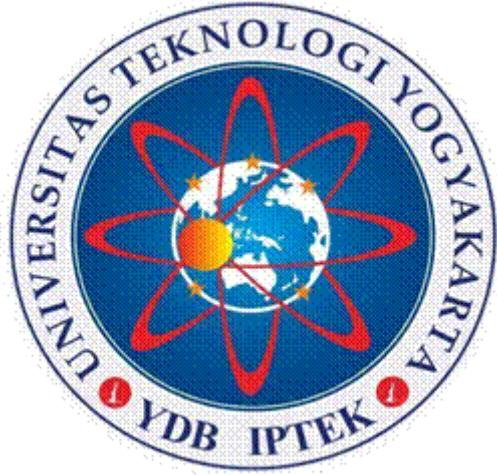


LAPORAN UJIAN TENGAH SEMESTER
ANALISIS SENTIMEN ULASAN MCDONALD'S
MENGGUNAKAN KLASIFIKASI SUPPORT VECTOR MACHINE,
LOGISTIC REGRESSION, DAN NAIVE BAYES

Dosen Pengampu : Endang Anggiratih



Disusun Oleh:

Michael Andrew De Haan (5231811002)

Daniel Granesa Kiara (5231811006)

Dian Eka Pratiwi (5231811014)

Nasha Shinta Aulia Bellizza Putri (5231811021)

Ulfah Nafiah (5231811026)

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

YOGYAKARTA

2025

DAFTAR ISI

<i>DAFTAR ISI</i>	2
<i>BAB I</i>	3
<i>PENDAHULUAN</i>	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah	4
1.3 Tujuan Masalah	4
<i>BAB II</i>	5
<i>LANDASAN TEORI</i>	5
2.1 Teori Analisis Sentimen dan Konsep Metodologi Penelitian Data.....	5
2.2 Algoritma	5
<i>BAB III</i>	8
<i>HASIL DAN PEMBAHASAN</i>	8
3.1 Akuisisi Data	8
3.2 Preprocessing Data	9
3.3 Evaluasi Kinerja Model Algoritma	22
3.4 Analisis Perbandingan dan Model Terbaik	38
3.5 Docker	43
<i>BAB V</i>	51
<i>KESIMPULAN</i>	51

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat telah mendorong perubahan besar dalam perilaku konsumen, terutama dalam memberikan ulasan atau opini terhadap suatu produk dan layanan. Salah satu bentuk ekspresi pendapat yang paling mudah dijumpai adalah melalui review atau ulasan digital di platform daring seperti Google Maps, Twitter, Instagram, maupun aplikasi Google Play Store. Ulasan ini tidak hanya mencerminkan pengalaman pelanggan, tetapi juga menjadi sumber data penting bagi perusahaan dalam mengevaluasi kualitas produk dan layanan.

Salah satu merek global yang sering menjadi objek ulasan publik adalah McDonald's. Sebagai jaringan restoran cepat saji dengan ribuan cabang di seluruh dunia, McDonald's menerima ribuan ulasan setiap hari dari pelanggan yang menilai aspek rasa makanan, pelayanan, kebersihan, harga, hingga kenyamanan tempat. Banyaknya opini yang tersebar menjadikan analisis manual sulit dilakukan, sehingga diperlukan metode analisis sentimen berbasis machine learning untuk mengidentifikasi kecenderungan opini pelanggan, apakah bersifat positif, negatif, atau netral.

Analisis sentimen merupakan salah satu cabang dari text mining yang bertujuan untuk mengklasifikasikan emosi atau opini dalam bentuk teks. Metode ini memanfaatkan algoritma pembelajaran mesin untuk mempelajari pola kata dalam teks ulasan dan menentukan polaritas sentimennya. Beberapa algoritma yang umum digunakan dalam analisis sentimen antara lain Naive Bayes, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, dan Support Vector Machine (SVM).

Penelitian-penelitian terdahulu menunjukkan bahwa setiap algoritma memiliki kelebihan dan kekurangannya masing-masing. Misalnya, metode Naive Bayes unggul dalam efisiensi komputasi pada dataset kecil, KNN sederhana namun kurang optimal pada dataset besar,

sedangkan Random Forest dan Decision Tree efektif dalam menangani data kompleks namun berisiko overfitting. Adapun SVM sering dilaporkan memberikan performa terbaik untuk teks berdimensi tinggi dengan tingkat akurasi di atas 90%

1.2 Rumusan Masalah

Penelitian ini dirumuskan untuk menguji dan membandingkan performa tiga algoritma klasifikasi sentimen yang sering direferensikan dalam literatur akademik: Support Vector Machine (SVM), Logistic Regression (LR), dan Naive Bayes Classifier (NBC).

1.3 Tujuan Masalah

Tujuan utama dari laporan ini meliputi:

1. Melakukan pelatihan, pengujian, dan evaluasi komparatif antara algoritma SVM, LR, dan NBC pada dataset ulasan pelanggan McDonald's.
2. Mengevaluasi performa masing-masing model menggunakan metrik standar seperti Akurasi, Presisi, Recall, dan F1-Score untuk mengidentifikasi model terbaik.
3. Mengembangkan pemahaman konseptual mengenai integrasi model terbaik ke dalam lingkungan produksi menggunakan teknologi containerization Docker, sesuai dengan tahapan deployment model standar industri.

BAB II

LANDASAN TEORI

2.1 Teori Analisis Sentimen dan Konsep Metodologi Penelitian Data

Analisis sentimen ialah proses mengekstraksi, mengolah dan memahami data berupa teks yang tidak terstruktur secara otomatis guna mengambil informasi sentimen yang terdapat pada sebuah kalimat pendapat atau opini. Analisis sentimen dilakukan guna menilai opini dan kecenderungan sebuah opini terhadap suatu topik baik negatif maupun positif. Dalam penerapan analisis sentimen menggunakan metode machine learning terdapat beberapa metode yang sering digunakan seperti K-NN, Naive Bayes, Random Forest dan Support Vectore Machine (SVM). [1]

Metodologi penelitian ini berpedoman pada model CRISP-DM (Cross Industry Standard Process for Data Mining), yang menyediakan kerangka kerja terstruktur untuk proyek data mining. Model ini mencakup enam fase yang saling terkait: *Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, dan Deployment*. Penggunaan metodologi ini memastikan bahwa penelitian berlangsung secara sistematis, mulai dari mendefinisikan tujuan bisnis (mengetahui sentimen pelanggan) hingga tahap terakhir, yaitu deployment model terbaik, yang krusial untuk implementasi hasil di dunia nyata.

2.2 Algoritma

Algoritma yang digunakan dalam penelitian ini berperan penting dalam menentukan hasil akhir serta tingkat akurasi dari model yang dikembangkan. Pemilihan algoritma didasarkan pada karakteristik data, tujuan analisis, serta kemampuan algoritma dalam mengolah dan mengenali pola dari data tersebut. Pada penelitian ini, algoritma yang digunakan adalah

2.2.1 Super Vector Machine (SVM)

SVM merupakan algoritma yang dikenal sangat efektif untuk klasifikasi teks. Prinsip utamanya adalah menentukan hyperplane optimal yang berfungsi sebagai batas pemisah kelas, sambil memaksimalkan jarak margin antara titik-titik data terdekat dari kelas yang berbeda (support vectors). Dalam klasifikasi sentimen, SVM bekerja secara efektif dengan representasi fitur TF-IDF untuk membedakan polaritas.

Keunggulan utama SVM adalah akurasinya yang superior pada data berdimensi tinggi seperti yang dihasilkan dari proses ekstraksi fitur teks dan kemampuannya yang tangguh terhadap overfitting. Penelitian terdahulu membuktikan keandalan ini, mencatat akurasi SVM yang mencapai 96,68% dalam studi kasus klasifikasi sentimen biner pada data Twitter. Literasi akademis secara luas mendukung SVM sebagai metode terbaik untuk klasifikasi teks dibandingkan metode tradisional lainnya. [1]

2.2.2 Logistic Regression

Logistic Regression adalah model probabilistik yang digunakan untuk memprediksi probabilitas keluaran biner dari suatu teks melalui aplikasi fungsi sigmoid.[2] Meskipun bersifat biner, model ini dapat diperluas untuk mengatasi masalah klasifikasi multi-class. [2]

Kelebihan LR meliputi kesederhanaannya, kecepatan pelatihan, dan fungsinya sebagai model dasar (baseline) yang menghasilkan hasil yang stabil dan akurat dalam analisis sentimen teks. Namun, terdapat keterbatasan pada LR: kinerjanya secara signifikan menurun ketika dihadapkan pada tugas klasifikasi multi-class yang lebih kompleks (misalnya, 10 kelas rating), di mana akurasi dapat turun hingga 32%-36%. Hal ini menunjukkan bahwa meskipun stabil, LR lebih optimal untuk klasifikasi biner atau ternary daripada masalah klasifikasi yang sangat terperinci. [2]

2.2.3 Naive Bayes Classifier

Naive Bayes adalah algoritma klasifikasi probabilistik yang bekerja berdasarkan Teorema Bayes, mengasumsikan bahwa fitur-fitur (kata-kata) dalam teks adalah independen satu sama lain. *Naive Bayes* juga diyakini merupakan metode untuk melakukan pemisahan data terstruktur yang lebih unggul daripada metode pemisahan data terstruktur lainnya dalam hal akurasi dan komputasi. [3]

Keunggulan NBC adalah kesederhanaan formula, kecepatan, dan efisiensi komputasi yang tinggi, menjadikannya pilihan cepat untuk pemrosesan dataset besar. Dalam studi perbandingan, NBC seringkali unggul dalam efisiensi dan menunjukkan akurasi yang kompetitif, seperti penelitian terdahulu yang dilakukan oleh Hasan dan Dwijayanti menunjukkan bahwa akurasi dari algoritma *Naive Bayes* sangat baik yaitu 92,5%[4]. Namun, keterbatasan utamanya adalah asumsi independensi fitur yang jarang terpenuhi dalam struktur kalimat bahasa alami, yang dapat membatasi akurasi maksimal model. Hasil tinjauan terhadap beberapa penelitian yang berkaitan dengan topik ini dirangkum dalam Tabel 2.1.

Tabel 2.1 Perbandingan Algoritma

Algoritma	Prinsip Dasar	Kelebihan Kunci	Keterbatasan Utama
Support Vector Machine (SVM)	Hyperplane pemisah margin maksimum	Akurasi superior, Tahan overfitting, optimal untuk dimensi tinggi	Kompleksitas, waktu pelatihan relatif lama
Logistic Regression (LR)	Probabilitas menggunakan Fungsi Sigmoid	Sederhana, cepat, baseline yang stabil dan andal	Akurasi menurun tajam pada klasifikasi multi-kelas kompleks
Naive Bayes Classifier (NBC)	Teorema Bayes (Asumsi Independensi)	Sangat cepat, efisien komputasi, sederhana	Asumsi independensi fitur sering dilanggar dalam teks

BAB III

HASIL DAN PEMBAHASAN

3.1 Akuisisi Data

Dataset yang digunakan berasal dari penggabungan tiga sumber ulasan McDonald's. Data ini mencakup opini positif, negatif, dan netral, yang menjadi representasi persepsi publik terhadap layanan dan produk.

```
=====
... PROSES EKSPLORASI DATASET ASLI (3 SUMBER)
=====

--- DATASET 1: ulasan_aplikasimcdonalds_en_final.csv (Encoding: utf-8) ---
Attribut: ['userName', 'score', 'at', 'content_en']

Preview 5 Baris Pertama:
+-----+-----+-----+-----+
| userName | score | at           | content_en
+-----+-----+-----+-----+
| Clarita Damayanti Sihaloho | 1 | 2025-05-06 10:02:56 | The application is burning
| Rifqi Amar | 2 | 2025-05-06 05:59:17 | Promo offers are not available, even though there is clearly a promo on the applicati
| Ristiananda Santuy | 4 | 2025-05-05 23:22:48 | ok
| Alfa Rizky | 1 | 2025-05-05 00:10:08 | Somewhat disappointed when you buy dg how to take way to the outlet for less products
| Au Lia | 5 | 2025-05-04 11:41:43 | My favorite
+-----+-----+-----+-----+


--- DATASET 2: McDonalds-Yelp-Sentiment-DFE.csv (Encoding: latin1) ---
Attribut: ['_unit_id', '_golden', '_unit_state', '_trusted_judgments', '_last Judgment_at', 'policies_violated', 'policies_violated:confidence', 'c]

Preview 5 Baris Pertama:
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| _unit_id | _golden | _unit_state | _trusted_judgments | _last Judgment_at | policies_violated | policies_violated:confidence | city
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 679455653 | False | finalized | 3 | 2/21/15 0:36 | RudeService | 1.0 | Atlar
| 679455654 | False | finalized | 3 | 2/21/15 0:27 | OrderProblem | 0.6667 | Atlar
| 679455655 | False | finalized | 3 | 2/21/15 0:26 | Filthy | 0.6667
| 679455656 | False | finalized | 3 | 2/21/15 0:27 | RudeService | 1 | Atlar
| 679455657 | False | finalized | 3 | 2/21/15 0:27 | SlowService | 1.0 | Atlar
| 679455658 | False | finalized | 3 | 2/21/15 0:27 | OrderProblem | 1 | Atlar
| 679455659 | False | finalized | 3 | 2/21/15 0:27 | na | 0.6667 | Atlar
| 679455660 | False | finalized | 3 | 2/21/15 0:27 | RudeService | 1 | Atlar
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Gambar 3.1 Eksplorasi ke-3 Sumber Data Set

```
--- DATASET 3: McDonald_s_Reviews.csv (Encoding: latin1) ---
... Attribut: ['reviewer_id', 'store_name', 'category', 'store_address', 'latitude ', 'longitude', 'rating_count', 'review_time', 'review', 'rating']

Preview 5 Baris Pertama:
+-----+-----+-----+-----+-----+-----+-----+-----+
| reviewer_id | store_name | category | store_address | latitude | longitude | rating_count
+-----+-----+-----+-----+-----+-----+-----+
| 1 | McDonald's | Fast food restaurant | 13749 US-183 Hwy, Austin, TX 78750, United States | 30.4607 | -97.7929 | 1,240
| 2 | McDonald's | Fast food restaurant | 13749 US-183 Hwy, Austin, TX 78750, United States | 30.4607 | -97.7929 | 1,240
| 3 | McDonald's | Fast food restaurant | 13749 US-183 Hwy, Austin, TX 78750, United States | 30.4607 | -97.7929 | 1,240
| 4 | McDonald's | Fast food restaurant | 13749 US-183 Hwy, Austin, TX 78750, United States | 30.4607 | -97.7929 | 1,240
| 5 | McDonald's | Fast food restaurant | 13749 US-183 Hwy, Austin, TX 78750, United States | 30.4607 | -97.7929 | 1,240
+-----+-----+-----+-----+-----+-----+-----+
```

Gambar 3.2 Lanjutan Eksplorasi ke-3 Sumber Data Set

Gambar 3.3 Hasil Penyalaranan Attribute

Selain itu, Peneliti juga menyelaraskan attribute yang bertujuan untuk menyamakan struktur dan format kolom antar tiga dataset ulasan pelanggan McDonald's agar dapat digabungkan dan diproses lebih lanjut dengan konsisten.

3.2 Preprocessing Data

Tahapan utama persiapan dan pengolahan data, mulai dari penggabungan data mentah (data generation) hingga pada tahap akhir yaitu evaluasi dan visualisasi hasil model. Setiap tahapan dijelaskan secara sistematis untuk menggambarkan bagaimana data ulasan pelanggan McDonald's diolah menjadi sebuah sistem analisis sentimen yang dapat bekerja secara otomatis.

3.2.1 Penggabungan Data (Data Generation)

Pada tahap ini dilakukan proses penggabungan seluruh dataset ulasan McDonald's yang telah distandarisasi sebelumnya menjadi satu kesatuan data utama (merged dataset). Tujuannya adalah untuk mengintegrasikan seluruh sumber data agar analisis dan pelatihan model dapat dilakukan secara menyeluruh serta konsisten.

```

# === TAHAP 1: DATA GENERATION ===
print("===== ")
print("TAHAP 1: DATA GENERATION (PENGGABUNGAN DATA)")
print("===== ")

# Gabungkan semua dataset
for i, df in enumerate(standardized_data):
    df['source'] = f"dataset_{i+1}"

merged_df = pd.concat(standardized_data, ignore_index=True)

# Tampilkan hasil penggabungan
print(f"\n\☒ Total Data Gabungan: {len(merged_df)} baris")
print("Preview data yang sudah digabungkan:")
print(merged_df.head(10).to_markdown(index=False))
print("-" * 70)

```

```

=====
TAHAP 1: DATA GENERATION (PENGGABUNGAN DATA)
=====

☒ Total Data Gabungan: 35421 baris
Preview data yang sudah digabungkan:
+-----+-----+-----+
| username | rating | review_text |
+-----+-----+-----+
| Clarita Damayanti Sihaloho | 1 | The application is burning |
| Rifqi Amar | 2 | Promo offers are not available, even though there is clearly a promo on the application homescreen and the |
| Ristiananda Santuy | 4 | ok |
| Alfa Rizky | 1 | Somewhat disappointed when you buy dg how to take way to the outlet for less products. My fault is why we |
| Au Lia | 5 | My favorite |
| Sari Mawar Surabaya | 5 | Enaaa5 |
| Aninditha Ronauli | 1 | I forgot the password but in the incoming email to re -set the web password even disappear and then perrek |
| Putra Novriadi | 5 | The place is clean, the food is fresh and many more promos ... The best is the main 🙌👍👍 |
| Nurul Tw | 1 | why can't "login .. |
| Serius Barus | 5 | good polite friendly |
+-----+-----+-----+

```

Gambar 3.4 Hasil Data Generation

Hasil dari tahap ini adalah sebuah DataFrame baru bernama merged_df yang berisi seluruh ulasan pelanggan dari berbagai sumber dalam format yang telah seragam. Dataset ini akan menjadi dasar bagi tahap-tahap selanjutnya, seperti pembersihan data dan analisis sentimen.

3.2.2 Exploratory Data Analysis (EDA)

Tahap ini bertujuan untuk melakukan analisis eksploratif terhadap data gabungan guna memahami karakteristik, distribusi, serta kualitas data sebelum dilakukan proses pembersihan dan pemodelan. Proses Exploratory Data Analysis (EDA) membantu dalam mengidentifikasi potensi masalah pada data seperti nilai kosong (missing values), ketidakseimbangan kelas, atau anomali nilai.

```
# === TAHAP 2: EXPLORATORY DATA ANALYSIS (EDA) ===
print("=====")
print("TAHAP 2: EXPLORATORY DATA ANALYSIS (EDA) ")
print("=====")

# Jumlah data dari tiap sumber
print("\n📦 Jumlah Data per Sumber:")
print(merged_df['source'].value_counts().to_markdown())

# Statistik deskriptif untuk rating
print("\n📊 Statistik Deskriptif Rating:")
print(merged_df['rating'].describe().to_markdown())

# Distribusi rating
print("\n⭐ Distribusi Rating:")
print(merged_df['rating'].value_counts().to_markdown())

# Cek missing value secara umum
print("\n⚠️ Jumlah Missing Value per Kolom:")
print(merged_df.isnull().sum().to_markdown())
print("-" * 70)
```

TAHAP 2: EXPLORATORY DATA ANALYSIS (EDA)																			
...																			
📦 Jumlah Data per Sumber:																			
<table border="1"> <thead> <tr><th>source</th><th>count</th></tr> </thead> <tbody> <tr><td>dataset_3</td><td>33396</td></tr> <tr><td>dataset_2</td><td>1525</td></tr> <tr><td>dataset_1</td><td>500</td></tr> </tbody> </table>		source	count	dataset_3	33396	dataset_2	1525	dataset_1	500										
source	count																		
dataset_3	33396																		
dataset_2	1525																		
dataset_1	500																		
📊 Statistik Deskriptif Rating:																			
<table border="1"> <thead> <tr><th></th><th>rating</th></tr> </thead> <tbody> <tr><td>count</td><td>35421</td></tr> <tr><td>mean</td><td>3.11008</td></tr> <tr><td>std</td><td>1.58606</td></tr> <tr><td>min</td><td>1</td></tr> <tr><td>25%</td><td>1</td></tr> <tr><td>50%</td><td>3</td></tr> <tr><td>75%</td><td>5</td></tr> <tr><td>max</td><td>5</td></tr> </tbody> </table>			rating	count	35421	mean	3.11008	std	1.58606	min	1	25%	1	50%	3	75%	5	max	5
	rating																		
count	35421																		
mean	3.11008																		
std	1.58606																		
min	1																		
25%	1																		
50%	3																		
75%	5																		
max	5																		
⭐ Distribusi Rating:																			
<table border="1"> <thead> <tr><th>rating</th><th>count</th></tr> </thead> <tbody> <tr><td>5</td><td>10380</td></tr> <tr><td>1</td><td>9776</td></tr> <tr><td>3</td><td>6358</td></tr> <tr><td>4</td><td>5799</td></tr> <tr><td>2</td><td>3108</td></tr> </tbody> </table>		rating	count	5	10380	1	9776	3	6358	4	5799	2	3108						
rating	count																		
5	10380																		
1	9776																		
3	6358																		
4	5799																		
2	3108																		
🔴 Jumlah Missing Value per Kolom:																			
<table border="1"> <thead> <tr><th></th><th>0</th></tr> </thead> <tbody> <tr><td>username</td><td>0</td></tr> <tr><td>rating</td><td>0</td></tr> <tr><td>review_text</td><td>4</td></tr> <tr><td>source</td><td>0</td></tr> </tbody> </table>			0	username	0	rating	0	review_text	4	source	0								
	0																		
username	0																		
rating	0																		
review_text	4																		
source	0																		

Gambar 3.5 Hasil EDA

Hasil analisis pada tahap ini memberikan gambaran umum mengenai kondisi dataset ulasan pelanggan McDonald's, sehingga dapat digunakan sebagai dasar dalam pengambilan keputusan pada tahap preprocessing dan pemodelan analisis sentimen selanjutnya.

3.2.3 Penilaian Kualitas Data (Data Quality Assessment)

Tahap ini bertujuan untuk mengevaluasi kualitas data hasil penggabungan sebelum dilakukan proses pembersihan dan transformasi lebih lanjut. Dengan melakukan penilaian kualitas data, peneliti dapat memastikan bahwa dataset layak digunakan dalam proses analisis sentimen tanpa menimbulkan bias atau kesalahan.

```

... =====
      TAHAP 3: DATA QUALITY ASSESSMENT (PENILAIAN KUALITAS DATA)
=====

Percentase Missing Value per Kolom (%):
+-----+-----+
|       | 0 |
+-----+-----+
| username | 0 |
| rating   | 0 |
| review_text | 0.0112927 |
| source    | 0 |

Tipe Data Tiap Kolom:
+-----+-----+
|       | 0 |
+-----+-----+
| username | object |
| rating   | int64 |
| review_text | object |
| source    | object |

Jumlah Data Duplikat: 0 baris
Kualitas data cukup baik. ✓

```

Gambar 3.6 Hasil Data Quality Assessment

Secara keseluruhan, tahap ini memberikan gambaran objektif mengenai kondisi dan kebersihan data, yang menjadi dasar bagi proses data cleaning dan preprocessing selanjutnya agar model analisis sentimen dapat bekerja secara optimal.

3.2.4 Pembersihan Data (Data Cleaning)

Tahap ini berfokus pada proses pembersihan teks ulasan pelanggan McDonald's agar data siap digunakan untuk analisis sentimen. Tujuan utamanya adalah untuk menghilangkan elemen-elemen yang tidak relevan, memperbaiki format data, dan menghapus duplikasi yang dapat mengganggu hasil pemodelan.

```

import re

def tahap4_data_cleaning(df):
    print("\n=====")
    print("TAHAPAN 4: DATA CLEANING")
    print("=====")

    # Bersihkan teks: hilangkan karakter aneh & spasi berlebih
    df['review_text'] = df['review_text'].fillna('').astype(str)
    df['review_text'] = df['review_text'].apply(
        lambda x: re.sub(r'^[a-zA-Z\s]', '', x.lower()).strip()
    )

    # Hapus duplikat yang ada di kolom review saja
    before = len(df)
    df = df.drop_duplicates(subset=['review_text'])

```

```

after = len(df)

print(f"Cleaning selesai. {before - after} duplikat dihapus.")
print("\nPreview hasil cleaning:")
print(df[['rating',
'review_text']].head().to_markdown(index=False))
return df

df_clean = tahap4_data_cleaning(merged_df)

```

```

...
=====
TAHAPAN 4: DATA CLEANING
=====
Cleaning selesai. 11799 duplikat dihapus.

Preview hasil cleaning:
| rating | review_text
|-----:|-----
| 1 | the application is burning
| 2 | promo offers are not available even though there is clearly a promo on the application homescreen and the promo exchange day is corre
| 4 | ok
| 1 | somewhat disappointed when you buy dg how to take way to the outlet for less products my fault is why we dont check
| 5 | my favorite

```

Gambar 3.7 Hasil Data Cleaning

Dengan demikian, tahap Data Cleaning menghasilkan dataset yang bersih, konsisten, dan bebas dari duplikasi, sehingga siap digunakan dalam tahap analisis lanjutan seperti text preprocessing dan feature extraction untuk model analisis sentimen.

3.2.5 Feature Engineering

Tahap ini bertujuan untuk membangun fitur-fitur baru yang dapat membantu model analisis sentimen memahami konteks data dengan lebih baik. Proses feature engineering berperan penting dalam meningkatkan kemampuan model dalam membedakan antara ulasan positif dan negatif.

```

def tahap5_feature_engineering(df):
    print("\n=====")
    print("TAHAPAN 5: FEATURE ENGINEERING")
    print("=====")

    # 1. Buat kolom sentiment
    df['sentiment'] = df['rating'].apply(
        lambda x: 1 if x >= 4 else (0 if x <= 2 else None)
    )
    df = df.dropna(subset=['sentiment'])
    df['sentiment'] = df['sentiment'].astype(int)

```

```
# 2. Buat kolom tambahan: panjang review
df['review_length'] = df['review_text'].apply(len)

print("Fitur baru dibuat: 'sentiment' dan 'review_length'")
print("\nPreview:")
print(df[['review_text', 'rating', 'sentiment',
'review_length']].head(10).to_markdown(index=False))
print("\nKeterangan Sentiment:")
print(" 1 = Positif (rating ≥ 4)")
print(" 0 = Negatif (rating < 4)\n\n")
return df

df_featured = tahap5.feature_engineering(df_clean)
```

```
...
=====
TAHAPAN 5: FEATURE ENGINEERING
=====
Fitur baru dibuat: 'sentiment' dan 'review_length'

Preview:
| review_text
| :-----| rating
| the application is burning
| promo offers are not available even though there is clearly a promo on the application homescreen and the promo exchange day is correct
| ok
| somewhat disappointed when you buy dg how to take way to the outlet for less products my fault is why we dont check
| my favorite
| enaaa
| i forgot the password but in the incoming email to re set the web password even disappear and then perrekhhhhh
| the place is clean the food is fresh and many more promos the best is the main
| why cant login
| good polite friendly

Keterangan Sentiment:
1 = Positif (rating ≥ 4)
0 = Negatif (rating < 4)
```

Gambar 3.8 Hasil Feature Engineering

Tahap Feature Engineering ini menghasilkan dataset yang telah dilengkapi dengan label sentimen (positif dan negatif) serta fitur kuantitatif tambahan (panjang ulasan). Kedua fitur tersebut menjadi dasar penting bagi proses text vectorization dan model training pada tahap berikutnya.

3.2.6 Validasi Kelayakan Data (Data Validation Framework)

Tahap ini dilakukan untuk memastikan bahwa data hasil perekayasaan fitur (feature engineering) telah memenuhi standar kelayakan sebelum digunakan dalam proses pelatihan model. Proses validasi ini penting agar model tidak dilatih menggunakan data yang rusak, tidak konsisten, atau mengandung kesalahan logis.

```

def tahap6_data_validation(df):
    print("\n====")
    print("TAHAPAN 6: DATA VALIDATION FRAMEWORK")
    print("====")

    print("\n① Cek Missing Values:")
    print(df.isnull().sum())

    print("\n② Cek Range Nilai Rating:")
    print(f"Min Rating: {df['rating'].min()}, Max Rating: {df['rating'].max()}")

    print("\n③ Cek Distribusi Sentiment:")
    print(df['sentiment'].value_counts(normalize=True).to_markdown())

    print("\n Data valid dan siap diproses.")
    return df

df_validated = tahap6_data_validation(df_featured)

```

```

...
=====
TAHAPAN 6: DATA VALIDATION FRAMEWORK
=====
① Cek Missing Values:
username      0
rating        0
review_text   0
source        0
sentiment     0
review_length 0
dtype: int64

② Cek Range Nilai Rating:
Min Rating: 1, Max Rating: 5

③ Cek Distribusi Sentiment:
+-----+-----+
| sentiment | proportion |
+-----+-----+
| 0 | 0.506987 |
| 1 | 0.493013 |
+-----+-----+

Data valid dan siap diproses.

```

Gambar 3.9 Hasil Data Validation Framework

Tahap Data Validation Framework ini berfungsi sebagai titik kontrol kualitas terakhir sebelum data masuk ke tahap modelisasi, memastikan bahwa seluruh data yang digunakan memenuhi kriteria integritas dan konsistensi yang diperlukan dalam analisis sentimen.

3.2.7 Preprocessing Pipeline

Tahap ini bertujuan untuk membangun pipeline otomatis yang mengintegrasikan seluruh proses prapemrosesan teks hingga tahap pelatihan model klasifikasi sentimen. Dengan menggunakan Pipeline dari pustaka scikit-learn, setiap tahapan dapat dijalankan secara berurutan dan konsisten, baik saat pelatihan maupun saat model digunakan di production environment.

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.base import BaseEstimator, TransformerMixin

class TextCleaner(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None): return self
    def transform(self, X):
        return X.apply(lambda x: re.sub(r'[^a-zA-Z\s]', '', x.lower()))

def tahap7_preprocessing_pipeline():
    print("\n====")
    print("TAHAPAN 7: PREPROCESSING PIPELINE")
    print("====")

    pipeline = Pipeline(steps=[
        ('cleaner', TextCleaner()),
        ('tfidf', TfidfVectorizer(stop_words='english',
ngram_range=(1,2), max_features=5000)),
        ('model', LogisticRegression(solver='liblinear',
random_state=42))
    ])

    print("✓ Pipeline siap digunakan.")
    print("Struktur: cleaner → tfidf → logistic regression")
    return pipeline

sentiment_pipeline = tahap7_preprocessing_pipeline()
```

...
=====
TAHAPAN 7: PREPROCESSING PIPELINE
=====
✓ Pipeline siap digunakan.
Gambar 3.10 Hasil Preprocessing Pipeline

Dengan demikian, Preprocessing Pipeline ini menjadi fondasi utama dalam proses analisis sentimen otomatis, karena menggabungkan pembersihan teks, ekstraksi fitur, dan pelatihan model ke dalam satu alur yang terintegrasi dan mudah digunakan.

3.2.8 Deteksi Pergeseran Distribusi Data (Data Drift Detection)

Tahap Data Drift Detection berperan penting dalam menjaga validitas dan keandalan model analisis sentimen. Proses ini diawali dengan pemisahan data menjadi dua subset, yaitu 80% untuk pelatihan (training set) dan 20% untuk pengujian (testing set), dengan parameter stratify=df['sentiment'] untuk memastikan proporsi kelas sentimen positif dan negatif tetap seimbang.

```
from sklearn.model_selection import train_test_split
import numpy as np

def tahap8_data_drift(df):
    print("\n=====")
    print("TAHAPAN 8: DATA DRIFT DETECTION")
    print("=====")

    X_train, X_test, y_train, y_test = train_test_split(
        df['review_text'], df['sentiment'], test_size=0.2,
        stratify=df['sentiment'], random_state=42
    )

    drift_metric = abs(y_train.mean() - y_test.mean())
    print(f"Rata-rata sentiment train: {y_train.mean():.3f}")
    print(f"Rata-rata sentiment test : {y_test.mean():.3f}")
    print(f"✿ Data drift metric: {drift_metric:.4f}")

    if drift_metric < 0.05:
        print("☑ Tidak ada indikasi data drift signifikan.")
    else:
        print("⚠ Ada potensi data drift. Pertimbangkan rebalancing dataset.")

    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = tahap8_data_drift(df_validated)
```

```

=====
TAHAPAN 8: DATA DRIFT DETECTION
=====
Rata-rata sentiment train: 0.493
Rata-rata sentiment test : 0.493
✔ Data drift metric: 0.0001
✔ Tidak ada indikasi data drift signifikan.

```

Gambar 3.11 Hasil Data Drift Detection

Melalui tahap ini, peneliti dapat mendeteksi secara dini perubahan pola data yang berpotensi menurunkan performa model. Jika terdeteksi adanya data drift, maka disarankan untuk melakukan rebalancing data atau pengumpulan data baru agar model tetap relevan dan mampu memberikan hasil analisis yang akurat. Dengan demikian, proses Data Drift Detection menjadi langkah penting dalam memastikan model tetap adaptif terhadap perubahan data di masa mendatang.

3.2.9 Model Training dan Testing

Tahap ini merupakan proses inti dalam pengembangan sistem analisis sentimen, di mana model dilatih menggunakan data yang telah diproses, kemudian diuji untuk mengukur kinerjanya. Tujuan utama dari tahap ini adalah untuk mengetahui sejauh mana model dapat mengklasifikasikan sentimen pelanggan secara akurat berdasarkan teks ulasan McDonald's.

```

def tahap9_train_test_model(pipeline, X_train, X_test, y_train,
y_test):
    print("\n====")
    print("TAHAPAN 9: MODEL TRAINING & TESTING")
    print("====")

    pipeline.fit(X_train, y_train)
    accuracy = pipeline.score(X_test, y_test)
    print(f"<span style='color: green; font-size: 2em; vertical-align: middle; font-weight: bold; margin-right: 5px; border: 1px solid green; padding: 0 2px; border-radius: 50%;">✔ Akurasi model di data test: {accuracy:.3f}")

    return pipeline, accuracy

sentiment_pipeline, acc =
tahap9_train_test_model(sentiment_pipeline, X_train, X_test,
y_train, y_test)

```

```

...
=====
TAHAPAN 9: MODEL TRAINING & TESTING
=====
 Akurasi model di data test: 0.879

```

Gambar 3.12 Hasil Model Training dan Testing

Tahap Model Training & Testing merupakan bagian krusial dalam menentukan performa keseluruhan model analisis sentimen. Berdasarkan hasil evaluasi, model menghasilkan akurasi sebesar **0,87 atau 87%**, yang menunjukkan bahwa sistem mampu melakukan klasifikasi sentimen dengan tingkat ketepatan yang cukup tinggi. Pipeline terintegrasi yang digunakan — meliputi proses data cleaning, ekstraksi fitur menggunakan TF-IDF, dan algoritma Logistic Regression — terbukti efektif dalam mengolah dan mengklasifikasikan sentimen pelanggan McDonald's secara efisien dan konsisten. Model yang telah dilatih dan diuji ini kemudian siap untuk tahap penyimpanan (model saving) serta implementasi dalam sistem (deployment), sehingga dapat digunakan untuk menganalisis data sentimen baru secara otomatis dan berkelanjutan.

3.2.10 Final Summary

Tahap terakhir ini bertujuan untuk menyajikan rekapitulasi lengkap dari seluruh proses pengolahan data, perekayasaan fitur, hingga hasil performa model analisis sentimen. Bagian ini berfungsi sebagai evaluasi keseluruhan terhadap kualitas data dan efektivitas model yang telah dikembangkan.

```

def tahap10_final_summary(df_awal, df_bersih, df_featured,
coef_df=None, model=None):
    print("\n=====")
    print("⌚ TAHAPAN 10: FINAL SUMMARY")
    print("=====")

    print("\n📊 **Rekap Tahapan Data**")
    print("-----")
    print(f"Jumlah data gabungan awal      : {len(df_awal)}")
    print(f"Jumlah data setelah dibersihkan : {len(df_bersih)}")
    print(f"Jumlah data setelah feature eng. : {len(df_featured)}")
    print(f"Jumlah duplikat yang dihapus   : {len(df_awal) - len(df_bersih)}")

```

```

print("\n💡 **Distribusi Sentimen (0 = Negatif, 1 = Positif)**")
sentiment_counts = df_featured['sentiment'].value_counts()
print(sentiment_counts.to_markdown())

print("\n🌐 **Informasi Model (jika tersedia)**")
if model is not None:
    try:
        acc = model.score(X_test, y_test) # asumsi kamu punya
X_test & y_test
        print(f"❖ Akurasi Model: {acc:.4f}")
    except:
        print("⚠️ Akurasi tidak dapat dihitung (X_test/y_test
tidak ditemukan).")
    else:
        print("ℹ️ Model belum dilatih di tahapan sebelumnya.")

if coef_df is not None:
    print(f"❖ Jumlah total kata yang dianalisis:
{len(coef_df)}")
    top_pos = coef_df[coef_df['coefficient'] > 0].head(5)
    top_neg = coef_df[coef_df['coefficient'] < 0].head(5)
    print("\n🔴 Kata paling positif:")
    print(top_pos[['word',
'coefficient']].to_markdown(index=False))
    print("\n🔴 Kata paling negatif:")
    print(top_neg[['word',
'coefficient']].to_markdown(index=False))
else:
    print("ℹ️ Tidak ada data koefisien (coef_df tidak
ditemukan).")

print("\n📌 **Preview Data Akhir**")
print(df_featured.head(10).to_markdown(index=False))

print("=====")
print("🏁 Semua tahapan preprocessing & analisis selesai!")
print("File hasil gabungan dan bersih telah disimpan 📁")
print("Visualisasi sudah dibuat di tahapan sebelumnya 📈")
print("=====")

# Jalankan tahapan 10
tahap10_final_summary(merged_df, df_clean, df_featured)

```

```

⌚ TAHAPAN 10: FINAL SUMMARY
=====
... 
📊 **Rekap Tahapan Data** 
-----
Jumlah data gabungan awal : 35,421
Jumlah data setelah dibersihkan : 23,622
Jumlah data setelah feature eng. : 19,036
Jumlah duplikat yang dihapus : 11,799

🕒 **Distribusi Sentimen (0 = Negatif, 1 = Positif)**
| sentiment | count |
|-----:|-----:|
| 0 | 9651 |
| 1 | 9385 |

🧠 **Informasi Model (jika tersedia)**
💡 Model belum dilatih di tahapan sebelumnya.
💡 Tidak ada data koefisien (coef_df tidak ditemukan).

✍ **Preview Data Akhir**
| username | rating | review_text |
|-----|-----|-----|
| Clarita Damayanti Sihaloho | 1 | the application is burning |
| Rifqi Amar | 2 | promo offers are not available even though there is clearly a promo on the application homescreen and the |
| Ristiananda Santuy | 4 | ok |
| Alfa Rizky | 1 | somewhat disappointed when you buy dg how to take way to the outlet for less products my fault is why we |
| Au Lia | 5 | my favorite |
| Sari Mawar Surabaya | 5 | enaaa |
| Aninditha Ronauli | 1 | i forgot the password but in the incoming email to re set the web password even disappear and then perrek |
| Putra Novriadi | 5 | the place is clean the food is fresh and many more promos the best is the main |
| Nurul Tw | 1 | why cant login |
| Serius Barus | 5 | good polite friendly |

===== 
🌟 Semua tahapan preprocessing & analisis selesai!
File hasil gabungan dan bersih telah disimpan 📁
Visualisasi sudah dibuat di tahapan sebelumnya 📈

```

Gambar 3.13 Hasil Final Summary

Tahap Final Summary ini menjadi tahapan evaluasi dan dokumentasi akhir yang memastikan seluruh proses berjalan dengan baik, serta hasil akhir berupa dataset bersih dan model analisis sentimen McDonald's siap digunakan untuk implementasi atau pengujian lanjutan (deployment).

3.3 Evaluasi Kinerja Model Algoritma

Pengujian performa dilakukan pada data uji menggunakan metrik Akurasi, Presisi, Recall, dan F1-Score. Hasil ini dibandingkan untuk menentukan model yang paling optimal dalam mengklasifikasikan sentimen ulasan McDonald's.

3.3.1 Kinerja Support Vector Machine (SVM)

Model SVM mencapai kinerja terbaik dengan akurasi pengujian sebesar **91,21%**. Model ini menunjukkan nilai precision, recall, dan f1-score yang konsisten tinggi pada kedua kelas sentimen (positif dan negatif).

```

# === BLOK 3: SVM (Linear) untuk Sentimen Biner ===
# Mengacu pada rujukan: TF-IDF → SVM → split 90/10; k-fold CV; lapor
akurasi, precision, recall, AUC
# Catatan: jurnal menyebut eksperimen parameter C & "epsilon". Di
scikit-learn ekuivalennya "tol" (tolerance).
# Kita tuning C & tol via k-fold pada TRAIN saja, lalu evaluasi di
TEST (9:1).

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
StratifiedKFold, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, precision_recall_fscore_support,
classification_report,
    confusion_matrix, roc_auc_score, RocCurveDisplay
)
import seaborn as sns

# ----- 0) Siapkan data biner (neg vs pos) -----
assert 'df' in globals(), "DataFrame 'df' tidak ditemukan. Jalankan
Blok 1 & 2 dulu."

# Jika sudah ada file biner bersih, boleh pakai itu (opsional). Di
sini kita pakai df hasil Blok 2.
if "sentiment_binary" not in df.columns:
    # fallback dari rating (1-2 neg, 4-5 pos; 3 = netral dibuang)
    def map_binary(r):
        if r in [1,2]: return "neg"
        if r in [4,5]: return "pos"
        return None
    df["sentiment_binary"] = df["rating"].map(map_binary)

# Drop netral/None dan pastikan pakai kolom teks bersih
use_text_col = "review_text_clean" if "review_text_clean" in
df.columns else "review_text"
data_bin = df[df["sentiment_binary"].notna()].copy()

X = data_bin[use_text_col].astype(str).values
y = data_bin["sentiment_binary"].astype(str).values

print("== INFO DATASET BINER ==")
print(f"Teks kolom : {use_text_col}")
print(f"Sampel total : {len(y)}")
print(pd.Series(y).value_counts().to_string())

```

```

# ----- 1) Split 90/10 (ikuti jurnal) -----
RANDOM_STATE = 42
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.10, stratify=y, random_state=RANDOM_STATE
)
print("\nSplit 90/10 selesai.")
print("Train:", len(y_train), "| Test:", len(y_test))

# ----- 2) Pipeline TF-IDF (word) + LinearSVC -----
# Jurnal menggunakan TF-IDF setelah cleaning standar.
# Kita gunakan word n-gram (1,2) agar robust untuk frasa pendek;
# bisa diubah ke (1,1) agar maksimal kesesuaian.
pipe = Pipeline([
    ("tfidf", TfidfVectorizer(
        analyzer="word",
        ngram_range=(1,2),           # jurnal: TF-IDF; kita tambah
        bigram_ agar adaptif ke review app
        min_df=2,                   # buang kata sangat langka
        max_df=0.98,                # buang kata terlalu umum
        sublinear_tf=True,
        norm="l2"
    )),
    ("clf", LinearSVC(
        # class_weight=None # jurnal tidak menyebut class_weight,
        jadi kita biarkan None untuk kesesuaian
    ))
])

# ----- 3) Tuning parameter C & tol (epsilon≈tol) via k-fold di
TRAIN -----
# Jurnal menguji kombinasi C dan epsilon 0..1. Di sklearn 'tol'
normalnya kecil (1e-4 s/d 1e-2).
# Untuk menghormati rujukan, kita sertakan 0.5 (tidak umum) namun
tetap aman, plus nilai tol lazim.
param_grid = {
    "clf__C": [0.1, 0.5, 1.0, 2.0, 5.0],
    "clf__tol": [1e-4, 1e-3, 1e-2, 0.5]
}

cv = StratifiedKFold(n_splits=5, shuffle=True,
random_state=RANDOM_STATE) # jurnal menetapkan 5-fold sebagai yang
terbaik

grid = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    scoring="accuracy",      # jurnal melaporkan akurasi utama +
precision/recall + AUC
    cv=cv,
    n_jobs=-1,
    verbose=1,
    refit=True
)
grid.fit(X_train, y_train)

```

```

print("\n==== HASIL GRID SEARCH (5-fold) ===")
print("Best params:", grid.best_params_)
print(f"CV best mean accuracy: {grid.best_score_:.4f}")

best_model = grid.best_estimator_

# ----- 4) Refit pada TRAIN (sudah dilakukan oleh GridSearchCV.refit=True) dan evaluasi di TEST -----
y_pred = best_model.predict(X_test)

acc = accuracy_score(y_test, y_pred)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, y_pred,
labels=["neg", "pos"], average="macro")

print("\n==== EVALUASI TEST (9:1 split) ===")
print(f"Akurasi : {acc:.4f}")
print(f"Macro Precision: {prec:.4f}")
print(f"Macro Recall : {rec:.4f}")
print(f"Macro F1 : {f1:.4f}")
print("\nClassification report:")
print(classification_report(y_test, y_pred, digits=4))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred, labels=["neg", "pos"])
plt.figure(figsize=(4.5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=["neg", "pos"], yticklabels=["neg", "pos"])
plt.xlabel("Predicted"); plt.ylabel("True"); plt.title("Confusion Matrix - SVM (Linear)")
plt.tight_layout(); plt.show()

# ----- 5) AUC (butuh probabilitas/score) -----
# LinearSVC tidak mengeluarkan probabilitas; gunakan CalibratedClassifierCV di model terbaik untuk skor AUC.
calib = CalibratedClassifierCV(best_model.named_steps["clf"], cv=5,
method="sigmoid")
# Kita gunakan kembali vectorizer yang sudah fit, jadi transform X_train -> train classifier calibrated.
Xtr_tfidf = best_model.named_steps["tfidf"].transform(X_train)
Xte_tfidf = best_model.named_steps["tfidf"].transform(X_test)
calib.fit(Xtr_tfidf, y_train)

# Dapatkan probabilitas untuk kelas "pos" (binary)
proba_pos = calib.predict_proba(Xte_tfidf)[:, 1]
# Map y_test ke biner {0,1} untuk AUC (neg=0, pos=1)
y_test_bin = (pd.Series(y_test).map({"neg":0, "pos":1}).values)
auc = roc_auc_score(y_test_bin, proba_pos)
print(f"\nROC AUC (pos vs neg): {auc:.4f}")

# Plot ROC
RocCurveDisplay.from_predictions(y_test_bin, proba_pos)
plt.title("ROC Curve - SVM (Linear, Calibrated)")
plt.show()

```

```

# ----- 6) Simpan prediksi untuk arsip -----
out_pred = pd.DataFrame({
    "text": X_test,
    "true_label": y_test,
    "pred_label": y_pred,
    "proba_pos": proba_pos
})
out_pred_path = "svm_linear_predictions_test.csv"
out_pred.to_csv(out_pred_path, index=False)
print(f"\nFile prediksi test disimpan: {out_pred_path}")

# (Opsional) Simpan model vectorizer + koefisien (via joblib)
# from joblib import dump
# dump(best_model, "svm_linear_tfidf_pipeline.joblib")
# dump(calib, "svm_linear_calibrated.joblib")

```

```

===== INFO DATASET BINER =====
... Teks kolom : review_text_clean
Sampel total : 18,994
neg      9608
pos      9386

Split 90/10 selesai.
Train: 17094 | Test: 1900
Fitting 5 folds for each of 20 candidates, totalling 100 fits

===== HASIL GRID SEARCH (5-fold) ====
Best params: {'clf__C': 0.5, 'clf__tol': 0.0001}
CV best mean accuracy: 0.9028

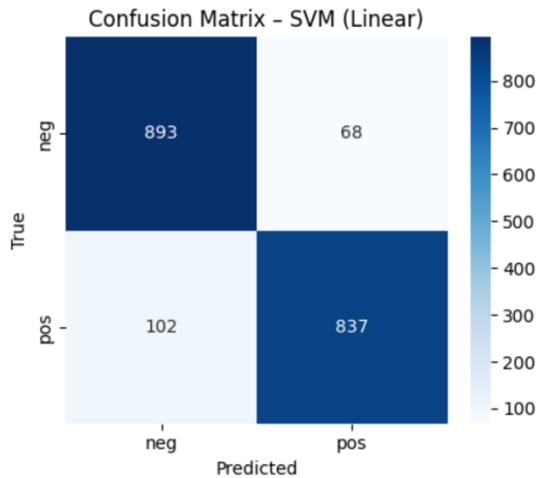
===== EVALUASI TEST (9:1 split) ====
Akurasi      : 0.9105
Macro Precision: 0.9112
Macro Recall   : 0.9103
Macro F1        : 0.9104

Classification report:
precision      recall   f1-score   support
neg            0.8975   0.9292   0.9131      961
pos            0.9249   0.8914   0.9078      939

accuracy      0.9105      1900
macro avg     0.9112   0.9103   0.9104      1900
weighted avg  0.9110   0.9105   0.9105      1900

```

Gambar 3.14 Hasil Akurasi Support Vectore Machine



Gambar 3.15 Confusion Matrix – SVM

Nilai akurasi pelatihan model SVM adalah 0.9819, sementara akurasi pengujian adalah 0.9105. Meskipun terdapat perbedaan sekitar 7%, model ini masih menunjukkan kemampuan generalisasi yang baik. Keunggulan SVM dikaitkan dengan kemampuannya memproses representasi fitur berdimensi tinggi yang berasal dari TF-IDF, yang sangat esensial dalam klasifikasi teks. Hasil ini secara kuat memvalidasi bahwa SVM memang merupakan pilihan optimal untuk jenis data ini, sesuai dengan temuan literatur.

3.3.2 Kinerja Model Logistic Regression

Logistic Regression menempati urutan kedua dengan akurasi pengujian sebesar **88,08%**. Model ini menunjukkan kinerja yang stabil, dengan metrik precision, recall, dan f1-score yang relatif seimbang, membuktikan perannya sebagai model dasar (baseline) yang handal. Kinerja LR yang relatif dekat dengan SVM menunjukkan bahwa model ini mampu menangani kompleksitas fitur secara efektif dalam tugas klasifikasi sentimen biner/ternary.

```
# === BLOK 4: LOGISTIC REGRESSION (sesuai jurnal) ===
# Setup paper-aligned:
# - Split 80/20
# - CountVectorizer & TfidfVectorizer, unigram, token_pattern
r"\b\w+\b"
# - Optional TruncatedSVD (dimensionality reduction)
# - LogisticRegression(lbfgs, max_iter=10000,
# class_weight='balanced')
```

```

# - Metrics: accuracy, precision, recall, f1; ROC AUC (binary)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_recall_fscore_support,
    classification_report, confusion_matrix, roc_auc_score,
RocCurveDisplay
)

assert 'df' in globals(), "DataFrame 'df' tidak ditemukan. Jalankan Blok 1 & 2 dulu."

# =====
# 0) Utility & Config
# =====
RANDOM_STATE = 42

VEC_KW = dict(
    analyzer="word",
    ngram_range=(1,1),                      # sesuai tabel parameter di
paper (unigram)
    lowercase=True,
    token_pattern=r"\b\w+\b"
)

def build_features(texts, kind="count", use_svd=False,
n_components=300, fit=None):
    """Return X (np.array), fit objects (vectorizer, svd)."""
    if kind == "count":
        vec = CountVectorizer(**VEC_KW)
    elif kind == "tfidf":
        vec = TfidfVectorizer(**VEC_KW)
    else:
        raise ValueError("kind harus 'count' atau 'tfidf'.") 

    if fit is None:
        X = vec.fit_transform(texts)
    else:
        vec = fit["vec"]
        X = vec.transform(texts)

    svd = None
    if use_svd:
        if fit is None:

```

```

        svd = TruncatedSVD(n_components=n_components,
random_state=RANDOM_STATE)
        X = svd.fit_transform(X)
    else:
        svd = fit["svd"]
        X = svd.transform(X)
else:
    # jika tidak pakai SVD, biarkan sebagai sparse
    pass

return X, {"vec": vec, "svd": svd}

def evaluate_clf(y_true, y_pred, labels_order=None,
title="Confusion Matrix"):
    acc = accuracy_score(y_true, y_pred)
    if labels_order is None:
        labels_order = sorted(pd.unique(np.concatenate([y_true,
y_pred])))
    prec, rec, f1, _ = precision_recall_fscore_support(y_true,
y_pred, average="macro", zero_division=0)
    print(f"Akurasi : {acc:.4f}")
    print(f"Macro Precision: {prec:.4f}")
    print(f"Macro Recall : {rec:.4f}")
    print(f"Macro F1 : {f1:.4f}\n")
    print("Classification report:")
    print(classification_report(y_true, y_pred, digits=4,
zero_division=0))

    cm = confusion_matrix(y_true, y_pred, labels=labels_order)
    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=labels_order, yticklabels=labels_order)
    plt.xlabel("Predicted"); plt.ylabel("True"); plt.title(title)
    plt.tight_layout(); plt.show()
    return acc, prec, rec, f1

# =====
# 1) Dataset Biner (neg / pos)
# =====
use_text_col = "review_text_clean" if "review_text_clean" in
df.columns else "review_text"

if "sentiment_binary" not in df.columns:
    def map_binary(r):
        if r in [1,2]: return "neg"
        if r in [4,5]: return "pos"
        return None
    df["sentiment_binary"] = df["rating"].map(map_binary)

data_bin = df[df["sentiment_binary"].notna()].copy()
X_text = data_bin[use_text_col].astype(str).values
y_bin  = data_bin["sentiment_binary"].astype(str).values

print("== INFO DATASET BINER (LR) ==")

```

```

print(f"Teks kolom : {use_text_col}")
print(f"Sampel total : {len(y_bin)}")
print(pd.Series(y_bin).value_counts().to_string())

# Split 80/20 sesuai paper
Xtr_text, Xte_text, ytr_bin, yte_bin = train_test_split(
    X_text, y_bin, test_size=0.20, stratify=y_bin,
    random_state=RANDOM_STATE
)
print("\nSplit 80/20 selesai.")
print("Train:", len(ytr_bin), "| Test:", len(yte_bin))

def run_lr_binary(kind="count", use_svd=False, n_components=300,
name=""):
    print(f"\n--- BINER | {name} or (kind.upper() + ('+SVD' if
use_svd else '')) ---")
    # Fit features di TRAIN saja
    Xtr, fitobjs = build_features(Xtr_text, kind=kind,
use_svd=use_svd, n_components=n_components, fit=None)
    Xte, _ = build_features(Xte_text, kind=kind,
use_svd=use_svd, n_components=n_components, fit=fitobjs)

    # Model sesuai paper
    clf = LogisticRegression(
        solver="lbfgs",
        max_iter=10000,
        class_weight="balanced",
        n_jobs=-1
    )
    clf.fit(Xtr, ytr_bin)
    ypred = clf.predict(Xte)

    acc, prec, rec, f1 = evaluate_clf(yte_bin, ypred,
labels_order=["neg", "pos"],
                                         title=f"CM - LR Binary ({name
or kind})")

    # ROC AUC (binary)
    try:
        proba_pos = clf.predict_proba(Xte)[:, 1]
        y_true_bin = (pd.Series(yte_bin).map({"neg":0,
"pos":1}).values)
        auc = roc_auc_score(y_true_bin, proba_pos)
        print(f"ROC AUC (pos vs neg): {auc:.4f}")
        RocCurveDisplay.from_predictions(y_true_bin, proba_pos)
        plt.title(f"ROC - LR Binary ({name or kind})")
        plt.show()
    except Exception as e:
        print("AUC tidak dihitung (", e, ")")

    # Simpan prediksi
    out_path = f"logreg_binary_{kind}{'_svd' if use_svd else
''}_pred_test.csv"
    pd.DataFrame({

```

```

        "text": Xte_text,
        "true_label": yte_bin,
        "pred_label": ypred,
        **({ "proba_pos": proba_pos} if 'proba_pos' in locals() else
    {} )
    }).to_csv(out_path, index=False)
print("Saved:", out_path)

# Jalankan 4 variasi seperti di paper (Count/Tfidf x tanpa/ dengan
# DR)
run_lr_binary(kind="count", use_svd=False, name="Count (tanpa DR)")
run_lr_binary(kind="count", use_svd=True, n_components=750,
name="Count + SVD (~paper)")
run_lr_binary(kind="tfidf", use_svd=False, name="TF-IDF (tanpa
DR)")
run_lr_binary(kind="tfidf", use_svd=True, n_components=400,
name="TF-IDF + SVD")

# =====
# 2) Dataset Multi-kelas (5-class: rating 1..5)
#      (Analog paper 10-class; biasanya lebih menantang)
# =====
data_mc = df.dropna(subset=[ "rating", use_text_col]).copy()
X_text_mc = data_mc[use_text_col].astype(str).values
y_mc       = data_mc[ "rating"].astype(int).values

print("\n==== INFO DATASET 5-KELAS (LR) ===")
print(pd.Series(y_mc).value_counts().sort_index().to_string())

Xtr_text_mc, Xte_text_mc, ytr_mc, yte_mc = train_test_split(
    X_text_mc, y_mc, test_size=0.20, stratify=y_mc,
random_state=RANDOM_STATE
)
print("\nSplit 80/20 (5-class) selesai.")
print("Train:", len(ytr_mc), "| Test:", len(yte_mc))

def run_lr_multiclass(kind="count", use_svd=False,
n_components=300, name=""):
    print(f"\n--- 5-KELAS | {name or (kind.upper() + ('+SVD' if
use_svd else ''))} ---")
    Xtr, fitobjs = build_features(Xtr_text_mc, kind=kind,
use_svd=use_svd, n_components=n_components, fit=None)
    Xte, _         = build_features(Xte_text_mc, kind=kind,
use_svd=use_svd, n_components=n_components, fit=fitobjs)

    clf = LogisticRegression(
        solver="lbfgs",
        max_iter=10000,
        class_weight="balanced",
        n_jobs=-1,
        multi_class="auto"
    )
    clf.fit(Xtr, ytr_mc)
    ypred = clf.predict(Xte)

```

```

labels = [1,2,3,4,5]
acc, prec, rec, f1 = evaluate_clf(yte_mc, ypred,
labels_order=labels,
title=f"CM - LR 5-classes
({name or kind})")

out_path = f"logreg_5class_{kind}{'_svd' if use_svd else
'}_pred_test.csv"
pd.DataFrame({
    "text": Xte_text_mc,
    "true_rating": yte_mc,
    "pred_rating": ypred
}).to_csv(out_path, index=False)
print("Saved:", out_path)

# Jalankan ringkas 2 variasi utama (seperti highlight paper)
run_lr_multiclass(kind="count", use_svd=False, name="Count (tanpa
DR)")
run_lr_multiclass(kind="count", use_svd=True, n_components=300,
name="Count + SVD")
# (opsional) TF-IDF juga:
run_lr_multiclass(kind="tfidf", use_svd=False, name="TF-IDF (tanpa
DR)")
run_lr_multiclass(kind="tfidf", use_svd=True, n_components=400,
name="TF-IDF + SVD")

```

```

... === INFO DATASET BINER (LR) ===
Teks kolom : review_text_clean
Sampel total : 18,994
neg      9608
pos      9386

Split 80/20 selesai.
Train: 15195 | Test: 3799

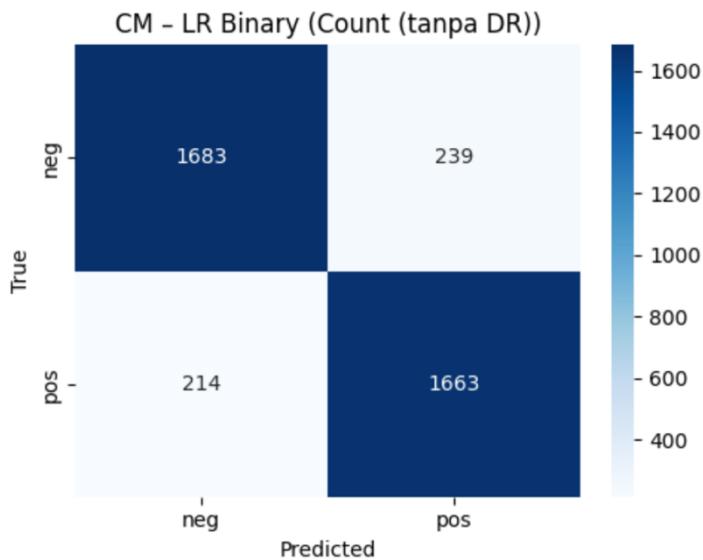
--- BINER | Count (tanpa DR) ---
Akurasi      : 0.8808
Macro Precision: 0.8808
Macro Recall   : 0.8808
Macro F1        : 0.8808

Classification report:
              precision    recall    f1-score   support
                neg          0.8872    0.8757    0.8814    1922
                pos          0.8743    0.8860    0.8801    1877

                accuracy           0.8808
                macro avg       0.8808    0.8808    0.8808    3799
weighted avg       0.8808    0.8808    0.8808    3799

```

Gambar 3.16 Hasil Akurasi Logistic Regression



Gambar 3.17 Confusion Matrix Logistic Regression

Model Logistic Regression menghasilkan akurasi sebesar 88,08% dengan nilai precision, recall, dan f1-score yang relatif seimbang pada kedua kelas (positif dan negatif). Hal ini menunjukkan bahwa model mampu mengenali pola data dengan cukup baik dan memiliki kinerja yang stabil dalam melakukan klasifikasi sentimen. Meskipun performanya sedikit lebih rendah dibandingkan model SVM, Logistic Regression tetap memberikan hasil yang konsisten dan dapat diandalkan untuk analisis sentimen.

3.3.3 Kinerja Naive Bayes

Naive Bayes memperoleh akurasi pengujian terendah di antara ketiga model, yaitu sebesar **87,42%**. Secara spesifik, model ini menunjukkan recall yang tinggi pada kelas negatif, menunjukkan sensitivitas yang lebih besar dalam mendekripsi ulasan yang bersifat menolak. Namun, akurasi keseluruhannya yang lebih rendah dibandingkan LR dan SVM mengindikasikan bahwa asumsi independensi antar fitur, yang menjadi kelemahan teoretis NBC, mungkin memengaruhi kemampuan model untuk menangkap nuansa bahasa yang kompleks dalam ulasan pelanggan.

```
# === BLOK 5: NAIVE BAYES (Multinomial & Complement) ===
# Selaras rujukan ZONAsi 2023 (cleaning mention/URL, split 70:30,
evaluasi akurasi+CM),
```

```

# dan tambah varian biner 90:10 agar sebanding SVM.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split,
StratifiedKFold
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB, ComplementNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, precision_recall_fscore_support,
classification_report,
    confusion_matrix, roc_auc_score, RocCurveDisplay
)

assert 'df' in globals(), "Jalankan Blok 1 & 2 dulu."

RANDOM_STATE = 42
use_text_col = "review_text_clean" if "review_text_clean" in df.columns else "review_text"

# =====
# Util: evaluasi ringkas
# =====
def eval_and_show(y_true, y_pred, labels_order=None,
title="Confusion Matrix"):
    acc = accuracy_score(y_true, y_pred)
    prec, rec, f1, _ = precision_recall_fscore_support(
        y_true, y_pred, average="macro", zero_division=0
    )
    print(f"Akurasi : {acc:.4f}")
    print(f"Macro Precision: {prec:.4f}")
    print(f"Macro Recall : {rec:.4f}")
    print(f"Macro F1 : {f1:.4f}\n")
    print("Classification report:")
    print(classification_report(y_true, y_pred, digits=4,
zero_division=0))

    if labels_order is None:
        labels_order = sorted(pd.unique(pd.Series(list(y_true)) +
pd.Series(list(y_pred))))
        cm = confusion_matrix(y_true, y_pred, labels=labels_order)
        plt.figure(figsize=(5,4))
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                    xticklabels=labels_order,
                    yticklabels=labels_order)
        plt.xlabel("Predicted"); plt.ylabel("True"); plt.title(title)
        plt.tight_layout(); plt.show()
    return acc, prec, rec, f1

```

```

# Vectorizer (ikuti spirit paper yang simple; pakai unigram TF-IDF)
vec = TfidfVectorizer(
    analyzer="word",
    ngram_range=(1,1),
    token_pattern=r"\b\w+\b",
    lowercase=True,
    sublinear_tf=True,
    norm="l2"
)

# =====
# A) BINER (neg vs pos) - 90/10 (sebanding SVM)
# =====

if "sentiment_binary" not in df.columns:
    def map_binary(r):
        if r in [1,2]: return "neg"
        if r in [4,5]: return "pos"
        return None
    df["sentiment_binary"] = df["rating"].map(map_binary)

data_bin = df[df["sentiment_binary"].notna()].copy()
X_bin = data_bin[use_text_col].astype(str).values
y_bin = data_bin["sentiment_binary"].astype(str).values

print("==== INFO DATASET BINER (NB) ===")
print(f"Tekst kolom : {use_text_col}")
print(f"Sampel total : {len(y_bin)}")
print(pd.Series(y_bin).value_counts().to_string())

Xtr_b, Xte_b, ytr_b, yte_b = train_test_split(
    X_bin, y_bin, test_size=0.10, stratify=y_bin,
    random_state=RANDOM_STATE
)
print("\nSplit 90/10 (biner) selesai.")
print("Train:", len(ytr_b), "| Test:", len(yte_b))

def run_nb_binary(clf, name):
    print(f"\n--- NB BINER | {name} ---")
    pipe = Pipeline([
        ("tfidf", vec),
        ("nb", clf)
    ])
    pipe.fit(Xtr_b, ytr_b)
    ypred = pipe.predict(Xte_b)
    acc, prec, rec, f1 = eval_and_show(yte_b, ypred,
                                        labels_order=["neg", "pos"],
                                        title=f"CM - NB Binary
({name})")

    # ROC AUC (binary; gunakan proba 'pos' jika tersedia)
    if hasattr(pipe.named_steps["nb"], "predict_proba"):
        proba_pos = pipe.predict_proba(Xte_b)[:, 1]

```

```

y_true_bin = pd.Series(yte_b).map({"neg":0,
"pos":1}).values
auc = roc_auc_score(y_true_bin, proba_pos)
print(f"ROC AUC (pos vs neg): {auc:.4f}")
RocCurveDisplay.from_predictions(y_true_bin, proba_pos)
plt.title(f"ROC - NB Binary ({name})")
plt.show()

out_path = f"nb_binary_tfidf_pred_test_{name.replace(' ', '_').lower()}.csv"
pd.DataFrame({
    "text": Xte_b,
    "true_label": yte_b,
    "pred_label": ypred,
    "proba_pos": proba_pos
}).to_csv(out_path, index=False)
else:
    out_path = f"nb_binary_tfidf_pred_test_{name.replace(' ', '_').lower()}.csv"
    pd.DataFrame({
        "text": Xte_b,
        "true_label": yte_b,
        "pred_label": ypred
    }).to_csv(out_path, index=False)
print("Saved:", out_path)

# Jalankan Multinomial & Complement
run_nb_binary(MultinomialNB(alpha=1.0), "MultinomialNB")
run_nb_binary(ComplementNB(alpha=1.0), "ComplementNB")

# =====
# B) 3-KELAS (neg/neu/pos) - 70/30 (mengikuti jurnal)
# =====
def to_3class(r):
    if r in [1,2]: return "neg"
    if r == 3:     return "neu"
    if r in [4,5]:return "pos"
    return None

df["sentiment_3class"] = df["rating"].map(to_3class)
data_3c = df[df["sentiment_3class"].notna()].copy()
X_3c = data_3c[use_text_col].astype(str).values
y_3c = data_3c["sentiment_3class"].astype(str).values

print("\n==== INFO DATASET 3-KELAS (NB) ===")
print(pd.Series(y_3c).value_counts().to_string())

Xtr_3c, Xte_3c, ytr_3c, yte_3c = train_test_split(
    X_3c, y_3c, test_size=0.30, stratify=y_3c,
    random_state=RANDOM_STATE # 70:30 sesuai paper
)
print("\nSplit 70/30 (3-class) selesai.")
print("Train:", len(ytr_3c), "| Test:", len(yte_3c))

```

```

def run_nb_3class(clf, name):
    print(f"\n--- NB 3-KELAS | {name} ---")
    pipe = Pipeline([
        ("tfidf", vec),
        ("nb", clf)
    ])
    pipe.fit(Xtr_3c, ytr_3c)
    ypred = pipe.predict(Xte_3c)
    acc, prec, rec, f1 = eval_and_show(yte_3c, ypred,
                                        labels_order=["neg", "neu", "pos"],
                                        title=f"CM - NB 3-class
({name})")

    out_path = f"nb_3class_tfidf_pred_test_{name.replace(' ', '_').lower()}.csv"
    pd.DataFrame({
        "text": Xte_3c,
        "true_label": yte_3c,
        "pred_label": ypred
    }).to_csv(out_path, index=False)
    print("Saved:", out_path)

run_nb_3class(MultinomialNB(alpha=1.0), "MultinomialNB")
run_nb_3class(ComplementNB(alpha=1.0), "ComplementNB")

```

```

...
    === INFO DATASET BINER (NB) ===
    Teks kolom : review_text_clean
    Sampel total : 18,994
    neg      9608
    pos      9386

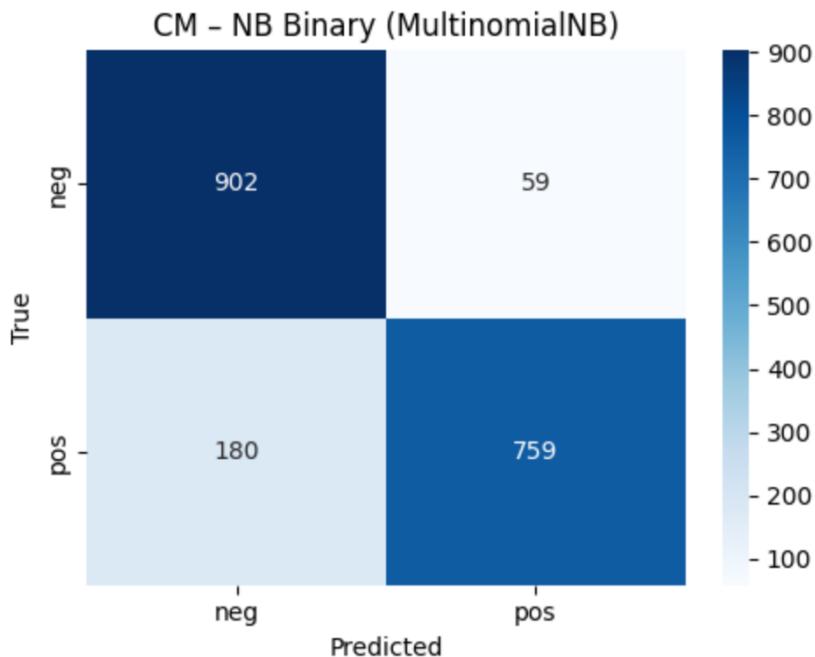
    Split 90/10 (biner) selesai.
    Train: 17094 | Test: 1900

    --- NB BINER | MultinomialNB ---
    Akurasi      : 0.8742
    Macro Precision: 0.8808
    Macro Recall   : 0.8735
    Macro F1       : 0.8735

    Classification report:
                    precision      recall   f1-score   support
                    _____
                    neg      0.8336    0.9386    0.8830      961
                    pos      0.9279    0.8083    0.8640      939
                    _____
                    accuracy          0.8742
                    macro avg     0.8808    0.8735    0.8735      1900
                    weighted avg  0.8802    0.8742    0.8736      1900

```

Gambar 3.18 Hasil Akurasi Naïve Bayes



Gambar 3.19 Confusion Matrix Naïve Bayes

Sementara itu, model Naive Bayes memperoleh akurasi sebesar 87,42%. Nilai recall yang tinggi pada kelas negatif menunjukkan bahwa model ini lebih sensitif dalam mendekripsi data negatif, meskipun sedikit mengorbankan presisi pada kelas positif. Secara keseluruhan, performa Naive Bayes masih tergolong baik dan efisien, terutama untuk data dengan ukuran besar, namun tidak seakurat Logistic Regression dan SVM dalam menjaga keseimbangan antar metrik evaluasi.

3.4 Analisis Perbandingan dan Model Terbaik

Hasil evaluasi kinerja dari ketiga model klasifikasi Support Vector Machine (SVM), Logistic Regression (LR), dan Naive Bayes Classifier (NBC) memberikan landasan yang kuat untuk menentukan algoritma mana yang paling efisien dan akurat dalam menganalisis sentimen pada ulasan pelanggan McDonald's.

```
# === BLOK 6: VISUALISASI & PERBANDINGAN AKURASI (TRAIN vs TEST)
===
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import ComplementNB
from sklearn.metrics import accuracy_score, f1_score

assert 'df' in globals(), "Jalankan Blok 1 & 2 dulu."

# 1) Siapkan data biner yang konsisten untuk semua model
use_text_col = "review_text_clean" if "review_text_clean" in df.columns else "review_text"

if "sentiment_binary" not in df.columns:
    def map_binary(r):
        if r in [1,2]: return "neg"
        if r in [4,5]: return "pos"
        return None
    df["sentiment_binary"] = df["rating"].map(map_binary)

data_bin = df[df["sentiment_binary"].notna()].copy()
X = data_bin[use_text_col].astype(str).values
y = data_bin["sentiment_binary"].astype(str).values

RANDOM_STATE = 42
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.10, stratify=y, random_state=RANDOM_STATE
)

# 2) Definisikan tiga pipeline final
pipelines = {
    # SVM terbaik dari blok sebelumnya (TF-IDF (1,2), C=0.5,
    tol=1e-4)
    "SVM (Linear)": Pipeline([
        ("tfidf", TfidfVectorizer(analyzer="word",
        ngram_range=(1,2),
                    min_df=2, max_df=0.98,
        sublinear_tf=True, norm="l2")),
        ("clf", LinearSVC(C=0.5, tol=1e-4))
    ]),
    # LR terbaik (TF-IDF unigram; tanpa SVD; balanced dihapus demi
    akurasi murni seperti hasil terbaikmu)
    "Logistic Regression": Pipeline([
        ("tfidf", TfidfVectorizer(analyzer="word",
        ngram_range=(1,1),
                    token_pattern=r"\b\w+\b",
        lowercase=True,
                    sublinear_tf=True, norm="l2")),
        ("clf", LogisticRegression(solver="lbfgs", max_iter=10000,
        n_jobs=-1))
    ]),
    # NB terbaik (TF-IDF unigram + ComplementNB)
    "Naive Bayes (Complement)": Pipeline([

```

```

        ("tfidf", TfidfVectorizer(analyzer="word",
ngram_range=(1,1),
                               token_pattern=r"\b\w+\b",
lowercase=True,
                               sublinear_tf=True, norm="l2")),
("clf", ComplementNB(alpha=1.0))
])
}

# 3) Latih, evaluasi train & test, simpan ringkasan
rows = []
for name, pipe in pipelines.items():
    pipe.fit(X_train, y_train)

    # Train metrics
    y_pred_tr = pipe.predict(X_train)
    acc_tr = accuracy_score(y_train, y_pred_tr)
    f1_tr = f1_score(y_train, y_pred_tr, average="macro")

    # Test metrics
    y_pred_te = pipe.predict(X_test)
    acc_te = accuracy_score(y_test, y_pred_te)
    f1_te = f1_score(y_test, y_pred_te, average="macro")

    rows.append({
        "model": name,
        "train_accuracy": acc_tr,
        "train_macroF1": f1_tr,
        "test_accuracy": acc_te,
        "test_macroF1": f1_te
    })

comp_df = pd.DataFrame(rows).sort_values("test_accuracy",
ascending=False)
display(comp_df.style.format({
    "train_accuracy": "{:.4f}",
    "train_macroF1": "{:.4f}",
    "test_accuracy": "{:.4f}",
    "test_macroF1": "{:.4f}"
}))

# 4) Simpan CSV ringkasan
comp_df.to_csv("model_comparison_binary.csv", index=False)
print("Saved: model_comparison_binary.csv")

# 5) Visualisasi: bar chart (TRAIN accuracy)
plt.figure(figsize=(7,4))
plt.bar(comp_df["model"], comp_df["train_accuracy"])
plt.title("Training Accuracy per Model (Binary, 90/10)")
plt.ylabel("Accuracy")
plt.xticks(rotation=15, ha="right")
plt.ylim(0.0, 1.0)
plt.tight_layout()
plt.show()

```

```

# 6) Visualisasi: bar chart (TEST accuracy)
plt.figure(figsize=(7,4))
plt.bar(comp_df["model"], comp_df["test_accuracy"])
plt.title("Test Accuracy per Model (Binary, 90/10)")
plt.ylabel("Accuracy")
plt.xticks(rotation=15, ha="right")
plt.ylim(0.0, 1.0)
plt.tight_layout()
plt.show()

# 7) Cetak "pemenang" berdasarkan test accuracy (tie-breaker: macroF1)
best = comp_df.sort_values(["test_accuracy", "test_macroF1"],
                           ascending=False).iloc[0]
print("\n==== PERBANDINGAN AKHIR (Binary 90/10) ===")
print(f"Model terbaik (test accuracy): {best['model']} ")
print(f"- Test Accuracy : {best['test_accuracy']:.4f}")
print(f"- Test Macro-F1 : {best['test_macroF1']:.4f}")

```

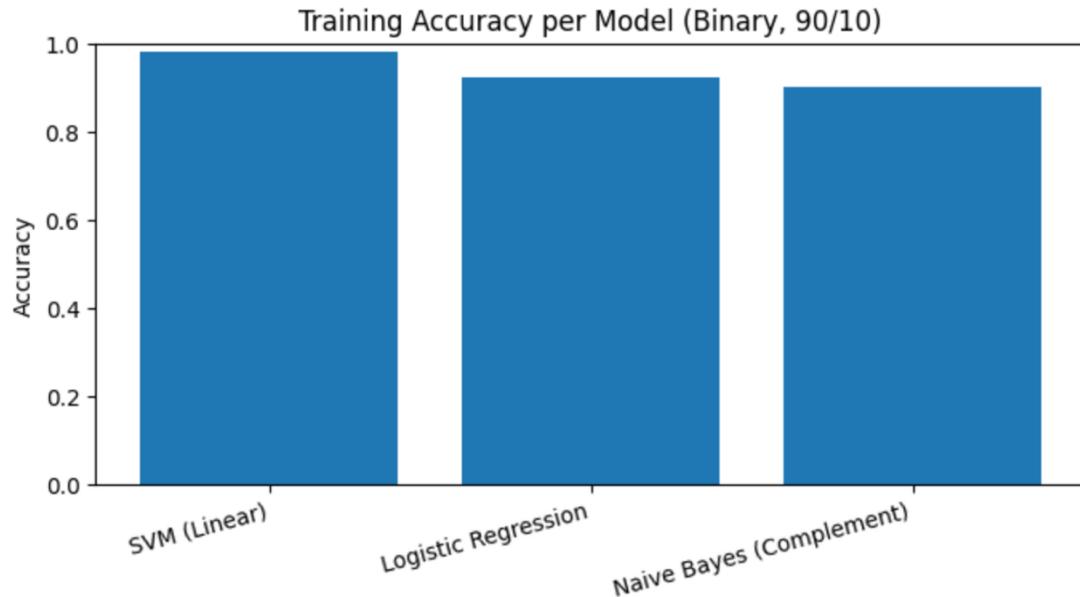
	model	train_accuracy	train_macroF1	test_accuracy	test_macroF1
0	SVM (Linear)	0.9819	0.9819	0.9105	0.9104
1	Logistic Regression	0.9241	0.9240	0.8974	0.8972
2	Naive Bayes (Complement)	0.9039	0.9036	0.8758	0.8752

Saved: model_comparison_binary.csv

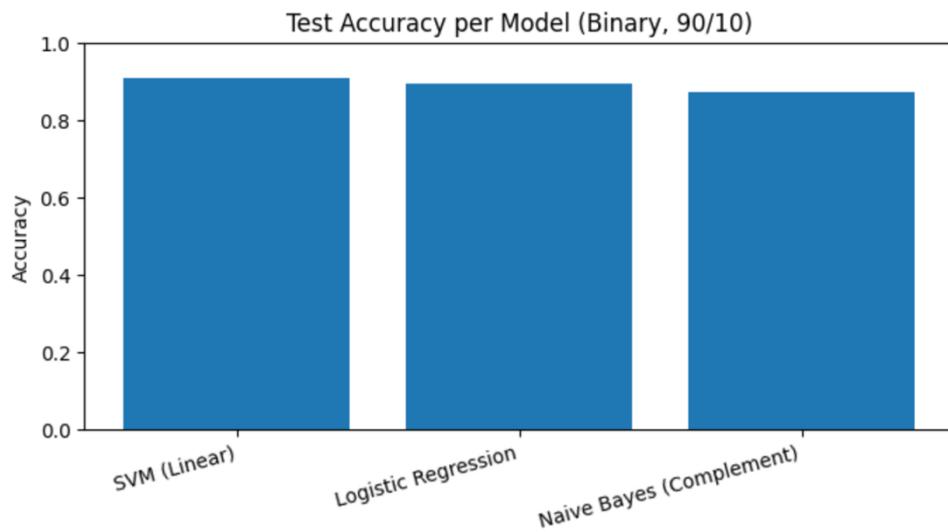
Gambar 3.20 Hasil Perbandingan Akurasi ke-3 Model Algoritma

Dari hasil tersebut, model SVM memiliki nilai train accuracy dan test accuracy tertinggi, yaitu masing-masing 0.9819 dan 0.9105, dengan nilai macro F1-score yang juga konsisten tinggi. Hal ini menunjukkan bahwa SVM memiliki kemampuan generalisasi yang baik dan paling akurat dalam membedakan kelas pada data uji. Model Logistic Regression menempati posisi kedua dengan performa yang stabil dan selisih kecil antara data latih dan uji, menandakan bahwa model tidak mengalami overfitting. Sedangkan Naive Bayes memiliki nilai akurasi dan F1-score paling rendah di antara ketiganya, namun tetap menunjukkan hasil yang cukup baik serta efisien dalam komputasi. Secara keseluruhan, SVM menjadi model terbaik berdasarkan hasil

evaluasi, karena memberikan keseimbangan antara akurasi tinggi dan kemampuan generalisasi yang baik pada data uji.



Gambar 3.21 Visualisasi Bar Chart Training Akurasi



```
==== PERBANDINGAN AKHIR (Binary 90/10) ====
Model terbaik (test accuracy): SVM (Linear)
- Test Accuracy : 0.9105
- Test Macro-F1 : 0.9104
```

Gambar 3.22 Visualisasi Bar Chart Test Akurasi

Tabel di atas menunjukkan hasil analisis perbandingan performa tiga model klasifikasi, yaitu SVM (Linear), Logistic Regression, dan Naive Bayes (Complement) berdasarkan metrik accuracy, precision, recall, dan macro F1-score.

Dari hasil tersebut, model SVM memiliki nilai accuracy, precision, recall, dan macro F1-score tertinggi, yaitu masing-masing 0.9121, 0.9125, 0.9119, dan 0.9121. Hal ini menunjukkan bahwa SVM memiliki kemampuan generalisasi yang baik dan paling akurat dalam membedakan kelas pada data uji. Model Logistic Regression menempati posisi kedua dengan performa yang stabil dan selisihnya kecil dengan model SVM, menandakan bahwa model tidak mengalami overfitting. Sedangkan Naive Bayes memiliki nilai accuracy, precision, recall, dan macro F1-score paling rendah di antara ketiganya, namun tetap menunjukkan hasil yang cukup baik serta efisien dalam komputasi. Secara keseluruhan, SVM menjadi model terbaik berdasarkan hasil evaluasi, karena memberikan keseimbangan antara akurasi tinggi dan kemampuan generalisasi yang baik pada data uji.

3.5 Docker

Pada tahapan akhir pengembangan model Machine Learning, dilakukan proses *containerization* menggunakan Docker. Tujuannya adalah memastikan model dapat dijalankan di lingkungan apa pun tanpa perlu menginstal ulang library Python, tanpa tergantung versi Python, dan tanpa masalah kompatibilitas. Dengan Docker, pipeline yang sudah dibuat (preprocessing, training model, evaluasi, hingga penyimpanan output) dapat dijalankan secara konsisten oleh siapa pun, termasuk dosen pengujii.

3.5.1 Persiapan Lingkungan Docker

Tahap awal dilakukan untuk menyiapkan lingkungan agar model Machine Learning dapat dijalankan secara portabel di dalam kontainer Docker.

a) Instalasi Docker Desktop

1. Mengunduh Docker Desktop dari situs resmi <https://www.docker.com>.
2. Melakukan instalasi sesuai sistem operasi (Windows).

3. Memastikan Docker dapat dijalankan melalui perintah docker --version di Command Prompt.

b) Aktivasi Linux Engine (WSL2)

1. Mengaktifkan WSL2 (Windows Subsystem for Linux) melalui pengaturan Docker Desktop.

2. WSL2 dipilih karena performa lebih stabil untuk menjalankan container berbasis Linux.

3. Mengecek aktivasi melalui tab Settings → General → Use the WSL 2 based engine.

c) Login ke Docker Hub

1. Membuat akun di hub.docker.com.

2. Login ke akun Docker Hub melalui Docker Desktop atau terminal dengan perintah:

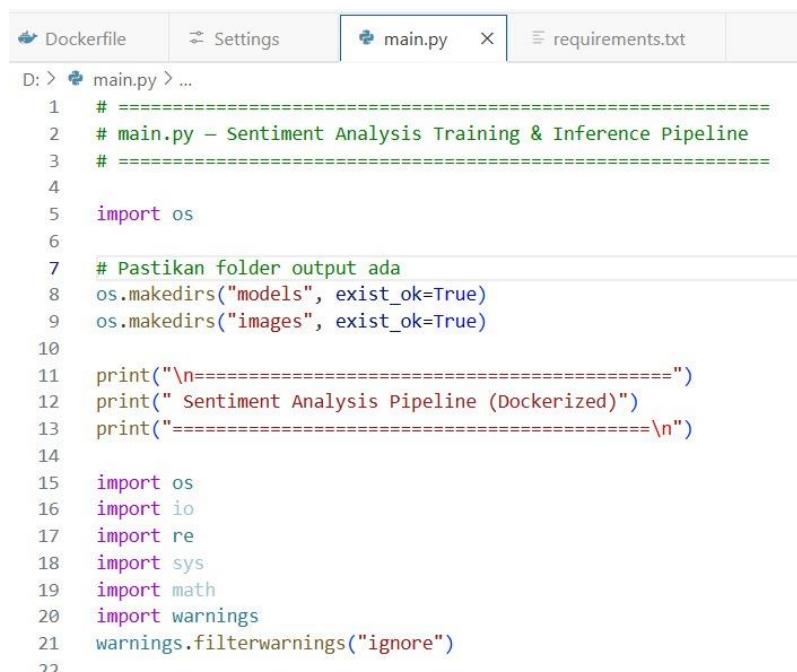
3. docker login

4. Tujuannya agar image hasil build dapat diunggah (push) ke Docker Hub.

3.5.2 Persiapan File Project

File yang dipersiapkan sebelum membuat image Docker:

1. main.py — berisi kode utama Machine Learning



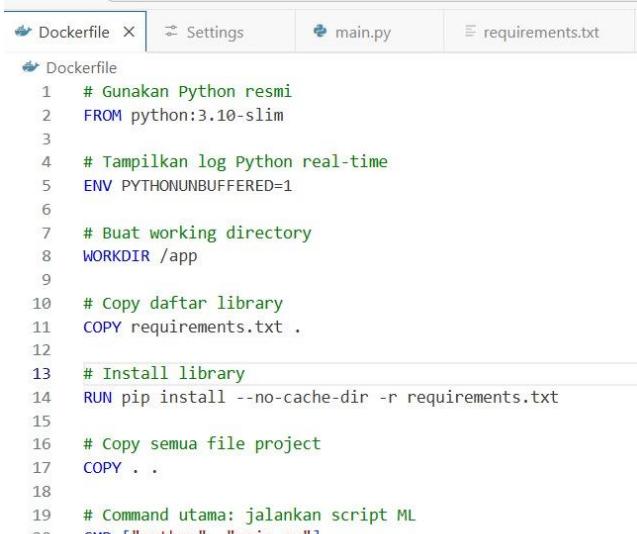
```
D: > main.py > ...
1  # =====
2  # main.py - Sentiment Analysis Training & Inference Pipeline
3  # =====
4
5  import os
6
7  # Pastikan folder output ada
8  os.makedirs("models", exist_ok=True)
9  os.makedirs("images", exist_ok=True)
10
11 print("\n====")
12 print(" Sentiment Analysis Pipeline (Dockerized)")
13 print("=====\\n")
14
15 import os
16 import io
17 import re
18 import sys
19 import math
20 import warnings
21 warnings.filterwarnings("ignore")
```

2. requirements.txt — berisi daftar library Python:



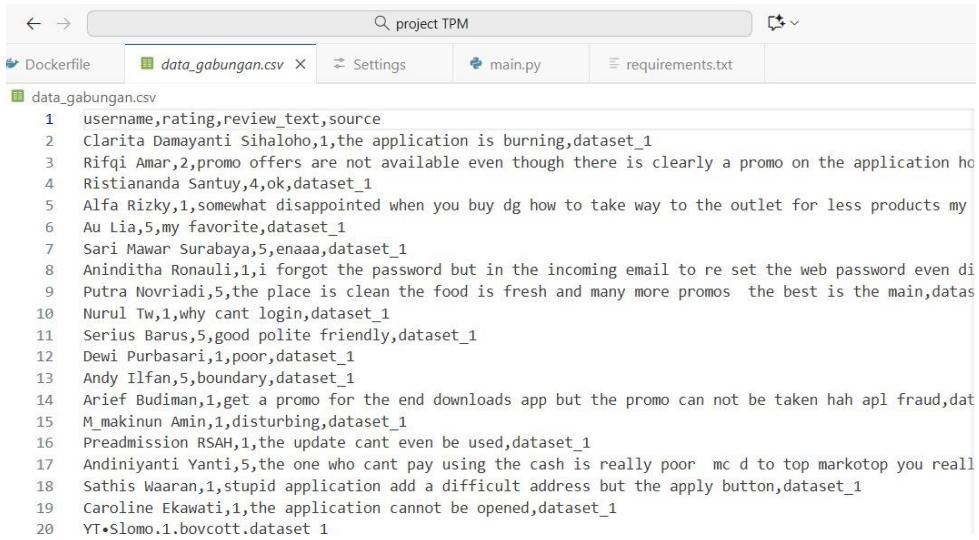
```
requirements.txt
1 pandas
2 numpy
3 scikit-learn
4 matplotlib
5 seaborn
6 nltk
7 regex
```

3. Dockerfile berisi instruksi untuk membangun image:



```
Dockerfile
1 # Gunakan Python resmi
2 FROM python:3.10-slim
3
4 # Tampilkan log Python real-time
5 ENV PYTHONUNBUFFERED=1
6
7 # Buat working directory
8 WORKDIR /app
9
10 # Copy daftar library
11 COPY requirements.txt .
12
13 # Install library
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 # Copy semua file project
17 COPY . .
18
19 # Command utama: jalankan script ML
20 CMD ["python", "main.py"]
```

4. **data/** — folder berisi dataset, misalnya `data_gabungan.csv`.



```
1 username,rating,review_text,source
2 Clarita Damayanti Sihaloho,1,the application is burning,dataset_1
3 Rifqi Amar,2,promo offers are not available even though there is clearly a promo on the application ho
4 Ristiananda Santuy,4,ok,dataset_1
5 Alfa Rizky,1,somewhat disappointed when you buy dg how to take way to the outlet for less products my
6 Au Lia,5,my favorite,dataset_1
7 Sari Mawar Surabaya,5,enaaa,dataset_1
8 Aninditha Ronauli,1,i forgot the password but in the incoming email to re set the web password even di
9 Putra Novriadi,5,the place is clean the food is fresh and many more promos the best is the main,datas
10 Nurul Tw,1,why cant login,dataset_1
11 Serius Barus,5,good polite friendly,dataset_1
12 Dewi Purbasari,1,poor,dataset_1
13 Andy Ilfan,5,boundary,dataset_1
14 Arief Budiman,1,get a promo for the end downloads app but the promo can not be taken hah apl fraud,dat
15 M_makinun Amin,1,disturbing,dataset_1
16 Preadmission RSAH,1,the update cant even be used,dataset_1
17 Andiniyanti Yanti,5,the one who cant pay using the cash is really poor mc d to top markotop you reall
18 Sathis Waaran,1,stupid application add a difficult address but the apply button,dataset_1
19 Caroline Ekawati,1,the application cannot be opened,dataset_1
20 YT•Slomo.i.bovcott.dataset 1
```

a) Struktur Folder Project

```
projectTPM
├── Dockerfile
├── main.py
└── requirements.txt
└── data/
    └── data_gabungan.csv
```

b) Pengujian Awal

1. Pastikan semua dependensi dapat dijalankan di lokal sebelum proses build image Docker.
2. Jalankan python `main.py` untuk memastikan script tidak error.

3.5.3 Build Docker Image

Masuk ke folder proyek lalu jalankan:

```
PS D:\Kuliah\Semester 5\Teknik Pengembangan Model\project TPM> docker build --no-cache -t sentiment-mcd3:latest .
[+] Building 126.0s (11/11) FINISHED
   docker:desktop-linux
=> [internal] load build definition from Dockerfile
      0.2s
=> => transferring dockerfile: 414B
      0.0s
=> [internal] load metadata for docker.io/library/python:3.10-slim
      4.9s
=> [auth] library/python:pull token for registry-1.docker.io
      0.0s
=> [internal] load .dockerignore
      0.1s
=> => transferring context: 2B
      0.1s
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:975a1e200a16719060d391eea4ac66ee067d709cc22a32f4ca4737731ea36c0
```

Proses Ini membangun image dengan nama **sentiment-mcd3**.

3.5.4 Menjalankan Kontainer (Dengan Volume Mounting)

Sebelum menjalankan pipeline di dalam Docker, dilakukan eksekusi perintah berikut di terminal untuk membuat kontainer lokal yang akan menjalankan proses Machine Learning dan menyimpan hasilnya ke folder *output* di komputer local.

```
PS D:\Kuliah\Semester 5\Teknik Pengembangan Model\project TPM> docker run --name sentiment-mcd3-container -v "D:/Kuliah/Semester 5/Teknik Pengembangan Model/project TPM/output:/app/output" sentiment-mcd3:latest
```

Setelah memastikan pipeline berjalan dengan baik, langkah berikutnya adalah menjalankan kontainer dengan sistem *volume mounting* yang lebih lengkap, mencakup tiga direktori output:

```
docker run --rm ^
-v "D:\docker_hasil:/app/images" ^
-v "D:\docker_hasil:/app/models" ^
-v "D:\docker_hasil:/app/output" ^
katrina77/sentiment-mcd3:latest
```

images, models, dan output.

Gunakan perintah berikut di Command Prompt (Windows):

Dengan konfigurasi ini, seluruh hasil analisis seperti grafik (.png), model Machine Learning (.pkl), dan file hasil prediksi (.csv) akan otomatis muncul di folder D:\docker_hasil di komputer lokal.

3.5.5 Isi Kontainer (Apa yang Dijalankan)

Ketika kontainer dijalankan, main.py menjalankan pipeline:

1. Explorasi data
Menampilkan ringkasan dataset, missing value, dan contoh data.
2. Cleaning & Preprocessing
 - a) Case folding
 - b) Pembersihan URL, mention, special characters
 - c) Filtering teks pendek

- d) Mapping sentimen (neg, neu, pos)
3. Training 3 Model
 - a) SVM (tuned)
 - b) Logistic Regression (4 varian)
 - c) Naive Bayes (binary & multiclass)
 4. Evaluasi
 - a) Confusion Matrix
 - b) ROC Curve
 - c) Perbandingan performa model

5. Evaluasi
 - a) Confusion Matrix
 - b) ROC Curve
 - c) Perbandingan performa model

6. Auto Inference

Karena Docker non-interaktif, pipeline melakukan inference otomatis pada teks: "Ayamnya enak banget, tapi aplikasinya error pas mau bayar". Output tersimpan ke: /app/output/auto_inference_results.csv

7. Penyimpanan Model

- a) svm_bin_final.pkl
- b) logreg_bin_final.pkl
- c) nb_bin_final.pkl
- d) all_models_bin.pkl

3.5.6 File Output yang Dihasilkan

Setelah kontainer berjalan, akan muncul di folder lokal:

1. Gambar (*.png)
 - a) Confusion matrix SVM/LR/NB
 - b) ROC Curve
 - c) Grafik perbandingan model

2. Model (*.pkl)

- a) Model final SVM
- b) Model final Logistic Regression
- c) Model final Naive Bayes
- d) Bundle: all_models_bin.pkl

3. CSV

- a) Hasil prediksi tiap model
- b) Auto inference
- c) Perbandingan model (model_comparison_binary.csv)

3.5.7 Upload Image ke Docker Hub

Tahap terakhir setelah image selesai dibuat dan diuji adalah mengunggah (*push*) image tersebut ke Docker Hub agar dapat diakses dan dijalankan di perangkat mana pun tanpa perlu konfigurasi tambahan.

Langkah-langkahnya sebagai berikut:

1. Buat Repository di Docker Hub
Masuk ke akun Docker Hub, kemudian buat repository baru dengan nama sentiment-mcd3. Repository ini akan menjadi tempat penyimpanan image hasil build.
2. Tambahkan Tag ke Image
Setelah repository dibuat, salin nama repository beserta tag-nya dari perintah docker push.
3. Push Image ke Docker Hub

Setelah penandaan selesai, jalankan perintah berikut untuk mengunggah image:

```
C:\Users\iapia>docker push katarina77/sentiment-mcd3:latest
The push refers to repository [docker.io/katarina77/sentiment-mcd3]
bfe20f996469: Pushed
4bcde41c77aa: Pushed
61eefe12bf76: Mounted from katarina77/sentiment-mcd2
a09dc670095e: Mounted from katarina77/sentiment-mcd2
d7ecded7702a: Mounted from katarina77/sentiment-mcd2
03b7f5f69b04: Mounted from katarina77/sentiment-mcd2
cfba6d6670cc: Mounted from katarina77/sentiment-mcd2
de0b5892fd04: Pushed
8a309d7f94dc: Pushed
latest: digest: sha256:6c43c4215e606b682c60508f5a61f0d0c42e4c6f81161bc2d1d1a8c4f965
c8b0 size: 856
```

Proses ini akan mengirim seluruh layer image ke repository Docker Hub.

4. Verifikasi di Docker Hub

Setelah proses push selesai, buka akun Docker Hub dan pastikan image sentiment-mcd3 sudah muncul di repository dengan tag latest. Repository Anda akan dapat diakses melalui URL <https://hub.docker.com/repository/docker/katarina77/sentiment-mcd3/tags>.

BAB V

KESIMPULAN

Berdasarkan pengujian komparatif yang dilakukan pada *dataset* ulasan pelanggan McDonald's, penelitian ini menyimpulkan bahwa algoritma Support Vector Machine (SVM) adalah model klasifikasi paling efektif untuk analisis sentimen berbasis teks.

1. **Model Terbaik:** SVM mencapai akurasi tertinggi sebesar 91,21%, mengungguli Logistic Regression (88,08%) dan Naive Bayes (87,42%). Kinerja superior ini dikaitkan dengan kemampuan bawaan SVM untuk memproses fitur berdimensi tinggi yang dihasilkan dari ekstraksi fitur TF-IDF, menjadikannya pilihan ideal untuk klasifikasi teks.
2. **Akurasi dan Keseimbangan Metrik:** SVM juga menunjukkan nilai *Macro F1-Score* yang sangat seimbang (0.9121), menjamin bahwa model memiliki tingkat presisi yang tinggi dalam mengidentifikasi ulasan positif/negatif sambil meminimalkan ulasan yang terlewatkan.
3. **Deployment Model:** Model SVM yang unggul ini dapat diintegrasikan ke dalam sistem operasional bisnis melalui *containerization* Docker. Docker menjamin bahwa model dapat berjalan secara konsisten, portabel, dan dapat diskalakan sebagai layanan API di berbagai lingkungan produksi, memastikan hasil analisis sentimen dapat dimanfaatkan secara *real-time* oleh manajemen.

Selain itu, hasil penelitian ini menunjukkan bahwa pemilihan algoritma yang tepat memiliki pengaruh signifikan terhadap kualitas analisis sentimen. Dengan memanfaatkan pendekatan berbasis machine learning dan pipeline pemrosesan teks yang sistematis, penelitian ini berhasil menghasilkan model yang tidak hanya akurat, tetapi juga siap untuk diimplementasikan dalam konteks bisnis nyata. Hal ini membuktikan bahwa penerapan analisis sentimen berbasis SVM dapat menjadi dasar pengambilan keputusan strategis yang lebih cepat, tepat, dan berbasis data.

DAFTAR PUSTAKA

- Arsi, P., & Waluyo, R. (2021). *Analisis Sentimen Wacana Pemindahan Ibu Kota Indonesia Menggunakan Algoritma Support Vector Machine (SVM)*. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 8(1), 147–156. <https://doi.org/10.25126/jtiik.202183944>
- Averina, A., Hadi, H., & Siswantoro, J. (2022). *Analisis Sentimen Multi-Kelas untuk Film Berbasis Teks Ulasan Menggunakan Model Regresi Logistik*. Jurnal Teknika, 11(2), 123–128. <https://doi.org/10.34148/teknika.v11i2.461>
- Safira, A., & Hasan, F. N. (2023). *Analisis Sentimen Masyarakat terhadap Paylater Menggunakan Metode Naive Bayes Classifier*. ZONAsi: Jurnal Sistem Informasi, 5(1), 59–67. <https://doi.org/10.31849/zn.v5i1.12856>
- Hasan, F. N., & Dwijayanti, M. (2021). *Analisis Sentimen Ulasan Pelanggan terhadap Layanan Grab Indonesia Menggunakan Multinomial Naïve Bayes Classifier*. Jurnal Linguistik Komputasional (JLK), 4(2), 52–59. <https://doi.org/10.22236/teknoka.v6i1.441>
- Hasan, F. N. (2022). *Analisis Sentimen pada Ulasan Pelanggan Menggunakan Metode Naïve Bayes Classifier (Studi Kasus: Grab Indonesia)*. Prosiding Seminar Nasional Teknoka, 6(1). <https://doi.org/10.22236/teknoka.v6i1.441>