

# UNIVERSITAS GUNADARMA



## PRAKTIKUM KECERDASAN ARTIFICIAL

### MANUAL BOOK

**“Analisis Kemiripan Teks dengan Embedding BERT Pretrained”**

|                  |          |   |
|------------------|----------|---|
| <b>NAMA</b>      | <b>:</b> | <b>Muhammad Tarmidzi Bariq</b>            |
| <b>NPM</b>       | <b>:</b> | <b>51422161</b>                           |
| <b>KELAS</b>     | <b>:</b> | <b>3IA11</b>                              |
| <b>PRAKTIKUM</b> | <b>:</b> | <b>Kecerdasan Artifisial</b>              |
| <b>PJ</b>        | <b>:</b> | <b>Gilbert Jefferson Faozato Mendrofa</b> |

**Ditulis Guna Melengkapi Sebagai Syarat Praktikum Kecerdasan Artificial  
jenjang S1**

**LABORATORIUM INFORMATIKA**

**UNIVERSITAS GUNADARMA**

**2024**

## **KATA PENGANTAR**

Puji syukur kehadiran Tuhan Yang Maha Esa atas segala karunia nikmatnya sehingga Manual Book yang berjudul Analisis Kemiripan Teks dengan Embedding BERT Pretrained ini dapat diselesaikan dengan maksimal, tanpa ada halangan yang berarti. Makalah ini disusun untuk memenuhi Praktikum Kecerdasan Artificial yang dibimbing oleh PJ yaitu Gilbert Jefferson Faozato Mendrofa

Makalah ini membicarakan penggunaan NLP dengan model BERT untuk menganalisis data kebugaran berdasarkan kesamaan teks. Saya harap makalah ini bisa membantu perkembangan teknologi di bidang kecerdasan buatan, terutama dalam personalisasi program kebugaran.

Terima kasih kepada Kakak Gilbert Jefferson Faozato Mendrofa sebagai pembimbing, dan semua yang telah mendukung penulisan makalah ini. Semoga makalah ini dapat bermanfaat bagi pembaca.

Depok, 12 November 2024

Muhammad Tarmidzi Bariq

## DAFTAR ISI

|                                      |           |
|--------------------------------------|-----------|
| <b>KATA PENGANTAR.....</b>           | <b>2</b>  |
| <b>DAFTAR ISI.....</b>               | <b>3</b>  |
| <b>BAB I.....</b>                    | <b>4</b>  |
| <b>PENDAHULUAN .....</b>             | <b>4</b>  |
| 1.1    Latar Belakang .....          | 4         |
| 1.2    Tujuan.....                   | 4         |
| <b>BAB II .....</b>                  | <b>5</b>  |
| <b>PEMBAHASAN .....</b>              | <b>5</b>  |
| 2.1    Artificial Intelligence ..... | 5         |
| 2.2    Pengertian NLP .....          | 5         |
| 2.3    Google Bert .....             | 6         |
| <b>BAB III.....</b>                  | <b>7</b>  |
| <b>LOGIKA PROGRAM.....</b>           | <b>7</b>  |
| <b>BAB IV .....</b>                  | <b>16</b> |
| <b>PENUTUP.....</b>                  | <b>16</b> |
| 4.1    Kesimpulan.....               | 16        |
| 4.2    Saran.....                    | 16        |

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Penggunaan data tentang latihan, waktu, dan detak jantung sangat penting dalam mendukung program kesehatan yang berhasil. Untuk meningkatkan analisis data teks seperti jenis latihan (Yoga, HIIT), diperlukan teknik NLP (Pemrosesan Bahasa Alamiah) yang canggih. Salah satu contoh model yang dapat digunakan adalah BERT (Bidirectional Encoder Representations from Transformers). BERT dapat membuat representasi vektor dari teks yang membantu kita memahami hubungan makna antara jenis latihan.

Dengan menggunakan BERT, penelitian ini ingin menganalisis hubungan antara kategori latihan melalui penggabungan teks. Downsampling dilakukan untuk menjaga keseimbangan antara kategori data agar analisis menjadi lebih tepat. Dengan menggunakan pendekatan ini, diharapkan dapat memperoleh pemahaman yang lebih mendalam dalam mengelola dan menyesuaikan program latihan bagi anggota gym. Hal ini juga dapat membuka peluang penggunaan teknologi ini dalam bidang kesehatan yang lebih luas.

### **1.2 Tujuan**

1. Menganalisis data kebugaran, khususnya jenis latihan, menggunakan teknik NLP untuk memahami hubungan antar kategori latihan.
2. Mengaplikasikan model BERT untuk menghasilkan embedding teks yang merepresentasikan jenis latihan dalam bentuk vektor, sehingga memungkinkan analisis kesamaan antar latihan.
3. Mengatasi ketidakseimbangan data kategori latihan melalui teknik downsampling untuk memperoleh hasil analisis yang lebih akurat dan tidak bias.

## **BAB II**

### **PEMBAHASAN**

#### **2.1 Artificial Intelligence**

Pada tahun 1950, Alan Turing, seorang pionir kecerdasan buatan (AI) dan ahli matematika asal Inggris, menjalankan sebuah percobaan yang dikenal sebagai Turing Test. Percobaan ini melibatkan penggunaan komputer yang terhubung melalui terminal di tempat yang berjauhan. Satu ujung terminal dilengkapi dengan perangkat lunak kecerdasan buatan, sementara ujung lainnya memiliki seorang operator. Menariknya, operator tersebut tidak mengetahui bahwa di ujung terminal lainnya terdapat perangkat lunak kecerdasan buatan. Selama percobaan ini, keduanya berkomunikasi, di mana terminal di ujung memberikan respons terhadap serangkaian pertanyaan yang diajukan oleh operator. Sang operator, tanpa menyadari keberadaan AI, meyakini bahwa ia sedang berinteraksi dengan operator manusia yang berada di terminal yang berbeda.

#### **2.2 Pengertian NLP**

Pemrosesan bahasa alami (NLP) adalah subbidang ilmu komputer dan kecerdasan buatan (AI) yang menggunakan machine learning untuk memungkinkan komputer memahami dan berkomunikasi dengan bahasa manusia.

NLP memungkinkan komputer dan perangkat digital untuk mengenali, memahami, dan menghasilkan teks dan ucapan dengan menggabungkan linguistik komputasi, pemodelan berbasis aturan bahasa manusia, bersama dengan pemodelan statistik, machine learning dan pembelajaran mendalam.

Penelitian NLP telah membantu memungkinkan era AI generatif, mulai dari keterampilan komunikasi model bahasa besar (LLM) hingga kemampuan model pembuatan gambar untuk memahami permintaan. NLP sudah menjadi bagian dari kehidupan sehari-hari bagi banyak orang, memberdayakan mesin pencari, mendukung chatbot untuk layanan pelanggan dengan perintah lisan, sistem GPS yang dioperasikan dengan suara, dan asisten digital penjawab pertanyaan di smartphone seperti Amazon Alexa, Apple Siri dan Cortana Microsoft. Ini menawarkan manfaat di banyak industri dan aplikasi.

1. Otomatisasi tugas yang berulang
2. Analisis dan insight data yang ditingkatkan
3. Pencarian yang ditingkatkan
4. Pembuatan Konten

### **2.3 Google Bert**

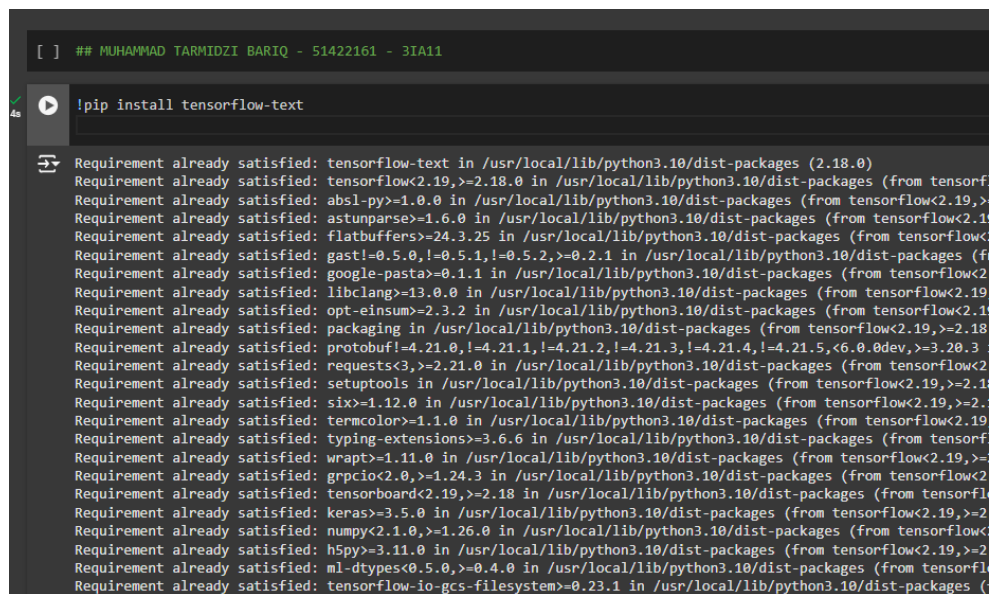
Teknik atau sistem berbasis neural network. Neural network sendiri adalah jaringan saraf tiruan dalam machine learning dan artificial intelligence yang mencoba meniru sistem kerja otak manusia. Sistem ini digunakan untuk pre-training natural language processing, di mana mesin bisa belajar dan meningkatkan kemampuannya. BERT lebih berfokus pada meningkatkan experience atau pengalaman pengguna saat melakukan pencarian. Hanya saja, tentu tetap penting untuk membuat konten situs seperti artikel dan lainnya dengan mempertimbangkan BERT.

## BAB III

### LOGIKA PROGRAM

#### 1. !pip install tensorflow-text

Ekstensi dari TensorFlow yang dirancang untuk membantu dalam pemrosesan teks. Paket ini digunakan untuk mengolah teks dan berinteraksi dengan model NLP berbasis TensorFlow, termasuk untuk tokenisasi dan preprocessing teks yang kompatibel dengan model seperti BERT.



```
[ ] ## MUHAMMAD TARMIDZI BARIQ - 51422161 - 3IA11

!pip install tensorflow-text

Requirement already satisfied: tensorflow-text in /usr/local/lib/python3.10/dist-packages (2.18.0)
Requirement already satisfied: tensorflow<2.19,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow-text)
```

#### 2. Import modul

- import numpy as np  
Digunakan untuk komputasi numerik dan manipulasi array.
- import pandas as pd  
Pustaka untuk manipulasi dan analisis data dalam bentuk tabel (data frame).
- import matplotlib.pyplot as plt  
Pustaka utama untuk visualisasi data dalam Python.
- import seaborn as sns  
Pustaka visualisasi data berbasis matplotlib yang lebih sederhana dan estetik.
- import tensorflow as tf  
Framework untuk machine learning dan deep learning.
- import tensorflow\_hub as hub

Modul untuk menggunakan model pra-latih dari TensorFlow Hub.

- `import tensorflow_text as text`

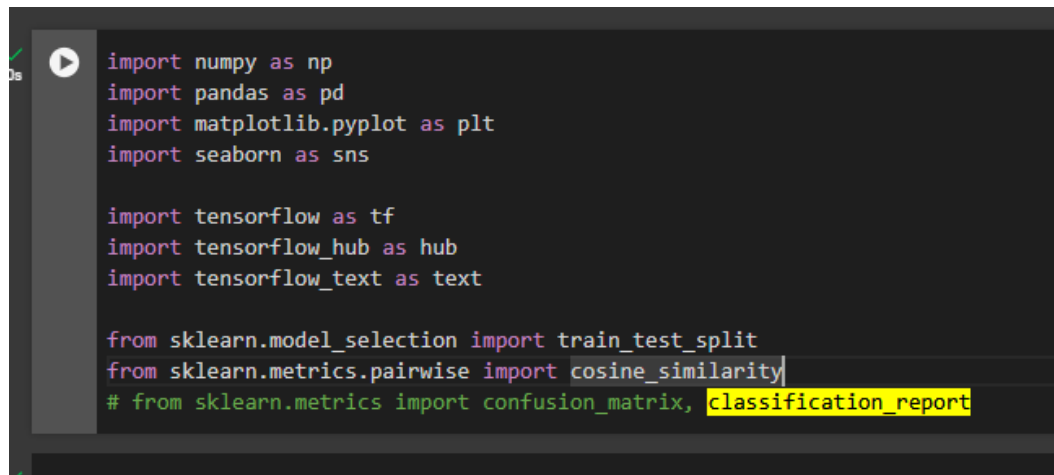
Ekstensi untuk tensorflow yang membantu dalam preprocessing teks, seperti tokenisasi, yang sering dibutuhkan saat bekerja dengan model bahasa alami seperti BERT.

- `from sklearn.model_selection import train_test_split`

Fungsi dari scikit-learn untuk membagi dataset menjadi data latih (training) dan data uji (testing).

- `from sklearn.metrics.pairwise import cosine_similarity`

Digunakan untuk mengukur kesamaan antara dua vektor, seperti embedding teks, berdasarkan sudut di antara vektor tersebut (similarity measure).



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

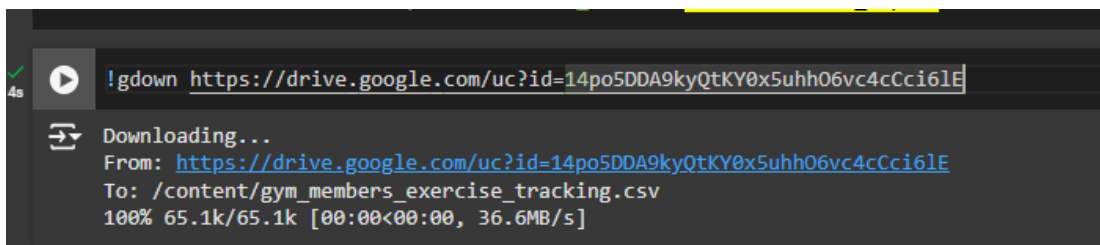
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text

from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity
# from sklearn.metrics import confusion_matrix, classification_report
```

### 3. upload file

`!gdown`

Mengunduh file dari Google Drive langsung ke dalam lingkungan notebook,



```
!gdown https://drive.google.com/uc?id=14po5DDA9kyQtKY0x5uhh06vc4cCci6lE
```

Downloading...

From: <https://drive.google.com/uc?id=14po5DDA9kyQtKY0x5uhh06vc4cCci6lE>

To: /content/gym\_members\_exercise\_tracking.csv

100% 65.1k/65.1k [00:00<00:00, 36.6MB/s]



#### 4. Membaca isi file

Menggunakan pustaka pandas untuk membaca file CSV bernama `gym_members_exercise_tracking.csv`.

```
df = pd.read_csv('gym_members_exercise_tracking.csv')
df
```

|     | Age | Gender | Weight (kg) | Height (m) | Max_BPM | Avg_BPM | Resting_BPM | Session_Duration (hours) | Calories_Burned | Workout_Type | Fat_Percentage | Water_Intake (liters) | Workout_Frequency (days/week) | Experience_Level | BMI   |
|-----|-----|--------|-------------|------------|---------|---------|-------------|--------------------------|-----------------|--------------|----------------|-----------------------|-------------------------------|------------------|-------|
| 0   | 56  | Male   | 88.3        | 1.71       | 180     | 157     | 60          | 1.69                     | 1313.0          | Yoga         | 12.6           | 3.5                   | 4                             | 3                | 30.20 |
| 1   | 46  | Female | 74.9        | 1.53       | 179     | 151     | 66          | 1.30                     | 883.0           | HIIT         | 33.9           | 2.1                   | 4                             | 2                | 32.00 |
| 2   | 32  | Female | 68.1        | 1.66       | 167     | 122     | 54          | 1.11                     | 677.0           | Cardio       | 33.4           | 2.3                   | 4                             | 2                | 24.71 |
| 3   | 25  | Male   | 53.2        | 1.70       | 190     | 164     | 56          | 0.59                     | 532.0           | Strength     | 28.8           | 2.1                   | 3                             | 1                | 18.41 |
| 4   | 38  | Male   | 46.1        | 1.79       | 188     | 158     | 68          | 0.64                     | 556.0           | Strength     | 29.2           | 2.8                   | 3                             | 1                | 14.39 |
| ... | ... | ...    | ...         | ...        | ...     | ...     | ...         | ...                      | ...             | ...          | ...            | ...                   | ...                           | ...              | ...   |
| 968 | 24  | Male   | 87.1        | 1.74       | 187     | 158     | 67          | 1.57                     | 1364.0          | Strength     | 10.0           | 3.5                   | 4                             | 3                | 28.77 |
| 969 | 25  | Male   | 66.6        | 1.61       | 184     | 166     | 56          | 1.38                     | 1260.0          | Strength     | 25.0           | 3.0                   | 2                             | 1                | 25.69 |
| 970 | 59  | Female | 60.4        | 1.76       | 194     | 120     | 53          | 1.72                     | 929.0           | Cardio       | 18.8           | 2.7                   | 5                             | 3                | 19.50 |
| 971 | 32  | Male   | 126.4       | 1.83       | 198     | 146     | 62          | 1.10                     | 883.0           | HIIT         | 28.2           | 2.1                   | 3                             | 2                | 37.74 |
| 972 | 46  | Male   | 88.7        | 1.63       | 166     | 146     | 66          | 0.75                     | 542.0           | Strength     | 28.8           | 3.5                   | 2                             | 1                | 33.38 |

973 rows x 15 columns

#### 5. `df.info()`

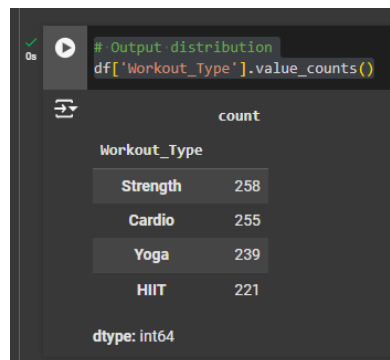
Berfungsi untuk menampilkan ringkasan informasi tentang DataFrame `df`,

```
# Data information
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 973 entries, 0 to 972
Data columns (total 15 columns):
 #   Column                                Non-Null Count  Dtype  
---  -
 0   Age                                   973 non-null   int64  
 1   Gender                               973 non-null   object  
 2   Weight (kg)                          973 non-null   float64 
 3   Height (m)                           973 non-null   float64 
 4   Max_BPM                              973 non-null   int64  
 5   Avg_BPM                              973 non-null   int64  
 6   Resting_BPM                          973 non-null   int64  
 7   Session_Duration (hours)             973 non-null   float64 
 8   Calories_Burned                      973 non-null   float64 
 9   Workout_Type                         973 non-null   object  
10   Fat_Percentage                       973 non-null   float64 
11   Water_Intake (liters)                973 non-null   float64 
12   Workout_Frequency (days/week)       973 non-null   int64  
13   Experience_Level                     973 non-null   int64  
14   BMI                                  973 non-null   float64 
dtypes: float64(7), int64(6), object(2)
memory usage: 114.1+ KB
```

6. `df['Workout_Type'].value_counts()`

Berfungsi untuk menampilkan distribusi nilai dalam kolom `Workout_Type` pada DataFrame `df`

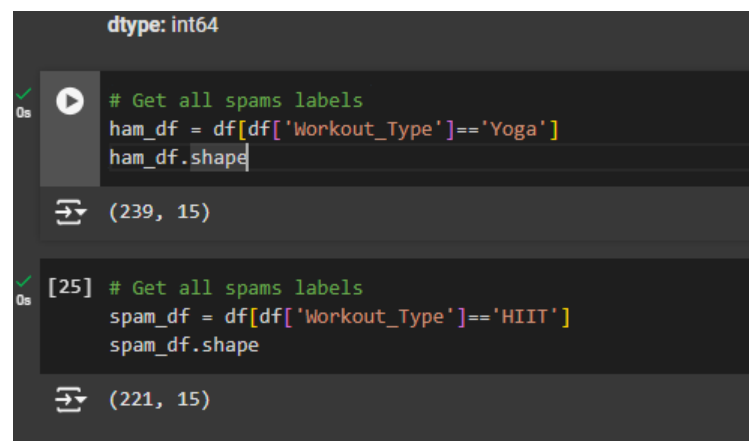


7. `ham_df = df[df['Workout_Type']=='Yoga']`

`ham_df.shape`

Kode ini bertujuan untuk memfilter DataFrame `df` agar hanya menyertakan baris yang memiliki nilai 'Yoga' pada kolom 'Workout\_Type'.

`.shape` digunakan untuk mendapatkan ukuran dari DataFrame yang difilter tersebut dalam bentuk tuple, di mana nilai pertama adalah jumlah baris dan nilai kedua adalah jumlah kolom.



8. `ham_df_downsampled = ham_df.sample(spam_df.shape[0])`

Baris ini mengambil sampel acak dari `ham_df` sebanyak jumlah baris yang ada pada `spam_df` (nilai `spam_df.shape[0]` mewakili jumlah baris). Ini dilakukan untuk menyeimbangkan data, sehingga `ham_df_downsampled` memiliki ukuran yang sama dengan `spam_df`.

ham\_df\_downsampled.shape

Bagian ini menampilkan ukuran dari ham\_df\_downsampled. Hasil (221, 15) berarti bahwa data ham\_df\_downsampled berisi 221 baris dan 15 kolom.

```
# Get random sample from ham data in size of spam data
ham_df_downsampled = ham_df.sample(spam_df.shape[0])
ham_df_downsampled.shape
```

(221, 15)

9. balanced\_df['Workout\_Type'].value\_counts()

Perintah ini menghitung frekuensi dari setiap nilai unik dalam kolom Workout\_Type pada DataFrame balanced\_df

```
[28] # New balanced data distribution
balanced_df['Workout_Type'].value_counts()
```

|              | count |
|--------------|-------|
| Workout_Type |       |
| Yoga         | 221   |
| HIIT         | 221   |

dtype: int64

10. balanced\_df['spam'] = balanced\_df['Workout\_Type'].apply(lambda x: 1 if x == 'Yoga' else 0)

Kode ini menambahkan kolom baru bernama spam ke DataFrame balanced\_df.

Kolom spam akan berisi nilai 1 jika Workout\_Type adalah Yoga dan 0 untuk nilai lain (dalam hal ini, HIIT).

Ini sering digunakan dalam tugas klasifikasi biner untuk memberikan label numerik (0 atau 1) bagi setiap kategori.

balanced\_df.sample(5)

Menampilkan 5 baris sampel acak dari balanced\_df untuk memverifikasi bahwa kolom spam telah ditambahkan dengan benar

Tabel hasil menunjukkan bahwa kolom spam mencerminkan nilai 1 untuk Yoga dan 0 untuk HIIT.

```
# Membuat kolom baru dengan nilai yang diencode: 'Yoga' sebagai 1 dan lainnya sebagai 0
balanced_df['spam'] = balanced_df['Workout_Type'].apply(lambda x: 1 if x == 'Yoga' else 0)

# Menampilkan sampel data untuk memverifikasi hasil
balanced_df.sample(5)
```

|     | Age | Gender | Weight (kg) | Height (m) | Max_BPM | Avg_BPM | Resting_BPM | Session_Duration (hours) | Calories_Burned | Workout_Type | Fat_Percentage | Water_Intake (liters) | Workout_Frequency (days/week) | Experience_Level | BMI   | spam |
|-----|-----|--------|-------------|------------|---------|---------|-------------|--------------------------|-----------------|--------------|----------------|-----------------------|-------------------------------|------------------|-------|------|
| 950 | 31  | Female | 76.7        | 1.62       | 174     | 127     | 74          | 1.39                     | 883.0           | Yoga         | 28.1           | 2.3                   | 4                             | 2                | 29.23 | 1    |
| 684 | 38  | Female | 56.3        | 1.60       | 180     | 142     | 53          | 1.21                     | 859.0           | Yoga         | 28.8           | 1.9                   | 4                             | 2                | 21.99 | 1    |
| 231 | 45  | Female | 44.5        | 1.65       | 162     | 148     | 71          | 1.44                     | 959.0           | Yoga         | 30.7           | 1.6                   | 3                             | 1                | 16.35 | 1    |
| 579 | 20  | Female | 72.5        | 1.73       | 199     | 160     | 68          | 1.27                     | 1016.0          | HIT          | 34.8           | 2.5                   | 3                             | 2                | 24.26 | 0    |
| 426 | 56  | Male   | 82.5        | 1.87       | 187     | 124     | 74          | 1.51                     | 927.0           | Yoga         | 13.2           | 3.5                   | 5                             | 3                | 23.59 | 1    |

## 11. Memisahkan Data Menjadi Fitur dan Target

- **X\_train** dan **X\_test**: Ini adalah set data fitur untuk pelatihan dan pengujian. Dalam hal ini, fitur yang digunakan adalah kolom **Workout\_Type**.
- **y\_train** dan **y\_test**: Ini adalah label target untuk pelatihan dan pengujian, diambil dari kolom **spam** yang telah dikonversi menjadi kelas biner.
- **stratify=balanced\_df['spam']**: Parameter ini memastikan bahwa pembagian data tetap mempertahankan proporsi kelas yang sama dalam set pelatihan dan pengujian, sehingga distribusi kelas tetap seimbang di kedua set.

```
[30] # Membagi data menjadi fitur dan target
X_train, X_test, y_train, y_test = train_test_split(
    balanced_df['Workout_Type'], # Fitur, di sini menggunakan Workout_Type
    balanced_df['spam'], # Target (kelas biner)
    stratify=balanced_df['spam'] # Stratifikasi untuk menjaga proporsi kelas
)
```

## 12. bert\_preprocess

Menangani tugas preprocessing seperti tokenisasi teks, normalisasi huruf kecil, dan penambahan token khusus yang dibutuhkan oleh model BERT.

### Bert\_encoder

Mengubah teks menjadi representasi vektor (embedding) yang bisa digunakan untuk analisis atau tugas-tugas NLP

```
# Preprocessing model
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")

# BERT model
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

### 13. Fungsi get\_sentence\_embedding

Mengambil teks (sentences), memprosesnya melalui bert\_preprocess, dan kemudian menghasilkan embedding menggunakan bert\_encoder. Hasil akhirnya adalah representasi vektor (embedding) dari kalimat tersebut

```
# Fungsi untuk menghasilkan embedding dari kalimat menggunakan BERT
def get_sentence_embedding(sentences):
    preprocessed_text = bert_preprocess(sentences) # Preprocessing BERT pada kalimat
    return bert_encoder(preprocessed_text)['pooled_output'] # Mengembalikan embedding BERT
```

### 14. sample\_sentences mengambil 6 contoh acak dari kolom Workout\_Type di balanced\_df, yang menggantikan kalimat manual seperti "banana" dan "elon musk".

get\_sentence\_embedding(sample\_sentences) menghasilkan embedding untuk kalimat-kalimat yang diambil dari file CSV.

cosine\_similarity menghitung kesamaan kosinus antara embedding untuk melihat seberapa mirip kategori Workout\_Type satu dengan lainnya.

```
# Ambil contoh nilai dari kolom Workout_Type untuk digunakan sebagai input
sample_sentences = balanced_df['Workout_Type'].sample(6).tolist() # Ambil 6 sampel acak

# Generate embedding menggunakan fungsi get_sentence_embedding
e = get_sentence_embedding(sample_sentences)

# Hitung kemiripan kosinus antara beberapa embedding
print("Cosine similarity between first and second:", cosine_similarity([e[0]], [e[1]]))
print("Cosine similarity between third and fourth:", cosine_similarity([e[2]], [e[3]]))
print("Cosine similarity between fifth and sixth:", cosine_similarity([e[4]], [e[5]]))
```

↔ Cosine similarity between first and second: [[1.]]  
Cosine similarity between third and fourth: [[1.0000001]]  
Cosine similarity between fifth and sixth: [[0.97140557]]

### 15. Membuat variable dengan type data Array

Digunakan untuk keterangan

```
sample_texts = [
    "Intense weightlifting session", # High strength
    "Moderate cardio workout",      # Moderate strength
    "Relaxing yoga class",          # Low strength
]
```

## 16. Menghitung kesamaan kosinus

- `similarity_1_2` menghitung kesamaan kosinus antara embedding teks pertama (Intense weightlifting session) dan embedding teks kedua (Moderate cardio workout).
- `similarity_2_3` menghitung kesamaan kosinus antara embedding teks kedua (Moderate cardio workout) dan embedding teks ketiga (Relaxing yoga class).
- Fungsi `cosine_similarity` mengembalikan nilai antara -1 dan 1, di mana nilai mendekati 1 menunjukkan bahwa kedua embedding tersebut sangat mirip

```
# Menghitung kesamaan antara teks untuk mengecek hubungan intensitas
similarity_1_2 = cosine_similarity([embeddings[0]], [embeddings[1]])[0][0]
similarity_2_3 = cosine_similarity([embeddings[1]], [embeddings[2]])[0][0]

print(f"Cosine Similarity between 'Intense weightlifting session' and 'Moderate cardio workout': {similarity_1_2}")
print(f"Cosine Similarity between 'Moderate cardio workout' and 'Relaxing yoga class': {similarity_2_3}")
```

Cosine Similarity between 'Intense weightlifting session' and 'Moderate cardio workout': 0.9610573053359985  
Cosine Similarity between 'Moderate cardio workout' and 'Relaxing yoga class': 0.9655733704566956

## 17. Fungsi Categorize\_strength

- Fungsi ini mengonversi text menjadi huruf kecil (`text.lower()`) untuk menghindari perbedaan huruf besar/kecil.
- Jika kalimat mengandung kata "intense" atau "heavy", maka fungsi akan mengembalikan "High Strength", yang menunjukkan tingkat kekuatan tinggi.
- Jika kalimat mengandung kata "moderate", maka fungsi akan mengembalikan "Moderate Strength", yang menunjukkan tingkat kekuatan sedang.
- Jika kalimat mengandung kata "relaxing" atau "light", maka fungsi akan mengembalikan "Low Strength", yang menunjukkan tingkat kekuatan rendah.
- Jika tidak ada kata kunci yang cocok, fungsi akan mengembalikan "Unknown Strength", yang berarti bahwa tingkat kekuatan tidak diketahui.

```
# Menilai "strength" dari kalimat berdasarkan nilai similaritas
def categorize_strength(text):
    if "intense" in text.lower() or "heavy" in text.lower():
        return "High Strength"
    elif "moderate" in text.lower():
        return "Moderate Strength"
    elif "relaxing" in text.lower() or "light" in text.lower():
        return "Low Strength"
    else:
        return "Unknown Strength"
```

## 18. Kategorikan contoh teks berdasarkan strength

```
strength_categories = [categorize_strength(text) for text in sample_texts]
```

- Baris ini membuat list comprehension untuk menerapkan fungsi `categorize_strength` pada setiap teks dalam `sample_texts`.
- Hasilnya adalah list `strength_categories` yang berisi kategori intensitas (misalnya, "High Strength", "Moderate Strength", atau "Low Strength") untuk setiap teks dalam `sample_texts`.

```
for text, strength in zip(sample_texts, strength_categories):
```

```
    print(f'{text}': {strength})
```

- Loop ini menggabungkan setiap teks dari `sample_texts` dengan kategori intensitas yang sesuai dari `strength_categories` menggunakan fungsi `zip`.
- Setiap pasangan teks dan kategorinya kemudian dicetak dalam format yang mudah dibaca.

```
[22] # Mengategorikan contoh teks berdasarkan strength
      strength_categories = [categorize_strength(text) for text in sample_texts]
      for text, strength in zip(sample_texts, strength_categories):
          print(f'{text}': {strength})

↳ 'Intense weightlifting session': High Strength
   'Moderate cardio workout': Moderate Strength
   'Relaxing yoga class': Low Strength
```

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Makalah ini membahas penggunaan kecerdasan buatan, terutama Natural Language Processing (NLP), dalam menganalisis kesamaan teks dalam data kesehatan menggunakan embedding dari model BERT. Dengan mengonversi deskripsi latihan menjadi representasi vektor menggunakan embedding BERT, kita bisa mengukur kesamaan kosinus antara teks untuk menemukan hubungan antara jenis latihan.

Selain itu, dalam makalah ini juga dijelaskan cara mengelompokkan tingkat intensitas latihan dengan menggunakan fungsi sederhana berdasarkan kata kunci dalam deskripsi teks. Hasilnya membantu memahami hubungan makna antara jenis latihan dan mengelompokkan latihan berdasarkan tingkat kekuatan. Ini dapat digunakan untuk membuat program kebugaran pribadi atau untuk analisis lebih lanjut di bidang kesehatan.

#### **4.2 Saran**

1. Penggunaan Dataset yang Lebih Luas dan Beragam
2. Pengembangan Model Klasifikasi Lanjutan
3. Waktu pengerjaan yang lebih lama