**TUGAS BESAR 2**

**Natural Language Processing**

**Analisis Sentimen Ulasan Pada Aplikasi Spotify**

****

**Dosen Pengampu:**

Inna Sabily Karima S.kom., M.Kom.

**Disusun Oleh:**

Amrullah Valentino Caesar Putra – 41522010216

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MERCU BUANA**

**TAHUN 2024**

# Latar Belakang

Spotify adalah salah satu aplikasi streaming musik paling populer di dunia, dengan jutaan pengguna aktif setiap bulannya. Ulasan pengguna yang tersedia di platform aplikasi dapat memberikan wawasan berharga tentang pengalaman, kepuasan, dan masalah yang mereka alami.

Analisis sentimen terhadap ulasan pengguna bertujuan untuk memahami bagaimana perasaan pengguna terhadap aplikasi ini secara umum, baik itu positif, negatif, maupun netral. Informasi ini dapat membantu pengembang aplikasi dalam meningkatkan kualitas layanan, memperbaiki fitur, dan menyesuaikan strategi bisnis sesuai kebutuhan pengguna.

Dataset yang digunakan dalam analisis ini adalah *Spotify App Reviews 2022* yang diambil dari Kaggle. Dataset ini memuat berbagai ulasan pengguna terkait aplikasi Spotify. Proses analisis dilakukan menggunakan Google Colab dengan teknik *Natural Language Processing* (NLP) untuk menentukan sentimen setiap ulasan.

# Metodologi

1. Pengumpulan Data

Dataset diambil dari Kaggle melalui tautan berikut: [Spotify App Reviews 2022](https://www.kaggle.com/datasets/mfaaris/spotify-app-reviews-2022). Dataset ini mencakup kolom-kolom seperti teks ulasan, tanggal ulasan, dan skor yang diberikan oleh pengguna.

1. Pra-Pemrosesan Data

Pra-pemrosesan dilakukan untuk membersihkan data ulasan agar lebih mudah dianalisis. Langkah-langkah yang dilakukan meliputi:

* Menghapus teks yang tidak relevan (misalnya simbol, angka, dan tautan).
* Mengubah semua teks menjadi huruf kecil untuk konsistensi.
* Melakukan tokenisasi untuk memisahkan kata-kata dalam ulasan.
* Menghapus kata-kata tidak penting (*stopwords*).

1. Analisis Sentimen

Teknik yang digunakan untuk analisis sentimen mencakup:

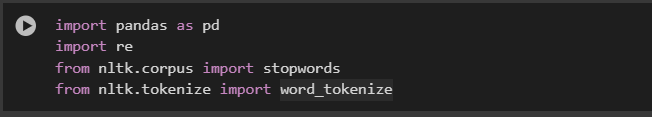
* *TextBlob* atau pustaka lain untuk mengklasifikasikan ulasan sebagai positif, negatif, atau netral.
* Menghitung distribusi sentimen untuk memahami pola keseluruhan.

1. Visualisasi Data

Hasil analisis divisualisasikan menggunakan diagram batang dan pie chart untuk menunjukkan distribusi sentimen dan ulasan pengguna secara keseluruhan.

# Hasil Dan Analisis

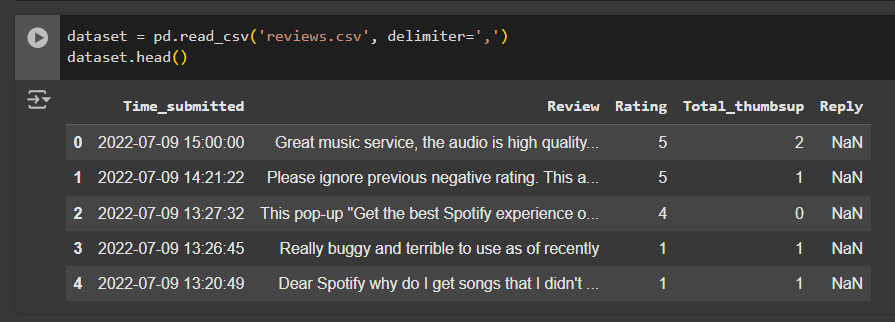
**IMPORT LIBRARY**



* Import pandas as = Mengimpor pustaka Pandas, yang merupakan pustaka Python untuk manipulasi dan analisis data, pd adalah alias standar untuk mempermudah penggunaan fungsi dari pustaka ini. Kegunaan Membaca, menulis, dan memanipulasi dataset dalam format seperti CSV, Excel, JSON,dll. Mengelola data dalam struktur tabel yang disebut DataFrame.
* Import re = Mengimpor pustaka **re** (regular expression) yang digunakan untuk pencocokan pola dan pengolahan teks. Kegunaan Membersihkan data teks, Mencari, mengganti, atau memodifikasi string menggunakan pola tertentu. Contoh: Menghapus simbol atau karakter tidak diinginkan dari teks ulasan.
* from nltk.corpus import stopwords = Mengimpor daftar stopwords dari pustaka NLTK (Natural Language Toolkit). Stopwords adalah kata-kata umum dalam suatu bahasa yang sering diabaikan dalam analisis teks, seperti "dan," "atau," "di," dll. Kegunaan Membersihkan data teks dengan menghapus kata-kata yang tidak memiliki makna signifikan untuk analisis, misalnya pada tugas analisis sentimen.
* from nltk.tokenize import word\_tokenize = Mengimpor fungsi word\_tokenize dari pustaka NLTK untuk memecah teks menjadi unit kata (token). Memisahkan kalimat menjadi kata-kata individual agar dapat diproses lebih lanjut.

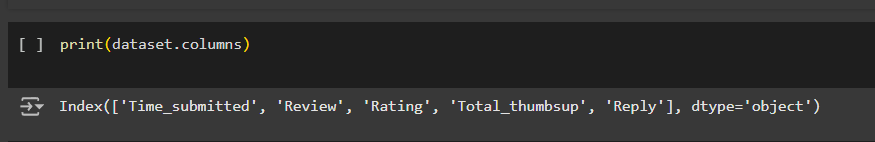
Contoh: Mengubah teks "Saya suka musik" menjadi daftar kata ['Saya', 'suka', 'musik'].

IMPORT DATA



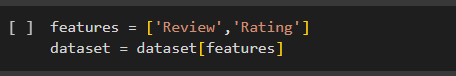
* dataset = pd.read\_csv('reviews.csv', delimiter=',') = embaca file CSV bernama reviews.csv dan memuatnya ke dalam DataFrame Pandas. Delimiter berfungsi untuk memisahkan kolom dalam file.
* dataset.head() = Menampilkan 5 baris pertama (secara default) dari DataFrame. Hal ini berguna untuk mendapatkan gambaran awal tentang struktur data, kolom yang tersedia, dan beberapa data awal.

CEK KOLOM DATA



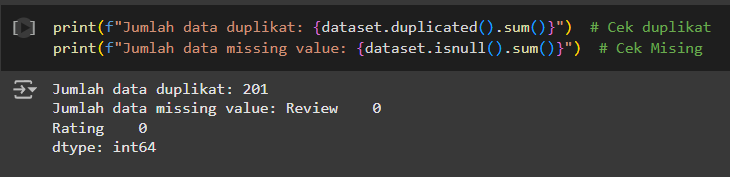
* print(dataset.columns) = digunakan untuk mencetak daftar nama kolom yang ada dalam DataFrame Pandas.

PILIH KOLOM



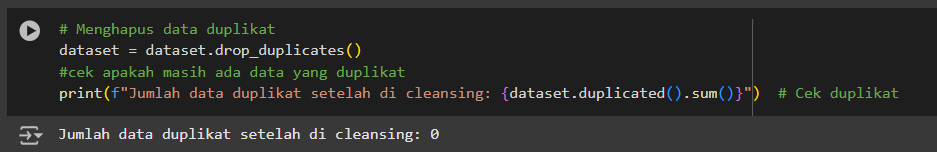
Membuat sebuah list bernama **features** yang berisi nama-nama kolom yang ingin dipilih dari DataFrame **dataset**. Memilih hanya kolom yang tercantum dalam **features** dari DataFrame **dataset**, lalu menyimpannya kembali ke variabel **dataset**.

CEK ANOMALI DATA



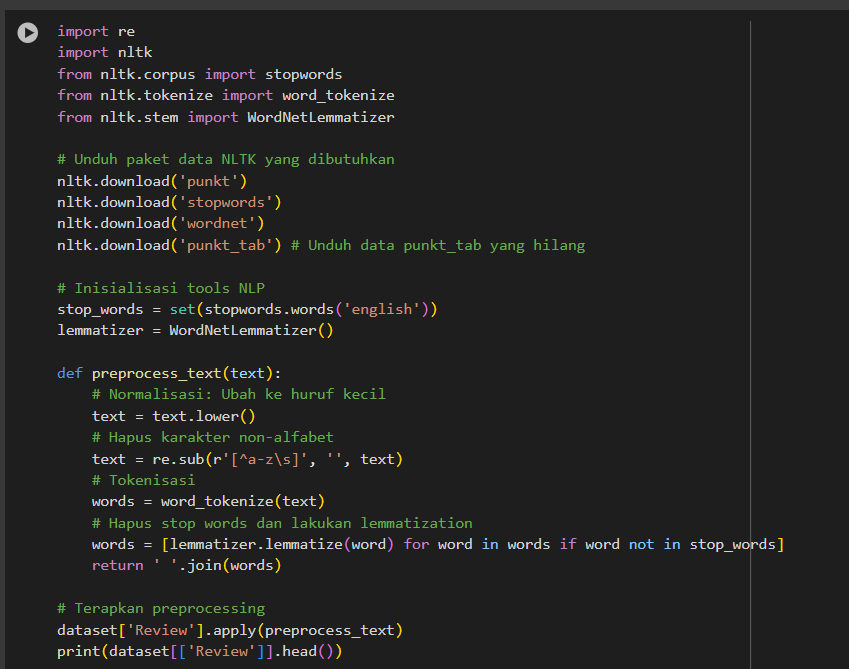
Gambar di atas menampilkan dua baris kode Python yang digunakan untuk memeriksa **data duplikat** dan **data yang hilang (missing values)** dalam DataFrame Pandas. Baris pertama menggunakan dataset.duplicated().sum() untuk menghitung jumlah baris duplikat, dan hasilnya menunjukkan terdapat **201 data duplikat**. Baris kedua menggunakan dataset.isnull().sum() untuk mengecek jumlah data hilang di setiap kolom, dengan hasil menunjukkan bahwa kolom **Review** dan **Rating** tidak memiliki data yang hilang (nilai 0 untuk keduanya). Dengan demikian, dapat disimpulkan bahwa meskipun tidak ada data hilang, terdapat sejumlah data duplikat yang dapat dibersihkan menggunakan metode seperti drop\_duplicates().

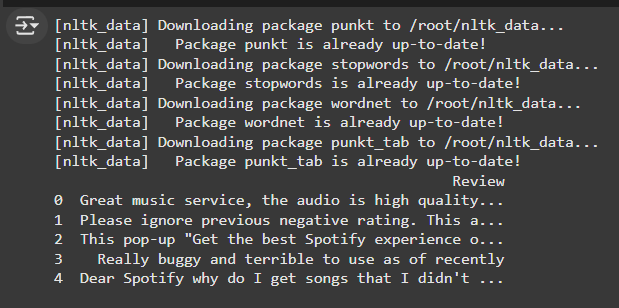
CLEANSING DATA



Kode di atas digunakan untuk menghapus data duplikat dari DataFrame Pandas dan memastikan bahwa dataset telah bersih. Metode drop\_duplicates() digunakan untuk menghapus semua baris duplikat dengan hanya menyisakan baris unik, kemudian hasilnya disimpan kembali ke variabel dataset. Setelah proses tersebut, dilakukan pengecekan ulang dengan menggunakan dataset.duplicated().sum() untuk menghitung jumlah baris duplikat yang tersisa. Hasilnya menunjukkan bahwa jumlah data duplikat setelah pembersihan adalah **0**, yang berarti dataset sudah bebas dari duplikat dan siap untuk dianalisis lebih lanjut.

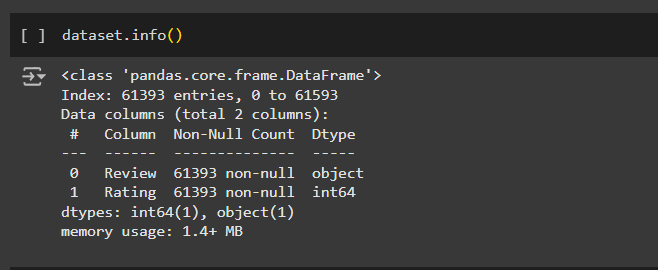
PROSES DATA





Kode di atas melakukan proses **pemrosesan teks** untuk membersihkan dan menyiapkan data dalam kolom **Review** pada DataFrame menggunakan teknik pemrosesan bahasa alami (NLP) dengan bantuan pustaka **NLTK**. Langkah pertama dalam kode ini adalah mengunduh beberapa paket yang diperlukan dari **NLTK**, seperti **punkt** untuk tokenisasi, **stopwords** untuk menghapus kata-kata umum, **wordnet** untuk lemmatization, dan **punkt\_tab** untuk memastikan tokenisasi berjalan dengan baik. Setelah paket diunduh, kode melanjutkan dengan menginisialisasi **stop words** dan **lemmatizer** yang digunakan dalam proses pembersihan teks. Fungsi **preprocess\_text()** kemudian diterapkan pada setiap ulasan dalam kolom **Review**. Fungsi ini pertama-tama mengubah teks menjadi huruf kecil untuk normalisasi, menghapus karakter non-alfabet menggunakan ekspresi reguler, dan kemudian melakukan **tokenisasi** untuk memisahkan teks menjadi kata-kata. Setelah tokenisasi, fungsi ini menghapus **stop words** (kata-kata umum yang tidak memberikan banyak informasi dalam analisis), lalu melakukan **lemmatization** untuk mengembalikan kata ke bentuk dasarnya, sehingga kata seperti "running" menjadi "run". Hasilnya adalah teks yang lebih bersih dan siap digunakan untuk analisis lebih lanjut. Setelah menerapkan fungsi tersebut pada kolom **Review**, hasilnya adalah ulasan yang telah diproses dan siap untuk langkah analisis berikutnya. Contoh ulasan dalam DataFrame menunjukkan bahwa teks telah diproses dengan menghapus kata-kata yang tidak diperlukan, menjadikannya lebih terstruktur dan berguna dalam analisis sentimen atau pemodelan lainnya.

CEK DATA INFO



Hasil dari perintah dataset.info() menunjukkan bahwa DataFrame memiliki dua kolom, yaitu **Review** dan **Rating**, dengan total **61.393 baris**. Kolom **Review** berisi ulasan pengguna dalam bentuk teks, dan kolom **Rating** berisi penilaian dalam bentuk angka (integer). Kedua kolom ini tidak memiliki nilai yang hilang, karena keduanya memiliki **61.393 entri non-null**. Tipe data kolom **Review** adalah **object**, yang berarti kolom ini menyimpan teks, sementara kolom **Rating** memiliki tipe data **int64**, yang menunjukkan bahwa ini adalah tipe data angka bulat. Dataset ini memakan **sekitar 1.4 MB** memori, yang mengindikasikan ukuran dataset secara keseluruhan. Dengan informasi ini, kita dapat menyimpulkan bahwa dataset sudah lengkap dan siap digunakan untuk analisis lebih lanjut.

PROSES DATA

import numpy as np

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

# Tokenisasi teks

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(dataset['Review'])

sequences = tokenizer.texts\_to\_sequences(dataset['Review'])

# Padding sequences

max\_len = 100

X = pad\_sequences(sequences, maxlen=max\_len)

# Label encoding

y = dataset['Rating'].apply(lambda x: 1 if x > 3 else 0).values  # Rating > 3 dianggap positif

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Model LSTM

model = Sequential([

    Embedding(input\_dim=len(tokenizer.word\_index) + 1, output\_dim=128, input\_length=max\_len),

    LSTM(128, return\_sequences=False),

    Dropout(0.2),

    Dense(64, activation='relu'),

    Dropout(0.2),

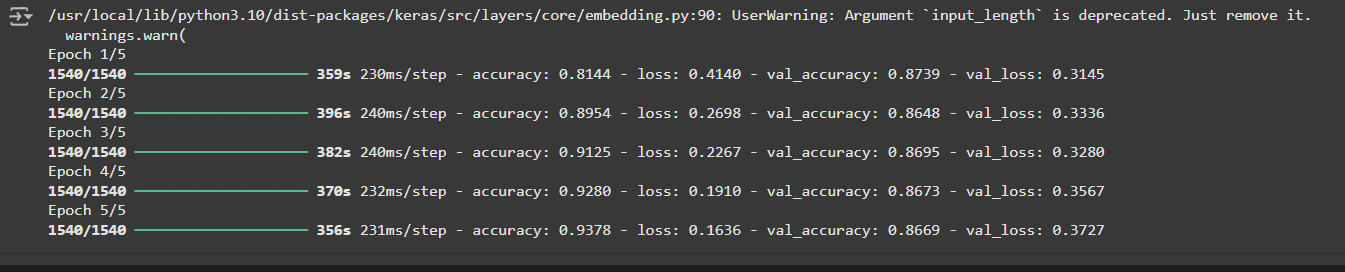
    Dense(1, activation='sigmoid')  # Output untuk klasifikasi biner

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train model

history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=5, batch\_size=32)



Kode di atas digunakan untuk membangun dan melatih model **LSTM (Long Short-Term Memory)** untuk analisis sentimen berdasarkan ulasan pengguna. Berikut adalah penjelasan langkah demi langkah:

1. **Tokenisasi Teks**:
   * **Tokenizer** digunakan untuk mengonversi teks menjadi token (angka yang mewakili kata-kata). Fungsi fit\_on\_texts() digunakan untuk membangun kamus kata dan mengonversi teks menjadi urutan angka. Kemudian, texts\_to\_sequences() mengubah ulasan menjadi urutan angka berdasarkan kamus yang sudah dibangun.
2. **Padding Sequences**:
   * Agar setiap ulasan memiliki panjang yang sama, digunakan **padding** dengan pad\_sequences(). Setiap urutan angka yang lebih pendek dari panjang maksimum (max\_len = 100) akan dipenuhi dengan angka nol di awal, sehingga semua urutan memiliki panjang yang konsisten.
3. **Label Encoding**:
   * Kolom **Rating** diklasifikasikan menjadi dua kelas: **positif (1)** jika rating lebih besar dari 3 dan **negatif (0)** jika rating 3 atau kurang. Hal ini dilakukan dengan fungsi apply() dan lambda untuk memetakan rating ke dalam dua kelas biner.
4. **Split Data**:
   * Dataset dibagi menjadi **training set (80%)** dan **test set (20%)** menggunakan train\_test\_split(). Ini memungkinkan pelatihan model pada sebagian data dan evaluasi model pada bagian yang tidak terlihat sebelumnya.
5. **Model LSTM**:
   * Model **Sequential** dibangun dengan beberapa lapisan:
     + **Embedding**: Mengubah urutan kata menjadi vektor berdimensi 128.
     + **LSTM**: Lapisan LSTM dengan 128 unit, yang dirancang untuk memproses data sekuensial seperti teks.
     + **Dropout**: Lapisan regularisasi yang mengurangi overfitting dengan menghapus secara acak beberapa neuron.
     + **Dense**: Lapisan sepenuhnya terhubung yang menghasilkan output akhir dengan fungsi aktivasi **sigmoid**, cocok untuk klasifikasi biner.
   * Model ini dikompilasi menggunakan **Adam optimizer**, **binary crossentropy** sebagai fungsi loss, dan **accuracy** sebagai metrik evaluasi.
6. **Training**:
   * Model dilatih dengan data **X\_train** dan **y\_train** selama 5 epoch dengan ukuran batch 32. Selama pelatihan, data validasi dari **X\_test** dan **y\_test** digunakan untuk memantau kinerja model di luar data pelatihan.

EVALUASI MODEL

from sklearn.metrics import classification\_report, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Prediksi pada data test

y\_pred = (model.predict(X\_test) > 0.5).astype(int)

# Evaluasi

print(classification\_report(y\_test, y\_pred))

# Confusion Matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

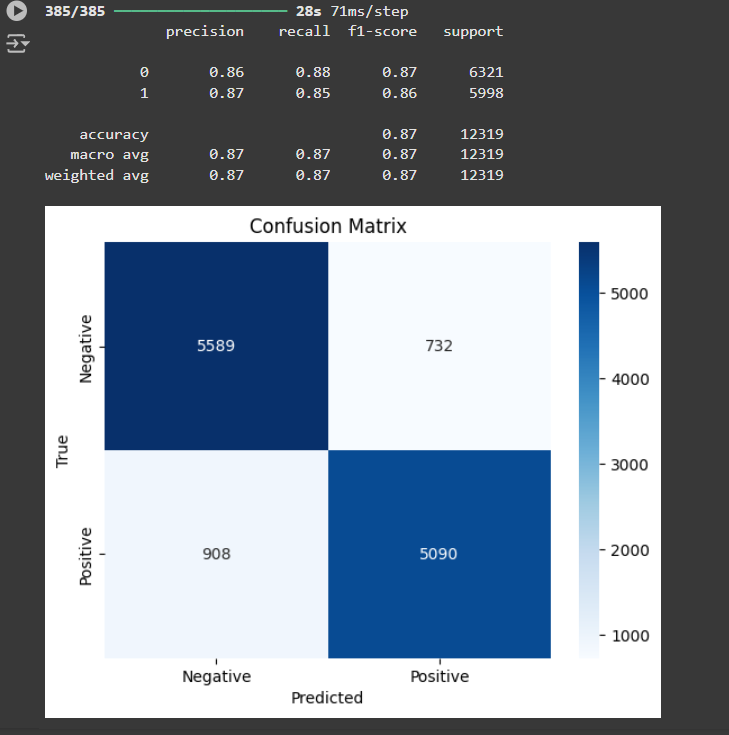
sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])

plt.xlabel('Predicted')

plt.ylabel('True')

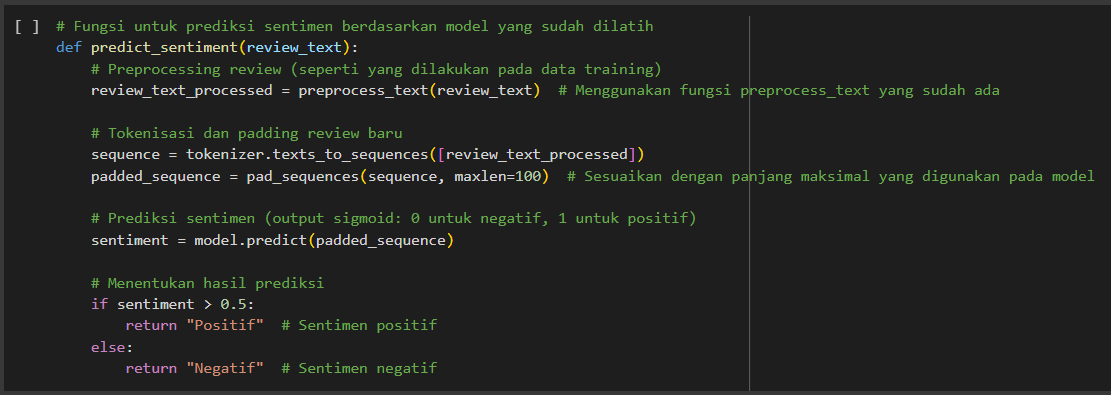
plt.title('Confusion Matrix')

plt.show()



Potongan kode yang diberikan digunakan untuk mengevaluasi kinerja model pembelajaran mesin pada tugas klasifikasi biner. Pertama, model membuat prediksi pada data uji menggunakan model yang telah dilatih, lalu mengevaluasi kinerja model menggunakan dua metrik umum: laporan klasifikasi dan matriks kebingungan.Laporan klasifikasi memberikan rincian runtut tentang presisi, recall, skor F1, dan dukungan model untuk setiap kelas. Informasi ini membantu menilai akurasi keseluruhan model dan mengidentifikasi bias spesifik kelas. Matriks kebingungan adalah representasi visual dari prediksi model dibandingkan dengan label sebenarnya. Ini menunjukkan jumlah true positif, true negatif, false positif, dan false negatif. Matriks kebingungan kemudian divisualisasikan menggunakan pustaka Seaborn, yang membuat heatmap yang memudahkan untuk menginterpretasikan kinerja model. Kode ini pertama membuat prediksi pada data uji menggunakan model yang dilatih, kemudian mencetak laporan klasifikasi, dan terakhir membuat visualisasi matriks kebingungan. Plot yang dihasilkan memberikan pemahaman yang jelas tentang kekuatan dan kelemahan model, yang dapat digunakan untuk lebih meningkatkan model atau memandu upaya pengembangan selanjutnya.

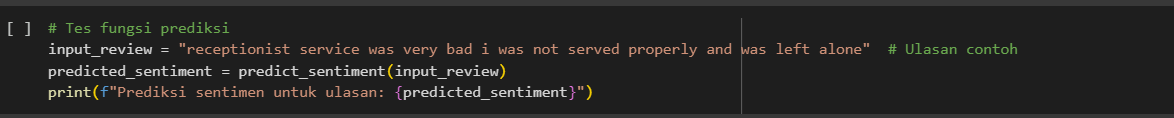
**PREDIKSI SENTIMEN**

****

Fungsi **predict\_sentiment()** di atas digunakan untuk memprediksi sentimen dari ulasan teks baru berdasarkan model LSTM yang telah dilatih. Berikut adalah penjelasan langkah demi langkah tentang cara kerja fungsi ini:

1. **Preprocessing Teks Ulasan**:
   * Teks ulasan yang diberikan (review\_text) diproses dengan menggunakan fungsi **preprocess\_text()** yang telah didefinisikan sebelumnya. Fungsi ini mengubah teks menjadi huruf kecil, menghapus karakter non-alfabet, dan melakukan tokenisasi serta lemmatization. Proses ini memastikan bahwa teks ulasan baru dipersiapkan dengan cara yang sama seperti data pelatihan.
2. **Tokenisasi dan Padding**:
   * Setelah ulasan diproses, teks tersebut diubah menjadi urutan angka (token) menggunakan **tokenizer.texts\_to\_sequences()** yang telah dilatih sebelumnya pada dataset. Kemudian, urutan angka tersebut diproses dengan **pad\_sequences()** untuk memastikan panjangnya sesuai dengan panjang input yang digunakan dalam model (100 kata, sesuai dengan pengaturan **maxlen** yang digunakan saat pelatihan).
3. **Prediksi Sentimen**:
   * Model yang telah dilatih digunakan untuk memprediksi sentimen dari teks ulasan yang telah diproses. Fungsi **model.predict()** menghasilkan nilai probabilitas antara 0 dan 1 yang menunjukkan kemungkinan sentimen positif.
   * Jika hasil prediksi lebih besar dari 0.5, maka model mengklasifikasikan ulasan sebagai **positif**, sebaliknya jika kurang dari atau sama dengan 0.5, maka ulasan diklasifikasikan sebagai **negatif**.
4. **Menentukan Hasil Prediksi**:
   * Berdasarkan hasil prediksi probabilitas, fungsi ini mengembalikan hasil **"Positif"** jika prediksi lebih besar dari 0.5, dan **"Negatif"** jika prediksi kurang dari atau sama dengan 0.5.

**TES FUNGSI**

****

Kode di atas menguji fungsi **predict\_sentiment()** dengan memberikan contoh ulasan input untuk melihat bagaimana model memprediksi sentimen dari ulasan tersebut. Berikut adalah penjelasan langkah-langkahnya:

1. **Ulasan Contoh**:
   * Sebuah ulasan contoh dimasukkan ke dalam variabel **input\_review**. Ulasan ini berisi kalimat **"receptionist service was very bad i was not served properly and was left alone"**, yang menggambarkan pengalaman negatif dengan layanan resepsionis.
2. **Prediksi Sentimen**:
   * Fungsi **predict\_sentiment(input\_review)** dipanggil dengan ulasan contoh tersebut sebagai argumen. Fungsi ini akan memproses teks, mengonversinya menjadi urutan token, melakukan padding, dan akhirnya memprediksi sentimen menggunakan model yang telah dilatih sebelumnya.
3. **Menampilkan Hasil**:
   * Hasil dari prediksi sentimen disimpan dalam variabel **predicted\_sentiment**. Kemudian, hasil prediksi ditampilkan dengan mencetak kalimat yang menunjukkan sentimen yang diprediksi (apakah **"Positif"** atau **"Negatif"**).

Jika ulasan contoh tersebut memiliki sentimen yang negatif, fungsi ini akan mencetak bahwa sentimen yang diprediksi adalah **"Negatif"**. Sebaliknya, jika model menganggap ulasan tersebut positif, maka yang dicetak adalah **"Positif"**.

**UJI FUNGSI PREDIKSI**

# Fungsi untuk prediksi sentimen untuk banyak ulasan sekaligus

def predict\_sentiments\_batch(reviews):

    sentiments = []  # Menyimpan hasil prediksi

    for review in reviews:

        sentiment = predict\_sentiment(review)  # Memanggil fungsi prediksi sentimen untuk setiap ulasan

        sentiments.append(sentiment)

    return sentiments

# Daftar ulasan untuk diuji

input\_reviews = [

    "The hotel was amazing, had a great stay!",

    "Worst experience ever, the room was dirty and uncomfortable.",

    "Nice place, but the service could be better.",

    "I loved the food, and the staff were friendly!",

    "The location was terrible, very noisy at night."

]

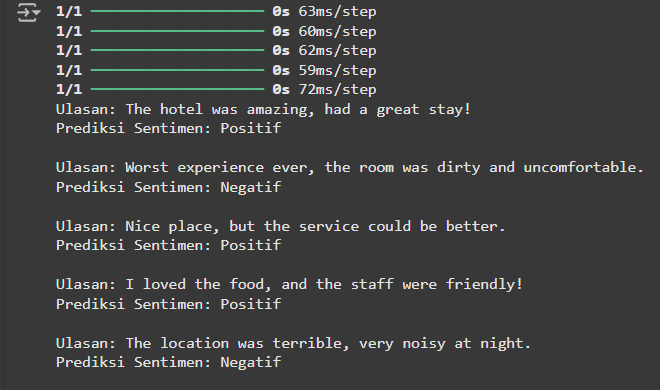
# Prediksi sentimen untuk banyak ulasan sekaligus

predicted\_sentiments = predict\_sentiments\_batch(input\_reviews)

# Menampilkan hasil prediksi untuk setiap ulasan

for review, sentiment in zip(input\_reviews, predicted\_sentiments):

    print(f"Ulasan: {review}\nPrediksi Sentimen: {sentiment}\n")

****

Fungsi **predict\_sentiments\_batch()** di atas digunakan untuk memprediksi sentimen dari banyak ulasan sekaligus. Berikut adalah penjelasan langkah demi langkah tentang cara kerjanya:

1. **Daftar Ulasan untuk Diuji**:
   * Daftar ulasan **input\_reviews** berisi beberapa teks ulasan yang akan diprediksi sentimennya. Ulasan-ulasan ini mencakup berbagai pengalaman, dari yang positif hingga yang negatif.
2. **Prediksi Sentimen untuk Banyak Ulasan**:
   * Fungsi **predict\_sentiments\_batch()** menerima daftar ulasan (reviews) sebagai input dan melakukan iterasi melalui setiap ulasan.
   * Untuk setiap ulasan dalam daftar, fungsi memanggil **predict\_sentiment()** untuk mendapatkan sentimen prediksi (positif atau negatif). Hasil prediksi ini disimpan dalam daftar **sentiments**.
3. **Menampilkan Hasil Prediksi**:
   * Setelah mendapatkan hasil prediksi untuk semua ulasan, program mencetak **ulasan** bersama dengan **prediksi sentimen** yang sesuai. Setiap ulasan akan dipasangkan dengan hasil prediksi, yang memungkinkan Anda untuk melihat bagaimana model mengklasifikasikan sentimen dari berbagai ulasan.

# KESIMPULAN

Berdasarkan *confusion matrix* yang ditampilkan:

1. **Akurasi Model**  
   Model memiliki akurasi sebesar **87%**, yang berarti 87% dari total prediksi berhasil diklasifikasikan dengan benar.
2. **Kinerja pada Kelas Sentimen**
   * **Sentimen Negatif (Label 0):**
     + **Precision**: 86%  
       Ini menunjukkan bahwa 86% prediksi sebagai sentimen negatif benar-benar negatif.
     + **Recall**: 88%  
       Sebanyak 88% ulasan negatif terdeteksi dengan benar.
   * **Sentimen Positif (Label 1):**
     + **Precision**: 87%  
       Ini berarti 87% prediksi sebagai sentimen positif benar-benar positif.
     + **Recall**: 85%  
       Sebanyak 85% ulasan positif berhasil diklasifikasikan dengan benar.
3. **Detail Confusion Matrix**
   * Sebanyak **5589 ulasan negatif** diklasifikasikan dengan benar sebagai negatif.
   * Sebanyak **5090 ulasan positif** diklasifikasikan dengan benar sebagai positif.
   * Terdapat **732 ulasan negatif** yang salah diklasifikasikan sebagai positif.
   * Terdapat **908 ulasan positif** yang salah diklasifikasikan sebagai negatif.
4. **Kesimpulan Umum**  
   Model menunjukkan performa yang **cukup baik** dalam mengklasifikasikan sentimen ulasan pengguna aplikasi Spotify, dengan distribusi kesalahan yang relatif seimbang antara kelas positif dan negatif. Namun, model sedikit lebih baik dalam mendeteksi ulasan negatif dibandingkan ulasan positif, yang ditunjukkan oleh perbedaan *recall*.

Untuk peningkatan lebih lanjut, model dapat dioptimalkan dengan:

* Menyeimbangkan jumlah data pada setiap kelas jika terjadi *class imbalance*.
* Menggunakan teknik *fine-tuning* atau *hyperparameter optimization*