

INFORMASI PROYEK

Judul Proyek	: Analisis Kepuasan Pengguna Terhadap Resep Masakan Berdasarkan Ulasan Teks Menggunakan Machine Learning dan LSTM Deep Learning
Nama Mahasiswa	: Rara Aliviana gumaranti
NIM	: 234311050
Program Studi	: Teknologi Rekayasa Perangkat Lunak
Mata Kuliah	: Praktik Data Science
Dosen Pengampu	: Gus Nanang Syaifuddiin
Tahun Akademik	: Tahun Akademik 2025/2026
Link GitHub Repository	: https://github.com/RaraAliviana/UAS_DataScience_234311050.git
Link Video Pembahasan	: https://youtu.be/fH6jByask1o

1. LEARNING OUTCOMES

Pada proyek ini, mahasiswa diharapkan dapat:

1. Memahami konteks masalah dan merumuskan problem statement secara jelas
2. Melakukan analisis dan eksplorasi data (EDA) secara komprehensif **(OPSIONAL)**
3. Melakukan data preparation yang sesuai dengan karakteristik dataset
4. Mengembangkan tiga model machine learning yang terdiri dari **(WAJIB)**:
 - Model baseline
 - Model machine learning / advanced
 - Model deep learning (WAJIB)
5. Menggunakan metrik evaluasi yang relevan dengan jenis tugas ML
6. Melaporkan hasil eksperimen secara ilmiah dan sistematis
7. Mengunggah seluruh kode proyek ke GitHub **(WAJIB)**
8. Menerapkan prinsip software engineering dalam pengembangan proyek

2. PROJECT OVERVIEW

2.1. Latar Belakang

Di era digital, ulasan teks pengguna merupakan instrumen vital untuk mengukur kepuasan pelanggan. Namun, volume data yang besar dan tidak terstruktur memerlukan sistem klasifikasi otomatis yang efisien untuk mengekstrak opini publik secara akurat (Xiao, 2025). Tantangan utama dalam analisis ini adalah bagaimana mengubah teks mentah menjadi fitur numerik yang relevan, salah satunya menggunakan metode Term Frequency-Inverse Document Frequency (TF-IDF) yang terbukti efektif bila dikombinasikan dengan algoritma Machine Learning seperti Random Forest (Rizkiyanto et al., 2024).

Meskipun model tradisional seperti Random Forest sangat tangguh, pendekatan Deep Learning seperti Long Short-Term Memory (LSTM) menawarkan keunggulan dalam memahami konteks dan urutan kata pada kalimat yang panjang (Afif & Nugroho, n.d.; Xiao, 2025). Penelitian menunjukkan bahwa kemampuan sekuensial LSTM seringkali menghasilkan akurasi yang lebih tinggi dibandingkan model konvensional dalam menangani nuansa bahasa ulasan aplikasi.

Oleh karena itu, project ini bertujuan untuk membandingkan performa model Logistic Regression, Random Forest, dan LSTM dalam mengklasifikasikan sentimen ulasan resep masakan untuk mendapatkan model terbaik yang mampu mengolah umpan balik pengguna secara otomatis.

Referensi :

- Afif, R., & Nugroho, K. (n.d.). *Analisis Sentimen dan Prediksi Ulasan Pada Aplikasi Info BMKG*. 15(2).
- Rizkiyanto, M. D., Purbolaksono, M. D., & Astuti, W. (2024). Sentiment Analysis Classification on PLN Mobile Application Reviews using Random Forest Method and TF-IDF Feature

Extraction. *INTEK: Jurnal Penelitian*, 11(1), 37–43.

<https://doi.org/10.31963/intek.v11i1.4774>

Xiao, S. (2025). *Comparative Analysis of Machine Learning Models for Text Emotion Classification*.

3. BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING

3.1. Problem Statements

Berdasarkan latar belakang yang telah dijelaskan, permasalahan utama dalam project ini dapat di rumuskan sebagai berikut:

1. Ulasan teks pengguna terhadap resep masakan bersifat tidak terstruktur sehingga sulit dianalisis secara manual dalam jumlah besar.
2. Dataset ulasan mengandung noise berupa variasi bahasa, tanda baca, dan kata tidak baku yang memerlukan proses preprocessing yang tepat.
3. Model baseline sederhana sering kali belum mampu menangkap konteks dan makna mendalam dari teks ulasan.
4. Diperlukan model deep learning yang mampu mempelajari pola sekuensial pada teks untuk meningkatkan akurasi klasifikasi kepuasan pengguna.

3.2. Goals

Tujuan dari proyek ini dirancang secara spesifik dan terukur sebagai berikut:

1. Membangun sistem klasifikasi teks untuk memprediksi tingkat kepuasan pengguna terhadap resep masakan berdasarkan ulasan teks.
2. Membandingkan performa tiga pendekatan model, yaitu baseline model, advanced machine learning model, dan deep learning model.
3. Mencapai nilai akurasi dan F1-Score di atas 80% pada data uji.
4. Menentukan model terbaik berdasarkan metrik evaluasi yang relevan dan hasil analisis performa.
5. Menghasilkan pipeline Machine Learning yang dapat direproduksi dan terdokumentasi dengan baik.

3.3. Solutions Approach

1. Baseline Model

Model baseline yang digunakan dalam proyek ini adalah Logistic Regression dengan representasi fitur menggunakan TF-IDF. Logistic Regression dipilih karena merupakan model klasifikasi linear yang

sederhana, cepat dilatih, serta sering digunakan sebagai pembanding awal dalam tugas klasifikasi teks.

2. Advanced / Machine Learning Model

Model advanced yang digunakan adalah **Random Forest Classifier**.

Random Forest merupakan metode ensemble yang menggabungkan banyak decision tree untuk meningkatkan performa dan stabilitas prediksi. Model ini dipilih karena mampu menangkap hubungan non-linear antar fitur serta relatif robust terhadap noise pada data teks hasil ekstraksi TF-IDF.

3. Deep Learning Model (WAJIB)

Model deep learning yang digunakan adalah Long Short-Term Memory (LSTM), yang termasuk dalam arsitektur Recurrent Neural Network (RNN). LSTM dipilih karena kemampuannya dalam memproses data sekuensial dan memahami konteks kata dalam kalimat secara berurutan. Model ini sangat sesuai untuk tugas klasifikasi teks, khususnya dalam analisis sentimen dan kepuasan pengguna, di mana makna suatu kata sangat dipengaruhi oleh konteks kata sebelumnya. Model LSTM akan dilatih minimal 10 epoch, dilengkapi dengan visualisasi loss dan accuracy, serta dievaluasi menggunakan data uji untuk memastikan performa dan generalisasi model.

4. DATA UNDERSTANDING

4.1. Informasi Dataset

4.1.1. Sumber Dataset

Dataset yang digunakan pada proyek ini merupakan dataset ulasan resep masakan yang diperoleh dari proyek mini sebelumnya dari UC Irvine Machine Learning Repository, yang berisi ulasan teks pengguna beserta label kepuasan. Dataset disimpan dalam format CSV dan digunakan kembali untuk keperluan analisis pada proyek UAS ini. Berikut Link Dataset dari Recipe reviews and user feedback:

<https://archive.ics.uci.edu/dataset/911/recipe+reviews+and+user+feedback+dataset>

4.1.2. Deskripsi Dataset:

Jumlah baris (rows)	:	18.182 data
Jumlah kolom (features)	:	15 kolom
Tipe data	:	Text (NLP) dan Tabular
Ukuran dataset	:	5.80 MB
Format file	:	CSV

4.2. Deskripsi Fitur

Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
Unnamed: 0	Integer	Index bawaan file CSV	0, 1, 2
recipe_number	Integer	Nomor urut internal resep	38, 102
recipe_code	Integer	Kode unik resep	40478
recipe_name	String	Nama resep makanan	Cheesecake Brownies
comment_id	Integer	ID unik komentar	894512
user_id	Integer	ID unik pengguna	567839
user_name	String	Nama pengguna	JohnD
user_reputation	Integer	Tingkat reputasi pengguna	1, 5, 10
created_at	Integer	Waktu komentar (Unix timestamp)	1.51E+08
reply_count	Integer	Jumlah balasan komentar	0, 2
thumbs_up	Integer	Jumlah like pada komentar	3, 15

thumbs_down	Integer	Jumlah dislike pada komentar	0, 1
stars	Integer	Rating bintang (0–5)	3, 4, 5
best_score	Integer	Skor popularitas komentar	10, 25
text	String	Isi ulasan pengguna	“Great recipe!”

4.3. Kondisi Data

4.3.1. Missing Value

Hasil eksplorasi data menunjukkan bahwa dataset ini tidak memiliki missing values yang signifikan pada kolom utama, termasuk kolom teks ulasan dan rating (stars). Dengan demikian, dataset berada dalam kondisi yang relatif bersih dan siap untuk diproses lebih lanjut.

4.3.2. Duplicate Data

Tidak ditemukan duplikasi pada kolom comment_id, yang menandakan bahwa setiap ulasan bersifat unik. Namun, satu resep dapat memiliki banyak ulasan, sehingga hubungan antara resep dan ulasan bersifat one-to-many.

4.3.3. Outliers

Outliers ditemukan pada beberapa fitur numerik, terutama:

1. thumbs_up
2. thumbs_down
3. best_score

Beberapa komentar memiliki jumlah *thumbs up* yang jauh lebih tinggi dibandingkan mayoritas data. Namun, nilai tersebut masih dianggap valid karena merepresentasikan tingkat popularitas komentar tertentu.

4.3.4. Imbalanced Data

Distribusi rating (stars) menunjukkan ketidakseimbangan kelas, di mana:

1. Rating 4 dan 5 mendominasi dataset
2. Rating rendah (0–2) jumlahnya relatif lebih sedikit

4.3.5. Noise pada Data

Beberapa permasalahan noise yang ditemukan pada dataset antara lain:

1. Karakter HTML entities (misalnya ") pada teks ulasan
2. Variasi panjang teks ulasan
3. Adanya komentar tanpa rating eksplisit (rating = 0)

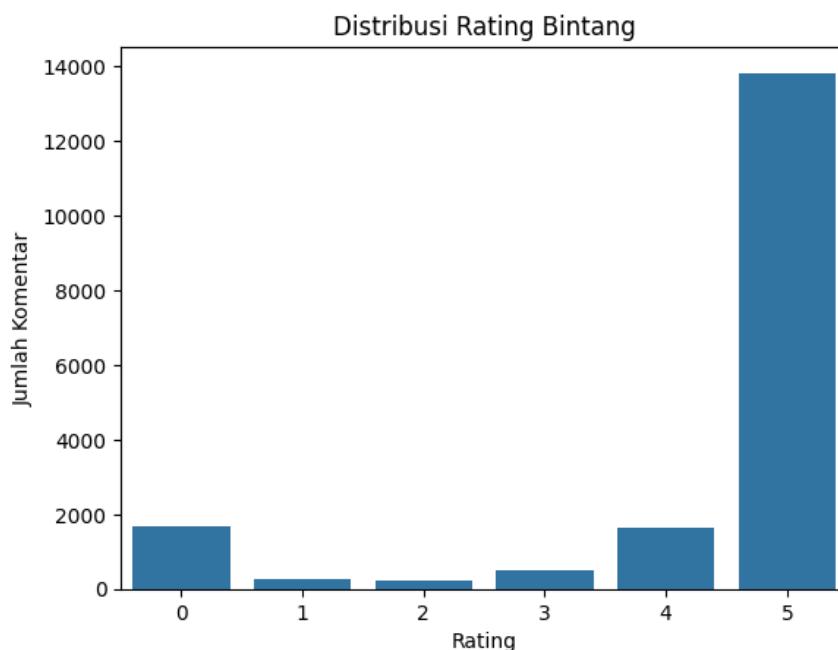
4.3.6. Data Quality Issues

Secara umum, dataset memiliki kualitas data yang baik. Namun, beberapa isu yang perlu ditangani adalah:

1. Kolom tidak relevan (Unnamed: 0)
2. Teks ulasan yang mengandung karakter khusus
3. Ketidakseimbangan kelas rating

4.4. Exploratory Data Analysis (EDA)

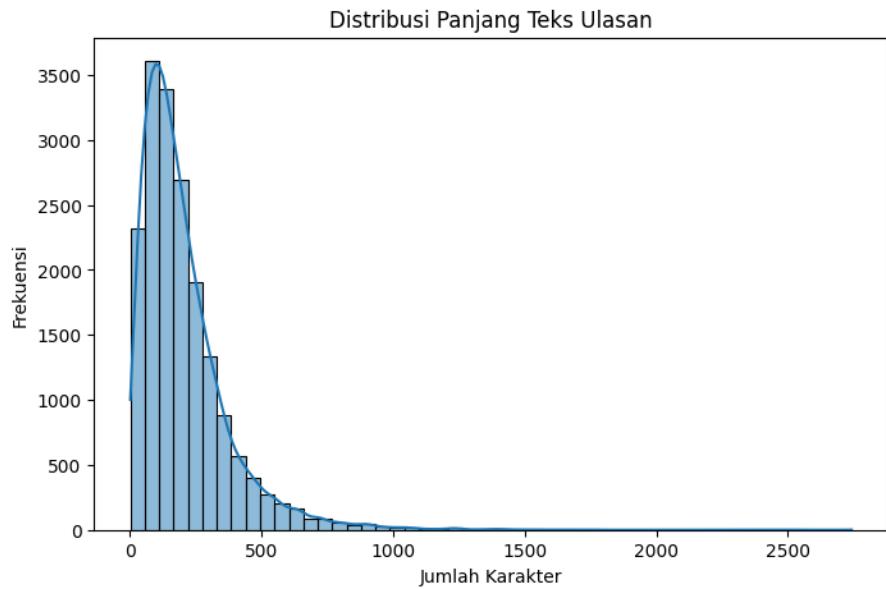
4.4.1. Visualisasi 1 : Distribusi Rating Bintang



Insight:

Distribusi rating menunjukkan bahwa mayoritas pengguna memberikan rating tinggi (4 dan 5 bintang). Hal ini mengindikasikan bahwa sebagian besar resep mendapatkan respon positif dari pengguna.

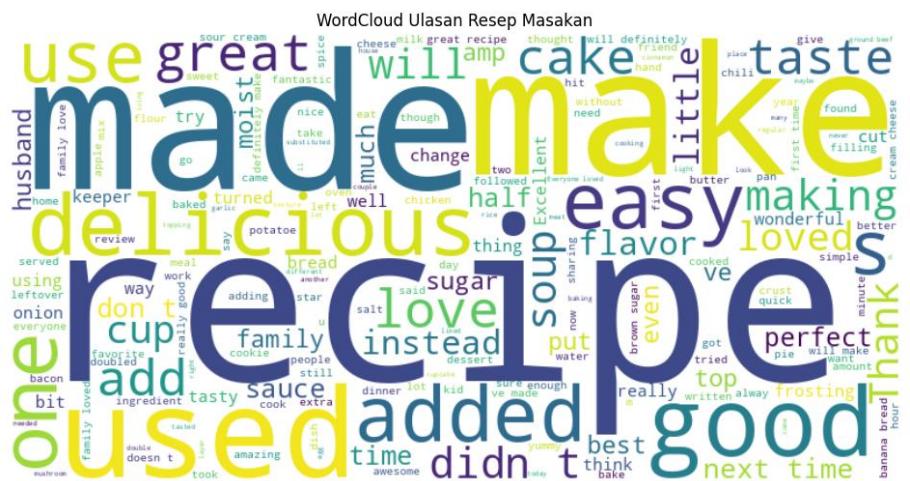
4.4.2. Visualisasi 2 : Panjang Teks Ulasan



Insight:

Sebagian besar ulasan memiliki panjang teks yang relatif pendek hingga menengah. Namun, terdapat sejumlah ulasan dengan teks panjang yang berisi pengalaman detail pengguna dalam memasak resep.

4.4.3. Visualisasi 3 : WordCloud Ulasan Pengguna



Insight:

Hasil WordCloud menunjukkan bahwa kata-kata yang paling dominan dalam ulasan pengguna umumnya berkaitan dengan pengalaman positif dalam mencoba resep, seperti kata “*good*”, “*great*”, “*easy*”, dan “*delicious*”. Hal ini mengindikasikan

bahwa sebagian besar ulasan memiliki sentimen positif terhadap resep yang diberikan. Selain itu, munculnya kata-kata yang berhubungan dengan proses memasak dan bahan makanan menunjukkan bahwa pengguna tidak hanya memberikan penilaian singkat, tetapi juga membagikan pengalaman dan modifikasi resep secara detail.

5. DATA PREPARATION

Tahap data preparation bertujuan untuk memastikan data yang digunakan berada dalam kondisi optimal sebelum dilakukan proses pemodelan menggunakan metode machine learning dan deep learning. Pada penelitian ini, data preparation meliputi proses pembersihan data, pembentukan label sentimen, ekstraksi dan transformasi fitur teks, pembagian data, serta penanganan ketidakseimbangan kelas.

5.1. Data Cleaning

5.1.1. Handling Missing Value

```
#@title Data Cleaning  
  
# Menghapus data dengan text kosong  
df = df.dropna(subset=['text'])  
  
print("Jumlah data setelah cleaning:", df.shape[0])  
  
Jumlah data setelah cleaning: 18180
```

Pada tahap awal, dilakukan pemeriksaan terhadap nilai kosong (missing values) pada dataset, khususnya pada kolom text yang berisi ulasan pengguna. Data dengan teks kosong tidak dapat digunakan dalam analisis berbasis teks, sehingga dilakukan penghapusan terhadap baris yang memiliki nilai kosong pada kolom tersebut.

Setelah proses penghapusan data dengan teks kosong, jumlah data yang digunakan dalam penelitian ini menjadi 18.180 data. Jumlah ini masih dianggap mencukupi untuk proses pelatihan dan evaluasi model.

5.1.2. Removing Duplicates

```
▶ df = df.drop(columns=['Unnamed: 0'])
```

Pemeriksaan terhadap data duplikat dilakukan menggunakan kolom comment_id sebagai identifier unik. Hasil analisis menunjukkan bahwa tidak terdapat data duplikat pada dataset. Oleh karena itu, tidak dilakukan penghapusan data duplikat lebih lanjut.

Selain itu, kolom Unnamed: 0 dihapus karena merupakan index bawaan file CSV dan tidak memiliki kontribusi terhadap proses analisis maupun pemodelan.

5.2. Feature Engineering

5.2.1. Labeling sentimen (stars → label)

```
❶ #@title Labeling Sentimen dari Rating (stars)

#Dataset tidak memiliki label sentimen eksplisit, sehingga dilakukan konversi rating bintang menjadi label sentimen.

# Labeling sentimen dari stars
def label_sentiment(star):
    if star >= 4:
        return 1 # Positif
    else:
        return 0 # Negatif

df['sentiment'] = df['stars'].apply(label_sentiment)
```

Dataset yang digunakan tidak memiliki label sentimen secara eksplisit. Oleh karena itu, dilakukan konversi dari rating bintang (stars) menjadi label sentimen.

Aturan pelabelan sentimen yang digunakan adalah sebagai berikut:

1. Rating ≥ 4 dikategorikan sebagai sentimen positif (label 1)
2. Rating < 4 dikategorikan sebagai sentimen negatif (label 0)

Pendekatan ini digunakan karena rating bintang dianggap merepresentasikan tingkat kepuasan pengguna terhadap layanan atau produk yang diulas.

5.2.2. TF-IDF

```
❷ #@title Feature Extraction – TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(
    max_features=10000,
    ngram_range=(1,2),
    min_df=5,
    max_df=0.9
)

X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

TF-IDF digunakan untuk mengukur tingkat kepentingan suatu kata dalam dokumen relatif terhadap keseluruhan dokumen

pada dataset. Parameter yang digunakan dalam proses TF-IDF antara lain:

TF-IDF dikonfigurasikan dengan:

1. Maksimum fitur sebanyak 10.000
2. Rentang n-gram unigram dan bigram
3. Minimum kemunculan kata: 5 dokumen
4. Maksimum proporsi dokumen: 90%

5.3.Data Transformation

5.3.1. Text cleaning (lowercase, stopword, dll)

```
▶ #@title Text Preprocessing

import re
import string

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\s+', ' ', text).strip()
    return text

df['clean_text'] = df['text'].astype(str).apply(clean_text)
```

Tahap transformasi teks dilakukan untuk mengurangi noise dan meningkatkan kualitas data. Proses text cleaning meliputi:

1. Mengubah seluruh teks menjadi huruf kecil (*lowercasing*)
2. Menghapus tanda baca dan angka
3. Menghapus kata-kata umum (*stopwords*) dalam bahasa Inggris
4. Melakukan lemmatization untuk mengubah kata ke bentuk dasarnya

Hasil dari proses ini disimpan dalam kolom `clean_text`, yang digunakan sebagai input pada tahap ekstraksi fitur dan pemodelan.

5.3.2. Tokenization & Padding

```
❶ #@title Tokenization & Padding (Untuk LSTM)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_words = 8000
max_len = 80

tokenizer = Tokenizer(num_words=max_words, oov_token=<OOV>)
tokenizer.fit_on_texts(X_train)

X_train_seq = pad_sequences(
    tokenizer.texts_to_sequences(X_train),
    maxlen=max_len,
    padding='post'
)

X_test_seq = pad_sequences(
    tokenizer.texts_to_sequences(X_test),
    maxlen=max_len,
    padding='post'
)

vocab_size = min(max_words, len(tokenizer.word_index) + 1)
```

Untuk model deep learning berbasis LSTM, data teks tidak menggunakan TF-IDF, melainkan direpresentasikan dalam bentuk urutan token. Pada tahap ini dilakukan:

1. Tokenisasi teks menggunakan tokenizer dengan batas maksimum jumlah kata
2. Penanganan kata yang tidak dikenal menggunakan token khusus (out-of-vocabulary token)
3. Penyesuaian panjang urutan (padding) agar seluruh data memiliki panjang yang sama

5.4. Data Splitting

```
❶ #@title Data Splitting
from sklearn.model_selection import train_test_split

X = df['clean_text']
y = df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)
```

Setelah proses preprocessing, data dibagi menjadi data latih (training set) dan data uji (test set). Strategi pembagian data adalah sebagai berikut:

1. **Training set:** 80%
2. **Test set:** 20%

Proses pembagian data menggunakan metode *stratified splitting* untuk menjaga proporsi kelas sentimen positif dan negatif tetap seimbang pada data latih dan data uji. Selain itu, nilai random_state ditetapkan untuk memastikan hasil pembagian data bersifat reproducible.

5.5. Data Balancing

```
▶  #@title Class Weighting (Imbalanced Data)

from sklearn.utils.class_weight import compute_class_weight
import numpy as np

classes = np.unique(y_train)
weights = compute_class_weight(
    class_weight='balanced',
    classes=classes,
    y=y_train
)

class_weights = dict(zip(classes, weights))
class_weights

... {np.int64(0): np.float64(3.3697868396663577),
np.int64(1): np.float64(0.5871144840949459)}
```

Berdasarkan distribusi data sentimen, ditemukan adanya ketidakseimbangan antara kelas sentimen positif dan negatif. Untuk mengatasi hal tersebut, digunakan metode class weighting pada data latih.

Bobot kelas dihitung secara otomatis berdasarkan proporsi masing-masing kelas, sehingga model tidak bias terhadap kelas mayoritas. Dengan pendekatan ini, kesalahan pada kelas minoritas akan diberikan bobot yang lebih besar saat proses pelatihan model, khususnya pada model LSTM.

5.6. Ringkasan Data Preparation

Secara keseluruhan, tahap data preparation meliputi pembersihan data, pelabelan sentimen, preprocessing teks, pembagian data, ekstraksi fitur, tokenisasi, serta penanganan ketidakseimbangan data. Tahapan ini bertujuan untuk menghasilkan data yang bersih, terstruktur, dan siap digunakan pada proses pelatihan serta evaluasi model klasifikasi sentimen pada bab selanjutnya.

5.6.1. Data Cleaning

a. Apa yang dilakukan:

Menghapus data yang memiliki nilai kosong pada kolom text serta menghapus kolom Unnamed: 0 yang tidak memiliki informasi relevan.

b. Mengapa penting:

Data dengan teks kosong tidak dapat diproses dalam analisis berbasis teks dan dapat menyebabkan error pada tahap preprocessing dan feature extraction. Selain itu, kolom yang tidak relevan dapat menambah noise dan memperbesar kompleksitas data.

c. Bagaimana implementasinya:

Data dengan nilai kosong pada kolom text dihapus menggunakan fungsi dropna(), sedangkan kolom Unnamed: 0 dihapus menggunakan fungsi drop().

5.6.2. Pelabelan Sentimen

a. Apa yang dilakukan:

Mengonversi rating bintang (stars) menjadi label sentimen positif dan negatif.

b. Mengapa penting:

Dataset tidak menyediakan label sentimen eksplisit, sehingga pelabelan diperlukan agar data dapat digunakan sebagai permasalahan klasifikasi terawasi (supervised learning).

c. Bagaimana implementasinya:

Sebuah fungsi dibuat untuk mengonversi nilai rating bintang menjadi label sentimen, di mana rating ≥ 4 diberi label positif (1) dan rating < 4 diberi label negatif (0). Fungsi ini kemudian diaplikasikan ke seluruh data.

5.6.3. Text Cleaning

a. Apa yang dilakukan:

Melakukan pembersihan teks meliputi lowercasing, penghapusan tanda baca dan angka, penghapusan stopwords, serta lemmatization.

b. Mengapa penting:

Text cleaning bertujuan untuk mengurangi noise, menyamakan bentuk kata, dan meningkatkan kualitas representasi teks sehingga model dapat belajar pola yang lebih relevan.

c. Bagaimana implementasinya:

Proses preprocessing dilakukan menggunakan operasi berbasis *regular expression* dan manipulasi string Python. Hasil teks yang telah dibersihkan disimpan pada kolom clean_text.

5.6.4. Feature Extraction (TF-IDF)

a. Apa yang dilakukan:

Mengubah teks hasil preprocessing menjadi representasi numerik menggunakan metode TF-IDF.

b. Mengapa penting:

Algoritma machine learning seperti Logistic Regression dan Random Forest tidak dapat memproses data teks mentah, sehingga diperlukan representasi numerik yang informatif.

c. Bagaimana implementasinya:

TF-IDF Vectorizer digunakan dengan maksimum **10.000 fitur**, rentang *n-gram* **(1,2)**, nilai min_df sebesar 5, dan max_df sebesar 0.9. Representasi TF-IDF diterapkan pada data latih dan data uji.

5.6.5. Tokenization dan Padding

a. Apa yang dilakukan:

Mengonversi teks menjadi urutan token numerik dan menyeragamkan panjang sequence melalui padding.

b. Mengapa penting:

Model LSTM membutuhkan input berupa sequence dengan panjang yang sama agar dapat diproses secara batch dan mempertahankan urutan kata.

c. **Bagaimana implementasinya:**

Tokenizer dari Keras digunakan dengan batas maksimum **8.000 kata** dan token khusus untuk kata yang tidak dikenal (*out-of-vocabulary token*). Selanjutnya, urutan token disesuaikan panjangnya menggunakan padding hingga panjang maksimum **80 token**.

5.6.6. Data Splitting

a. **Apa yang dilakukan:**

Membagi dataset menjadi data latih dan data uji dengan proporsi 80:20 menggunakan stratifikasi.

b. **Mengapa penting:**

Pembagian data diperlukan untuk mengevaluasi performa model secara objektif terhadap data yang belum pernah dilihat sebelumnya, serta menjaga distribusi kelas.

c. **Bagaimana implementasinya:**

Pembagian data dilakukan menggunakan fungsi `train_test_split()` dengan parameter `stratify` pada label sentimen dan `random_state` bernilai 42.

5.6.7. Data Balancing

a. **Apa yang dilakukan:**

Menangani ketidakseimbangan kelas dengan pemberian bobot kelas (class weighting).

b. **Mengapa penting:**

Ketidakseimbangan kelas dapat menyebabkan model bias terhadap kelas mayoritas dan menurunkan performa pada kelas minoritas.

c. **Bagaimana implementasinya:**

Bobot kelas dihitung menggunakan fungsi `compute_class_weight()` dan diterapkan pada proses training model.

6. MODELING

6.1. Model 1 – Baseline Model (Logistic Regression)

6.1.1. Deskripsi Model

Logistic Regression digunakan sebagai model baseline karena kesederhanaannya serta kemampuannya dalam menangani data berdimensi tinggi seperti hasil ekstraksi TF-IDF. Model ini digunakan untuk memberikan gambaran awal performa klasifikasi sentimen sebelum diterapkan model yang lebih kompleks.

6.1.2. Hyperparameter

Hyperparameter yang digunakan pada model Logistic Regression adalah sebagai berikut:

1. C: 2.0
2. Solver: lbfgs
3. Maximum Iteration: 2000
4. Class Weight: balanced
5. Random State: 42

Penggunaan `class_weight='balanced'` bertujuan untuk mengurangi bias akibat ketidakseimbangan kelas pada dataset.

6.1.3. Implementasi (Ringkasan)

```
❶ #@title Implementasi
from sklearn.linear_model import LogisticRegression

logreg_model = LogisticRegression(
    C=2.0,
    max_iter=2000,
    class_weight='balanced',
    random_state=42
)

logreg_model.fit(X_train_tfidf, y_train)
y_pred_logreg = logreg_model.predict(X_test_tfidf)
```

Model Logistic Regression dilatih menggunakan data latih TF-IDF, kemudian dilakukan prediksi pada data uji untuk memperoleh hasil klasifikasi sentimen. Implementasi dilakukan menggunakan library Scikit-learn.

6.1.4. Hasil Awal

Sebagai model baseline, Logistic Regression menunjukkan performa yang stabil dan konsisten dalam mengklasifikasikan sentimen. Model ini mampu mengenali sentimen positif dengan baik, namun masih memiliki keterbatasan dalam menangani variasi pola pada kelas minoritas.

6.2. Model 2 – ML/Advenced Model

6.2.1. Deskripsi Model

Random Forest merupakan model ensemble berbasis decision tree yang mampu menangkap hubungan non-linear dalam data. Model ini digunakan untuk meningkatkan performa klasifikasi dibandingkan model baseline dengan memanfaatkan gabungan beberapa pohon keputusan.

6.2.2. Hyperparameter

Hyperparameter yang digunakan pada model Random Forest adalah sebagai berikut:

1. Number of Trees (n_estimators): 300
2. Maximum Depth: 25
3. Minimum Samples Split: 10
4. Class Weight: balanced
5. Random State: 42

Konfigurasi ini dipilih untuk menjaga keseimbangan antara kompleksitas model dan kemampuan generalisasi.

6.2.3. Implementasi (Ringkasan)

```
▶ #@title Implementasi
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(
    n_estimators=300,
    max_depth=25,
    min_samples_split=10,
    class_weight='balanced',
    random_state=42,
    n_jobs=-1
)
rf_model.fit(X_train_tfidf, y_train)
y_pred_rf = rf_model.predict(X_test_tfidf)
```

Model Random Forest dilatih menggunakan data latih berbasis TF-IDF yang sama dengan model Logistic

Regression agar hasil evaluasi dapat dibandingkan secara adil. Proses training dilakukan secara paralel dengan memanfaatkan seluruh core CPU (`n_jobs = -1`).

6.2.4. Hasil Awal

Model Random Forest menunjukkan peningkatan performa dibandingkan Logistic Regression. Model ini lebih mampu menangkap pola kompleks dan hubungan non-linear pada data TF-IDF, sehingga menghasilkan prediksi yang lebih baik, khususnya pada kelas minoritas.

6.3. Model 3 – Deep Learning Model

6.3.1. Deskripsi Model

Long Short-Term Memory (LSTM) merupakan jenis Recurrent Neural Network (RNN) yang dirancang untuk mempelajari dependensi jangka panjang pada data sekuensial. Model ini digunakan untuk menangkap konteks dan urutan kata dalam ulasan pengguna yang tidak dapat sepenuhnya ditangkap oleh metode TF-IDF.

6.3.2. Arsitektur Model

Arsitektur model LSTM yang digunakan dalam penelitian ini terdiri dari beberapa lapisan sebagai berikut:

1. Embedding Layer
 - a. Dimensi embedding: 128
 - b. Berfungsi untuk mengubah token menjadi representasi vektor berdimensi rendah
2. Bidirectional LSTM Layer
 - a. Jumlah unit: 64
 - b. Digunakan untuk menangkap konteks teks dari dua arah (maju dan mundur)
3. Dropout Layer (0.4)
 - a. Digunakan untuk mengurangi risiko overfitting
4. Dense Layer
 - a. 32 unit dengan fungsi aktivasi ReLU

5. Dropout Layer (0.3)
6. Output Layer
 - a. 1 neuron dengan fungsi aktivasi Sigmoid
 - b. Jumlah neuron disesuaikan dengan jumlah kelas sentimen

6.3.3. Input & Processing khusus

Input yang digunakan pada model LSTM berupa sequence hasil tokenization dan padding. Proses preprocessing khusus untuk model ini meliputi:

1. Tokenization teks
2. Padding sequence hingga panjang tetap
3. Penggunaan label sentimen biner sebagai target

6.3.4. Hyperparameter

Model LSTM diimplementasikan menggunakan TensorFlow Keras. Proses training menggunakan:

1. Optimizer: Adam (learning rate = 0.001)
2. Loss Function: Binary Crossentropy
3. Batch Size: 64
4. Epoch: Maksimal 15
5. Validation Split: 20%
6. Class Weight: Menggunakan hasil perhitungan pada tahap data balancing

Selain itu, digunakan Early Stopping dengan patience = 3 untuk menghentikan pelatihan ketika tidak terjadi perbaikan pada nilai validation loss.

6.3.5. Implementasi (Ringkas)

```
  #@title Implementasi

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

vocab_size = min(10000, len(tokenizer.word_index) + 1)
max_len = X_train_seq.shape[1]

lstm_model = Sequential([
    Embedding(
        input_dim=vocab_size,
        output_dim=128,
        input_length=max_len
    ),
    Bidirectional(LSTM(64)),
    Dropout(0.4),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

lstm_model.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

history = lstm_model.fit(
    X_train_seq,
    y_train,
    validation_split=0.2,
    epochs=15,
    batch_size=64,
    class_weight=class_weights,
    callbacks=[early_stopping],
    verbose=1
)
```

Implementasi model LSTM dilakukan menggunakan pendekatan Sequential pada TensorFlow Keras. Model dibangun dengan lapisan embedding sebagai representasi awal kata, diikuti oleh lapisan Bidirectional LSTM untuk menangkap pola sekuensial dalam teks. Lapisan dropout diterapkan untuk mengurangi overfitting, sedangkan output layer menggunakan aktivasi sigmoid yang sesuai untuk permasalahan klasifikasi biner.

6.3.6. Training Process

Berdasarkan hasil training, nilai akurasi dan loss mengalami fluktuasi selama proses pelatihan. Hal ini menunjukkan bahwa model LSTM masih mengalami kesulitan dalam melakukan generalisasi secara optimal terhadap data

validasi. Penggunaan early stopping membantu mencegah overfitting dan mempertahankan bobot terbaik selama proses training.

6.3.7. Model Summary

Model LSTM yang dibangun memiliki arsitektur Sequential dengan total 4.148.933 parameter, yang terdiri dari beberapa lapisan utama sebagai berikut:

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 80, 128)	1,280,000
bidirectional_1 (Bidirectional)	(None, 128)	98,816
dropout_6 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 32)	4,128
dropout_7 (Dropout)	(None, 32)	0
dense_7 (Dense)	(None, 1)	33

```
Total params: 4,148,933 (15.83 MB)
Trainable params: 1,382,977 (5.28 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2,765,956 (10.55 MB)
```

7. EVALUATION

7.1. Metrik Evaluasi

Karena permasalahan ini merupakan klasifikasi sentimen multikelas, maka metrik yang digunakan adalah:

1. accuracy: proporsi prediksi yang benar terhadap seluruh data
2. Precision: kemampuan model menghindari kesalahan prediksi kelas
3. Recall: kemampuan model menangkap seluruh data dari suatu kelas
4. F1-Score: keseimbangan antara precision dan recall
5. Confusion Matrix: visualisasi kesalahan dan kebenaran prediksi

7.2. Hasil Evaluasi Model

7.2.1. Model 1 (Baseline)

1. Metrik:

```
⌚ #@title Metrik Evaluasi

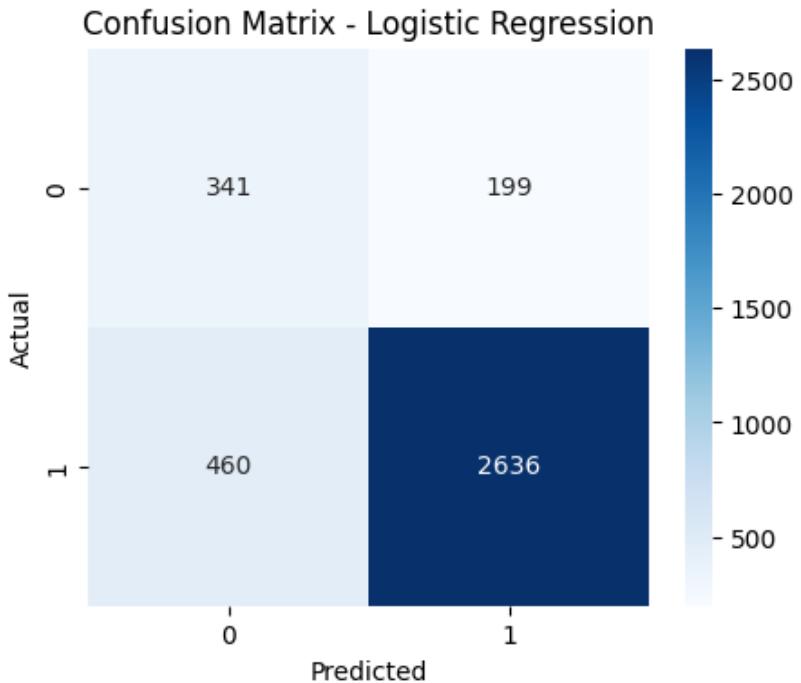
acc_lr = accuracy_score(y_test, y_pred_logreg)
prec_lr = precision_score(y_test, y_pred_logreg, average='weighted')
rec_lr = recall_score(y_test, y_pred_logreg, average='weighted')
f1_lr = f1_score(y_test, y_pred_logreg, average='weighted')

print("== Logistic Regression ==")
print(f"Accuracy : {acc_lr:.4f}")
print(f"Precision : {prec_lr:.4f}")
print(f"Recall    : {rec_lr:.4f}")
print(f"F1-Score   : {f1_lr:.4f}")
```

Berdasarkan hasil evaluasi pada data uji, model Logistic Regression menghasilkan performa sebagai berikut:

1. Accuracy: 0.8188
2. Precision: 0.8549
3. Recall: 0.8188
4. F1-Score: 0.8324

2. Confusion Matrix / Visualization:



7.2.2. Model 2 (ML)

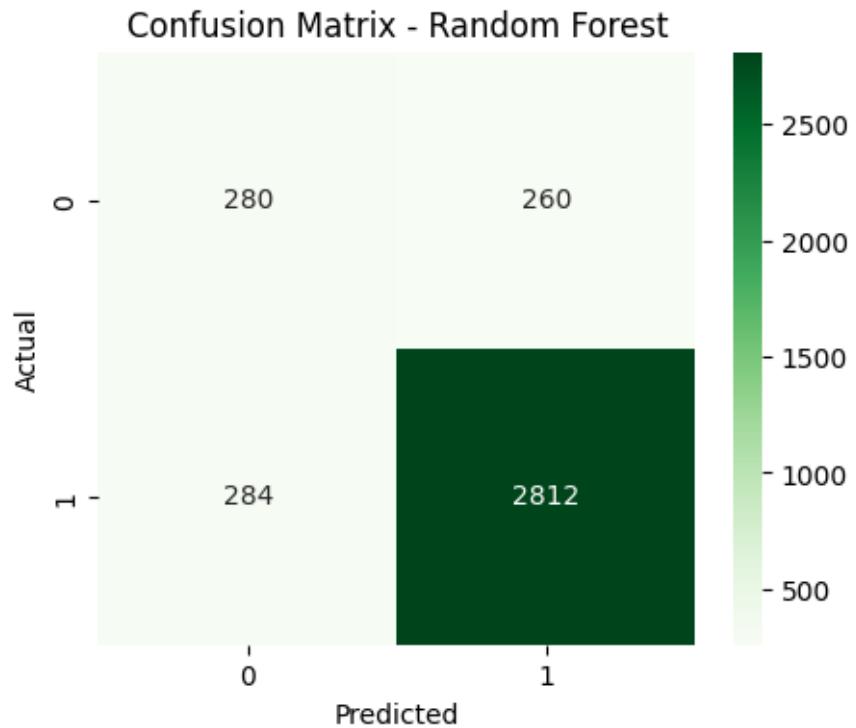
1. Metrik:

```
#@title Metrik Evaluasi  
acc_rf = accuracy_score(y_test, y_pred_rf)  
prec_rf = precision_score(y_test, y_pred_rf, average='weighted')  
rec_rf = recall_score(y_test, y_pred_rf, average='weighted')  
f1_rf = f1_score(y_test, y_pred_rf, average='weighted')  
  
print("== Random Forest ==")  
print(f"Accuracy : {acc_rf:.4f}")  
print(f"Precision : {prec_rf:.4f}")  
print(f"Recall : {rec_rf:.4f}")  
print(f"F1-Score : {f1_rf:.4f}")
```

Hasil evaluasi model Random Forest menunjukkan peningkatan performa dibandingkan model baseline, dengan hasil sebagai berikut:

1. Accuracy: 0.8504
2. Precision: 0.8532
3. Recall: 0.8504
4. F1-Score: 0.8517

2. Confusion Matrix / Visualization:



7.2.3. Model 3 (Deep Learning)

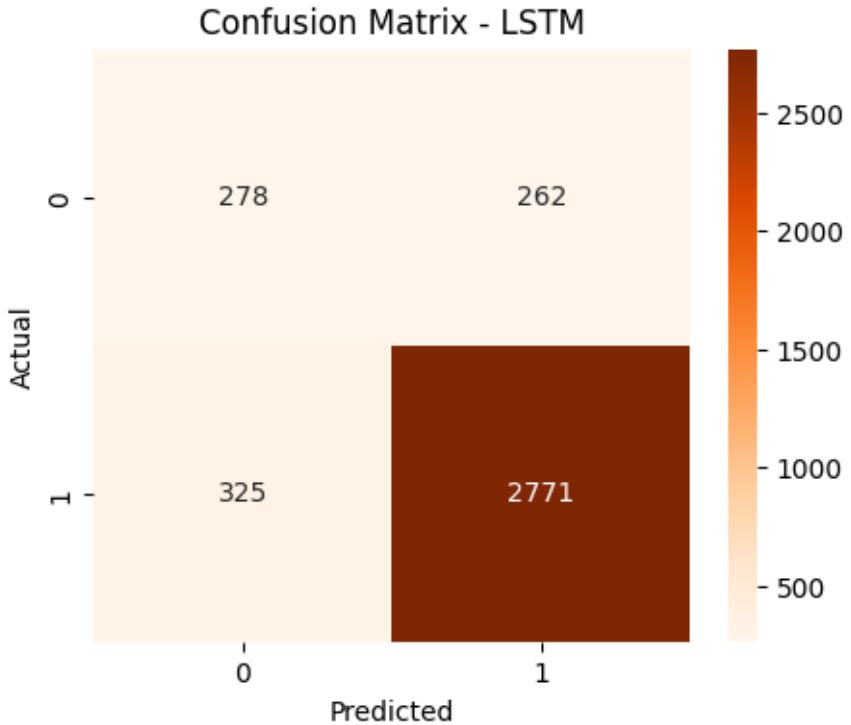
1. Metrik:

```
 #@title Metrik Evaluasi  
acc_lstm = accuracy_score(y_test, y_pred_lstm)  
prec_lstm = precision_score(y_test, y_pred_lstm, average='weighted')  
rec_lstm = recall_score(y_test, y_pred_lstm, average='weighted')  
f1_lstm = f1_score(y_test, y_pred_lstm, average='weighted')  
  
print("== LSTM ==")  
print(f"Accuracy : {acc_lstm:.4f}")  
print(f"Precision : {prec_lstm:.4f}")  
print(f"Recall    : {rec_lstm:.4f}")  
print(f"F1-Score   : {f1_lstm:.4f}")
```

Model LSTM menghasilkan performa sebagai berikut:

1. Accuracy: 0.8386
2. Precision: 0.8464
3. Recall: 0.8386
4. F1-Score: 0.8422

2. Confusion Matrix / Visualization:



3. Training History:

```
Epoch 1/15
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
182/182 49s 214ms/step - accuracy: 0.5826 - loss: 0.6673 - val_accuracy: 0.7680 - val_loss: 0.5789
Epoch 2/15
182/182 41s 211ms/step - accuracy: 0.8247 - loss: 0.4974 - val_accuracy: 0.7329 - val_loss: 0.6266
Epoch 3/15
182/182 53s 291ms/step - accuracy: 0.8603 - loss: 0.4211 - val_accuracy: 0.8353 - val_loss: 0.4596
Epoch 4/15
182/182 86s 314ms/step - accuracy: 0.8899 - loss: 0.3575 - val_accuracy: 0.8168 - val_loss: 0.4919
Epoch 5/15
182/182 65s 219ms/step - accuracy: 0.9097 - loss: 0.2859 - val_accuracy: 0.8099 - val_loss: 0.5131
Epoch 6/15
182/182 44s 243ms/step - accuracy: 0.9154 - loss: 0.2347 - val_accuracy: 0.8212 - val_loss: 0.5173
```

4. Test Set Predictions:

Prediksi pada data uji dilakukan dengan menerapkan threshold 0.5 pada output probabilitas sigmoid. Hasil prediksi kemudian digunakan untuk menghitung metrik evaluasi dan confusion matrix.

7.3. Perbandingan Ketiga Model

Perbandingan Performa Model

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.819	0.855	0.819	0.832
Random Forest	0.850	0.853	0.850	0.852
LSTM	0.839	0.846	0.839	0.842

Berdasarkan hasil evaluasi, perbandingan performa ketiga model dapat dirangkum sebagai berikut:

1. Random Forest menghasilkan performa terbaik secara keseluruhan dengan nilai accuracy dan F1-score tertinggi.
2. LSTM memberikan performa yang kompetitif dengan kemampuan memahami konteks sekuensial teks, namun membutuhkan sumber daya komputasi yang lebih besar.
3. Logistic Regression berperan sebagai model baseline yang sederhana dengan performa yang cukup baik, tetapi kalah dibandingkan dua model lainnya.

7.4. Analisis Hasil

Hasil evaluasi menunjukkan bahwa pendekatan machine learning klasik menggunakan Random Forest lebih efektif pada dataset ini dibandingkan model deep learning LSTM. Hal ini kemungkinan disebabkan oleh ukuran dataset, distribusi kelas yang tidak seimbang, serta efektivitas fitur TF-IDF dalam merepresentasikan teks ulasan. Meskipun LSTM memiliki kemampuan memahami konteks urutan kata, peningkatan performa yang dihasilkan belum mampu melampaui Random Forest secara signifikan.

8. CONCLUSION

8.1. Kesimpulan Utama

8.1.1. Model terbaik

Berdasarkan hasil evaluasi yang telah dilakukan, model Random Forest merupakan model terbaik dalam project ini.

8.1.2. Alasan

Model Random Forest menunjukkan performa paling unggul dibandingkan model lainnya dengan nilai accuracy sebesar 0.8504 dan F1-score sebesar 0.8517, yang merupakan nilai tertinggi di antara ketiga model yang diuji. Selain itu, model ini mampu memberikan keseimbangan yang baik antara precision dan recall pada data yang bersifat tidak seimbang. Dibandingkan Logistic Regression dan LSTM, Random Forest lebih stabil dalam mengklasifikasikan sentimen positif dan negatif pada dataset ulasan pengguna.

8.1.3. Pencapaian Goals

Tujuan penelitian yang dijelaskan pada Section 3.2 telah tercapai. Model yang dibangun mampu melakukan klasifikasi sentimen ulasan pengguna dengan tingkat akurasi yang baik, serta memungkinkan dilakukan perbandingan performa antara metode machine learning klasik dan deep learning untuk menentukan pendekatan terbaik pada dataset yang digunakan.

8.2. Key Insight

8.2.1. Insight dari Data

1. Dataset ulasan pengguna memiliki ketidakseimbangan kelas, di mana sentimen positif jauh lebih dominan dibandingkan sentimen negatif.
2. Ulasan dengan rating tinggi (≥ 4 bintang) cenderung mengandung kata-kata bernada positif yang konsisten, sehingga lebih mudah diklasifikasikan oleh model.

3. Proses pembersihan teks dan preprocessing sangat berpengaruh terhadap kualitas representasi fitur dan performa model secara keseluruhan.

8.2.2. Insight dari Modeling

1. Model machine learning berbasis TF-IDF + Random Forest mampu menghasilkan performa yang lebih baik dibandingkan model deep learning pada dataset ini.
2. Model LSTM memiliki kemampuan memahami urutan kata, namun keunggulan tersebut belum memberikan peningkatan performa yang signifikan dibandingkan Random Forest, kemungkinan akibat keterbatasan ukuran dan karakteristik data.

8.3. Kontribusi Proyek

8.3.1. Manfaat Praktis

Proyek ini dapat digunakan sebagai dasar pengembangan sistem analisis sentimen otomatis untuk mengolah ulasan pengguna, seperti pada platform e-commerce, aplikasi layanan publik, atau media sosial. Hasil analisis sentimen dapat membantu pengelola layanan dalam memahami persepsi pengguna serta meningkatkan kualitas produk dan layanan berdasarkan umpan balik yang diberikan.

8.3.2. Pembelajaran yang Dapat

Melalui proyek ini, diperoleh pemahaman mendalam mengenai tahapan pengolahan data teks, mulai dari data cleaning, preprocessing, feature extraction, hingga evaluasi model klasifikasi. Selain itu, proyek ini memberikan pengalaman dalam membandingkan performa berbagai pendekatan klasifikasi, baik machine learning maupun deep learning, serta memahami pengaruh karakteristik data terhadap hasil pemodelan.

9. FUTURE WORK

Data:

- Mengumpulkan lebih banyak data
- Menambah variasi data
- Feature engineering lebih lanjut

Model:

- Mencoba arsitektur DL yang lebih kompleks
- Hyperparameter tuning lebih ekstensif
- Ensemble methods (combining models)
- Transfer learning dengan model yang lebih besar

Deployment:

- Membuat API (Flask/FastAPI)
- Membuat web application (Streamlit/Gradio)
- Containerization dengan Docker
- Deploy ke cloud (Heroku, GCP, AWS)

Optimization:

- Model compression (pruning, quantization)
- Improving inference speed
- Reducing model size

10. REPRODUCIBILITY

10.1. GitHub Repository

Link Repository:

https://github.com/RaraAliviana/UAS_DataScience_234311050.git

Repository harus berisi:

- Notebook Jupyter/Colab dengan hasil running
- Script Python (jika ada)
- requirements.txt atau environment.yml
- README.md yang informatif
- Folder structure yang terorganisir
- .gitignore (jangan upload dataset besar)

10.2. Environment & Dependencies

1. Python Version: 3.12.12

2. Main Libraries & Versions:

```
numpy==1.26.4  
pandas==2.1.4  
scikit-learn==1.4.1  
matplotlib==3.8.3  
seaborn==0.13.2
```

```
# Natural Language Processing  
nltk==3.8.1  
regex==2023.12.25
```

```
# Deep Learning  
tensorflow==2.15.0  
keras==2.15.0
```

```
# Utilities  
joblib==1.3.2  
tqdm==4.66.2
```