**MAKALAH SAINS DATA**

# Model Penentuan Tingkat Stress Berdasarkan Pekerjaan, Awareness Kesehatan Mental dan Catatan Kesehatan.

*Untuk Memenuhi Tugas Akhir Kelompok Mata*
*Kuliah Sains Data*

**SAINS DATA**

Disusun Oleh:


| | | |
|---|---|---|
| Cecilia Susanto | 2206052881 | 2022 |
| Fattan Raditya Anggoro | 2206812533 | 2022 |
| Fakhri Rayhan Alaudin | 2206048814 | 2022 |
| Achmad Miftakhul Rachman Ardiva | 2106705820 | 2021 |
| Cherien Stevie Agustiara Suldi | 2206048524 | 2022 |


**UNIVERSITAS INDONESIA**

**DEPOK**

**2024**

# ABSTRAK

*Neural network* adalah suatu metode *Artificial Intelligence* yang dapat mengidentifikasi suatu pola dan menyelesaikan suatu masalah. Pada kesempatan kali ini, akan ditunjukkan pengaplikasian *Neural Network* untuk menyelesaikan masalah kesehatan mental dan prediksi variabel target *growing stress* kepada sampel random.

Tujuan adanya penelitian ini adalah untuk memprediksi apakah seseorang mempunyai masalah kesehatan mental berdasarkan fitur-fitur lain. Metode yang digunakan dalam penelitian ini adalah Artificial Neural Network. Sedangkan data yang digunakan berasal dari Kaggle.com yang berisi informasi data diri orang-orang beserta riwayat mereka terhadap fitur yang berkaitan dengan kesehatan mental dan berukuran 292364 sampel (baris) dan 16 fitur beserta target fiturnya.

Hasil penelitian kami membuahkan hasil berupa model dengan nilai akurasi sebesar 58,79% yang sudah jauh lebih baik dari iterasi sebelum diimprovisasi, namun tentunya dibutuhkan pengembangan model lebih lanjut untuk dapat digunakan di situasi dunia nyata.

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Stres adalah salah satu masalah kesehatan mental yang signifikan di era modern ini. Dengan tingkat kehidupan yang semakin kompleks dan tuntutan yang meningkat di tempat kerja maupun kehidupan pribadi, masyarakat sering kali mengalami tekanan yang dapat memengaruhi kesejahteraan fisik dan emosional mereka. Dalam konteks ini, pemahaman tentang faktor-faktor yang mempengaruhi tingkat stres menjadi penting untuk membantu mengembangkan strategi intervensi yang efektif.

Faktor-faktor yang menjadi fokus analisis meliputi riwayat keluarga, jenis perawatan yang diterima, jumlah waktu yang dihabiskan di dalam ruangan, tingkat stres yang dialami seiring waktu, perubahan kebiasaan, riwayat kesehatan mental, fluktuasi mood, kesulitan dalam mengatasi stres, minat dalam pekerjaan, kelemahan sosial, wawancara kesehatan mental, dan pilihan perawatan yang tersedia.

Melalui pendekatan klasifikasi, kita dapat mengeksplorasi hubungan antara faktor-faktor ini dan tingkat stres yang dialami oleh individu. Analisis yang mendalam terhadap data yang terkait dengan faktor-faktor ini diharapkan dapat memberikan wawasan yang berharga dalam memahami dinamika stres dan memungkinkan pengembangan strategi intervensi yang lebih efektif.

Dengan demikian, laporan ini tidak hanya memberikan pemahaman yang lebih baik tentang faktor-faktor yang berkontribusi terhadap stres, tetapi juga memiliki potensi untuk memberikan kontribusi dalam pengembangan pendekatan yang lebih holistik dalam penanganan stres dalam konteks kesehatan mental.

## 1.2 Rumusan Masalah

- Seberapa akurat model penentuan tingkat stres dengan *input* data pekerjaan, *awareness* kesehatan mental dan catatan kesehatan yang dibuat dengan fungsi aktivasi ReLU dan softmax?

## 1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui tingkat stress pada setiap individu bergantung kepada karakteristik pribadi yang berhubungan dengan pekerjaan, *awareness* kesehatan mental dan catatan kesehatan. Penelitian ini juga bertujuan sebagai *Project* Ujian Akhir Semester mata kuliah Sains Data Departemen Matematika FMIPA UI.

# BAB II
# DATA DAN METODE

## 2.1 Data

Penulis memperoleh sebuah dataset publik yang diperoleh dari Kaggle.com. Dataset terkait berisi tentang informasi mengenai data diri orang-orang beserta riwayat mereka terhadap fitur yang berkaitan dengan kesehatan mental. Dataset ini berukuran 292364 sampel (baris) dan 16 fitur beserta target fiturnya.

Lebih lanjut, berikut penjelasan terperinci fitur-fitur yang ada:

| | |
|---|---|
| Gender | Jenis kelamin |
| Family History | Riwayat penyakit kesehatan mental di keluarga |
| Change Habits | Apakah kebiasaan orang terkait berubah ketika bermasalah secara mental? |
| Work Interest | Ketertarikan terhadap pekerjaan |
| Country | Kewarganegaraan |
| Treatment | Apakah pernah menjalani perawatan kesehatan mental? |
| Mental Health History | Riwayat penyakit kesehatan mental pribadi |
| Social Weakness | Apa punya kesulitan bersosialisasi? |
| Occupation | Pekerjaan |
| Days Indoors | Waktu berada di rumah dalam hari per bulan |
| Mood Swings | Frekuensi *mood swing* |
| Mental Health Interview | Apakah pernah menjalani konseling? |
| Self Employed | Apakah bekerja sendiri? |
| Growing Stress | Variabel target |
| Coping Struggles | Apakah mempunyai kesulitan untuk menghadapi masalah kesehatan mental? |
| Care Options | Mempunyai subskripsi *‘care options’* |

## 2.2 Metode

### 2.2.1 Pre-Processing

Tahap ini adalah proses mempersiapkan data agar bisa diolah pada proses pemrosesan dan pembuatan model.

### 2.2.2 Observasi Data

Observasi data dilakukan untuk mengetahui isi data. Diketahui dataset yang kami gunakan memiliki 292364 baris dan 16 fitur dengan kesemuanya adalah data kategorik.

```
[93] df = pd.read_csv("/content/Mental Health Dataset.csv")
     df = df.drop(['Timestamp'], axis = 1) #Timestamp di deskripsi dataset nya tuh "Time the survey was submitted", jadi fitur ini tidak berpengaruh
     df
```

| | Gender | Country | Occupation | self_employed | family_history | treatment | Days_Indoors | Growing_Stress | Changes_Habits | Mental_Health_History | Moo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Female | United States | Corporate | NaN | No | Yes | 1-14 days | Yes | No | Yes | |
| 1 | Female | United States | Corporate | NaN | Yes | Yes | 1-14 days | Yes | No | Yes | |
| 2 | Female | United States | Corporate | NaN | Yes | Yes | 1-14 days | Yes | No | Yes | |
| 3 | Female | United States | Corporate | No | Yes | Yes | 1-14 days | Yes | No | Yes | |
| 4 | Female | United States | Corporate | No | Yes | Yes | 1-14 days | Yes | No | Yes | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 292359 | Male | United States | Business | Yes | Yes | Yes | 15-30 days | No | Maybe | No | |
| 292360 | Male | South Africa | Business | No | Yes | Yes | 15-30 days | No | Maybe | No | |

```
[94] display(df.info())
     display(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292364 entries, 0 to 292363
Data columns (total 16 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Gender                  292364 non-null  object
 1   Country                 292364 non-null  object
 2   Occupation              292364 non-null  object
 3   self_employed           287162 non-null  object
 4   family_history          292364 non-null  object
 5   treatment               292364 non-null  object
 6   Days_Indoors            292364 non-null  object
 7   Growing_Stress          292364 non-null  object
 8   Changes_Habits          292364 non-null  object
 9   Mental_Health_History   292364 non-null  object
 10  Mood_Swings             292364 non-null  object
 11  Coping_Struggles        292364 non-null  object
 12  Work_Interest           292364 non-null  object
 13  Social_Weakness         292364 non-null  object
 14  mental_health_interview 292364 non-null  object
 15  care_options            292364 non-null  object
dtypes: object(16)
```

## 2.2.3 Menangani Data Hilang

```
#Identifikasi Missing Values
df.isnull().sum()
```

```
Gender                     0
Country                    0
Occupation                 0
self_employed           5202
family_history             0
treatment                  0
Days_Indoors               0
Growing_Stress             0
Changes_Habits             0
Mental_Health_History      0
Mood_Swings                0
Coping_Struggles           0
Work_Interest              0
Social_Weakness            0
mental_health_interview    0
care_options               0
dtype: int64
```

```
#Identifikasi Missing Values
df.isnull().mean()
```

```
Gender                     0.000000
Country                    0.000000
Occupation                 0.000000
self_employed              0.017793
family_history             0.000000
treatment                  0.000000
Days_Indoors               0.000000
Growing_Stress             0.000000
Changes_Habits             0.000000
Mental_Health_History      0.000000
Mood_Swings                0.000000
Coping_Struggles           0.000000
Work_Interest              0.000000
Social_Weakness            0.000000
mental_health_interview    0.000000
care_options               0.000000
```

```
#Menghapus Baris dengan Missing Values (data yang hilang < 0.1)
df2 = df.copy()
df2.dropna(axis=0, inplace = True)
df2.isnull().sum()
```

```
Gender                    0
Country                   0
Occupation                0
self_employed             0
family_history            0
treatment                 0
Days_Indoors              0
Growing_Stress            0
Changes_Habits            0
Mental_Health_History     0
Mood_Swings               0
Coping_Struggles          0
Work_Interest             0
Social_Weakness           0
mental_health_interview   0
care_options              0
```

```
#Imputasi Missing Values menggunakan Mode (data yang hilang kategorikal
df3 = df.copy()
df3 = df3.fillna(df3.mode().iloc[0])
df3.isnull().sum()
```

```
Gender                    0
Country                   0
Occupation                0
self_employed             0
family_history            0
treatment                 0
Days_Indoors              0
Growing_Stress            0
Changes_Habits            0
Mental_Health_History     0
Mood_Swings               0
Coping_Struggles          0
Work_Interest             0
Social_Weakness           0
mental_health_interview   0
care_options              0
```

Akan dilakukan dua cara untuk menangani data outlier, yaitu dengan imputasi dan dengan menghapus baris dengan *missing values*. Kita dapat menghapus baris dengan *missing values* karena jumlah data yang hilang sangat kecil.

### 2.2.4 Label Encoding

Tahap ini akan dilakukan encoding atau pengubahan data-data pada fitur yang bernilai kategorik menjadi data numerik. Proses ini dilakukan karena pada tahap pemrosesan data untuk pembuatan model, data haruslah dalam bentuk numerik di dalam *array*.

```
#Label encoder untuk df3
le2 = LabelEncoder()

# Apply LabelEncoder to each column
encoded_df3 = df3.apply(le2.fit_transform)

encoded_df3.head(10)
```

| | Gender | Country | Occupation | self_employed | family_history | treatment | Days_Indoors | Growing_Stress | Changes_Habits | Mental_Health_History | Mood_Swi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 34 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 2 | |
| 1 | 0 | 34 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | |
| 2 | 0 | 34 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | |
| 3 | 0 | 34 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | |
| 4 | 0 | 34 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | |
| 5 | 0 | 25 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 2 | |
| 6 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 2 | |
| 7 | 0 | 34 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 2 | |
| 8 | 0 | 34 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 2 | |

### 2.2.5 Artificial Neural Network (ANN)

Artificial Neural Network adalah jaringan dari sekelompok unit pemroses kecil (neural/neuron) yang dimodelkan dan ditiru berdasarkan seperti jaringan saraf manusia, yaitu neuron. Artificial Neural Network merupakan suatu sistem yang dapat beradaptasi dengan mengubah struktur-strukturnya dan akan menyelesaikan suatu masalah berdasarkan informasi-informasi yang mengalir baik itu eksternal maupun internal.

```
[ ] Model definition

    import tensorflow as tf
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Dense(8, input_shape = (X_train.shape[1], ), activation = "relu"))
    model.add(tf.keras.layers.Dense(y_train.shape[1], activation = "softmax"))
    model.summary()
```

Baris ini mengimpor library TensorFlow dan menginisialisasi alias `tf` untuk memudahkan referensi, membuat sebuah instance dari `Sequential`, yang merupakan sebuah model linear dari lapisan-lapisan neural network. Model ini akan memiliki susunan lapisan-lapisan yang berurutan.

- `model.add(tf.keras.layers.Dense(8, ...))` menambahkan lapisan dense (fully connected layer) dengan 8 neuron.
- `input_shape = (X_train.shape[1], )` menentukan bentuk input data. `X_train.shape[1]` merujuk pada jumlah fitur dalam data training. Ini digunakan hanya pada lapisan pertama untuk menetapkan ukuran input.
- `activation = "relu"` menetapkan fungsi aktivasi ReLU (Rectified Linear Unit) untuk lapisan ini.
- `model.add(tf.keras.layers.Dense(y_train.shape[1], ...))` menambahkan lapisan dense dengan jumlah neuron sama dengan jumlah kelas dalam data target `y_train`. `y_train.shape[1]` merujuk pada jumlah kelas yang ada.
- `activation = "softmax"` menetapkan fungsi aktivasi Softmax untuk lapisan ini, yang umum digunakan untuk klasifikasi multi-kelas.

```
[ ] X.shape
```

Baris ini menampilkan bentuk dari data input `X`. Informasi ini berguna untuk memahami dimensi data yang akan dimasukkan ke dalam model.

```
[ ] tf.keras.utils.plot_model(
        model,
        show_shapes = True,
        show_layer_activations = True,
        to_file = "keras_sequential_model2.png"
    )
```

- `tf.keras.utils.plot_model` digunakan untuk menghasilkan diagram visual dari arsitektur model.
- `model` adalah model yang didefinisikan sebelumnya.
- `show_shapes = True` menampilkan bentuk setiap lapisan dalam diagram.
- `show_layer_activations = True` menampilkan fungsi aktivasi yang digunakan pada setiap lapisan dalam diagram.
- `to_file = "keras_sequential_model2.png"` menyimpan diagram dalam format gambar PNG dengan nama file yang ditentukan.

# BAB III
# IMPLEMENTASI PROGRAM DAN PEMBAHASAN

## 3.1 Memisahkan Fitur dan Target

```
[ ]  #Splitting Features and Target
     target_variable = 'Growing_Stress'
     X = encoded_df3.drop([target_variable], axis = 1)
     y = encoded_df3[target_variable]
     y
```

Dipilih ‘Growing Stress’ sebagai kolom yang ingin diprediksi oleh model dengan X adalah DataFrame yang berisi semua fitur kecuali kolom target. Dengan ‘encoded_df3’ adalah DataFrame yang sudah di encode sebelumnya yang berarti sudah diubah menjadi format numerik. Dilakukan juga penghapusan kolom target agar X hanya berisi fitur. Y adalah nilai nilai dari kolom target dan memastikan bahwa Y berisi tabel yang sesuai dengan baris fitur di X.

## 3.2 Melakukan Pemisahan data dan Standarisasi

```
[ ]  from sklearn.model_selection import train_test_split

     SEED = 42 #Menetapkan seed untuk mengatur kerandoman splitting data
     ukuran_test = 0.25 #Persentase data yang digunakan sebagai test set

     X_train, X_test, y_train, y_test = train_test_split(X, y_categorical, test_size = ukuran_test, random_state = SEED)
     X_test #Standarisasi
```

Diambil nilai acak dalam pemisahan data yaitu 42 yang artinya jika kode ini dijalankan dengan acak, hasil pemisahan datanya akan sama.Agar metode ANN dapat memberikan hasil yang baik, diperlukan standarisasi terlebih dahulu. Standarisasi membantu dalam konvergensi yang stabil selama model neural network dijalankan. Ditentukan juga untuk proporsi data, yaitu 25% digunakan sebagai data *test*, dan sisanya (75%) digunakan sebagai data *train*.

## 3.3 Menentukan Parameter

```
Model definition #modelnya disusun

import tensorflow as tf
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(8, input_shape = (X_train.shape[1], ), activation = "relu"))
model.add(tf.keras.layers.Dense(y_train.shape[1], activation = "softmax"))
model.summary()
```

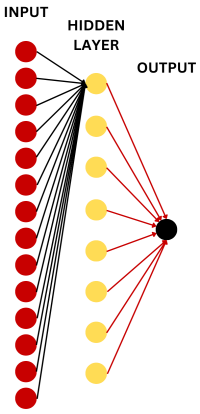
Dilakukan import tensorflow untuk membangun dan melatih data menggunakan model ANN. Dilakukan inisiasi model sequental yang berguna agar nantinya setiap lapisan memiliki input dari lapisan sebelumnya dan memiliki output dari lapisan berikutnya. Ditentukan juga terdapat 8 neuron lalu menetapkan bentuk untuk lapisan pertama. Digunakan fungsi ReLU untuk membantu mengatasi masalah dan mempercepat konvergensi dalam data *train*. Digunakan juga fungsi softmax untuk mengubah output dari lapisan menjadi probabilitas untuk setiap kelas.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 8)                 128

 dense_9 (Dense)             (None, 3)                 27

=================================================================
Total params: 155 (620.00 Byte)
Trainable params: 155 (620.00 Byte)
Non-trainable params: 0 (0.00 Byte)
```

Didapat jumlah parameter yang dapat di *train* dalam model ini, yaitu 155. Ini adalah jumlah dari semua parameter yang dapat di *train* dalam semua lapisan. Karena semua parameter bisa di *train* didapat parameter yang tidak bisa di *train* ada 0.

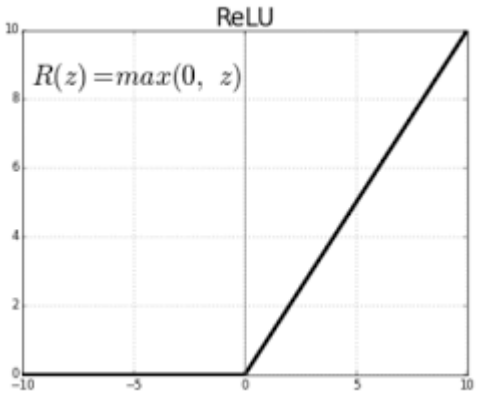
```
[ ]  X.shape

(292364, 15)
```

Lalu didapat ada sebanyak 292364 baris dan 15 kolom fitur dengan ilustrasi ANN sebagai berikut:




Catatan: operasi ke *hidden layer* pertama juga berlaku ke 7 *hidden layer* lainnya.

Hidden layer dalam model ini terdiri dari satu lapisan tersembunyi dengan 8 neuron dan fungsi aktivasi ReLU. Lapisan ini memungkinkan model untuk mempelajari representasi yang lebih kompleks dari data input sebelum menghasilkan output yang diklasifikasikan oleh lapisan output menggunakan fungsi aktivasi softmax.




Rumus pertama yang kita pakai adalah salah satu fungsi aktivasi paling umum yang dikenal sederhana, efisien, dan kecenderungannya mengatasi masalah vanishing gradient pada pelatihan model. Nilai x adalah input ke fungsi, akan menghasilkan output yang sama dengan inputnya jika inputnya positif, dan 0 jika inputnya negatif.

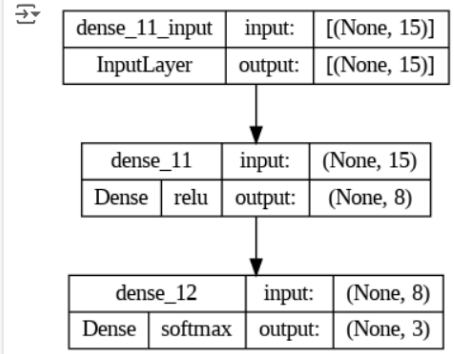
$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Rumus kedua yang kita pakai adalah softmax yaitu fungsi untuk mengubah nilai-nilai dalam vektor menjadi distribusi probabilitas, dengan penjelasan:

- z adalah vektor input dengan K elemen,
- e adalah bilangan Euler (konstanta eksponensial),
- softmax(z)i adalah probabilitas bahwa input termasuk ke dalam kelas i, dan
- $\sum_{j=1}^{K} e^{z_j}$ adalah jumlah dari nilai eksponensial semua elemen dalam vektor zzz, yang disebut sebagai denominasi atau normalisasi.

### 3.4 Visualisasi Arsitektur Model

```
tf.keras.utils.plot_model(
    model,
    show_shapes = True,
    show_layer_activations = True,
    to_file = "keras_sequential_model2.png"
)
```




Dibuat input layer yang memiliki 15 kolom dan diturunkan menjadi 8 kolom dab akhirnya outputnya 3 kolom sesuai ukuran matrix target.

### 3.5 Melakukan training pada model

```
model.compile(optimizer = "adam",
              loss = "categorical_crossentropy",
              metrics = ["accuracy"]) #ditentukan hyperparameter: optimizer, loss function dan accuracy
model.fit(X_train, y_train, validation_split = 0.2, batch_size = 100, epochs = 25, )
```

Digunakan Adaptive Moment Estimation untuk mengatur bagaimana bobot diperbarui. Lalu loss function untuk menghitung seberapa baik prediksi model dibandingkan target sebenarnya dan Metrik yang digunakan untuk mengevaluasi performa dari model. dengan X sebagai data fitur dan Y sebagai data train. Akan digunakan 20% dari X dan Y untuk mengevaluasi performa model di setiap epoch. Data *train* dibagi menjadi batch kecil dengan masing - masing 100 sampel dan model akan melewati seluruh data *train* sebanyak 25 kali. Setiap epoc memberi kesempatan bagi model untuk memperbarui bobot dan mengurangi loss.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$
$$\hat{m}_t = m_t / (1 - \beta_1^t)$$
$$\hat{v}_t = v_t / (1 - \beta_2^t)$$
$$\theta = \theta - (\alpha * \hat{m}_t / \sqrt{(\hat{v}_t + \varepsilon)})$$

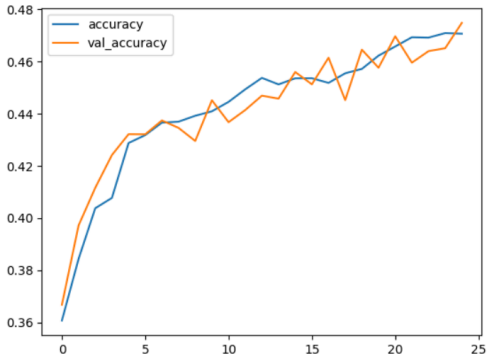
Berikut adalah rumus dari algoritma adam, dengan penjelasan berikut:

1. Baris pertama ialah proses memperbarui estimasi momentum pertama yang bias.
2. Baris kedua ialah proses memperbarui estimasi momentum kedua yang bias.
3. Baris terakhir ialah memperbarui parameter.

### 3.6 Melakukan perhitungan akurasi

```
hasil = pd.DataFrame(model.history.history)
display(hasil[['accuracy', 'val_accuracy']].plot())
```

```
<Axes: >
```




Dari model tersebut didapat plot dengan garis biru sebagai akurasi data training dan garis jingga sebagai akurasi data test.

```
[ ] predictions = model.predict(X_test)
    predictions
```

```
2285/2285 [==============================] - 3s 1ms/step
array([[0.33432436, 0.28199503, 0.38368064],
       [0.42459875, 0.2590447 , 0.31635663],
       [0.36856315, 0.28319478, 0.34824204],
       ...,
       [0.3632024 , 0.34731618, 0.28948146],
       [0.41625294, 0.05647166, 0.5272754 ],
       [0.18697894, 0.36436808, 0.44865295]], dtype=float32)
```

Metode yang digunakan untuk menghasilkan output prediksi dari model untuk fitur X agar dapat menguji kinerja model.

```
[ ] import numpy as np
    predictions = np.round(predictions)
    predictions
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       ...,
       [0., 0., 1.],
       [0., 0., 0.],
       [0., 0., 1.]], dtype=float32)
```

Dilakukan pembulatan nilai prediksi probabilitas yang dihasilkan oleh model menjadi nilai diskrit (0 atau 1).

```
from sklearn.metrics import accuracy_score
print("Hasil skor akurasi: ", accuracy_score(y_test, predictions) * 100, "%")
```

```
Hasil skor akurasi:  24.987449226415954 %
```

Dilakukan penghitungan terhadap akurasi klasifikasi dengan membandingkan setiap elemen di Y dan prediksi untuk menghitung persentase yang sesuai.

### 3.7 Melakukan improvisasi dengan menambah layer

```
[ ] model2 = tf.keras.Sequential()
    model2.add(tf.keras.layers.Dense(8, input_shape = (X_train.shape[1], ), activation = "relu"))
    model2.add(tf.keras.layers.Dense(4, activation = "relu"))
    model2.add(tf.keras.layers.Dense(y_train.shape[1], activation = "softmax"))
    model2.summary()
```

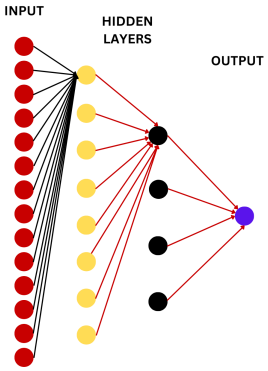
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 8)                 128

 dense_6 (Dense)             (None, 4)                 36

 dense_7 (Dense)             (None, 3)                 15

=================================================================
Total params: 179 (716.00 Byte)
Trainable params: 179 (716.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Dilakukan penambahan pada lapisan dense kedua sebanyak 4 neuron. Setelah di improvisasi, nilai parameter meningkat secara signifikan karena adanya lapisan tersembunyi yang ditambah membuat terhubungnya neuron yang ada di lapisan tersebut terhubung dengan neuron lapisan sebelumnya dan sesudahnya. Didapat parameter baru sebanyak 179.

```
model2.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics = ["accuracy"])
early_stop = tf.keras.callbacks.EarlyStopping(monitor = "val_loss", patience = 10)
model2.fit(X_train, y_train, validation_split = 0.2, batch_size = 100, epochs = 500, callbacks = [early_stop])
```

Jika pada epoch tertentu, metrik yang dipantau tidak membaik, maka akan dihentikan *training* tersebut lebih awal. Pada kasus ini jika val_loss tidak membaik setelah 10 epoch berturut-turut maka *training* akan dihentikan. Lalu pada kasus ini akan digunakan 20% data untuk mengevaluasi performa model selama pelatihan dengan *batch size* nya 100 dan maksimum 500 epoch.

Untuk model kedua ini dapat ditunjukkan dengan ilustrasi sebagai berikut:



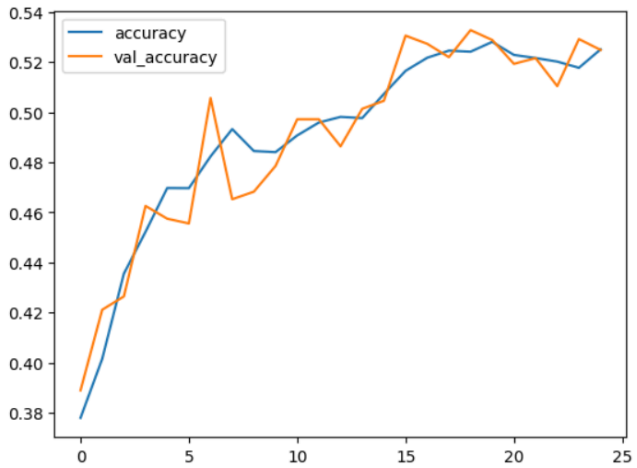

Catatan: operasi ke *hidden layer* pertama dan kedua juga berlaku ke 7 *hidden layer* pertama lainnya dan 3 *hidden layer* kedua lainnya.

Arsitektur model yang dimodifikasi ini memiliki dua hidden layer: satu dengan 8 neuron dan aktivasi ReLU, serta satu lagi dengan 4 neuron dan aktivasi ReLU. Penambahan hidden layer kedua dapat membantu dalam mempelajari representasi yang lebih baik dari data sebelum klasifikasi akhir dilakukan oleh lapisan output. Lapisan ini memungkinkan model untuk mempelajari representasi yang lebih kompleks dari data *hidden layer* sebelum menghasilkan output yang diklasifikasikan oleh lapisan output menggunakan fungsi aktivasi softmax.

### 3.8 Menghitung akurasi pada model 2

```
hasil2 = pd.DataFrame(model2.history.history)
display(hasil[['accuracy', 'val_accuracy']].plot())
```

```
<Axes: >
```




Dengan iterasi epoch yang semakin besar didapat grafik seperti diatas.

```
[ ]  predictions2 = model2.predict(X_test)
     predictions2
```

```
685/685 [==============================] - 1s 1ms/step
array([[2.9795402e-01, 4.4035947e-01, 2.6168656e-01],
       [9.3660660e-02, 2.1886048e-04, 9.0612048e-01],
       [3.8329414e-01, 2.0175118e-02, 5.9653080e-01],
       ...,
       [4.4076553e-01, 1.8275574e-02, 5.4095888e-01],
       [2.7137464e-01, 6.3368790e-03, 7.2228849e-01],
       [2.2074443e-01, 7.7999145e-02, 7.0125645e-01]], dtype=float32)
```

```
[ ]  predictions2 = np.round(predictions2)
     predictions2
     print("Hasil skor akurasi: ", accuracy_score(y_test, predictions2) * 100, "%")
```

```
Hasil skor akurasi:  58.796951302998494 %
```

Lalu didapat nilai akurasi sebesar 58,79% yang sudah jauh lebih baik dari iterasi sebelum di improvisasi.

# BAB IV
# KESIMPULAN

Setelah menjalankan dan mengevaluasi program yang telah dibuat, pada model pertama kami hanya mendapat akurasi di angka 24,98% dimana angka masih cukup rendah dan dapat ditingkatkan. Sehingga, kami membuat model kedua dimana menambahkan fungsi aktivasi dan hyper parameter ke dalam model awal kami, dimana kali ini tingkat akurasi meningkat hingga 58,79%.

Maka dengan membuat kedua model tersebut, kami mendapatkan kesimpulan bahwa Metode Artificial Neural Network yang telah kami buat mencapai akurasi sebesar 58,79% setelah dilakukan proses improvisasi. Model ini tentunya masih dapat ditingkatkan guna memenuhi standar bidang kesehatan yang umumnya membutuhkan tingkat keakuratan yang cukup tinggi. Akan tetapi, dalam kasus atau masalah kami, model ini dapat memberikan gambaran yang cukup baik dalam menangani kompleksitas masalah yang kami hadapi. Selain itu, model ini masih memiliki banyak ruang untuk dikembangkan lebih lanjut sehingga menjadi semakin efektif untuk mendiagnosa penyakit stress di kemudian harinya.

## DAFTAR PUSTAKA

Towards Data Science. (2018). Categorical Feature Encoding. Diakses pada Mei 2024 dari https://towardsdatascience.com/categorical-feature-encoding-547707acf4e5#45b1

Real Python. (n.d.). Building A Simple AI Game Bot With Python. Diakses pada Mei 2024 dari https://realpython.com/python-ai-neural-network/