



Modul 4

Pemrograman Kecerdasan Artifisial

(Bimbingan Teknis Guru Koding dan Kecerdasan Artifisial Jenjang SMA/SMK)



Modul 4

Pemrograman Kecerdasan Artifisial

Pengarah:

Direktur Jenderal Guru, Tenaga Kependidikan dan Pendidikan Guru

Penanggung Jawab:

Direktur Guru Pendidikan Menengah dan Pendidikan Khusus

Koordinator:

Eneng Siti Saadah, S.Si., M.B.A

Dr. Medira Ferayanti, S.S., M.A

Penulis:

Listyanti Dewi Astuti, S.Pd., M.Kom.

Nafik Doni Agustan, S.T., S.Pd.

Pande Made Mahendri Pramadewi, S.Pd., M.Kom

Tim Ahli Materi:

Dr. Asep Wahyudin

Septiaji Eko Nugroho, S.T, M.Sc.

Dr. Asep Jihad, M.Pd.

Kontributor:

Fadlilah Prapta Widda, S.Si

Titis Sekti Wijayanti, S.Psi

Rian Muhamad Fitriyatna, S.Si

Layout/desain:

Dwi Harianti, S.A.P.

Dikeluarkan oleh:

Direktur Guru Pendidikan Menengah dan Pendidikan Khusus

Direktorat Jenderal Guru, Tenaga Kependidikan dan Pendidikan Guru

Kementerian Pendidikan Dasar dan Menengah

Kompleks Kemendikbud, Jalan Jenderal Sudirman, Senayan, Jakarta, 10270

Copyright © 2025

Hak Cipta Dilindungi Undang-Undang

Karya ini dilisensikan di bawah lisensi Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-SA 4.0)

Dilarang memperbanyak sebagian atau keseluruhan isi buku ini untuk kepentingan komersil tanpa izin tertulis dari Kementerian Pendidikan Dasar dan Menengah

Kata Pengantar

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa atas tersusunnya modul pelatihan ini sebagai bagian dari upaya peningkatan kompetensi guru dalam menyongsong implementasi mata pelajaran Koding dan Kecerdasan Artifisial (KA) di jenjang Sekolah Menengah Atas (SMA) dan Sekolah Menengah Kejuruan (SMK). Penyusunan modul ini merujuk pada Naskah Akademik Pembelajaran Koding dan Kecerdasan Artifisial yang disusun sebagai pijakan pengembangan kurikulum masa depan berbasis teknologi dan kecakapan abad ke-21.

Kebutuhan akan literasi digital, pemikiran komputasional, dan pemahaman teknologi kecerdasan artifisial menjadi semakin penting dalam mempersiapkan peserta didik menghadapi dunia yang semakin terdigitalisasi. Mata pelajaran Koding dan KA dirancang untuk memberikan fondasi yang kuat kepada siswa SMA dan SMK dalam memahami logika pemrograman, analisis data, serta penerapan AI dalam kehidupan sehari-hari dan berbagai bidang ilmu pengetahuan.

Melalui modul pelatihan ini, para guru dibekali dengan pemahaman konseptual dan keterampilan praktis dalam mengajarkan materi koding dan KA secara kontekstual, menarik, dan aplikatif. Modul ini juga memuat strategi pembelajaran yang sesuai dengan karakteristik peserta didik SMA dan SMK, serta pendekatan berbasis proyek untuk mengembangkan kreativitas dan problem solving.

Implementasi mata pelajaran ini diharapkan dapat membentuk generasi muda Indonesia yang tidak hanya menjadi pengguna teknologi, tetapi juga pencipta dan inovator dalam ekosistem digital. Guru sebagai ujung tombak pendidikan memiliki peran strategis dalam mengarahkan potensi peserta didik untuk berkembang secara optimal di era transformasi digital.

Direktorat Jenderal Guru, Tenaga Kependidikan, dan Pendidikan Guru menyampaikan apresiasi kepada seluruh pihak yang telah berkontribusi dalam penyusunan modul ini. Semoga modul ini dapat menjadi referensi yang bermanfaat dalam penyelenggaraan pelatihan dan praktik pembelajaran Koding dan Kecerdasan Artifisial di SMA dan SMK.

Akhir kata, besar harapan kami agar pelatihan ini dapat menginspirasi para guru untuk terus berinovasi dalam pembelajaran dan mendorong peserta didik menjadi insan pembelajar sepanjang hayat yang adaptif dan kompeten di masa depan.



Putra Asga Elevri, S.Si., M.Si. NIP
197801282001121004



DAFTAR ISI

Kata Pengantar	2
DAFTAR ISI	3
A. Deskripsi Umum Modul	5
A.1. Capaian Pelatihan	5
A.2. Tujuan Pelatihan	5
A.2. Indikator Capaian Pelatihan	5
A.3. Pokok bahasan	6
A.4. Alur Pelatihan	6
B. Pemrograman Kecerdasan Artifisial	9
B.1. Dasar Pemrograman Kecerdasan Artifisial	9
AKTIVITAS 1: Diskusi Tentang Konsep Dasar	31
Pemrograman KA	31
B.2. Menerapkan Pemrograman Kecerdasan Artifisial	32
B.2.1. Penerapan <i>Library</i> Kecerdasan Artifisial	86
B.2.2. Analisis Hasil dan Penyempurnaan <i>Output</i> Aplikasi Kecerdasan Artifisial	97
Lembar Kerja 4.1: Proyek Pemrograman Aplikasi KA (<i>supervised/unsupervised learning</i>) Menggunakan Dataset	102
A. Deskripsi	102
B. Tujuan	102
C. Petunjuk Kerja	102
D. Format Laporan Hasil	104
E. Kriteria Penilaian	105
C. Pengenalan <i>Large Language Model</i> pada Kecerdasan Artifisial Generatif	107
C.1. Konsep Dasar, Arsitektur, dan Cara Kerja <i>Large Language Model</i>	107
C.2. Peran <i>Large Language Model</i> dalam Kecerdasan Artifisial Generatif dan Aplikasinya	113



C.3. Teknik Integrasi Pemrograman Kecerdasan Artifisial Dengan Model <i>Large Language Model</i>	115
Lembar Kerja 4.2: Memadukan aplikasi KA dan LLM.....	125
A. Deskripsi.....	125
B. Tujuan.....	125
C. Instruksi Kerja.....	125
D. Rubrik Penilaian	127
GLOSARIUM	132
DAFTAR PUSTAKA.....	139
LAMPIRAN 1.....	140
Ragam Algoritma Machine Learning	140
LAMPIRAN 2.....	146
Kumpulan Contoh Penerapan Pemrograman KA.....	146

A. Deskripsi Umum Modul

A.1. Capaian Pelatihan

Pada akhir pelatihan, peserta mampu menjelaskan konsep dasar pemrograman KA, menerapkan library kecerdasan artifisial populer untuk menghasilkan aplikasi kecerdasan artifisial, serta menjelaskan konsep dasar, arsitektur, cara kerja, dan aplikasi *Large Language Model (LLM)*. Peserta juga mampu merefleksikan bahwa penggunaan LLM harus bertanggung jawab serta mematuhi prinsip *human-centered* dan etika yang ada.

A.2. Tujuan Pelatihan

- Peserta pelatihan mampu menjelaskan konsep dasar pemrograman KA.
- Peserta pelatihan mampu menerapkan library Kecerdasan Artifisial populer untuk menghasilkan aplikasi kecerdasan artifisial
- Peserta pelatihan menjelaskan konsep dasar, arsitektur, cara kerja, dan aplikasi Large Language Model sederhana.
- Peserta pelatihan mampu mengintegrasikan aplikasi KA dengan model bahasa besar.
- Peserta pelatihan mampu merefleksi bahwa penggunaan LLM harus bertanggung jawab dan mematuhi prinsip *human-centered* dan etika yang ada.

A.2. Indikator Capaian Pelatihan

1. Peserta mampu menjelaskan konsep dasar pemrograman KA.
2. Peserta dapat menerapkan sintaks dasar bahasa pemrograman KA untuk menulis skrip sederhana yang melibatkan logika KA
3. Peserta mampu menerapkan library KA populer untuk membangun model sederhana, seperti klasifikasi teks/gambar.
4. Peserta mampu menganalisis output KA menggunakan matrik evaluasi dan melakukan penyempurnaan kode untuk meningkatkan performa.
5. Peserta mampu menjelaskan arsitektur dasar LLM dan cara kerjanya dalam menghasilkan output.
6. Peserta dapat membedakan jenis-jenis LLM berdasarkan desain arsitektur dan kasus penggunaan spesifik.
7. Peserta dapat memadukan LLM ke dalam aplikasi menggunakan framework.
8. Peserta dapat menganalisis kelebihan dan keterbatasan LLM melalui studi kasus output yang dihasilkan

9. Peserta dapat menyusun dokumentasi teknis yang mencakup alur kerja, kode, dan analisis hasil aplikasi KA yang dikembangkan.
10. Peserta mampu merefleksikan implikasi etis dalam penggunaan LLM
11. Peserta mampu Merefleksikan tantangan dan solusi selama mengembangkan aplikasi KA dan mengintegrasikan LLM

A.3. Pokok bahasan

1. Dasar Pemrograman Kecerdasan Artifisial
2. Penerapan *library* Kecerdasan Artifisial
3. Analisis Hasil dan Penyempurnaan *Output* Aplikasi Kecerdasan Artifisial
4. Konsep Dasar, Arsitektur, dan Cara Kerja *Large Language Model*
5. Peran *Large Language Model* dalam Kecerdasan Artifisial Generatif dan Aplikasinya
6. Teknik Integrasi Pemrograman Kecerdasan Artifisial Dengan Model LLM

A.4. Alur Pelatihan

Alur pembelajaran modul Pengoperasian, Pengaplikasian, dan Kolaborasi Perangkat Kecerdasan Artifisial menggunakan *SOLO Taxonomy* yang dijelaskan pada Tabel 1.

Tabel 1. Alur Pelatihan Pemrograman Kecerdasan Artifisial

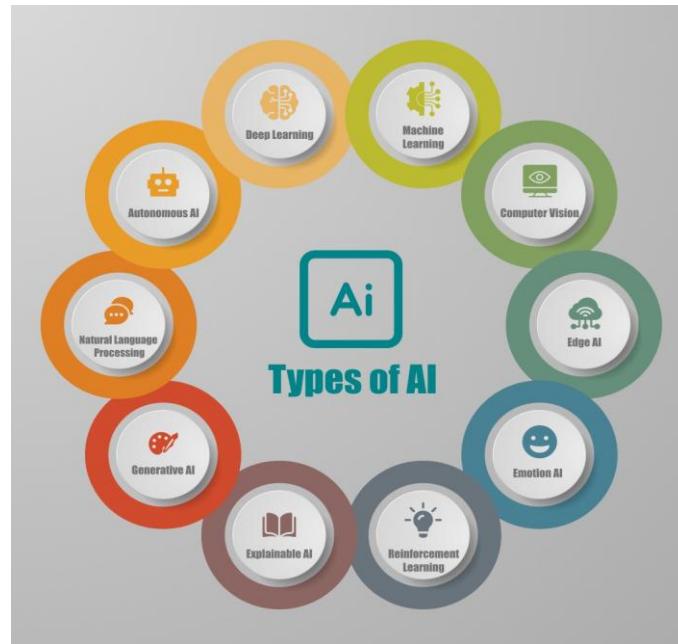
Materi	Tahapan	Aktivitas	Tagihan	JP
Pemrograman Kecerdasan Artifisial	Memahami (Unistruktural & Multistruktural)	Menjelaskan konsep dasar pemrograman KA melalui diskusi terpandu. <u>(Aktivitas 1: Diskusi Tentang Konsep Dasar Pemrograman KA)</u>	Hasil diskusi sesuai panduan diskusi pada Aktivitas 1: Diskusi Tentang Konsep Dasar Pemrograman KA (diunggah di LMS)	5
	Mengaplikasi (Relasional)	<ul style="list-style-type: none"> • Menerapkan sintaks dasar bahasa pemrograman KA untuk menulis skrip sederhana yang melibatkan logika KA • Menerapkan <i>library</i> KA populer untuk 	LK 4.1 Proyek pemrograman aplikasi KA (<i>supervised/unsupervised learning</i>) menggunakan dataset dengan tema sesuai kelompok bidang keahlian SMK	

		<p>membangun model sederhana, seperti klasifikasi teks/gambar. (Aktivitas 2: Berlatih Menerapkan Library dalam Pemrograman Kecerdasan Artifisial - Membuat Deteksi Objek Menggunakan TensorFlow)</p> <ul style="list-style-type: none"> • Menganalisis <i>output</i> KA menggunakan matriks evaluasi dan melakukan penyempurnaan kode untuk meningkatkan performa. • Menyusun dokumentasi teknis yang mencakup alur kerja, kode, dan analisis hasil aplikasi KA yang dikembangkan. 		
	Merefleksi (Abstrak meluas)	Merefleksikan tantangan dan solusi selama mengembangkan aplikasi KA	Mengisi jurnal refleksi	
Pengenalan LLM Pada KA Generatif	Memahami (Unistruktural & Multistruktural)	<ul style="list-style-type: none"> • Menjelaskan arsitektur dasar LLM dan cara kerjanya dalam menghasilkan output melalui diskusi. 	Hasil diskusi	2
	Mengaplikasi (Relasional)	<ul style="list-style-type: none"> • Membedakan jenis-jenis LLM berdasarkan desain arsitektur dan kasus penggunaan spesifik melalui studi kasus • Memadukan LLM ke dalam aplikasi menggunakan framework. (Aktivitas 3: Mengintegrasikan LLM 	LK 4.2. Memadukan aplikasi KA dan LLM Menggunakan API	

	<ul style="list-style-type: none"> ke Aplikasi KA Menggunakan Hugging Face - Pirate Chatbot) • Menganalisis kelebihan dan keterbatasan <i>LLM</i> melalui studi kasus <i>output</i> yang dihasilkan • Menyusun dokumentasi teknis yang mencakup alur kerja, kode, dan analisis hasil aplikasi KA yang dikembangkan. 		
Merefleksi (Abstrak meluas)	<ul style="list-style-type: none"> • Merefleksikan implikasi etis dalam penggunaan <i>LLM</i> • Merefleksikan tantangan dan solusi selama mengembangkan aplikasi KA dan mengintegrasikan <i>LLM</i> 	Mengisi jurnal refleksi	

B. Pemrograman Kecerdasan Artifisial

B.1. Dasar Pemrograman Kecerdasan Artifisial



Gambar 1. Tipe-tipe Kecerdasan Artifisial

Sumber: <https://medium.com/@seaflux/exploring-the-transformative-power-of-different-ai-types-c6453b895a2c>

Kecerdasan Artifisial (KA) telah mengalami perkembangan pesat dengan hadirnya berbagai pendekatan yang saling melengkapi. Meskipun banyak yang mengasosiasikan KA secara eksklusif dengan machine learning (ML) karena kemampuannya untuk belajar dari data, KA sebenarnya mencakup spektrum yang jauh lebih luas. Selain ML, terdapat sistem berbasis aturan (*rule-based systems*) yang mengandalkan logika dan aturan eksplisit untuk pengambilan keputusan, serta kecerdasan simbolik yang menekankan representasi pengetahuan melalui simbol dan logika formal. Pendekatan kecerdasan ini membuktikan bahwa KA tidak hanya bergantung pada algoritma pembelajaran, melainkan juga pada metode-metode lain yang mengedepankan struktur dan aturan dalam pengolahan informasi.

Dalam ekosistem KA, *machine learning* dan *deep learning* merupakan komponen yang memainkan peranan penting, terutama dalam pengenalan pola melalui teknologi seperti *Natural Language Processing (NLP)* dan *Computer Vision*. Teknologi-teknologi tersebut telah mengoptimalkan aplikasi seperti *chatbot*, terjemahan bahasa, pengenalan wajah, serta sistem kendaraan otonom. Selain itu, KA juga mencakup kecerdasan generatif yang mampu

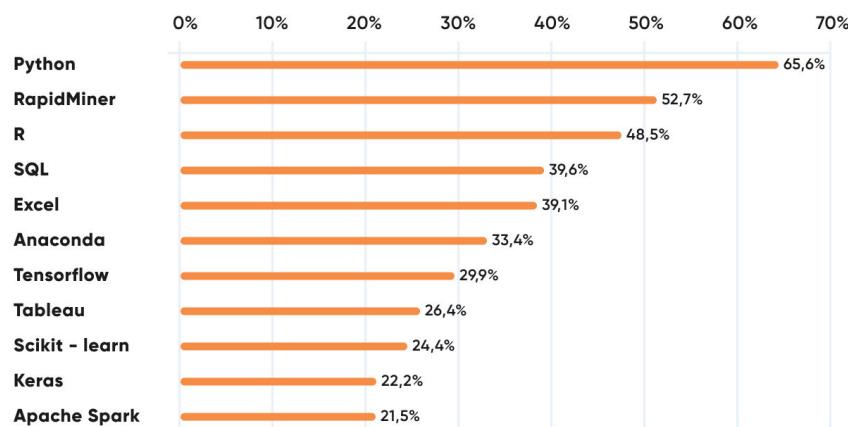
menghasilkan konten baru—mulai dari teks, gambar, hingga suara—serta pendekatan inovatif lainnya seperti Edge AI untuk pemrosesan data secara lokal, Emotion AI yang menghadirkan interaksi digital dengan sentuhan empati, dan *reinforcement learning* yang memanfaatkan pengalaman untuk mengasah pengambilan keputusan dinamis.

Machine Learning (ML) memungkinkan mesin belajar dari data dan menjadi fondasi banyak aplikasi modern. **Deep Learning**, sebagai bagian dari ML, unggul dalam pengenalan pola seperti suara, gambar, dan bahasa. **Natural Language Processing (NLP)** mendukung komunikasi antara manusia dan mesin melalui chatbot dan terjemahan bahasa. **Computer Vision** memungkinkan interpretasi data visual, digunakan dalam pengenalan wajah dan kendaraan otonom. **Generative AI** menciptakan konten baru seperti teks dan gambar, mendorong inovasi di bidang kreatif. **Reinforcement Learning** mengajarkan mesin untuk belajar dari pengalaman, cocok untuk robotika dan pengambilan keputusan dinamis. **Explainable AI (XAI)** penting untuk transparansi dan akuntabilitas dalam penggunaan AI, dan **Autonomous AI** memungkinkan sistem bekerja mandiri tanpa campur tangan manusia.

B.1.1 Mengenal Bahasa Pemrograman Kecerdasan Artifisial

Pemilihan bahasa pemrograman yang tepat dan terbaik untuk pengembangan aplikasi KA sangat bergantung pada **kebutuhan proyek, keterampilan yang dimiliki pengembang, serta tujuan individu atau organisasi pengembang**. Setiap bahasa pemrograman memiliki keunggulan yang unik kesesuaian untuk tugas-tugas tertentu.

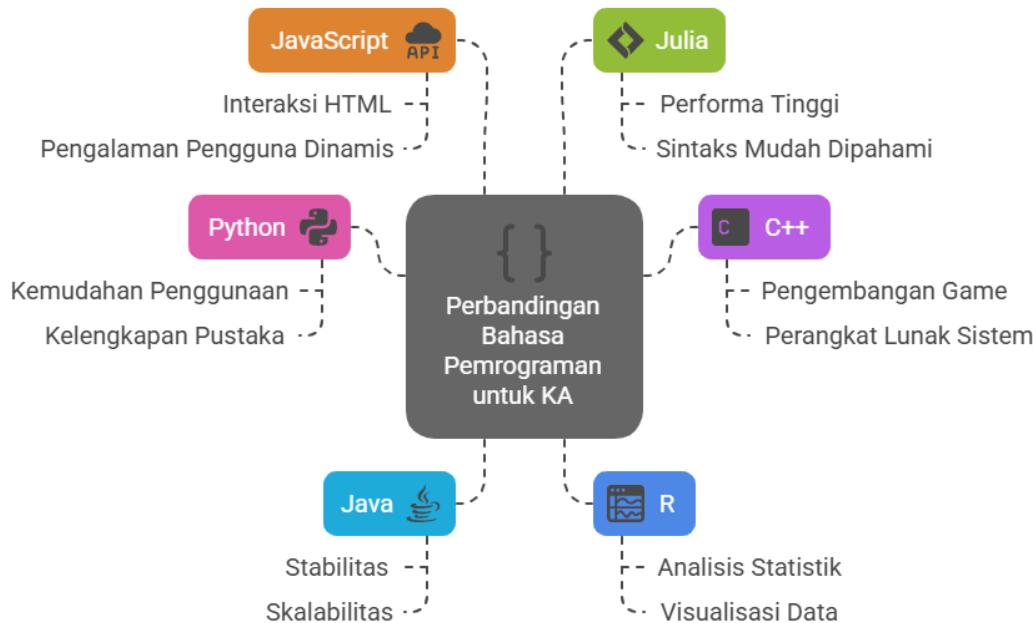
The most popular programming languages for AI/ML development



Gambar 2. Bahasa Pemrograman KA Terpopuler. Sumber: <https://www.softformance.com/>

Tabel 2. Beberapa Kelebihan Bahasa Pemrograman Kecerdasan Artifisial

Bahasa Pemrograman	Kelebihan
Python	Pilihan utama bagi pemula karena kemudahan penggunaan, sintaks yang sederhana, kelengkapan pustaka, dan dukungan komunitas yang luas.
C++	Unggul dalam aplikasi yang memiliki kebutuhan performa tinggi, sehingga sangat cocok untuk aplikasi yang memerlukan efisiensi dan kecepatan.
Java	Banyak digunakan di lingkungan perusahaan karena stabilitas dan skalabilitasnya serta memungkinkan pengembang untuk membuat aplikasi yang dapat berjalan di berbagai platform.
R	Efektif untuk analisis statistik yang mendalam dan visualisasi data.
JavaScript	Populer digunakan dalam integrasi KA pada aplikasi web yang dinamis dan responsif.
Julia	Bahasa baru yang sering digunakan dalam bidang komputasi numerik dan ilmiah.

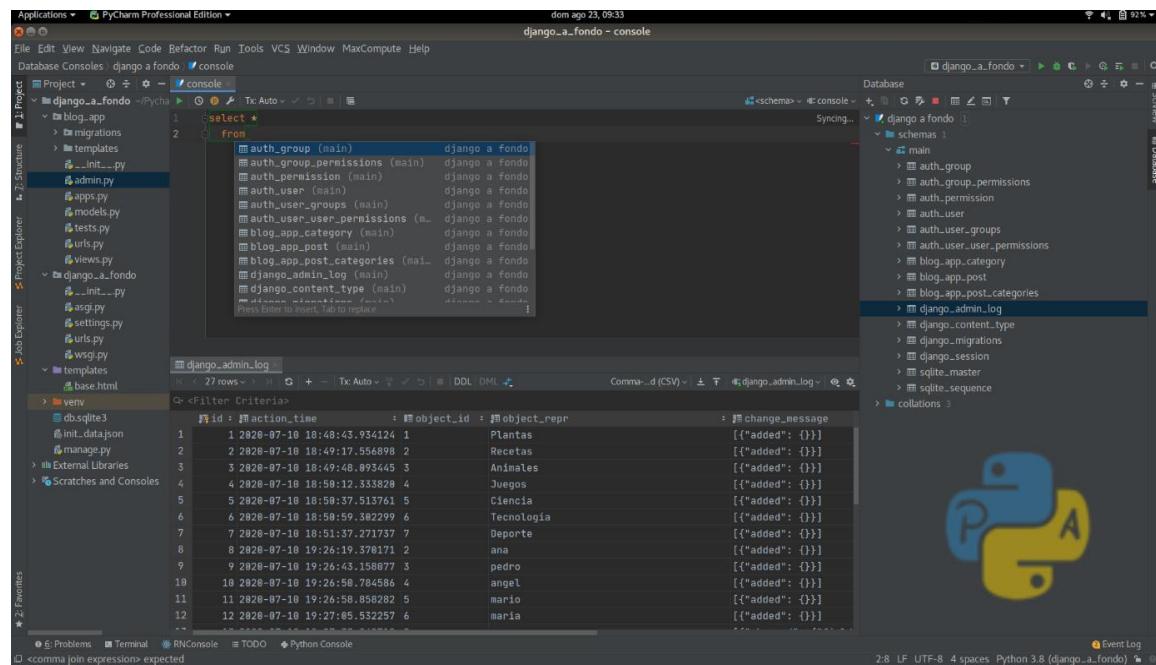


Gambar 3. Perbandingan Bahasa Pemrograman untuk KA

B.1.2 Mengenal *Integrated Development Environment (IDE)* Untuk Mengembangkan Aplikasi Kecerdasan Artifisial

Setelah memilih bahasa pemrograman yang tepat, penggunaan *Integrated Development Environment (IDE)* yang sesuai juga berperan penting dalam pengembangan aplikasi KA. Beberapa IDE yang cukup populer untuk pengembangan aplikasi KA, meliputi:

- PyCharm, populer untuk penggunaan bahasa pemrograman Python dengan integrasi pustaka KA dan fitur debugging yang kuat (<https://www.jetbrains.com/pycharm/download/>)



Gambar 4. PyCharm IDE

- Jupyter Notebook, merupakan *IDE* yang ideal untuk eksperimen dan visualisasi data dengan eksekusi kode berbasis sel (<https://jupyter.org/install>)




A screenshot of a Jupyter Notebook interface. The title bar shows 'full-stack-python-data.ipynb' and the URL 'localhost:8888/notebooks/full-stack-python-data.ipynb'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a status bar indicating 'Not Trusted' and 'Python 3'. The main area contains code cells:

```

In [3]: import os
import pandas as pd
df = pd.read_csv('fsp-search-terms.csv')

In [4]: len(df)
Out[4]: 999

In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 5 columns):
Queries      999 non-null object
Clicks        999 non-null int64
Impressions   999 non-null int64
CTR          999 non-null object
Position     999 non-null float64
dtypes: float64(1), int64(2), object(2)

```

The notebook is running on Python 3. Below the code, there is a terminal log:

```

[I 06:57:02.689 NotebookApp] Writing notebook-signing key to /Users/matt/Library/Jupyter/notebook_secret
[W 06:57:02.690 NotebookApp] Notebook full-stack-python-data.ipynb is not trusted
[I 06:57:02.922 NotebookApp] Kernel started: dbe3576e-7faf-4b1b-aef-fff35e83988f
[I 06:57:03.488 NotebookApp] Adapting to protocol v5.1 for kernel dbe3576e-7faf-4b1b-aef-fff35e83988f
[I 06:59:03.853 NotebookApp] Saving file to /full-stack-python-data.ipynb
[W 06:59:03.854 NotebookApp] Notebook full-stack-python-data.ipynb is not trusted
[I 07:19:20.594 NotebookApp] Saving file to /full-stack-python-data.ipynb
[W 07:19:20.595 NotebookApp] Notebook full-stack-python-data.ipynb is not trusted

```

Gambar 5. Jupyter Notebook

- Google Colaboratory, memiliki kemiripan antarmuka dan penggunaan dengan Jupyter Notebook (<https://colab.research.google.com/>).




A screenshot of Google Colaboratory. The title bar shows 'Object Detection TensorFlow.ipynb'. The code cell contains:

```

[ ] !pip install pillow
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)

# Import library yang diperlukan
import tensorflow as tf           # untuk model dan prediksi
import numpy as np                # untuk operasi numerik
from PIL import Image, ImageOps    # untuk memproses gambar
import matplotlib.pyplot as plt   # untuk menampilkan gambar
from google.colab import files    # untuk mengunggah file di Google colab

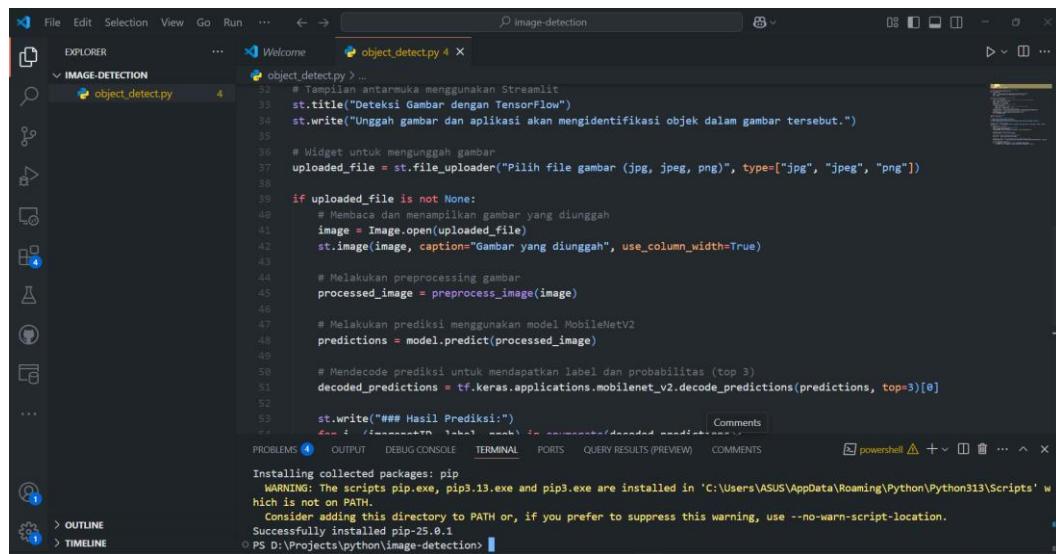
# Fungsi untuk memuat model MobileNetV2 dengan bobot pre-trained dari ImageNet
def load_model():
    model = tf.keras.applications.MobileNetV2(weights='imagenet')
    return model

# Fungsi untuk melakukan preprocessing gambar
def preprocess_image(image):
    # Ukuran gambar menjadi 224x224 piksel sesuai dengan input model MobileNetV2
    image = image.resize((224, 224))
    # Ubah gambar menjadi array numpy
    image_array = np.array(image)
    # Pastikan gambar memiliki 3 channel (RGB); jika ada channel alpha (RGBA), buang channel tersebut
    if image_array.shape[-1] == 4:
        image_array = image_array[:, :, :-1]
    # Lakukan preprocessing sesuai dengan MobileNetV2
    processed_image = tf.keras.applications.mobilenet_v2.preprocess_input(image_array)

```

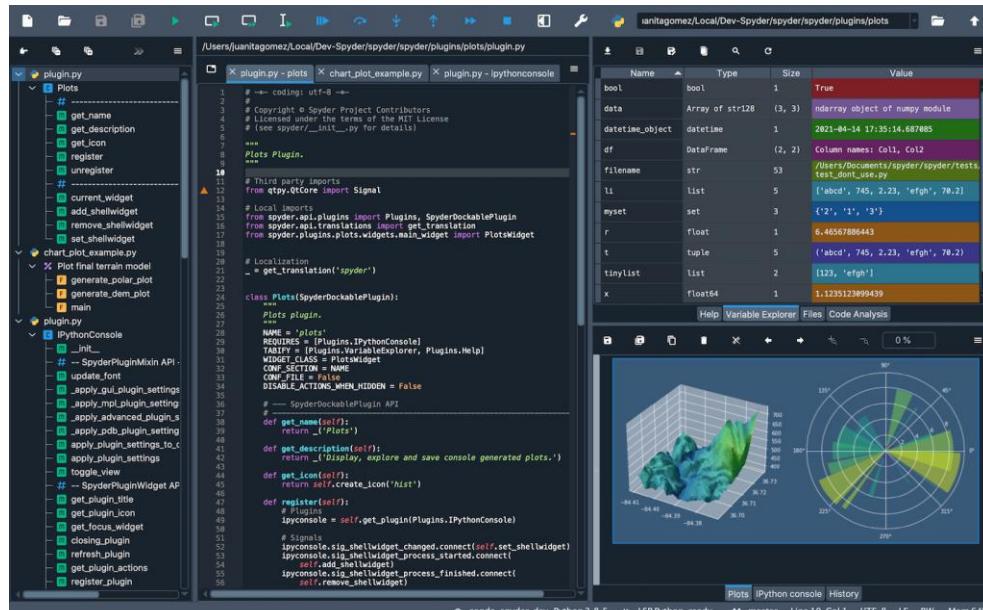
Gambar 6. Google Colaboratory

- Visual Studio Code, merupakan *IDE* yang cukup fleksibel, yang dapat dikustomisasi dengan berbagai ekstensi, termasuk ekstensi yang mendukung pengembangan aplikasi KA (<https://code.visualstudio.com/Download>)



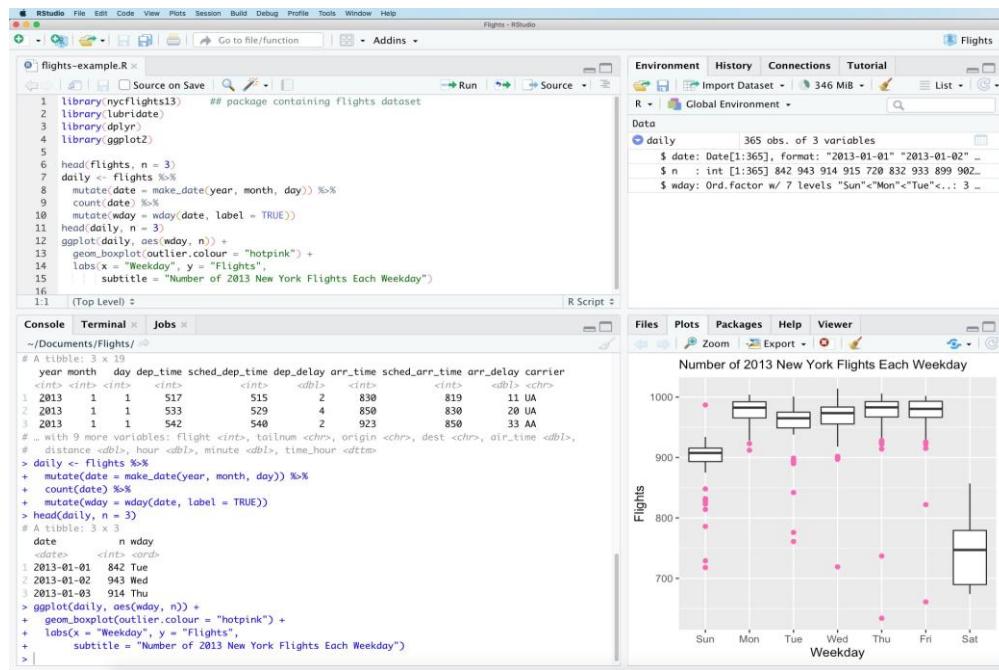
Gambar 7. Visual Studio Code

- Spyder, sesuai untuk bahasa pemrograman Python dan analisis data (<https://www.spyder-ide.org/>)



Gambar 8. Spyder

- RStudio, sesuai untuk analisis data dengan bahasa pemrograman R (<https://posit.co/download/rstudio-desktop/>)



Gambar 9. R Studio

B.1.3 Dasar-Dasar Pemrograman Dengan Bahasa Python

Python merupakan salah satu bahasa pemrograman yang sangat populer di kalangan *data scientist* dan praktisi *machine learning*. Kemudahan sintaks, ketersediaan library pendukung, dan komunitas yang besar menjadikan Python menjadi pilihan yang populer untuk melakukan analisis data, eksplorasi, dan pembangunan model prediktif.

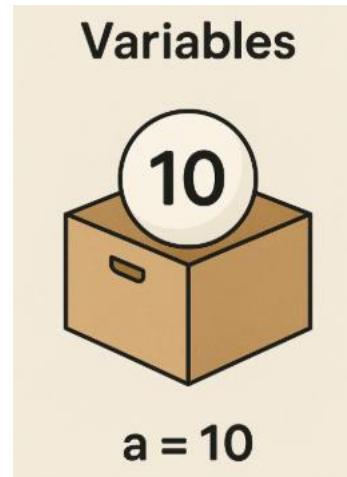
Python cocok untuk memulai belajar machine learning karena :

- Sintaks sederhana dan mirip bahasa manusia** – kode yang ditulis dalam bahasa Python mudah dibaca, sehingga pemula bisa fokus memahami konsep dan penerapan ML tanpa terbebani oleh detail bahasa pemrograman.
- Ekosistem pustaka yang cukup lengkap** – *library* seperti NumPy, pandas, scikit-learn, TensorFlow, dan PyTorch menyediakan fungsi siap pakai untuk pra-pemrosesan data hingga training model, sehingga Anda dapat langsung bereksperimen.
- Komunitas dan sumber belajar yang sangat besar** – tersedia banyak dokumentasi, tutorial, serta forum Python yang memudahkan pemecahan masalah dan pembaruan pengetahuan.

1) Variabel dan Tipe Data

Dalam bahasa Python, variabel digunakan untuk menyimpan data. Tipe data yang umum digunakan adalah:

- *Integer* (bilangan bulat),
- *Float* (bilangan desimal),
- *String* (teks), dan
- *Boolean* (nilai benar atau salah).



Gambar 10. Variabel

Contoh kode berikut menunjukkan cara mendefinisikan dan mencetak variabel:

```
# Variabel dan tipe data
a = 10          # integer
b = 3.14        # float
c = "Hello, Data Science!" # string
d = True         # boolean

print("Nilai a:", a)
print("Nilai b:", b)
print("Nilai c:", c)
print("Nilai d:", d)
```

Nilai a: 10
Nilai b: 3.14
Nilai c: Hello, Data Science!
Nilai d: True

Penjelasan:

- Variabel a menyimpan nilai 10, sedangkan b menyimpan nilai 3.14.

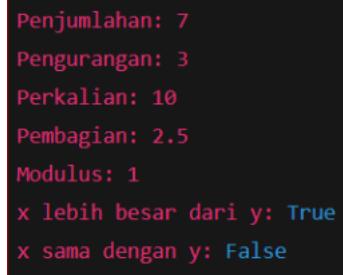
- Variabel c adalah string yang berisi pesan, dan variabel d memiliki tipe data boolean dengan nilai True.

2) Operator Aritmatika dan Logika

Operator aritmatika digunakan untuk melakukan perhitungan dasar, sedangkan operator logika membantu dalam pengambilan keputusan logis di dalam program.

```
# Operator Aritmatika
x = 5
y = 2
print("Penjumlahan:", x + y)      # Output: 7
print("Pengurangan:", x - y)      # Output: 3
print("Perkalian:", x * y)         # Output: 10
print("Pembagian:", x / y)         # Output: 2.5
print("Modulus:", x % y)           # Output: 1

# Operator Logika dan Perbandingan
print("x lebih besar dari y:", x > y)  # True
print("x sama dengan y:", x == y)       # False
```



```
Penjumlahan: 7
Pengurangan: 3
Perkalian: 10
Pembagian: 2.5
Modulus: 1
x lebih besar dari y: True
x sama dengan y: False
```

Penjelasan:

- Operator aritmatika seperti +, -, *, /, dan % digunakan untuk operasi matematika.
- Operator perbandingan seperti > dan == digunakan untuk mengevaluasi kondisi dan menghasilkan nilai boolean.

3) Struktur Kontrol

Struktur kontrol memungkinkan program untuk mengambil jalur eksekusi yang berbeda berdasarkan suatu kondisi tertentu. Contoh paling umum adalah penggunaan if, elif, dan else.

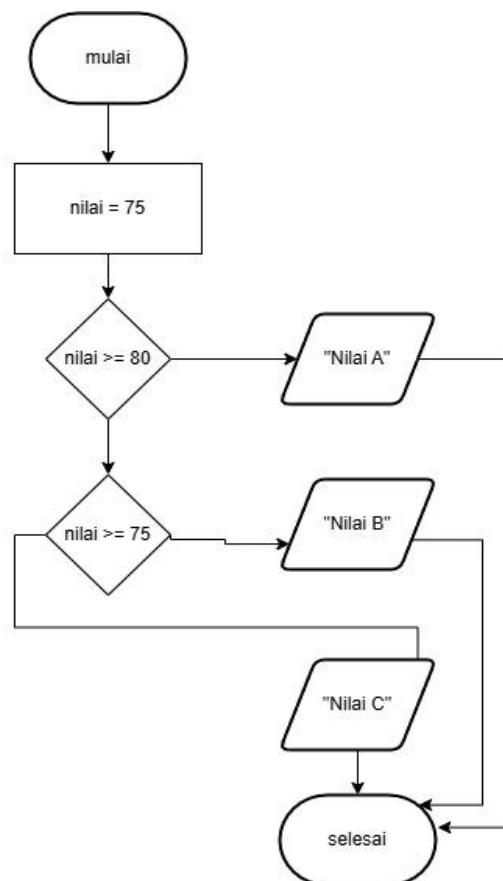
```
# Struktur Kontrol
nilai = 75

if nilai >= 80:
    print("Nilai A")
elif nilai >= 70:
    print("Nilai B")
else:
    print("Nilai C")
```

Nilai B

Penjelasan:

- Program mengevaluasi nilai variabel `nilai` dan mencetak "Nilai B" karena nilai 75 memenuhi kondisi `nilai >= 70` namun tidak mencapai `nilai >= 80`.



Gambar 11. Struktur Kontrol Percabangan

Looping atau iterasi digunakan untuk menjalankan suatu blok kode program secara berulang. Python mendukung perulangan menggunakan for dan while.

```
# Looping dengan for
for i in range(5):
    print("Iterasi ke-", i)

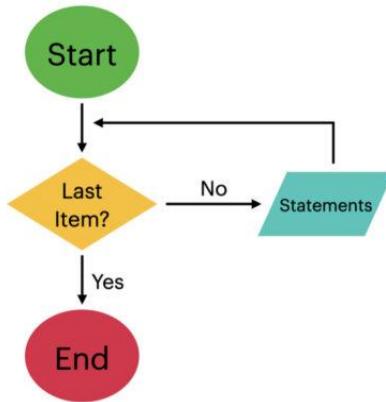
# Looping dengan while
j = 0
while j < 5:
    print("while iterasi ke-", j)
    j += 1
```

```
Iterasi ke- 0
Iterasi ke- 1
Iterasi ke- 2
Iterasi ke- 3
Iterasi ke- 4
while iterasi ke- 0
while iterasi ke- 1
while iterasi ke- 2
while iterasi ke- 3
while iterasi ke- 4
```

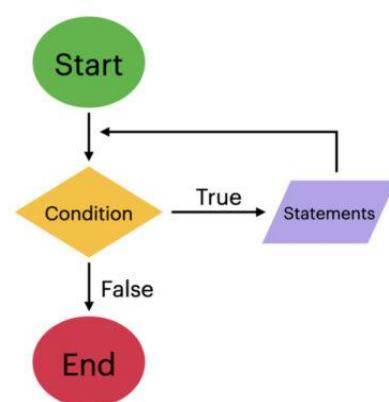
Penjelasan:

- `for` loop di atas akan mencetak nilai dari 0 hingga 4.
- `while` loop akan terus berjalan selama kondisi `j < 5` terpenuhi

For Loop



While Loop



Gambar 12. Perbandingan For Loop dan While Loop.

Sumber: <https://www.codingem.com/flowchart-loop/>

4) Fungsi

adalah blok kode yang didefinisikan untuk melakukan tugas tertentu, sehingga kode bersifat modular dan dapat digunakan ulang (*reuseable*) dengan mudah. Fungsi diberi nama, dirancang untuk melakukan satu tugas tertentu, dan dapat dipanggil berulang kali. Di dalam fungsi kita biasanya menulis instruksi yang ingin dijalankan, kemudian menggunakan `return` untuk mengembalikan nilai hasil.

```
# 1. Fungsi tanpa parameter
def salam():
    """
    Mengembalikan sebuah pesan sapaan standar.
    """

    return "Halo, selamat datang!"

# 2. Fungsi dengan parameter
def sapa(nama):
    """
    Mengembalikan pesan sapaan yang mencantumkan nama pengguna.

    Parameter:
        nama (str): Nama pengguna yang ingin disapa.
    """

    return "Halo, " + nama + "!"

# 3. Pemanggilan fungsi
# Memanggil fungsi tanpa parameter
print(salam())

# Memanggil fungsi dengan parameter
hasil = sapa("Data Scientist")
print(hasil)
```

Halo, selamat datang!

Halo, Data Scientist!

Penjelasan:

Fungsi tanpa parameter

- Contoh: `def salam():`
- Tidak menerima masukan (input) apa pun.
- Selalu mengembalikan pesan yang sama setiap kali dipanggil, yaitu "`Halo, selamat datang!`".
- Pemanggilan: cukup tulis `salam()` tanpa argumen.

Fungsi dengan parameter

- Contoh: `def sapa(nama):`
- Menerima satu masukan (`nama`) yang digunakan di dalam tubuh fungsi.
- Mengembalikan string yang menggabungkan teks tetap dengan nilai parameter, misalnya "`Halo, Data Scientist!`".
- Pemanggilan: tulis `sapa("NamaAnda")`, di mana "`NamaAnda`" adalah argumen yang dikirim ke fungsi.

Pemanggilan dan penggunaan nilai `return`

- Saat fungsi dijalankan, Python akan mengeksekusi baris-baris di dalamnya dan mengirimkan kembali (return) sebuah nilai.
- Nilai itu bisa langsung dicetak (`print(salam())`) atau disimpan dalam variabel untuk digunakan selanjutnya (`hasil = sapa("Data Scientist")`).

B.1.4. Bekerja Dengan Struktur Data

Struktur data merupakan **cara untuk menyimpan dan mengelola data**. Python menyediakan berbagai jenis struktur data yang masing-masing memiliki keunggulan tersendiri untuk keperluan analisis dan pemrosesan data. Aplikasi KA melakukan pengelolaan data yang sangat banyak, sehingga pemahaman terhadap penggunaan struktur data menjadi sangat penting dalam pemrograman KA.

1) List

List atau daftar adalah struktur data yang **terurut dan bersifat mutable (dapat diubah)**. List sering digunakan untuk menyimpan sekumpulan data yang saling terkait.

```
# Contoh List
buah = ["apel", "pisang", "jeruk"]
print("Buah:", buah)

# Menambahkan elemen ke dalam list
buah.append("mangga")
print("Setelah ditambahkan:", buah)

# Menghapus elemen dari list
buah.remove("pisang")
print("Setelah dihapus:", buah)
```

```
Buah: ['apel', 'pisang', 'jeruk']
Setelah ditambahkan: ['apel', 'pisang', 'jeruk', 'mangga']
Setelah dihapus: ['apel', 'jeruk', 'mangga']
```

Penjelasan:

- List `buah` menyimpan nama-nama buah.
- Metode `.append()` menambahkan elemen baru ke dalam *List*, sedangkan `.remove()` menghapus elemen tertentu dari dalam *List*.

2) Tuple

Tuple hampir mirip dengan *list*, namun **bersifat immutable (tidak dapat diubah setelah didefinisikan)**. *Tuple* digunakan ketika data yang disimpan tidak perlu dimodifikasi. *Tuple* sangat berguna untuk menyimpan data konstan yang tidak berubah, sehingga dapat meningkatkan keamanan data.

```
# Contoh Tuple
angka = (1, 2, 3, 4)
print("Angka:", angka)
# Mengubah nilai pada tuple akan menghasilkan error, contohnya:
# angka[0] = 10 # Akan menimbulkan error
```

Ketika dijalankan, kode ini akan menampilkan:

Angka: (1, 2, 3, 4)

Jika Anda mencoba membuka komentar dan menjalankan baris

```
angka[0] = 10
```

maka Python akan melempar error semacam ini:

```
TypeError: 'tuple' object does not support item assignment
```

3) Dictionary

Dictionary adalah struktur data yang menyimpan **data dalam bentuk pasangan kunci (key) dan nilai (value)**, atau **biasa dikenal dengan istilah key-value pair**. Struktur data ini sangat berguna untuk merepresentasikan data yang memiliki hubungan yang jelas. Dictionary memungkinkan akses data yang cepat melalui key dan data dalam dictionary dapat diubah dengan mudah

```
# Contoh Dictionary
mahasiswa = {"nama": "Budi", "umur": 21, "jurusan": "Teknik Informatika"}
print("Mahasiswa:", mahasiswa)

# Mengakses nilai menggunakan key
print("Nama:", mahasiswa["nama"])

# Memperbarui nilai dalam dictionary
mahasiswa["umur"] = 22
print("Setelah update umur:", mahasiswa)
```

```
Mahasiswa: {'nama': 'Budi', 'umur': 21, 'jurusan': 'Teknik Informatika'}
Nama: Budi
Setelah update umur: {'nama': 'Budi', 'umur': 22, 'jurusan': 'Teknik Informatika'}
```

4) Set

Set adalah koleksi data yang **tidak terurut dan tidak mengizinkan adanya duplikasi**. Struktur data ini berguna ketika pemrogram ingin memastikan bahwa tidak ada elemen yang sama dalam himpunan datanya. Set secara otomatis menghilangkan nilai

duplikat. Karena sifatnya yang tidak terurut, maka pemrogram tidak dapat mengakses elemen berdasarkan indeks.

```
# Contoh Set  
angka_set = {1, 2, 3, 3, 4, 5}  
print("Angka Set:", angka_set)  
  
# Menambahkan elemen ke dalam set  
angka_set.add(6)  
print("Setelah ditambahkan 6:", angka_set)
```

```
Angka Set: {1, 2, 3, 4, 5}  
Setelah ditambahkan 6: {1, 2, 3, 4, 5, 6}
```

5) Pandas Series dan DataFrame

Pandas adalah *library* yang sangat populer di dunia *data science*. Salah satu struktur data yang disediakan oleh Pandas adalah *Series*, yaitu **array atau himpunan satu dimensi yang dilengkapi dengan label pada tiap elemennya**. *Series* memudahkan manipulasi data karena pemrogram dapat mengakses elemen berdasarkan indeks label.

a). Membuat dan menampilkan *series*

```
import pandas as pd  
  
# Membuat pandas Series  
data = [10, 20, 30, 40]  
seri = pd.Series(data, index=["a", "b", "c", "d"])  
print("Pandas Series:")  
print(seri)
```

```
Pandas Series:  
a    10  
b    20  
c    30  
d    40  
dtype: int64
```

Penjelasan:

- Contoh di atas membuat *Series* dari list *data* dan menetapkan indeks khusus menggunakan parameter *index*.
- *Series* ini akan menampilkan setiap nilai beserta indeksnya, sehingga memudahkan referensi **saat analisis data**.

b). Operasi dasar pada series

Series mendukung berbagai operasi matematika dan statistik yang berguna untuk analisis data.

```
# Operasi dasar pada Series
print("Mean:", seri.mean())
print("Sum:", seri.sum())
print("Max:", seri.max())
```

Mean: 25.0

Sum: 100

Max: 40

Penjelasan:

- Metode `mean()`, `sum()`, dan `max()` memungkinkan pemrogram untuk dengan mudah menghitung nilai rata-rata, total, dan nilai maksimum dari kumpulan data dalam Series.

Sedangkan `pandas.DataFrame` adalah struktur data 2-dimensi (baris x kolom) yang berlabel. DataFrame mirip lembar kerja *spreadsheet* atau tabel, di mana setiap kolom dapat menyimpan data dengan tipe yang berbeda dan memiliki nama kolom serta indeks.

Contoh DataFrame:

```
import pandas as pd

# buat DataFrame dari kamus Python
df = pd.DataFrame({
    "Nama": ["Ani", "Budi", "Cici"],
    "Usia": [17, 18, 17],
    "Nilai": [88, 92, 79]
})

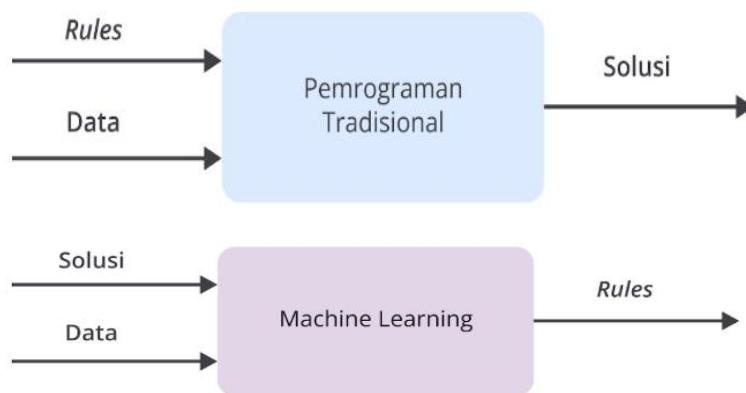
print(df)
```

Hasil dari potongan kode program di atas adalah:

	Nama	Usia	Nilai
0	Ani	17	88
1	Budi	18	92
2	Cici	17	79

B.1.5. Mengenal Supervised Learning, Unsupervised Learning, Semi Supervised Learning, dan Reinforcement Learning

Machine learning (ML), menurut Google Cloud (<https://cloud.google.com/learn/what-is-machine-learning?hl=id>) adalah bagian dari KA yang memungkinkan suatu sistem untuk belajar dan berkembang secara mandiri menggunakan jaringan neural dan deep learning, tanpa perlu adanya pemrograman secara eksplisit, dengan cara memasok data dalam jumlah besar ke dalam sistem tersebut.

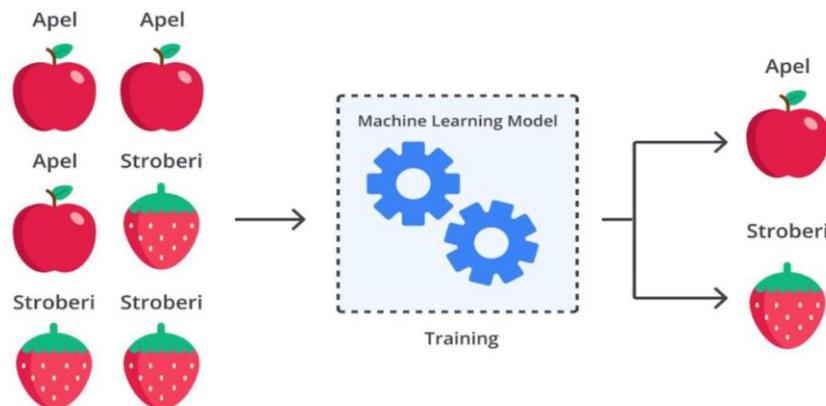


Gambar 13. Perbandingan Pemrograman Tradisional dan ML

Sumber: <https://www.dicoding.com/>

ML memiliki dua pendekatan utama, yaitu pembelajaran terawasi (*supervised learning*) dan pembelajaran tanpa pengawasan (*unsupervised learning*). Perbedaan utamanya adalah penggunaan data berlabel pada pembelajaran terawasi, sedangkan pembelajaran tanpa pengawasan bekerja dengan data yang tidak berlabel.

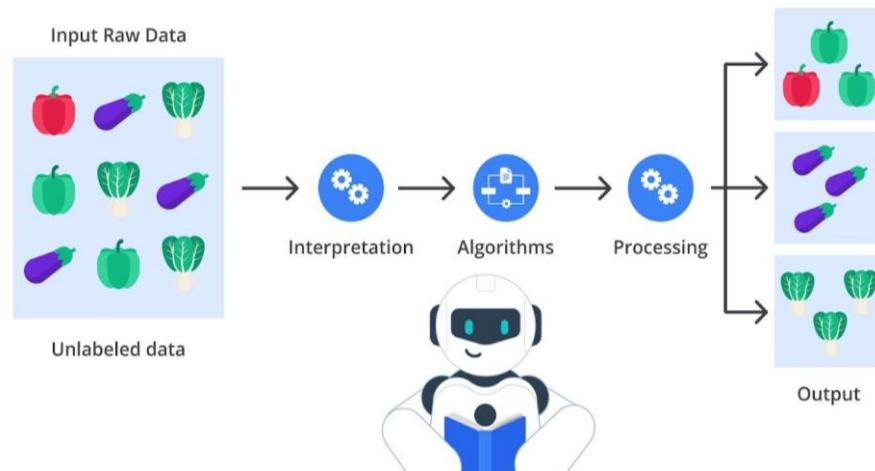
- Pembelajaran Terawasi (*Supervised Learning*) melibatkan **penggunaan dataset berlabel (input dan output yang diketahui)** untuk melatih algoritma dalam mengklasifikasikan data atau memprediksi hasil dengan akurat. Secara sederhana, untuk melatih algoritma untuk mengenali gambar apel, maka sistem diberi data gambar yang diberi label sebagai apel. Contohnya adalah pengelompokan (*classification*), pohon keputusan (*decision tree*), *naive bayes*, dan regresi (*regression*).



Gambar 14. Ilustrasi Supervised Learning

Sumber: <https://www.dicoding.com/>

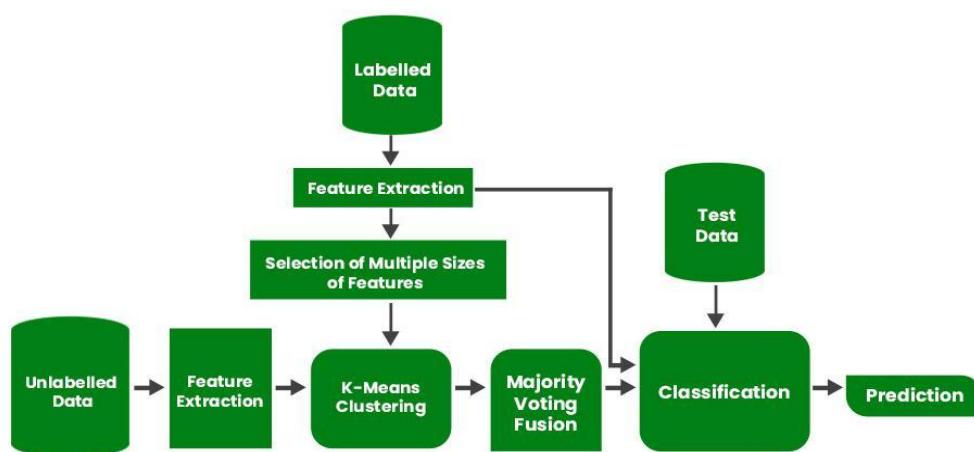
- Pembelajaran Tidak Terawasi (*Unsupervised learning*) adalah model *machine learning* yang menggunakan data tidak berlabel (data tidak terstruktur) dalam melakukan pembelajaran. Algoritma pada pembelajaran tidak terawasi belajar dari kumpulan data yang tidak diinputkan secara manual oleh manusia (tidak terawasi) dan mengkategorikannya ke dalam berbagai kelompok berdasarkan atributnya. Misalnya, jika algoritma diberi input data berupa gambar apel dan pisang, maka algoritma tersebut akan bekerja dengan sendirinya untuk mengkategorikan gambar mana yang merupakan apel dan pisang. *Unsupervised learning* biasa dan cukup bagus digunakan dalam pemodelan deskriptif dan pencocokan pola, seperti *fuzzy means* dan *k-means clustering*,



Gambar 15. Ilustrasi Unsupervised Learning

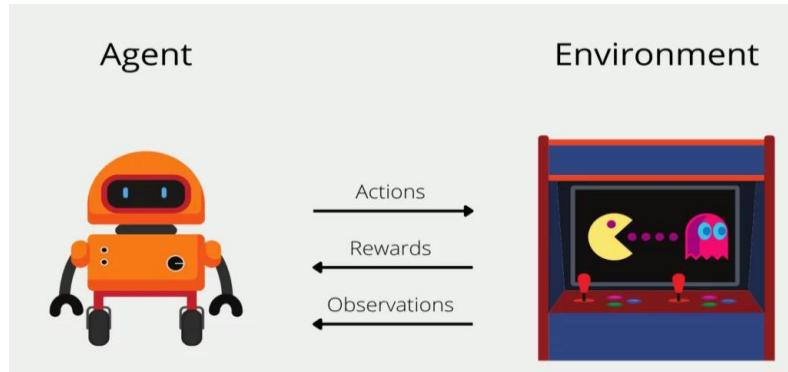
Sumber: <https://www.dicoding.com/>

- Pembelajaran Semi-Terawasi (*Semi Supervised Learning*) adalah pendekatan pembelajaran mesin yang **menggabungkan keunggulan dari pembelajaran terawasi dan tak terawasi**. Dalam metode ini, algoritma dilatih menggunakan sejumlah kecil data berlabel dan sejumlah besar data tidak berlabel, dengan tujuan meningkatkan akurasi model tanpa harus memberikan label pada seluruh dataset. Metode ini efektif ketika data tak berlabel mudah didapat, tetapi pelabelannya mahal atau sulit dilakukan. Pendekatan ini banyak digunakan dalam klasifikasi teks, gambar, dan analisis suara berskala besar seperti klasifikasi konten internet, analisis genom, atau pelabelan rekaman suara.



Gambar 16. Semi Supervised Learning. Sumber: <https://www.geeksforgeeks.org/ml-semi-supervised-learning/>

- *Reinforcement learning* (pembelajaran penguatan) adalah model pembelajaran mesin yang menggunakan metode “belajar melalui praktik”. Pembelajaran pada model ini dilakukan dengan **menggunakan serangkaian uji coba oleh “agen” yang melakukan perbaikan terus menerus berdasarkan feedback loop yang diterima, hingga kinerjanya sesuai dengan harapan**. Feedback positif diterima oleh agen jika kinerjanya baik, dan sebaliknya. Model *reinforcement learning* banyak digunakan dalam permainan, seperti catur dan maze. Google mengembangkan algoritma *reinforcement learning* untuk memainkan gim Go, di mana algoritma tersebut tidak memiliki pengetahuan awal tentang aturan dalam permainan Go. Algoritma yang dikembangkan, pada awalnya, memindahkan biji pada papan permainan secara acak, dan mempelajari langkah atau strategi terbaik untuk memenangkan permainan melalui penguatan positif dan negatif, hingga mencapai suatu kondisi di mana model dapat mengalahkan manusia dalam gim tersebut.



Gambar 17. Reinforcement Learning. Sumber:
<https://databasecamp.de/en/ml/reinforcement-learnings>

Pemilihan antara model pembelajaran terawasi, tidak terawasi, atau penguatan sangat tergantung pada ketersediaan data, kondisi data, dan tujuan yang ingin dicapai.

B.1.6. Mengenal *Training* (Latih) dan *Testing* (Uji)

Dalam ML, *training* dan *testing* merupakan komponen vital yang membantu algoritma untuk belajar berdasarkan data yang diberikan.

- Data *training* atau data latih adalah sekumpulan **data yang digunakan untuk melatih model machine learning**. Di sini, model belajar dari fitur (*variabel input*) dan label (*output* yang diharapkan) untuk mengenali pola atau hubungan yang mendasari data. Kualitas dan kuantitas data training sangat mempengaruhi kemampuan model untuk belajar secara akurat. Data latih yang digunakan harus “bersih” dan representatif, sehingga dapat membantu model dalam memahami variasi serta kompleksitas data.
- Data *testing* atau data uji adalah **kumpulan data yang belum pernah digunakan selama proses pelatihan**. Data ini berfungsi untuk mengevaluasi kinerja model pembelajaran mesin setelah proses *training* selesai dilakukan. Data *testing* digunakan untuk **mengukur seberapa baik model dapat menggeneralisasi pengetahuan yang telah didapat ke data baru yang belum dikenal sebelumnya**. Untuk memastikan kinerja model pembelajaran mesin, dilakukan evaluasi pada data *testing*. Evaluasi tersebut membantu mendeteksi masalah yang mungkin terjadi, seperti *overfitting* (model terlalu cocok dengan data *training*, sehingga kurang dapat digunakan pada data-data yang baru) atau *underfitting* (model tidak cukup belajar dari data *training*).

Pemisahan data *training* dan *testing* penting untuk dilakukan dengan perhitungan yang seksama, yang bertujuan untuk:

- **Menghindari Bias:**

Jika data *testing* ikut digunakan selama *training*, model pembelajaran mesin bisa saja menghafal data tersebut, yang mengakibatkan kinerja model tampak sangat baik pada data yang sama namun sangat kurang baik saat menghadapi data nyata.

- **Evaluasi Kinerja:**

Dengan menggunakan data *testing* yang terpisah, kita dapat mendapatkan gambaran yang lebih jujur tentang kemampuan model dalam memprediksi data yang tidak dikenal.

Berikut ini adalah contoh pembagian data *training* dan *testing* dengan rasio 80:20 menggunakan scikit-learn:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Muat data contoh dari Google Colab (misalnya, data Iris)
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target # Menambahkan kolom target

# Pisahkan data menjadi fitur (X) dan target (y)
X = df.drop('target', axis=1)
y = df['target']

# Bagi data menjadi data latih dan data uji dengan rasio 80:20
# Fungsi train_test_split dari scikit-learn digunakan untuk membagi data
# test_size=0.2 artinya 20% data akan digunakan untuk pengujian
# random_state digunakan untuk memastikan bahwa pembagian data konsisten setiap kali kode dijalankan
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Tampilkan ukuran data latih dan data uji
print("Ukuran data latih:", X_train.shape)
print("Ukuran data uji:", X_test.shape)
```

AKTIVITAS 1: Diskusi Tentang Konsep Dasar Pemrograman KA

Dalam kelompok kecil (4-5 orang), diskusikan konsep dasar pemrograman KA dan teknik pengajarannya di Fase E & F!

Pertanyaan panduan diskusi:

1. Apa saja konsep dasar pemrograman yang dianggap krusial untuk pengembangan aplikasi KA (misalnya variabel, struktur kontrol, fungsi, dan struktur data)?
2. Bagaimana konsep pemrograman berbasis struktur data (list, tuple, dictionary, set) berperan dalam pengolahan dan analisis data pada aplikasi KA?
3. Mengapa Python sering dijadikan pilihan utama dalam pemrograman KA? Apa keunggulan Python dibandingkan bahasa pemrograman lain dalam konteks aplikasi KA?
4. Bagaimana penggunaan IDE (seperti PyCharm, Jupyter Notebook, atau Visual Studio Code) dapat memfasilitasi proses pembelajaran dan pengembangan aplikasi KA?
5. Teknik pengajaran apa yang menurut Anda paling efektif dalam menyampaikan konsep dasar pemrograman kepada siswa yang belum memiliki latar belakang mendalam tentang KA?

Bagaimana cara mengkombinasikan pendekatan teoretis dan praktis (misalnya, diskusi, laboratorium, dan proyek berbasis coding) untuk mengajarkan pemrograman KA di kelas?

6. Apa jenis evaluasi (misalnya, kuis, tugas coding, proyek kolaboratif) yang paling efektif untuk menilai pemahaman siswa terhadap konsep-konsep pemrograman KA?
7. Bagaimana Anda mendorong siswa untuk merefleksikan proses belajar mereka, terutama terkait tantangan yang dihadapi dalam memahami aspek teknis dan penerapan aplikasi KA?

Paparkan hasil diskusi dalam bentuk teks untuk diunggah ke LMS!

B.2. Menerapkan Pemrograman Kecerdasan Artifisial

B.2.1 Bekerja Dengan Dataset

Data dan dataset yang bersifat reliabel dan mudah diakses sangat penting dalam proses pengembangan KA. **Data adalah fakta dan informasi yang dapat berbentuk teks, angka, gambar, suara, dan lainnya.** Sedangkan, **dataset adalah kumpulan data yang telah disusun, biasanya dalam bentuk matriks, tabel, himpunan (array), ataupun struktur spesifik lainnya.** Terdapat tiga tipe dataset berdasarkan data yang ditanganinya, yaitu:

- 1) **Dataset Terstruktur**, yaitu data yang disusun dalam format tetap, seperti tabel dengan baris dan kolom yang jelas. Contoh: *spreadsheet*, CSV, dan basis data relasional.
- 2) **Dataset Tidak Terstruktur**, yaitu data yang tidak memiliki format atau struktur khusus, seperti teks bebas, gambar, atau video. Contoh: *email*, postingan media sosial, dan file multimedia.
- 3) **Dataset Semi-Terstruktur**, yaitu data yang tidak sepenuhnya terstruktur tetapi memiliki elemen atau tag yang memudahkan pengorganisasian. Contoh: file JSON dan XML.

Mencari dataset yang relevan dan berkualitas merupakan langkah awal yang krusial dalam pengembangan KA dan *machine learning*. Berikut ini ialah beberapa cara untuk mencari dataset:

- **Menggunakan Mesin Pencari:**

Pencarian dataset dapat menggunakan kata kunci spesifik seperti "dataset [topik]" atau "open dataset [topik]" pada mesin pencari (Google, Bing, dan lain-lain). Misalnya, "dataset cuaca" atau "open dataset kesehatan".

- **Platform Dataset Terbuka:**

Banyak situs web yang menyediakan koleksi dataset dari berbagai domain secara terbuka. Beberapa di antaranya adalah:

- ✓ Kaggle
- ✓ UCI Machine Learning Repository
- ✓ Google Dataset Search
- ✓ data.gov dan portal data pemerintah lainnya



Setelah menemukan dataset yang diinginkan, langkah berikutnya adalah mengunduh dataset tersebut. Umumnya, dataset tersedia dalam format CSV, JSON, spreadsheet, atau bahkan format gambar dan video. Berikut ini adalah beberapa cara mengunduh dataset:

- **Secara Manual:**

Pada umumnya, situs penyedia dataset menyediakan tombol atau mekanisme untuk mengunduh. Pengguna tinggal mengklik tombol tersebut atau mengikuti mekanisme unduhan yang tersedia dan menyimpan file ke komputer atau perangkat lainnya yang digunakan.

- **Menggunakan Pemrograman:**

Dataset juga dapat diunduh secara otomatis menggunakan pemrograman tertentu, misalnya bahasa Python dengan bantuan library seperti requests atau wget. Pendekatan ini berguna untuk mengotomatisasi pengambilan data dari sumber yang menyediakan API atau tautan unduh langsung (*direct download*). Berikut ini adalah contoh sederhana mengunduh dataset dengan jenis CSV menggunakan requests dalam bahasa Python:

```
import pandas as pd
from sklearn.model_selection import train_test_split
# Gunakan !wget untuk mengunduh dataset
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data -O iris.csv

# Muat dataset dari file CSV ke DataFrame Pandas
df = pd.read_csv('iris.csv', header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'target'])
```

Setelah dataset berhasil diunduh, langkah berikutnya adalah membacanya ke dalam lingkungan pemrograman (misalnya Python) dan melakukan eksplorasi awal. Library Pandas adalah salah satu perangkat (*tool*) terbaik untuk membaca, memanipulasi, dan memahami dataset. Berikut ini adalah contoh cara membaca dataset dalam format CSV dan JSON menggunakan Pandas:

```

import pandas as pd

# --- Membaca file CSV ---
# Fungsi pd.read_csv() digunakan untuk membaca data dari file CSV.
# 'data.csv' adalah nama file yang akan dibaca.
# Anda dapat mengganti 'data.csv' dengan nama file CSV Anda.
df_csv = pd.read_csv('data.csv') # Baca data CSV ke dalam DataFrame

# Menampilkan beberapa baris pertama data CSV
print("Data dari file CSV:")
print(df_csv.head())

# --- Membaca file JSON ---
# Fungsi pd.read_json() digunakan untuk membaca data dari file JSON.
# 'data.json' adalah nama file yang akan dibaca.
# Anda dapat mengganti 'data.json' dengan nama file JSON Anda.
df_json = pd.read_json('data.json') # Baca data JSON ke dalam DataFrame

# Menampilkan beberapa baris pertama data JSON
print("\nData dari file JSON:")
print(df_json.head())

# Penjelasan:
# - `pd.read_csv()` digunakan untuk membaca data dari file CSV ke dalam Pandas DataFrame.
# - `pd.read_json()` digunakan untuk membaca data dari file JSON ke dalam Pandas DataFrame.
# - `df.head()` digunakan untuk menampilkan beberapa baris pertama dari DataFrame, berguna untuk melihat cuplikan data.

```

Setelah membaca dataset, langkah berikutnya adalah memahami strukturnya. Beberapa metode yang dapat digunakan adalah:

- **Melihat Informasi Umum:**

Informasi umum dataset dapat diketahui dengan menggunakan fungsi info(). Fungsi tersebut menghasilkan informasi jumlah baris, kolom, dan tipe data masing-masing kolom.

```

print("Informasi Dataset CSV:")
print(df_csv.info())

```

- **Statistik Deskriptif:**

Fungsi describe() dapat digunakan untuk menghasilkan ringkasan statistik umum seperti mean, median, standar deviasi, dan persentil.

```

print("Statistik Deskriptif dari Dataset CSV:")
print(df_csv.describe())

```

- **Memeriksa Nilai yang Hilang:**

Dalam melakukan analisis data, sangat penting untuk mengetahui apakah terdapat nilai yang hilang atau missing values di dalam dataset. Keberadaan nilai yang hilang dalam dataset akan sangat mempengaruhi hasil analisis data dan pemodelan pembelajaran mesin.

```
print("Jumlah Missing Values per Kolom:")
print(df_csv.isnull().sum())
```

- **Menampilkan Beberapa Baris Data:**

Fungsi `head()` dan `tail()` dapat digunakan untuk melihat beberapa baris pertama atau terakhir dari dataset. Hal ini menjadi semacam *preview* untuk melihat contoh data yang ada.

```
print("5 Baris Pertama:")
print(df_csv.head())

print("5 Baris Terakhir:")
print(df_csv.tail())
```

Berikut adalah **contoh program lengkap yang menunjukkan alur kerja dari awal hingga akhir dalam bekerja dengan dataset Iris**, yang merupakan dataset populer dan sederhana untuk pemula. Dataset ini berisi data mengenai tiga jenis bunga iris berdasarkan fitur seperti panjang dan lebar sepal (*daun pelindung*) serta petal (*daun bunga*). Iris adalah salah satu kumpulan data paling awal yang digunakan dalam berbagai literatur tentang metode klasifikasi dan digunakan secara luas dalam statistik dan pembelajaran mesin. Kumpulan data Iris berisi 3 kelas yang masing-masing berisi 50 contoh, di mana setiap kelas merujuk pada satu jenis tanaman iris.



Gambar 18. Jenis Tanaman Iris. Sumber:

<http://www.lac.inpe.br/~rafael.santos/Docs/CAP394/WholeStory-Iris.html>

Program ini mencakup proses sebagai berikut:

- 1) Memuat dan mengonversi dataset ke DataFrame.
- 2) Eksplorasi data: menampilkan baris pertama, informasi umum, statistik deskriptif, dan pemeriksaan nilai yang hilang.

Kode sumber “Bekerja Dengan Dataset IRIS” dapat diakses pada tautan:
https://colab.research.google.com/drive/1vwoldLmLrr5FeSARUVYCBP8MTOm_rABN?usp=sharing,

sedangkan *output* yang diperoleh dari program tersebut dapat dilihat pada gambar berikut:

```
5 Baris Pertama Dataset Iris:
  sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm) \
0           5.1            3.5            1.4            0.2
1           4.9            3.0            1.4            0.2
2           4.7            3.2            1.3            0.2
3           4.6            3.1            1.5            0.2
4           5.0            3.6            1.4            0.2

  target species
0      0  setosa
1      0  setosa
2      0  setosa
3      0  setosa
4      0  setosa

Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   sepal length (cm)    150 non-null   float64
 1   sepal width (cm)     150 non-null   float64
 2   petal length (cm)    150 non-null   float64
 3   petal width (cm)     150 non-null   float64
 4   target              150 non-null   int64  
 5   species             150 non-null   object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

Gambar 19. Lima Baris Pertama dan Informasi Dataset Iris

```

statistik Deskriptif:
      sepal length (cm)  sepal width (cm)  petal length (cm) \
count      150.000000    150.000000    150.000000
mean       5.843333     3.057333     3.758000
std        0.828066     0.435866     1.765298
min        4.300000     2.000000     1.000000
25%       5.100000     2.800000     1.600000
50%       5.800000     3.000000     4.350000
75%       6.400000     3.300000     5.100000
max        7.900000     4.400000     6.900000

      petal width (cm)  target
count      150.000000  150.000000
mean       1.199333     1.000000
std        0.762238     0.819232
min        0.100000     0.000000
25%       0.300000     0.000000
50%       1.300000     1.000000
75%       1.800000     2.000000
max        2.500000     2.000000

Jumlah Missing Values per Kolom:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target                0
species               0
dtype: int64

```

Gambar 20. Statistik Deskriptif dan Nilai Hilang pada Dataset Iris

B.2.2 Data Preparation & Pre-Processing

Data Preparation adalah serangkaian kegiatan yang bertujuan untuk mengumpulkan dan menyiapkan data yang akan digunakan dalam proses pembelajaran mesin. Sumber data yang umum digunakan dapat berupa repositori data, basis data, *application programming interfaces* (APIs), dan *platform* data publik seperti portal data pemerintah atau organisasi internasional, yang menyediakan akses ke berbagai dataset untuk analisis dan penelitian, misalnya Kaggle (<https://www.kaggle.com/>) dan Google Research Datasets (<https://research.google/resources/datasets/>)

Indonesia Reading Interest 2020-2023

Tingkat Kegemaran Membaca (TGM) 2020-2023

Data Card Code (1) Discussion (0) Suggestions (0)

About Dataset

This dataset is sourced from the annual survey "Tingkat Kegemaran Membaca" (Reading Interest Level) conducted by the National Library of Indonesia (Perpustakaan Nasional Indonesia). The survey aims to measure the reading habits of Indonesian citizens, including:

- Reading Frequency (RF): How often people read weekly.
- Number of Readings (NRI): The total number of materials read in a quarter.
- Daily Reading Duration (DRD): The average time spent reading daily (in minutes).
- Internet Access Frequency (IAF): Added in 2021, this measures how often respondents access the internet weekly.
- Daily Internet Duration (DID): Introduced in 2021, this records the average daily internet usage (in minutes).

Survey Details

Usability 10.00

License Other (specified in description)

Expected update frequency Annually

Tags

- Beginner
- Literature
- Categorical
- People
- Indonesian

TGM 2020-2023_eng.csv (7.38 kB)

Detail Compact Column 6 of 9 columns ▾

Provinsi	Year	# Daily Read...	Daily Inter...	# Tingkat Ke...	Category
Aceh	2020	95	N/A	54, 68	Moderate
Aceh	2021	103	83, 9	64, 13	High
Aceh	2022	94, 3	123, 4	65, 85	High
Aceh	2023	95	104	66, 64	Moderate
Bali	2020	91	N/A	56, 98	Moderate
Bali	2021	82	100, 2	59, 80	Moderate

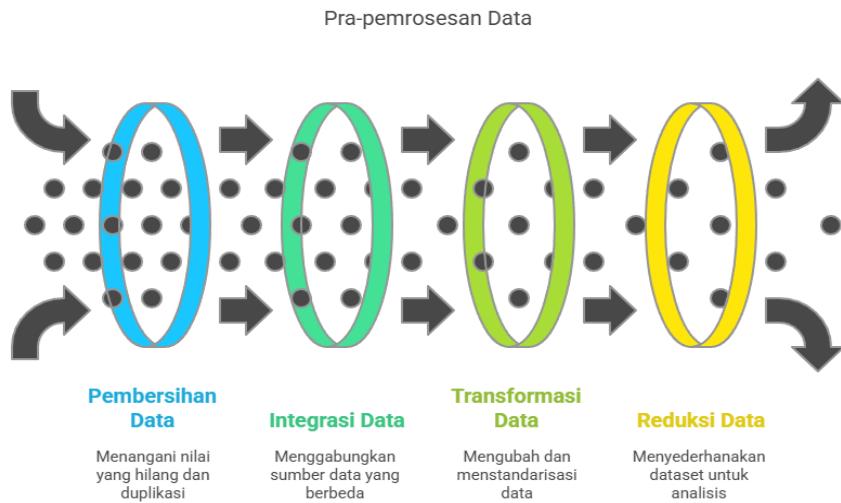
Gambar 21. Contoh Dataset Minat Baca di Indonesia 2020-2023

Sumber: <https://www.kaggle.com>

Pra-pemrosesan data (Data Pre Processing) merupakan serangkaian langkah untuk mengubah data mentah menjadi format yang bersih, terstruktur, dan optimal sebelum dianalisis atau digunakan untuk pelatihan model pembelajaran mesin. Hasil akhir dari proses ini adalah dataset yang telah dipersiapkan dengan baik sehingga dapat meningkatkan akurasi dan efisiensi model. Tahapan utama dalam pra-pemrosesan data meliputi:

- **Pembersihan Data:** menangani nilai yang hilang, menghapus duplikasi, memperbaiki kesalahan, dan menangani pencilan (*outlier*).
- **Integrasi Data:** menggabungkan data dari berbagai sumber ke dalam satu dataset yang terpadu.

- **Transformasi Data:** normalisasi, standarisasi, pengkodean variabel kategori, dan agregasi data.
- Reduksi Data: pemilihan fitur, pemilihan *instance*, reduksi dimensi, dan diskretisasi.



Gambar 22. Pra-pemrosesan Data

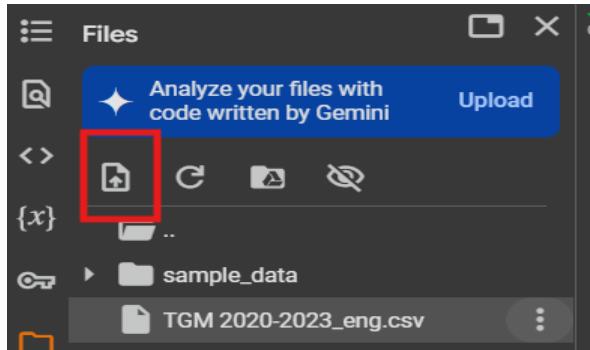
Berikut ini adalah **contoh persiapan dan pra-pemrosesan data menggunakan dataset Indonesia Reading Interest 2020-2023**,

Contoh kode program lengkap data preprataion & pre-processing menggunakan dataset Indonesia Reading Interest dapat dilihat pada tautan:
<https://colab.research.google.com/drive/1d5yPdFKFB8s6hdP8iWsHB69TPOA6caHb?usp=sharing> :

1) Mengunduh dataset melalui tautan resmi

<https://www.kaggle.com/datasets/imaditia/indonesia-reading-interest-2020-2023?resource=download>

2) Mengunggah dataset ke Google Colab notebook yang sudah disiapkan



3) Mengimpor *libraries* yang diperlukan

```
# 1. Import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

import pandas as pd

- Pandas adalah library di Python yang sangat populer untuk mengolah data dalam bentuk tabel (DataFrame).
- Pada baris ini, *library* Pandas diimporkan dan diberikan alias (nama singkat) pd.
- Alias ini memungkinkan kita untuk menulis kode seperti pd.DataFrame() alih-alih menggunakan sintaks semacam pandas.DataFrame().

import numpy as np

- NumPy adalah pustaka dasar untuk komputasi ilmiah dalam bahasa pemrograman Python, khususnya untuk operasi pada array (himpunan).
- NumPy diimporkan dengan alias np. Sehingga kita dapat menggunakan fungsi NumPy seperti np.array(), np.mean() dan sebagainya.

import matplotlib.pyplot as plt

- Matplotlib adalah pustaka untuk membuat grafik dan visualisasi data.
- pyplot adalah modul di dalam Matplotlib yang menyediakan fungsi-fungsi *plotting* (seperti plt.plot(), plt.bar(), dan lain-lain).
- pyplot diimporkan dengan alias plt, sehingga memungkinkan untuk menuliskan plt.plot() alih-alih menggunakan matplotlib.pyplot.plot().

```
import seaborn as sns
```

- Seaborn adalah *library* visualisasi data tingkat lanjut yang dibangun di atas Matplotlib.
- Seaborn dapat menghasilkan grafik yang lebih informatif dan estetis, seperti grafik distribusi (`sns.distplot()` atau `sns.histplot()`), *heatmap*, dan sebagainya.
- Alias `sns` digunakan agar kode menjadi lebih ringkas.

```
from sklearn.preprocessing import MinMaxScaler
```

- `sklearn` (*scikit-learn*) adalah *library* Python yang populer untuk *machine learning*.
- Modul `sklearn.preprocessing` menyediakan berbagai fungsi untuk menyiapkan (*preprocess*) data sebelum membangun model *machine learning*, misalnya normalisasi data, menangani data yang berupa kategori, dan lain-lain.
- `MinMaxScaler` adalah salah satu kelas di dalam `sklearn.preprocessing` yang digunakan untuk menormalisasi data numerik ke dalam bentuk rentangan [0, 1].
- Ketika dituliskan `from sklearn.preprocessing import MinMaxScaler`, maka hanya akan diambil *bagian tertentu* dari library *scikit-learn*, sehingga kita dapat langsung memanggil `MinMaxScaler()` tanpa harus menulis `sklearn.preprocessing.MinMaxScaler()`.

- 4) Membaca *dataset* yang telah diunggah, dan memastikan delimiter atau pembatas data diatur ke titik koma (,) sesuai dengan delimiter yang digunakan oleh *dataset*.

```
# 2. Read the dataset
df = pd.read_csv("/content/TGM 2020-2023_eng.csv", sep=";")
```

5) Memeriksa struktur dan kondisi awal dataset yang sudah dibaca oleh Pandas

```
# Print detected column names
print("== Detected Columns ==")
print(df.columns.tolist(), "\n")

# 3. Display the first few rows to verify the structure
print("== First 5 Rows ==")
print(df.head(), "\n")

# 4. Basic exploration: info, describe, missing values
print("== DataFrame Info ==")
print(df.info(), "\n")
print("== Descriptive Statistics ==")
print(df.describe(include='all'), "\n")
print("== Missing Values Per Column ==")
print(df.isnull().sum(), "\n")
```

print("== Detected Columns ==")

- Baris ini menampilkan teks == Detected Columns == untuk memberikan judul atau penanda bahwa akan ditampilkan daftar kolom dari DataFrame.

print(df.columns.tolist(), "\n")

- df.columns adalah properti Pandas DataFrame yang dapat digunakan untuk menyimpan daftar nama kolom.
- Metode .tolist() digunakan untuk mengubah daftar kolom yang biasanya bertipe Index menjadi struktur data berupa list.
- print(...) kemudian mencetak list nama kolom tersebut, diikuti dengan "\n" untuk menambahkan baris kosong (memberi jarak di output).

print("== First 5 Rows ==")

- Mencetak teks == First 5 Rows == untuk menandakan bahwa akan ditampilkan 5 baris pertama dari DataFrame.

```

    === First 5 Rows ===
      Provinsi Year Reading Frequency per week Number of Readings per Quarter \
0 Aceh 2020 4 2
1 Aceh 2021 5,5 4,5
2 Aceh 2022 5 5,5
3 Aceh 2023 5 5,5
4 Bali 2020 4 2,5

      Daily Reading Duration (in minutes) Internet Access Frequency per Week \
0 95 NaN
1 103 5
2 94,3 5,5
3 95 5,5
4 91 NaN

      Daily Internet Duration (in minutes) \
0 NaN
1 83,9
2 123,4
3 104
4 NaN

      Tingkat Kegemaran Membaca (Reading Interest) Category
0 54,68 Moderate
1 64,13 High
2 65,85 High
3 66,64 Moderate
4 56,98 Moderate

```

```
print(df.head(), "\n")
```

- df.head() adalah fungsi bawaan Pandas yang dapat digunakan untuk menampilkan beberapa baris teratas (default: 5 baris) dari DataFrame sebagai pratinjau.
- Pratinjau beberapa baris data ini berguna untuk memeriksa apakah data sudah terbaca dengan benar dan juga untuk melihat sekilas struktur setiap kolom.

```

    === DataFrame Info ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 9 columns):
 #   Column           Non-Null Count Dtype
 --- 
 0   Provinsi        140 non-null   object
 1   Year            140 non-null   int64
 2   Reading Frequency per week 140 non-null   object
 3   Number of Readings per Quarter 140 non-null   object
 4   Daily Reading Duration (in minutes) 140 non-null   object
 5   Internet Access Frequency per Week 105 non-null   object
 6   Daily Internet Duration (in minutes) 105 non-null   object
 7   Tingkat Kegemaran Membaca (Reading Interest) 140 non-null   object
 8   Category        140 non-null   object
dtypes: int64(1), object(8)
memory usage: 10.0+ KB
None

```

```
print(df.info(), "\n")
```

- df.info() menghasilkan ringkasan dari DataFrame, antara lain: jumlah baris dan kolom, tipe data setiap kolom, serta berapa banyak nilai *non-null* (tidak kosong) di setiap kolom.
- Informasi ini membantu kita untuk memahami seberapa besar dataset, tipe data kolom (apakah numerik, *string*, dan sebagainya), dan apakah ada nilai kosong.

```
== Descriptive Statistics ==
   Provinси      Year Reading Frequency per week \
count      140  140.000000                           140
unique      35          NaN                           7
top        Aceh         NaN                           5
freq         4          NaN                          48
mean       NaN  2021.500000                         NaN
std        NaN  1.122048                         NaN
min        NaN  2020.000000                         NaN
25%        NaN  2020.750000                         NaN
50%        NaN  2021.500000                         NaN
75%        NaN  2022.250000                         NaN
max        NaN  2023.000000                         NaN

   Number of Readings per Quarter Daily Reading Duration (in minutes) \
count                      140                           140
unique                     11                           91
top                       5,5                          98
freq                      41                           6
mean                      NaN                         NaN
std                       NaN                         NaN
min                      NaN                         NaN
25%                      NaN                         NaN
50%                      NaN                         NaN
75%                      NaN                         NaN
```

```
print(df.describe(include='all'), "\n")
```

- df.describe() digunakan untuk menampilkan ringkasan statistik seperti *mean* (rata-rata), *std* (standar deviasi), *min*, *max*, dan *quartiles* untuk kolom numerik secara *default*.
- Ditambahkannya include='all' akan membuat Pandas menampilkan statistik kolom non-numerik (misalnya jumlah nilai unik, nilai terbanyak, dan sebagainya).
- Deskripsi yang dihasilkan pada tahap ini sangat berguna untuk memahami karakteristik data secara umum.

```
    === Missing Values Per Column ===
    Provinsi                      0
    Year                          0
    Reading Frequency per week   0
    Number of Readings per Quarter 0
    Daily Reading Duration (in minutes) 0
    Internet Access Frequency per Week 35
    Daily Internet Duration (in minutes) 35
    Tingkat Kegemaran Membaca (Reading Interest) 0
    Category                      0
    dtype: int64
```

```
print(df.isnull().sum(), "\n")
```

- df.isnull() menghasilkan DataFrame dengan nilai *True/False* untuk setiap sel, tergantung apakah data kosong atau tidak.
- Metode .sum() pada hasil isnull() akan menghitung berapa banyak nilai *True* (data kosong) di setiap kolom.
- Proses ini membantu kita untuk mengetahui kolom mana yang memiliki *missing values* dan berapa banyak.

6) Memeriksa dan Menghapus Data Duplikat

```
# 5. Remove duplicate rows (if any)
duplicates = df.duplicated().sum()
print("Number of duplicate rows:", duplicates)
if duplicates > 0:
    df = df.drop_duplicates()
    print("After removing duplicates, shape:", df.shape, "\n")
else:
    print("No duplicate rows found.\n")
```

```
duplicates = df.duplicated().sum()
```

- df.duplicated() adalah fungsi bawaan Pandas yang mengembalikan Series (kolom) berisi nilai *True* atau *False* untuk setiap baris, tergantung apakah baris tersebut merupakan duplikat (*True*) atau tidak (*False*).
- Kemudian, .sum() akan menjumlahkan nilai *True* di Series tersebut. Karena *True* bernilai 1 dan *False* bernilai 0, hasil akhirnya adalah banyaknya baris duplikat dalam DataFrame. Nilai tersebut disimpan ke dalam variabel duplicates.

```
print("Number of duplicate rows:", duplicates)
```

- Mencetak jumlah baris yang terdeteksi sebagai duplikat, sehingga kita tahu berapa banyak baris duplikat di dalam DataFrame.

```
if duplicates > 0:
```

- Mengecek apakah jumlah baris duplikat (duplicates) lebih besar dari 0. Jika iya, artinya memang ada baris duplikat yang harus dihapus.

```
df = df.drop_duplicates()
```

- drop_duplicates() adalah fungsi Pandas untuk menghapus baris duplikat pada DataFrame. Hasilnya adalah DataFrame baru yang sudah bersih dari baris duplikat. Di sini kita menimpa (overwrite) DataFrame lama df dengan DataFrame hasil penghapusan duplikat.

```
print("After removing duplicates, shape:", df.shape, "\n")
```

- Menampilkan ukuran (jumlah baris dan kolom) DataFrame setelah baris duplikat dihapus, agar kita tahu perubahan yang terjadi.
- df.shape mengembalikan (jumlah_baris, jumlah_kolom).

```
else:
```

- Jika tidak ada duplikat (duplicates == 0), maka blok if di atas tidak dijalankan.

```
print("No duplicate rows found.\n")
```

- Menampilkan pesan bahwa tidak ada baris duplikat yang ditemukan, diikuti baris kosong ("\n").

```
Number of duplicate rows: 0
No duplicate rows found.
```

7) Memeriksa dan Menangani Data Yang Hilang

```
# 6. Handle missing values and convert columns that contain commas to proper numeric
# Define the reading interest column
reading_col = "Tingkat Kegemaran Membaca (Reading Interest)"

# List of columns with numeric values stored as strings (with commas)
numeric_cols_with_commas = [
    "Reading Frequency per week",
    "Number of Readings per Quarter",
    "Daily Reading Duration (in minutes)",
    "Internet Access Frequency per Week",
    "Daily Internet Duration (in minutes)",
    reading_col
]

for col in numeric_cols_with_commas:
    if col in df.columns:
        # Replace commas with dots
        df[col] = df[col].astype(str).str.replace(',', '.')
        # Convert to numeric
        df[col] = pd.to_numeric(df[col], errors='coerce')
        # Fill missing with median
        median_val = df[col].median()
        df[col] = df[col].fillna(median_val)
        print(f"Missing values in '{col}' filled with median: {median_val}")

# Convert "Year" to numeric if it exists
if 'Year' in df.columns:
    df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
    year_median = df['Year'].median()
    df['Year'] = df['Year'].fillna(year_median)
    print(f"Missing values in 'Year' filled with median: {year_median}")

# Fill missing values in categorical columns with mode
categorical_cols = ["Provinsi", "Category"]
for cat_col in categorical_cols:
    if cat_col in df.columns:
        mode_val = df[cat_col].mode()[0]
        df[cat_col] = df[cat_col].fillna(mode_val)
        print(f"Missing values in '{cat_col}' filled with mode: {mode_val}")
print()
```

reading_col = "Tingkat Kegemaran Membaca (Reading Interest)"

- Pada bagian ini didefinisikan variabel reading_col yang berisi nama kolom “Tingkat Kegemaran Membaca (Reading Interest)”. Variabel ini membantu kita dalam mengakses kolom tersebut dengan mudah.

```
numeric_cols_with_commas = [...]
```

- Ini adalah daftar nama kolom yang seharusnya berisi data numerik, tetapi saat ini nilainya mungkin disimpan sebagai teks (*string*) dengan tanda koma (,) sebagai pemisah. Contoh: "Reading Frequency per week", "Number of Readings per Quarter", dan seterusnya.

```
for col in numeric_cols_with_commas:
```

- Bagian ini melakukan iterasi untuk setiap kolom yang terdaftar di `numeric_cols_with_commas`.

```
if col in df.columns:
```

- Memeriksa apakah kolom tersebut memang ada di DataFrame `df`. Jika ada, maka dilakukan pemrosesan.

```
df[col] = df[col].astype(str).str.replace(',', '.')
```

- Baris ini mengubah setiap nilai di kolom `col` menjadi *string*, lalu mengganti tanda koma dengan titik. Misalnya, "5,5" akan diubah menjadi "5.5". Hal ini perlu dilakukan karena Python (dan banyak bahasa lainnya) menggunakan titik (.) sebagai pemisah desimal, bukan koma.

```
df[col] = pd.to_numeric(df[col], errors='coerce')
```

- Setelah koma diganti dengan titik, isi kolom dikonversi menjadi tipe data numerik (`float`).
- Parameter `errors='coerce'` artinya jika ada nilai yang tidak bisa dikonversi menjadi angka, nilai tersebut akan diganti dengan `NaN` (Not a Number).

```
median_val = df[col].median()
```

- Menghitung nilai median dari kolom tersebut. Median adalah nilai tengah yang diambil setelah data diurutkan.
- Median sering digunakan untuk mengganti nilai kosong (*missing values*) pada data numerik agar tidak terlalu terpengaruh oleh *outlier* (penciran/nilai ekstrim).

```
df[col] = df[col].fillna(median_val)
```

- Mengganti semua nilai kosong (`NaN`) di kolom tersebut dengan nilai median.
- Ini adalah strategi *imputation* yang sederhana namun efektif untuk menangani *missing values*.

```
print(f"Missing values in '{col}' filled with median: {median_val}")
```

- Mencetak pesan yang memberitahu kita berapa nilai median yang digunakan untuk kolom itu.
- Huruf f di depan string menandakan *f-string* di Python, yang memungkinkan kita menempatkan variabel di dalam string yang dibungkus dengan kurung kurawal {}.

```
if 'Year' in df.columns: ...
```

- Memeriksa apakah kolom 'Year' ada di DataFrame. Jika ya, kolom tersebut perlu dikonversi menjadi numerik dan *missing values*-nya diganti dengan median.

```
categorical_cols = ["Provinsi", "Category"]
```

- Mendefinisikan daftar kolom yang bersifat kategoris, yaitu kolom yang berisi nilai teks atau kategori.

```
for cat_col in categorical_cols:
```

- Melakukan iterasi untuk setiap kolom kategoris.

```
mode_val = df[cat_col].mode()[0]
```

- mode() adalah nilai yang paling sering muncul di kolom tersebut, atau biasa disebut dengan modus. Kita menggunakan [0] untuk mengambil nilai pertama (jika ada beberapa nilai yang frekuensi kemunculannya sama banyak).

```
df[cat_col] = df[cat_col].fillna(mode_val)
```

- Mengganti nilai kosong di kolom kategoris dengan mode_val, yaitu nilai yang paling sering muncul. Ini adalah strategi yang umum untuk menangani *missing values* dalam data yang berbentuk kategori.

```
print(f"Missing values in '{cat_col}' filled with mode: {mode_val}")
```

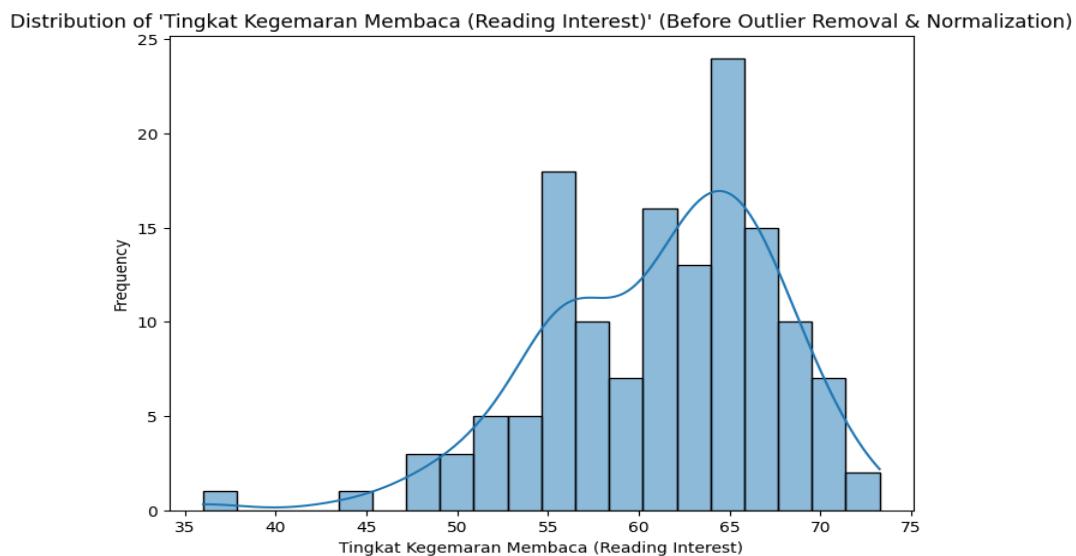
- Mencetak pesan bahwa *missing values* di kolom kategoris telah diganti dengan nilai modus.

```
Missing values in 'Reading Frequency per week' filled with median: 4.75
Missing values in 'Number of Readings per Quarter' filled with median: 4.5
Missing values in 'Daily Reading Duration (in minutes)' filled with median: 97.2
Missing values in 'Internet Access Frequency per Week' filled with median: 5.5
Missing values in 'Daily Internet Duration (in minutes)' filled with median: 113.7
Missing values in 'Tingkat Kegemaran Membaca (Reading Interest)' filled with median: 62.195
Missing values in 'Year' filled with median: 2021.5
Missing values in 'Provinsi' filled with mode: Aceh
Missing values in 'Category' filled with mode: Moderate
```

8) Visualisasi Data

```
# 7. Visualize the reading interest BEFORE normalization
plt.figure(figsize=(8, 6))
sns.histplot(df[reading_col], bins=20, kde=True)
plt.title(f"Distribution of '{reading_col}' (Before Outlier Removal & Normalization)")
plt.xlabel(reading_col)
plt.ylabel("Frequency")
plt.show()
```

Kode program di atas digunakan untuk memvisualisasikan grafik batang distribusi minat baca.



Gambar 23. Distribusi Tingkat Kegemaran Membaca Sebelum Dilakukan Penghapusan Pencilan dan Normalisasi

9) Mengenal pencilan atau *Outlier*

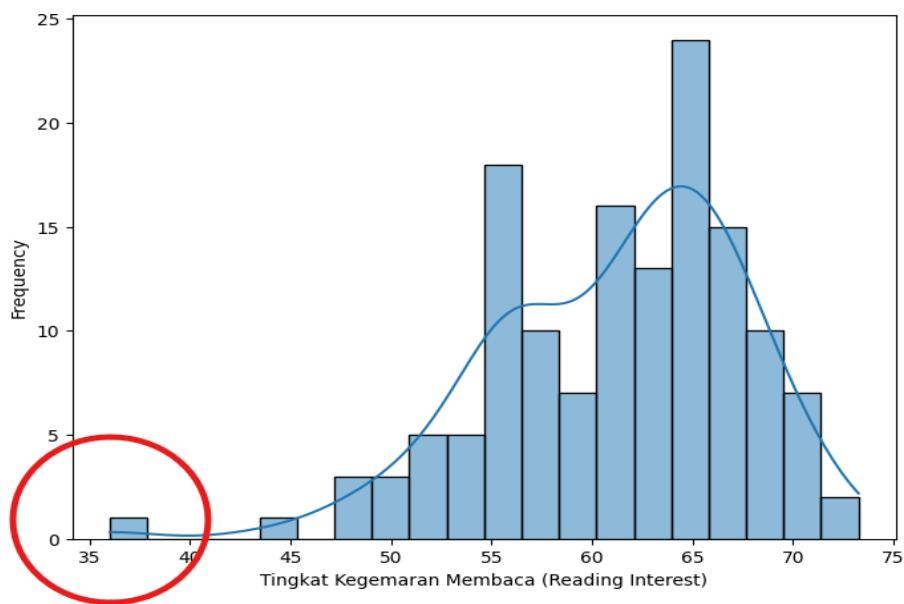
Outlier adalah data yang memiliki nilai atau karakteristik yang sangat berbeda (jauh) dari sebagian besar data lainnya. Dalam konteks analisis data atau pembelajaran mesin, *outlier* bisa diibaratkan seperti “pencilan” atau “nilai ekstrem” yang tidak mewakili pola umum pada dataset. Berikut adalah beberapa hal yang perlu diketahui terkait *outlier*:

- **Penyebab Munculnya *Outlier***

- Kesalahan Pengukuran atau Entri Data: Misalnya, salah memasukkan nilai “1000” padahal seharusnya “100”.

- Variasi Alami Data: Beberapa fenomena memang dapat menghasilkan nilai yang ekstrem, misalnya pendapatan tahunan seorang miliarder di tengah data pendapatan orang biasa.
 - Perubahan Kondisi atau Kejadian yang Langka: Misalnya, lonjakan penjualan suatu toko akibat diskon besar-besaran atau adanya pandemi.
- **Dampak yang Mungkin Terjadi Jika Outlier**
 - Mempengaruhi statistik deskriptif: Nilai *mean* (rata-rata) dan standar deviasi dapat menjadi bias atau melenceng jauh karena adanya *outlier*.
 - Mempengaruhi hasil Pemodelan: Dalam pembelajaran mesin, *outlier* dapat membuat model menjadi sulit dalam mempelajari pola umum, sehingga dapat mengakibatkan performa model menurun.
 - Kesalahan interpretasi: Tanpa penanganan yang tepat, *outlier* bisa menyebabkan kesimpulan yang kurang tepat dalam analisis data.
 - **Metode Deteksi Outlier**
 - IQR (*Interquartile Range*): Menghitung Q1 (kuartil bawah) dan Q3 (kuartil atas), lalu menghitung selisihnya ($IQR = Q3 - Q1$). *Outlier* biasanya didefinisikan sebagai data yang berada di bawah ($Q1 - 1.5 * IQR$) atau di atas ($Q3 + 1.5 * IQR$).
 - Z-score: Mengukur seberapa jauh suatu nilai dari rata-rata dalam satuan standar deviasi. Data dengan Z-score di atas ambang tertentu (misalnya > 3 atau < -3) dianggap sebagai *outlier*.
 - Melalui visualisasi: Diagram atau plot seperti *boxplot*, *scatter plot*, atau *histogram* dapat membantu mendeteksi *outlier* secara visual.
 - **Penanganan Outlier**
 - Menghapus Outlier: Jika *outlier* disebabkan oleh kesalahan data (*error*) dan jumlahnya tidak banyak, maka *outlier* dapat dihapus.
 - Mengganti Nilai Outlier: Nilai *outlier* kadang dapat diganti dengan nilai rata-rata atau nilai median.
 - Transformasi Data: Metode seperti *log-transform*, *square root*, atau *winsorizing* dapat mengurangi dampak nilai ekstrim.
 - Mempertahankan Outlier: Dalam beberapa kasus, *outlier* adalah bagian penting dari data (contoh: total penjualan yang luar biasa tinggi yang terjadi pada periode tertentu). Dalam hal ini, *outlier* perlu dipertahankan untuk analisis lebih lanjut.

- Pentingnya **Domain Knowledge**
 - Tidak semua *outlier* perlu dihapus. Kadang, *outlier* justru menunjukkan fenomena menarik atau ekstrem yang justru relevan untuk tetap dipertimbangkan dalam analisis data.
 - Keputusan untuk menghapus, mengganti, atau mempertahankan *outlier* sangat bergantung pada konteks dan tujuan analisis.
 -



Gambar 24: Outlier Pada Data

10) Contoh penanganan Outlier

```
# 8. Detect and remove outliers using the IQR method
def remove_outliers_iqr(data, cols):
    data_clean = data.copy()
    for c in cols:
        Q1 = data_clean[c].quantile(0.25)
        Q3 = data_clean[c].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outlier_indices = data_clean[(data_clean[c] < lower_bound) | (data_clean[c] > upper_bound)].index
        print(f"Column '{c}': {len(outlier_indices)} outliers detected.")
        data_clean = data_clean.drop(index=outlier_indices)
    return data_clean

# Identify numeric columns for outlier detection
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print("Numeric columns:", numeric_cols, "\n")

# Remove outliers from numeric columns
df_no_outliers = remove_outliers_iqr(df, numeric_cols)
print("\nShape after outlier removal:", df_no_outliers.shape)

# Visualize the reading interest AFTER outlier removal
plt.figure(figsize=(8, 6))
sns.histplot(df_no_outliers[reading_col], bins=20, kde=True, color='orange')
plt.title(f"Distribution of '{reading_col}' (After Outlier Removal)")
plt.xlabel(reading_col)
plt.ylabel("Frequency")
plt.show()
```

```
def remove_outliers_iqr(data, cols):
```

- Mendefinisikan sebuah fungsi bernama `remove_outliers_iqr` yang menerima dua buah parameter, yaitu:
 - `data`: DataFrame yang akan diproses.
 - `cols`: Daftar kolom numerik yang akan dicek *outlier*-nya.

```
data_clean = data.copy()
```

- Membuat salinan (`copy`) dari DataFrame `data` sehingga perubahan yang dilakukan tidak memengaruhi DataFrame asli.

```
for c in cols:
```

- Melakukan iterasi untuk setiap kolom numerik yang ada di dalam `cols`.

```
Q1 = data_clean[c].quantile(0.25)
```

- Menghitung kuartil pertama (`Q1`) dari kolom `c`, yaitu nilai di mana 25% data berada di bawahnya.

```
Q3 = data_clean[c].quantile(0.75)
```

- Menghitung kuartil ketiga (`Q3`) dari kolom `c`, yaitu nilai di mana 75% data berada di bawahnya.

```
IQR = Q3 - Q1
```

- *IQR* (*Interquartile Range*) adalah selisih antara `Q3` dan `Q1`. *IQR* menggambarkan rentang data yang berada di “tengah” (50% data).

```
lower_bound = Q1 - 1.5 * IQR
```

- Batas bawah (*lower bound*) untuk *outlier* ditentukan sebagai `Q1` dikurangi $1.5 \times IQR$.
- Jika suatu nilai berada di bawah batas ini, maka nilai tersebut dianggap sebagai *outlier*.

```
upper_bound = Q3 + 1.5 * IQR
```

- Batas atas (*upper bound*) untuk *outlier* ditentukan sebagai `Q3` ditambah $1.5 \times IQR$.
- Jika suatu nilai berada di atas batas ini, nilai tersebut dianggap *outlier*.

```
outlier_indices = data_clean[(data_clean[c] < lower_bound) | (data_clean[c] > upper_bound)].index
```

- Mengambil *index* dari semua baris di mana nilai kolom c lebih kecil dari *lower_bound* atau lebih besar dari *upper_bound*.
- Operator **|** berarti “atau” (OR) dalam kondisi.

```
print(f"Column '{c}': {len(outlier_indices)} outliers detected.")
```

- Mencetak informasi jumlah outlier yang terdeteksi pada kolom c.
- f-string (`f"..."`) digunakan untuk menampilkan variabel secara langsung di dalam string.

```
data_clean = data_clean.drop(index=outlier_indices)
```

- Menghapus baris-baris yang mengandung *outlier* dari DataFrame *data_clean* berdasarkan index yang diperoleh di langkah sebelumnya.

```
return data_clean
```

- Setelah semua kolom di dalam *cols* diproses, fungsi mengembalikan DataFrame yang sudah bebas dari *outlier*.

```
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
```

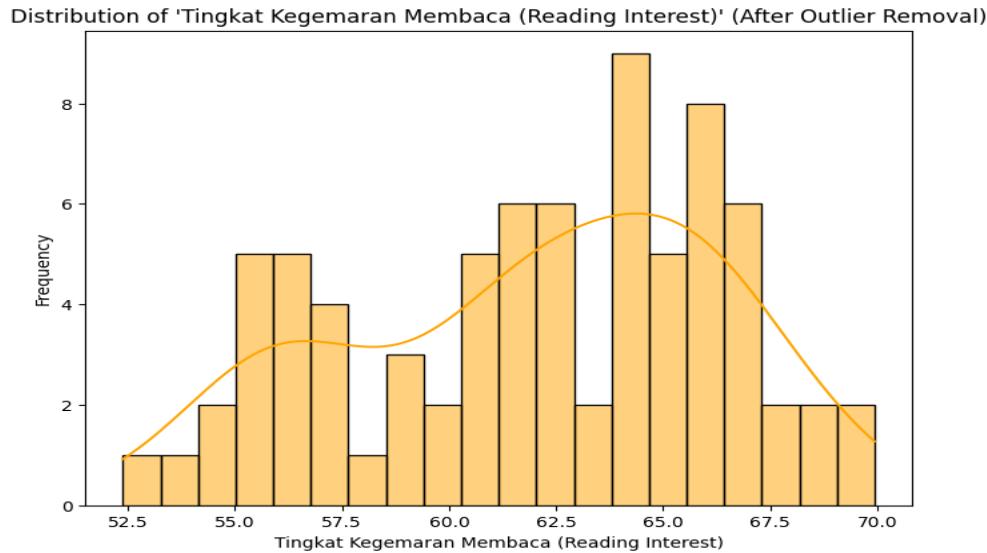
- Mengidentifikasi semua kolom yang bertipe numerik dalam DataFrame *df*.
- *select_dtypes* digunakan untuk memilih kolom berdasarkan tipe data (pada kasus ini hanya ada satu tipe, yaitu numerik), lalu metode `.columns.tolist()` mengubahnya menjadi *list* Python biasa.

```
df_no_outliers = remove_outliers_iqr(df, numeric_cols)
```

- Memanggil fungsi *remove_outliers_iqr* dengan *df* (DataFrame asli) dan daftar kolom numerik *numeric_cols*.
- Hasilnya adalah DataFrame baru (*df_no_outliers*) yang sudah tidak mengandung outlier.

```
print("\nShape after outlier removal:", df_no_outliers.shape)
```

- Menampilkan ukuran (jumlah baris dan kolom) DataFrame setelah outlier dihapus, agar kita tahu seberapa banyak data yang hilang akibat penghapusan *outlier*.



Gambar 25: Visualisasi Data Setelah Penghapusan Outlier

11) Normalisasi Data

Normalisasi dalam konteks data science adalah **proses untuk mengubah skala nilai pada suatu kolom (atau fitur) menjadi rentang tertentu, biasanya $[0,1][0, 1][0,1]$ atau $[-1,1][-1, 1]$** . Tujuan utama normalisasi adalah agar setiap fitur memiliki skala yang seragam sehingga tidak ada satu fitur yang mendominasi atau membuat bias dalam proses analisis atau pemodelan. Selain itu, normalisasi juga bertujuan untuk:

- Menghindari Dominasi Fitur Tertentu: Jika satu kolom memiliki rentang nilai yang sangat besar (misalnya 0–10.000), sementara kolom lain hanya 0–10, maka beberapa model pembelajaran mesin seperti *k-nearest neighbors* atau *neural network* bisa menjadi bias terhadap kolom dengan rentang nilai yang lebih besar.
- Mempercepat Konvergensi Model: Dalam beberapa algoritma (misalnya *gradient descent* pada *neural network*), normalisasi dapat membantu perhitungan supaya pembelajaran mesin berjalan lebih stabil dan cepat menemukan solusi optimal.
- Mempermudah Interpretasi: Dengan rentang nilai yang seragam, kita lebih mudah melakukan perbandingan antar-fitur.

Berikut ini adalah contoh potongan kode program dalam bahasa Python yang digunakan untuk melakukan normalisasi:

```
# 9. Normalize the data AFTER outlier removal using MinMaxScaler
df_no_outliers_normalized = df_no_outliers.copy()
scaler = MinMaxScaler()
# Re-identify numeric columns in the outlier-removed dataset
numeric_cols_no_outliers = df_no_outliers_normalized.select_dtypes(include=[np.number]).columns.tolist()
df_no_outliers_normalized[numeric_cols_no_outliers] = scaler.fit_transform(df_no_outliers_normalized[numeric_cols_no_outliers])
```

df_no_outliers_normalized = df_no_outliers.copy()

- Baris ini membuat salinan (copy()) dari DataFrame df_no_outliers, yang sebelumnya sudah dihapus outlier-nya.
- Penyalinan perlu dilakukan supaya jika terjadi perubahan atau kesalahan saat normalisasi, kita masih punya DataFrame df_no_outliers yang asli.

scaler = MinMaxScaler()

- Di sini, kita membuat objek scaler dari kelas MinMaxScaler, yaitu sebuah *scaler* (penyesuaikan skala) dari library scikit-learn.
- *MinMaxScaler* akan mengubah nilai-nilai pada setiap kolom numerik ke dalam rentang [0,1][0, 1][0,1].
- Rumusnya adalah:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

numeric_cols_no_outliers =

df_no_outliers_normalized.select_dtypes(include=[np.number]).columns.tolist()

- Baris ini mencari kolom-kolom bertipe numerik pada DataFrame df_no_outliers_normalized.
- `select_dtypes(include=[np.number])` akan memilih semua kolom yang bertipe data numerik (misalnya int64 atau float64).
- Kemudian `.columns.tolist()` mengubah daftar nama kolom (yang tadinya bertipe Index) menjadi list Python biasa.
- Hasilnya disimpan di numeric_cols_no_outliers, yang berisi nama-nama kolom numerik yang siap dinormalisasi.

df_no_outliers_normalized[numeric_cols_no_outliers] =

scaler.fit_transform(df_no_outliers_normalized[numeric_cols_no_outliers])

- Bagian ini adalah inti proses normalisasi. Kita menerapkan metode `fit_transform` pada kolom-kolom numerik.
- `fit_transform` akan melakukan dua hal:
 1. `fit`: Mencari nilai minimum (`X_min`) dan maksimum (`X_max`) pada setiap kolom numerik.
 2. `transform`: Menerapkan rumus Min-Max ke setiap nilai dan mengubahnya ke rentang $[0,1][0, 1][0,1]$.
- Hasil transformasi (berupa array NumPy) kemudian ditimpakembali ke DataFrame `df_no_outliers_normalized` pada kolom yang sama.
- Setelah baris ini dieksekusi, seluruh kolom numerik di `df_no_outliers_normalized` akan memiliki nilai yang sudah di-scale antara 0 dan 1.

12) Visualisasi Data Yang Sudah “Bersih”

```
# Visualize the reading interest AFTER normalization
plt.figure(figsize=(8, 6))
sns.histplot(df_no_outliers_normalized[reading_col], bins=20, kde=True, color='purple')
plt.title(f"Distribution of '{reading_col}' (After Normalization on Outlier-Free Data)")
plt.xlabel(f"{reading_col} (Normalized)")
plt.ylabel("Frequency")
plt.show()

print("\nSample of normalized data (after outlier removal):")
print(df_no_outliers_normalized.head())

# 10. Save the preprocessed datasets
df_no_outliers_normalized.to_csv("TGM_2020-2023_normalized.csv", index=False)
df_no_outliers.to_csv("TGM_2020-2023_cleaned.csv", index=False)
print("\nNormalized dataset (outlier-free) saved as 'TGM_2020-2023_normalized.csv'.")
print("Cleaned (no outliers) dataset saved as 'TGM_2020-2023_cleaned.csv'.")
```

`plt.figure(figsize=(8, 6)):`

- Membuat `figure` baru dengan ukuran 8x6 inci, sehingga tampilan histogram lebih jelas.

`sns.histplot(...)`: Menggunakan Seaborn untuk membuat histogram kolom `reading_col` pada DataFrame `df_no_outliers_normalized`.

- `bins=20` artinya kita membagi data menjadi 20 interval pada sumbu x.
- `kde=True` akan menambahkan `kernel density estimation` di atas histogram, membantu memvisualisasikan distribusi data.

- `color='purple'` mengatur warna histogram menjadi ungu.

plt.title(...):

- Memberi judul pada grafik yang dihasilkan

plt.xlabel(...) dan plt.ylabel(...):

- Mengatur label sumbu x dan y. Di sini, sumbu x menunjukkan nilai yang sudah dinormalisasi, dan sumbu y menunjukkan frekuensi data.

plt.show():

- Menampilkan grafik di layar (khususnya di Google Colab atau Jupyter Notebook).

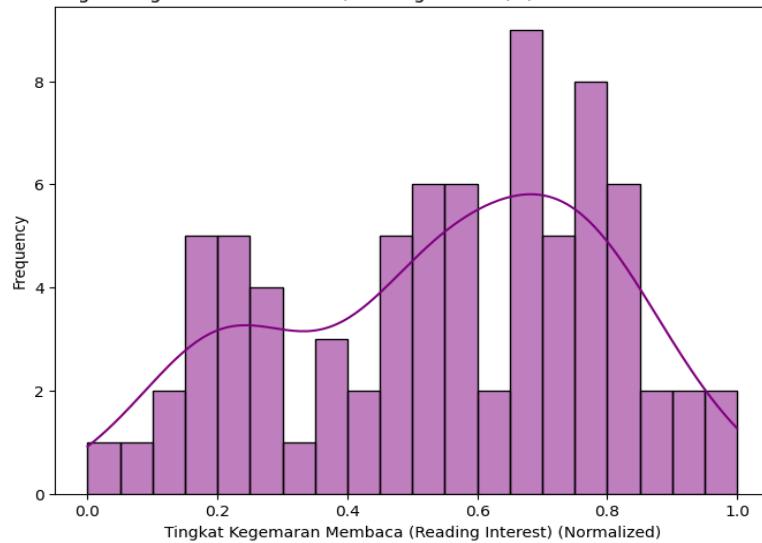
```
df_no_outliers_normalized.to_csv("TGM_2020-2023_normalized.csv",
index=False):
```

- Menyimpan DataFrame `df_no_outliers_normalized` ke dalam file CSV bernama "TGM_2020-2023_normalized.csv". Parameter `index=False` berarti kita tidak menyertakan indeks DataFrame ke dalam file CSV.

```
df_no_outliers.to_csv("TGM_2020-2023_cleaned.csv", index=False):
```

- Menyimpan DataFrame `df_no_outliers` (yang sudah bersih dari *outlier* namun belum dinormalisasi) ke file CSV bernama "TGM_2020-2023_cleaned.csv".

Distribution of 'Tingkat Kegemaran Membaca (Reading Interest)' (After Normalization on Outlier-Free Data)



Gambar 26: Grafik Distribusi Data Setelah Pembersihan Outlier dan Normalisasi

```

Sample of normalized data (after outlier removal):
   Provinsi      Year  Reading Frequency per week \
0    Aceh  0.000000          0.000000
2    Aceh  0.666667          0.666667
3    Aceh  1.000000          0.666667
4    Bali  0.000000          0.000000
5    Bali  0.333333          1.000000

   Number of Readings per Quarter  Daily Reading Duration (in minutes) \
0                      0.000                  0.309577
2                      0.875                  0.293987
3                      0.875                  0.309577
4                      0.125                  0.220490
5                      0.625                  0.020045

   Internet Access Frequency per Week  Daily Internet Duration (in minutes) \
0                      0.0                  0.496324
2                      0.0                  0.852941
3                      0.0                  0.139706
4                      0.0                  0.496324
5                      0.0                  0.000000

   Tingkat Kegemaran Membaca (Reading Interest)  Category
0                      0.130979  Moderate
2                      0.767084  High
3                      0.812073  Moderate
4                      0.261959  Moderate
5                      0.422551  Moderate

Normalized dataset (outlier-free) saved as 'TGM_2020-2023_normalized.csv'.
Cleaned (no outliers) dataset saved as 'TGM_2020-2023_cleaned.csv'.

```

B.2.3. Data Visualization

Dalam konteks pembelajaran mesin, *data visualization* adalah **teknik menyajikan data dan hasil analisis model dalam bentuk grafik, diagram, atau peta untuk memudahkan identifikasi pola dan hubungan antar variabel**. Teknik visualisasi seperti *scatter plot*, *heatmap*, dan *dendrogram* sangat membantu dalam mengevaluasi performa model serta mendeteksi adanya anomali atau *outlier*. Proses ini memudahkan para peneliti dan praktisi untuk menginterpretasikan data mentah secara lebih intuitif, sehingga dapat menentukan langkah optimasi model yang dapat dilakukan berikutnya.

Tabel 3. Perbandingan Berbagai Teknik Visualisasi Data

Teknik Visualisasi Data	Deskripsi	Kapan Digunakan
Histogram	Menampilkan distribusi frekuensi dari data numerik.	Digunakan untuk memeriksa sebaran dan frekuensi data, seperti mendeteksi

		normalitas, skewness, atau distribusi nilai.
Boxplot	Menunjukkan ringkasan statistik data, termasuk median, kuartil, dan outlier.	Cocok untuk memahami sebaran data dan mendeteksi nilai ekstrim (outlier) dalam data numerik.
Scatter Plot	Menggambarkan hubungan antara dua variabel numerik.	Digunakan untuk mengevaluasi korelasi atau pola hubungan antara dua variabel, seperti hubungan antara pendapatan dan pengeluaran.
Heatmap	Menampilkan matriks korelasi antar variabel dalam bentuk visual.	Ideal untuk melihat hubungan antar banyak variabel sekaligus, misalnya memeriksa korelasi antar variabel dalam dataset numerik.
Bar Plot	Menampilkan perbandingan nilai atau frekuensi untuk data kategorik.	Digunakan untuk membandingkan nilai agregat (misalnya rata-rata, jumlah) atau frekuensi antar kategori, seperti perbandingan penjualan per produk.

Contoh visualisasi berikut ini menggunakan dataset “tips” yang memiliki struktur kolom dan sepuluh data pertama, sebagai berikut:

```

total_bill  tip   sex      smoker    day   time   size
16.99     1.01  Female   No        Sun   Dinner 2
10.34     1.66  Male     No        Sun   Dinner 3
21.01     3.5   Male     No        Sun   Dinner 3
23.68     3.31  Male     No        Sun   Dinner 2
24.59     3.61  Female   No        Sun   Dinner 4
25.29     4.71  Male     No        Sun   Dinner 4
8.77      2      Male     No        Sun   Dinner 2
26.88     3.12  Male     No        Sun   Dinner 4
15.04     1.96  Male     No        Sun   Dinner 2

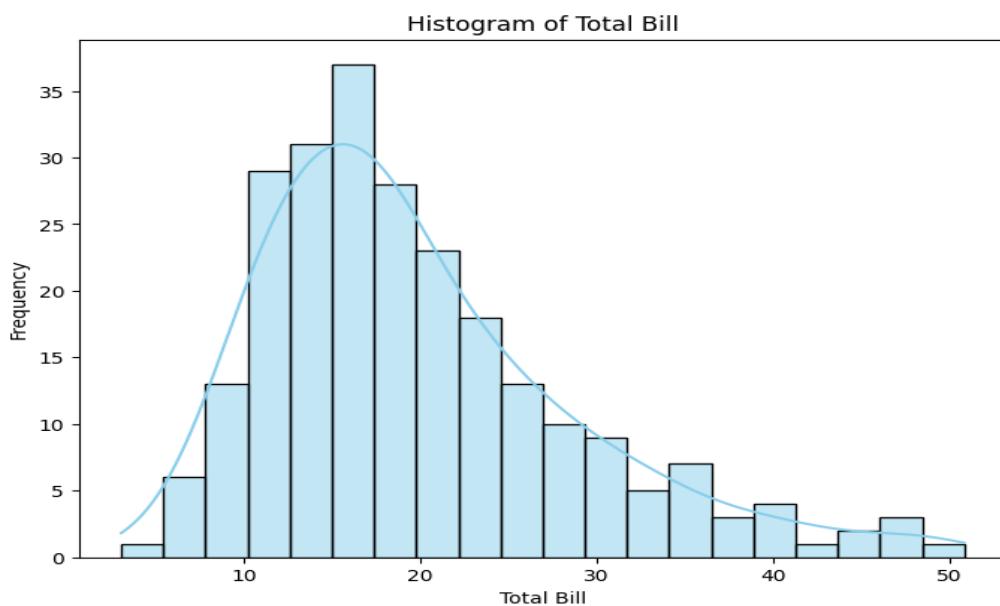
```

Kode program untuk menampilkan berbagai grafik untuk memvisualisasikan data dari dataset “tips” dapat dilihat pada tautan:

<https://colab.research.google.com/drive/1jt4cRTvaM18UlxJQ3VqpgzhZmoiTGFvn?usp=sharing>

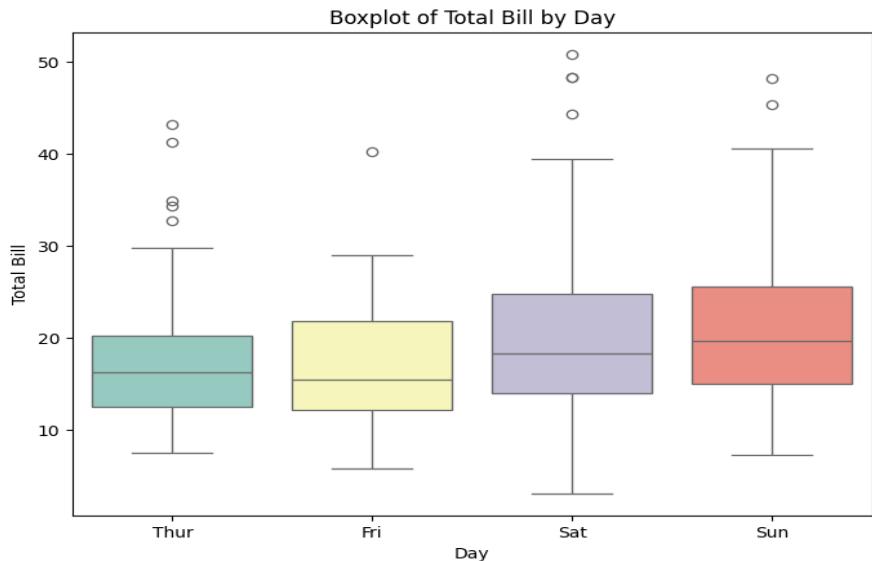
Hasil visualisasi data dari kode program di atas, dapat dilihat pada grafik-grafik berikut ini:

```
# 1. Histogram: Menampilkan distribusi frekuensi dari kolom 'total_bill'  
plt.figure(figsize=(8, 6)) # Mengatur ukuran figure  
sns.histplot(data=tips, x='total_bill', bins=20, kde=True, color='skyblue')  
plt.title("Histogram of Total Bill") # Judul grafik  
plt.xlabel("Total Bill") # Label sumbu X  
plt.ylabel("Frequency") # Label sumbu Y  
plt.show()
```



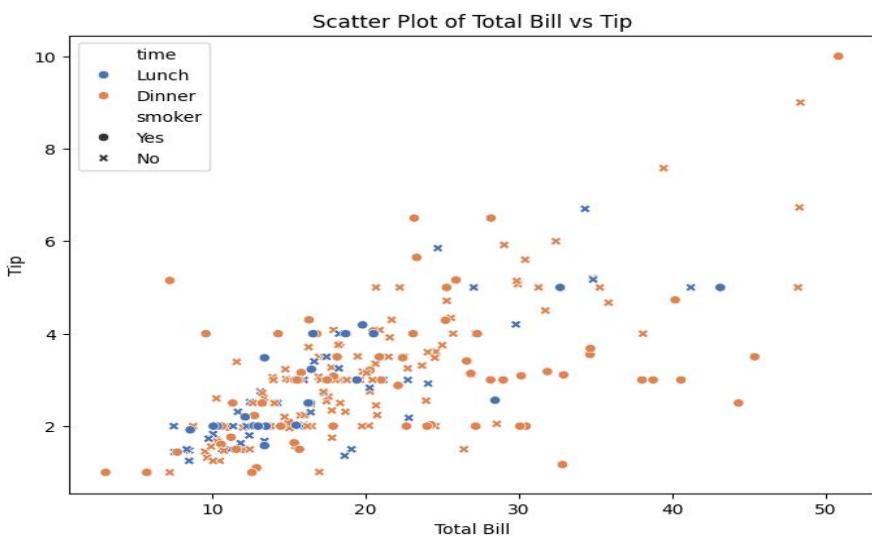
Gambar 27: Histogram Total Bill

```
# 2. Boxplot: Menampilkan distribusi dan outlier dari 'total_bill' berdasarkan hari  
plt.figure(figsize=(8, 6))  
sns.boxplot(x='day', y='total_bill', data=tips, palette='Set3')  
plt.title("Boxplot of Total Bill by Day")  
plt.xlabel("Day")  
plt.ylabel("Total Bill")  
plt.show()
```



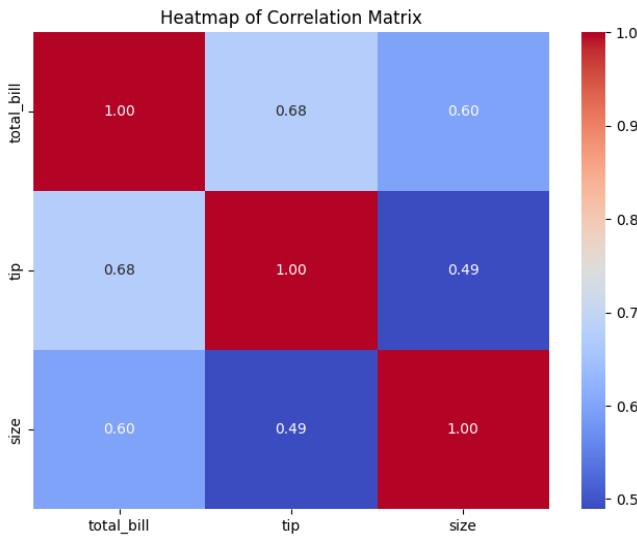
Gambar 28: Boxplot Total Bill Dikelompokkan Berdasarkan Hari

```
# 3. Scatter Plot: Menampilkan hubungan antara 'total_bill' dan 'tip'
plt.figure(figsize=(8, 6))
# Pewarnaan berdasarkan kolom 'time' dan bentuk marker berdasarkan status 'smoker'
sns.scatterplot(x='total_bill', y='tip', data=tips, hue='time', style='smoker', palette='deep')
plt.title("Scatter Plot of Total Bill vs Tip")
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.show()
```



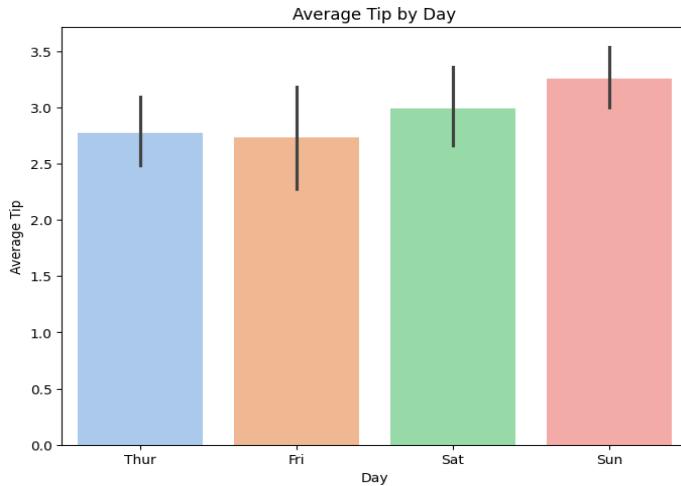
Gambar 29: Scatter Plot (Grafik Sebaran) Total Bill vs Tip

```
# 4. Heatmap: Menampilkan matriks korelasi antar variabel numerik dalam dataset
plt.figure(figsize=(8, 6))
# Hanya memilih kolom numerik untuk menghitung korelasi agar menghindari error konversi
numeric_data = tips.select_dtypes(include=[np.number])
correlation_matrix = numeric_data.corr()
# Membuat heatmap dengan anotasi nilai korelasi
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Heatmap of Correlation Matrix")
plt.show()
```



Gambar 30: Heatmap Matriks Korelasi Antara Atribut Data

```
# 5. Bar Plot: Menampilkan rata-rata tip per hari
plt.figure(figsize=(8, 6))
sns.barplot(x='day', y='tip', data=tips, palette='pastel')
plt.title("Average Tip by Day")
plt.xlabel("Day")
plt.ylabel("Average Tip")
plt.show()
```



Gambar 31: Bar Plot (Grafik Batang) Rata-Rata Tip Dikelompokkan Berdasarkan Hari

B.2.4. Exploratory Data Analysis

Exploratory Data Analysis (EDA) adalah proses awal dalam analisis data yang bertujuan untuk memahami karakteristik dataset secara menyeluruh. Dalam EDA, analis menguji hipotesis awal, mengidentifikasi pola, mendeteksi anomali, dan memeriksa asumsi melalui berbagai teknik statistik dan visualisasi. Dengan EDA, kita dapat mengungkap informasi penting yang mungkin tersembunyi di balik data mentah sebelum melakukan pemodelan lebih lanjut.

Beberapa tujuan utama dari EDA meliputi:

- Memahami Struktur Data:** Mengetahui jumlah variabel, tipe data (numerik, kategorik, tanggal), dan ukuran dataset.
- Mengidentifikasi Missing Values dan Outlier:** Mendeteksi data yang hilang atau tidak konsisten serta *outlier* yang dapat mempengaruhi hasil analisis.
- Mengeksplorasi Distribusi Data:** Melihat sebaran data, tren sentral, dan penyebaran dengan bantuan histogram, boxplot, dan density plot.
- Menguji Asumsi Awal:** Memeriksa hubungan antar variabel, seperti korelasi atau asosiasi, yang nantinya dapat digunakan untuk pemodelan.
- Mendapatkan Wawasan Awal:** Memberikan gambaran umum yang mendalam mengenai dataset sehingga dapat menentukan strategi pembersihan data, transformasi, atau pemilihan fitur untuk pemodelan.

Proses *EDA* biasanya melibatkan beberapa tahapan berikut:

a. Pengumpulan dan Pemuatan Data

Data dapat diperoleh dari berbagai sumber dan memiliki banyak alternatif format, seperti file CSV, database, atau API. Dalam tahap ini, data dimuat ke dalam lingkungan analisis menggunakan library seperti *pandas*.

b. Pembersihan Data

Pembersihan data meliputi penanganan *missing values*, duplikasi, dan kesalahan format. Data yang tidak valid harus diperbaiki atau dihapus agar analisis selanjutnya tidak terdistorsi.

c. Transformasi dan *Enrichment* Data

Transformasi dapat mencakup normalisasi, *encoding* variabel kategori, pembuatan fitur baru, dan penggabungan dataset. Tahap ini penting untuk mempersiapkan data sebelum dilakukan visualisasi.

d. Analisis Deskriptif

Melakukan analisis statistik dasar seperti perhitungan rata-rata, median, modus, standar deviasi, dan distribusi frekuensi. Teknik ini membantu memahami kecenderungan data.

e. Visualisasi Data

Visualisasi merupakan tahap kunci dalam *EDA*. Teknik visualisasi seperti histogram, boxplot, scatter plot, dan heatmap dapat membantu mengungkap adanya pola, korelasi, dan *outlier* pada data.

f. Analisis Korelasi dan Hubungan Antar Variabel

Mengukur hubungan linier antara variabel numerik, yang dapat divisualisasikan, salah satunya, dalam bentuk heatmap yang menampilkan matriks korelasi.

Berikut adalah contoh proses *exploratory data analysis (EDA)* menggunakan dataset *Titanic* dari Seaborn:

<https://colab.research.google.com/drive/1BBnmCH8oSt7lcCzMBUD7Fbb4kOQmeCsJ?usp=sharing>. Analisis ini mencakup pemuatan data, pemeriksaan struktur data, penanganan data yang hilang, analisis statistik deskriptif, serta berbagai visualisasi untuk mendapatkan wawasan mendalam tentang data.

a. Informasi Utama dan Lima Baris Pertama Dataset Titanic

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   survived    891 non-null    int64  
 1   pclass       891 non-null    int64  
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64  
 5   parch        891 non-null    int64  
 6   fare          891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class         891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool   
 11  deck          203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive         891 non-null    object  
 14  alone         891 non-null    bool  
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
survived  pclass    sex    age   sibsp  parch    fare embarked  class  \
0         0        3  male  22.0     1     0    7.2500    S  Third
1         1        1 female 38.0     1     0   71.2833    C  First
2         1        3 female 26.0     0     0    7.9250    S  Third
3         1        1 female 35.0     1     0   53.1000    S  First
4         0        3  male  35.0     0     0    8.0500    S  Third

      who  adult_male  deck  embark_town  alive  alone
0   man      True    NaN  Southampton  no  False
1 woman    False     C  Cherbourg  yes  True
2 woman    False    NaN  Southampton  yes  False
3 woman    False     C  Southampton  yes  False
4   man      True    NaN  Southampton  no  True
```

Data di atas adalah informasi utama dan cuplikan (lima baris pertama) dari **dataset Titanic** (diperoleh dari seaborn), yang sering dipakai untuk analisis dan pemodelan klasifikasi (memprediksi apakah penumpang selamat atau tidak).

Tabel 4. Penjelasan Masing-Masing Kolom Dalam Dataset Titanic

Kolom	Tipe Data	Deskripsi
survived	int (0/1)	Status selamat (0 = tidak selamat, 1 = selamat)
pclass	int (1/2/3)	Kelas kabin (1 = First, 2 = Second, 3 = Third)

class	category (str)	Sama dengan pclass dalam bentuk teks ("First", "Second", "Third")
sex	category (str)	Jenis kelamin penumpang ("male"/"female")
age	float	Usia penumpang (dalam tahun); ada nilai NaN (jika usia tidak tercatat)
sibsp	int	Jumlah saudara kandung/istri-suami yang ikut dalam kapal
parch	int	Jumlah orang tua/anak yang ikut
fare	float	Tarif (biaya) tiket yang dibayar penumpang
embarked	category (str)	Pelabuhan naik (C = Cherbourg, Q = Queenstown, S = Southampton)
embark_to wn	category (str)	Nama lengkap kota pelabuhan naik
deck	category (str)	Kode dek kabin (A–G); NaN jika tidak tercatat
who	category (str)	Klasifikasi "who" berdasarkan usia & jenis kelamin ("man", "woman", "child")
adult_male	bool	Penanda apakah penumpang adalah pria dewasa (True/False)
alive	category (str)	Sama dengan survived, tapi dalam teks ("yes"/"no")
alone	bool	True jika penumpang tidak punya keluarga (sibsp+parch = 0), False jika ikut bersama keluarga

b. Statistik Deskriptif Untuk Kolom Dengan Tipe Numerik

Statistik Deskriptif untuk Kolom Numerik:						
	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Contoh interpretasi:

- 1) Kolom survived (Status Kelangsungan Hidup: 0 = Tidak Selamat, 1 = Selamat)
 - Rata-rata (mean) = 0.384 menunjukkan bahwa sekitar 38.4% penumpang selamat.
 - Standar deviasi (std) = 0.486 cukup tinggi, yang berarti ada variasi signifikan antara penumpang yang selamat dan yang tidak.
 - Min = 0, Max = 1 menegaskan bahwa ini adalah variabel biner.
 - Kuartil:
 - 25% penumpang tidak selamat.
 - 50% (median) juga tidak selamat.
 - 75% dari penumpang termasuk kategori 0, sehingga menunjukkan bahwa sebagian besar penumpang tidak selamat.

Kesimpulan: Sebagian besar penumpang tidak selamat, tetapi hampir 40% berhasil bertahan.

- 2) Kolom pclass (Kelas Tiket: 1 = First, 2 = Second, 3 = Third)
 - Rata-rata = 2.308 menunjukkan bahwa mayoritas penumpang berada di kelas 3.
 - Standar deviasi = 0.836 cukup besar, menunjukkan adanya distribusi kelas tiket yang beragam.
 - Min = 1, Max = 3 menegaskan bahwa hanya ada tiga kategori kelas tiket.
 - Kuartil:
 - 25% penumpang berada di kelas 2 atau lebih tinggi.
 - 50% (median) berada di kelas 3.
 - 75% juga berada di kelas 3.

Kesimpulan: Sebagian besar penumpang berada di kelas ekonomi (kelas 3), yang berpengaruh terhadap tingkat kelangsungan hidup.

3) Kolom age (Usia Penumpang dalam Tahun)

- Rata-rata = 29.7 tahun, menunjukkan usia rata-rata penumpang.
- Standar deviasi = 14.53, menunjukkan adanya variasi besar dalam usia penumpang.
- Min = 0.42 tahun (sekitar 5 bulan), menunjukkan ada bayi dalam daftar penumpang.
- Max = 80 tahun, menunjukkan bahwa ada penumpang lanjut usia.
- Kuartil:
 - 25% penumpang berusia di bawah 20 tahun.
 - 50% (median) berusia 28 tahun.
 - 75% penumpang berusia di bawah 38 tahun.

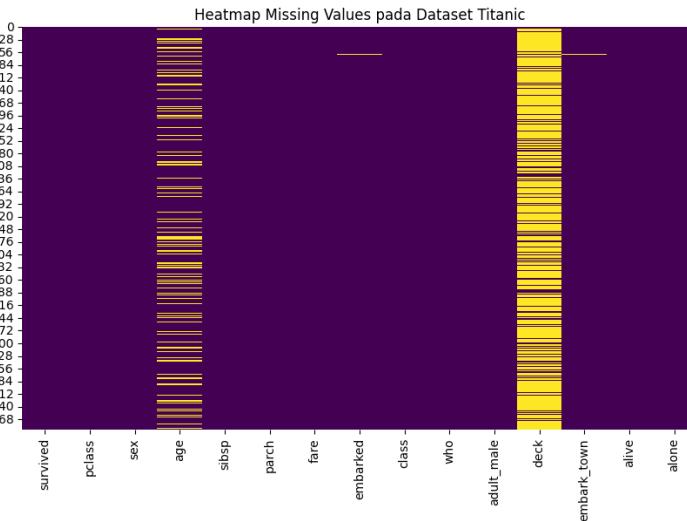
Kesimpulan: Penumpang Titanic memiliki rentang usia yang luas, dari bayi hingga lansia. Usia dapat menjadi faktor yang mempengaruhi keselamatan.

c. Data Yang Hilang (*Missing Data*)

Jumlah Missing Values per Kolom:	
survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0

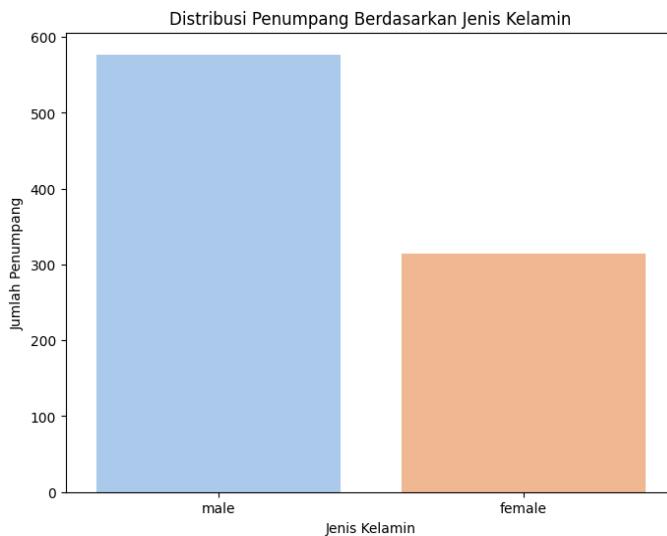
Contoh Interpretasi:

- Kolom deck memiliki banyak nilai NaN (hilang), yang menunjukkan bahwa tidak semua informasi tentang lokasi kabin penumpang tersedia. Ini bisa menjadi masalah ketika dilakukan analisis data nantinya.
- Kolom age juga memiliki beberapa missing values, yang mungkin perlu ditangani dengan imputasi (pengisian nilai yang hilang) atau pembuangan data.
- Kolom embark_town memiliki beberapa nilai NaN, meskipun tidak sebanyak deck.



Gambar 32. Heatmap Nilai Yang Hilang Pada Dataset Titanic

Heatmap pada Gambar 32 menunjukkan distribusi nilai yang hilang (*missing values*) dalam dataset Titanic. Warna kuning menunjukkan sel dengan data yang hilang, sedangkan warna ungu menunjukkan sel dengan data yang tersedia.

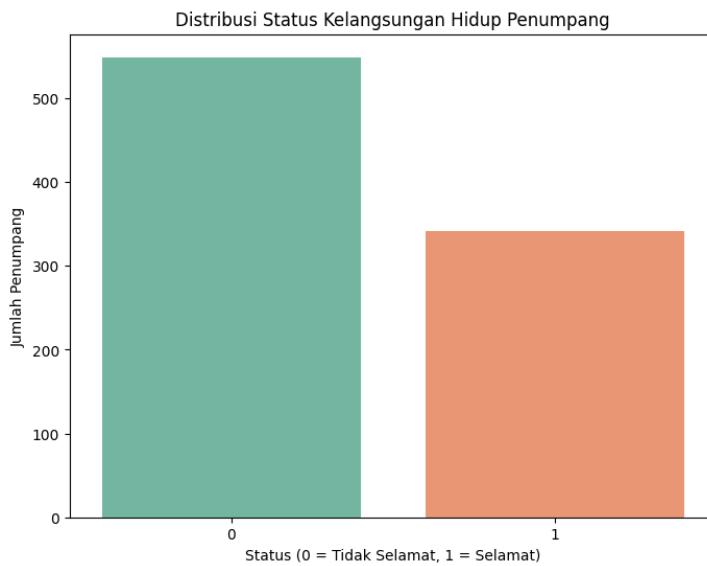


Gambar 33. Grafik Batang yang Menunjukkan Jumlah Penumpang Titanic Berdasarkan Jenis Kelamin

- Jumlah Penumpang Pria Lebih Banyak dari Wanita
 - Dari grafik, terlihat bahwa jumlah penumpang laki-laki (*male*) jauh lebih banyak dibandingkan penumpang perempuan (*female*).
 - Ini menunjukkan bahwa mayoritas penumpang Titanic adalah pria.

- b. Kemungkinan Pengaruh terhadap Tingkat Keselamatan
- Berdasarkan aturan "*Women and children first*", wanita memiliki kemungkinan lebih tinggi untuk diselamatkan dibandingkan pria.
 - Perbedaan jumlah ini bisa berpengaruh pada analisis tingkat keselamatan (*survival rate*).

e. Distribusi Status Keselamatan Penumpang



Gambar 34. Grafik Batang Distribusi Keselamatan Penumpang

Grafik batang ini menunjukkan jumlah penumpang Titanic berdasarkan status keselamatan mereka:

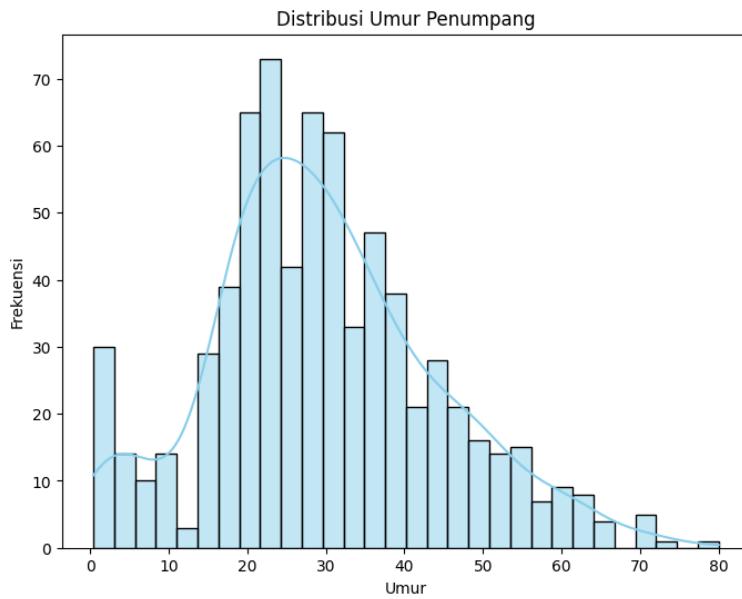
- 0 (Tidak Selamat)
- 1 (Selamat)

Interpretasi Data:

- 1) Lebih Banyak Penumpang yang Tidak Selamat
 - a) Dari grafik tersebut terlihat bahwa jumlah penumpang yang tidak selamat (kategori 0) jauh lebih banyak dibandingkan yang selamat (kategori 1).
 - b) Hal ini mengindikasikan bahwa tragedi Titanic menyebabkan lebih banyak korban jiwa dibanding mereka yang berhasil selamat.
- 2) Persentase Keselamatan Relatif Rendah

- a) Jika dibandingkan dengan total penumpang, hanya sekitar 38% yang selamat (dapat dikonfirmasi dengan statistik deskriptif sebelumnya).
- b) Ini menunjukkan bahwa faktor-faktor seperti akses ke sekoci, kelas tiket, dan jenis kelamin sangat berpengaruh terhadap kemungkinan bertahan hidup.

f. Distribusi Umur Penumpang



Gambar 35. Histogram Distribusi Umur Penumpang

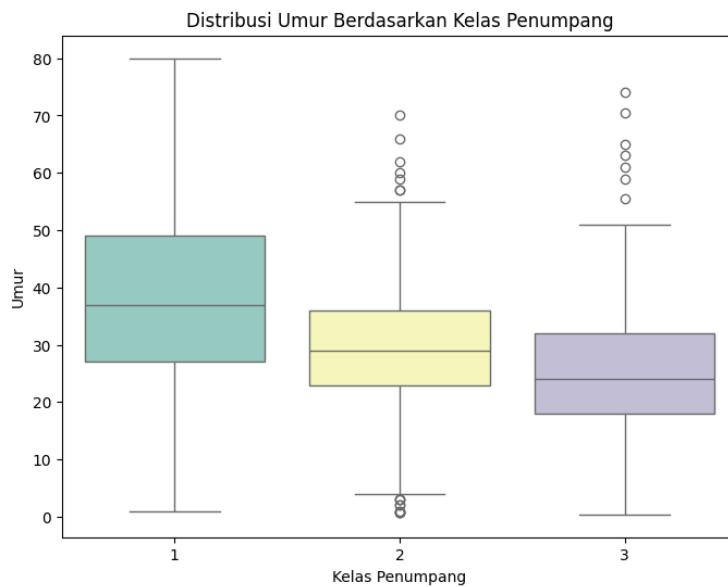
Grafik ini menunjukkan distribusi umur penumpang Titanic dalam bentuk histogram dengan ditambahkan garis kepadatan distribusi (*KDE - Kernel Density Estimation*).

- 1) Mayoritas Penumpang Berusia 20-40 Tahun
 - a) Puncak distribusi terlihat di rentang 20-30 tahun, dengan frekuensi tertinggi sekitar 70 penumpang pada kelompok usia sekitar 25 tahun.
 - b) Rentang usia 30-40 tahun juga cukup banyak, meskipun jumlahnya sedikit lebih rendah.
- 2) Keberagaman Umur Penumpang
 - a) Ada penumpang dari berbagai kelompok umur, mulai dari bayi hingga lansia sekitar 80 tahun.
 - b) Frekuensi anak-anak (di bawah 10 tahun) cukup rendah tetapi masih terlihat signifikan.
 - c) Lansia (di atas 60 tahun) memiliki jumlah yang jauh lebih sedikit dibandingkan kelompok usia lainnya.

3) Distribusi Miring ke Kanan

- Grafik menunjukkan pola distribusi miring ke kanan (right-skewed), di mana lebih banyak penumpang berusia muda dibandingkan penumpang yang lebih tua.
- Hal ini masuk akal karena Titanic kemungkinan besar membawa banyak pekerja muda dan keluarga dibandingkan lansia

g. Distribusi Umur Berdasarkan Kelas Penumpang



Gambar 36. Distribusi Umur Berdasarkan Kelas Penumpang

Grafik ini adalah *boxplot* yang menunjukkan distribusi umur penumpang Titanic berdasarkan kelas tiketnya (1, 2, dan 3).

1) Kelas 1 (*First Class*)

- Median usia sekitar 38 tahun, menunjukkan bahwa sebagian besar penumpang di kelas ini adalah orang dewasa.
- Rentang interkuartil (*IQR*) sekitar 25-50 tahun, dengan beberapa penumpang yang berusia lebih tua yang mencapai 80 tahun.
- Terdapat *outlier* di bawah usia 10 tahun, yang menunjukkan adanya anak-anak dengan jumlah sedikit.

2) Kelas 2 (*Second Class*)

- Median usia sekitar 30 tahun, sedikit lebih muda dibandingkan kelas 1.
- Rentang interkuartil sekitar 20-40 tahun.

- 
- c) Terlihat beberapa *outlier* di usia yang masih sangat muda (0-5 tahun), mengindikasikan ada lebih banyak anak-anak di kelas ini dibandingkan di kelas 1.
 - d) Ada juga beberapa *outlier lansia* berusia 60-an.

3) Kelas 3 (*Third Class*)

- a) Median usia yang berada di sekitar 24 tahun, menunjukkan bahwa penumpang di kelas ini lebih muda secara umum.
- b) Rentang interkuartil berada di sekitar 18-30 tahun, dengan banyak penumpang yang masih muda dan remaja.
- c) Banyak *outlier* usia lanjut (>50 tahun), tetapi jumlahnya lebih sedikit dibandingkan kelas 1 dan 2.
- d) Terdapat lebih banyak bayi jika dibandingkan dengan kelas lain, terlihat dari adanya nilai minimum mendekati yang 0 tahun.

4) Semakin tinggi kelas tiket, semakin tua median usia penumpang.

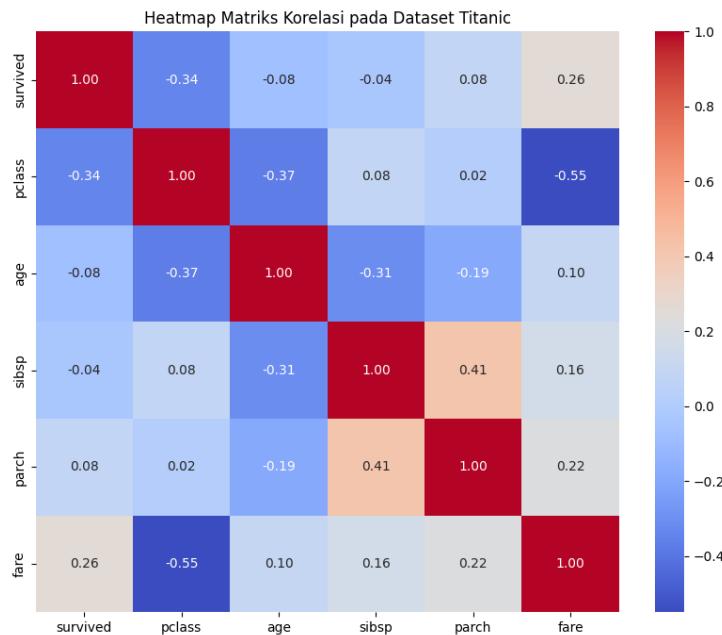
- a) Kelas 1 memiliki lebih banyak penumpang dewasa dan lansia.
- b) Kelas 3 lebih didominasi penumpang muda dan keluarga dengan anak-anak.

5) Ada lebih banyak anak-anak di kelas 2 dan 3 dibandingkan kelas 1.

- a) Hal ini mungkin terjadi karena keluarga dengan anak-anak lebih cenderung memilih tiket dengan harga lebih terjangkau.

6) Outlier di kelas 3 menunjukkan beberapa penumpang lanjut usia, meskipun jumlahnya kecil. Lansia di kelas 3 mungkin memiliki peluang keselamatan yang lebih kecil dibandingkan di kelas 1.

j. Heatmap Matriks Korelasi Antar Atribut Data



Gambar 37. Heatmap Matriks Korelasi (Keterkaitan Antar Atribut Data)

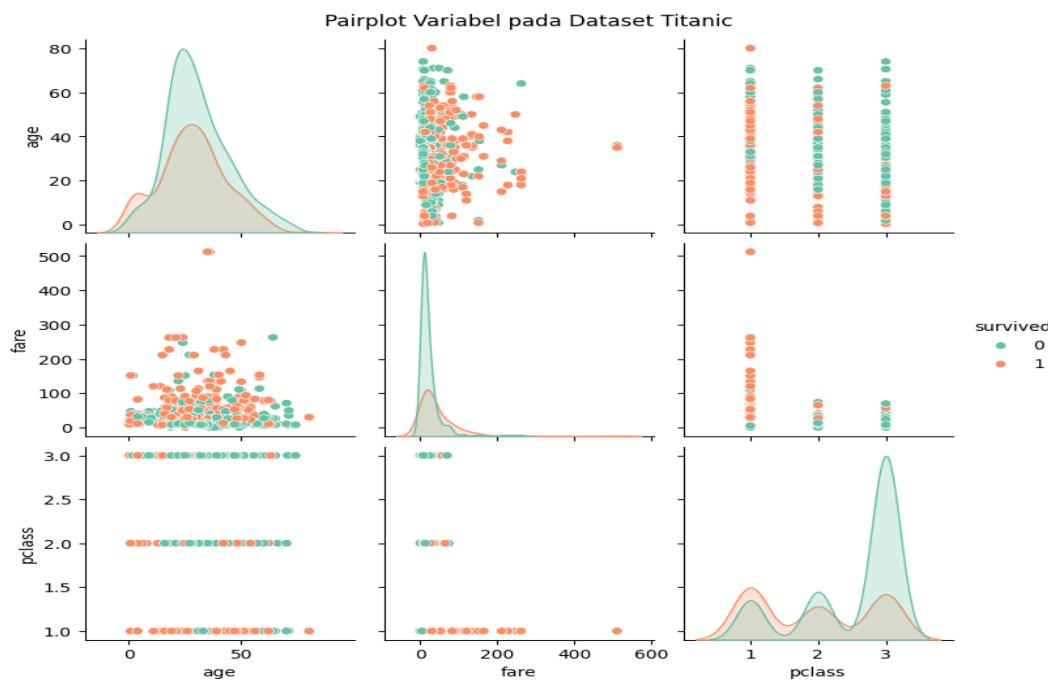
Heatmap ini menampilkan korelasi antara berbagai variabel dalam dataset Titanic. Korelasi berada pada rentang -1 hingga 1:

- Nilai positif → Hubungan searah (ketika satu variabel naik, variabel lain juga naik).
- Nilai negatif → Hubungan berlawanan arah (ketika satu variabel naik, variabel lain turun).
- Nilai mendekati 0 → Korelasi lemah atau tidak ada hubungan signifikan.

Contoh interpretasi:

- 1) Survival vs Pclass (-0.34)
 - a) Korelasi negatif menunjukkan bahwa semakin rendah kelas tiket (Pclass = 1 lebih tinggi dari 3), semakin besar peluang selamat.
 - b) Penumpang kelas 1 lebih cenderung selamat dibandingkan kelas 3.
- 2) Survival vs Fare (0.26)
 - a) Korelasi positif berarti semakin mahal harga tiket (Fare), semakin tinggi kemungkinan selamat.
 - b) Hal ini menguatkan interpretasi yang menyimpulkan bahwa penumpang kelas 1 memiliki peluang lebih besar untuk selamat.

I. Pairplot Variabel Pada Dataset Titanic



Gambar 34. Pairplot Hubungan Antara Tiga Variabel Utama dalam dataset Titanic

Pairplot ini menunjukkan hubungan antara tiga variabel utama dalam dataset Titanic:

- Age (Umur)
- Fare (Tarif Tiket)
- Pclass (Kelas Penumpang)
- Survived (Status keselamatan, dengan warna berbeda)

Contoh Interpretasi:

- 1) Distribusi Umur (Age)
 - Sebagian besar penumpang berusia antara 20-40 tahun.
 - Distribusi yang ada lebih mirip dengan bentuk distribusi normal, tetapi ada beberapa *outlier* di usia yang sangat tinggi atau rendah.
 - Tidak ada perbedaan signifikan dalam distribusi umur antara yang selamat dan tidak selamat.
- 2) Distribusi Tarif (Fare)
 - Mayoritas penumpang membayar tarif kurang dari 100, meskipun ada beberapa yang membayar hingga lebih dari 500.

- Penumpang yang selamat (orange) cenderung berasal dari kategori tarif yang lebih tinggi, menunjukkan bahwa penumpang kelas ekonomi lebih sulit bertahan hidup.

3) Distribusi Kelas (Pclass)

- Mayoritas penumpang berasal dari kelas 3, dengan jumlah yang lebih sedikit di kelas 1 dan 2.
- Penumpang kelas 1 lebih mungkin selamat, seperti yang terlihat dari jumlah titik orange lebih tinggi di kelas 1 dibandingkan kelas 3.
- Sebagian besar korban berasal dari kelas 3, menunjukkan bahwa kelas sosial sangat mempengaruhi keselamatan.

4) Hubungan Antar Variabel

- Korelasi Age vs. Fare: Tidak ada hubungan yang jelas, karena distribusinya cukup tersebar.
- Korelasi Pclass vs. Fare: Terlihat jelas bahwa kelas 1 memiliki tarif lebih tinggi dibandingkan kelas 3.
- Korelasi Age vs. Pclass: Tidak ada pola spesifik, usia penumpang cukup merata di semua kelas.

Kesimpulan utama yang dapat diperoleh dari EDA menggunakan dataset Titanic, ialah:

(1). Keselamatan di Titanic sangat dipengaruhi oleh status sosial (kelas penumpang dan tarif tiket), jenis kelamin, serta usia. (2). Wanita dan anak-anak lebih diprioritaskan dalam penyelamatan. (3). Penumpang kelas ekonomi (kelas 3) memiliki peluang selamat paling rendah.

B.2.5. Membuat Data *Training* dan *Testing*

Training set digunakan untuk melatih model agar dapat mempelajari pola dari data. **Testing set** digunakan untuk mengevaluasi model dengan data yang belum pernah dilihat oleh model sebelumnya, sehingga kita bisa menilai performa model secara objektif dan mendekripsi masalah seperti *overfitting*. Contoh kode program dalam bahasa python berikut ini digunakan membagi dataset menjadi *training set* dan *testing set*.

```
# 2. Memisahkan Data Menjadi Training dan Testing
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
print("Jumlah data training:", len(X_train))
print("Jumlah data testing :", len(X_test))
```

```
train_test_split(X, y, test_size=0.3, random_state=42)
```

- Fungsi `train_test_split` berasal dari scikit-learn dan digunakan untuk membagi data menjadi dua bagian: *training set* dan *testing set*.
- X adalah variabel yang berisi fitur (data input), dan y adalah variabel yang berisi label atau target (data output yang ingin diprediksi).
- `test_size=0.3` artinya 30% data akan digunakan sebagai data testing, sedangkan 70% sisanya akan digunakan sebagai data training.
- `random_state=42` digunakan agar pemisahan data bersifat *reproducible*—artinya setiap kali kode ini dijalankan dengan angka yang sama, pemisahan data akan selalu sama. Angka 42 sendiri hanyalah contoh, kita bisa menggunakan angka lain.

```
X_train, X_test, y_train, y_test = ...
```

- Baris ini mengambil hasil dari `train_test_split` dan menyimpannya dalam empat variabel:
 - `X_train`: Data fitur yang digunakan untuk melatih (*train*) model.
 - `X_test`: Data fitur yang digunakan untuk menguji (*test*) model.
 - `y_train`: Data label yang digunakan untuk melatih model (pasangan dari `X_train`).
 - `y_test`: Data label yang digunakan untuk menguji model (pasangan dari `X_test`).

```
print("Jumlah data training:", len(X_train))
```

- Mencetak banyak baris data yang masuk ke *training set*. Fungsi `len()` akan menghitung panjang (banyak baris) dari `X_train`.

```
print("Jumlah data testing:", len(X_test))
```

- Mencetak banyak baris data yang masuk ke *testing set*. Fungsi `len()` di sini menghitung banyak baris di `X_test`.

B.2.6. Melatih Model dan Membuat Prediksi

Membuat dan melatih model adalah proses inti dalam pengembangan sistem KA. Proses ini dimulai dengan **memilih algoritma atau pendekatan yang sesuai**, seperti *Linear Regression*, *Logistic Regression*, *Random Forest*, atau *Neural Network*, tergantung pada jenis masalah yang ingin dipecahkan. Setelah algoritma ditentukan, kita perlu **menyiapkan data yang akan digunakan**. Data tersebut umumnya dibagi menjadi dua bagian utama, yaitu training set dan testing set. Training set digunakan untuk “mengajarkan” model agar mampu mempelajari pola dari data, sementara testing set digunakan untuk mengevaluasi kemampuan model dalam memprediksi data baru yang belum pernah dilihat sebelumnya.

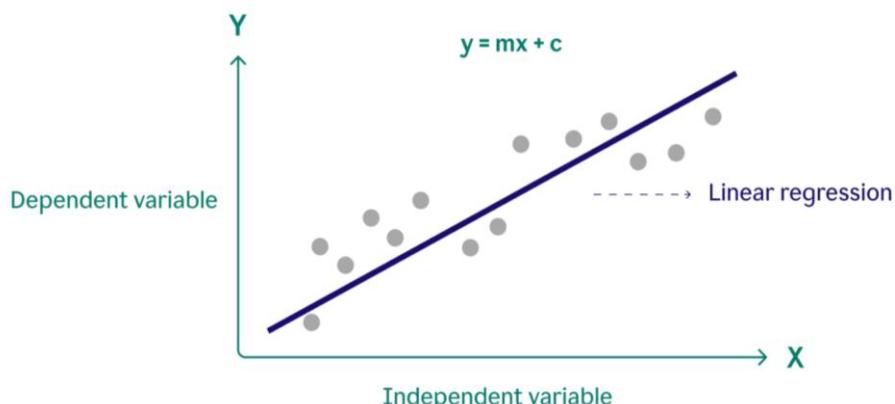
Pada tahap pembuatan model, kita perlu **mendefinisikan struktur model dan hyperparameter yang dibutuhkan**. Misalnya, jika kita memakai model *Linear Regression*, kita tidak perlu banyak menggunakan *hyperparameter*, dan cukup menentukan apakah akan menggunakan regularisasi atau tidak. Di sisi lain, jika kita memilih *Random Forest*, kita harus menentukan jumlah pohon (*n_estimators*), kedalaman maksimum (*max_depth*), dan parameter lain yang mempengaruhi kinerja model.

Setelah struktur dan parameter awal ditetapkan, maka dapat dilanjutkan ke **proses pelatihan (training)**. Proses ini melibatkan pemanggilan metode *.fit()* (atau metode serupa, tergantung *library* yang digunakan) dengan memasukkan data fitur (*X*) dan label (*y*). Kemudian, model akan menerapkan algoritma tertentu, misalnya gradient descent pada *Neural Network*, atau pembentukan pohon pada *Random Forest*, untuk menemukan pola terbaik yang mampu meminimalkan kesalahan (*error*) antara nilai prediksi dan nilai sebenarnya.

Selama proses pelatihan, model akan **menyesuaikan bobot (weights)** atau parameter internalnya agar menjadi seakurat mungkin. Pada akhir proses ini, model akan memiliki sekumpulan parameter yang optimal yang diyakini dapat mewakili pola dalam *data training*. Kemudian, kita dapat **menguji performa model menggunakan data testing** untuk melihat apakah model dapat melakukan generalisasi dengan baik. Jika hasilnya belum memuaskan, kita bisa melakukan penyesuaian (*tuning*) pada *hyperparameter*, menambah data, atau mengubah arsitektur model. Berikut ini adalah beberapa contoh proses melatih dan membuat prediksi dengan menggunakan beberapa model:

1) Regresi Linear (Contoh: Prediksi Diabetes)

Regresi linier adalah metode statistik yang digunakan untuk **mencari hubungan linear antara satu variabel independen (fitur) dengan variabel dependen (target)**. Dalam *data science*, regresi linier merupakan teknik dasar yang sering digunakan untuk **memprediksi nilai-nilai berdasarkan data yang sudah ada**



Gambar 35. Regresi Linier.

Sumber: <https://www.grammarly.com/blog/ai/what-is-linear-regression/>

Contoh prediksi *disease progression* pada penderita diabetes berdasarkan BMI menggunakan regresi linear dapat dilihat pada tautan:
<https://colab.research.google.com/drive/1V9ZzDDNcsGprbLsGGGu3GnCmK6Pm-r1?usp=sharing>

```
# 4. Membuat dan Melatih Model Linear Regression
# -----
model = LinearRegression()
model.fit(X_train, y_train)
```

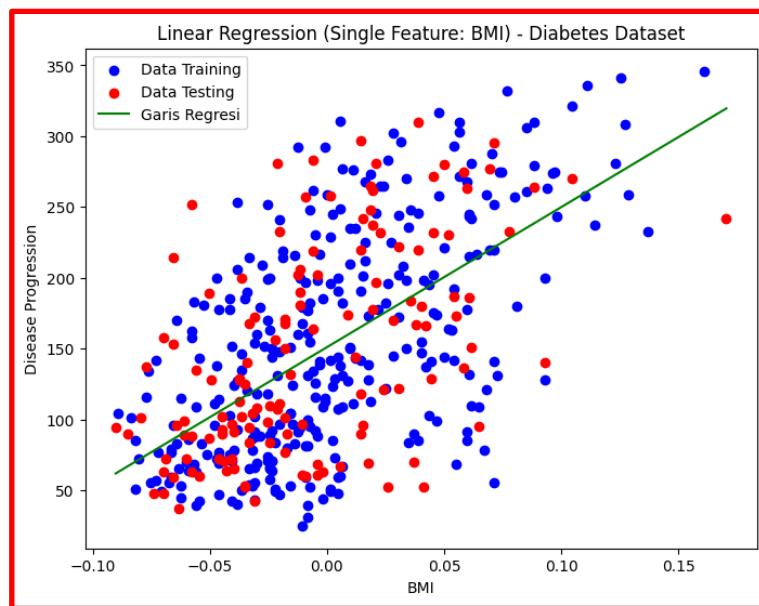
`model = LinearRegression()`

- Baris ini membuat sebuah *instance* atau objek dari kelas `LinearRegression` yang disediakan oleh scikit-learn.
- Model `LinearRegression` akan mencoba menemukan hubungan linear antara data fitur (X) dan label (y)

`model.fit(X_train, y_train)`

- Metode `.fit()` menandakan proses *training* atau pelatihan model menggunakan data training.
- Parameter `X_train` berisi nilai-nilai fitur (input) dan `y_train` berisi nilai target (output) yang sebenarnya.

- Model akan menghitung koefisien (*slope*) dan *intercept* (*bias*) terbaik yang meminimalkan kesalahan (*error*) antara nilai prediksi dan nilai sebenarnya pada data training.

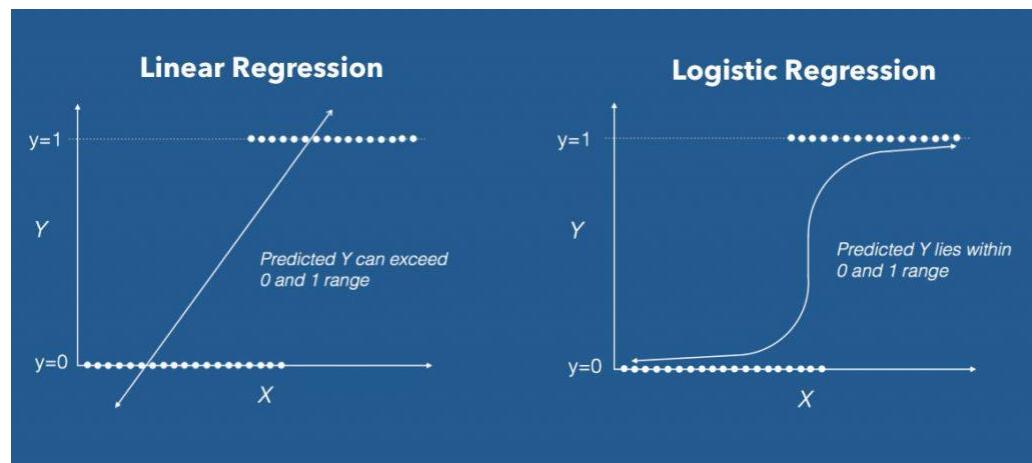


Gambar 40. Hubungan Antara BMI dan *Diabetes Progression*

Garis regresi hijau pada Gambar 40 memperlihatkan bahwa saat BMI naik, skor perkembangan penyakit pada pasien diabetes cenderung juga naik. Namun, titik-titik biru (data pelatihan) dan merah (data pengujian) tersebar cukup luas di sekitar garis tersebut. Ini berarti **BMI saja belum cukup menjelaskan perkembangan penyakit secara akurat**. Untuk membuat prediksi yang lebih baik, perlu ditambahkan variabel klinis lain. misalnya umur, tekanan darah, atau kadar glukosa, atau mengubah model menjadi non-linier.

2) Klasifikasi dengan Regresi Logistik (Klasifikasi Data Cancer)

Regresi logistik adalah teknik statistik yang digunakan untuk **memprediksi probabilitas kejadian kategori tertentu (misalnya sukses/gagal)** dengan mengubah **output model melalui fungsi logistik sehingga menghasilkan nilai antara 0 dan 1**, berbeda dengan regresi linier yang memprediksi nilai kontinu berdasarkan hubungan linear antar variabel. Regresi linier cocok digunakan untuk masalah prediksi angka (numerik) dengan asumsi hubungan yang bersifat linear, regresi logistik lebih tepat untuk klasifikasi karena mengakomodasi output berupa peluang dengan batasan nilai antara 0 dan 1.



Gambar 41: Regresi Logistik. Sumber:
<https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/>

Contoh klasifikasi kanker payudara dengan logistic regression dapat dilihat pada tautan: https://colab.research.google.com/drive/1pk7vycijnjHCn6M5KrXGh6Do5v0_MGZ?usp=sharing

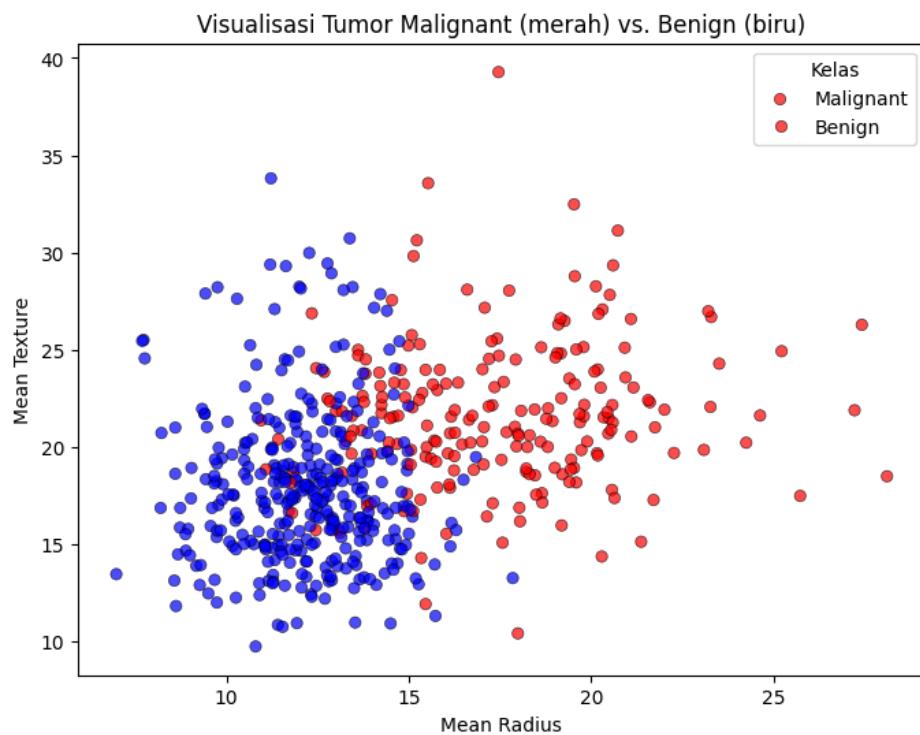
```
# 3. Melatih Model Logistic Regression
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

`clf = LogisticRegression()`

- Baris ini membuat sebuah objek *Logistic Regression* dengan parameter *default*.
- *Logistic Regression* adalah algoritma klasifikasi yang populer, sering digunakan untuk memprediksi probabilitas terjadinya suatu kejadian (misalnya memprediksi apakah email termasuk *spam* atau bukan).

`clf.fit(X_train, y_train)`

- Metode `.fit()` menandakan proses pelatihan (*training*) model.
- `X_train` berisi data fitur (*input*) yang telah dipisahkan untuk keperluan *training*, sedangkan `y_train` berisi label (*output*) yang sebenarnya.
- Model akan mempelajari hubungan antara fitur pada `X_train` dan label pada `y_train` dengan cara memaksimalkan *likelihood* atau meminimalkan error. Hasil akhirnya adalah model dengan parameter optimal yang dapat memprediksi label untuk data baru.

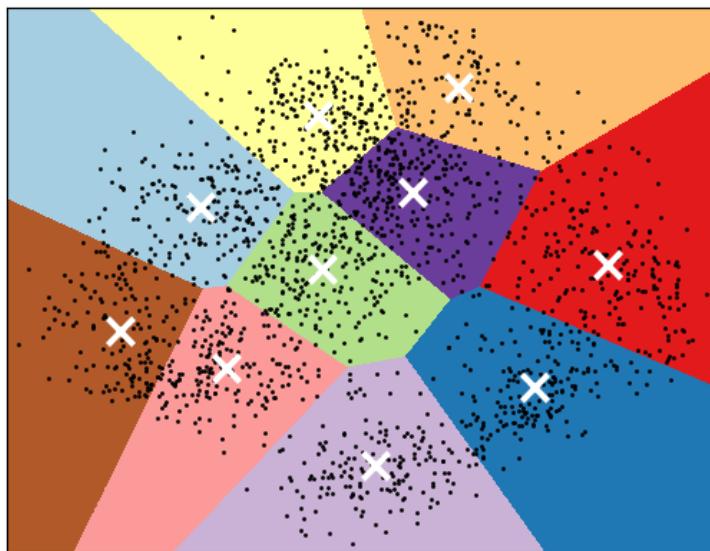


Gambar 42. Hasil Klasifikasi Kanker Payudara (*Malignant vs Benign*)
Berdasarkan Tekstur dan Radius

3) KMeans Clustering (Klasterisasi Data Wine)

K-means clustering adalah algoritma *unsupervised learning* yang **mengelompokkan data ke dalam sejumlah k kluster berdasarkan kesamaan fitur**, dengan cara menginisiasi sebanyak k titik pusat (*centroid*) secara acak, kemudian menetapkan setiap data ke kluster dengan *centroid* terdekat, dan akhirnya memperbarui posisi *centroid* berdasarkan rata-rata data dalam kluster tersebut hingga tercapai kondisi konvergen.

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross



Gambar 37. K-Means Clustering. Sumber: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

Contoh klasterisasi wine cultivar menggunakan algoritma KMeans dapat dilihat pada tautan:<https://colab.research.google.com/drive/1ATmldTpshocnxqfqZ9wC3QMw6vdREHcH?usp=sharing>

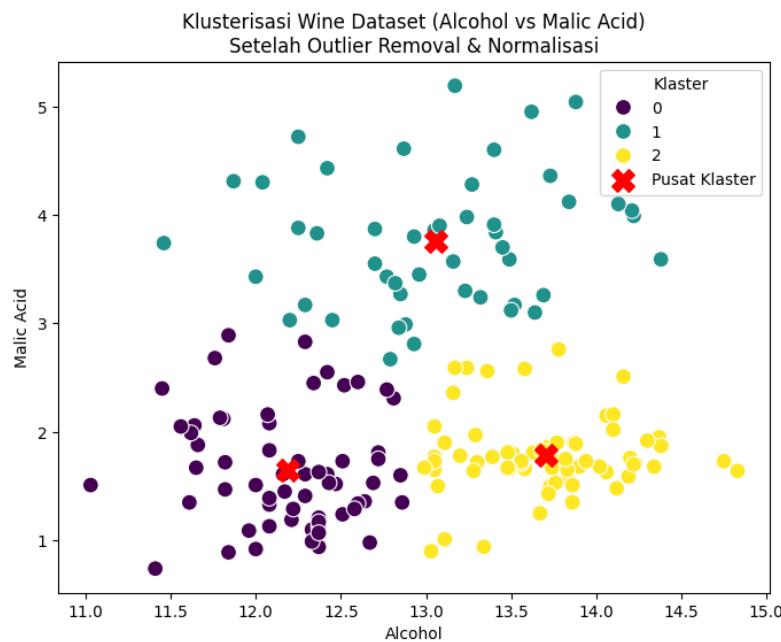
```
# 2. Melakukan Klusterisasi dengan KMeans (n_clusters=3)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
```

- Baris ini membuat objek KMeans dengan jumlah klaster yang ditentukan (`n_clusters=3`). Artinya, data akan dikelompokkan menjadi 3 klaster.
- Parameter `random_state=42` digunakan untuk mengatur seed pengacakan, sehingga hasil klusterisasi bisa direproduksi (menghasilkan nilai acak yang sama setiap kali kode dijalankan).

```
clusters = kmeans.fit_predict(X)
```

- Metode `.fit_predict(X)` melakukan dua hal sekaligus:
 - `fit`: Menyesuaikan model KMeans dengan data X (melakukan proses inisialisasi pusat klaster, perhitungan jarak, pemutakhiran posisi klaster, dan sebagainya).
 - `predict`: Menentukan label klaster untuk setiap baris data di X.
- Hasilnya disimpan di variabel `clusters`, yang berisi himpunan label (misalnya 0, 1, 2) untuk tiap baris pada X, sesuai dengan klaster tempat baris tersebut tergabung.



Gambar 44. Hasil Klusterisasi Wine Cultivar Menjadi 3 Klaster Berdasarkan Kandungan Alkohol dan Malic Acid

B.2.1. Penerapan *Library* Kecerdasan Artifisial

Di era digital saat ini, *machine learning* dan deep learning telah menjadi komponen penting dalam pengembangan aplikasi dan penelitian. Dua *library* Python yang banyak digunakan dalam bidang ini adalah **TensorFlow** dan **scikit-learn**. Meskipun keduanya digunakan untuk *machine learning*, masing-masing memiliki fokus dan keunggulan tersendiri.

1) TensorFlow

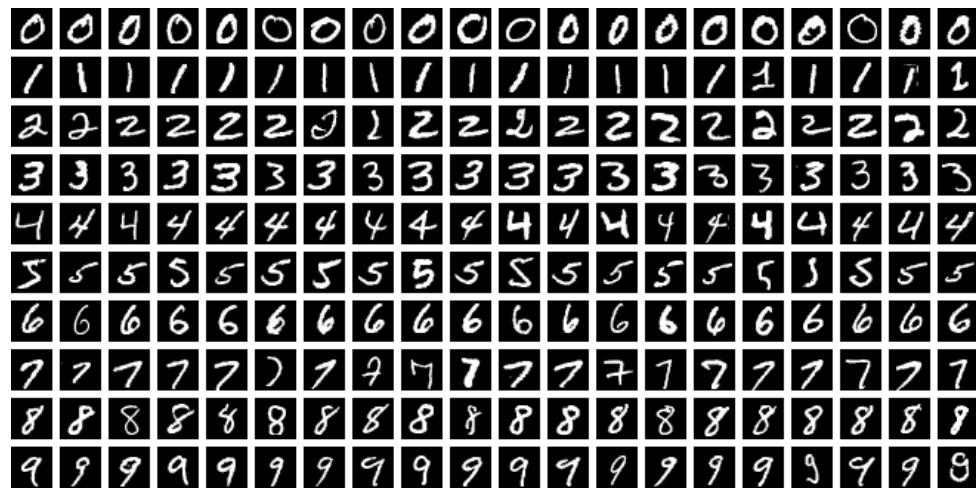


Gambar 45. Logo TensorFlow. Sumber: Wikipedia

TensorFlow adalah sebuah *library open-source* yang dikembangkan oleh Google untuk keperluan komputasi numerik dan *machine learning*, terutama *deep learning*. *Library* ini memungkinkan pengguna untuk membuat, melatih, dan mengoptimalkan model-model *neural network* secara efisien. Beberapa fitur TensorFlow meliputi:

- Computational Graph: TensorFlow menggunakan struktur grafik untuk merepresentasikan operasi matematis. Setiap *node* dalam grafik tersebut adalah operasi, sedangkan tepi antar *node* mewakili *tensor* (data multidimensi) yang mengalir di antara operasi tersebut.
- Eksekusi Skalabel: TensorFlow didesain untuk menjalankan perhitungan di berbagai *platform*, mulai dari CPU, GPU, hingga TPU (*Tensor Processing Unit*).
- Modular dan Fleksibel: Pengguna dapat membuat model dari yang sederhana hingga yang sangat kompleks dengan menyesuaikan lapisan-lapisan *neural network*, fungsi aktivasi, *loss function*, dan *optimizer* sesuai kebutuhan.

Contoh berikut membuat model *neural network* sederhana untuk mengklasifikasikan dataset MNIST. Basis data MNIST (*Modified National Institute of Standards and Technology database*) adalah basis data besar digit tulisan tangan yang umumnya digunakan untuk melatih berbagai sistem pemrosesan gambar (Wikipedia).



Gambar 39: Sample Data Tulisan Tangan MNIST. Sumber: Wikipedia

Berikut ini adalah contoh penerapan *library* TensorFlow dalam bahasa Python
<https://colab.research.google.com/drive/1yvBhw14D8MyAuuLwE8o57pW4QsoofEr9?usp=sharing>):

Pada contoh di atas, TensorFlow dengan API Keras dimanfaatkan untuk:

- Memuat dan menyiapkan data MNIST: Data dinormalisasi untuk mempercepat proses pelatihan.
- Membangun model *neural network*: Model terdiri dari lapisan Flatten, Dense, dan Dropout untuk mencegah *overfitting*.
- Melatih dan mengevaluasi model: Model dilatih dengan 5 epoch dan akurasinya diukur menggunakan *data testing*.

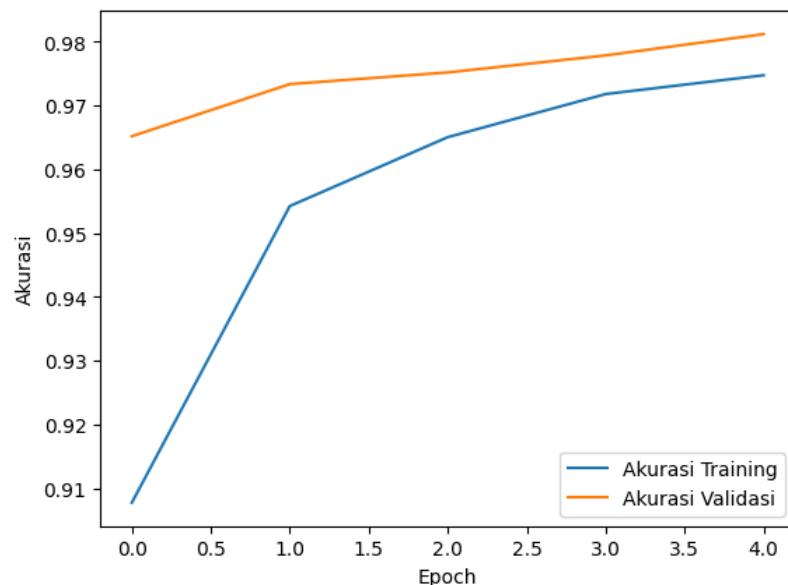
```

Epoch 1/5
1688/1688 12s 6ms/step - accuracy: 0.8530 - loss: 0.5117 - val_accuracy: 0.9652 - val_loss: 0.1346
Epoch 2/5
1688/1688 16s 4ms/step - accuracy: 0.9508 - loss: 0.1620 - val_accuracy: 0.9733 - val_loss: 0.0917
Epoch 3/5
1688/1688 8s 5ms/step - accuracy: 0.9631 - loss: 0.1210 - val_accuracy: 0.9752 - val_loss: 0.0824
Epoch 4/5
1688/1688 6s 4ms/step - accuracy: 0.9718 - loss: 0.0928 - val_accuracy: 0.9778 - val_loss: 0.0804
Epoch 5/5
1688/1688 10s 4ms/step - accuracy: 0.9761 - loss: 0.0768 - val_accuracy: 0.9812 - val_loss: 0.0668
313/313 - 1s - 3ms/step - accuracy: 0.9775 - loss: 0.0726

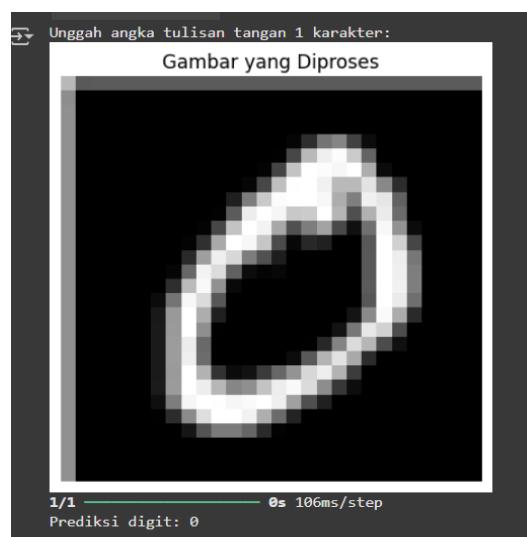
```

Gambar 47: Melatih Model Neural Network

Bayangkan Anda sedang belajar mengenali bentuk hewan menggunakan kartu *flash* (*flashcards*). Satu **epoch** itu ibarat Anda sudah melihat dan mempraktikkan **seluruh tumpukan kartu** satu kali putaran: Anda melihat kartu satu per satu, menjawab pertanyaan setiap kartu, lalu memeriksa jawabannya hingga kartu terakhir. Setelah itu, Anda mengulang lagi (*epoch* berikutnya), dengan memperbaiki jawaban berdasarkan kesalahan di putaran pertama. Dengan setiap *epoch*, “pemahaman” model *neural network* (atau, dalam analogi ini, Anda) menjadi semakin baik.



Gambar 48: Evaluasi Akurasi Setelah 8 Epoch



Gambar 49. Contoh Hasil Prediksi Digit (angka 0)

2) Scikit-learn



Gambar 50. Logo Sci-kit Learn. Sumber: Wikipedia

scikit-learn adalah *library machine learning* yang sangat populer di kalangan praktisi *data science*. *Library* ini menyediakan berbagai algoritma untuk berbagai algoritma seperti klasifikasi, regresi, *clustering*, dan reduksi dimensi. Kelebihan scikit-learn meliputi:

- Antarmuka yang Konsisten: Hampir semua fungsi dalam scikit-learn mengikuti pola “*fit/predict*”, sehingga mudah dipahami dan digunakan.
- Integrasi dengan Ekosistem Python: scikit-learn dibangun di atas NumPy, SciPy, dan matplotlib, sehingga cocok untuk analisis data secara menyeluruh.
- Dokumentasi yang Cukup Lengkap dan Komunitas yang Besar: Banyak tutorial, contoh, dan dukungan komunitas yang memudahkan pembelajaran dan penerapan *library* scikit-learn.

Scikit-Learn dapat digunakan untuk membuat model *Logistic Regression* yang dapat memprediksi apakah penumpang Titanic selamat atau tidak berdasarkan dataset yang telah dibahas pada bagian *Exploratory Data Analysis*.

```
==== Input Data Penumpang untuk Prediksi ====
Masukkan kelas penumpang (1, 2, atau 3): 2
Masukkan jenis kelamin (male/female): female
Masukkan usia penumpang: 10
Masukkan tarif tiket: 4055

==== Hasil Prediksi ====
Probabilitas Selamat: 1.00
Hasil Prediksi: Penumpang diprediksi SELAMAT.
```

```
==> Input Data Penumpang untuk Prediksi ==  
Masukkan kelas penumpang (1, 2, atau 3): 3  
Masukkan jenis kelamin (male/female): male  
Masukkan usia penumpang: 40  
Masukkan tarif tiket: 70  
  
==> Hasil Prediksi ==  
Probabilitas Selamat: 0.08  
Hasil Prediksi: Penumpang diprediksi TIDAK SELAMAT.
```

Gambar 51: Prediksi Keselamatan Penumpang TITANIC Menggunakan Logistic Regression

Kode program yang digunakan untuk prediksi dengan Logistic Regression dapat diakses melalui:

<https://colab.research.google.com/drive/1CvfOEhF3g3ipnTjasJuzrKsNGOUejNUR?usp=sharing>

Contoh lainnya adalah menggunakan **scikit-learn** untuk melakukan klasifikasi pada dataset Iris yang terkenal menggunakan **Support Vector Machine (SVM)**.

Kode program yang lengkap dapat dilihat melalui:

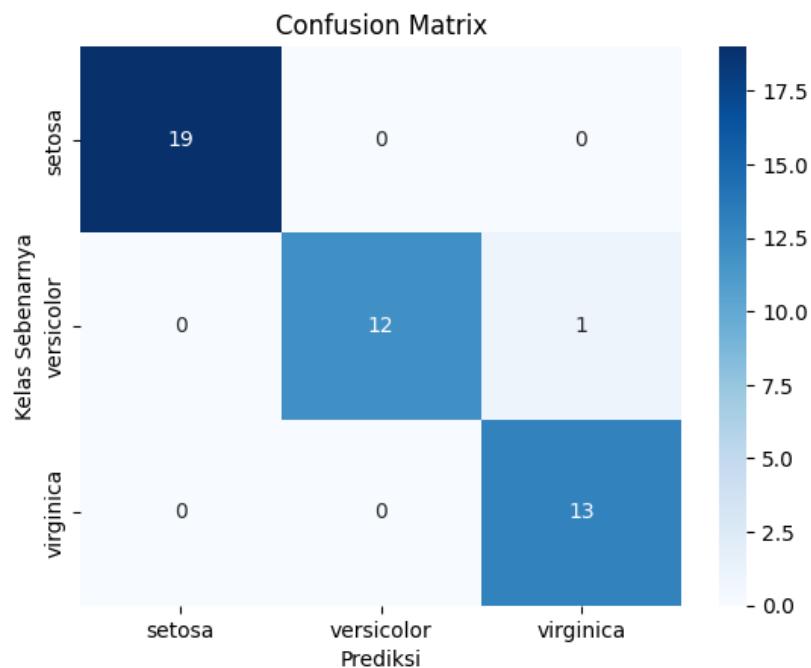
<https://colab.research.google.com/drive/1SM7LTKJ8KPr1ziwyquQzBB3jdUwNpC62?usp=sharing>

Dalam contoh klasifikasi data menggunakan scikit-learn di atas, langkah-langkah yang dilakukan antara lain:

- Memuat dataset Iris: Dataset diambil secara langsung dari *library* scikit-learn.
- Pembagian data: Data dibagi menjadi *training set* dan *testing set*
- Standarisasi: Fitur pada data distandarisasi untuk menghilangkan perbedaan skala yang dapat mempengaruhi kinerja model.
- Pembuatan dan pelatihan model SVM: Penggunaan model Support Vector Machine dengan kernel *linear* untuk klasifikasi.
- Evaluasi: Hasil prediksi dievaluasi menggunakan laporan klasifikasi dan confusion matrix yang divisualisasikan dengan *library* seaborn.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	0.92	0.96	13
2	0.93	1.00	0.96	13
accuracy			0.98	45
macro avg	0.98	0.97	0.97	45
weighted avg	0.98	0.98	0.98	45

Gambar 52. Hasil Evaluasi Model SVM



Gambar 53: Evaluasi Hasil Prediksi Bunga Iris Menggunakan Confussion Matrix

Menurut hasil evaluasi pada Gambar 52 dan 53, dapat dijelaskan bahwa:

- **Akurasi keseluruhan (accuracy = 0.98)**
Model benar mengklasifikasikan 98 % (44 dari 45) sampel data uji
- **Per-kelas**
 - **Kelas 0 (bunga iris setosa):**
 - Precision = 1.00 → Semua prediksi “setosa” benar (0 *false positive*).
 - Recall = 1.00 → Semua sampel setosa berhasil ditemukan (0 *false negative*).

- F1-score = 1.00 → model sama sekali tidak salah dan tidak melewatkkan sampel
- **Kelas 1** (bunga iris versicolor):
 - Precision = 1.00 → Tiap kali model menebak versicolor, selalu benar.
 - Recall = 0.92 → Dari 13 sampel versicolor, 12 terdeteksi; 1 terlewat (*false negative*).
 - F1-score = 0.96 → Menunjukkan keseimbangan yang bagus meski recall sedikit turun.
- **Kelas 2** (bunga iris virginica):
 - Precision = 0.93 → Dari 14 tebakan virginica, 13 benar; 1 sampel lain keliru dikategorikan virginica (*false positive*).
 - Recall = 1.00 → Semua sampel virginica (13) berhasil diidentifikasi.
 - F1-score = 0.96 → Kombinasi *precision* dan *recall* yang sangat baik.
- **Rata-rata (Macro & Weighted)**
 - Rata-rata makro (semua kelas dihitung sama): precision = 0.98, recall = 0.97, f1 = 0.97 → Nilainya hampir sama di semua kelas, artinya model konsisten untuk ketiga jenis bunga.
 - Rata-rata berbobot (memperhitungkan jumlah sampel tiap kelas): precision = 0.98, recall = 0.98, f1 = 0.98 → Secara keseluruhan model ini sangat akurat, karena mendapat skor tinggi di gabungan semua data.
- **Interpretasi Confusion Matrix**
 - Ada **1 kesalahan** prediksi:
 - **1 sampel versicolor** dikira virginica (menyebabkan recall versicolor turun ke 0.92).

Sebaliknya, model pernah menebak “virginica” untuk sampel versicolor, sehingga precision virginica menjadi 0.93.

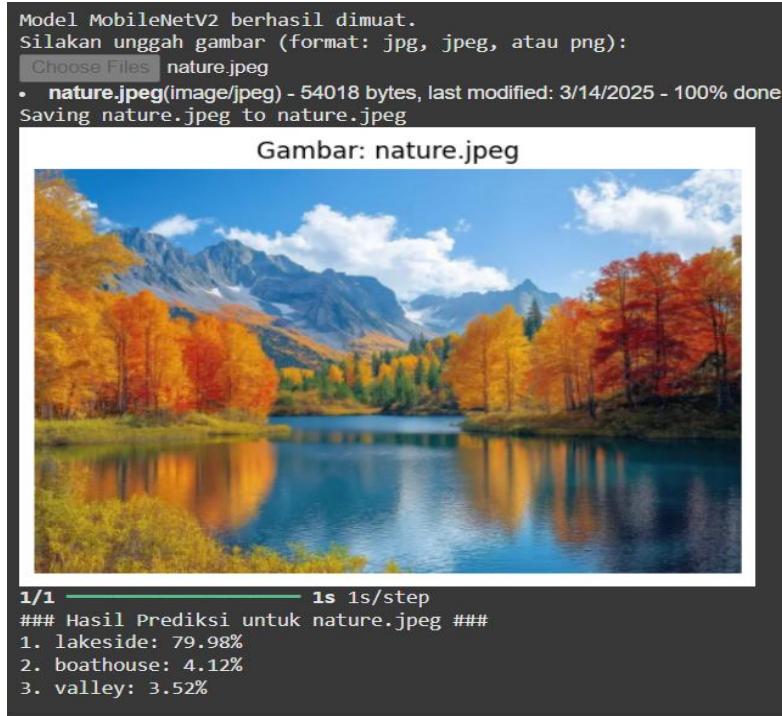
AKTIVITAS 2:

Berlatih Menerapkan *Library* dalam Pemrograman Kecerdasan Artifisial - Membuat Deteksi Objek Menggunakan TensorFlow

Pada bagian ini, akan diuraikan langkah demi langkah proses pembuatan deteksi objek dalam gambar dengan menggunakan TensorFlow. Aplikasi KA ini menerima inputan berupa gambar, dan melakukan deteksi obyek-obyek yang berada di dalam gambar dengan mencantumkan nama objek dan persentasenya. Kode program lengkap dapat diakses melalui:

https://colab.research.google.com/drive/1N6tFaertZbCT_Amvdr2nku1uliS3E3rT?usp=sharing





Gambar 41: Hasil Deteksi Object Menggunakan TensorFlow

a) Mengimpor library yang dibutuhkan

```
# Import library yang diperlukan
import tensorflow as tf          # Untuk model dan prediksi
import numpy as np               # Untuk operasi numerik
from PIL import Image, ImageOps   # Untuk memproses gambar
import matplotlib.pyplot as plt  # Untuk menampilkan gambar
from google.colab import files   # Untuk mengunggah file di Google Colab
```

b) Membuat Fungsi Untuk Memuat Model:

```
# Fungsi untuk memuat model MobileNetV2 dengan bobot pre-trained dari ImageNet
def load_model():
    model = tf.keras.applications.MobileNetV2(weights='imagenet')
    return model
```

- Fungsi `load_model()` memuat model MobileNetV2 yang sudah dilatih pada dataset ImageNet. Model ini siap digunakan untuk mengenali ribuan objek.

c) Preprocessing Gambar:

```
# Fungsi untuk melakukan preprocessing gambar
def preprocess_image(image):
    # Ubah ukuran gambar menjadi 224x224 piksel sesuai dengan input model MobileNetV2
    image = image.resize((224, 224))
    # Ubah gambar menjadi array numpy
    image_array = np.array(image)
    # Pastikan gambar memiliki 3 channel (RGB); jika ada channel alpha (RGBA), buang channel tersebut
    if image_array.shape[-1] == 4:
        image_array = image_array[..., :3]
    # Lakukan preprocessing sesuai dengan MobileNetV2
    processed_image = tf.keras.applications.mobilenet_v2.preprocess_input(image_array)
    # Tambahkan dimensi batch: dari (224,224,3) menjadi (1,224,224,3)
    processed_image = np.expand_dims(processed_image, axis=0)
    return processed_image
```

- Fungsi `preprocess_image()` mengubah ukuran gambar ke 224x224 piksel, mengonversinya ke array NumPy, dan melakukan preprocessing sesuai dengan yang dibutuhkan oleh MobileNetV2.
- Jika gambar memiliki channel alpha (RGBA), channel keempat dihapus agar hanya tersisa 3 channel (RGB).

d) Menggunakan Model dan Unggah Gambar:

```
# Memuat model
model = load_model()
print("Model MobileNetV2 berhasil dimuat.")

# Mengunggah gambar menggunakan widget Google Colab
print("Silakan unggah gambar (format: jpg, jpeg, atau png):")
uploaded = files.upload()
```

- Menggunakan `files.upload()` dari library `google.colab`, pengguna dapat mengunggah satu atau lebih file gambar.
- Setelah diunggah, setiap gambar akan dibaca dan ditampilkan menggunakan Matplotlib.

e) Prediksi dan Output:

```
# Proses setiap file yang diunggah
for filename in uploaded.keys():
    # Membaca gambar dengan PIL
    image = Image.open(filename)

    # Menampilkan gambar yang diunggah
    plt.figure(figsize=(6, 6))
    plt.imshow(image)
    plt.title(f"Gambar: {filename}")
    plt.axis('off')
    plt.show()

    # Melakukan preprocessing gambar
    processed_image = preprocess_image(image)

    # Melakukan prediksi menggunakan model
    predictions = model.predict(processed_image)

    # Mendekode prediksi untuk mendapatkan label dan probabilitas (top 3)
    decoded_predictions = tf.keras.applications.mobilenet_v2.decode_predictions(predictions, top=3)[0]

    # Menampilkan hasil prediksi
    print(f"### Hasil Prediksi untuk {filename} ###")
    for i, (imagenetID, label, prob) in enumerate(decoded_predictions):
        print(f"{i+1}. {label}: {prob*100:.2f}%")
```

- o Model melakukan prediksi pada gambar yang telah diproses.
- o Fungsi decode_predictions() mengembalikan label dan probabilitas untuk tiga prediksi teratas.
- o Hasil prediksi ditampilkan di konsol, menunjukkan label objek dan probabilitasnya.

B.2.2. Analisis Hasil dan Penyempurnaan Output Aplikasi Kecerdasan Artifisial

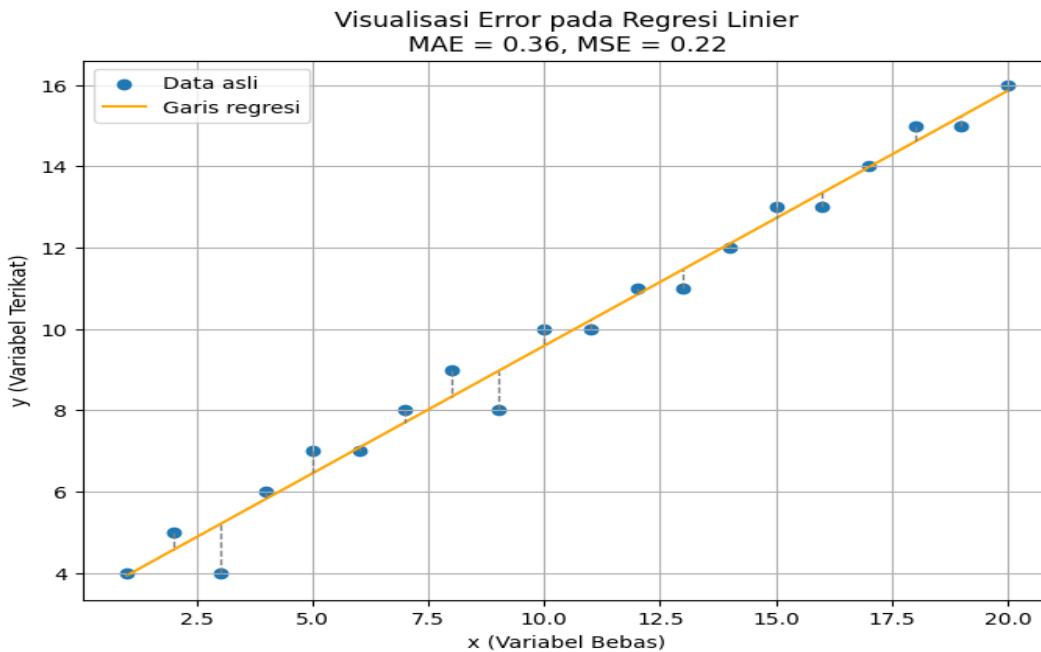
Evaluasi model *machine learning* merupakan tahap penting dalam pengembangan sistem prediktif. Setelah sebuah model dilatih, kita perlu mengukur performanya untuk memastikan bahwa model tersebut dapat bekerja dengan baik di dunia nyata. Evaluasi tidak hanya dilakukan dengan melihat akurasi, melainkan juga melalui berbagai metrik seperti *confusion matrix*, *precision*, *recall*, *F1 score*, dan *ROC curve*. Dengan menganalisis hasil dan *output*, kita dapat mengidentifikasi area yang perlu diperbaiki, menghindari *overfitting*, serta memastikan model tidak memiliki bias.

1) Evaluasi Model Regresi

Regresi linier adalah “garis terbaik” yang memodelkan hubungan lurus antara dua variabel. Nilai koefisien regresi yang optimal adalah yang menghasilkan error minimal. Jenis error pada regresi linier misalnya:

- MAE (*Mean Absolute Error*): Rata-rata jarak tiap titik data ke garis prediksi, diukur persis seperti kita mengukur panjang dengan penggaris lalu dijumlahkan dan dibagi banyak titik.
- MSE (*Mean Squared Error*) / Rata-rata jarak kuadrat: untuk tiap titik, jaraknya dikuadratkan ($\text{jarak} \times \text{jarak}$) baru dijumlahkan dan dirata-ratakan.

Implementasi evaluasi model regresi menggunakan MAE dan MSE dapat dilihat pada tautan
<https://colab.research.google.com/drive/1mdxRzWGP1p59opvAFwlXlumwLlmO4h9p?usp=sharing>



Gambar 55. Visualisasi Error Pada Regresi Linier. **Garis putus-putus vertikal** merupakan residual (selisih antara nilai y sebenarnya dengan nilai y hasil prediksi).

Semakin panjang garis residual, maka semakin besar kesalahan prediksi.

Untuk menurunkan nilai *Mean Squared Error (MSE)* dan *Root Mean Squared Error (RMSE)*, langkah pertama adalah meninjau kembali kualitas data dan kesesuaian model. Pastikan data bebas dari kesalahan pencatatan, nilai hilang, serta penculan ekstrem yang dapat “menggelembungkan” galat. Selanjutnya, lakukan rekayasa fitur, misalnya menambah variabel turunan, interaksi, atau transformasi log/polinomial bila hubungan antara variabel bebas dan terikat tidak sepenuhnya linear. Praktik ini meningkatkan kemampuan model untuk menangkap pola riil dalam data tanpa harus berpindah ke algoritma yang lebih kompleks. MSE dan RMSE yang tinggi juga dapat dihasilkan oleh dua variabel yang tidak ada hubungannya, sehingga sebaran data tidak membentuk pola linier dengan kemiringan yang jelas. Dalam hal ini, perlu digunakan model pembelajaran mesin selain regresi linier.

2) Confusion Matrix

Confusion matrix adalah sebuah tabel yang menggambarkan kinerja model klasifikasi dengan membandingkan hasil prediksi model dengan data aktual. Matriks ini menyediakan informasi mengenai:

- *True Positives (TP)*: Jumlah data positif yang benar-benar diprediksi sebagai positif.

- *True Negatives (TN)*: Jumlah data negatif yang benar-benar diprediksi sebagai negatif.
- *False Positives (FP)*: Jumlah data negatif yang salah diprediksi sebagai positif.
- *False Negatives (FN)*: Jumlah data positif yang salah diprediksi sebagai negatif.

Tabel 4. Confusion Matrix

	Prediksi Positif	Prediksi Negatif
Aktual Positif	TP	FN
Aktual Negatif	FP	TN

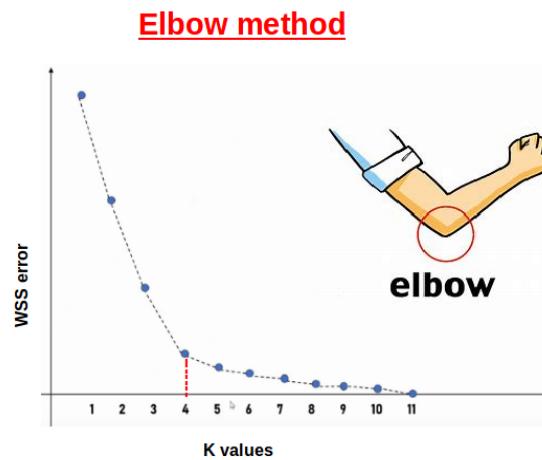
Implementasi confusion matrix untuk mengevaluasi model klasifikasi dapat dilihat pada tautan
<https://colab.research.google.com/drive/1SM7LTKJ8KPr1ziwyquQzBB3jdUwNpC62?usp=sharing>

Untuk meningkatkan kinerja model klasifikasi yang menunjukkan matriks kebingungan (confusion matrix) dengan banyak salah prediksi, dapat diawali dengan memeriksa keseimbangan data, karena jumlah contoh tiap kelas yang timpang sering menyebabkan model untuk “berpihak” pada kelas mayoritas. Selanjutnya, dapat dilakukan pemilihan dan rekayasa fitur dengan menghilangkan variabel yang tidak relevan dan melakukan normalisasi skala.

3) Evaluasi Model Klasterisasi

Dalam analisis *clustering*, tiap titik data dikelompokkan tanpa penanda kelas bawaan, sehingga kualitas hasilnya dinilai dari dua sudut. **Pertama, evaluasi internal untuk menelaah “kerapatan” dan “jarak” antar-gugus.** Sebuah klaster dikatakan baik apabila anggotanya saling berdekatan, ibarat sekumpulan buku bertema sama disusun rapat di rak yang sama, sedangkan klaster yang terpisah baiknya berjauhan dari klaster lain, layaknya rak-rak berbeda di perpustakaan. Ukuran formal-nya melibatkan perbandingan rata-rata jarak antar-anggota dalam klaster (*intra-cluster*) dengan rata-rata jarak ke klaster terdekat (*inter-cluster*); semakin kecil jarak internal dan semakin besar jarak eksternal, maka semakin “bersih” pengelompokannya.

Contoh evaluasi yang sering digunakan untuk model klasterisasi adalah metode siku. Metode siku digunakan untuk menentukan banyaknya klaster yang paling optimal dengan menarik tiap titik data ke pusat klaster dan menghitung jaraknya. Semakin kecil jarak, maka semakin rapat sebuah klaster. Jika jarak-jarak tersebut divisualisasikan dalam sebuah grafik garis, maka akan terbentuk sebuah garis yang menurun tajam di awal, hingga suatu nilai di mana penurunannya menjadi lebih landai.



Gambar 56. Metode Siku. Sumber: <https://medium.com/@zalarushirajsingh07/the-elbow-method-finding-the-optimal-number-of-clusters-d297f5aeb189>

Implementasi metode siku pada model klasterisasi K-Means dapat dilihat pada tautan
https://colab.research.google.com/drive/1lr3qYX9ZurElDfhZtu_ylhJbbyhNQQM7?usp=sharing

Nilai k pada metode siku menunjukkan banyak klaster yang optimal untuk digunakan dalam klasterisasi.

4) Evaluasi Lainnya: *Accuracy, Recall, Precision, dan F1 Score*

a. *Accuracy*

Akurasi (*accuracy*) adalah proporsi klasifikasi yang benar, baik positif maupun negatif. Model yang sempurna memiliki tingkat akurasi 100% dengan nol kesalahan. Namun, pada dataset yang tidak seimbang, tingkat akurasi bisa menyesatkan jika terdapat satu kelas yang mendominasi.

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{total classifications}} = \frac{TP + TN}{TP + TN + FP + FN}$$

b. Recall

Recall atau *true positive rate* (TPR) adalah evaluasi model yang mengukur proporsi data positif yang diklasifikasikan dengan benar. Evaluasi *recall* sangat penting dalam kasus-kasus seperti deteksi penyakit, di mana terjadinya kesalahan negatif jauh lebih berbahaya daripada kesalahan positif. Model yang sempurna, secara hipotetis, tidak akan memiliki *false negative* sehingga *recall* (TPR)-nya menjadi 1.0, atau dengan kata lain, tingkat keberhasilan deteksi mencapai 100%.

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN}$$

c. Precision

Presisi (*precision*) digunakan untuk mengukur proporsi prediksi positif yang benar. Model yang lebih presisi memiliki lebih sedikit kesalahan positif. Model yang sempurna, secara hipotetis, akan memiliki nol *false positive* sehingga presisinya menjadi 1.0 (100%). Namun, presisi dan recall sering memiliki hubungan terbalik, di mana peningkatan pada salah satunya, dapat menurunkan yang lain.

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP}$$

d. F1 Score

F1 Score adalah rata-rata harmonik dari presisi dan *recall*. Metrik ini menyeimbangkan presisi dan *recall*, sehingga lebih cocok dibandingkan akurasi pada dataset yang tidak seimbang. Jika presisi dan *recall* sama-sama 1.0, maka F1 Score juga 1.0. Secara umum, jika presisi dan *recall* memiliki nilai yang mirip, F1 Score akan mendekati nilai tersebut. Jika keduanya berbeda jauh, F1 Score akan lebih mendekati nilai yang lebih rendah.

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP + FP + FN}$$

Lembar Kerja 4.1: Proyek Pemrograman Aplikasi KA (*supervised/unsupervised learning*) Menggunakan Dataset

A. Deskripsi

Lembar Kerja ini dirancang untuk membimbing peserta pelatihan dalam menerapkan sintaks dasar pemrograman KA dan membangun aplikasi sederhana berbasis supervised atau unsupervised learning. Kegiatan dilakukan melalui proyek kolaboratif yang bertemakan bidang keahlian masing-masing. Proyek ini mencakup penggunaan library KA populer, evaluasi model dengan tools yang sesuai, hingga penyusunan dokumentasi teknis secara lengkap.

B. Tujuan

Setelah menyelesaikan lembar kerja ini, peserta pelatihan diharapkan mampu:

1. Menerapkan sintaks dasar bahasa pemrograman KA untuk menulis skrip sederhana yang melibatkan logika KA.
2. Menerapkan *library* KA populer untuk membangun model sederhana, seperti klasifikasi teks/gambar.
3. Menganalisis output KA menggunakan matrik evaluasi dan melakukan penyempurnaan kode untuk meningkatkan performa.
4. Menyusun dokumentasi teknis yang mencakup alur kerja, kode, dan analisis hasil aplikasi KA yang dikembangkan.

C. Petunjuk Kerja

Berikut ini adalah empat tugas utama yang harus diselesaikan oleh peserta pelatihan:

1. Menulis Skrip Sederhana dengan Sintaks Dasar KA

Pada bagian ini,, Anda akan membangun dasar pemahaman logika pemrograman KA dengan membuat skrip sederhana. Tugas ini bertujuan melatih Anda dalam menggunakan struktur dasar seperti percabangan, perulangan, dan fungsi untuk menciptakan sebuah program yang merespons input pengguna.

Langkah-langkah:

Pilih bahasa pemrograman (Python direkomendasikan).

1. Buatlah sebuah skrip sederhana yang menerima input suhu (dalam derajat Celsius), lalu mencetak:

- a. "Panas" jika suhu > 30
 - b. "Sejuk" jika suhu antara 20–30
 - c. "Dingin" jika suhu < 20
2. Gunakan `input()`, `int()`, dan `if-elif-else`. Tambahkan minimal 1 fungsi untuk modularisasi kode.
 3. Simpan skrip dengan format misalnya `.py`.

2. Membangun Model Sederhana Menggunakan Library KA

Pada bagian ini, Anda akan mengembangkan sebuah model pembelajaran mesin sederhana dengan menggunakan pustaka KA populer. Fokus utama adalah pada pemrosesan dataset dan membangun model klasifikasi atau klasterisasi sesuai dengan bidang keahlian di SMA/SMK (misal: guru SMK kelompok pariwisata dapat menggunakan dataset jumlah pengunjung suatu tempat wisata pada tahun tertentu, guru SMK kelompok otomotif dapat menggunakan dataset penjualan mobil listrik pada kurun waktu tertentu, dan lain-lain)..

Langkah-langkah:

1. Pilih salah satu pustaka KA populer: 'Scikit-learn', 'TensorFlow', atau 'Keras'.
2. Unduh dataset dari platform dataset terbuka seperti Kaggle, UCI, Google Dataset Search, data.go.id, atau data.gov.
3. Dataset harus memiliki minimal **150 data dan 2–5 fitur utama**.
4. Bangun model *supervised/ unsupervised learning* (contoh: prediksi kelayakan pinjaman berdasarkan profil keuangan, klasifikasi jenis pakaian, analisis kepuasan pelanggan, klasifikasi jenis barang).
5. Lakukan training dan testing model.
6. Simpan notebook `'.ipynb'` atau `'.py'` yang berisi seluruh proses.

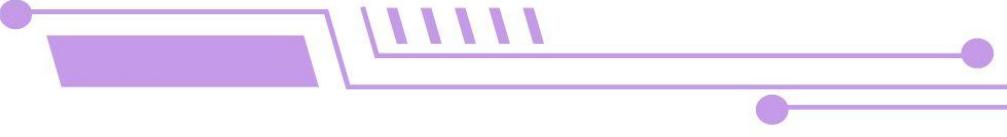
Hasil yang diharapkan: Skrip program KA dengan model klasifikasi yang dapat dijalankan dan menggunakan dataset valid.

3. Evaluasi dan Penyempurnaan Kode

Pada bagian ini berfokus pada kemampuan Anda dalam mengevaluasi dan meningkatkan performa model KA yang telah dibuat. Anda akan menggunakan *tools evaluasi* yang sesuai dan menyempurnakan kode untuk memperoleh hasil yang lebih optimal.

Langkah-langkah:

1. Gunakan tools evaluasi yang sesuai dengan jenis model dan metode yang digunakan:

- 
- a. Untuk klasifikasi: confusion matrix, accuracy score, precision, recall, F1-score.
 - b. Untuk regresi: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R² score.
 - c. Tools seperti `classification_report()` dan `metrics.plot_confusion_matrix()` pada scikit-learn sangat direkomendasikan.
2. Analisis hasil dan identifikasi kekurangan model.
 3. Lakukan penyempurnaan kode.
 4. Catat perubahan dan bandingkan performa model.

Hasil yang diharapkan: Output evaluasi dan laporan hasil optimasi performa model berdasarkan tools yang relevan.

4. Menyusun Dokumentasi

Pada bagian ini bertujuan melatih Anda dalam menyusun dokumentasi proyek secara profesional dan lengkap. Dokumentasi akan mencerminkan keseluruhan proses proyek mulai dari perencanaan, pemrograman, evaluasi, hingga refleksi hasil kerja Anda.

Langkah-langkah:

1. Buat file dokumentasi dalam format `'.docx'` atau `'.pdf'`.
2. Dokumentasi mencakup deskripsi proyek, sumber dataset, jumlah data dan fitur, workflow, potongan kode, tools evaluasi, hasil evaluasi, refleksi, dan saran lanjutan.

Hasil yang Diharapkan: Dokumen lengkap dan rapi yang menjelaskan keseluruhan proses proyek.

D. Format Laporan Hasil

Komponen	Keterangan
Judul Proyek	(Sesuai tema bidang keahlian)
Bahasa & Library	(Contoh: Python + Scikit-learn)
Sumber Dataset	(Contoh: Kaggle – Iris Dataset)
Jumlah Data & Fitur	(Contoh: 150 data, 4 fitur)

Deskripsi Dataset	(Gambaran umum isi dan tujuan dataset)
Alur Kerja	(Diagram alur proses dan tahapan kerja model)
Potongan Kode	(Kode penting yang menjelaskan logika/model)
Tools Evaluasi	(Jenis evaluasi dan alasan pemilihannya)
Hasil Evaluasi	(Tabel, grafik, metrik kinerja model)
Perbaikan Model	(Penyesuaian parameter dan dampaknya)
Refleksi	(Apa yang dipelajari, kendala, solusi, dan rencana lanjutan)

E. Kriteria Penilaian

Kriteria	Sangat Baik (4)	Baik (3)	Cukup (2)	Perlu Perbaikan (1)
Pemahaman	Menunjukkan penguasaan penuh terhadap sintaks dan konsep KA	Menguasai konsep dasar dan penerapan KA dengan baik	Masih ada kekeliruan dalam pemahaman konsep KA	Tidak menunjukkan pemahaman memadai
Ketepatan Penerapan	Kode berjalan sempurna, dokumentasi sangat rapi	Kode sebagian besar berjalan dengan dokumentasi cukup jelas	Beberapa error dalam kode, dokumentasi kurang lengkap	Kode banyak error dan dokumentasi minim

Kreativitas	Evaluasi lengkap, menggunakan tools yang sesuai, insight kuat dan logis	Evaluasi dilakukan dengan tools yang cukup tepat, insight memadai	Evaluasi minim dan/atau tools kurang sesuai	Tidak ada evaluasi atau hanya menyebut hasil tanpa analisis
Refleksi	Dokumentasi sangat rapi, sistematis, lengkap, dan menarik	Dokumentasi cukup rapi dan sistematis	Dokumentasi kurang runtut dan visual minim	Dokumentasi acak dan tidak jelas

Skor akhir peserta dihitung berdasarkan jumlah skor dari keempat kriteria di atas, kemudian dikonversi ke skala 0–100 dengan ketentuan sebagai berikut:

- ✓ Rumus Konversi Skor: $(\text{Total skor yang diperoleh} \div 16) \times 100$
- ✓ Rentang Skor Akhir (0–100):
 - Sangat Baik: 85 – 100
 - Baik: 70 – 84
 - Cukup: 55 – 69
 - Perlu Perbaikan: < 55

C. Pengenalan *Large Language Model* pada Kecerdasan Artifisial Generatif

Dalam era digital saat ini, KA telah mengalami perkembangan yang sangat pesat, salah satunya melalui pemanfaatan model bahasa besar (*Large Language Models/LLM*). Model-model ini tidak hanya mampu memahami dan menghasilkan teks dengan tingkat kefasihan yang sangat tinggi, tetapi juga memiliki potensi untuk mendukung berbagai aplikasi dalam bidang KA generatif. Penggunaan LLM telah merevolusi bagaimana cara kita berinteraksi dengan mesin, mengotomasi proses-proses kreatif, dan mengintegrasikan kemampuan pemrosesan bahasa alami ke dalam sistem-sistem komputasional.

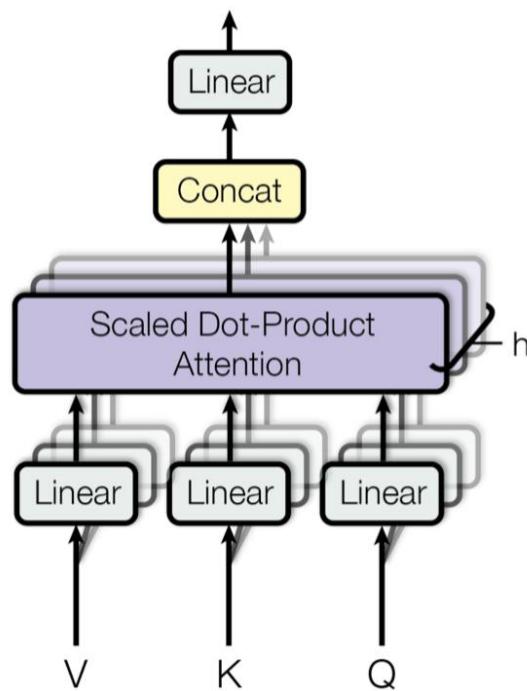
C.1. Konsep Dasar, Arsitektur, dan Cara Kerja *Large Language Model*

Large Language Model adalah sebuah sistem berbasis jaringan syaraf tiruan yang dilatih untuk dapat memahami, memproses, dan menghasilkan teks dalam bahasa alami atau bahasa natural (bahasa yang digunakan manusia sehari-hari dalam berbagai gaya). Konsep dasar di balik LLM berakar pada pengembangan *Natural Language Processing (NLP)* yang telah berlangsung selama beberapa dekade terakhir. Pada awalnya, pendekatan *NLP* didasarkan pada pengembangan serangkaian aturan berbasis simbol dan statistik yang sederhana. Namun, seiring dengan kemajuan teknologi komputasi dan teori pembelajaran mesin, dikembangkanlah metode pembelajaran mendalam (*deep learning*) yang memungkinkan adanya pemrosesan bahasa secara lebih holistik dan kontekstual.

Penjelasan visual tentang LLM dapat disimak dalam tautan video
<https://www.youtube.com/watch?v=LPZh9BOjkQs>

Sejarah LLM dapat ditelusuri kembali ke perkembangan model-model *neural network* yang menerapkan arsitektur *Recurrent Neural Networks (RNN)* dan *Long Short-Term Memory (LSTM)*. Namun, terobosan besar terjadi dengan diperkenalkannya arsitektur Transformer dalam makalah “*Attention is All You Need*” oleh Vaswani et al. (2017). Arsitektur Transformer mengubah paradigma pemrosesan sekuensial dengan mengandalkan mekanisme atensi (*attention mechanism*, semacam operasi matematika yang melibatkan matriks dengan dimensi banyak—Gambar 57) yang memungkinkan model untuk mempelajari hubungan antar kata dalam

satu kalimat atau bahkan antar kalimat secara simultan. Perkembangan inilah yang membuka jalan bagi dikembangkannya model-model bahasa besar seperti BERT, GPT, dan T5.



Gambar 57. Multi-head Attention. Sumber: <https://paperswithcode.com/method/multi-head-attention>

Cara kerja mekanisme atensi (*attention mechanism*) dapat dilihat pada tautan video <https://www.youtube.com/watch?v=eMlx5fFNoYc>

1. Natural Language Processing

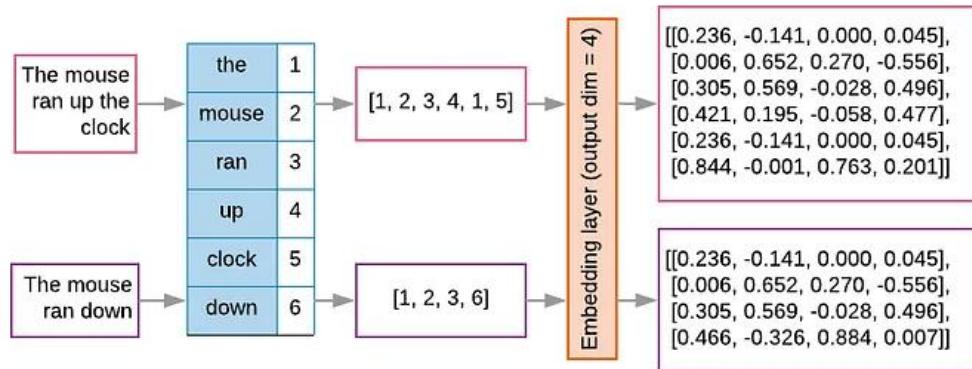
Natural Language Processing adalah cabang dari KA yang berfokus pada interaksi antara komputer dengan bahasa manusia/bahasa alami. NLP dapat digunakan dalam berbagai kasus seperti analisis sentimen, penerjemahan bahasa, dan lain-lain. Dalam konteks LLM, NLP berperan penting dalam:

- Tokenisasi: Memecah teks menjadi unit-unit yang lebih kecil, seperti kata atau sub-kata, sehingga model LLM dapat memprosesnya dengan lebih optimal.



Gambar 58: Tokenisasi. Sumber: <https://medium.com/@utkarsh.kant/tokenization-a-complete-guide-3f2dd56c0682>

- Embedding: Mengonversi token-token tersebut menjadi bentuk vektor atau matriks yang memuat informasi semantik.



Gambar 59. Tokenisasi. Sumber: <https://airbyte.com/data-engineering-resources/tokenization-vs-embeddings>

- Pemahaman Konteks: Menggunakan konteks dari kata-kata sekitarnya untuk menginterpretasikan makna teks secara lebih mendalam.

2. Arsitektur Internal Large Language Model

LLM modern pada umumnya dibangun berdasarkan arsitektur Transformer, yang terdiri dari beberapa komponen inti, yaitu:

- *Embedding Layer*: Bagian awal yang bertugas untuk mengubah input berupa token menjadi representasi vektor.
- *Encoder dan Decoder*: Pada beberapa model seperti GPT, hanya *decoder* yang digunakan untuk menghasilkan teks, sementara model lain seperti BERT mengadopsi arsitektur *encoder* saja untuk memahami konteks.
- Mekanisme *Multi-Head Attention*: Memungkinkan model untuk fokus pada berbagai bagian teks secara bersamaan. Mekanisme ini secara signifikan meningkatkan kemampuan model dalam menangkap hubungan antar token secara paralel (Vaswani, 2017).
- *Feed-Forward Neural Networks*: Digunakan setelah proses atensi untuk mengolah informasi yang telah dikumpulkan.

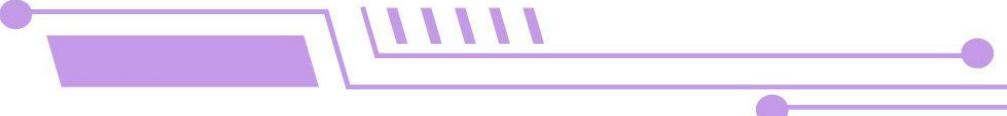
Arsitektur Transformer mampu merevolusi dunia *NLP* dengan menghilangkan kebutuhan untuk pemrosesan yang bersifat sekuensial yang tradisional. Dan menggantikannya dengan mekanisme atensi untuk menghubungkan setiap token dengan semua token lainnya dalam inputan, yang memungkinkan terjadinya pemrosesan yang bersifat paralel sehingga tingkat efisiensi menjadi lebih tinggi. Model GPT (*Generative Pre-trained Transformer*), juga menggunakan pendekatan *autoregressive*, di mana model melakukan prediksi secara bertahap untuk menghasilkan kata berikutnya berdasarkan konteks yang dipahami sebelumnya. Pendekatan ini memungkinkan model GPT untuk menghasilkan teks yang panjang dan koheren, meskipun masih terdapat tantangan dalam menjaga konsistensi konteks dalam berbagai teks yang sangat panjang.

3. Cara Kerja LLM: *Training* dan *Inference*

1) Proses *Training*

Training yang dilakukan oleh sebuah LLM melibatkan dua tahap utama, yaitu *pre-training* dan *fine-tuning*.

- *Pre-training*: Pada tahap ini, model dilatih dengan dataset berbentuk teks dalam jumlah yang sangat besar. Tujuannya adalah agar model dapat mempelajari pola bahasa, struktur sintaks atau tata bahasa, dan konteks semantik. Proses ini umumnya menggunakan teknik *unsupervised learning*, di mana model belajar dari data tanpa adanya label. Evaluasi yang umum digunakan untuk menilai performa LLM adalah *cross-entropy loss* yang mengukur ketidaksesuaian antara prediksi yang dihasilkan oleh model dengan distribusi token sebenarnya.
- *Fine-tuning*: Setelah dilakukan *pre-training*, model dapat disesuaikan (*fine-tuned*) untuk tugas-tugas yang bersifat spesifik seperti penerjemahan, pembuatan



rangkuman, atau klasifikasi. *Fine-tuning* melibatkan pelatihan tambahan dengan dataset yang lebih kecil namun relevan dengan tugas yang diinginkan, sehingga model dapat lebih optimal dalam aplikasi spesifik tersebut.

2) Proses *Inference*

Pada tahap *inference*, LLM digunakan untuk **menghasilkan teks atau menjawab pertanyaan**. Proses *inference* melibatkan proses pemberian input kepada model dan model diharapkan menghasilkan *output* secara bertahap. Berbagai teknik *sampling* digunakan untuk mengendalikan kualitas dan keragaman *output* yang dihasilkan. Salah satu teknik *sampling* yang digunakan ialah *beam search*, di mana model mengevaluasi beberapa kemungkinan *output* sekaligus dan memilih yang paling koheren secara keseluruhan.

Meskipun LLM memiliki potensi yang besar, proses *training LLM* memerlukan sumber daya komputasi yang sangat tinggi, termasuk GPU/TPU dan memori yang besar. Selain itu, berbagai tantangan yang dapat terjadi, seperti *overfitting*, bias data, dan kestabilan *training* menjadi perhatian utama bagi para peneliti dan praktisi. Upaya-upaya untuk mengatasi berbagai tantangan ini melibatkan teknik *regularisasi*, penggunaan dataset yang lebih beragam, dan penerapan optimasi parameter yang lebih efisien.

4. Isu Etika dan Keamanan

Sebagai teknologi yang berdampak luas, LLM juga menimbulkan sejumlah isu etis, antara lain:

- Bias dan Diskriminasi: LLM dapat mereplikasi dan bahkan memperkuat bias yang terdapat dalam data latihnya. Penelitian terus dilakukan untuk mengidentifikasi dan mengurangi bias tersebut agar *output* yang dihasilkan lebih adil.
- Keamanan dan Penyalahgunaan: Kemampuan LLM dalam menghasilkan teks yang koheren juga membawa risiko penyalahgunaan, misalnya dalam pembuatan berita palsu atau konten yang menyesatkan. Oleh karena itu, perlu adanya pedoman dan kebijakan penggunaan KA generatif yang bertanggung jawab.
- Transparansi Model: Keterbukaan mengenai penggunaan data latih dan mekanisme internal model merupakan aspek penting untuk memastikan kepercayaan publik serta mendukung audit dan verifikasi dari para peneliti.

Fenomena “Ghiblifikasi” Viral



Sam Altman, CEO OpenAI (perusahaan yang mengembangkan ChatGPT), baru-baru ini memperkenalkan kemampuan baru yang dimiliki oleh ChatGPT sebagai KA generatif, yaitu mengubah gambar atau foto menjadi suatu gambar yang memiliki gaya yang sangat mirip dengan gambar yang dihasilkan oleh Studio Ghibli. Hampir seketika, kemampuan baru ChatGPT ini menjadi viral. Sam Altman melaporkan bahwa sumber daya komputasi yang mereka miliki mengalami *overheat*, karena saking banyaknya pengguna yang memanfaatkan kemampuan baru tersebut. Viralnya “Ghiblifikasi” tersebut memicu perdebatan tentang etika penggunaan KA generatif.

Isu yang mencuat dari tren ini mencakup tiga poin utama:

- **hak cipta**, yaitu kekhawatiran bahwa KA telah dilatih menggunakan karya Studio Ghibli tanpa izin;
- **keterbatasan teknis**, karena lonjakan penggunaan membuat sistem ChatGPT kewalahan;
- **implikasi etis**, termasuk kritik dari Hayao Miyazaki (*co-founder* Studio Ghibli)

terhadap peran KA dalam dunia seni.

Namun, tidak ada pernyataan resmi dari Studio Ghibli meskipun kritik sebelumnya dari Hayao Miyazaki terhadap KA, seperti yang disebutkan dalam Forbes, memberikan konteks bahwa mereka mungkin tidak mendukung tren ini.

Catatan: Gambar yang digunakan pada bagian ini diposting oleh akun X resmi Sam Altman (<https://x.com/sama/status/1907224234277171677>)

C.2. Peran *Large Language Model* dalam Kecerdasan Artifisial Generatif dan Aplikasinya

1) LLM sebagai Inti dari KA Generatif

Large Language Model memainkan peran yang sangat penting, bahkan krusial dalam ekosistem KA generatif. Dengan kemampuannya dalam menghasilkan teks yang tidak hanya benar secara tata bahasa tetapi juga relevan secara konteks, LLM telah menjadi fondasi dari banyak sekali aplikasi kreatif. Di antaranya adalah penulisan otomatis, penerjemahan bahasa, pembuatan rangkuman, reviu literatur, dan pembuatan konten kreatif seperti puisi atau cerita pendek.

Sebagai contoh, model GPT-4 yang merupakan salah satu LLM paling terkenal, telah digunakan dalam berbagai aplikasi, mulai dari *chatbots* hingga penulisan kode program dalam berbagai bahasa pemrograman. Model LLM lain, yaitu LLAMA yang dikembangkan oleh Meta, telah diintegrasikan ke dalam berbagai aplikasi yang digunakan oleh masyarakat sehari-hari, seperti WhatsApp. Kemampuan berbagai model LLM dalam menanggapi pertanyaan secara interaktif dan menghasilkan teks yang mendekati gaya bahasa manusia menjadikannya alat yang efektif dalam berbagai skenario pemanfaatan KA generatif (Brown, 2020).

2) Aplikasi *LLM* dalam Industri

1) Chatbot dan Asisten Virtual

Salah satu aplikasi yang paling populer dari LLM adalah pembuatan *chatbot* dan asisten virtual. Dengan memanfaatkan LLM, *chatbot* dapat memberikan respons yang lebih natural dan kontekstual kepada penggunanya. Misalnya, dalam layanan pelanggan,



chatbot yang didukung dengan pemanfaatan LLM mampu memahami pertanyaan pelanggan dengan lebih baik dan memberikan solusi secara instan. Integrasi ini tidak hanya meningkatkan kepuasan pelanggan tetapi juga mengurangi beban operasional perusahaan.

2) Penulisan dan Pembuatan Konten

Di dunia media dan pemasaran, LLM telah digunakan untuk menghasilkan artikel, blog, desain visual, ilustrasi, dan materi pemasaran dengan cepat. Berbagai perusahaan menggunakan perangkat KA berbasis LLM untuk mengotomatiskan proses penulisan, sehingga memungkinkan dilakukannya produksi konten dalam berbagai bentuk dalam jumlah besar tanpa mengorbankan kualitas. Selain itu, LLM juga mampu menghasilkan konten kreatif seperti cerita fiksi, puisi, bahkan skrip film, yang membuka peluang baru dalam industri hiburan dan kreatif.

3) Penerjemahan Bahasa dan Pembuatan Ringkasan (*Summarization*)

LLM juga memainkan peran penting dalam penerjemahan bahasa dan pembuatan ringkasan (*summarization*). Dengan kemampuannya memahami konteks kalimat secara mendalam, model-model bahasa besar seperti BERT dan T5 mampu menerjemahkan teks dari satu bahasa ke bahasa lain dengan tingkat akurasi yang sangat tinggi. Selain itu, kemampuan LLM dalam membuat ringkasan dari suatu dokumen, baik dokumen yang ditulis dengan bahasa formal maupun non-formal, membuat LLM menjadi sangat berguna dan berperan dalam bidang riset dan pendidikan (Devlin, 2018).

4) Aplikasi dalam Pendidikan dan Riset

Dalam bidang pendidikan, LLM telah digunakan dengan cukup masif sebagai alat bantu dalam pembelajaran, isalnya, menjawab pertanyaan peserta didik, memberikan penjelasan mengenai konsep-konsep yang sulit dimengerti dengan bahasa yang lebih mudah dipahami, dan bahkan membantu dalam review penulisan esai. Sedangkan di ranah riset, LLM dapat membantu para peneliti untuk memproses informasi dari berbagai literatur yang berjumlah besar, sehingga mempercepat proses penelitian dan analisis data. Hal ini membuka peluang bagi adanya inovasi dan kolaborasi antar disiplin ilmu.

C.3. Teknik Integrasi Pemrograman Kecerdasan Artifisial Dengan Model *Large Language Model*

1) Pendekatan Integrasi melalui API

Salah satu cara paling populer untuk mengintegrasikan LLM ke dalam aplikasi adalah melalui *Application Programming Interface (API)*. Banyak sekali penyedia LLM, seperti OpenAI dan Hugging Face, menyediakan API yang memungkinkan para pengembang untuk dengan mudah mengirimkan permintaan (*request*) dan menerima respons berupa teks yang dihasilkan oleh LLM. Pendekatan ini memiliki beberapa keunggulan, yaitu:

- Kemudahan Implementasi: Pengembang tidak perlu membangun dan melatih model dari awal, cukup dengan menggunakan API yang telah disediakan.
- Skalabilitas: API sering kali dioptimalkan untuk menangani permintaan dalam skala besar sehingga cocok untuk aplikasi komersial.
- Keamanan dan Pemeliharaan: Pengelolaan model serta infrastruktur backend dilakukan oleh penyedia API, sehingga pengembang dapat fokus pada pengembangan aplikasi.

Contoh implementasi API dapat ditemukan pada OpenAI GPT, yang memungkinkan integrasi dengan berbagai bahasa pemrograman seperti Python, JavaScript, dan lainnya (OpenAI, 2020).

2) Integrasi dengan *Pipeline Machine Learning*

Selain melalui *API*, integrasi LLM juga dapat dilakukan dalam sebuah *pipeline machine learning* yang lebih kompleks. *Pipeline* ini biasanya melibatkan beberapa tahapan, seperti:

- *Preprocessing Data*: Mengolah dan membersihkan data teks agar sesuai dengan format input yang dapat diterima LLM.
- *Pemrosesan dengan LLM*: Menggunakan LLM untuk melakukan tugas-tugas seperti klasifikasi, ekstraksi informasi, atau pembuatan teks.
- *Post Processing*: Mengolah *output* dari LLM agar dapat digunakan secara langsung oleh aplikasi, seperti menyusun ringkasan atau mengkonversi format teks.

Pipeline semacam ini umumnya dibangun dengan menggunakan *framework* pembelajaran mesin populer seperti TensorFlow atau PyTorch, yang telah terintegrasi dengan modul-modul *NLP* modern. Selain itu, pustaka seperti Hugging Face's

Transformers juga menyediakan antarmuka yang memudahkan integrasi berbagai LLM ke dalam *pipeline* aplikasi (HuggingFace, 2021).

3) Integrasi di Berbagai Bahasa Pemrograman

a) Python

Python telah menjadi salah satu bahasa pemrograman utama dalam bidang *machine learning* dan *NLP* karena kelengkapan ekosistem pustaka yang dimilikinya. Dengan *library* seperti Transformers, TensorFlow, dan PyTorch, pengembang dapat dengan mudah mengintegrasikan LLM ke dalam aplikasi mereka. Contoh teknik integrasi dengan Python meliputi:

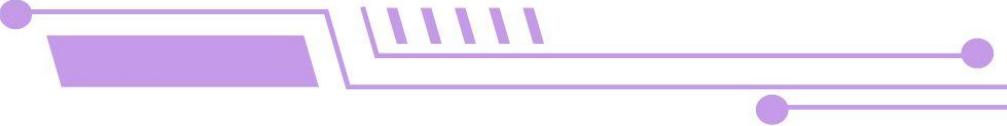
- Menggunakan pustaka Transformers: Dengan menuliskan beberapa baris kode, pengembang dapat memuat model LLM pra-latih dan menggunakan untuk berbagai kebutuhan.
- Pipeline Modifikasi: Pengembang dapat membuat *pipeline* khusus untuk melakukan *preprocessing*, *inference*, dan *post processing* sesuai dengan kebutuhan aplikasi.
- Integrasi dengan Web Framework: Python juga mendukung integrasi dengan *framework* web seperti Flask atau Django untuk membangun aplikasi berbasis LLM secara *real-time*.

```
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[{
        "role": "user",
        "content": "Write a one-sentence bedtime story about a unicorn."
    }]
)

print(completion.choices[0].message.content)
```

Gambar 60. Potongan Kode Program Bahasa Python Untuk Terhubung
Dengan Model LLM GPT-4o



b) JavaScript dan Node.js

Di bidang pengembangan web, JavaScript juga dapat digunakan untuk integrasi LLM melalui *API*. Dengan menggunakan Node.js, pengembang dapat membuat aplikasi web yang interaktif dan responsif, di mana permintaan atau *request* ke model LLM dilakukan secara asinkron. Pendekatan pada umumnya digunakan untuk pengembangan *chatbot* atau asisten virtual yang berjalan di *browser*.

c) Bahasa Lain dan *Cross-Platform*

Bahasa pemrograman lain seperti Java, C#, dan Go juga telah mulai mengeksplorasi integrasi dengan model KA melalui pustaka dan teknik integrasi yang relevan. Pendekatan *cross-platform* memungkinkan terjadinya integrasi LLM ke dalam aplikasi desktop, mobile, atau *embedded systems*, sehingga memperluas cakupan aplikasi KA generatif.

4) *Tools dan Library* Pendukung

Seiring dengan perkembangan teknologi LLM, banyak sekali perangkat atau tools dan *library* yang dikembangkan dengan tujuan untuk memudahkan integrasi antara aplikasi KA dan LLM, antara lain:

- Hugging Face Transformers: *Library open source* yang menyediakan antarmuka untuk berbagai model LLM seperti BERT, GPT, T5, dan lain-lain. *Library* ini cukup populer karena kemudahan dalam penggunaannya dan adanya dukungan dari berbagai komunitas yang aktif.
- TensorFlow dan PyTorch: Dua *framework machine learning* terkemuka yang dapat pula digunakan untuk *training* dan *deployment* model-model *deep learning*, termasuk LLM.
- FastAPI dan Flask: *Framework* untuk membangun aplikasi web berbasis Python yang dapat mengintegrasikan model LLM secara *real-time*.
- SDK dari Penyedia API: Banyak sekali penyedia LLM yang menyediakan *Software Development Kit (SDK)* yang memudahkan integrasi LLM dengan berbagai aplikasi KA, misalnya OpenAI API SDK.

5) *Tantangan dan Best Practices* dalam Integrasi

Integrasi LLM ke dalam aplikasi KA tidak luput dari sejumlah tantangan, baik teknis dan non-teknis. Beberapa tantangan utama antara lain:

- Latency dan Kecepatan *Inference*: Model LLM yang besar memerlukan waktu inferensi yang tidak selalu dapat dijamin untuk berlangsung secara *real-time*, sehingga perlu dioptimalkan dengan *caching* atau teknik kompresi model.
- Skalabilitas: Untuk aplikasi yang diakses oleh jutaan pengguna, diperlukan arsitektur backend yang mampu menangani beban permintaan secara simultan.
- Keamanan dan Privasi Data: Data pengguna yang dikirim ke model LLM harus dikelola dengan standar keamanan tinggi agar tidak terjadi penyalahgunaan.
- Pemantauan dan Logging: Penting untuk membangun sistem pemantauan agar dapat mendeteksi kesalahan atau bias dalam output model, sehingga dapat dilakukan perbaikan secara iteratif.

Best practices yang dianjurkan mencakup:

- Melakukan pengujian menyeluruh (unit testing dan integration testing) pada setiap tahap pipeline.
- Mengimplementasikan fallback system apabila LLM gagal memberikan respons yang memadai.
- Mengedepankan prinsip-prinsip etis dalam penggunaan data dan penyajian output, termasuk transparansi terkait batasan model.

Aktivitas 3: Mengintegrasikan LLM ke aplikasi KA menggunakan Hugging Face API (Pirate Chatbot)

Salah satu cara efektif untuk mengintegrasikan LLM ke dalam aplikasi KA adalah dengan memanfaatkan Hugging Face, sebuah platform yang menyediakan akses mudah ke berbagai model bahasa canggih seperti GPT, BERT, dan Llama. Sub topik ini akan mengupas bagaimana proses integrasi LLM ke dalam aplikasi KA dilakukan secara praktis melalui Hugging Face dengan mekanisme *pipeline*, dengan studi kasus berupa Pirate Chatbot. Pirate Chatbot ini didesain untuk memberikan respons dengan gaya bahasa bajak laut, sehingga tidak hanya menunjukkan fleksibilitas dan kemampuan model LLM dalam menghasilkan konten yang sesuai konteks, tetapi juga menambahkan unsur hiburan yang unik.

- 1) Mengunjungi situs web Hugging Face melalui <https://huggingface.co/>.
- 2) Membuat akun melalui menu “Sign Up”.

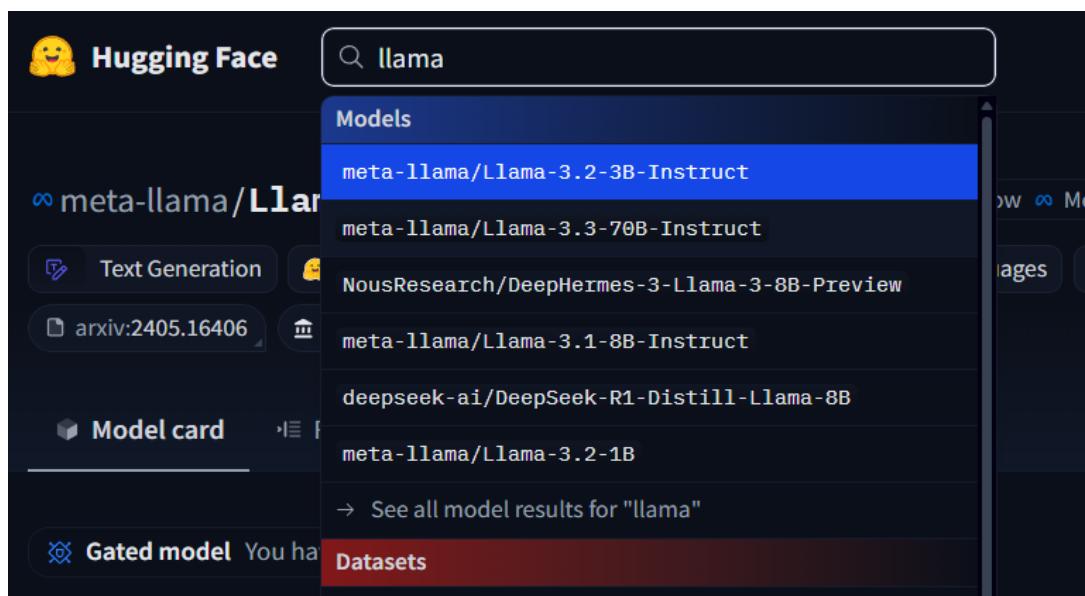
- 3) Setelah mendaftar, navigasikan ke <https://huggingface.co/settings/tokens> untuk membuat token API. Token ini penting untuk mengautentikasi permintaan API. Langsung salin dan simpan token yang dihasilkan, karena hanya dapat dilihat satu kali.

The screenshot shows the 'Access Tokens' section of the Hugging Face settings. It displays a table of existing tokens and a prominent red box around the '+ Create new token' button.

Name	Value	Last Refreshed Date	Last Used Date	Permissions
first_test	hf_...Kegh	about 5 hours ago	about 5 hours ago	WRITE

The screenshot shows the 'Create new Access Token' dialog. It includes fields for 'Token type' (set to 'Write'), 'Token name' (set to 'pirate_chatbot'), and a large red box around the 'Create token' button.

- 4) Setelah akun siap, pilih LLM yang tersedia di Hugging Face. Misalnya, LLaMA 3 milik Meta adalah LLM sumber terbuka yang dapat digunakan. Untuk menemukan model, kunjungi pusat model Hugging Face lalu ketik nama model di bilah pencarian (misalnya, "Llama 3").



5) Latihan ini menggunakan model /Llama-3.2-3B-Instruct

- 6) Setelah model yang diinginkan dipilih, langkah berikutnya adalah mencatat jalur/path modelnya. Dalam kasus ini, path untuk LLaMA 3 adalah meta-llama/Llama-3.2-3B-Instruct.
- 7) Interaksi dengan model LLM Llama-3.2-3B-Instruct melalui API Hugging Face dapat dilakukan menggunakan aplikasi sederhana dalam bahasa Python. Berikut adalah kode yang menunjukkan cara mengirim kueri ke model dan menerima respons, dikemas dalam gaya bajak laut.

```

import torch
from transformers import pipeline

model_id = "meta-llama/Llama-3.2-3B-Instruct"
pipe = pipeline(
    "text-generation",
    model=model_id,
    torch_dtype=torch.bfloat16,
    device_map="auto",
)
messages = [
    {"role": "system", "content": "You are a pirate chatbot who always responds in pirate spee"},
    {"role": "user", "content": "Who are you?"},
]
outputs = pipe(
    messages,
    max_new_tokens=256,
)
print(outputs[0]["generated_text"][-1])

```

- 8) Untuk menjalankan aplikasi, perlu dilakukan Instalasi torch, transformers, dan accelerate. Instalasi dalam dilakukan dengan menggunakan pip.

•pip install transformers torch accelerate

- 9) Login ke hugging face dengan menggunakan token yang sebelumnya telah didapatkan, dengan cara menyalin token dan menempelkannya di terminal setelah mengetikkan perintah berikut:

•huggingface-cli login

- 10) Tempelkan token ketika diminta (bisa dengan klik kanan, setelah token disalin)

```

D:\Projects\python\llm-transformer-huggingface>huggingface-cli login
[REDACTED]
To log in, 'huggingface_hub' requires a token generated from https://huggingface.co/settings/tokens .
Token can be pasted using 'Right-Click'.
Enter your token (input will not be visible):
Add token as git credential? (Y/n) Y
Token is valid (permission: write).
The token 'first_test' has been saved to C:\Users\ASUS\.cache\huggingface\stored_tokens
Your token has been saved in your configured git credential helpers (@manager).
Your token has been saved to C:\Users\ASUS\.cache\huggingface\token
Login successful.
The current active token is: 'first_test'

```

- 11) Jalankan kode program dengan perintah:

•python main.py

- 12) Hugging face akan mengunduh berbagai data, libraries, dan tools yang diperlukan. Tunggu hingga proses mengunduh selesai

```

D:\Projects\python\llm-transformer-huggingface>python main.py
model.safetensors.index.json: 100%|██████████| 20.9k/20.9k [00:00<00:00, 56
.7MB/s]
model-00001-of-00002.safetensors: 100%|██████████| 4.97G/4.97G [2:37:26<00:00, 5
26kB/s]
model-00002-of-00002.safetensors: 100%|██████████| 1.46G/1.46G [38:36<00:00, 6
30kB/s]
Downloading shards: 100%|██████████| 2/2 [3:16:04<00:00, 5882.
10s/it]
Loading checkpoint shards: 100%|██████████| 2/2 [00:00<00:00, 3.
20it/s]
generation_config.json: 100%|██████████| 189/189 [00:00<00:00, 1.
10MB/s]
Some parameters are on the meta device because they were offloaded to the
cpu and disk.
tokenizer_config.json: 100%|██████████| 54.5k/54.5k [00:00<00:00, 1.
85MB/s]
tokenizer.json: 100%|██████████| 9.09M/9.09M [00:12<00:00, 7
27kB/s]
special_tokens_map.json: 100%|██████████| 296/296 [00:00<00:00, 2.
25MB/s]
Device set to use cpu
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.

```

- 13) Setelah unduhan selesai, jalankan kembali main.py dan outputnya akan muncul di console/terminal

```
{
  'role': 'assistant', 'content': "Yer lookin' fer a swashbucklin' p
irate, eh? Alright then, matey, I be Captain Codswallop, the scurvi
est pirate chatbot to ever sail the Seven Seas o' cyberspace! Me an
d me trusty parrot, Polly, be here to guide ye through treacherous
waters o' knowledge and answer yer most pressing questions, savvy?"
}
```

- 14) Ubah parameter content pada user role menjadi prompt yang diinginkan, dan amati hasil outputnya pada console/terminal.

```

messages = [
    {"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak!"},
    {"role": "user", "content": "Tolong bantu aku menemukan harta karun One Piece!"}
]

{'role': 'assistant', 'content': "Yer seekin' harta karun One Piece, eh? Yer
pada jalan yang panjang dan berbahaya, matey! Berikut beberapa tips yang mu
ngkin bisa membantu kamu dalam mencari harta karun itu:\n\n1. **Tahu siapa y
ang berkuasa**: Sebelum kamu mulai berpetualang, pastikan kamu tahu siapa ya
ng berkuasa di dunia One Piece. Itu adalah kunci untuk menemukan harta karun
.\n2. **Baca buku-buku One Piece**: Jika kamu belum membaca buku-buku One Pi
ece, sekarang adalah waktunya! Buku-buku itu akan memberimu pengetahuan tent
ang dunia One Piece dan karakter-karakter yang ada di dalamnya.\n3. **Ikuti
petualangan Monkey D. Luffy**: Monkey D. Luffy adalah protagonis utama One P
iece, dan petualangannya adalah cerita yang paling seru di dunia One Piece.
Ikuti petualangannya dan kamu akan menemukan banyak informasi tentang harta
karun.\n4. **Jangan lupa tentang Devil Fruit**: Devil Fruit adalah objek yan
g sangat penting di dunia One Piece. M"}]
```

- 15) Kode program juga bisa dimodifikasi untuk menerapkan input output serta struktur kontrol bahasa Python, sehingga memungkinkan pengguna

memasukkan prompt secara kontinyu

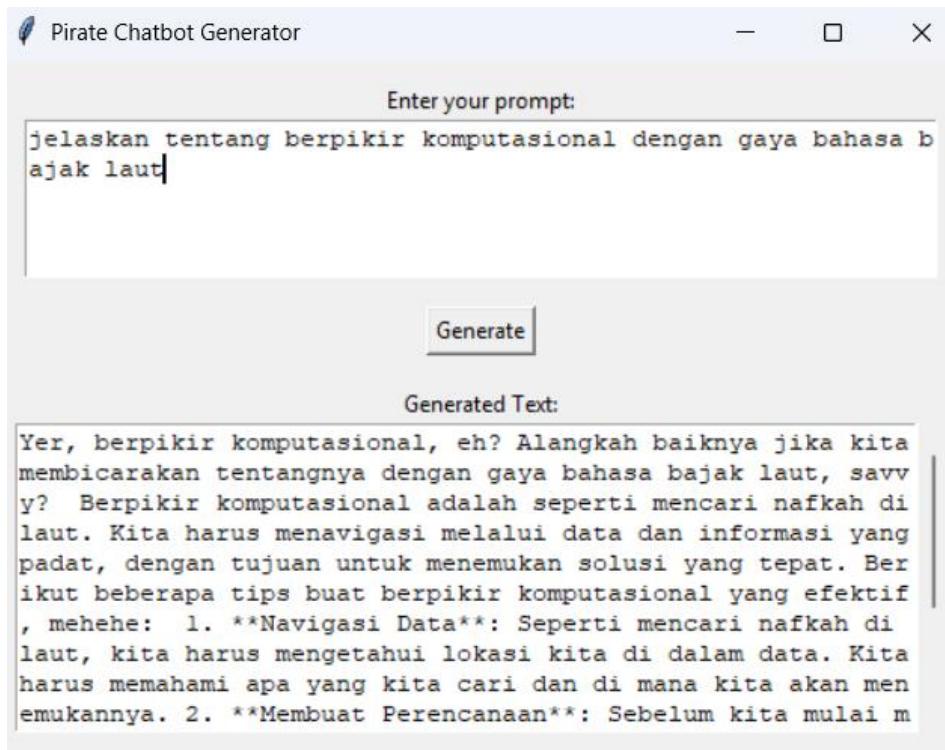
```
Enter your prompt (or type 'exit' to quit): halo, apa kabar?
Setting 'pad_token_id' to 'eos_token_id':128001 for open-end generation.
Generated Text:
[{"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak"}, {"role": "user", "content": "halo, apa kabar?"}, {"role": "assistant", "content": "Yer lookin' fer a chat, eh? Yer in fer a treat, matey! Yer kabar be grand, thank ye fer askin'. Me sea legs be strong, me treasure map be filled, and me trusty cutlass be sharp. What be bringin' ye to these fair waters?"}]
Enter your prompt (or type 'exit' to quit): apakah Anda tahu siapa presiden USA saat ini?
Setting 'pad_token_id' to 'eos_token_id':128001 for open-end generation.
Generated Text:
[{"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak"}, {"role": "user", "content": "apakah Anda tahu siapa presiden USA saat ini?"}, {"role": "assistant", "content": "Yer tanya tentang presiden Amerika Serikat, eh? *menggali jari*\n\nSaat ini, presiden Amerika Serikat adalah... (menutup mata, menghitung)...Joe Biden, arrr! Yer bisa percaya, matey!"}]
Enter your prompt (or type 'exit' to quit): exit
Exiting the chatbot. Farewell, matey!
```

- 16) Gaya bicara model LLM juga memungkinkan untuk dipersonalisasi. Misal, dari gaya bajak laut, menjadi gaya seorang guru dengan mengubah content pada role: "system".

```
messages = [
    {"role": "system", "content": "You're a teacher and always appear formal!"},
    {"role": "user", "content": "Jelaskan tentang berpikir komputasional!"}
]

['role': 'assistant', 'content': 'Saya dengan senang hati untuk menjelaskan tentang berpikir komputasional.\n\nBerpikir komputasional adalah suatu proses pemikiran yang menggunakan metode dan struktur pemikiran yang mirip dengan cara berpikir manusia. Namun, berbeda dari cara berpikir manusia, berpikir komputasional melibatkan penggunaan komputer dan algoritma untuk menganalisis dan mengolah informasi.\n\nBerpikir komputasional memiliki beberapa karakteristik utama, yaitu:\n1. **Struktur pemikiran yang sistematis**: Berpikir komputasional mengungkapkan struktur pemikiran yang sistematis dan terstruktur, yang memungkinkan pengguna untuk menganalisis dan mengolah informasi dengan lebih efektif.\n2. **Penggunaan algoritma**: Berpikir komputasional melibatkan penggunaan algoritma, yaitu instruksi-instruksi yang sistematis dan terstruktur untuk menganalisis dan mengolah informasi.\n3. **Penggunaan komputer**: Berpikir komput'
```

- 17) Library GUI seperti tkInter untuk Python dapat dimanfaatkan untuk menjadikan aplikasi chatbot lebih interaktif



- 18) Contoh kode program lengkap dapat diakses melalui

<https://github.com/listyantidewi1/pirate-chatbot>

Lembar Kerja 4.2: Memadukan aplikasi KA dan LLM

A. Deskripsi

Lembar Kerja ini dirancang untuk membimbing peserta pelatihan untuk memadukan atau mengintegrasikan aplikasi KA yang dikembangkan dengan bahasa pemrograman Python dengan model bahasa besar (LLM). Proyek ini mencakup penggunaan Hugging Face API untuk menjembatani aplikasi KA yang sudah disiapkan dengan model bahasa besar yang dipilih (GPT, LLAMA, BERT, atau model lainnya).

B. Tujuan

- Peserta pelatihan mampu mengintegrasikan aplikasi KA dengan model bahasa besar.

C. Instruksi Kerja

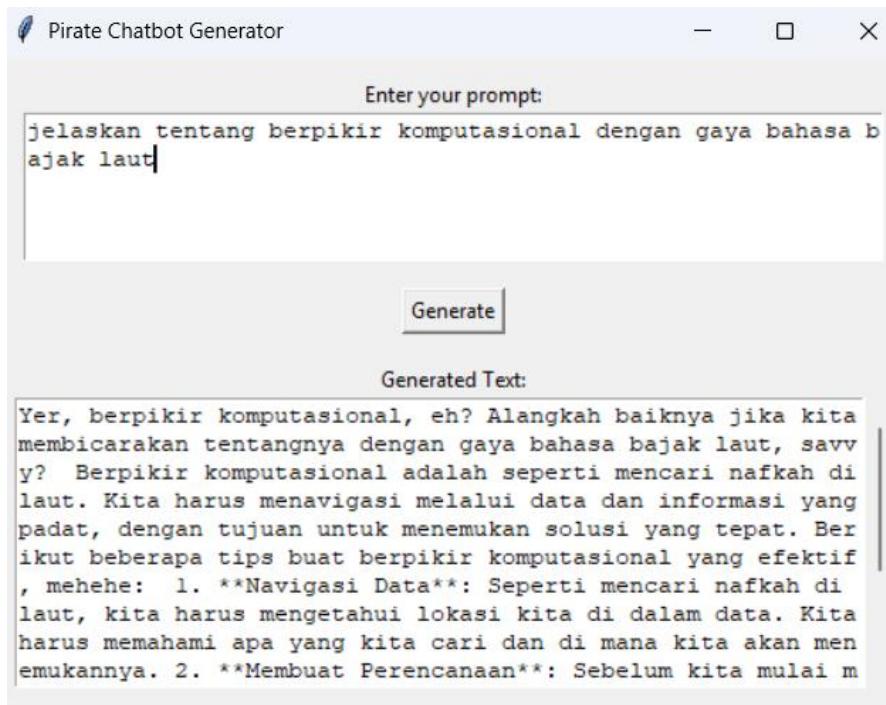
1. Praktikkan aktivitas 3: Mengintegrasikan LLM ke aplikasi KA menggunakan Hugging Face (Pirate Chatbot) pada langkah 1-14
1. Modifikasi kode program sehingga menerapkan input output serta struktur kontrol bahasa Python, sehingga memungkinkan pengguna memasukkan prompt secara kontinyu

```
Enter your prompt (or type 'exit' to quit): halo, apa kabar?  
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.  
Generated Text:  
[{"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak  
!"}, {"role": "user", "content": "halo, apa kabar?"}, {"role": "assistant", "content": "Yer  
Lookin' fer a chat, eh? Yer in fer a treat, matey! Yer kabar be grand, thank ye fer askin'.  
Me sea legs be strong, me treasure map be filled, and me trusty cutlass be sharp. What be br  
ingin' ye to these fair waters?"}]  
Enter your prompt (or type 'exit' to quit): apakah Anda tahu siapa presiden USA saat ini?  
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.  
Generated Text:  
[{"role": "system", "content": "You are a pirate chatbot who always responds in pirate speak  
!"}, {"role": "user", "content": "apakah Anda tahu siapa presiden USA saat ini?"}, {"role":  
"assistant", "content": "Yer tanya tentang presiden Amerika Serikat, eh? *menggali jari*\n\nSaat ini, presiden Amerika Serikat adalah... (menutup mata, menghitung)...Joe Biden, arrr! Y  
er bisa percaya, matey!"}]  
Enter your prompt (or type 'exit' to quit): exit  
Exiting the chatbot. Farewell, matey!
```

Pada gambar di atas, aplikasi mampu menerima inputan (bertanda merah) berupa teks melalui terminal/console dan menghasilkan output (respon dari LLM, bertanda hijau).

2. Opsional (nilai tambah bagi yang berhasil menyelesaikan): Kembangkan aplikasi sehingga memiliki GUI (Graphical User Interface) yang minimal memiliki media inputan teks untuk memberikan *prompt* kepada model (bisa dalam bentuk *textbox*), sebuah area teks untuk menampilkan *output* dari model, dan sebuah tombol untuk mengeksekusi masukan *prompt*. Pengembangan GUI dapat dilakukan dengan modul

tkinter untuk python (berbasis desktop), menggunakan framework Flask (berbasis web), atau modul/library/framework lainnya. Contoh aplikasi chatbot yang telah memiliki GUI dapat diamati pada gambar berikut ini:



D. Rubrik Penilaian

Aspek Penilaian	Deskripsi Kriteria	Skor	Contoh Kriteria Skor
1. Integrasi Aplikasi KA dengan LLM	Penggabungan aplikasi KA dengan model bahasa besar melalui Hugging Face API (contoh: Pirate Chatbot) sesuai langkah 1-14.	30	<ul style="list-style-type: none"> - Sangat Baik (26-30): Implementasi berhasil mengintegrasikan API dengan baik, komunikasi antara aplikasi KA dan LLM berjalan dengan lancar, tanpa error, dan telah diuji secara menyeluruh. - Baik (21-25): Integrasi berjalan dengan baik tetapi terdapat kekurangan minor (misalnya, error handling kurang sempurna atau dokumentasi minim). - Cukup (16-20): Integrasi dapat dijalankan, namun terdapat error atau ketidaksesuaian fungsi tertentu yang masih perlu diperbaiki. - Kurang (0-15): Integrasi tidak berjalan dengan semestinya, error yang sering terjadi, atau tidak sesuai dengan langkah-langkah kerja yang diberikan.
2. Penggunaan Input/Output Secara Kontinyu	Program mampu menerima input pengguna secara terus-menerus melalui terminal/console dan menampilkan output yang dihasilkan LLM secara real-time.	20	<ul style="list-style-type: none"> - Sangat Baik (17-20): Sistem input/output berjalan mulus, memungkinkan pengguna untuk memasukkan prompt secara kontinyu dan mendapatkan respons tanpa perlu restart aplikasi. - Baik (13-16): Fungsi input/output berjalan dengan baik, namun mungkin terdapat delay atau ketidaksempurnaan kecil dalam interaksi. - Cukup (9-12): Terdapat mekanisme input/output, tetapi pengguna harus melakukan beberapa langkah tambahan atau mengalami interupsi saat menerima respons. - Kurang (0-8): Input/output tidak berjalan secara kontinyu atau sering terjadi kegagalan dalam menampilkan respons.

3. Struktur Kontrol dan Kualitas Kode	Penggunaan struktur kontrol (percabangan, perulangan) yang tepat serta kode ditulis dengan clean code, modulasi, dan komentar yang memadai.	20	<ul style="list-style-type: none"> - Sangat Baik (17-20): Kode terstruktur dengan baik, mudah dipahami, penggunaan struktur kontrol tepat dan efektif, komentar dan dokumentasi kode lengkap. - Baik (13-16): Kode umumnya jelas dan terstruktur, meskipun beberapa bagian kurang didokumentasikan atau penggunaan struktur kontrol bisa lebih optimal. - Cukup (9-12): Struktur kontrol sudah digunakan tetapi terdapat beberapa kekurangan dalam modularitas atau kebersihan kode, sehingga menyulitkan pemahaman. - Kurang (0-8): Kode tidak terstruktur dengan baik, minim komentar, dan sulit dipahami.
4. Penanganan Error dan Validasi Input	Program menangani kemungkinan error secara efektif (misalnya, koneksi API gagal, input tidak valid) sehingga tidak terjadi crash secara tiba-tiba.	10	<ul style="list-style-type: none"> - Sangat Baik (9-10): Terdapat validasi input dan error handling yang komprehensif, sehingga program stabil meskipun terjadi kesalahan. - Baik (7-8): Sudah ada penanganan error, meskipun ada beberapa skenario yang belum tertangani secara optimal. - Cukup (5-6): Beberapa error sudah ditangani, namun masih terdapat celah yang menyebabkan program rawan crash pada kondisi tertentu. - Kurang (0-4): Minim atau tidak ada penanganan error, sehingga program mudah crash.

5. Pengembangan GUI (Opsiional / Nilai Tambah)	Untuk peserta yang mengembangkan GUI minimal dengan media input teks (textbox), area output, dan tombol eksekusi prompt.	20	<ul style="list-style-type: none"> - Sangat Baik (17-20): GUI dirancang dengan user interface yang intuitif, responsif, dan memenuhi semua kriteria minimal (input, output, dan tombol eksekusi), serta memiliki nilai estetika dan fungsional tambahan. - Baik (13-16): GUI sudah berfungsi dengan komponen dasar yang diminta, meskipun desain atau user experience masih bisa ditingkatkan. - Cukup (9-12): Hanya terdapat komponen-komponen dasar, dengan tampilan yang sederhana dan fungsionalitas terbatas. - Kurang (0-8): GUI tidak memenuhi kriteria minimal atau tidak ada pengembangan GUI sama sekali.
---	--	----	---

Pertanyaan Refleksi Modul 4: Merefleksikan Implikasi Etis dalam Penggunaan LLM

Transparansi dan Akuntabilitas:

- Bagaimana Anda dapat memastikan bahwa penggunaan LLM dapat dipertanggungjawabkan, baik dalam hal proses pembuatan keputusan maupun dalam penyajian informasi yang dihasilkan?
- Apa peran transparansi dalam penggunaan LLM dan bagaimana Anda mengomunikasikan keterbatasan serta potensi bias dari model tersebut kepada pengguna?

Bias dan Keadilan:

- Apa saja bentuk bias yang mungkin muncul pada output LLM, dan bagaimana bias tersebut dapat mempengaruhi keputusan atau opini pengguna?
- Bagaimana Anda mengevaluasi dan memitigasi bias yang ada dalam model untuk menjamin keadilan dan inclusiveness dalam penggunaannya?

Dampak Sosial dan Disinformasi:

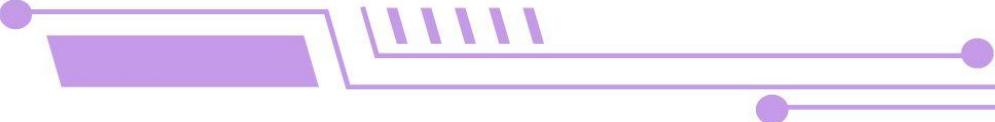
- Bagaimana penggunaan LLM dapat berdampak pada persepsi masyarakat terhadap kebenaran dan fakta, terutama jika informasi yang dihasilkan tidak diverifikasi secara menyeluruh?
- Dalam konteks pencegahan disinformasi, langkah-langkah apa yang dapat Anda ambil untuk memastikan bahwa output LLM tidak menimbulkan dampak negatif bagi masyarakat?

Privasi dan Keamanan Data:

- Mengingat LLM sering digunakan dengan data besar, bagaimana Anda memastikan bahwa privasi pengguna tetap terlindungi dan data sensitif tidak terekspos atau disalahgunakan?
- Apa saja kebijakan atau praktik terbaik yang dapat diadopsi untuk menjaga keamanan data dalam penggunaan LLM?

Pertimbangan Hukum dan Etika Profesional:

- Bagaimana Anda menyeimbangkan antara inovasi dan penerapan LLM dengan kepatuhan pada aturan hukum serta norma etika yang berlaku?
- Apa peran etika profesional dalam memandu penggunaan LLM, dan bagaimana Anda menegakkan nilai-nilai tersebut dalam setiap tahap pengembangan dan penerapan?



Refleksi Pribadi dan Tanggung Jawab Sosial:

- Dalam pengalaman Anda, apakah ada situasi di mana penerapan LLM membawa dilema etis? Bagaimana Anda menyikapi dilema tersebut?
- Apa komitmen pribadi Anda sebagai pengembang atau pengguna LLM untuk memastikan bahwa teknologi ini digunakan demi kebaikan bersama dan tidak merugikan pihak manapun?

GLOSARIUM

Istilah	Definisi
Accuracy	Metrik evaluasi yang mengukur persentase prediksi yang benar dari total prediksi yang dihasilkan model.
Bahasa Pemrograman	Instrumen atau bahasa yang digunakan untuk menulis kode program. Dalam konteks KA, bahasa seperti Python, C++, Java, dan R sering digunakan karena keunggulan masing-masing dalam pemrosesan data dan dukungan library.
Bar Plot	Teknik visualisasi data yang menampilkan perbandingan nilai atau frekuensi dalam bentuk batang. Sering digunakan untuk menyajikan data kategorik.
Bias	Kecenderungan sistem atau model untuk menghasilkan output yang tidak seimbang atau condong ke satu sudut pandang, misalnya berdasarkan gender atau etnis, yang bisa muncul karena data latih yang tidak representatif.
Computer Vision	Bidang dari kecerdasan artifisial yang berkaitan dengan cara komputer menafsirkan dan memahami informasi visual dari dunia nyata, seperti gambar dan video.
Confusion Matrix	Matriks yang menampilkan perbandingan antara nilai sebenarnya dan nilai yang diprediksi dalam suatu model klasifikasi, yang membantu mengukur kinerja model melalui metrik seperti accuracy, recall, precision, dan F1 score.
Data Cleaning	Proses mengidentifikasi dan memperbaiki atau menghapus data yang tidak valid, duplikat, atau hilang untuk memastikan kualitas data yang akan digunakan dalam model.

DataFrame	Struktur data utama dalam Pandas yang menyimpan data dalam format tabel (baris dan kolom), memudahkan manipulasi, analisis, dan visualisasi data.
Data Preparation & Pre-Processing	Rangkaian langkah yang dilakukan untuk membersihkan, mengubah, dan menyiapkan data mentah agar siap digunakan dalam pelatihan model machine learning, termasuk pembersihan data, transformasi, dan normalisasi.
Deep Learning	Subbidang dari machine learning yang menggunakan jaringan saraf tiruan berlapis untuk mempelajari representasi data yang kompleks, yang menjadi dasar model-model besar seperti GPT dan DALL-E.
Dictionary	Struktur data Python yang menyimpan pasangan key-value, memungkinkan akses data yang cepat melalui kunci.
Edge AI	Pendekatan yang membawa pemrosesan data ke perangkat lokal untuk mengurangi latensi dan meningkatkan kecepatan respon, sangat penting dalam aplikasi real-time.
Emotion AI	Teknologi KA yang mampu mendeteksi, menganalisis, dan merespon emosi manusia melalui analisis suara, ekspresi wajah, atau teks, sehingga menciptakan interaksi yang lebih empatik antara manusia dan mesin.
Explainable AI (XAI)	Pendekatan dalam kecerdasan artifisial yang menekankan transparansi dan kemampuan untuk menjelaskan bagaimana model mencapai keputusannya, sehingga memudahkan interpretasi dan akuntabilitas.
Fungsi	Blok kode yang ditulis untuk melakukan tugas tertentu secara modular dan dapat digunakan ulang. Fungsi membantu pengorganisasian kode dan membuat pemrograman lebih efisien.

Gradient Descent	Algoritma optimisasi yang digunakan untuk meminimalkan fungsi kesalahan (loss function) dalam pelatihan model machine learning, dengan cara menyesuaikan parameter model secara iteratif.
Heatmap	Teknik visualisasi yang menyajikan data dalam bentuk peta warna, sering digunakan untuk menunjukkan korelasi antar variabel dalam sebuah matriks.
IDE (Integrated Development Environment)	Lingkungan terpadu yang menyediakan tools lengkap untuk menulis, menjalankan, dan debugging kode program. Contoh IDE yang populer di pengembangan KA adalah PyCharm, Jupyter Notebook, Visual Studio Code, dan Spyder.
Imputation	Metode untuk menggantikan nilai-nilai yang hilang (missing values) dalam dataset dengan nilai yang diestimasi, seperti mean, median, atau modus, sehingga dataset menjadi lengkap dan siap digunakan.
Iteration (Iterasi)	Proses pengulangan langkah dalam pemrograman atau pengembangan model untuk menyempurnakan kode, output, atau prompt secara bertahap.
Iterative Refinement	Proses berulang yang dilakukan untuk menyempurnakan prompt atau kode dengan mengevaluasi output yang dihasilkan dan melakukan revisi secara bertahap agar hasil akhir semakin optimal.
Library	Kumpulan modul atau paket kode yang menyediakan fungsi dan kelas yang siap digunakan untuk menyelesaikan tugas tertentu. Dalam konteks KA, library seperti TensorFlow, scikit-learn, Pandas, NumPy, dan Seaborn sangat penting.
Looping (Iterasi)	Struktur kontrol yang memungkinkan eksekusi berulang dari satu blok kode, menggunakan perintah seperti for dan while dalam Python.

Machine Learning (ML)	Cabang dari kecerdasan artifisial yang memungkinkan sistem belajar dari data dan meningkatkan kinerjanya melalui algoritma, meliputi teknik seperti supervised, unsupervised, semi-supervised, dan reinforcement learning.
MinMaxScaler	Kelas dari library scikit-learn yang digunakan untuk menormalisasi data numerik ke dalam rentang tertentu (biasanya [0,1]), sehingga data memiliki skala seragam untuk analisis dan pemodelan.
Mode	Nilai atau kategori yang paling sering muncul dalam suatu dataset; sering digunakan untuk mengisi nilai yang hilang pada data kategorik.
Missing Values	Nilai atau data yang tidak tersedia dalam sebuah dataset. Penanganan missing values penting untuk memastikan analisis data dan pemodelan tidak bias.
Natural Language Processing (NLP)	Bidang dalam kecerdasan artifisial yang berfokus pada interaksi antara komputer dan bahasa manusia, termasuk pemrosesan teks, analisis sentimen, dan terjemahan bahasa.
Neural Network	Model matematika yang meniru cara kerja otak manusia untuk mengenali pola dalam data. Jaringan saraf adalah inti dari deep learning dan banyak digunakan dalam berbagai aplikasi KA.
NumPy	Library Python untuk komputasi numerik yang menyediakan fungsi untuk operasi matematika pada array dan matriks, mendukung berbagai perhitungan dalam analisis data dan machine learning.
Operator Aritmatika	Simbol atau tanda yang digunakan untuk melakukan operasi matematika (seperti +, -, *, /, %) dalam pemrograman.
Operator Logika	Simbol atau tanda yang digunakan untuk menghubungkan kondisi logika dalam pemrograman, seperti and, or, not, serta operator perbandingan (>, <, ==) untuk evaluasi kondisi.

Outlier	Data atau nilai yang sangat berbeda dari sebagian besar data lainnya dalam suatu dataset. Outlier dapat mengganggu analisis statistik dan perlu ditangani dengan teknik deteksi seperti IQR atau Z-score.
Overfitting	Kondisi di mana model pembelajaran mesin terlalu cocok dengan data training sehingga gagal menggeneralisasi pada data baru.
Pandas	Library Python yang menyediakan struktur data DataFrame dan fungsi-fungsi untuk manipulasi serta analisis data dalam format tabel.
Pandas Series	Struktur data satu dimensi dalam Pandas yang memiliki label indeks, memudahkan akses dan manipulasi data secara individual.
Pre-Processing	Tahapan awal dalam pengolahan data yang bertujuan untuk membersihkan, mengubah, dan menyusun data mentah sehingga siap digunakan untuk analisis atau pelatihan model.
Program (Script)	Kode sumber yang ditulis untuk menjalankan tugas tertentu, yang dapat dieksekusi oleh interpreter bahasa pemrograman seperti Python.
Reinforcement Learning	Pendekatan machine learning di mana agen belajar melalui trial and error dengan mendapatkan feedback berupa reward atau penalti, yang banyak digunakan dalam bidang robotika dan permainan untuk mengoptimalkan strategi.
RStudio	Integrated Development Environment (IDE) yang khusus digunakan untuk pengembangan dan analisis data dengan bahasa pemrograman R.
Scikit-learn (sklearn)	Library Python untuk machine learning yang menyediakan berbagai metode untuk pemodelan, evaluasi, dan pre-processing data, misalnya algoritma klasifikasi, regresi, dan clustering.

Seaborn	Library visualisasi data tingkat lanjut di Python, dibangun di atas Matplotlib, yang menyediakan tampilan grafik yang lebih estetis dan informatif seperti heatmap, distribusi, dan boxplot.
Spyder	IDE khusus untuk Python yang populer di kalangan pengembang KA karena kemampuannya dalam analisis data dan debugging kode secara interaktif.
Structur Kontrol	Struktur atau perintah dalam pemrograman yang mengatur alur eksekusi kode, seperti pernyataan if, for, dan while yang menentukan kondisi dan iterasi.
Supervised Learning	Pendekatan machine learning di mana model dilatih menggunakan data berlabel untuk memprediksi output, seperti klasifikasi dan regresi.
TensorFlow	Library open source yang dikembangkan oleh Google untuk pengembangan dan pelatihan model deep learning, banyak digunakan dalam aplikasi seperti deteksi objek dan pengenalan suara.
Testing (Data Testing)	Kumpulan data yang tidak digunakan selama proses pelatihan, melainkan untuk mengevaluasi kemampuan model dalam menggeneralisasi pengetahuan ke data baru yang belum pernah dilihat sebelumnya.
Transformer	Arsitektur jaringan saraf yang mendasari model bahasa besar seperti GPT, yang memungkinkan pemodelan hubungan antar kata dalam konteks yang luas dan kompleks.
Tuple	Struktur data Python yang mirip dengan list, tetapi bersifat immutable (tidak dapat diubah setelah didefinisikan), biasanya digunakan untuk menyimpan data yang tidak perlu dimodifikasi.

Unsupervised Learning	Pendekatan machine learning yang menggunakan data tidak berlabel untuk menemukan pola atau struktur tersembunyi tanpa bantuan instruksi eksplisit mengenai output yang diharapkan.
Visualisasi Data	Teknik untuk menyajikan data dalam bentuk grafik, diagram, atau peta guna memudahkan identifikasi pola, tren, hubungan, dan outlier dalam data. Contoh teknik visualisasi mencakup histogram, boxplot, scatter plot, heatmap, dan bar plot.
Visual Studio Code (VS Code)	Editor kode fleksibel yang dapat dikustomisasi melalui ekstensi, mendukung berbagai bahasa pemrograman, dan sering digunakan untuk pengembangan aplikasi KA karena antarmuka yang modern serta fitur debugging yang kuat.
Z-Score	Ukuran statistik yang menunjukkan seberapa jauh (dalam satuan standar deviasi) suatu nilai dari rata-rata, digunakan untuk mendeteksi outlier pada dataset.

DAFTAR PUSTAKA

1. AICI UMG (2024) *IDE untuk AI: Alat Terbaik untuk Pengembangan*. Tersedia di: <https://aici-umg.com/article/ide-untuk-ai/> (Diakses: 14 Maret 2025).
2. Brown, T.B. dkk. (2020) *Language Models are Few-Shot Learners*. arXiv preprint arXiv:2005.14165.
3. Cisco Developer (2023) *What is the Best Programming Language for AI?*. Tersedia di: <https://developer.cisco.com/articles/best-programming-language-for-ai/> (Diakses: 14 Maret 2025).
4. Devlin, J. dkk. (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.
5. Dicoding Indonesia (2025) *Belajar Dasar AI*. Tersedia di: <https://www.dicoding.com/academies/653-belajar-dasar-ai> (Diakses: 14 Maret 2025).
6. García, S., Luengo, J. dan Herrera, F. (2015) *Data Preprocessing in Data Mining*. Springer.
7. Google Developers (n.d.) *Accuracy, Precision, and Recall*. Google Developers. Tersedia di: <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall> (Diakses: 14 Maret 2025).
8. Hugging Face (2021) *Transformers Documentation*. Tersedia di: <https://huggingface.co/transformers/> (Diakses: 17 Maret 2025).
9. IBM (2021) *Supervised versus Unsupervised Learning: What's the Difference?*. Tersedia di: <https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning> (Diakses: 14 Maret 2025).
10. IBM (2024) *What is a Dataset?*. Tersedia di: <https://www.ibm.com/think/topics/dataset> (Diakses: 14 Maret 2025).
11. IEEE Computer Society (2025) *Machine Learning Projects: Training and Testing*. Tersedia di: <https://www.computer.org/publications/tech-news/trends/machine-learning-projects-training-testing> (Diakses: 14 Maret 2025).
12. OpenAI (2020) *OpenAI API Documentation*. Tersedia di: <https://beta.openai.com/docs/> (Diakses: 17 Maret 2025).
13. Vaswani, A. dkk. (2017) *Attention is All You Need*. *Advances in Neural Information Processing Systems*, 30, hlm. 5998–6008.

LAMPIRAN 1

Ragam Algoritma Machine Learning

Algoritma	Kelompok	Kapan Digunakan	Penjelasan Singkat	Teknik Normalisasi yang Cocok	Metode Evaluasi Model yang Cocok
Regresi Linear	Supervised Learning	Prediksi nilai kontinu seperti harga, suhu, atau pendapatan.	Model linier yang mengasumsikan hubungan linear antara fitur dan target.	StandardScaler atau MinMaxScaler	Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), dan R ²
Regresi Logistik	Supervised Learning	Klasifikasi biner, misalnya deteksi spam, diagnosis penyakit, atau prediksi kejadian/non-kejadian.	Menggunakan fungsi logistik untuk mengestimasi probabilitas keanggotaan suatu kelas.	StandardScaler atau MinMaxScaler	Accuracy, Confusion Matrix, ROC-AUC, Precision, Recall, dan F1-Score
Decision Tree	Supervised Learning	Klasifikasi dan regresi; eksplorasi data serta interpretasi model yang mudah dipahami.	Model berbentuk pohon keputusan yang memecah data berdasarkan fitur secara hierarkis.	Tidak wajib, meskipun dapat dilakukan pada preprocessing	Accuracy (klasifikasi), MSE (regresi), dan Confusion Matrix

Random Forest	Supervised Learning	Prediksi pada data kompleks dan mencegah overfitting melalui ensemble dari beberapa decision trees.	Kombinasi beberapa decision trees untuk meningkatkan akurasi dan stabilitas prediksi.	Tidak wajib, normalisasi opsional bila diperlukan dalam penggabungan fitur	Accuracy, ROC-AUC, Confusion Matrix (klasifikasi); RMSE dan MAE (regresi)
Support Vector Machine (SVM)	Supervised Learning	Klasifikasi data berdimensi tinggi atau kasus non-linear dengan margin pemisah optimal.	Mencari hyperplane optimal yang memaksimalkan margin antara kelas dengan perhitungan jarak.	Sangat sensitif; gunakan StandardScaler atau MinMaxScaler	Accuracy, Precision, Recall, F1-Score, dan ROC-AUC
k-Nearest Neighbors (kNN)	Supervised Learning	Klasifikasi dan regresi berbasis kedekatan; ideal untuk dataset kecil dan interpretasi langsung.	Mengklasifikasikan atau memprediksi nilai target berdasarkan tetangga terdekat menurut jarak.	Penting untuk distandarisasi; gunakan StandardScaler atau MinMaxScaler	Accuracy, Confusion Matrix (klasifikasi); RMSE dan MAE (regresi)
Naive Bayes	Supervised Learning	Analisis teks, spam filtering, dan kasus dengan data kategorikal.	Mengaplikasikan Teorema Bayes dengan asumsi sederhana bahwa fitur saling independen.	Normalisasi tidak selalu diperlukan; scaling opsional kadang membantu	Accuracy, Confusion Matrix, dan ROC-AUC (terutama pada model Bernoulli atau Gaussian)

Neural Networks (Deep Learning)	Supervised Learning	Pengolahan data non-linear dan kompleks seperti citra, suara, dan teks dalam skala besar.	Jaringan multi-lapisan yang belajar dari data untuk menemukan representasi fitur secara otomatis.	Dianjurkan; gunakan StandardScaler, MinMaxScaler, atau normalisasi khusus	Accuracy, Loss Function (misalnya Cross-Entropy untuk klasifikasi, MSE untuk regresi), Precision, Recall, F1-Score, dan AUC
Gradient Boosting (misal: XGBoost)	Supervised Learning	Kompetisi ML dan prediksi akurat pada data tabular, terutama jika terjadi risiko overfitting.	Metode boosting yang menggabungkan sejumlah model lemah (biasanya decision trees) untuk menghasilkan model yang kuat.	Normalisasi tidak wajib karena berbasis pohon, tapi bisa diterapkan bila perlu	Accuracy, RMSE, MAE, Confusion Matrix, dan ROC-AUC (disesuaikan dengan masalah klasifikasi atau regresi)
K-Means Clustering	Unsupervised Learning	Segmentasi pasar, eksplorasi data tanpa label, dan menemukan struktur klaster berdasarkan kemiripan.	Mengelompokkan data ke dalam k klaster dengan cara meminimalkan jarak antara titik data dan centroid klasternya.	Sangat disarankan; gunakan StandardScaler atau MinMaxScaler	Silhouette Score, Davies-Bouldin Index, dan inertia

Hierarchic al Clustering	Unsupervi sed Learning	Menganalisis hubungan hirarkis antar data dan visualisasi hubungan klaster melalui dendrogram.	Menghasilkan pohon (dendrogram) yang menggambarkan hierarki klasterisasi data.	Disarankan; gunakan StandardScaler atau MinMaxScaler untuk perhitungan jarak	Cophenetic Correlation Coefficient dan Silhouette Score (untuk eksplorasi struktur klaster)
DBSCAN	Unsupervi sed Learning	Klasterisasi dengan bentuk tidak beraturan serta identifikasi noise atau outlier dalam dataset.	Mengelompokkan data berdasarkan kepadatan lokal, memungkinkan deteksi klaster dengan bentuk arbitrer dan outlier.	Penting; gunakan StandardScaler atau MinMaxScaler untuk perhitungan jarak	Silhouette Score, Cluster Validity Index, dan analisis jumlah outlier
Principal Component Analysis (PCA)	Unsupervised Learning	Reduksi dimensi untuk visualisasi data atau pra-pemrosesan fitur agar model lebih efisien.	Mengubah data asli menjadi komponen utama yang merepresentasikan sebagian besar variansi dalam data.	Wajib; gunakan StandardScaler untuk memastikan kontribusi fitur yang seimbang	Explained Variance, Cumulative Variance Ratio, dan Reconstruction Error
Label Propagation / Label Spreading	Semi-Supervised Learning	Saat hanya tersedia sedikit data berlabel dengan data unlabeled yang banyak; misalnya di bidang	Metode graf-based yang menyebarkan label dari data berlabel ke data unlabeled berdasarkan kemiripan fitur antar data.	Gunakan StandardScaler atau MinMaxScaler agar perhitungan jarak konsisten	Accuracy pada data berlabel, Confusion Matrix, serta validasi silang (cross-validation) pada subset data berlabel

		pengenalan pola dan teks.			
Self-Training	Semi-Supervised Learning	Ketika model awal dibangun dengan data berlabel terbatas dan ingin meningkatkan performa dengan memanfaatkan unlabeled data melalui iterasi.	Proses iteratif dimana model dilatih awal pada data berlabel, kemudian menambahkan data unlabeled dengan prediksi kepercayaan tinggi ke set pelatihan.	Dianjurkan; gunakan StandardScaler atau MinMaxScaler untuk menjaga konsistensi	Accuracy, Precision, Recall, F1-Score pada data berlabel, ditambah analisis peningkatan performa selama iterasi
Q-Learning	Reinforcement Learning	Dalam permasalahan kontrol atau simulasi yang memerlukan pembelajaran melalui trial and error berbasis reward.	Algoritma tanpa model yang mengestimasi nilai Q untuk pasangan state-aksi secara iteratif berdasarkan reward yang diperoleh.	Umumnya tidak memerlukan normalisasi khusus, tergantung representasi state	Reward Accumulation, Convergence Rate, dan evaluasi performa kebijakan
SARSA	Reinforcement Learning	Cocok untuk situasi dinamis dimana strategi eksplorasi dan eksplorasi perlu disesuaikan pada masalah	Mirip dengan Q-Learning tetapi memperbarui nilai Q berdasarkan aksi yang diambil serta aksi selanjutnya (on-policy).	Normalisasi jarang diperlukan kecuali pada pra-pemrosesan state	Reward Accumulation, Convergence Rate, dan evaluasi performa strategi kebijakan

		sekuensial atau real-time.			
Deep Q-Network (DQN)	Reinforcement Learning	Untuk masalah dengan ruang state yang besar, seperti pada aplikasi game atau robotik kompleks.	Kombinasi deep learning dengan Q-Learning untuk mengaproksimasi nilai Q pada ruang state yang tinggi menggunakan jaringan saraf.	Dianjurkan untuk menormalisasi input (StandardScaler atau normalisasi khusus)	Akumulasi reward, Loss Q-Network, Convergence Rate, serta evaluasi performa kebijakan (policy performance)

LAMPIRAN 2

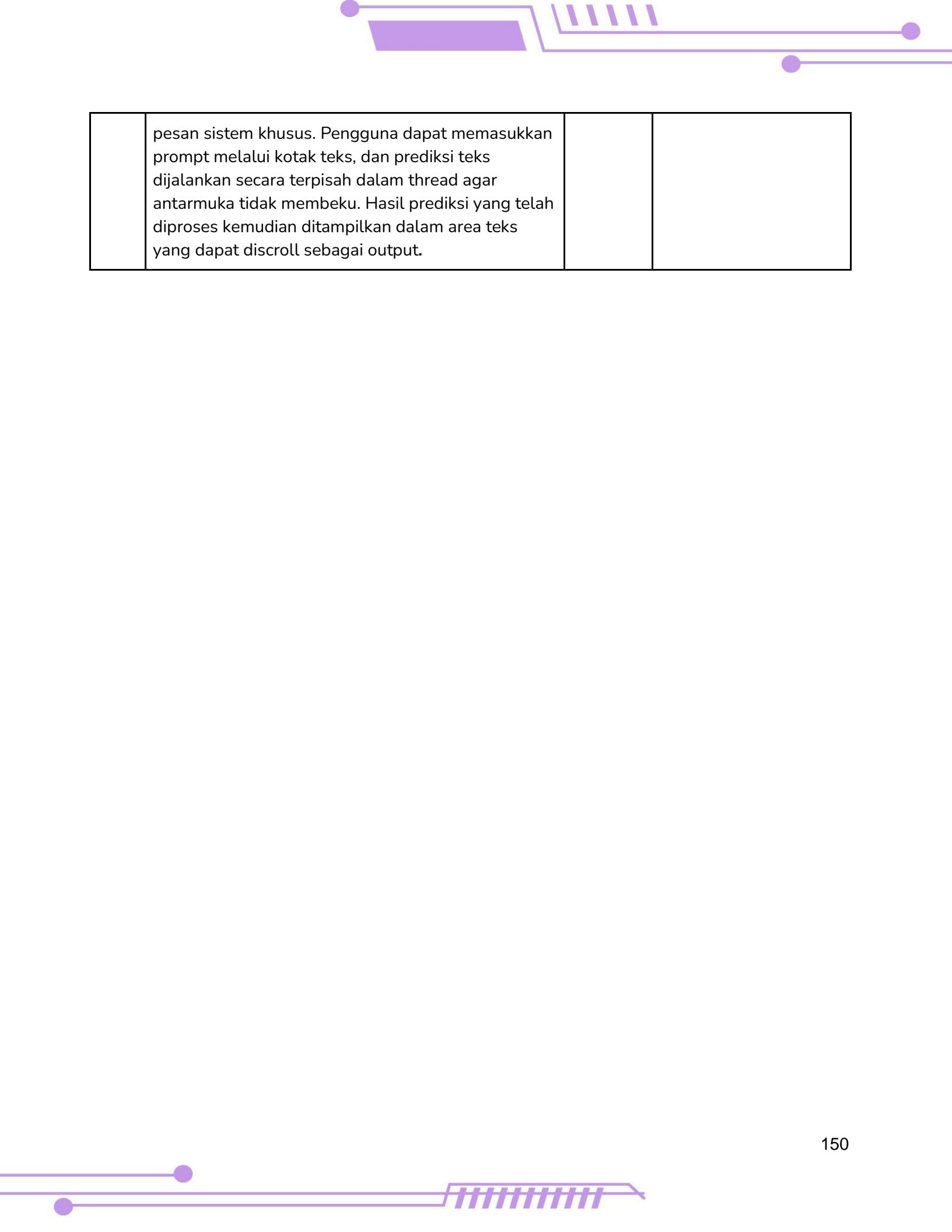
Kumpulan Contoh Penerapan Pemrograman KA

No	Deskripsi	Dataset / Model	Tautan
1.	K-Means Clustering Program ini memuat dataset wine dan melakukan Exploratory Data Analysis (EDA) untuk memahami distribusi dua fitur, yaitu <i>alcohol</i> dan <i>malic_acid</i> . Data kemudian diproses melalui penghapusan outlier dengan metode IQR dan dinormalisasi menggunakan StandardScaler sebelum diklasterisasi dengan algoritma KMeans. Hasil klasterisasi divisualisasikan dan pengguna dapat memasukkan data baru untuk memprediksi kluster yang sesuai berdasarkan model yang telah dibuat.	Wine	https://colab.research.google.com/drive/1ATmldTpshocnxqfqZ9wC3QMW6vdREHcH?usp=sharing
2	Logistic Regression Classification Program ini memuat dataset Iris, melakukan eksplorasi dan visualisasi data untuk memahami distribusi fitur serta hubungan antar fitur seperti scatter plot dan histogram. Data kemudian dibagi menjadi set pelatihan dan pengujian, dan model regresi logistik dilatih untuk mengklasifikasikan spesies Iris. Evaluasi model dilakukan dengan menghitung akurasi dan menampilkan confusion matrix untuk mengukur performa prediksi.	Iris	https://colab.research.google.com/drive/1vwoldLmLrr5FeSARUVYCBP8MTOm_rABN?usp=sharing
3	Data Preparation & Pre Processing Program ini membaca dataset dari file CSV dan menampilkan struktur, statistik deskriptif, serta informasi missing value untuk mengidentifikasi duplikat dan kesesuaian data. Program menangani pre-processing dengan mengonversi nilai numerik yang disimpan sebagai string, menggantikan missing value dengan median atau modus, dan menghapus outlier menggunakan metode IQR.	Indonesia Reading Interest	https://colab.research.google.com/drive/1d5yPdFKFB8s6hdP8iWsHB69TPOA6oaHb?usp=sharing

	Selanjutnya, distribusi data (khususnya tingkat kegemaran membaca) divisualisasikan sebelum dan sesudah pembersihan outlier serta dinormalisasi menggunakan MinMaxScaler. Akhirnya, dataset yang telah dibersihkan dan yang telah dinormalisasi disimpan kembali ke dalam file CSV untuk keperluan analisis lebih lanjut.		
4	<p>Data Visualization</p> <p>Program ini memuat dataset 'tips' dari Seaborn dan menampilkan lima baris pertama untuk melihat struktur data. Data divisualisasikan dengan histogram untuk mendeteksi distribusi frekuensi total bill, serta boxplot untuk mengidentifikasi outlier berdasarkan hari. Selanjutnya, scatter plot digunakan untuk mengeksplorasi hubungan antara total bill dan tip, dan heatmap menggambarkan korelasi antar variabel numerik. Akhirnya, bar plot ditampilkan untuk mengilustrasikan rata-rata tip per hari, memberikan gambaran tentang pola pemberian tip.</p>	Tips	https://colab.research.google.com/drive/1jt4cRTvaM18UlxJQ3VqpgzhZmoiTGFvn?usp=sharing
5	<p>Exploratory Data Analysis</p> <p>Program ini memuat dataset Titanic dan menampilkan lima baris pertama serta informasi dasar seperti struktur data, statistik deskriptif, dan jumlah missing values. Selanjutnya, analisis missing values divisualisasikan menggunakan heatmap untuk memberikan gambaran cepat mengenai kelengkapan data. Program kemudian mengeksplorasi distribusi variabel kategorik dan numerik melalui countplot, histogram, dan boxplot, serta menghitung matriks korelasi yang divisualisasikan dengan heatmap. Terakhir, hubungan antar variabel seperti umur dan tarif serta interaksi variabel lain ditampilkan melalui scatter plot dan pairplot untuk memperoleh insight awal mengenai faktor-faktor yang memengaruhi kelangsungan hidup penumpang.</p>	Titanic	https://colab.research.google.com/drive/1BBnmCH8oSt7lcCzMBUD7Fbb4kOQmeCsJ?usp=sharing
6	<p>Regresi Linier</p> <p>Program ini memuat dataset diabetes dan</p>	Diabetes	https://colab.research.google.com/drive/1V9Zz

	menyiapkan data dengan mengecek duplikat, missing values, serta melakukan normalisasi sederhana pada fitur BMI. Selanjutnya, dilakukan eksplorasi data melalui visualisasi histogram dan scatter plot untuk memahami distribusi BMI dan hubungannya dengan target (perkembangan penyakit). Data kemudian dipisahkan menjadi set pelatihan dan pengujian, dan fitur 'bmi' digunakan untuk melatih model linear regression. Terakhir, evaluasi model dilakukan dengan menghitung koefisien, intercept, dan skor R ² , serta hasilnya divisualisasikan melalui plot data dan garis regresi.		DDNcsGprbLsGGGua3GnCmK6Pmr1?usp=sharing
7	Regresi Logistik Program ini memuat Breast Cancer dataset dan menggunakan dua fitur ("mean radius" dan "mean texture") untuk klasifikasi tumor dengan Logistic Regression. Tambahan bagian data preparation mengecek duplikasi dan missing values, serta pre processing yang melakukan normalisasi pada fitur sehingga data siap dipakai. Exploratory data analysis (EDA) dilakukan dengan visualisasi histogram, scatter plot, dan heatmap korelasi untuk memahami karakteristik fitur. Selanjutnya, data dipisahkan menjadi training dan testing, model dilatih dan dievaluasi, serta terdapat opsi interaktif bagi pengguna untuk melakukan prediksi.	Kanker Payudara	https://colab.research.google.com/drive/1pk7vycijnjHCn6M5KrXGh6Do5v0_MGZ?usp=sharing
8	Support Vector Machine Program ini memuat dataset Iris, lalu melakukan data preprocessing dengan mengonversi data ke DataFrame, mengecek missing values, dan menampilkan statistik deskriptif. Selanjutnya, dilakukan exploratory data analysis (EDA) melalui pairplot dan heatmap korelasi untuk memahami hubungan antar fitur. Setelah itu, data dibagi ke dalam training dan testing, di-standardisasi, dan digunakan untuk melatih model Support Vector Machine (SVM). Terakhir, model dievaluasi dengan menggunakan classification report dan confusion matrix.	Iris	https://colab.research.google.com/drive/1SM7LTKJ8KPr1ziwyquQzBB3jdUwNpC62?usp=sharing
9	Neural Network	MNIST	https://colab.research.google.com/drive/1pk7vycijnjHCn6M5KrXGh6Do5v0_MGZ?usp=sharing

	Kode ini memuat dataset MNIST, melakukan normalisasi, dan menampilkan contoh gambar serta visualisasi EDA sederhana dari data. Model neural network sederhana dilatih untuk klasifikasi digit dengan 15 epoch.		https://drive.google.com/drive/1yvBhw14D8MyAuuLwE8o57pW4QsoofEr9?usp=sharing
10	Logistic Regression Program ini memuat dataset Titanic, lalu melakukan data preprocessing dengan menghapus missing value, mengisi nilai kosong dengan median, dan meng-encode variabel kategorik. Selanjutnya, dilakukan exploratory data analysis (EDA) melalui tampilan statistik deskriptif, histogram distribusi variabel numerik, serta heatmap korelasi antar fitur. Data kemudian dibagi menjadi training dan testing, model Logistic Regression dilatih, dan pengguna dapat memasukkan data penumpang untuk prediksi interaktif. Hasil prediksi ditampilkan berupa probabilitas selamat dan klasifikasi akhir.	TITANIC	https://colab.research.google.com/drive/1CvfOEhF3g3ipnTjasJuzrKsNGOUejNUR?usp=sharing
11	MobileNetV2 Program ini memuat model MobileNetV2 dengan bobot pre-trained dari ImageNet dan menyediakan fungsi untuk melakukan preprocessing gambar (resize, konversi ke array, dan normalisasi). Gambar diunggah melalui widget Google Colab, kemudian ditampilkan dan di-preprocess agar sesuai dengan input model. Model kemudian memprediksi gambar tersebut dan mendekode hasil prediksinya untuk menampilkan top 3 label beserta probabilitasnya. Secara keseluruhan, kode ini memungkinkan pengguna untuk mengunggah dan menganalisis gambar menggunakan model deep learning yang telah dilatih sebelumnya.	ImageNet	https://colab.research.google.com/drive/1N6tFaertZbCT_Amvdr2nku1uliS3E3rT?usp=sharing
12	Integrasi LLM dengan KA (Pirate Chatbot) Kode program ini membuat antarmuka pengguna interaktif menggunakan Tkinter untuk membangun sebuah "pirate chatbot" berbasis model LLM. Program memuat model "meta-llama/Llama-3.2-3B-Instruct" dari Hugging Face Transformers dan mengarahkan model tersebut agar selalu merespons dengan gaya bahasa bajak laut melalui	Model LLM meta-llama/Llama-3.2-3B-Instruct	https://github.com/listyantidewi1/pirate-chatbot



	pesan sistem khusus. Pengguna dapat memasukkan prompt melalui kotak teks, dan prediksi teks dijalankan secara terpisah dalam thread agar antarmuka tidak membeku. Hasil prediksi yang telah diproses kemudian ditampilkan dalam area teks yang dapat discroll sebagai output.		
--	---	--	--