**TUGAS 1**

INTEGRASI DECISION TREE DAN LSTM UNTUK PREDIKSI STRES DAN ANALISIS SENTIMEN KESEHATAN MENTAL DENGAN DATASET “MENTAL HEALTH DATASET” DAN “SENTIMENT ANALYSIS FOR MENTAL HEALTH”

**LAPORAN**

**Oleh:**

**Tim GCD**

**IF-A PAGI**

* **211110347 - CINDY SINTIYA**
* **211110948 - GRACE HELENA HUTAGAOL**
* **211111930 - DAVID BATE'E**



****

**PROGRAM STUDI S-1**

**FAKULTAS INFORMATIKA**

**TEKNIK INFORMATIKA**

**UNIVERITAS MIKROSKIL**

**2024/2025**

# DAFTAR ISI

[PENDAHULUAN 1](#_Toc181705775)

[Latar Belakang 1](#_Toc181705776)

[Tujuan 1](#_Toc181705777)

[PEMBAHASAN 3](#_Toc181705778)

[Detail Dataset 3](#_Toc181705779)

[1. Prediksi Stres 3](#_Toc181705780)

[1.1. Sumber Dataset 3](#_Toc181705781)

[1.2. Fitur Independen (x) 3](#_Toc181705782)

[1.3. Fitur Dependen (y) 4](#_Toc181705783)

[1.4. Ukuran Dataset 4](#_Toc181705784)

[2. Analisis Sentimen Kesehatan Mental 4](#_Toc181705785)

[2.1. Sumber Dataset 4](#_Toc181705786)

[2.2. Fitur Independen (x) 4](#_Toc181705787)

[2.3. Fitur Dependen (y) 4](#_Toc181705788)

[2.4. Ukuran Dataset 4](#_Toc181705789)

[Preprocessing & Train-Test Split 4](#_Toc181705790)

[1. Prediksi Stres 4](#_Toc181705791)

[1.1. Data Cleaning 4](#_Toc181705792)

[1.2. Feature & Label Encoding 5](#_Toc181705793)

[1.3. Feature Selection 5](#_Toc181705794)

[1.4. Train-Test Splitting 6](#_Toc181705795)

[2. Analisis Sentimen 6](#_Toc181705796)

[2.1. Data Cleaning 6](#_Toc181705797)

[2.2. Data Balancing 6](#_Toc181705798)

[2.3. Label Encoding 7](#_Toc181705799)

[2.4. Train-Test Splitting 7](#_Toc181705800)

[2.5. Tokenization 7](#_Toc181705801)

[2.6. Pad Sequences 8](#_Toc181705802)

[2.7. Batch Data Loading 8](#_Toc181705803)

[Pemilihan Algoritma dan Evaluasi Model 8](#_Toc181705804)

[1. Prediksi Stres 8](#_Toc181705805)

[1.1. Algoritma Decision Tree 8](#_Toc181705806)

[1.2. Evaluasi Model Decision Tree 8](#_Toc181705807)

[2. Analisis Sentimen 9](#_Toc181705808)

[2.1. Algoritma LSTM 9](#_Toc181705809)

[2.2. Evaluasi Model LSTM 9](#_Toc181705810)

[PENGEMBANGAN APLIKASI 10](#_Toc181705811)

[Requirements 10](#_Toc181705812)

[1. Versi Notebook 10](#_Toc181705813)

[2. Versi Aplikasi *Web* (*end-user*) 11](#_Toc181705814)

[3. Versi Aplikasi *Mobile* (*end-user*) 11](#_Toc181705815)

[User Interface 12](#_Toc181705816)

[PENUTUP 13](#_Toc181705817)

[Kesimpulan 13](#_Toc181705818)

[Rekomendasi 13](#_Toc181705819)

[Link Video Presentasi 13](#_Toc181705820)

[Link Notebook 13](#_Toc181705821)

[Link GitHub/ Source Code Lengkap 13](#_Toc181705822)

# PENDAHULUAN

## Latar Belakang

Latar belakang penelitian ini didasari oleh meningkatnya tantangan kesehatan mental di seluruh dunia, dimana tingkat stres dan kondisi mental lainnya terus meningkat secara signifikan. Dalam beberapa tahun terakhir, perubahan sosial yang cepat, tekanan lingkunagan, dan dampak luas dari pandemi COVID-19 yang berkontribusi pada peningkatan stres juga membuat kebutuhan akan dukungan kesehatan mental yang mudah diakses. Namun, layanan kesehatan mental yang ada masih terbatas, terutama dalam hal alat yang dipersonalisasi dan berbasis data yang mudah diakses oleh masyarakat umum.

Pendekatan tradisional untuk memantau kesehatan mental biasanya bergantung pada penilaian berkala oleh profesional, yang cakupannya bisa terbatas dan kurang efisien untuk dilakukan. Dengan kemajuan *Machine Learning*, terdapat peluang untuk memanfaatkan data dari interaksi harian, seperti posting media sosial atau pengumpulan formulir, guna mendapatkan wawasan berkelanjutan tentang kondisi mental seseorang. Melalui integrasi prediksi stres menggunakan Decision Tree dan analisis sentimen menggunakan LSTM, penelitian ini bertujuan untuk menjembatani kesenjangan antara dukungan kesehatan mental klinis dan pelacakan kesejahteraan mental sehari-hari.

Pemilihan algoritma ini juga dilakukan dengan pertimbangan khusus, yaitu Decision Tree yang menawarkan ketahanan dalam menangani data terstruktur terkait stres, sementara LSTM yang efektif untuk data sekuensial, sehingga ideal untuk analisis sentimen dalam data teks. Dengan menggabungkan kedua pendekatan ini, penelitian ini berupaya menciptakan alat yang komprehensif, yang tidak hanya memprediksi tingkat stres tetapi juga memberikan wawasan lebih dalam tentang tren kesehatan mental, sehingga memfasilitasi intervensi dini dan mendukung perawatan kesehatan mental yang proaktif.

## Tujuan

1. Memprediksi apakah seseorang mengalami stres atau tidak melalui beberapa variabel, seperti pekerjaan, sosialisasi, riwayat gangguan mental sebelumnya, dan sebagainya.
2. Memprediksi seberapa besar kemungkinan seseorang mengalami gangguan mental, seperti depresi, gangguan kecemasan, kecenderungan bunuh diri, dan sebagainya.
3. Menggabungkan kedua model dalam 1 aplikasi untuk membantu pengguna mendeteksi adanya stres ataupun gangguan kesehatan mental bagi mereka yang takut berkonsultasi langsung dengan orang lain ataupun ahli medis profesional.

# PEMBAHASAN

## Detail Dataset

Sumber dataset utama yang digunakan berasal dari kaggle. Dataset tambahan juga sudah dicari dari *website-website* lain untuk menyeimbangkan beberapa kelas data pada dataset “sentiment.csv”, namun ternyata dataset yang di kaggle sudah gabungan dari banyak *website* sehingga diputuskan untuk menggunakan dataset dari kaggle saja.

### Prediksi Stres

#### Sumber Dataset

[kaggle.com/datasets/bhavikjikadara/mental-health-dataset](https://www.kaggle.com/datasets/bhavikjikadara/mental-health-dataset)

#### Fitur Independen (x)

|  |  |  |
| --- | --- | --- |
| *Timestamp* | : | waktu pengiriman form |
| *Gender* | : | jenis kelamin (*Male/ Female*) |
| *Country* | : | negara |
| *Occupation* | : | pekerjaan (*Business/ Corporate/ Housewife/ Student/ Other*) |
| *self\_employed* | : | wiraswasta (*Yes/ No*) |
| *family\_history* | : | apakah ada anggota keluarga dengan riwayat gangguan mental (*Yes/ No*) |
| *treatment* | : | apakah sudah pernah mencari pengobatan untuk masalah gangguan kesehatan mental (*Yes/ No*) |
| *Days\_Indoors* | : | sudah berapa lama stay dirumah saja (*Go out Every day/ 1-14 days/ 15-30 days/ 31-60 days/ More than 2 months*) |
| *Changes\_Habits* | : | perubahan kebiasaan (*Yes/ No/ Maybe*) |
| *Mental\_Health\_History* | : | riwayat gangguan mental (*Yes/ No/ Maybe*) |
| *Mood\_Swings* | : | perubahan mood/ suasana hati (*Low/ Medium/ High*) |
| *Coping\_Struggles* | : | apakah berjuang/ berusaha mengatasinya (*Yes/ No*) |
| *Work\_Interest* | : | sangat terikat dengan pekerjaan (*Yes/ No/ Maybe*) |
| *Social\_Weakness* | : | lemah dalam bersosialisasi (*Yes/ No/ Maybe*) |
| *mental\_health\_interview* | : | apakah akan menceritakan terkait masalah kesehatan saat *interview* kerja (*Yes/ No/ Maybe*) |
| *care\_options* | : | kepedulian terhadap masalah (*Yes/ No/ Not Sure*) |

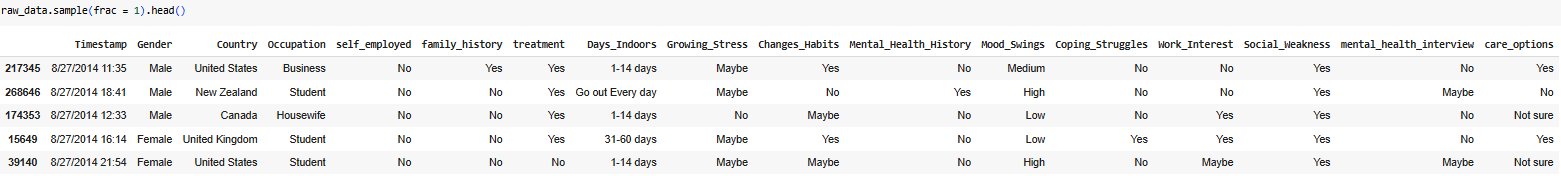
#### Fitur Dependen (y)

*Growing\_Stress* (*Yes/ No/ Maybe*)

#### Ukuran Dataset

292.364 baris data, 17 kolom fitur

#### Preview Dataset



### Analisis Sentimen Kesehatan Mental

#### Sumber Dataset

[kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health](https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health)

#### Fitur Independen (x)

*statement* : sentimen dalam bentuk teks kalimat

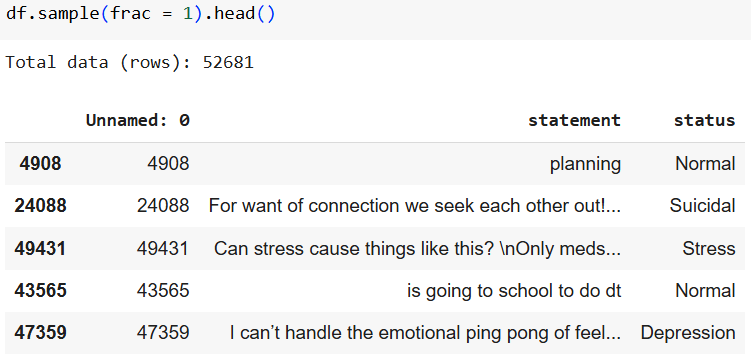
#### Fitur Dependen (y)

*status* (*Normal/ Depression/ Suicidal/ Anxiety/ Stress/ Bipolar/ Personality disorder*)

#### Ukuran Dataset

53.043 baris data, 2 kolom fitur (+ 1 kolom *index*)

#### Preview Dataset



## Preprocessing & Train-Test Split

### Prediksi Stres

#### Data Cleaning

Pembersihan data adalah tahapan pertama yang wajib dilakukan untuk dataset yang sudah dikumpulkan. Untuk dataset “stress.csv” yang diperoleh dari kaggle berisi 292.364 baris data terlihat cukup mencurigakan karena jumlah data yang sangat banyak.

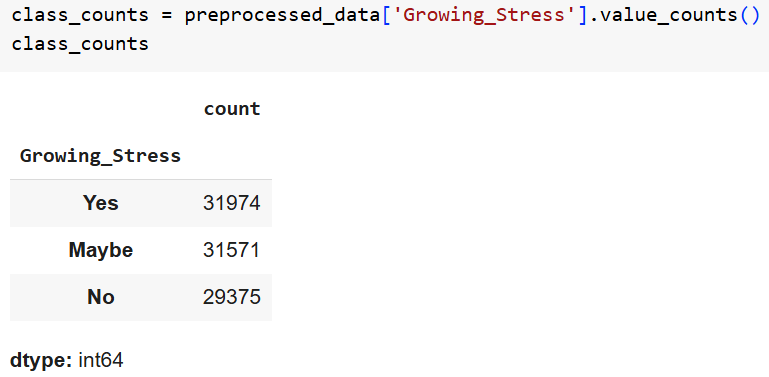
Bagian pertama yang dilakukan adalah membuang baris data yang berisi nilai kosong (*NaN*) karena data tersebut dianggap *invalid* dan akan mengganggu proses *build* model.

Selanjutnya, pengecekan baris data duplikat juga dilakukan untuk membuang data yang sama persis agar mencegah model *overfitting* dengan konsumsi data yang sama berulang kali. Dan hasilnya, hanya tersisa 92.920 baris data valid.

|  |  |
| --- | --- |
| *Before* | *After* |
|  |  |
|  |  |

Bagian terakhir yang dilakukan adalah mengecek apakah adanya ketidakseimbangan (*imbalanced*) kelas data agar mencegah model terlalu condong ke salah satu atau beberapa kelas dengan jumlah data mayoritas dan mengabaikan kelas yang memiliki jumlah data yang lebih sedikit.

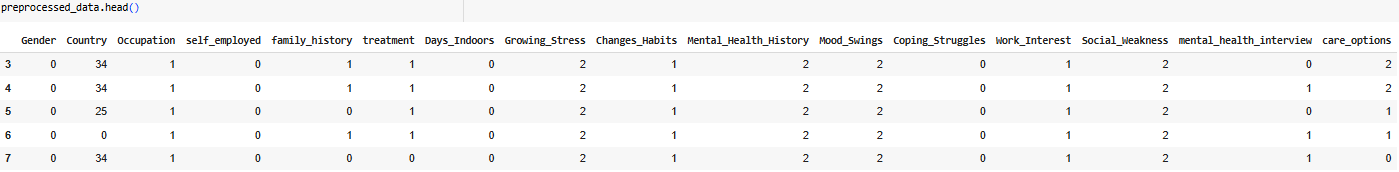
Untuk dataset “stress.csv”, jumlah data untuk tiap kelas sudah seimbang, dengan masing-masing kelas lebih kurang 30.000 data sehingga *data balancing* tidak perlu dilakukan lagi.



#### Feature & Label Encoding

Model *machine learning* biasanya menerima data input dalam bentuk angka. Dengan data numerik pula, model dapat belajar lebih baik dan memprediksi dengan lebih akurat.

Dataset yang digunakan berisi nilai-nilai kategorikal yang perlu diubah terlebih dahulu ke dalam format numerik, yang akan dilakukan dengan mengkodekan (*encoding*) masing-masing kolom menjadi bentuk bilangan bulat (0, 1, 2, ...), termasuk juga fitur independen (y) yang dilabeli sebagai 1, 2, dan 3.



|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Feature Selection

Diantara 17 kolom yang ada, kolom “*Growing\_Stress*” dipilih sebagai fitur dependen (y) yang akan dijadikan kelas hasil prediksi model dengan nilai *Yes*, *No*, dan *Maybe* untuk menentukan apakah seseorang mengalami stres.

Untuk fitur independen (x) akan dipilih dari 16 kolom yang tersisa. Kolom *Timestamp* yang merupakan kolom waktu pengumpulan/ pengiriman formulir akan dihapus karena dianggap tidak relavan.

Beberapa percobaan juga sudah dilakukan untuk memilih kombinasi kolom-kolom terbaik yang akan dijadikan fitur x, sehingga tersisa kolom-kolom berikut:

|  |  |  |
| --- | --- | --- |
| *Occupation* | : | pekerjaan (*Business/ Corporate/ Housewife/ Student/ Other*) |
| *treatment* | : | apakah sudah pernah mencari pengobatan untuk masalah gangguan kesehatan mental (*Yes/ No*) |
| *Days\_Indoors* | : | sudah berapa lama stay dirumah saja (*Go out Every day/ 1-14 days/ 15-30 days/ 31-60 days/ More than 2 months*) |
| *Changes\_Habits* | : | perubahan kebiasaan (*Yes/ No/ Maybe*) |
| *Mental\_Health\_History* | : | riwayat gangguan mental (*Yes/ No/ Maybe*) |
| *Mood\_Swings* | : | perubahan mood/ suasana hati (*Low/ Medium/ High*) |
| *Coping\_Struggles* | : | apakah berjuang/ berusaha mengatasinya (*Yes/ No*) |
| *Work\_Interest* | : | sangat terikat dengan pekerjaan (*Yes/ No/ Maybe*) |
| *Social\_Weakness* | : | lemah dalam bersosialisasi (*Yes/ No/ Maybe*) |

#### Train-Test Splitting

Dataset yang sudah bersih kemudian akan dibagi dengan rasio 7:3, di mana 70% data akan dipakai untuk *training* dan sisa 30% lainnya akan dipakai sebagai data *testing* dengan *random state* 1.000 untuk mempertahankan variasi pengacakan setiap kali *code* dijalankan.

### Analisis Sentimen

#### Data Cleaning

Pembersihan data adalah tahapan pertama yang wajib dilakukan untuk dataset yang sudah dikumpulkan. Untuk dataset “sentiments.csv” yang diperoleh dari kaggle berisi 53.043 baris data.

Bagian pertama yang dilakukan adalah membuang baris data yang berisi nilai kosong (*NaN*) karena data tersebut dianggap *invalid* dan akan mengganggu proses *build* model.

Selanjutnya, pengecekan baris data duplikat juga dilakukan untuk membuang data yang sama persis agar mencegah model *overfitting* dengan konsumsi data yang sama berulang kali. Hasilnya, tidak terdapat data duplikat dan jumlah data tersisa 52.681 baris data valid.

#### Data Balancing

Pengecekan jumlah data tiap kelas dilakukan untuk mencegah model terlalu condong ke salah satu atau beberapa kelas dengan jumlah data mayoritas dan mengabaikan kelas yang memiliki jumlah data yang lebih sedikit.

Hasil pengecekan menunjukkan bahwa jumlah data dari tiap kelas memiliki *range* yang terlalu jauh mulai dari 16.000-an data untuk kelas *Normal* hingga hanya 1.000-an data untuk kelas *Personality disorder* yang menunjukkan bahwa terdapat imbalanced data untuk kelas *Anxiety*, *Bipolar*, *Stres*, dan *Personality disorder*.

Untuk menyeimbangkan data tiap kelas, salah satu cara yang dipilih adalah *Filtering* atau *Class Removal*, dengan menghapus data pada kelas minoritas dan menyisakan kelas mayoritas agar model tetap dapat memprediksi dari dataset dengan jumlah yang seimbang.

#### Label Encoding

Masing-masing kelas yang masih bersifat kategorikal akan di-*encode* menjadi label numerik dalam bentuk bilangan bulat dengan label 0 untuk *Normal*, 1 untuk *Depression*, dan 2 untuk *Suicidal*.

Proses *encoding* dilakukan secara manual bukan menggunakan *library* karena hasil *training* menunjukkan bahwa akurasi model dengan *encoding* manual lebih bagus.

#### Train-Test Splitting

Tahapan preprocessing data belum selesai, namun pembagian data *training* dan *testing* dilakukan terlebih dahulu dengan beberapa pertimbangan sebagai berikut:

1. Mempertahankan distribusi data *training* dan *testing*
2. Mencegah kebocoran data dimana data *training* dan *testing* akan ditokenisasi secara terpisah

Dataset akan dibagi dengan rasio 7:3, di mana 70% data akan dipakai untuk *training* dan sisa 30% lainnya akan dipakai sebagai data *testing* dengan *random state* 1.000 untuk mempertahankan variasi pengacakan setiap kali *code* dijalankan.

#### Tokenization

Perlakukan data teks sentimen tentu saja berbeda dengan data teks yang bersifat kategorikal yang bisa langsung dilakukan *label* *encoding*. Proses tokenisasi perlu dilakukan untuk mengkonversi data teks menjadi kumpulan/ rangkaian angka agar dapat diproses oleh model.

Setiap kata yang ditokenisasi dapat diibaratkan sebagai kamus *vocabulary*. Sebagai contoh, kalimat “I fell so happy today” akan ditokenisasi menjadi [2, 15, 1, 37, 8] dimana setiap angka mewakili kata tertentu dalam kamus.

#### Pad Sequences

Karena model seringkali mengharapkan input sekuensial dengan panjang data yang sama, *padding* perlu ditambahkan agar ukuran data yang dimasukkan memiliki panjang yang seragam.

Proses *padding* akan menambahkan nilai tambahan (biasa berupa nilai 0) pada data dengan panjang data yang lebih pendek sesuai dengan panjang maksimal yang sudah ditentukan.

Sebagai contoh, [2, 15, 1, 37, 8] dan [3, 29, 15] akan dimodifikasi menjadi [2, 15, 1, 37, 8] dan [3, 29, 15, 0, 0] jika panjang maksimal yang ditentukan adalah 5.

#### Batch Data Loading

*Batch data loading* dilakukan agar model dapat melakukan *training* pada beberapa data sekaligus dalam 1 *batch* untuk mengurangi dan mengefisiensikan memori dan waktu yang terpakai untuk *training* serta *testing* model.

Terkhusus untuk model yang dibangun dengan pytorch, pembagian *batch* harus dilakukan secara manual (jika di tensorflow, jumlah *batch* bisa langsung dimasukkan pada parameter yang tersedia).

## Pemilihan Algoritma dan Evaluasi Model

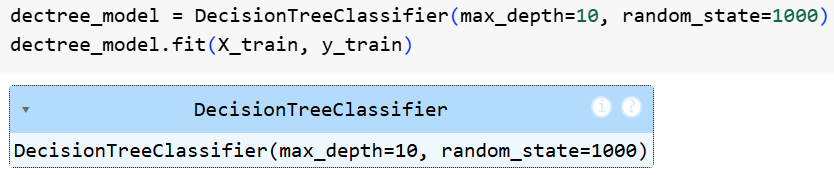
### Prediksi Stres

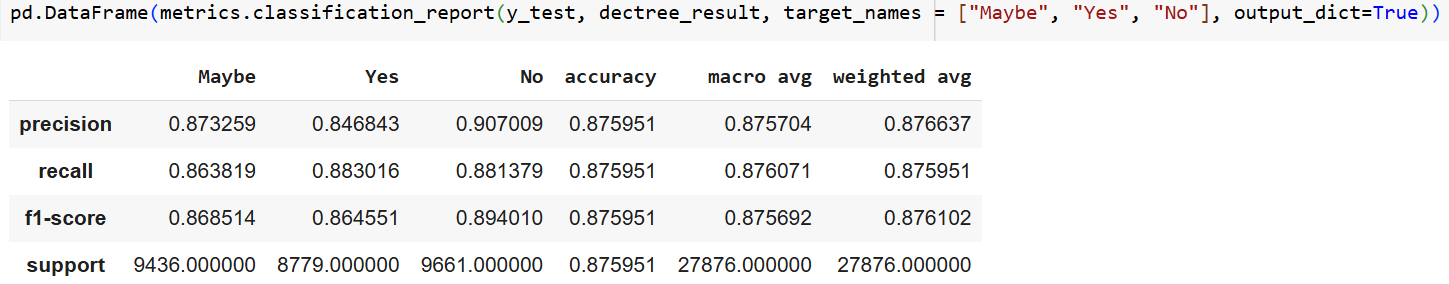
#### Algoritma Decision Tree

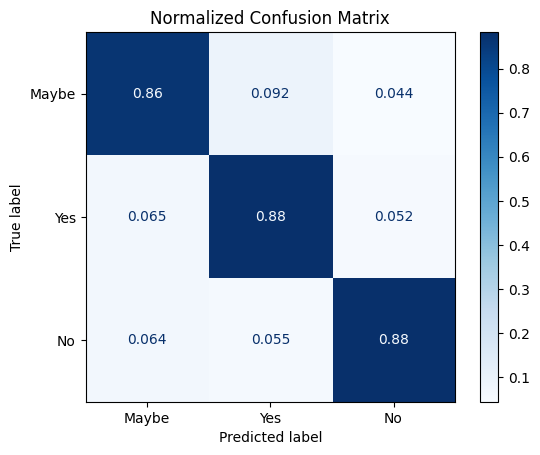
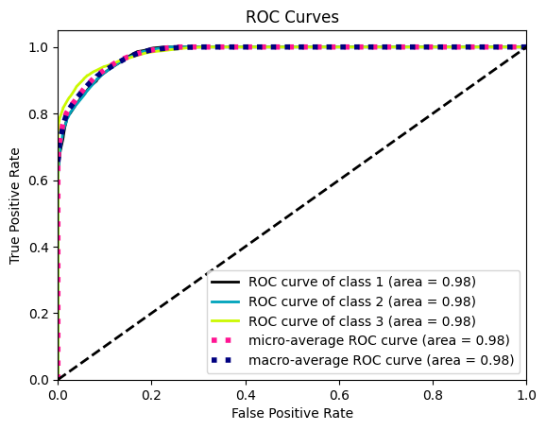
Percobaan dilakukan menggunakan beberapa algoritma, antara lain KNN, Decision Tree, dan Random Forest. Dari ketiga model yang dibangun, model Decision Tree dan Random Forest (setelah proses *hyperparameter tuning*) memberikan akurasi yang sama, sehingga diputuskan untuk memilih model Decision Tree karena memiliki keunggulan sebagai berikut:

1. Mudah dipahami, divisualisasi, dan diinterpretasikan karena berbasis “*If-Then*”
2. Proses *build* dan *training* model lebih cepat karena lebih simpel dan tidak sekompleks model Random Forest
3. Cocok untuk data yang memiliki korelasi antar variabel yang non-linear

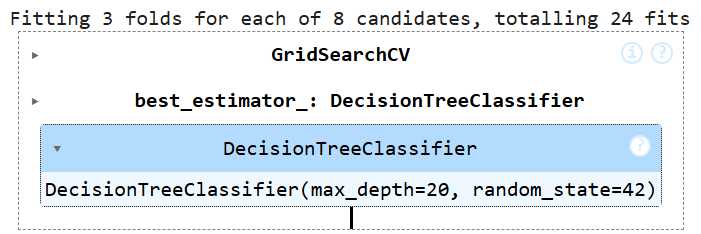
#### Evaluasi Model Decision Tree

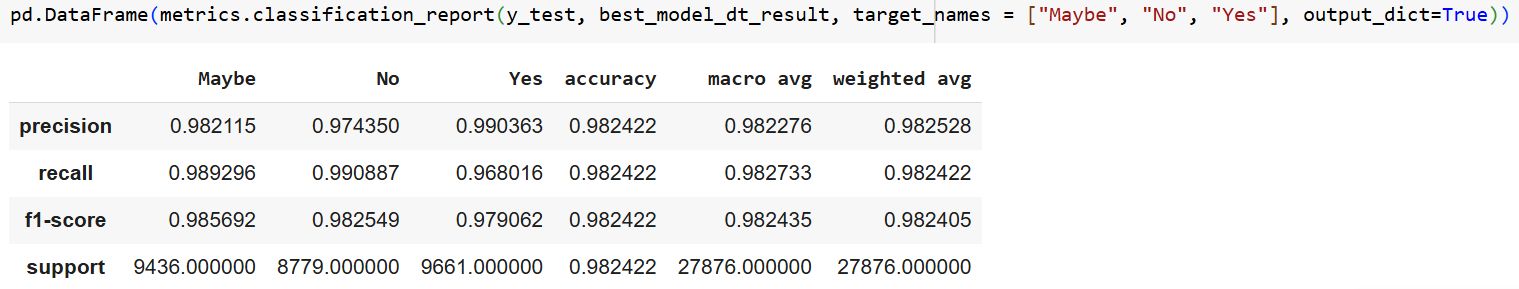


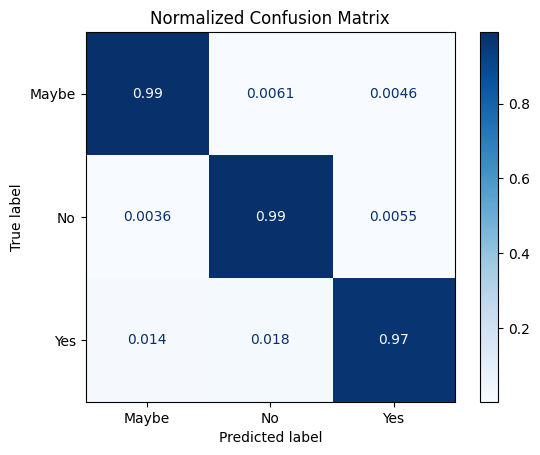
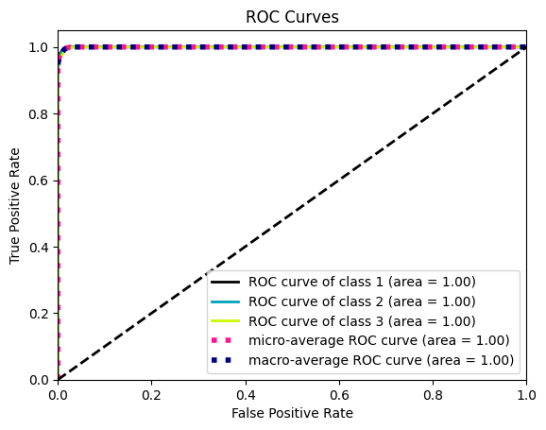


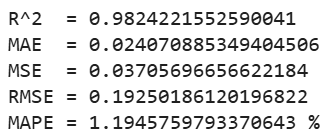
 

#### Hyperparameter Tuning







### Analisis Sentimen

#### Algoritma LSTM

Algoritma yang paling efektif dalam pemrosesan bahasa alami (NLP) termasuk analisis sentimen adalah LSTM (*Long-Short Term Memory*), yang sudah digunakan secara luas karena kinerjanya dalam menangkan konteks data sekuensial seperti urutan kata dalam teks yang mempengaruhi maknanya pula. Setiap urutan kata mengandung emosi dan pernyataan sentimen, sehingga mekanisme *memory gate* pada LSTM dapat sangat bermanfaat dalam memfokuskan pada bagian teks yang relavan dan meningkatkan sensitivitas terhadap perubahan emosi dan mendukung klasifikasi sentimen yang kuat.

#### Evaluasi Model LSTM

# PENGEMBANGAN APLIKASI

## Requirements

### Versi Notebook

Aplikasi yang dibangun membutuhkan model yang di-*generate* dari *notebook* agar mempercepat proses prediksi tanpa perlu melakukan proses *training* setiap kali prediksi dilakukan. Detail dan grafik evaluasi model juga dapat dilihat pada *file* *notebook* agar lebih rapi dan terstruktur. Berikut langkah-langkahnya:

1. Akses [link notebook Google Colab](link) atau bisa menggunakan *file* “ML\_Mental\_Health\_PT.ipynb” (model dibangun menggunakan pytorch) atau “ML\_Mental\_Health\_TF.ipynb” (model dibangun menggunakan tensorflow) yang ada di dalam folder “notebook”.

Disarankan untuk menggunakan *file* *Colab* agar menghemat waktu eksekusi dan *training* model menggunakan GPU yang telah disediakan oleh *Google Colab Compute Engine Backend*.

1. Siapkan API kaggle atau *file* dataset dalam format csv.

Untuk mengambil dataset dapat dilakukan pengunduhan secara manual melalui [link kaggle berikut](https://www.kaggle.com/datasets/bhavikjikadara/mental-health-dataset) ataupun menggunakan *file* “stress.csv” untuk prediksi stres dan juga [link kaggle berikut](https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health) ataupun menggunakan *file* “sentiments.csv” untuk analisis sentimen, yang sudah tersedia di folder “data”.

* Jika ingin mengunduh secara otomatis dari *notebook*, pastikan sudah menyiapkan API kaggle yang disimpan dalam *file* “kaggle.json”, lalu *upload* ke *runtime* *notebook* Colab.
* Jika menggunakan *file* yang sudah tersedia di folder “data”, pastikan sudah *upload* *file* dataset pada *runtime* *notebook* Colab dan melakukan *comment* dan *uncomment* pada *code* sesuai dengan video penjelasan yang ada.

1. Eksekusi 1 per 1 *cell* secara berurutan (opsional).

Jika ingin menggunakan model dari proses ini, pastikan sudah menjalankan semua *cell* (minimal hingga “Model Evaluation” untuk *save* keseluruhan model hasil *training* dan digunakan pada aplikasi *web* sebagai *pre-trained model*).

Step ini bersifat opsional karena *file* *pre-trained model* juga sudah tersedia pada folder “data” dan dapat langsung digunakan pada aplikasi *web*.

### Versi Aplikasi *Web* (*end-user*)

Model yang sudah diperoleh dari *notebook* akan digunakan sebagai *pre-trained model* untuk mempercepat proses prediksi. Untuk memudahkan pengguna menggunakan aplikasi, model akan diimplementasikan dalam bentuk aplikasi *web* yang lebih ringan, ringkas, dan dapat diakses darimana saja. Berikut langkah-langkah menjalankan aplikasi *web*:

1. Lakukan instalasi *library* yang tercatat dalam file “requirement.txt”

Jalankan

pip install -r requirement.txt

pada terminal dan tunggu hingga instalasi selesai.

Jika menggunakan Anaconda, pastikan *base* conda sudah aktif dengan menjalankan perintah berikut:

conda activate base

atau

C:/Users/<device name>/anaconda3/Scripts/activate

(sesuaikan dengan lokasi Anaconda perangkat Anda)

lalu jalankan:

conda install Flask Jinja2 flask-cors numpy pandas tensorflow torch scikit-learn tqdm deep-translator joblib

dan tunggu hingga instalasi selesai.

Atau bisa juga membaca file “README.md” yang ada untuk detail instalasi *library*.

1. Jika ingin mengganti model dengan yang baru di-*save* dari notebook, pastikan *file* model dan konfigurasi diletakkan di folder yang sama dengan lokasi *file* model awalnya (folder “data”) agar aplikasi dapat berjalan tanpa *error*.
2. Jalankan aplikasi *web* dengan perintah

python app.py

Jika menggunakan Anaconda, pastikan *base* conda sudah aktif sebelum menjalankan aplikasi.

1. Tunggu hingga aplikasi berhasil *running*, kemudian akses [http://127.0.0.1:5000](http://127.0.0.1:5000/) di *browser* atau sesuaikan dengan *link* yang muncul di terminal.
2. Aplikasi web siap digunakan.

### Versi Aplikasi *Mobile* (*end-user*)

Selain aplikasi web, aplikasi *mobile* juga semakin berkembang dan banyak digunakan karena mudah diakses dan tidak perlu menghapal *link* URL aplikasi agar dapat digunakan. Pengembangan aplikasi mobile sebagai lanjutan dari aplikasi web juga dilakukan agar lebih banyak pengguna dapat menikmati fitur dari model yang sudah dibangun. Berikut langkah-langkah menjalankan aplikasi *mobile*:

1. Buka terminal (cmd/ powershell), lalu jalankan perintah

flutter pub get

1. Untuk dapat menggunakan versi *mobile*, pastikan aplikasi *web* sudah aktif dan berjalan karena akan digunakan sebagai API untuk aplikasi *mobile*.
2. Jalankan aplikasi *mobile* dengan cara *debug* (tekan tombol F5 pada *keyboard*) atau eksekusi perintah

flutter run

pada terminal, lalu pilih *device* yang akan digunakan untuk *running* aplikasi *mobile* (selain *device* Android, bisa juga menggunakan *browser* seperti Chrome ataupun Edge untuk menjalankan aplikasi).

1. Masukkan IP Address/ IP Host dari aplikasi *web* yang sudah berjalan (IP dapat dilihat pada terminal aplikasi *web*) dengan menekan tombol *Setting* yang ada di bagian sudut kanan atas *interface*, lalu klik tombol “*Save and Try Again*”.
2. Tunggu hingga model selesai di-*fetch* dan di-*load* dari *backend* API *web,* dan aplikasi *mobile* siap digunakan.

## User Interface

# PENUTUP

## Kesimpulan

## Rekomendasi

## Link Video Presentasi

[Akses video presentasi GCD - Mental Health Sentiment Analysis disini](link)

## Link Notebook

[Akses notebook GCD - Mental Health Sentiment Analysis disini](link)

## Link GitHub/ Source Code Lengkap

[Akses GitHub GCD - Mental Health Sentiment Analysis disini](link)