

Analisis Sentimen Ulasan Produk

[COMP6885001] - Natural Language Processing

KELAS LB01



Oleh Kelompok 1:

Ivan Setiawan – 2602109473

Yithro Paulus Tjendra – 2602112051

Jonathan Edmund William – 2602152982

Muhammad Fathan – 2602196435

Semester Genap 2024/2025

BINUS Kemanggisan

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1. PENDAHULUAN.....	3
Kasus.....	3
Solusi.....	3
Tautan.....	3
BAB 2. DATASET.....	4
Sumber Dataset.....	4
Kondisi Awal Data.....	4
Analisa dan Persiapan Data.....	4
Visualisasi.....	5
Pra Proses Data.....	6
BAB 3. PELATIHAN DAN PEMODELAN.....	7
Pembagian Data.....	7
Word Embedding.....	7
Logistic Regression.....	8
Gaussian Naive Bayes.....	9
Multinomial Naive Bayes.....	10
Complement Naive Bayes.....	11
Grid Search Cross Validation.....	12
BAB 4. HASIL DAN DISKUSI.....	14
Confusion Matrix.....	14
Evaluasi Performa.....	14
BAB 5. DEPLOYMENT.....	15

BAB 1. PENDAHULUAN

Kasus

Rina adalah seorang *social media influencer* Indonesia yang terkenal di Toktok. Dia sering menjual berbagai macam produk kecantikan melalui *live streaming* dan *video feeds* di akun Toktok-nya. Namun sayangnya, aplikasi Toktok tidak mempunyai fitur yang dapat membedakan antara komentar positif atau negatif dari *viewers* dan *followers* mengenai produk yang dia jual. Hal ini membuat Rina kesulitan untuk mengetahui kualitas produk yang dijualnya karena Toktok tidak memiliki sistem *rating* di dalam komen videonya.

Solusi

Maka dari itu, kami sebagai *developer* Toktok mendengarkan keluhan dari user aplikasi kami. Sehingga kami menawarkan proposal mengenai penambahan fitur kepada bos kami. Solusi yang kami tawarkan adalah menganalisa sentimen ulasan produk berbasis *machine learning* agar komen dapat diprediksi dengan cepat dan akurat. Dengan demikian, Rina sebagai *social media influencer* dapat mengetahui statistika baik-buruknya kualitas produknya berdasarkan komen yang muncul.

Tautan

Website: <https://id-prediksiulasan.streamlit.app/>

Analisa: <https://github.com/vanssnn/id-analisis-sentimen-ulasan-produk/blob/main/utills/analysis.ipynb>

Preprocess: <https://github.com/vanssnn/id-analisis-sentimen-ulasan-produk/blob/main/utills/preprocess.ipynb>

Dataset: <https://github.com/vanssnn/id-analisis-sentimen-ulasan-produk/tree/main/dataset>

Repositori GitHub: <https://github.com/vanssnn/id-analisis-sentimen-ulasan-produk>

BAB 2. DATASET

Sumber Dataset

Data diambil dari situs repositori GitHub Rhiosutoyo. Dataset yang bernama PRDECT - ID ini merupakan kumpulan data yang mencantumkan ulasan-ulasan pelanggan mengenai suatu produk. Berikut adalah tautan repositori GitHub dari sumber data yang kami gunakan dalam project, <https://github.com/rhiosutoyo/PRDECT-ID-Indonesian-Product-Reviews-Dataset>.

Kondisi Awal Data

Dataset yang kami pakai merupakan data dari 5400 pelanggan dengan 11 kolom yang mengandung informasi seperti kategori produk, harga produk, dan ulasan mereka terhadap suatu produk. Pada proyek ini, informasi dataset yang akan dipakai cukup berupa ulasan produk mereka dan sentimen mereka terhadap produk tersebut.

Fitur	Tipe Data	Deskripsi
Customer Review	Object (String)	Ulasan pelanggan mengenai produk
Sentiment	Object (String)	Sentiment pelanggan terhadap produk

Table 1. Deskripsi fitur

Analisa dan Persiapan Data

Tahapan EDA bertujuan untuk melakukan eksplorasi dan analisis terhadap kondisi dataset. Pada tahap ini, kita mengetahui bahwa ada 5400 baris, 11 kolom, tidak ada *missing value* dan terdapat 95 baris yang terduplikat.

Visualisasi

1. Count Plot

Visualisasi *count plot* untuk menampilkan jumlah frekuensi berdasarkan label sentimen pelanggan. Terlihat dari visualisasi berikut bahwa data yang tersedia sudah seimbang.

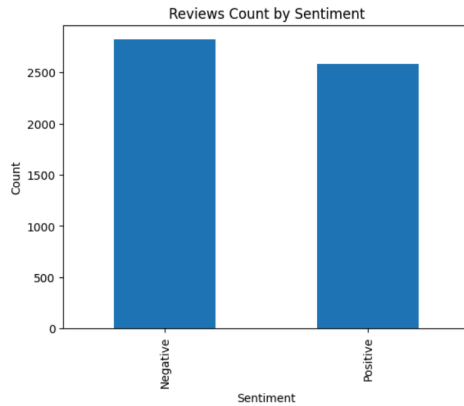


Figure 1. Bar Chart

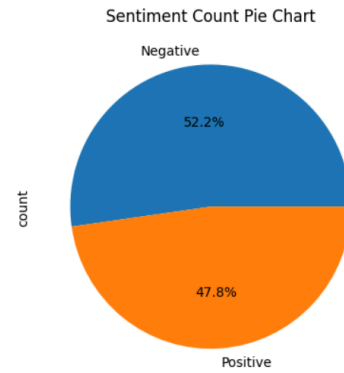


Figure 2. Pie Chart

2. Distribusi Jumlah Kata

Terlihat dari visualisasi berikut bahwa kebanyakan ulasan tidak mencapai 25 kata.

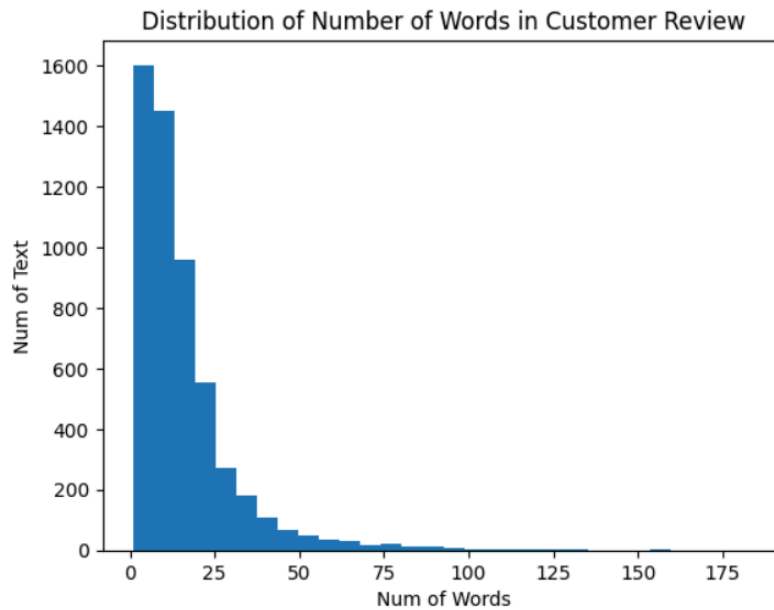


Figure 3. Histogram

3. Word Cloud

Terlihat dari visualisasi berikut bahwa semakin besar kata tersebut, semakin banyak juga kata tersebut digunakan di ulasan.



Figure 4. Word Cloud

Pra Proses Data

Sebelum data dipakai data harus dipersiapkan dan dibersihkan terlebih dahulu sehingga data yang masuk ke dalam model tidak bersifat data kotor atau terhitung *noise* (data yang kotor dan mengandung banyak *noise* akan menurunkan akurasi model). Hal ini bertujuan untuk mempersiapkan data sebelum dipakai oleh model. *Code* dapat dilihat melalui tautan berikut, <https://github.com/vanssnn/id-analisis-sentimen-ulasan-produk/blob/main/utlis/preprocess.ipynb>

Tahapan-Tahapan Pra Proses Data:

1. Membuang kolom yang tidak digunakan (11 kolom menjadi 2 yaitu *Customer Review* dan *Sentiment*).
2. Membuang data yang bersifat *null* (*missing value*) atau data duplikat (95 baris) mengubah 5400 baris menjadi 5305 baris.
3. Dataset akan diterima per kalimat dimana kalimat tersebut akan dipisah menjadi kata menggunakan *word_tokenize* lalu dijadikan huruf kecil melalui fungsi *lower*.
4. Membuang *stopwords*, *punctuation*, *num_words*, dan imbuhan. Dimana:
 - a. **stopwords**: kata-kata yang dianggap kurang memiliki nilai (contoh: yang). Beberapa kata *slang* juga ditambahkan dan dimasukkan menjadi stopwords untuk dihapus.
 - b. **punctuation**: symbol-symbol (contoh: %' ^ * & ').
 - c. **num_words**: angka yang direpresentasikan dengan huruf (contoh: satu, dua).
 - d. **imbuhan**: Sufiks maupun afiks dari kata-kata akan dibuang menggunakan *stemmer* (contoh: memakan menjadi makan).

BAB 3. PELATIHAN DAN PEMODELAN

Pembagian Data

Data yang siap dipakai akan dibagi menjadi dua bagian, data *training* dan data *testing*. Disini kami menggunakan fungsi *train_test_split* dari *library sklearn* untuk memecah *dataset* menjadi 80% untuk *training* dan 20% untuk *testing*. Data yang diambil akan menggunakan *stratify* sehingga pengambilan persentase distribusi label data sama.

Word Embedding

Word embedding merupakan pembuatan representasi sebuah kata dalam bentuk vektor numerik agar dapat diterima model. *Word embedding* bertujuan untuk mencoba menangkap makna semantik dari kata-kata. Salah satu teknikny adalah TF-IDF yang mengukur seberapa penting sebuah kata dalam sebuah dokumen relatif terhadap kumpulan dokumen (corpus).

Komponen TF-IDF:

1. TF, menghitung frekuensi sebuah kata muncul dalam dokumen.
2. IDF, menghitung seberapa penting suatu kata dari kumpulan dokumen yang ada.
3. TF-IDF, nilai dari TF dan IDF akan dikalikan sehingga menghasilkan sebuah nilai numerik yang menjadi patokan seberapa penting kata tersebut dalam corpus.

Berikut adalah rumus dari TF-IDF Vectorizer:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Dimana:

- **Term Frequency (TF):** Mengukur seberapa sering suatu istilah t muncul dalam d .
 - Hitungan mentah: $TF(t, d) = f_{t,d}$, dimana $f_{t,d}$ adalah jumlah kemunculan istilah t dalam dokumen d .
 - Term frequency: $TF(t, d) = \frac{f_{t,d}}{n_d}$, dimana n_d adalah jumlah total istilah dalam dokumen d .
 - Log-normalisasi: $TF(t, d) = 1 + \log(f_{t,d})$ jika $f_{t,d} > 0$.
- **Inverse Document Frequency (IDF):** Mengukur seberapa penting suatu istilah dalam seluruh korpus D . IDF dihitung sebagai:
 - $TF(t, D) = \log(\frac{N}{1-n_t}) + 1$, dimana N adalah jumlah total dokumen dalam korpus D dan n_t adalah jumlah dokumen yang mengandung istilah t . Menambahkan 1 pada n_t memastikan istilah yang ada di semua dokumen tidak mengakibatkan pembagian dengan nol. Menambahkan 1 pada log memastikan bahwa istilah yang tidak ada di dokumen mana pun tidak menjadi nol.

Logistic Regression

Logistic Regression (LR) adalah salah satu model yang efektif dalam pengklasifikasian dimana data bersifat *linearly separable*. LR mencari fungsi-fungsi yang dapat menjelaskan antar *features* dalam dataset dan menghitung *value* dari masing-masing. LR akan menghasilkan probabilitas sebuah data *points* dalam kelas yang telah ditetapkan

Sebelum memahami cara kerja LR kita harus mengenal:

- Fungsi Logit

Fungsi *logit* adalah fungsi linear dari fitur-fitur input yang memiliki rumus berikut:

$$z = w^T \cdot X + b$$

Dimana:

- w^T adalah vektor bobot yang di *transpose*.
- X adalah vektor fitur input. (*feature*)
- b adalah bias (*intercept*).

- Fungsi Sigmoid:

Fungsi *logit* diteruskan ke fungsi *sigmoid* untuk mengubah nilai linear menjadi probabilitas antara 0 dan 1:

$$P(y = 1 | X) = \sigma(z) = \sigma(w^T \cdot X + b)$$

- Fungsi Loss

Menggunakan *log-loss* (*binary cross-entropy*) sebagai fungsi kerugian.

Cara kerja Logistic Regression:

- Setelah menetapkan fungsi *logit* awal, bobot dan bias biasanya diacak), Kita akan menggunakan algoritma optimasi seperti Gradient Descent kita dapat menyesuaikan bobot yang cocok terhadap masing-masing fitur. Hal ini untuk menentukan seberapa besar pengaruh suatu fitur input.
- Untuk menggunakan Gradient Descent kita harus menggunakan fungsi *loss* sebagai acuan. Ketika fungsi metode optimasi sudah berhenti. Maka model siap dipakai untuk klasifikasi biner. Dimana besar probabilitas menentukan kelas suatu input.

Parameter yang digunakan:

<i>Parameter</i>	<i>Value</i>	Penjelasan
<i>C</i>	[0.1, 0.5, 1.0, 1.5, 2.0]	Ini adalah kebalikan dari kekuatan regularisasi. Nilai <i>C</i> yang lebih kecil berarti regularisasi yang lebih kuat. Regularisasi digunakan untuk mencegah overfitting dengan memberikan penalti pada koefisien yang besar.
<i>max_iter</i>	[100, 200, 300, 400, 500]	Ini adalah jumlah iterasi maksimum yang akan dijalankan oleh solver untuk mencapai konvergensi. Jika solver tidak konvergen dalam jumlah iterasi ini, ia akan berhenti dan memberikan peringatan.
<i>solver</i>	['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']	Parameter ini menentukan algoritma optimasi yang akan digunakan untuk menemukan bobot optimal. Berbagai solver memiliki karakteristik dan efisiensi komputasi yang berbeda: <ul style="list-style-type: none">- newton-cg, lbfgs, dan sag hanya menangani regularisasi L2.- liblinear bekerja dengan regularisasi L1 dan L2.- saga dapat menangani regularisasi L1, L2, dan elastic-net.

Table 2. Logistic Regression Hyperparameter

Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi berdasarkan Teorema *Bayes* dengan asumsi bahwa fitur-fitur yang digunakan untuk klasifikasi mengikuti distribusi Gaussian (normal). Algoritma ini termasuk dalam keluarga model probabilistik dan bekerja sangat baik untuk dataset dengan fitur kontinu.

Sebelum memahami cara kerja *GNB* kita harus mengenal:

- *Asumsi Naïve*

Mengasumsikan bahwa semua fitur dalam dataset saling independen satu sama lain.

- *Distribusi Gaussian*

Mengasumsikan bahwa distribusi nilai fitur mengikuti distribusi normal (Gaussian).

Cara kerja GNB:

- GNB merupakan pengembangan dari MNB, dimana perbedaannya terletak bagaimana cara menghitung probabilitas kemunculan suatu benda pada suatu kelas. Dimana GNB menggunakan Distribusi Gaussian untuk menghitung probabilitasnya. Hal inilah yang membuat algoritma GNB dapat membuat sebuah variabel memiliki bobot yang berbeda ketika memprediksi (*feature importance*). Sebagai bentuk optimasi, bisa menggunakan fungsi logaritma untuk menghitung probabilitas dan menggunakan *cross validation* untuk menentukan variabel yang memiliki bobot yang sesuai.

Parameter yang digunakan:

<i>Parameter</i>	<i>Value</i>	Penjelasan
<i>var_smoothing</i>	[1e-9, 1e-8, 1e-7, 1e-6, 1e-5]	Parameter ini digunakan untuk menambahkan nilai kecil ke varians setiap fitur untuk memastikan stabilitas numerik. Ini sangat berguna ketika varians suatu fitur sangat kecil, yang dapat menyebabkan masalah komputasi.

Table 3. Gaussian Naive Bayes Hyperparameter

Multinomial Naive Bayes

Sebelum memahami cara kerja Multinomial Naive Bayes (MNB) kita harus mengenal:

- Laplace Smoothing

Menambahkan 1 di pembilang dan V untuk penyebut pada perhitungan probabilitas seperti pada rumus berikut:

$$P(w) = \frac{\text{Count}(w) + 1}{N + V}$$

V merupakan vocabulary dari total *unique words* yang ada. Hal ini dilakukan untuk mencegah nilai 0 pada perhitungan probabilitas.

Cara kerja MNB:

MNB bekerja dengan cara menghitung probabilitas kemunculan suatu benda pada kedua kelas, semisal pada kasus ini kemunculan kata x keluar pada kelas positif maupun negatif akan dihitung. Lalu probabilitas yang lebih tinggi akan menentukan kelas input.

Parameter yang digunakan:

<i>Parameter</i>	<i>Value</i>	Penjelasan
<i>alpha</i>	[0.1, 0.5, 1.0, 1.5, 2.0]	Ini adalah parameter pelicinan aditif. Pelicinan digunakan untuk menangani probabilitas nol dalam data. Nilai alpha yang lebih tinggi menghasilkan pelicinan lebih banyak.
<i>fit_prior</i>	[<i>True</i> , <i>False</i>]	Parameter boolean ini menentukan apakah akan mempelajari probabilitas prior kelas atau tidak. Jika <i>True</i> , model mempelajari probabilitas prior dari data. Jika <i>False</i> , diasumsikan semua kelas memiliki probabilitas prior yang sama.

Table 4. Multinomial Naive Bayes Hyperparameter

Complement Naive Bayes

Sebelum memahami cara kerja Complement Naive Bayes (CNB) kita harus mengenal:

- Laplace Smoothing

Menambahkan 1 di pembilang dan V untuk penyebut pada perhitungan probabilitas seperti pada rumus berikut:

$$P(w) = \frac{\text{Count}(w) + 1}{N + V}$$

V merupakan vocabulary dari total *unique words* yang ada. Hal ini dilakukan untuk mencegah nilai 0 pada perhitungan probabilitas.

Cara kerja CNB:

CNB bekerja dengan cara menghitung probabilitas kemunculan suatu benda diluar kelas target. Sayangnya pada kasus sentimen analisis model CNB menghitung nilai probabilitas yang sama. Karena pada kasus sentiment analysis hanya ada dua kelas sehingga fungsi CNB untuk bekerja pada distribusi kelas target yang tidak rata kurang terlihat.

Parameter yang digunakan:

<i>Parameter</i>	<i>Value</i>	Penjelasan
<i>alpha</i>	[0.1, 0.5, 1.0, 1.5, 2.0]	Ini adalah parameter pelicinan aditif. Pelicinan digunakan untuk menangani probabilitas nol dalam data. Nilai alpha yang lebih tinggi menghasilkan pelicinan lebih banyak.
<i>fit_prior</i>	[True, False]	Parameter boolean ini menentukan apakah akan mempelajari probabilitas prior kelas atau tidak. Jika True, model mempelajari probabilitas prior dari data. Jika False, diasumsikan semua kelas memiliki probabilitas prior yang sama.

Table 5. Complement Naive Bayes

Grid Search Cross Validation

Grid search digunakan untuk mencari parameter yang terbaik untuk dipakai dalam model secara iteratif. Semua model akan di tes dan validasi menggunakan 4-fold cross validation secara iteratif berdasarkan parameter yang telah ditentukan sebelumnya. 4-fold cross validation sendiri bekerja dengan membagi dataset training menjadi 4 segmen (4-fold), yang kemudian dibagi lagi menjadi 3 segmen train dan 1 segmen validation untuk setiap kombinasi. Dari percobaan tersebut, akan didapatkan parameter, skor, dan standar deviasi dari setiap model. Untuk skor, kami menggunakan metode f1 skor untuk memastikan bahwa model memiliki keseimbangan antara recall dan precision yang baik.



Figure 5. 4-Fold Cross Validation

<i>Model</i>	<i>Best Param</i>	<i>Best Score</i>	<i>Standard Deviation</i>
<i>lgr</i>	{'C': 2.0, 'max_iter': 100, 'solver': 'lbfgs'}	0.914973	0.004248
<i>gnb</i>	{'var_smoothing': 1e-05}	0.857889	0.008306
<i>mnb</i>	{'alpha': 0.5, 'fit_prior': False}	0.911902	0.005001
<i>cnb</i>	{'alpha': 0.5, 'fit_prior': True}	0.911902	0.005001

Table 6. Grid Search Result

Berdasarkan hasil dari Grid Search, akhirnya kami menggunakan Logistic Regression untuk pengembangan aplikasi analisis sentimen ulasan produk. Kami memilih Logistic Regression karena nilai skor f1 yang dihasilkan paling tinggi dari pada yang lain dengan nilai 0.914971 dan standar deviasi terendah yaitu 0.004248.

BAB 4. HASIL DAN DISKUSI

Berikut merupakan hasil evaluasi dari model Logistic Regression yang di *train* menggunakan seluruh data *training* dan diuji menggunakan data yang belum dilihat sebelumnya, yaitu data *testing*.

Confusion Matrix

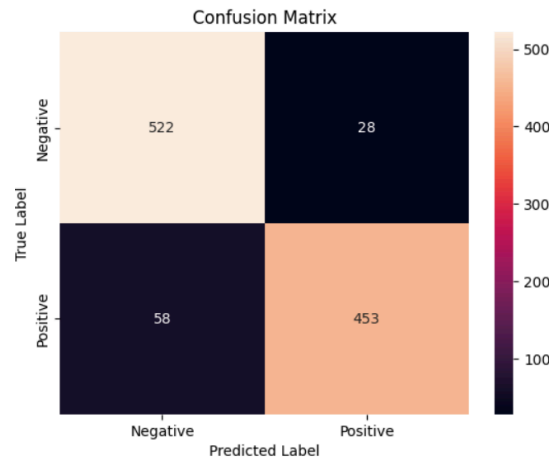


Figure 6. Confusion Matrix

Evaluasi Performa

	Precision	Recall	F1-Score	Support
Negative	0.90	0.95	0.92	550
Positive	0.94	0.89	0.91	511
Accuracy				1061
Macro Average	0.92	0.92	0.92	1061
Weighted Average	0.92	0.92	0.92	1061

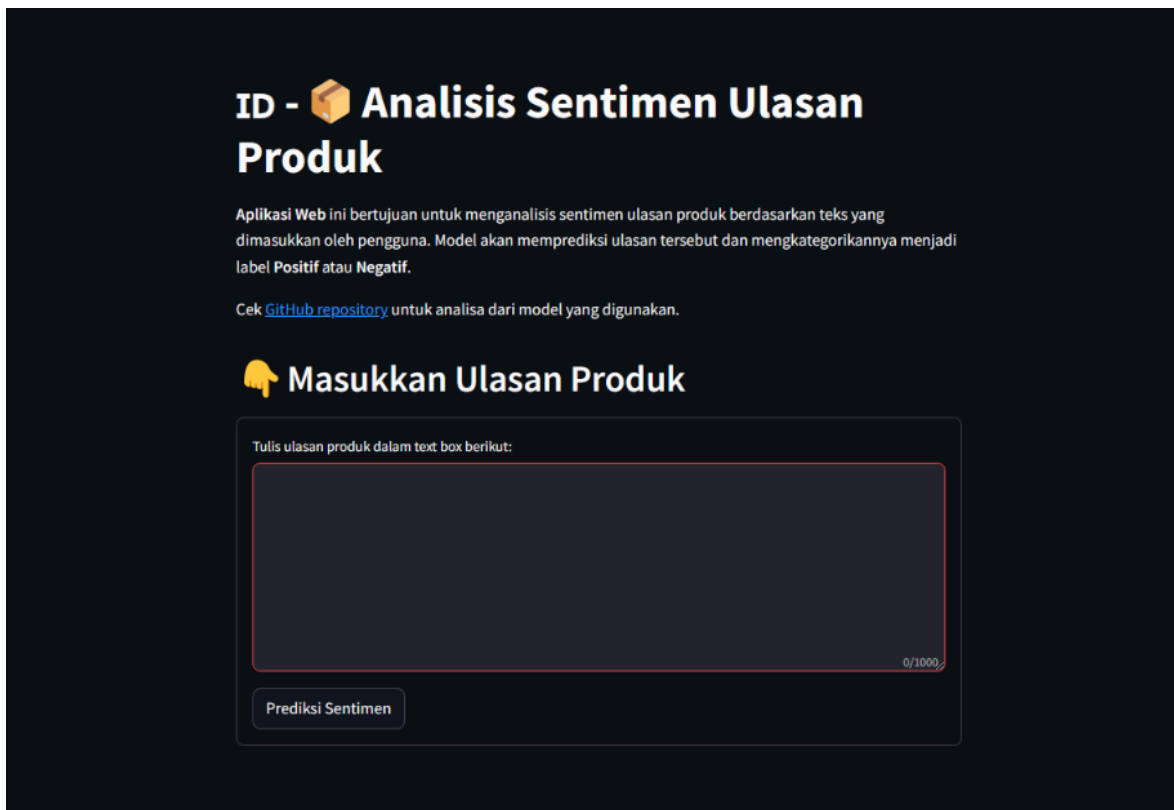
Table 7. Classification Report

ROC AUC Score: 0.9177939868350

Terlihat bahwa model memiliki performa yang baik di angka sekitar 0.92 untuk *Accuracy*, *Precision*, *Recall*, *F1-Score*, dan ROC AUC. Dapat disimpulkan bahwa model dapat membedakan label secara efektif. Namun, model belum bisa menangkap konteks semantik yang sebenarnya merupakan komponen penting dalam menganalisis sentimen. Hal ini dikarenakan tipe Word Embedding yang digunakan, yaitu TF-IDF yang merupakan teknik tradisional dan kurang dapat menangkap makna semantik dari sebuah kalimat. Teknik *word embedding* lain seperti Word2Vec, fastText, dan GloVe dapat dicoba dan digunakan untuk penelitian yang lebih lanjut. Model yang lebih kompleks seperti ANN, RNN atau LSTM, hingga *pretrained* seperti IndoBERT dan LLM dapat diimplementasikan dengan potensial performa yang lebih tinggi.

BAB 5. DEPLOYMENT

Setelah menyelesaikan seluruh proses analisis, kami membangun aplikasi web menggunakan Streamlit, yang merupakan Python *framework* untuk membuat aplikasi web interaktif dengan lebih mudah. Streamlit mempermudah pembuatan aplikasi web dengan menggunakan fungsionalitas widget, memungkinkan input pengguna dengan minimal baris kode. Streamlit merender konten secara berbeda dibandingkan dengan kerangka kerja situs web biasanya. Streamlit selalu menggunakan urutan dari atas ke bawah untuk merender dan menjalankan kode. Setelah mendapatkan input pengguna, kami memuat semua aset yang disimpan dalam analisis, memproses data, dan menampilkan prediksi. Kemudian, kami *deploy* analisis dan aplikasi web ke GitHub dan menggunakan Streamlit Community Cloud untuk meng-host aplikasi web di internet sehingga dapat diakses. web App telah di host dan di deploy di tautan <https://id-prediksiulasan.streamlit.app/>, berikut tampilan dari web app kami.



The screenshot shows a web application interface with a dark background. At the top, the title "ID - Analisis Sentimen Ulasan Produk" is displayed in white, with a small orange cube icon next to "ID". Below the title, a paragraph explains the application's purpose: "Aplikasi Web ini bertujuan untuk menganalisis sentimen ulasan produk berdasarkan teks yang dimasukkan oleh pengguna. Model akan memprediksi ulasan tersebut dan mengkategorikannya menjadi label Positif atau Negatif." Below this, a link to the GitHub repository is provided: "Cek [GitHub repository](#) untuk analisa dari model yang digunakan." The main section is titled "Masukkan Ulasan Produk" with a hand icon. It contains a text input area with the placeholder text "Tulis ulasan produk dalam text box berikut:" and a character count "0/1000". Below the input area is a button labeled "Prediksi Sentimen".

Figure 7. Tampilan Web App