count

    cvect = CountVectorizer()
    TF_vector_train = cvect.fit_transform(train_data)

    # Normalisasi TF vector pada train set
    normalized_TF_vector_train = normalize(TF_vector_train, norm='l1', axis=1)

    # Perhitungan TF vector pada test set menggunakan CountVectorizer yang
    sudah dilatih pada train set
    TF_vector_test = cvect.transform(test_data)

    # Normalisasi TF vector pada test set
    normalized_TF_vector_test = normalize(TF_vector_test, norm='l1', axis=1)

    # Persentase fitur yang ingin dipilih setelah seleksi (10%)
    percent = 30

    # Menghitung jumlah fitur yang diinginkan berdasarkan persentase
    k = int(percent / 100 * normalized_TF_vector_train.shape[1])

    # Menerapkan seleksi fitur dengan chi-square pada train set
    selector = SelectPercentile(chi2, percentile=percent)
    tf_mat_train_selected = selector.fit_transform(
        normalized_TF_vector_train, train_labels)

    # Mengaplikasikan seleksi fitur yang sama pada test set
    tf_mat_test_selected = selector.transform(normalized_TF_vector_test)

    input_vector = cvect.transform([input_text])

    # Normalisasi vektor fitur
    normalized_input_vector = normalize(input_vector, norm='l1', axis=1)

    # Terapkan seleksi fitur pada vektor fitur input
    input_vector_selected = selector.transform(normalized_input_vector)

    model = joblib.load('multinomial_nb_30_percent_model.pkl')
```

Model yang digunakan dalam program adalah model Naive Bayes yang telah dilatih sebelumnya dan disimpan sebagai file

"multinomial\_nb\_30

percent\_model.pkl"

menggunakan joblib.dump().

Selain itu menggambarkan juga sebuah fungsi bernama

predict\_text() yang menerima input teks dan melakukan prediksi klasifikasi teks tersebut

menggunakan model yang telah dilatih sebelumnya

```
# Lakukan prediksi menggunakan model yang telah Anda bangun
prediction = model.predict(input_vector_selected)

# Cetak output klasifikasi
if prediction == 0:
    st.write('Ulasannya negatif:sob:')
else:
    st.write('Ulasannya positif dong:smiley:')
```

## 2.2.4 Preprocecing.py

```
from lib import*

def preprocess_text(text):
    text= text.lower()
    # remove tab, new line, and backslash
    text = text.replace('\t', ' ').replace('\n', ' ').replace('\ ', ' ')
    # remove non ASCII (emoticon, Chinese word, etc)
    text = text.encode('ascii', 'replace').decode('ascii')
    # remove mention, link, hashtag
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\w+\/\S+)", " ",
text).split())
    # remove incomplete URL
    text = text.replace("http://", " ").replace("https://", " ")
    # remove numbers
    text = re.sub(r"\d+", "", text)
    # remove punctuation
    text = text.translate(str.maketrans("", "", string.punctuation))
    # remove leading and trailing whitespace
    text = text.strip()
    # remove multiple whitespace into single whitespace
    text = re.sub('\s+', ' ', text)
    # remove single character
    text = re.sub(r"\b[a-zA-Z]\b", "", text)
    # tokenize words
    tokens = word_tokenize(text.lower())
    # remove stopwords
    stopword_list = set(stopwords.words('indonesian'))
    tokens = [word for word in tokens if word not in stopword_list]
    return tokens

normalized_word = pd.read_excel("normalisasi.xlsx")
normalized_word_dict = {}
for index, row in normalized_word.iterrows():
    if row[0] not in normalized_word_dict:
        normalized_word_dict[row[0]] = row[1]

def normalized_term(document):
    return [normalized_word_dict[term] if term in normalized_word_dict else
term for term in document]
```

Gambar ini menjelaskan mengenai fungsi `preprocess_text(text)` yang bertujuan untuk melakukan preprocessing teks, dan juga pembuatan kamus normalisasi kata untuk digunakan dalam tahap preprocessing data teks yang lebih lanjut

## 2.2.5 Preprocessing.ipynb

```
import pandas as pd
import string
import re #regex library

from nltk.tokenize import word_tokenize
```

Library pandas (pd) terlebih dahulu akan membaca data.xlsx untuk kolom 'label' dan 'reviews' yang akan digunakan untuk tahap

|  |  |
|--|--|
| <pre> from nltk.probability import FreqDist from nltk.corpus import stopwords from Sastrawi.Stemmer.StemmerFactory import StemmerFactory import swifter  data = pd.read_excel("reviews.xlsx") selected_columns = ['Label', 'Reviews'] selected_data = data[selected_columns] data = selected_data data </pre>  | <p>preprocessing selanjutnya.</p>  |
| <pre> def remove(text):     text = re.sub(r'[0-9]', ' ', text)     text = re.sub(r'^\w\s', ' ', text)     text = re.sub(r'^A-Za-z\s', ' ', text)     text = re.sub(r'(\w)\1+', r'\1', text)     text = text.strip()     text = re.sub(r's+', ' ', text)     text = text.replace('\n', '')     text = text.replace('_', '')     return text data['Reviews'] = data['Reviews'].apply(remove) # NLTK word rokenize def tokenize(text):     return text.split() data['Reviews_tokens'] = data['Reviews'].apply(tokenize) data </pre>   | <p>Tahap tokenizing dan cleansing yaitu proses membagi teks menjadi unit-unit kecil yang disebut token dan teks diolah untuk dibersihkan dari elemen yang tidak relevan (menghapus karakter non-alfanumerik dan mengubah teks menjadi huruf kecil)</p> |
| <pre> # get stopword from NLTK stopword # get stopword indonesia list_stopwords = stopwords.words('indonesian') # manually add stopword # append additional stopword list_stopwords.extend(["yg", "nya", "sih"]) # add stopword from txt file # read txt stopword using pandas txt_stopword = pd.read_csv("stopwords.txt", names= ["stopwords"], header = None) # convert stopword string to list &amp; append additional stopword list_stopwords.extend(txt_stopword["stopwords"][0].split( ' ')) # convert list to dictionary list_stopwords = set(list_stopwords)  #remove stopword pada list token def stopwords_removal(words):     return [word for word in words if word not in list_stopwords]  data['Reviews_Stopword'] = data['Reviews_tokens'].apply(stopwords_removal) data </pre> | <p>Tahap stopwords removal. Menghapus stopword yaitu kata-kata umum yang sering muncul dalam bahasa tertentu dan tidak memberikan informasi penting dalam analisis teks, seperti "dan," "atau," dan lain sebagainya.</p>                               |
| <pre> normalizad_word = pd.read_excel("normalisasi.xlsx")  normalizad_word_dict = {}  for index, row in normalizad_word.iterrows():     if row[0] not in normalizad_word_dict:         normalizad_word_dict[row[0]] = row[1]  def normalized_term(document):     return [normalizad_word_dict[term] if term in </pre>  | <p>Tahap normalisasi yaitu proses mengubah teks menjadi bentuk standar atau normal agar lebih mudah untuk dianalisis dan dibandingkan.</p>   |



|  |   |
|--|---|
| <pre>normalized_word_dict else term for term in document]  data['reviews_normalized'] = data['Reviews_Stopword'].apply(normalized_term)  data['reviews_normalized'].head(10)</pre>   |   |
| <pre># stemmed def stemming(token):     factory = StemmerFactory()     stemmer = factory.create_stemmer()     return [stemmer.stem(word) for word in token]  data['stemming'] = data['reviews_normalized'].apply(lambda x: stemming(x)) data</pre> | <p>Tahap stemming. Proses mengubah kata-kata menjadi bentuk dasarnya untuk mengurangi variasi kata yang memiliki akar kata yang sama.</p> |

## 2.2.6 MNB modeling use 10 percent.ipynb

|   |  |
|---|--|
| <pre># Memisahkan data menjadi data train dan data test train_data, test_data, train_labels, test_labels = train_test_split(text, label, test_size=0.2, random_state=42)  # Menggabungkan data train dan labels menjadi satu dataframe train_df = pd.DataFrame({'text': train_data, 'label': train_labels})  # Menghitung jumlah data positif dan negatif pada data train positive_count_train = (train_df['label'] == 1).sum() negative_count_train = (train_df['label'] == 0).sum()  # Menentukan kelas dengan jumlah data terkecil min_count = min(positive_count_train, negative_count_train)  # Mengambil sampel acak dari kelas positif dan negatif dengan ukuran yang sama positive_samples = train_df[train_df['label'] == 1].sample(n=min_count, random_state=42) negative_samples = train_df[train_df['label'] == 0].sample(n=min_count, random_state=42)  # Menggabungkan kembali sampel positif dan negatif menjadi satu dataframe balanced_train_df = pd.concat([positive_samples, negative_samples])  # Memisahkan data dan labels pada data train yang seimbang balanced_train_data = balanced_train_df['text'] balanced_train_labels = balanced_train_df['label']  # Menghitung rasio data positif dan negatif pada data train yang seimbang total_count_train = len(balanced_train_labels) positive_ratio_train = (balanced_train_labels == 1).sum() / total_count_train negative_ratio_train = (balanced_train_labels == 0).sum() / total_count_train</pre> | <p>Tahap ini melakukan beberapa operasi pada data teks dengan tujuan menghasilkan dataset latih yang seimbang untuk melatih model klasifikasi. Dimulai dari memisahkan data menjadi data latih dan data uji, menggabungkan data latih dan label menjadi satu dataframe, menghitung jumlah data positif dan negatif pada data latih, menentukan kelas dengan jumlah data terkecil, mengambil sampel acak dengan ukuran yang sama dari kelas positif dan negatif, menggabungkan kembali sampel positif dan negatif menjadi satu dataframe, memisahkan data dan label pada data latih yang seimbang, Menghitung jumlah data dan rasio positif dan negatif pada data uji lalu dicetak. Selanjutnya melakukan perhitungan term frequency (TF) vector pada data latih dan data uji, serta melakukan normalisasi pada vektor.</p> |
|---|--|

|   |   |
|---|---|
| <pre> # Menghitung jumlah data positif dan negatif pada data test positive_count_test = (test_labels == 1).sum() negative_count_test = (test_labels == 0).sum()  # Menghitung rasio data positif dan negatif pada data test total_count_test = len(test_labels) positive_ratio_test = positive_count_test / total_count_test negative_ratio_test = negative_count_test / total_count_test  # Mencetak jumlah data training dan testing untuk kelas positif dan negatif print("Jumlah data latih positif:", positive_count_train) print("Jumlah data latih negatif:", negative_count_train) print("Jumlah data uji positif:", positive_count_test) print("Jumlah data uji negatif:", negative_count_test)  # Perhitungan TF vector pada train set cvect = CountVectorizer() TF_vector_train = cvect.fit_transform(train_data)  # Normalisasi TF vector pada train set normalized_TF_vector_train = normalize(TF_vector_train, norm='l1', axis=1)  # Perhitungan TF vector pada test set menggunakan CountVectorizer yang sudah dilatih pada train set TF_vector_test = cvect.transform(test_data)  # Normalisasi TF vector pada test set normalized_TF_vector_test = normalize(TF_vector_test, norm='l1', axis=1) </pre> |   |
| <pre> # Persentase fitur yang ingin dipilih setelah seleksi (10%) percent = 30  # Menghitung jumlah fitur yang diinginkan berdasarkan persentase k = int(percent / 100 * normalized_TF_vector_train.shape[1])  # Menerapkan seleksi fitur dengan chi-square pada train set selector = SelectPercentile(chi2, percentile=percent) tf_mat_train_selected = selector.fit_transform(normalized_TF_vector_train, train_labels)  # Mengaplikasikan seleksi fitur yang sama pada test set tf_mat_test_selected = selector.transform(normalized_TF_vector_test) </pre>  | <p>Selanjutnya melakukan seleksi fitur dengan menggunakan metode chi-square pada data latih dan data uji.</p>   |
| <pre> # Membuat objek model Multinomial Naive Bayes dengan class_prior yang sesuai model = MultinomialNB(class_prior=[negative_ratio, positive_ratio])  # Melatih model dengan data latih yang sudah diseleksi model.fit(tf_mat_train_selected, train_labels) </pre>  | <p>Tahap ini yaitu melatih model Naive Bayes Multinomial dengan menggunakan data latih yang sudah diseleksi fiturnya dan melakukan prediksi pada data uji yang juga</p> |

|  |   |
|--|---|
| <pre> # Melakukan prediksi pada data uji yang sudah diseleksi predictions = model.predict(tf_mat_test_selected)  prediksi_benar = (predictions == test_labels).sum() prediksi_salah = (predictions != test_labels).sum()  # Menghitung jumlah data positif dan negatif pada data uji positive_count_test = (test_labels == 1).sum() negative_count_test = (test_labels == 0).sum()  # Mencetak jumlah data positif dan negatif pada data uji print("Jumlah data uji positif :", positive_count_test) print("Jumlah data uji negatif :", negative_count_test)  print('Jumlah prediksi benar\t:', prediksi_benar) print('Jumlah prediksi salah\t:', prediksi_salah)  prediksi_positif = (predictions == 1).sum() prediksi_negatif = (predictions == 0).sum()  print('Jumlah prediksi positif\t:', prediksi_positif) print('Jumlah prediksi negatif\t:', prediksi_negatif) </pre> | <p>sudah diseleksi fiturnya. Hasil prediksi dan evaluasi performa model (jumlah prediksi benar, salah, positif, dan negatif) dicetak untuk dinilai.</p>   |
| <pre> # Menghitung akurasi accuracy = accuracy_score(test_labels, predictions) print("Akurasi:", accuracy)  # Menghitung precision precision = precision_score(test_labels, predictions) print("Precision:", precision)  # Menghitung recall recall = recall_score(test_labels, predictions) print("Recall:", recall)  # Menghitung F1-score f1 = f1_score(test_labels, predictions) print("F1-score:", f1) </pre>   | <p>Selanjutnya melakukan menghitung beberapa metrik evaluasi performa model klasifikasi, yaitu akurasi, precision, recall, dan F1-score untuk memberikan informasi tentang seberapa baik model klasifikasi yang telah dilatih dapat melakukan prediksi pada data uji.</p>   |
| <pre> # Generate classification report from sklearn.metrics import confusion_matrix from sklearn.metrics import classification_report from sklearn.metrics import accuracy_score # Menghitung confusion matrix cm = confusion_matrix(test_labels, predictions) print(cm) classification_rep = metrics.classification_report(test_labels, predictions, target_names=['negatif', 'positif']) print('Classification Report:\n', classification_rep)  # Menyimpan model joblib.dump(model, 'multinomial_nb_30_percent_model.pkl') </pre>   | <p>Kemudian menghasilkan Classification report untuk informasi yang lebih rinci tentang performa model dalam memprediksi setiap kelas. Termasuk di dalamnya adalah precision, recall, F1-score, dan support (jumlah sampel) untuk setiap kelas. Sehingga dapat digunakan untuk membandingkan model yang berbeda. Lalu hasil model akan disimpan dengan fungsi joblib.</p> |

## **BAB III**

### **PENUTUP**

#### **3.1 Kesimpulan**

Dalam laporan ini, telah dibangun sebuah sistem aplikasi untuk mengidentifikasi sentimen atau emosi dari ulasan teks menggunakan algoritma Naive Bayes. Tujuan utama laporan ini adalah untuk memperoleh model klasifikasi yang dapat mengklasifikasikan ulasan teks menjadi sentimen positif atau negatif. Melalui tahapan preprocessing, training, dan testing, sistem ini mampu memberikan hasil yang cukup baik dalam mengenali sentimen pada data ulasan. Tahapan preprocessing melibatkan ekstraksi fitur dengan menggunakan pembobotan kata berdasarkan metode *Term Frequency* (TF). Selanjutnya, fitur-fitur yang relevan dipilih menggunakan metode Chi-Square. Training dilakukan menggunakan algoritma Multinomial Naive Bayes untuk menghasilkan model klasifikasi yang optimal. Pada tahap testing, dilakukan evaluasi menggunakan metrik akurasi, presisi, recall, dan F1-Score untuk mengukur performa model.

Dalam pengujian yang dilakukan menggunakan dataset ulasan teks, sistem ini mampu mengklasifikasikan sentimen dengan akurasi yang memuaskan yaitu 92,8%. Hal ini menunjukkan bahwa penggunaan algoritma Naive Bayes dalam identifikasi sentimen pada data teks memiliki potensi yang baik. Selanjutnya, model yang dihasilkan dapat diimplementasikan dalam sistem aplikasi berbasis website. Pengguna dapat memasukkan data teks dan mendapatkan output berupa hasil sentimen atau identifikasi emosi. Hal ini memberikan manfaat praktis dalam menganalisis sentimen atau emosi dari ulasan pengguna.

Dengan demikian, sistem identifikasi sentimen data teks menggunakan algoritma Naive Bayes merupakan solusi yang efektif dalam mengklasifikasikan sentimen pada ulasan teks, yang dapat digunakan dalam berbagai aplikasi seperti analisis ulasan produk, peninjauan sosial media, dan banyak lagi.

### **3.2 Saran**

Saran untuk laporan ini agar kedepannya bisa memberikan hasil yang lebih baik adalah sebagai berikut.

- a. Melakukan eksplorasi dan pengujian lebih lanjut terhadap metode preprocessing yang digunakan, seperti penyesuaian stop words dan teknik stemming yang lebih tepat, guna meningkatkan kualitas fitur ekstraksi dari data ulasan
- b. Mengembangkan antarmuka pengguna yang intuitif dan responsif pada aplikasi website, sehingga pengguna dapat dengan mudah memasukkan data teks dan menerima hasil identifikasi sentimen atau emosi secara cepat dan akurat.

## DAFTAR PUSTAKA

- [1] Dewi, A. K., & Sulastri. (2022). Analisis Sentimen Ekspedisi Sicepat Dari Ulasan Google Play Menggunakan Algoritma Naïve Bayes. *Jurnal Teknik Informatika dan Sistem Informasi*, 9(2), 796-805. ISSN: 2407-4322. E-ISSN: 2503-2933. URL: <http://jurnal.mdp.ac.id>. Email: [jatisi@mdp.ac.id](mailto:jatisi@mdp.ac.id)
- [2] Muhabatin, H., Prabowo, C., Ali, I., Rohmat, C. L., & Amalia, D. R. (2021). Klasifikasi Berita Hoax Menggunakan Algoritma Naïve Bayes Berbasis PSO. *Informatics for Educators and Professionals*, 5(2), 156-165. E-ISSN: 2548-3412.
- [3] Hakim, B. (2021). Analisa Sentimen Data Text Preprocessing Pada Data Mining Dengan Menggunakan Machine Learning [Sentiment Analysis Data Text Preprocessing in Data Mining Using Machine Learning]. *Journal of Base Research*, 4(2). Versi Online: <http://journal.ubm.ac.id/index.php/jbase>. DOI: <http://dx.doi.org/10.30813/jbase.v4i2.3000>.
- [4] Rofiqi, M. A., Fauzan, A. C., Agustin, A. P., Saputra, A. A., & Fahma, H. D. (2019). Implementasi Term-Frequency Inverse Document Frequency (TFIDF) Untuk Mencari Relevansi Dokumen Berdasarkan Query [Implementation of Term-Frequency Inverse Document Frequency (TFIDF) for Document Relevance Based on Query]. *ILKOMNIKA: Journal of Computer Science and Applied Informatics*, 1(2), 58-64. E-ISSN: 2715-2731. URL: <http://journal.unublitar.ac.id/ilkomnika>. DOI: <https://doi.org/10.28926/ilkomnika.v1i2.18>.