**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](#_Toc182157066)

[Latar Belakang 1](#_Toc182157067)

[Tujuan 1](#_Toc182157068)

[PEMBAHASAN 3](#_Toc182157069)

[Detail Dataset 3](#_Toc182157070)

[1. Prediksi Stres 3](#_Toc182157071)

[1.1. Sumber Dataset 3](#_Toc182157072)

[1.2. Fitur Independen (x) 3](#_Toc182157073)

[1.3. Fitur Dependen (y) 4](#_Toc182157074)

[1.4. Ukuran Dataset 4](#_Toc182157075)

[1.5. Preview Dataset 4](#_Toc182157076)

[2. Analisis Sentimen Kesehatan Mental 4](#_Toc182157077)

[2.1. Sumber Dataset 4](#_Toc182157078)

[2.2. Fitur Independen (x) 4](#_Toc182157079)

[2.3. Fitur Dependen (y) 4](#_Toc182157080)

[2.4. Ukuran Dataset 4](#_Toc182157081)

[2.5. Preview Dataset 4](#_Toc182157082)

[Preprocessing & Train-Test Split 5](#_Toc182157083)

[1. Prediksi Stres 5](#_Toc182157084)

[1.1. Data Cleaning 5](#_Toc182157085)

[1.2. Feature & Label Encoding 6](#_Toc182157086)

[1.3. Feature Selection 6](#_Toc182157087)

[1.4. Train-Test Splitting 7](#_Toc182157088)

[2. Analisis Sentimen 7](#_Toc182157089)

[2.1. Data Cleaning 7](#_Toc182157090)

[2.2. Data Balancing 8](#_Toc182157091)

[2.3. Label Encoding 9](#_Toc182157092)

[2.4. Feature Selection and Lower Casing 9](#_Toc182157093)

[2.5. Train-Test Splitting 9](#_Toc182157094)

[2.6. Tokenization 10](#_Toc182157095)

[2.7. Pad Sequences 10](#_Toc182157096)

[2.8. Batch Data Loading 10](#_Toc182157097)

[Pemilihan Algoritma dan Evaluasi Model 11](#_Toc182157098)

[1. Prediksi Stres 11](#_Toc182157099)

[1.1. Algoritma Decision Tree 11](#_Toc182157100)

[1.2. Evaluasi Model Decision Tree 11](#_Toc182157101)

[1.3. Hyperparameter Tuning 12](#_Toc182157102)

[2. Analisis Sentimen 14](#_Toc182157103)

[2.1. Algoritma LSTM 14](#_Toc182157104)

[2.2. Struktur Model LSTM 14](#_Toc182157105)

[2.3. Evaluasi Model LSTM 15](#_Toc182157106)

[PENGEMBANGAN APLIKASI 17](#_Toc182157107)

[Requirements 17](#_Toc182157108)

[1. Versi Notebook 17](#_Toc182157109)

[2. Versi Aplikasi *Web* (*end-user*) 18](#_Toc182157110)

[3. Versi Aplikasi *Mobile* (*end-user*) 18](#_Toc182157111)

[User Interface 19](#_Toc182157112)

[1. Aplikasi *Web* 19](#_Toc182157113)

[2. Aplikasi *Mobile* 21](#_Toc182157114)

[PENUTUP 23](#_Toc182157115)

[Kesimpulan 23](#_Toc182157116)

[Rekomendasi 23](#_Toc182157117)

[Link Folder 23](#_Toc182157118)

[Link Notebook 23](#_Toc182157119)

[Link GitHub/ Source Code Lengkap 23](#_Toc182157120)

# 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*) |

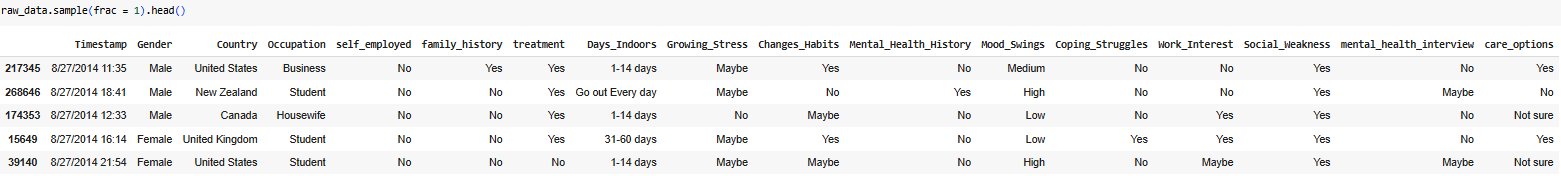
#### Fitur Dependen (y)

*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

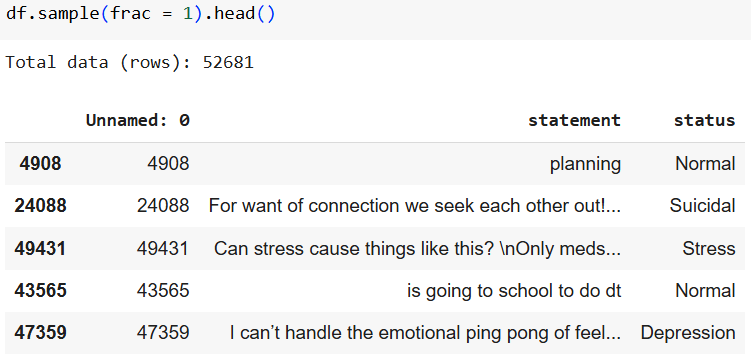
#### Fitur Dependen (y)

*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.

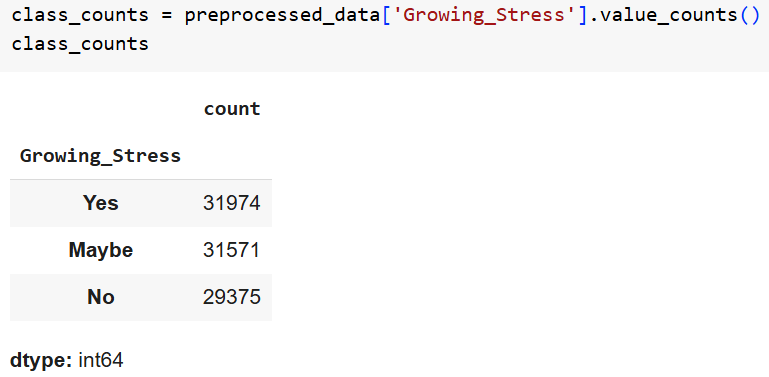
Bagian pertama yang dilakukan adalah membuang baris data yang berisi nilai kosong (*NaN*) karena data tersebut dianggap *invalid* dan akan mengganggu proses *build* model.

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.

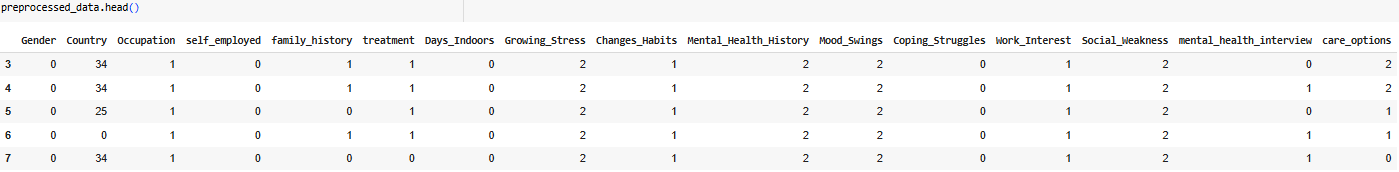
Untuk dataset “stress.csv”, jumlah data untuk tiap kelas sudah seimbang, dengan masing-masing kelas lebih kurang 30.000 data sehingga *data balancing* tidak perlu dilakukan lagi.



#### 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 mengurung diri 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.

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

#### 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, yang juga dapat membuat model *overfitting* dan menurunkan akurasi model.

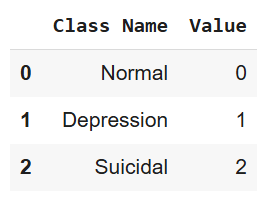
Hasil pengecekan menunjukkan bahwa jumlah data dari tiap kelas memiliki *range* yang terlalu jauh mulai dari 16.000-an data untuk kelas *Normal* hingga hanya 1.000-an data untuk kelas *Personality disorder* yang menunjukkan bahwa terdapat *imbalanced* data untuk kelas *Anxiety*, *Bipolar*, *Stres*, dan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

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

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*.

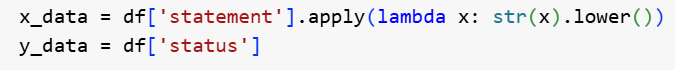


Proses *encoding* dilakukan secara manual bukan menggunakan *library* karena hasil *training* menunjukkan bahwa akurasi model dengan *encoding* manual lebih bagus.

#### Feature Selection and Lower Casing

Kolom yang akan dijadikan sebagai label/ kelas menggunakan kolom *Status*. Maka tersisa kolom *statement* dan *Unnamed:0* (merupakan kolom *index*), sehingga fitur independen (x) hanya akan menggunakan kolom *statement*.

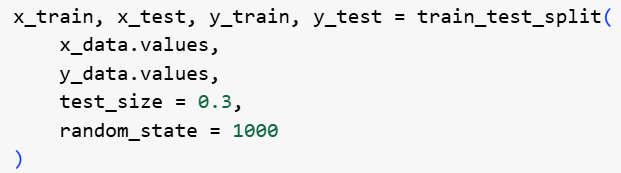
Semua data teks yang terdapat pada kolom *statement* juga perlu disamakan terlebih dahulu huruf besar kecilnya.



#### Train-Test Splitting

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

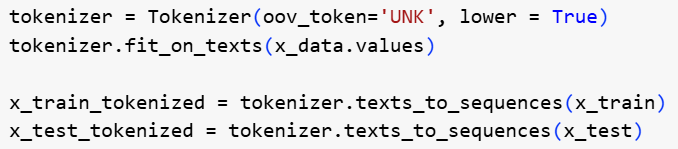
1. Mempertahankan distribusi data *training* dan *testing*
2. Mencegah kebocoran data dimana data *training* dan *testing* akan ditokenisasi secara terpisah



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 rangkaian angka agar dapat diproses oleh model.



Setiap kata yang ditokenisasi dapat diibaratkan sebagai kamus *vocabulary*. Sebagai contoh, kalimat “I feel 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. Panjang maksimal dapat ditentukan dari data sekuensial hasil *tokenizer* yang memiliki jumlah token paling banyak atau ukuran paling panjang.

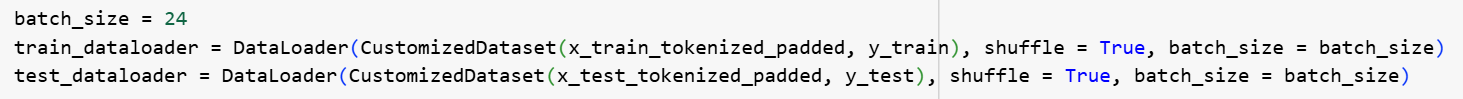


Sebagai contoh, [2, 15, 1, 37, 8] dan [3, 29, 15] akan dimodifikasi menjadi [2, 15, 1, 37, 8] dan [3, 29, 15, 0, 0] jika panjang sekuensial maksimal yang ditentukan adalah 5.

#### Batch Data Loading

*Batch data loading* dilakukan agar model dapat melakukan *training* pada beberapa data sekaligus dalam 1 (satu) *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

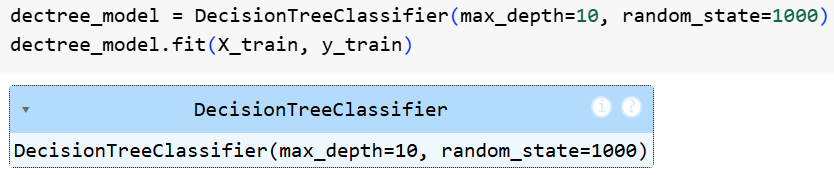
#### Algoritma Decision Tree

Percobaan dilakukan menggunakan beberapa algoritma, antara lain Random Forest, KNN, dan Decision Tree. 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:

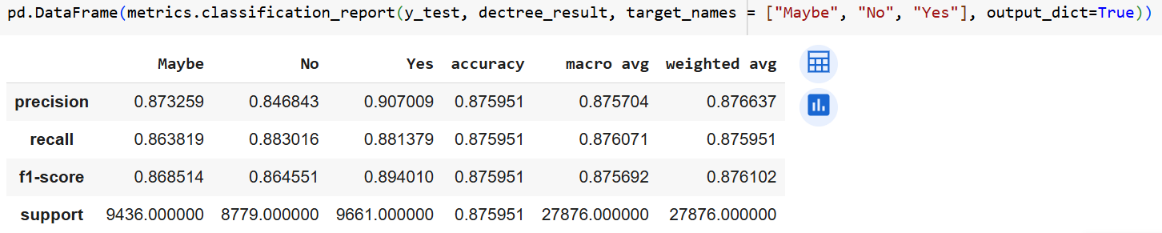
1. Mudah dipahami, divisualisasi, dan diinterpretasikan karena berbasis “*If-Then-Else*”
2. Proses *build* dan *training* model lebih cepat karena lebih simpel dan tidak sekompleks model Random Forest
3. Cocok untuk data yang memiliki korelasi antar variabel yang non-linear

#### Evaluasi Model Decision Tree

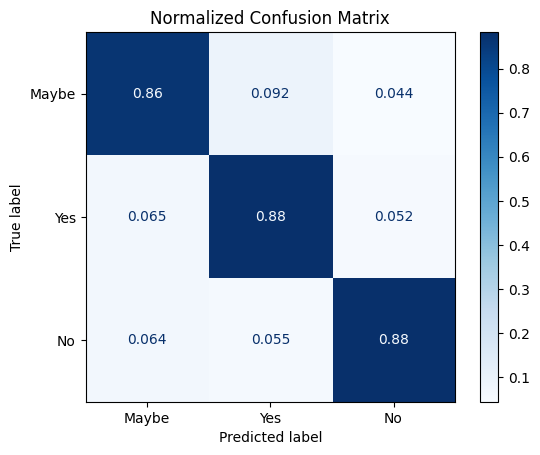
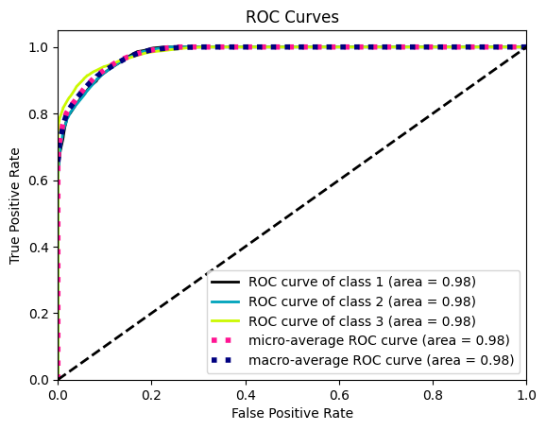
*Hyperparameter* dari model Decision Tree tidak sebanyak dan sekompleks model Random Forest. Percobaan pertama untuk penentuan kedalaman maksimal (*max depth*) dari pohon diambil dengan pertimbangan jumlah kolom/ fitur yang digunakan agar model tidak mengalami *overfitting* akibat model yang terlalu dalam dan kompleks. *Random state* 1.000 digunakan agar model memberikan hasil yang sama setiap kali *code* dijalankan ulang.



Hasil training dan testing model menunjukkan akurasi 87,57% dengan nilai *precision*, *recall*, dan F1 yang bisa dibilang cukup merata untuk masing-masing kelas data.

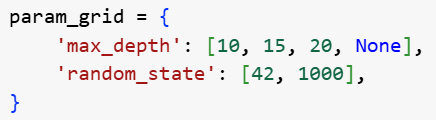


Dilihat dari grafik *Confusion* *Matrix* dibawah, jumlah kesalahan hasil prediksi dari data aktual juga cukup kecil, walau nilai akurasi masih belum mencapai target yakni 90%. Nilai ROC juga sudah bagus yang ditunjukkan oleh garis grafik yang melengkung ke bagian kiri atas dengan nilai 0.98.

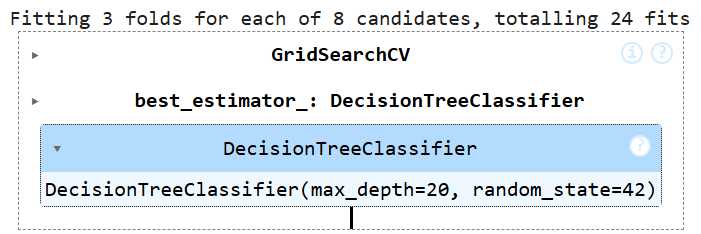
#### Hyperparameter Tuning

*Hyperparameter* *Tuning* dilakukan untuk mencari kombinasi *hyperparameter* terbaik untuk model agar dapat memberikan prediksi dengan akurasi terbaik.

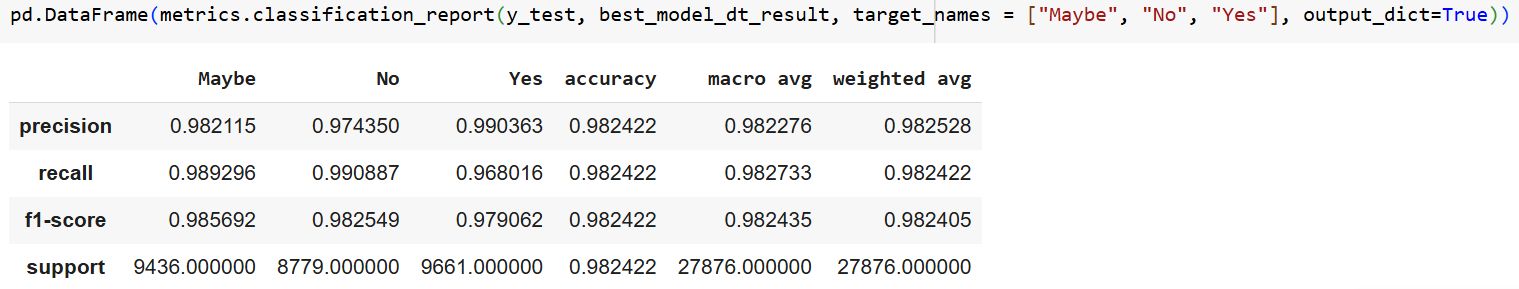


Salah satu *tools* scikit-learn yang dapat digunakan untuk *Hyperparameter Tuning* model secara otomatis adalah GridSearchCV. Terdapat 3 pilihan *max\_depth* (kedalaman pohon), 2 pilihan *random\_state*, dan 3-*fold cross validation* sehingga terdapat total 6x2x3 = 24 kombinasi model yang dibangun dan metrik pemilihan model terbaik akan ditentukan berdasarkan akurasi model tertinggi.

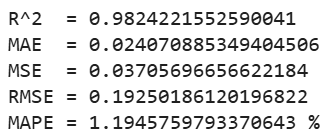
Hasilnya, *hyperparameter* terbaik yang dipilih adalah *max\_depth* (kedalaman maksimal) pohon sebesar 20 level dan nilai *random\_state* di angka 42.



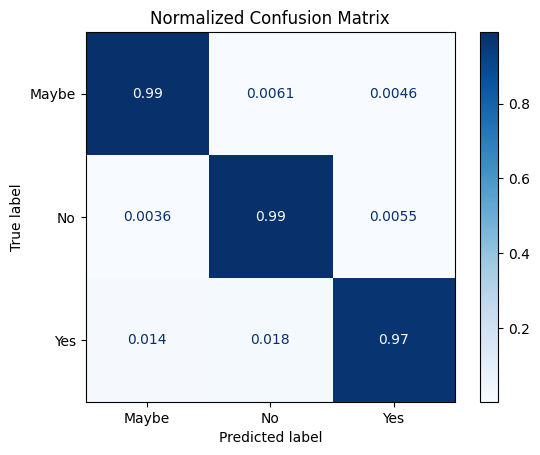
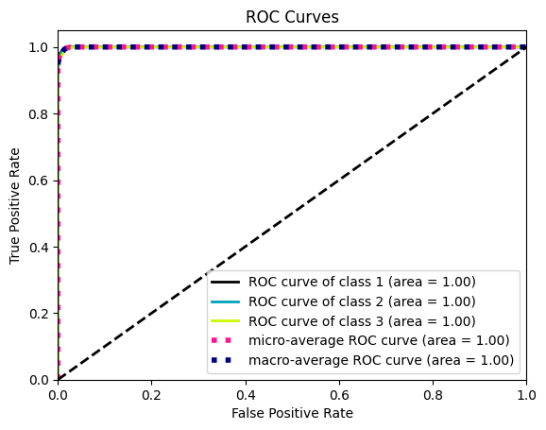
Akurasi model meningkat dari 87,59% menjadi 98,24% yang menunjukkan bahwa kombinasi hyperparameter hasil tuning sudah sesuai dan akurasi model juga sudah mencapai target (>90%).



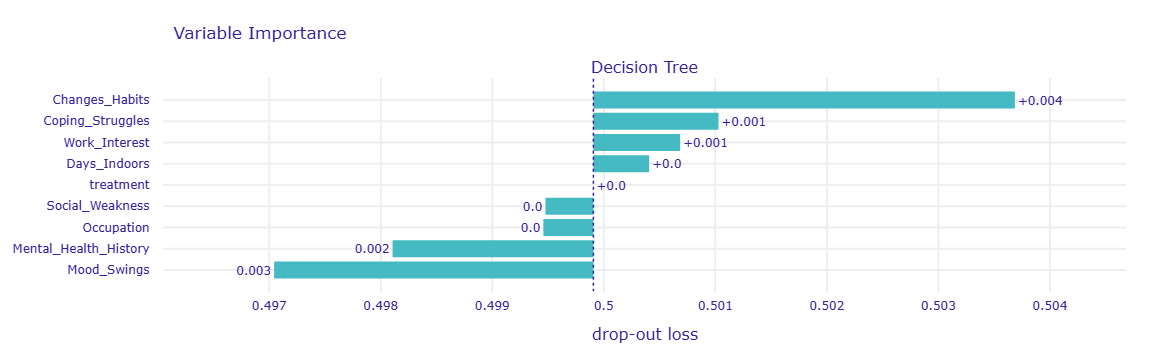
Selain akurasi, presisi, dan *recall*, perhitungan *error* juga dilakukan pada model hasil *hyperparameter* *tuning* dengan beberapa metrik. Nilai *R-square* model berada di angka 0.98 atau 98,24% yang menunjukkan bahwa model dapat menjelaskan variasi pada variabel target dengan sangat baik dan fitur-fitur yang dipilih sebagai fitur independen (x) sudah sesuai. Untuk metrik lainnya, perhitungan MAPE juga menunjukkan *error* yang kecil di angka 1,2%.



Grafik *Confusion* *Matrix* dibawah menunjukkan bahwa model sudah dapat memprediksi dengan sangat baik untuk masing-masing kelas dan tidak condong ke salah satu kelas karena data yang diinput sudah *balanced*. Nilai ROC juga naik menjadi 1.00 yang menunjukkan model sudah dapat mengklasifikasikan setiap kelas data dengan baik dan tepat.

Beberapa variabel yang berperan penting dalam prediksi menurut model yang sudah dibangun antara lain: *Changes\_Habits* (perubahan kebiasaan akhir-akhir ini), *Coping\_Struggles* (ada tidaknya keinginan untuk menghadapi dan mengatasi masalah mental yang dihadapi), *Work\_Interest* (seberapa terikat dengan pekerjaan), dan *Days\_Indoors* (sudah berapa lama berada atau “mengurung diri” dirumah saja), yang jika dipikirkan secara logika cukup masuk akal karena fitur tersebut dapat untuk menjawab apakah seseorang mengalami stres atau tidak.

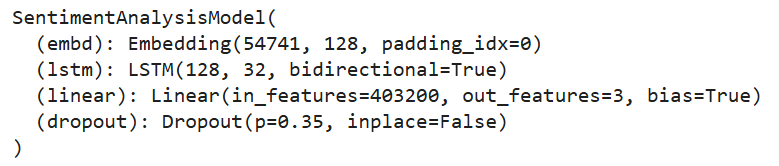


### Analisis Sentimen

#### 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 menangkap 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.

#### Struktur Model LSTM



Model LSTM yang dibangun terdiri dari 4 lapisan, yaitu:

1. Embedding

*Layer* ini menerima input berupa token, dengan ukuran *vocab* 54.741 data token unik yang akan direpresentasikan dalam vektor 128 dimensi. Penambahan *padding* bernilai 0 untuk menyeragamkan ukuran sekuensial.

1. LSTM

*Layer* inti dari model yang akan menerima data sekuensial berdimensi 128, memiliki 32 unit *neuron cell* pada setiap *hidden* *layer* LSTM, dan diproses secara *forward* dan *backward* (*bidirectional*).

1. Linear

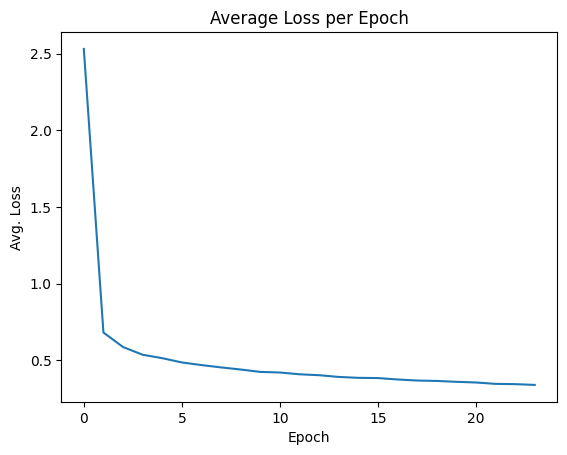
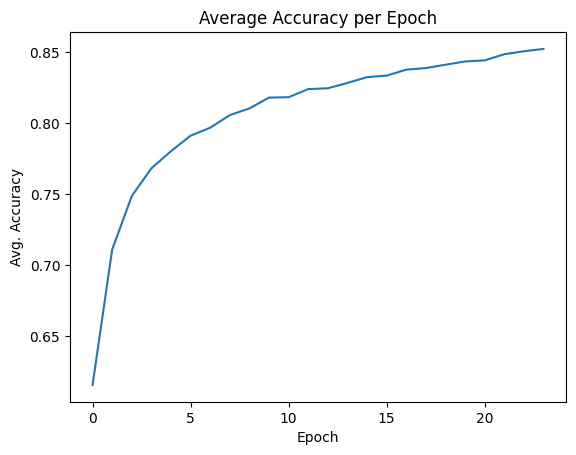
Disebut juga *Fully* *Connected* layer. *Layer* ini menerima input berukuran 403.200 (2x32x6300) yang merupakan hasil *Flattened* dari *layer* LSTM sebelumnya. Hasil output dari layer ini akan berada diantara 3 kelas, yaitu *Yes*/ *No*/ *Maybe*.

1. Dropout

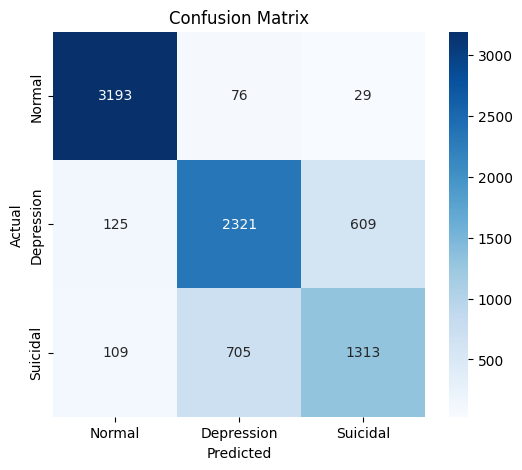
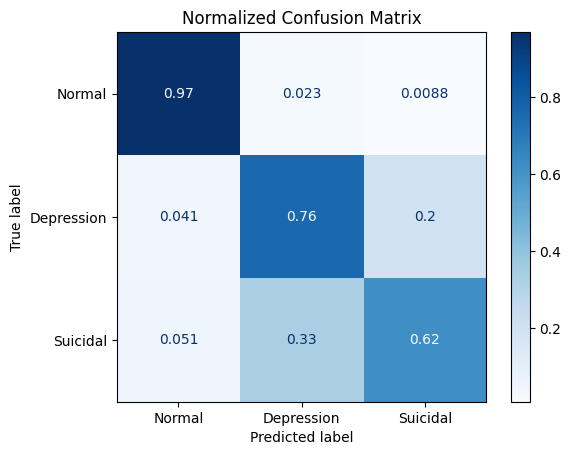
Untuk mencegah terjadinya *overfitting* selama proses *training* model, penambahan *layer* Dropout dilakukan dengan pembuangan 35% unit *neuron*.

#### Evaluasi Model LSTM

Setelah di-*looping* sebanyak 24 *epochs* untuk proses *training*, nilai *loss* model menurun drastis pada *epoch* ke-3 dan nilai akurasi model meningkat hingga 85,23% di akhir *epoch*. Bisa dilihat bahwa model belajar dengan cukup baik selama proses *training*.

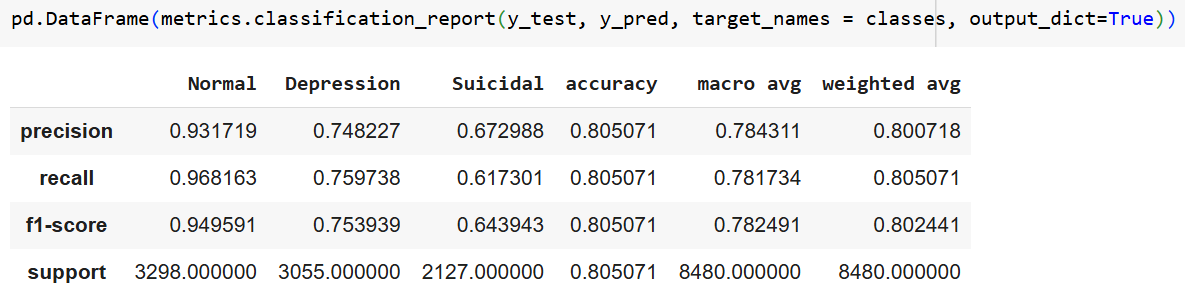
 

Grafik *Confusion Matrix* menunjukkan lemahnya model dalam memprediksi label *Depression* dan *Suicidal* dengan tepat walaupun *imbalanced* data sudah ditangani di tahapan *preprocessing*, dimana masih terdapat *range* sekitar 6.000 data antara data berlabel *Normal* dengan *Suicidal*.

Hal ini juga tentu berdampak pada menurunnya akurasi model karena model cenderung memprediksi label *Depression* dan *Suicidal* secara terbalik satu sama lain. Sedangkan untuk label *Normal*, nilai *precision*, *recall*, dan F1 menunjukkan angka sekitar 95% yang berarti model mampu memprediksi dengan sangat baik.

Nilai akurasi model setelah di-*testing* adalah 80,5% yang masih belum mencapai target akurasi di angka 90%. Nilai akurasi model yang turun dari 85% pada proses training menjadi 80,5% pada proses testing menunjukkan bahwa model tidak terlalu *overfitting*, namun kinerja model masih kurang baik.



# 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 data stres](https://www.kaggle.com/datasets/bhavikjikadara/mental-health-dataset) ataupun menggunakan *file* “stress.csv” untuk prediksi stres dan juga melalui [link kaggle data sentimen](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 masuk ke folder mobile

cd mobile

kemudian 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 *mode* *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).

Jika menggunakan perangkat Android, maka dapat langsung menjalankan perintah

flutter install

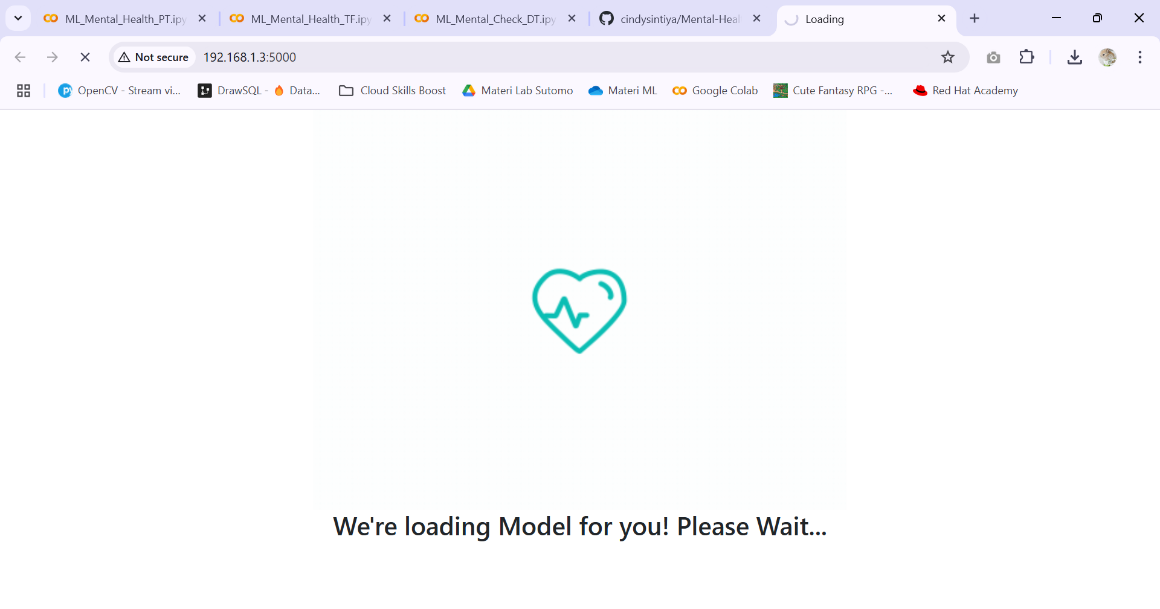
untuk menginstal aplikasi ke perangkat Android Anda (disarankan).

1. Masukkan *IP* *Address*/ *IP* *Host* dari aplikasi *web* yang sudah berjalan (alamat *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

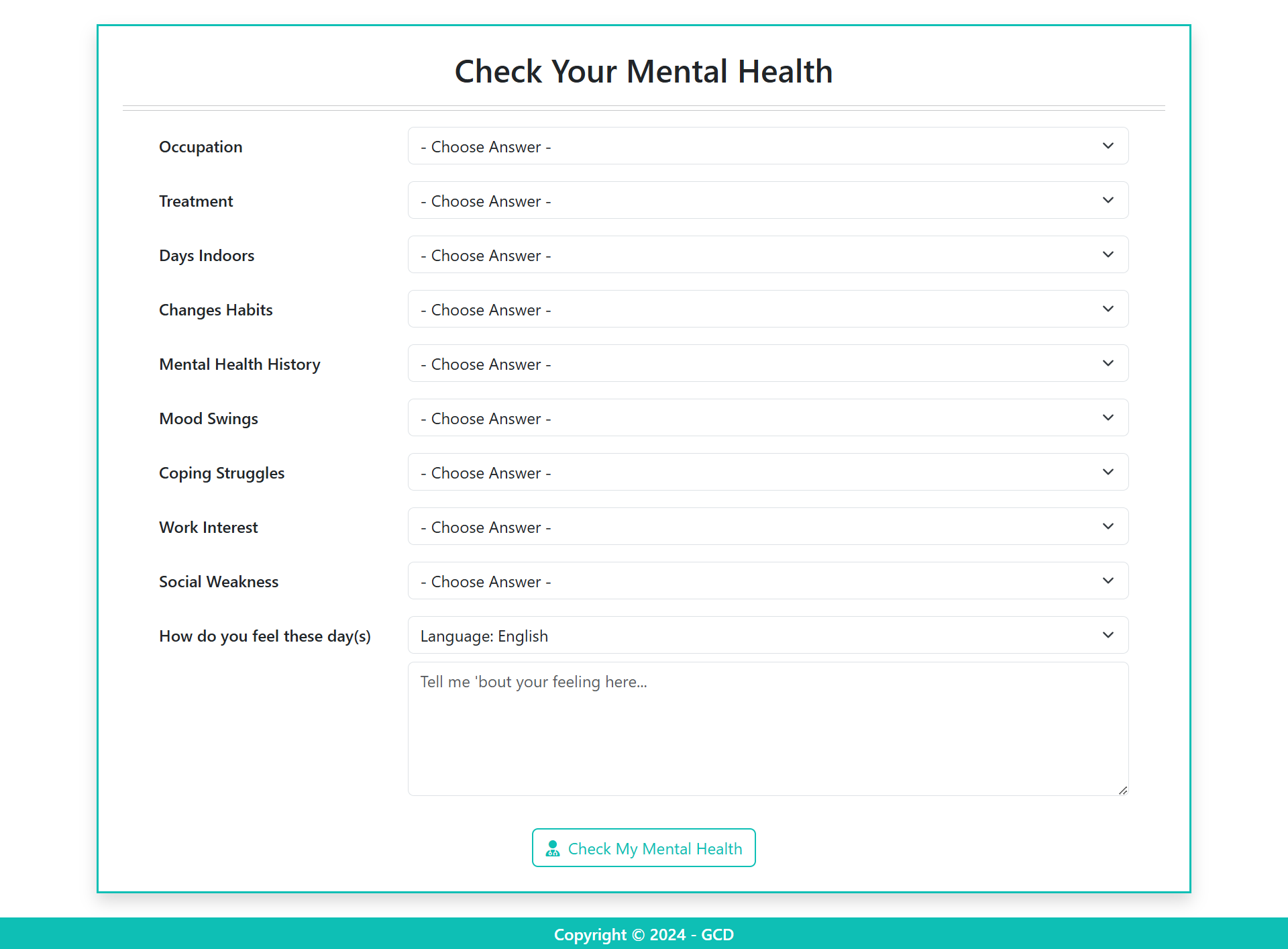
### Aplikasi *Web*

1. Saat pertama kali *website* dibuka, akan ada proses *loading* model *pre-trained*. Hal ini dilakukan agar pada saat proses prediksi, pengguna tidak perlu menunggu model *loading* terus menerus setiap kali *submit* formulir dan bisa langsung mendapatkan hasil prediksi dengan cepat.

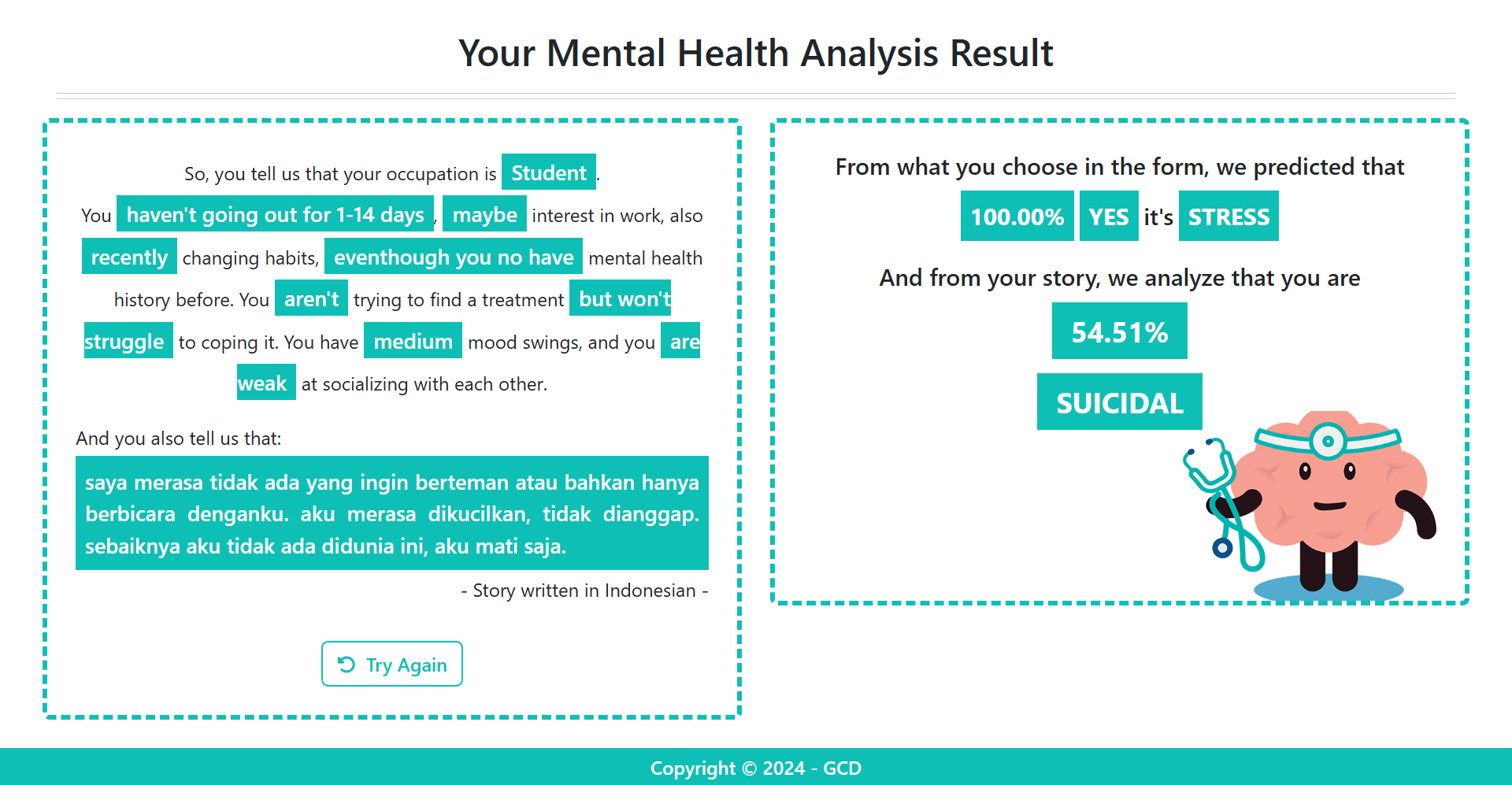


1. Setelah model selesai di-*load*, pengguna akan langsung melihat halaman formulir yang setiap *field*-nya wajib diisi.

Bagian formulir yang berupa pilihan *dropdown* akan digunakan sebagai data input model prediksi stres (Decision Tree). Pada bagian “How do you feel these day(s)?”, pengguna dapat menceritakan (curhat) terkait perasaan yang akan digunakan sebagai data input model analisis sentimen (LSTM).



1. Hasil prediksi dapat langsung dilihat setelah pengguna menekan tombol *submit* atau “Check My Mental Health”. Bagian ini terbagi atas 2 prediksi, yaitu stres dan analisis sentimen.



1. Untuk melakukan pengisian formulir lagi, pengguna dapat menekan tombol “*Try Again*” yang terdapat dibagian bawah rangkuman formulir.

### Aplikasi *Mobile*

1. Saat pertama kali aplikasi *mobile* dibuka, pengguna perlu memasukkan IP Address dari API yang diperoleh dari aplikasi *web* yang sudah dijalankan sebelumnya.

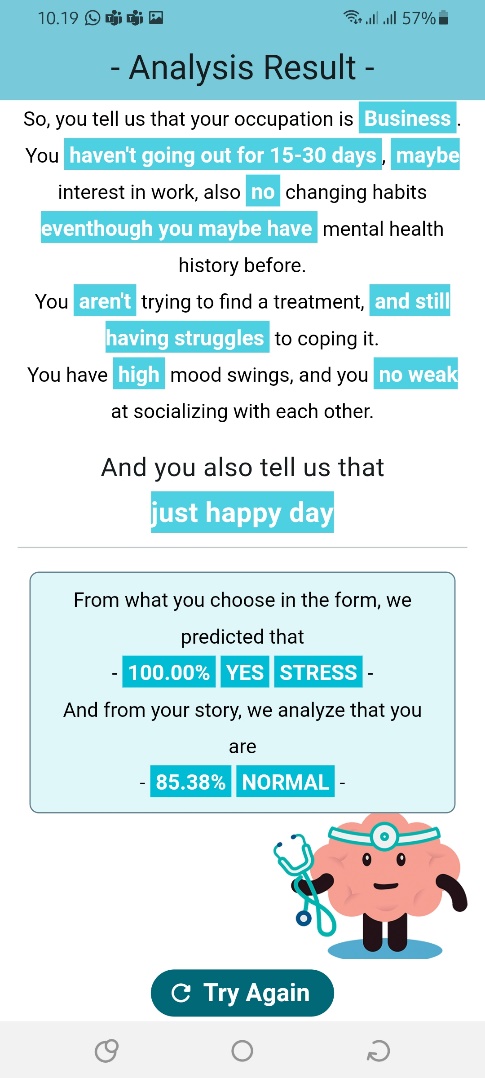
|  |  |  |
| --- | --- | --- |
|  |  |  |

Jika IP yang dimasukkan ternyata salah atau server/ aplikasi *web* belum berjalan, pengguna akan mendapatkan *error* *Timeout* setelah 45 detik atau dapat langsung mengganti IP yang valid pada menu *Setting*.

1. Setelah itu, akan ada proses *load* model jika sebelumnya model belum di-*load*. Pengguna kemudian diarahkan ke halaman formulir.

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

1. Sama seperti aplikasi *web*, hasil prediksi dapat langsung dilihat setelah pengguna menekan tombol “*Submit*”. Bagian ini terbagi atas 2 prediksi, yaitu stres dan analisis sentimen.



1. Untuk melakukan pengisian formulir lagi, pengguna dapat menekan tombol “*Try Again*” yang terdapat dibagian bawah hasil prediksi.

# PENUTUP

## Kesimpulan

Dari 3 model yang dicoba untuk prediksi stres, yakni Random Forest, KNN, dan Decision Tree, akurasi prediksi tertinggi ada pada model Random Forest dan Decision Tree dengan akurasi yang sama di angka 98,24%. Karena algoritma Decision Tree lebih sederhana, mudah dipahami dan diinterpretasikan, serta sesuai untuk dataset dengan kompleksitas rendah yang memiliki 9 fitur independen, model ini dipilih untuk melakukan prediksi stres karena sudah mampu memberikan hasil yang akurat dengan tingkat akurasi 98,24%.

Sementara itu, model LSTM yang digunakan untuk analisis sentimen masih perlu dilakukan perbaikan dan peningkatan akurasi yang hanya mentok di angka 80%.

## Rekomendasi

Untuk analisis sentimen kesehatan mental, sebaiknya menambahkan data aktual ataupun mencari dataset dari sumber lain agar dapat menyeimbangkan data *imbalanced* untuk meningkatkan akurasi model. Eksplorasi teknik pemrosesan teks yang lebih lanjut seperti penggunaan *embeddings* yang lebih spesifik juga dapat dicoba untuk memperbaiki kualitas model, dan juga mencoba algoritma NLP yang lain seperti BERT mungkin bisa meningkatkan akurasi model dalam mengklasfikasikan sentimen.

## Link Folder

[Akses folder lengkap (video & dokumen) GCD - Mental Health Analysis](https://mikroskilacid-my.sharepoint.com/:f:/g/personal/211111930_students_mikroskil_ac_id/EuqScwkWlntMimutfgRcGskBZEhPjG2JS_KBkN3Jgp4gNQ?e=n1MXCR)

## Link Notebook

[Akses notebook GCD - Stress Prediction Decision Tree](https://colab.research.google.com/drive/17E7s-cgwUiIgSd-8syjLd_YS-Z9U2XZ1?usp=sharing)

[Akses notebook GCD - Sentiment Analysis LSTM - Pytorch](https://colab.research.google.com/drive/1oMMMTWQgHKxrm781LjDc1oIWVhhN4rKj?usp=sharing)

[Akses notebook GCD - Sentiment Analysis LSTM - Tensorflow](https://colab.research.google.com/drive/15fbIuiTUOoc0PVInvBOSJABQeo3aFH5t?usp=sharing)

## Link GitHub/ Source Code Lengkap

[Akses GitHub GCD - Mental Health Analysis disini](https://github.com/cindysintiya/Mental-Health_Sentiment-Analysis)