

**LAPORAN PROYEK  
UJIAN AKHIR SEMESTER  
DATA SCIENCE**

**Analisis Klasifikasi Status Batu Empedu pada Dataset Klinis  
Menggunakan Machine Learning dan Deep Learning**



Disusun Oleh:  
Widya Wulandari (TRPL 5B / 234311056)

Dosen Pengampu:  
Gus Nanang Syaifuddiin, S.Kom., M.Kom.

**POLITEKNIK NEGERI MADIUN  
JURUSAN TEKNIK  
PROGRAM STUDI  
TEKNOLOGI REGAKAYASA PERANGKAT LUNAK  
2025**

- Judul Proyek: Analisis Klasifikasi Status Batu Empedu pada Dataset Klinis Menggunakan Machine Learning dan Deep Learning
- Nama Mahasiswa: Widya Wulandari
- NIM: 234311056
- Program Studi: Teknologi Rekayasa Perangkat Lunak
- Mata Kuliah: Data Science
- Dosen Pengampu: Gus Nanang Syaifuddiin, S.Kom., M.Kom.
- Tahun Akademik: 2024/2025
- Link GitHub Repository: <https://github.com/widyawuland/Klasifikasi-Status-Batu-Empedu.git>
- Link Video Pembahasan: [https://youtu.be/1G7PUQp244I?si=eK\\_1BDEDG3ZDbner](https://youtu.be/1G7PUQp244I?si=eK_1BDEDG3ZDbner)

## 1. LEARNING OUTCOMES

Pada proyek ini, mahasiswa diharapkan dapat:

- 1) Memahami konteks masalah dan merumuskan problem statement secara jelas
- 2) Melakukan analisis dan eksplorasi data (EDA) secara komprehensif (**OPSIONAL**)
- 3) Melakukan data preparation yang sesuai dengan karakteristik dataset
- 4) Mengembangkan tiga model machine learning yang terdiri dari (**WAJIB**):
  - Model baseline
  - Model machine learning / advanced
  - Model deep learning (**WAJIB**)
- 5) Menggunakan metrik evaluasi yang relevan dengan jenis tugas ML
- 6) Melaporkan hasil eksperimen secara ilmiah dan sistematis
- 7) Mengunggah seluruh kode proyek ke GitHub (**WAJIB**)
- 8) Menerapkan prinsip software engineering dalam pengembangan proyek

## 2. PROJECT OVERVIEW

### 2.1. Latar Belakang

Penyakit batu empedu (gallstone) merupakan salah satu gangguan pada sistem pencernaan yang memiliki dampak signifikan terhadap kualitas hidup pasien serta berpotensi menimbulkan komplikasi serius apabila tidak terdeteksi sejak dini. Identifikasi awal individu yang memiliki risiko tinggi terkena batu empedu sangat penting untuk mendukung upaya pencegahan dan penanganan yang lebih efektif.

Metode konvensional dalam diagnosis batu empedu, seperti ultrasonografi, meskipun akurat, memerlukan peralatan khusus, tenaga medis terlatih, serta tidak selalu efektif dalam mendeteksi batu empedu berukuran kecil. Oleh karena itu, diperlukan pendekatan berbasis data yang dapat memprediksi risiko batu empedu menggunakan informasi klinis yang lebih mudah diperoleh, seperti data bioimpedansi tubuh dan parameter laboratorium metabolik.

Dengan memanfaatkan teknik machine learning dan deep learning, proyek ini bertujuan untuk membangun model klasifikasi penyakit batu empedu berdasarkan karakteristik komposisi tubuh dan data laboratorium pasien. Pendekatan ini diharapkan mampu mengidentifikasi pola kompleks yang berkaitan dengan pembentukan batu empedu serta berfungsi sebagai alat bantu skrining awal berbasis data.

#### **Manfaat Proyek:**

- Peneliti/Klinisi: Memberikan wawasan hubungan komposisi tubuh dan faktor metabolismik terhadap risiko batu empedu.
- Kesehatan Masyarakat: Menyediakan alat skrining awal yang non-invasif dan efisien.
- Pengembangan Sistem Medis: Menjadi dasar sistem pendukung keputusan berbasis data.

#### **Referensi:**

- Esen, İ., Arslan, H., Aktürk Esen, S., Gülşen, M., Kültekin, N., & Özdemir, O. (2024). Early prediction of gallstone disease with a machine learning-based method from bioimpedance and laboratory data. *Medicine*, 103(12), e37258. <https://doi.org/10.1097/MD.00000000000037258>
- Chaudhari, G. S., Shinde, P. S., Phatak, M. P., & Deshpande, V. M. (2023). A review on prediction of gallstone disease using machine learning techniques. *Current Trends in Biomedical Engineering & Bioscience*, 22(4), 1-8. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10217597/>

### **3. BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING**

#### **3.1. Problem Statements**

- 1) Bagaimana membangun model machine learning dan deep learning untuk mengklasifikasikan status batu empedu berdasarkan data bioimpedansi dan parameter laboratorium?
- 2) Model mana yang memberikan performa terbaik dalam memprediksi risiko batu empedu?
- 3) Apakah model deep learning mampu memberikan peningkatan performa dibandingkan model machine learning tradisional?

#### **3.2. Goals**

- 1) Mengembangkan model klasifikasi status batu empedu menggunakan data tabular klinis.
- 2) Membandingkan performa baseline model, advanced machine learning model, dan deep learning model.
- 3) Mengevaluasi performa model menggunakan metrik klasifikasi yang relevan.

### 3.3. Solution Approach

#### **Model 1 – Baseline Model**

Model: Logistic Regression

Logistic Regression digunakan sebagai model baseline karena merupakan algoritma klasifikasi biner yang sederhana, efisien, dan banyak digunakan pada analisis data medis. Model ini memprediksi probabilitas kelas target menggunakan fungsi logistik sehingga hasilnya mudah diinterpretasikan. Selain itu, Logistic Regression memiliki waktu training yang cepat dan sering dijadikan benchmark awal untuk mengevaluasi peningkatan performa pada model yang lebih kompleks.

#### **Model 2 – Advanced / ML Model**

Model: Gradient Boosting Classifier

Gradient Boosting Classifier dipilih sebagai model advanced karena mampu menangkap hubungan non-linear antar fitur dan memiliki performa yang sangat baik pada data tabular klinis. Model ini bekerja dengan menggabungkan beberapa weak learner secara bertahap untuk meminimalkan kesalahan prediksi, sehingga menghasilkan model yang lebih akurat dibandingkan model linear. Selain itu, Gradient Boosting juga menyediakan informasi feature importance yang membantu analisis kontribusi fitur terhadap prediksi.

#### **Model 3 – Deep Learning Model (WAJIB)**

Model: Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) digunakan sebagai model deep learning karena sesuai untuk data tabular dan mampu mempelajari pola serta interaksi kompleks antar fitur. MLP terdiri dari beberapa hidden layer dengan fungsi aktivasi non-linear sehingga dapat menangkap hubungan yang tidak dapat dimodelkan secara optimal oleh algoritma tradisional. Model ini dilatih selama beberapa epoch, dilengkapi dengan visualisasi loss dan accuracy per epoch, serta dievaluasi menggunakan data uji untuk membandingkan performanya dengan model baseline dan advanced.

## **4. DATA UNDERSTANDING**

### 4.1. Informasi Dataset

Sumber Dataset: Kaggle (<https://www.kaggle.com/datasets/xixama/gallstone-dataset-uci>)

Deskripsi Dataset:

- Jumlah baris (rows): 319 data sempel
- Jumlah kolom (columns/features): 39 columns (38 fitur dan 1 target)
- Tipe data: Tabular
- Ukuran dataset: 55 KB
- Format file: CSV

Dataset klinis dikumpulkan dari Klinik Rawat Jalan Penyakit Dalam di Rumah Sakit Ankara VM Medical Park dan mencakup data dari 319 individu (Juni 2022–

Juni 2023), 161 di antaranya didiagnosis menderita penyakit batu empedu. Dataset ini berisi 38 fitur, termasuk data demografis, bioimpedansi, dan laboratorium, dan telah disetujui secara etis oleh Komite Etika Rumah Sakit Kota Ankara (E2-23-4632). Variabel demografis meliputi usia, jenis kelamin, tinggi badan, berat badan, dan BMI. Data bioimpedansi meliputi air total, ekstraseluler, dan intraseluler, massa otot dan lemak, protein, luas lemak visceral, dan lemak hati. Fitur laboratorium meliputi glukosa, kolesterol total, HDL, LDL, trigliserida, AST, ALT, ALP, kreatinin, GFR, CRP, hemoglobin, dan vitamin D.

#### 4.2. Deskripsi Fitur

Jelaskan setiap fitur/kolom yang ada dalam dataset.

| Nama Fitur                    | Tipe Data   | Deskripsi  | Contoh Nilai |
|-------------------------------|-------------|--|--------------|
| Gallstone Status              | Integer     | Variabel target, Batu empedu ada(0), dan tidak ada(1)  | 0, 1         |
| Age                           | Integer     | Usia orang tersebut (tahun)  | 50, 47, 61   |
| Gender                        | Categorical | Jenis kelamin orang tersebut, 0 (Laki-laki), 1 (Perempuan)   | 0, 1         |
| Comorbidity                   | Categorical | penyakit penyerta (komorbiditas), 0 (Tidak ada komorbiditas yang ada), 1 (Satu kondisi komorbid), 2 (Dua kondisi komorbid), 3 (Tiga atau lebih kondisi komorbid) | 0, 1, 2, 3   |
| Coronary Artery Disease (CAD) | Binary      | Penyakit kardiovaskular (penyakit yang melibatkan jantung atau pembuluh darah), 0 (Tidak), 1 (Ya)  | 0, 1         |
| Hypothyroidism                | Binary      | Kelenjar tiroid yang kurang aktif atau Hipotiroidisme, 0 (Tidak), 1 (Ya)   | 0, 1         |
| Hyperlipidemia                | Binary      | Kadar lemak tinggi dalam darah, 0 (Tidak), 1 (Ya)  | 0, 1         |

|  |         |  |                     |
|--|---------|--|---------------------|
| Diabetes Mellitus (DM)                         | Binary  | Gula darah tinggi, 0 (Tidak), 1 (Ya)   | 0, 1                |
| Height   | Integer | Tinggi badan   | 185, 176, 171       |
| Weight   | Float   | Berat badan  | 92.8, 94.5, 91.1    |
| Body Mass Index (BMI)                          | Float   | Rasio berat badan terhadap tinggi badan  | 27.1, 30.5, 31.2    |
| Total Body Water (TBW)                         | Float   | Total air dalam tubuh  | 52.9, 43.1, 47.2    |
| Extracellular Water (ECW)                      | Float   | Extracellular Water/ECW(semua cairan tubuh yang berada di luar sel. Ini mencakup plasma darah, cairan interstisial (cairan di sekitar sel), dan cairan transseluler (seperti cairan serebrospinal, cairan synovial)) | 21.2, 19.5, 20.1    |
| Intracellular Water (ICW)                      | Float   | Intracellular Water/ICW (Adalah semua cairan yang terkandung di dalam sel-sel tubuh)   | 31.7, 23.6, 27.1    |
| Extracellular Fluid/Total Body Water (ECF/TBW) | Float   | Extracellular Water Content/ECW (merujuk pada jumlah total cairan dalam tubuh yang berada di luar sel-sel (intraseluler))  | 40, 45, 43          |
| Total Body Fat Ratio (TBFR)                    | Float   | Total Lemak Tubuh (%)  | 19.2, 32.8, 27.3    |
| Lean Mass (LM)                                 | Float   | Massa tubuh tanpa lemak (%)  | 80.84, 67.2, 72.67  |
| Body Protein Content (Protein)                 | Float   | Blok bangunan esensial untuk tubuh (seperti Protein, Lemak, Karbohidrat, Vitamin dan Mineral) (%)  | 18.88, 16.68, 16.35 |
| Visceral Fat Rating (VFR)                      | Integer | Kadar lemak organ dalam  | 9, 15, 6            |
| Bone Mass (BM)                                 | Float   | Berat tulang   | 3.7, 3.2, 3.3       |
| Muscle Mass (MM)                               | Float   | Massa otot   | 71.4, 60.3, 62.9    |
| Obesity  | Float   | Kelebihan lemak tubuh (%)  | 23.4, 38.8, 41.7    |
| Total Fat Content (TFC)                        | Float   | Total kadar lemak  | 17.8, 31, 24.9      |

|                                  |             |   |                      |
|----------------------------------|-------------|---|----------------------|
| Visceral Fat Area (VFA)          | Float       | Area jaringan lemak dalam   | 10.6, 18.4, 16.2     |
| Visceral Muscle Area (VMA)       | Float       | Area otot dalam (kg)  | 39.7, 32.7, 34       |
| Hepatic Fat Accumulation (HFA)   | Categorical | Penumpukan lemak di hati, 0 (Tidak ada penumpukan lemak), 1 (Tingkat 1 (ringan)), 2 (Tingkat 2 (sedang)), 3 (Tingkat 3 (parah)), 4 (Tingkat 4 (sangat parah))                           | 0, 1, 2, 3, 4        |
| Glucose                          | Float       | Gula darah  | 102, 94, 103         |
| Total Cholesterol (TC)           | Float       | Total Cholesterol (Merupakan ukuran gabungan dari semua jenis kolesterol dalam darah Anda, termasuk kolesterol HDL (baik), kolesterol LDL (jahat), dan trigliserida (jenis lemak lain)) | 250, 172, 179        |
| Low Density Lipoprotein (LDL)    | Float       | Kolesterol jahat  | 175, 108, 124        |
| High Density Lipoprotein (HDL)   | Float       | Kolesterol baik   | 40, 43, 59           |
| Triglyceride                     | Float       | Jenis lemak yang ditemukan dalam darah  | 134, 103, 69         |
| Aspartat Aminotransferaz (AST)   | Float       | Jenis enzim hati  | 20, 14, 18           |
| Alanin Aminotransferaz (ALT)     | Float       | Enzim yang berhubungan dengan hati  | 22, 13, 14           |
| Alkaline Phosphatase (ALP)       | Float       | Jenis enzim hati dan tulang   | 87, 46, 66           |
| Creatinine                       | Float       | Indikator fungsi ginjal   | 0.82, 0.87, 1.25     |
| Glomerular Filtration Rate (GFR) | Float       | Laju filtrasi ginjal  | 112.47, 107.1, 65.51 |
| C-Reactive Protein (CRP)         | Float       | Indikator peradangan  | 0, 0.11, 1.57        |
| Hemoglobin (HGB)                 | Float       | Protein dalam darah yang membawa oksigen  | 16, 14.4, 16.2       |

|           |       |   |              |
|-----------|-------|---|--------------|
| Vitamin D | Float | Vitamin esensial untuk kesehatan tulang | 33, 25, 30.2 |
|-----------|-------|---|--------------|

#### 4.3. Kondisi Data

Jelaskan kondisi dan permasalahan data:

- Missing Values: Tidak ada
- Duplicate Data: Tidak ada
- Outliers: Ada

Jumlah outliers pada fitur :

|  |    |
|--|----|
| Age  | 2  |
| Comorbidity                                    | 2  |
| Coronary Artery Disease (CAD)                  | 12 |
| Hypothyroidism                                 | 9  |
| Hyperlipidemia                                 | 8  |
| Diabetes Mellitus                              | 43 |
| Weight   | 2  |
| Body Mass Index (BMI)                          | 5  |
| Total Body Water (TBW)                         | 2  |
| Extracellular Water (ECW)                      | 1  |
| Intracellular Water (ICW)                      | 1  |
| Extracellular Fluid/Total Body Water (ECF/TBW) | 7  |
| Body Protein Content (Protein) (%)             | 8  |
| Visceral Fat Rating (VFR)                      | 3  |
| Muscle Mass (MM)                               | 1  |
| Obesity (%)                                    | 7  |
| Total Fat Content (TFC)                        | 7  |
| Visceral Fat Area (VFA)                        | 5  |
| Glucose  | 25 |
| Total Cholesterol (TC)                         | 5  |
| Low Density Lipoprotein (LDL)                  | 4  |
| High Density Lipoprotein (HDL)                 | 5  |
| Triglyceride                                   | 20 |
| Aspartat Aminotransferaz (AST)                 | 23 |
| Alanin Aminotransferaz (ALT)                   | 27 |
| Alkaline Phosphatase (ALP)                     | 12 |
| Creatinine                                     | 2  |
| Glomerular Filtration Rate (GFR)               | 13 |
| C-Reactive Protein (CRP)                       | 34 |
| Hemoglobin (HGB)                               | 3  |
| Vitamin D                                      | 2  |

Meskipun terdeteksi adanya outliers pada beberapa fitur, nilai-nilai tersebut tidak dihapus karena masih berada dalam rentang variasi biologis yang wajar pada data klinis dan laboratorium. Dalam konteks medis, nilai ekstrem sering kali merepresentasikan kondisi kesehatan tertentu yang justru penting untuk membedakan pasien dengan dan tanpa risiko batu empedu. Menghapus nilai tersebut berpotensi menghilangkan informasi klinis yang relevan dan menurunkan kemampuan model dalam melakukan prediksi secara realistik, sehingga outliers tetap dipertahankan dalam dataset.

- Imbalanced Data: Tidak ada

| Gallstone Status | Jumlah Data | Persentase (%) |
|------------------|-------------|----------------|
| 0                | 161         | 50.47          |
| 1                | 158         | 49.53          |

- Noise: Ada

Berdasarkan hasil pengecekan, sebagian besar fitur tidak mengandung noise yang signifikan. Beberapa fitur seperti Obesity (%), Triglyceride, ALT, ECF/TBW, dan LM memiliki sedikit nilai di luar batas awal, namun masih wajar secara klinis dan jumlahnya relatif kecil. Oleh karena itu, data tersebut tetap dipertahankan agar model dapat mempelajari variasi kondisi medis yang sebenarnya. Selain itu, model yang digunakan (Logistic Regression, Gradient Boosting, dan MLP) cukup robust terhadap noise ringan, terutama setelah proses standardisasi data.

- Data Quality Issues: Tidak ada

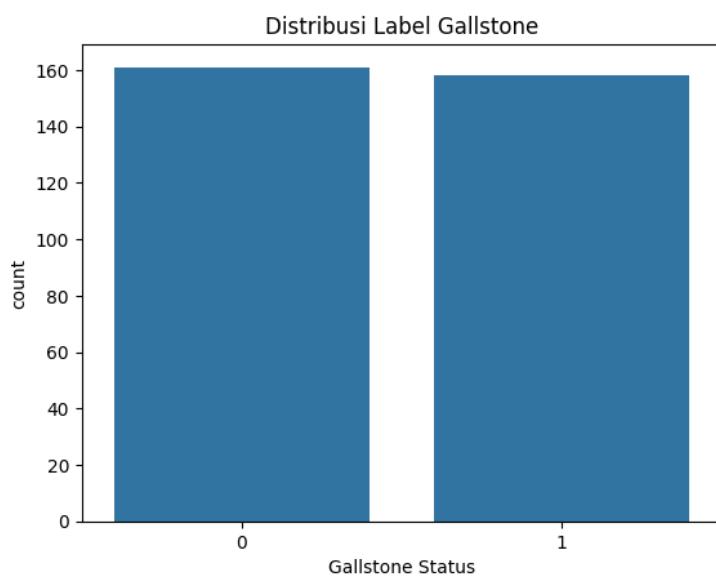
Dataset tidak memiliki data quality issues yang signifikan. Seluruh data telah lolos pengecekan kualitas, dan meskipun terdapat outliers serta noise ringan pada beberapa fitur laboratorium, nilainya masih wajar secara biologis sehingga tidak mempengaruhi kualitas data secara keseluruhan.

#### 4.4. Exploratory Data Analysis (EDA) - (OPSIONAL)

**Requirement:** Minimal 3 visualisasi yang bermakna dan insight-nya. **Contoh jenis visualisasi yang dapat digunakan:**

- Histogram (distribusi data)
- Heatmap korelasi (hubungan antar fitur)
- Boxplot (deteksi outliers)

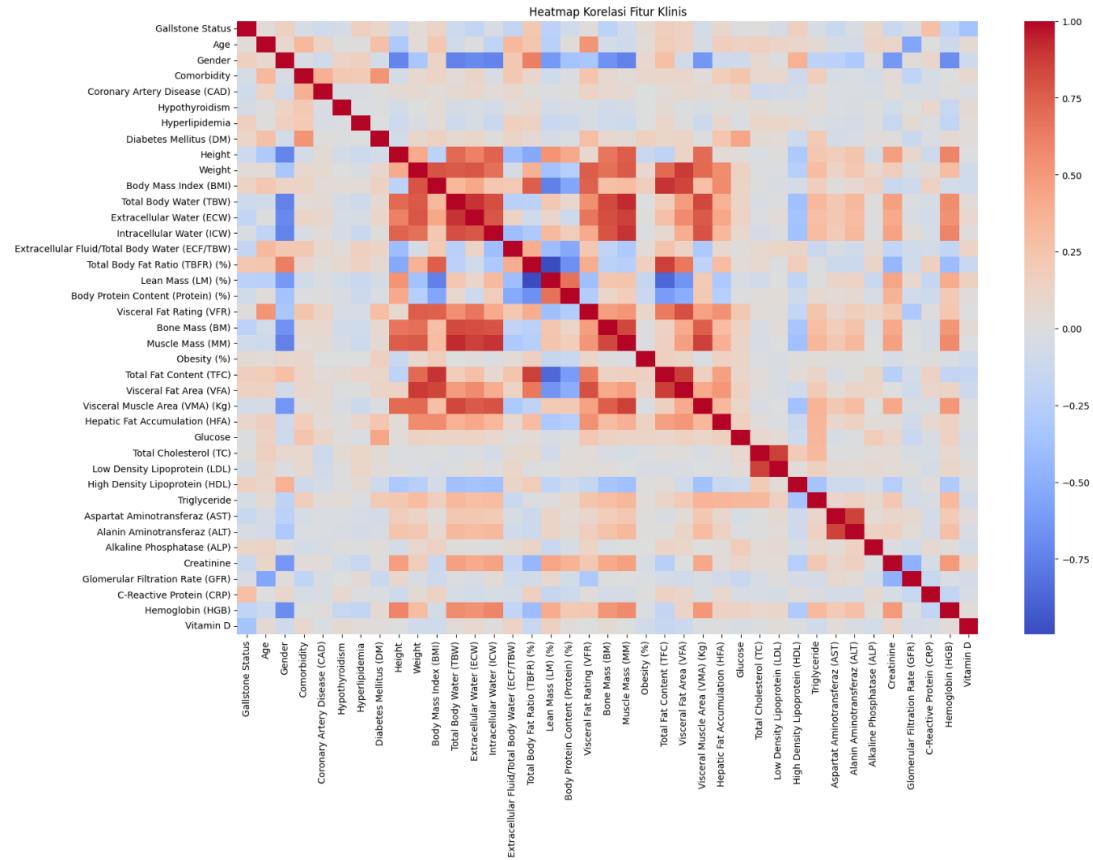
##### Visualisasi 1: Histogram (distribusi target Gallstone Status)



### Insight:

Histogram distribusi label Gallstone Status menunjukkan bahwa jumlah pasien dengan batu empedu (label 1) dan tanpa batu empedu (label 0) relatif seimbang. Hal ini mengindikasikan bahwa dataset tidak mengalami permasalahan class imbalance yang signifikan. Kondisi ini menguntungkan karena model machine learning dapat belajar dari kedua kelas secara proporsional tanpa memerlukan teknik penanganan khusus seperti oversampling atau undersampling.

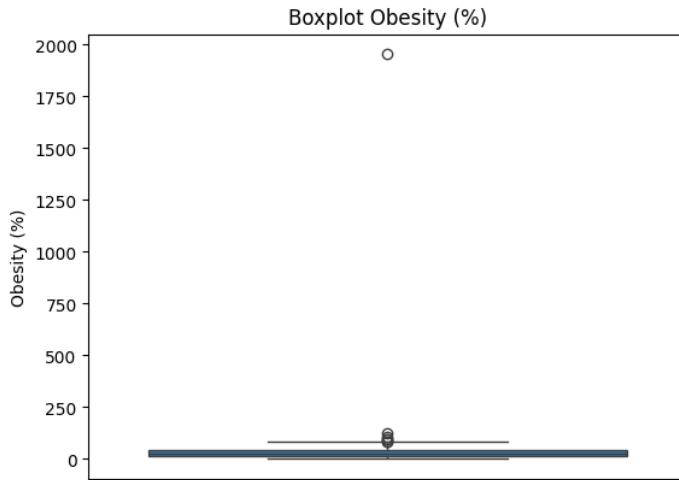
### Visualisasi 2: Heatmap Korelasi (Hubungan Antar Fitur Klinis)



### Insight:

Heatmap korelasi menunjukkan bahwa sebagian besar fitur memiliki hubungan yang lemah hingga sedang dengan status batu empedu. Hal ini menandakan bahwa risiko batu empedu dipengaruhi oleh kombinasi berbagai faktor klinis dan metabolismik, bukan oleh satu fitur tunggal, sehingga diperlukan pendekatan machine learning untuk menangkap pola yang lebih kompleks.

### Visualisasi 3: Boxplot (Deteksi Outliers Obesity (%))



#### **Insight:**

Boxplot Obesity (%) menunjukkan adanya beberapa outliers dengan nilai obesitas yang sangat tinggi. Nilai ini masih mungkin terjadi secara klinis pada pasien dengan kondisi metabolik tertentu, sehingga tidak dihapus dan tetap dipertahankan agar model dapat mempelajari variasi kondisi pasien yang sebenarnya.

## **5. DATA PREPARATION**

Dataset yang digunakan merupakan data klinis pasien yang terdiri dari beberapa fitur numerik dan kategorikal, seperti profil lipid, indikator bioimpedansi, serta variabel klinis lainnya. Jumlah data relatif terbatas sehingga menjadi pertimbangan penting dalam pemilihan model.

### **5.1. Data Cleaning**

Pada tahap data cleaning, dilakukan beberapa proses untuk memastikan kualitas data sebelum digunakan dalam pemodelan machine learning.

#### **Aktivitas:**

- Handling missing values
- Removing duplicates
- Handling outliers
- Data type conversion

#### **Missing Values:**

- Seluruh fitur pada dataset diperiksa menggunakan fungsi isnull().sum(). Hasil pengecekan menunjukkan tidak terdapat missing values pada dataset. Oleh karena itu, tidak dilakukan proses imputasi data.

```
#cek nilai yang hilang/missing values
missing_values=data.isnull().sum()

if missing_values.any():
    print("Ada missing values", missing_values)
else:
    print("Tidak ada missing values")
```

### Duplicate Data:

- Pengecekan data duplikat dilakukan menggunakan fungsi duplicated(). Hasil menunjukkan tidak ditemukan data duplikat sehingga seluruh data dipertahankan untuk proses analisis dan pemodelan.

```
#cek duplikat data yang ada didalam dataset
num_duplicates = data.duplicated().sum()
if num_duplicates > 0:
    print(f"\nMenghapus {num_duplicates} data duplikat.")
    data.drop_duplicates(inplace=True)
else:
    print("\nTidak ditemukan data duplikat.")
```

### Data Type:

- Pemeriksaan tipe data dilakukan menggunakan fungsi data.dtypes. Seluruh fitur memiliki tipe data yang sesuai (numerik dan kategorikal) jadi tidak diperlukan konversi tipe data tambahan.

```
#cek tipedata
data.dtypes
```

### Outliers:

- Deteksi outliers dilakukan pada fitur numerik menggunakan metode Interquartile Range (IQR). Ditemukan outliers pada beberapa fitur klinis dan laboratorium.
- Outliers tidak dihapus karena masih berada dalam rentang variasi biologis yang wajar dan merepresentasikan kondisi medis nyata pasien.

```
#cek outliers pada dataset
numeric_cols = data.select_dtypes(include=['int64', 'float64']).columns

Q1 = data[numeric_cols].quantile(0.25)
Q3 = data[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

outlier_mask = (data[numeric_cols] < (Q1 - 1.5 * IQR)) | (data[numeric_cols] > (Q3 + 1.5 * IQR))

outlier_count = outlier_mask.sum()
print("jumlah outliers pada kolom")
print(outlier_count)
```

### Noise Data:

- Pengecekan noise dilakukan dengan membandingkan nilai setiap fitur terhadap batas logis dan batas biologis yang wajar secara medis.
- Beberapa fitur terdeteksi memiliki noise ringan dengan jumlah yang relatif kecil. Jadi, data tetap dipertahankan agar model dapat mempelajari variasi kondisi pasien yang sebenarnya.

```

#cek noise
# skala ECF/TBW (ubah persen + rasio)
if "Extracellular Fluid/Total Body Water (ECF/TBW)" in data.columns:
    data["Extracellular Fluid/Total Body Water (ECF/TBW)"] = \
        data["Extracellular Fluid/Total Body Water (ECF/TBW)"] / 100

# batas wajar: 0 - 200
corrected_VMA_limit = (0, 200)

noise_limits = {
    "Age": (0, 120),
    "Height": (80, 250), # cm
    "Weight": (20, 300), # kg
    "Body Mass Index (BMI)": (5, 80),
    "Total Body Water (TBW)": (10, 80),
    "Extracellular Water (ECW)": (5, 40),
    "Intracellular Water (ICW)": (5, 60),
    "Extracellular Fluid/Total Body Water (ECF/TBW)": (0.3, 0.5),
    "Total Body Fat Ratio (TBFR) (%)": (1, 80),
    "Lean Mass (LM) (%)": (10, 90),
    "Body Protein Content (Protein) (%)": (5, 40),
    "Visceral Fat Rating (VFR)": (1, 30),
    "Bone Mass (BM)": (0.5, 10),
    "Muscle Mass (MM)": (5, 150),
    "Obesity (%)": (1, 100),
    "Total Fat Content (TFC)": (1, 100),
    "Visceral Fat Area (VFA)": (1, 300),
    "Visceral Muscle Area (VMA) (Kg)": corrected_VMA_limit,
    "Glucose": (40, 500),
    "Total Cholesterol (TC)": (50, 500),
    "Low Density Lipoprotein (LDL)": (10, 400),
    "High Density Lipoprotein (HDL)": (5, 200),
    "Triglyceride": (20, 1500),
    "Aspartat Aminotransferaz (AST)": (5, 500),
    "Alanin Aminotransferaz (ALT)": (5, 500),
    "Alkaline Phosphatase (ALP)": (10, 1000),
    "Creatinine": (0.2, 10),
    "Glomerular Filtration Rate (GFR)": (10, 200),
    "C-Reactive Protein (CRP)": (0, 300),
    "Hemoglobin (HGB)": (5, 25),
    "Vitamin D": (1, 150)
}

noise_report = {}

for col in noise_limits:
    low, high = noise_limits[col]
    if col in data.columns:
        count_noise = data[(data[col] < low) | (data[col] > high)].shape[0]
        noise_report[col] = count_noise
# Tampilkan hasil
pd.DataFrame.from_dict(noise_report, orient='index', columns=["Jumlah Noise"])

```

### Class Imbalance:

- Distribusi kelas pada variabel target Gallstone Status dianalisis. Hasil menunjukkan kelas relatif seimbang. Jadi, tidak diperlukan teknik penanganan imbalance seperti oversampling atau undersampling.

```

# cek imbalance data

import matplotlib.pyplot as plt
import seaborn as sns

#hitung jumlah tiap kelas
class_counts = data['Gallstone Status'].value_counts()

# Hitung persentase tiap kelas
class_percent = data['Gallstone Status'].value_counts(normalize=True) * 100

# Tampilkan jumlah & persentase
imbalance_df = pd.DataFrame({
    'Jumlah Data': class_counts,
    'Persentase (%)': class_percent.round(2)
})

print(imbalance_df)

# Visualisasi distribusi kelas
plt.figure(figsize=(5,4))
sns.countplot(x='Gallstone Status', data=data)
plt.title('Distribusi Kelas Gallstone Status')
plt.xlabel('Gallstone Status')
plt.ylabel('Jumlah Data')
plt.show()

```

## 5.2. Feature Engineering

### Aktivitas:

- Creating new features

Pada feature engineering saya melakukan Creating new feature dengan menambahkan fitur baru berupa rasio profil lipid dan indeks aterogenik untuk memperkaya informasi klinis pada dataset. Fitur yang dibuat meliputi rasio Total Cholesterol terhadap HDL (TC/HDL), rasio LDL terhadap HDL (LDL/HDL), serta Atherogenic Index of Plasma (AIP) yang dihitung dari logaritma rasio trigliserida terhadap HDL. Penambahan fitur-fitur ini didasarkan pada relevansi medisnya dalam menggambarkan risiko metabolismik dan gangguan lipid, yang diketahui berperan dalam proses pembentukan batu empedu, sehingga diharapkan dapat membantu model dalam mengenali pola risiko secara lebih akurat.

```

# Menambahkan rasio kolesterol (TC/HDL dan LDL/HDL) dan AIP
eps = np.finfo(float).eps
data['TC_to_HDL'] = data['Total Cholesterol (TC)'] / data['High Density Lipoprotein (HDL)'].replace(0, eps)
data['LDL_to_HDL'] = data['Low Density Lipoprotein (LDL)'] / data['High Density Lipoprotein (HDL)'].replace(0, eps)
data['AIP'] = np.log([data['Triglyceride'] / data['High Density Lipoprotein (HDL)'].replace(0, eps)])

```

## 5.3. Data Transformation

### Untuk Data Tabular:

- Scaling (Standardization)

Transformasi data dilakukan untuk memastikan seluruh fitur dapat diproses dengan baik oleh model. Fitur numerik distandarisasi menggunakan StandardScaler agar berada pada skala yang seragam, sehingga tidak ada fitur yang mendominasi proses pembelajaran model. Sebelum proses scaling, nilai yang hilang ditangani dengan imputasi menggunakan median untuk fitur numerik dan most frequent untuk fitur kategorikal. Seluruh tahapan ini diimplementasikan menggunakan Pipeline dan

ColumnTransformer agar proses transformasi lebih rapi dan konsisten. Transformasi hanya di-fit pada data training dan diterapkan pada data test untuk mencegah terjadinya data leakage.

```
#standarisasi
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

num_cols = X.select_dtypes(include=['int64','float64']).columns
cat_cols = X.select_dtypes(include=['object','bool']).columns

num_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent'))
])

preprocessor = ColumnTransformer([
    ('num', num_pipe, num_cols),
    ('cat', cat_pipe, cat_cols)
])

X_train_p = preprocessor.fit_transform(X_train)
X_test_p = preprocessor.transform(X_test)
```

## 5.4. Data Splitting

### Strategi pembagian data:

Menggunakan stratified split untuk mempertahankan distribusi kelas:

- Training: 80% (255 samples)
- Test: 20% (64 samples)
- Random state: 42 untuk reproducibility

Dataset dengan total 319 sampel dibagi menjadi data training sebesar 80% (255 data) dan data testing sebesar 20% (64 data). Rasio 80:20 dipilih karena ukuran dataset yang relatif kecil, sehingga sebagian besar data dapat dimanfaatkan secara optimal untuk proses pelatihan. Pembagian dilakukan menggunakan metode stratified split untuk menjaga proporsi kelas pada variabel target (Gallstone Status) tetap seimbang pada kedua subset. Strategi ini penting agar model dapat belajar dan dievaluasi secara adil tanpa bias distribusi kelas. Selain itu, penggunaan random state = 42 memastikan proses pembagian data dapat direproduksi.

```
# Split data training dan testing (training 80%, testing 20%)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

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

## 5.5. Ringkasan Data Preparation

### 1) Data Cleaning

- Apa yang dilakukan: Dilakukan pengecekan missing values, data duplikat, tipe data, outliers, imbalance kelas, dan noise pada seluruh fitur dataset.
- Mengapa penting: Data cleaning diperlukan untuk memastikan kualitas data sebelum pemodelan, sehingga model tidak terpengaruh oleh kesalahan data, inkonsistensi, atau nilai yang tidak wajar.
- Bagaimana implementasinya: Pengecekan missing values dilakukan menggunakan `isnull().sum()`, duplikasi data menggunakan `duplicated()`, serta pemeriksaan tipe data dengan `data.dtypes`. Outliers dideteksi menggunakan metode IQR, imbalance kelas dianalisis melalui distribusi target, dan noise diperiksa dengan membandingkan nilai fitur terhadap batas biologis yang wajar. Outliers dan noise ringan tidak dihapus karena masih relevan secara klinis.

### 2) Feature Engineering

- Apa yang dilakukan: Dilakukan pembuatan fitur baru berupa rasio profil lipid dan indeks aterogenik.
- Mengapa penting: Fitur tambahan ini dapat memperkaya informasi klinis dan membantu model menangkap risiko metabolik yang berhubungan dengan pembentukan batu empedu.
- Bagaimana implementasinya: Fitur baru yang ditambahkan meliputi rasio Total Cholesterol terhadap HDL (TC/HDL), LDL terhadap HDL (LDL/HDL), serta Atherogenic Index of Plasma (AIP) yang dihitung dari logaritma rasio trigliserida terhadap HDL.

### 3) Data Transformation

- Apa yang dilakukan: Dilakukan imputasi nilai hilang dan standardisasi pada fitur numerik.
- Mengapa penting: Meskipun tidak ditemukan missing values pada dataset, langkah imputasi tetap disertakan dalam pipeline sebagai tindakan pencegahan untuk menjaga kestabilan proses preprocessing dan memastikan pipeline tetap dapat digunakan pada data baru di masa mendatang. Standardisasi diperlukan agar seluruh fitur berada pada skala yang seragam sehingga proses pembelajaran model menjadi lebih stabil dan optimal.
- Bagaimana implementasinya: Imputasi nilai hilang dilakukan menggunakan median untuk fitur numerik dan most frequent untuk fitur kategorikal. Standardisasi diterapkan menggunakan `StandardScaler` yang diintegrasikan dalam Pipeline dan `ColumnTransformer`.

### 4) Data Splitting

- Apa yang dilakukan: Dataset dibagi menjadi data training dan data testing.

- Mengapa penting: Pembagian data diperlukan untuk melatih model dan mengevaluasi performa secara objektif pada data yang belum pernah dilihat sebelumnya.
- Bagaimana implementasinya: Data dibagi menggunakan metode stratified split dengan rasio 80% data training dan 20% data testing. Strategi ini menjaga proporsi kelas target tetap seimbang dan random state digunakan untuk memastikan hasil dapat direproduksi.

## 6. Modeling

### 6.1. Model 1 — Baseline Model

#### 6.1.1. Deskripsi Model

- **Nama Model:** Logistic Regression
- **Teori Singkat:** Logistic Regression merupakan algoritma klasifikasi linear yang memodelkan probabilitas suatu kelas menggunakan fungsi sigmoid.
- **Alasan Pemilihan:** Logistic Regression dipilih sebagai model baseline karena:
  - Model ini sederhana dan mudah diinterpretasikan.
  - Cocok untuk dataset berukuran kecil hingga menengah.
  - Memberikan tolok ukur awal (baseline) yang kuat sebelum menggunakan model yang lebih kompleks.
  - Sering digunakan sebagai pembanding awal dalam penelitian klasifikasi biner.

#### 6.1.2. Hyperparameter

##### Parameter yang digunakan:

- max\_iter: 500, digunakan untuk memastikan proses optimasi konvergen dengan baik.
- solver: ‘lbfgs’ (default), cocok untuk klasifikasi biner dan dataset berukuran kecil hingga menengah.
- C (regularization): 1.0 (default), menggunakan nilai bawaan Scikit-learn karena tidak dilakukan tuning parameter pada model baseline.

#### 6.1.3. Implementasi (Ringkas)

```
logreg = LogisticRegression(max_iter=500)
logreg.fit(X_train_p, y_train)
end = time.time()

y_pred_lr = logreg.predict(X_test_p)
y_prob_lr = logreg.predict_proba(X_test_p)[:,1]
```

#### 6.1.4. Hasil Awal

Model Logistic Regression menghasilkan performa yang stabil dengan accuracy sebesar 81% dan F1-score 0.81.

## 6.2. Model 2 — ML / Advanced Model

### 6.2.1. Deskripsi Model

- **Nama Model:** Gradient Boosting Classifier
- **Teori Singkat:** Gradient Boosting merupakan algoritma ensemble learning yang membangun model secara bertahap dengan menggabungkan banyak decision tree berukuran kecil (weak learners). Setiap model baru dilatih untuk memperbaiki kesalahan yang dibuat oleh model sebelumnya dengan meminimalkan fungsi loss menggunakan pendekatan gradient descent.
- **Alasan Pemilihan:** Gradient Boosting dipilih sebagai model lanjutan karena:
  - Mampu menangkap hubungan non-linear antar fitur.
  - Termasuk algoritma ensemble yang kuat untuk tugas klasifikasi.
  - Sering memberikan performa yang lebih baik dibandingkan model linear pada dataset dengan pola kompleks.
- **Keunggulan:**
  - Mampu memodelkan hubungan non-linear.
  - Memiliki kemampuan generalisasi yang baik pada banyak kasus.
  - Menyediakan informasi feature importance.
- **Kelemahan:**
  - Lebih sensitif terhadap ukuran dataset kecil.
  - Waktu pelatihan lebih lama dibandingkan model linear.
  - Rentan terhadap overfitting jika parameter tidak dikontrol dengan baik.

### 6.2.2. Hyperparameter

#### Parameter yang digunakan:

- n\_estimators: 100
- max\_depth: default
- learning\_rate: 0.1 (default)
- random\_state: 42

#### Hyperparameter Tuning (jika dilakukan):

- Metode: Tidak dilakukan
- Best Parameters: Menggunakan parameter default karena fokus saya adalah untuk membandingkan model, dan juga untuk menjaga kesederhanaan serta menghindari overfitting mengingat ukuran dataset yang relatif kecil

### 6.2.3. Implementasi (Ringkas)

```

gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb.fit(X_train_p, y_train)
end = time.time()

y_pred_gb = gb.predict(X_test_p)
y_prob_gb = gb.predict_proba(X_test_p)[:,1]

```

#### 6.2.4. Hasil Model

Model Gradient Boosting menghasilkan accuracy sebesar 0.77. Performa ini belum melampaui baseline, yang kemungkinan disebabkan oleh keterbatasan jumlah data, sehingga model kompleks cenderung kurang optimal.

### 6.3. Model 3 — Deep Learning Model (WAJIB)

#### 6.3.1. Deskripsi Model

**Nama Model:** MLP

\*\* (Centang) Jenis Deep Learning: \*\*

- Multilayer Perceptron (MLP) - untuk tabular
- Convolutional Neural Network (CNN) - untuk image
- Recurrent Neural Network (LSTM/GRU) - untuk sequential/text
- Transfer Learning - untuk image
- Transformer-based - untuk NLP
- Autoencoder - untuk unsupervised
- Neural Collaborative Filtering - untuk recommender

**Alasan Pemilihan:**

MLP dipilih karena:

1. Cocok untuk data tabular dengan fitur hasil preprocessing.
2. Mampu mempelajari hubungan non-linear yang tidak dapat ditangkap oleh model linear.
3. Digunakan untuk mengevaluasi peningkatan performa dengan pendekatan deep learning dibandingkan metode klasik.

#### 6.3.2. Arsitektur Model

**Deskripsi Layer:**

Model Multilayer Perceptron (MLP) yang digunakan terdiri dari dua hidden layer dengan fungsi aktivasi ReLU. Regularisasi L2 diterapkan pada setiap hidden layer untuk mengurangi risiko overfitting, terutama karena ukuran dataset yang relatif terbatas. Layer output menggunakan satu neuron dengan fungsi aktivasi sigmoid, sesuai dengan permasalahan klasifikasi biner.

Note: Input layer menerima data tabular hasil preprocessing dengan bentuk (input\_dim,), di mana input\_dim merupakan jumlah fitur setelah proses encoding dan scaling. Layer ini berfungsi sebagai penghubung antara data input dan layer Dense pertama.

| No. | Layer (Tipe)  | Jumlah Neuron / Unit | Fungsi Aktivasi | Regularisasi | Output Shape      |
|-----|---------------|----------------------|-----------------|--------------|-------------------|
| 1.  | Input Layer   | input_dim            | -               | -            | (None, input_dim) |
| 2.  | Dense Layer 1 | 16                   | ReLU            | L2 (0.001)   | (None, 16)        |
| 3.  | Dense Layer 2 | 8                    | ReLU            | L2 (0.001)   | (None, 8)         |
| 4.  | Output Layer  | 1                    | Sigmoid         | -            | (None, 1)         |

Total parameters: 817

Trainable parameters: 817

### 6.3.3. Input & Preprocessing Khusus

**Input shape:** (jumlah\_fitur,) → (41,)

(41 fitur adalah jumlah fitur hasil preprocessing yang digunakan sebagai input model)

#### Preprocessing khusus untuk DL:

Data telah melalui preprocessing yaitu meliputi feature engineering dan standarisasi (scaling).

- Feature engineering dilakukan dengan menambahkan fitur turunan berbasis rasio klinis, yaitu TC/HDL, LDL/HDL, dan Atherogenic Index of Plasma (AIP), yang dihitung dari kombinasi fitur lipid yang sudah ada untuk meningkatkan representasi informasi medis pada data.
- Seluruh fitur numerik kemudian dilakukan standarisasi menggunakan StandardScaler untuk menyamakan skala antar fitur sebelum dimasukkan ke dalam model Deep Learning.

### 6.3.4. Hyperparameter

**Training Configuration:**

- Optimizer: Adam
- Loss Function: Binary Crossentropy
- Metrics: Accuracy
- Batch Size: 16
- Epochs: 25
- Validation: Menggunakan validation set terpisah (test set)
- Callbacks: EarlyStopping (patience = 10, restore\_best\_weights = True)

### 6.3.5. Implementasi (Ringkas)

**Framework:** TensorFlow/Keras

```

# Contoh kode TensorFlow/Keras

mlp = Sequential([
    Dense(16, activation='relu',
          input_shape=(input_dim,), 
          kernel_regularizer=l2(0.001)),
    Dense(8, activation='relu',
          kernel_regularizer=l2(0.001)),
    Dense(1, activation='sigmoid')
])

# Model Summary
print("\n==== Model Summary (MLP) ===")
mlp.summary()

mlp.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

```

### 6.3.6. Training Process

- **Training Time:** Waktu pelatihan model Deep Learning MLP adalah 4.9746 detik untuk 25 epoch.
- **Computational Resource:** CPU, Platform: Google Colab.
- **Training History Visualization:**  
 Epoch 1/25  
**16/16** **1s** 18ms/step -  
 accuracy: 0.4427 - loss: 0.8100 - val\_accuracy: 0.5938  
 - val\_loss: 0.7132  
 Epoch 2/25  
**16/16** **0s** 6ms/step -  
 accuracy: 0.5261 - loss: 0.7528 - val\_accuracy: 0.5781  
 - val\_loss: 0.7035  
 Epoch 3/25  
**16/16** **0s** 6ms/step -  
 accuracy: 0.5617 - loss: 0.7182 - val\_accuracy: 0.5469  
 - val\_loss: 0.6962  
 Epoch 4/25  
**16/16** **0s** 6ms/step -  
 accuracy: 0.5695 - loss: 0.7138 - val\_accuracy: 0.5781  
 - val\_loss: 0.6895  
 Epoch 5/25  
**16/16** **0s** 7ms/step -  
 accuracy: 0.6215 - loss: 0.6936 - val\_accuracy: 0.5781  
 - val\_loss: 0.6817  
 Epoch 6/25  
**16/16** **0s** 7ms/step -  
 accuracy: 0.6223 - loss: 0.6897 - val\_accuracy: 0.6094  
 - val\_loss: 0.6756  
 Epoch 7/25  
**16/16** **0s** 9ms/step -  
 accuracy: 0.6383 - loss: 0.6787 - val\_accuracy: 0.6719  
 - val\_loss: 0.6704  
 Epoch 8/25

**16/16** ————— **0s** 7ms/step -  
accuracy: 0.6670 - loss: 0.6620 - val\_accuracy: 0.6562  
- val\_loss: 0.6658  
Epoch 9/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.6628 - loss: 0.6600 - val\_accuracy: 0.7031  
- val\_loss: 0.6606  
Epoch 10/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.6824 - loss: 0.6383 - val\_accuracy: 0.7031  
- val\_loss: 0.6539  
Epoch 11/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7385 - loss: 0.6256 - val\_accuracy: 0.7188  
- val\_loss: 0.6466  
Epoch 12/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7755 - loss: 0.6199 - val\_accuracy: 0.7188  
- val\_loss: 0.6447  
Epoch 13/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7328 - loss: 0.6364 - val\_accuracy: 0.7344  
- val\_loss: 0.6384  
Epoch 14/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7499 - loss: 0.6044 - val\_accuracy: 0.7344  
- val\_loss: 0.6335  
Epoch 15/25

**16/16** ————— **0s** 7ms/step -  
accuracy: 0.7215 - loss: 0.5970 - val\_accuracy: 0.7188  
- val\_loss: 0.6262  
Epoch 16/25

**16/16** ————— **0s** 7ms/step -  
accuracy: 0.7613 - loss: 0.5711 - val\_accuracy: 0.7188  
- val\_loss: 0.6203  
Epoch 17/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7932 - loss: 0.5561 - val\_accuracy: 0.7188  
- val\_loss: 0.6125  
Epoch 18/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7298 - loss: 0.5640 - val\_accuracy: 0.7344  
- val\_loss: 0.6073  
Epoch 19/25

**16/16** ————— **0s** 6ms/step -  
accuracy: 0.7973 - loss: 0.5329 - val\_accuracy: 0.7344  
- val\_loss: 0.5946  
Epoch 20/25

**16/16** ————— **0s** 7ms/step -  
accuracy: 0.7970 - loss: 0.5338 - val\_accuracy: 0.7188  
- val\_loss: 0.5892  
Epoch 21/25

**16/16** ————— **0s** 7ms/step -  
accuracy: 0.8211 - loss: 0.5097 - val\_accuracy: 0.7188  
- val\_loss: 0.5830  
Epoch 22/25

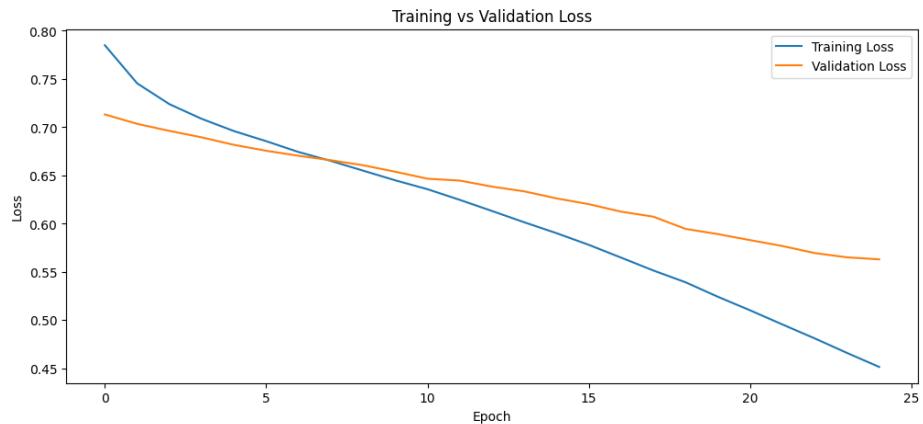
```

16/16 _____ 0s 6ms/step -
accuracy: 0.8011 - loss: 0.5065 - val_accuracy: 0.7188
- val_loss: 0.5769
Epoch 23/25
16/16 _____ 0s 7ms/step -
accuracy: 0.8655 - loss: 0.4641 - val_accuracy: 0.7344
- val_loss: 0.5696
Epoch 24/25
16/16 _____ 0s 6ms/step -
accuracy: 0.8370 - loss: 0.4704 - val_accuracy: 0.7344
- val_loss: 0.5652
Epoch 25/25
16/16 _____ 0s 6ms/step -
accuracy: 0.8370 - loss: 0.4532 - val_accuracy: 0.7344
- val_loss: 0.5631
2/2 _____ 0s 19ms/step

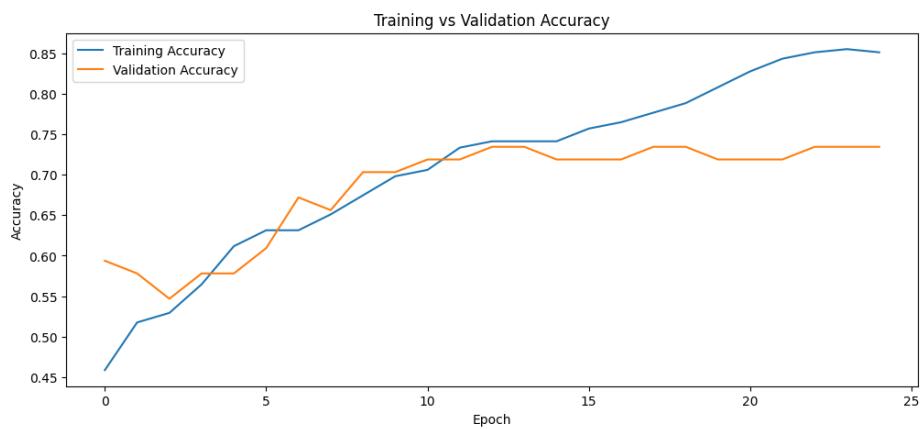
```

### Contoh visualisasi yang WAJIB:

#### 1. Training & Validation Loss per epoch



#### 2. Training & Validation Accuracy/Metric per epoch



### Analisis Training:

- Apakah model mengalami overfitting? Ya (overfitting ringan).

#### Penjelasan:

- Berdasarkan grafik Training vs Validation Accuracy dan Training vs Validation Loss:

- 1) Training accuracy meningkat secara konsisten dari  $\pm 46\%$  hingga  $\pm 85\%$ .
  - 2) Validation accuracy meningkat di awal, kemudian cenderung stagnan di kisaran 71–73%.
  - 3) Training loss terus menurun secara signifikan.
  - 4) Validation loss juga menurun, tetapi dengan laju yang lebih lambat dibandingkan training loss.
- Mulai sekitar epoch 15 ke atas, terlihat adanya gap yang semakin besar antara training accuracy dan validation accuracy. Hal ini menunjukkan bahwa model semakin baik dalam mempelajari data latih, namun peningkatan tersebut tidak sepenuhnya tertransfer ke data validasi.
  - Dikarenakan validation loss tidak meningkat dan validation accuracy tidak menurun secara drastic, maka overfitting yang terjadi bersifat ringan (tidak parah) dan masih dalam batas yang dapat diterima.
- 
- Apakah model sudah converge? Ya
- Penjelasan:**
- Convergence dapat dilihat dari kondisi ketika penurunan loss mulai melambat dan akurasi validasi tidak mengalami peningkatan signifikan lagi.
- Pada hasil training pada:
- Validation accuracy mulai stabil di kisaran 0.7188 – 0.7344 sejak sekitar epoch 11
  - Validation loss menurun sangat perlahan setelah epoch 15 Perubahan metrik pada epoch-epoch akhir relatif kecil, Hal ini menunjukkan bahwa model telah mencapai titik konvergensi, di mana penambahan epoch tidak lagi memberikan peningkatan performa yang signifikan pada data validasi.
- 
- Apakah perlu lebih banyak epoch? Tidak
- Penjelasan:**
- Penambahan epoch setelah epoch ke-25 tidak disarankan, karena:
- Validation accuracy tidak menunjukkan tren peningkatan
  - Gap antara training dan validation accuracy semakin besar
  - Model berisiko mengalami overfitting yang lebih kuat
- Dengan adanya Early Stopping dan pola grafik yang stabil, jumlah epoch yang digunakan saat ini sudah cukup optimal. Penambahan epoch kemungkinan hanya akan meningkatkan performa pada data latih tanpa meningkatkan generalisasi model.

### 6.3.7. Model Summary

```
Model: "sequential_13"
```

| Layer (type)     | Output Shape | Param # |
|------------------|--------------|---------|
| dense_39 (Dense) | (None, 16)   | 672     |
| dense_40 (Dense) | (None, 8)    | 136     |
| dense_41 (Dense) | (None, 1)    | 9       |

```
Total params: 817 (3.19 KB)
Trainable params: 817 (3.19 KB)
Non-trainable params: 0 (0.00 B)
```

## 7. Evaluation

### 7.1. Metrik Evaluasi

Untuk Klasifikasi:

- **Accuracy:** Proporsi prediksi yang benar terhadap seluruh data.
- **Precision:** Mengukur ketepatan prediksi positif,  $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- **Recall:** Mengukur kemampuan model mendekripsi kelas positif,  $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
- **F1-Score:** Harmonic mean dari precision dan recall.
- **ROC-AUC:** Mengukur kemampuan model membedakan dua kelas secara keseluruhan.
- **Confusion Matrix:** Visualisasi distribusi prediksi benar dan salah.

Metrik-metrik ini dipilih karena dataset relatif kecil dan klasifikasi bersifat biner, sehingga evaluasi tidak hanya bergantung pada accuracy saja.

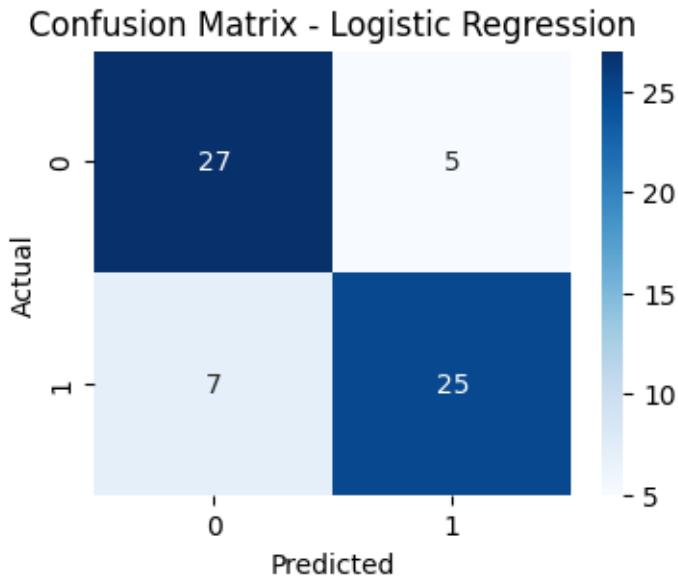
### 7.2. Hasil Evaluasi Model

#### 7.2.1. Model 1 Logistic Regression (Baseline)

Metrik:

- Accuracy: 0.8125
- Precision: 0.8333
- Recall: 0.7812
- F1-Score: 0.8065
- ROC-AUC: 0.8965
- Training Time: 0.0180 detik

Confusion Matrix / Visualization:



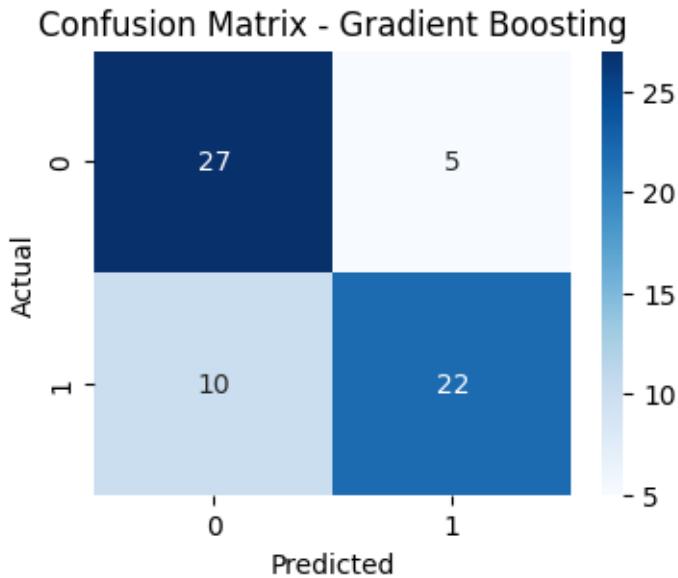
Model Logistic Regression menunjukkan performa yang stabil dan efektif dengan accuracy 81.25% dan ROC-AUC 0.8965, yang menandakan kemampuan pemisahan kelas yang sangat baik. Berdasarkan confusion matrix, sebagian besar data berhasil diklasifikasikan dengan benar, dengan jumlah kesalahan prediksi yang relatif rendah. Selain itu, waktu pelatihan yang sangat cepat (0.018 detik) menjadikan model ini efisien dan andal sebagai baseline pada dataset ini.

### **7.2.2. Model 2 Gradient Boosting (Advanced/ML)**

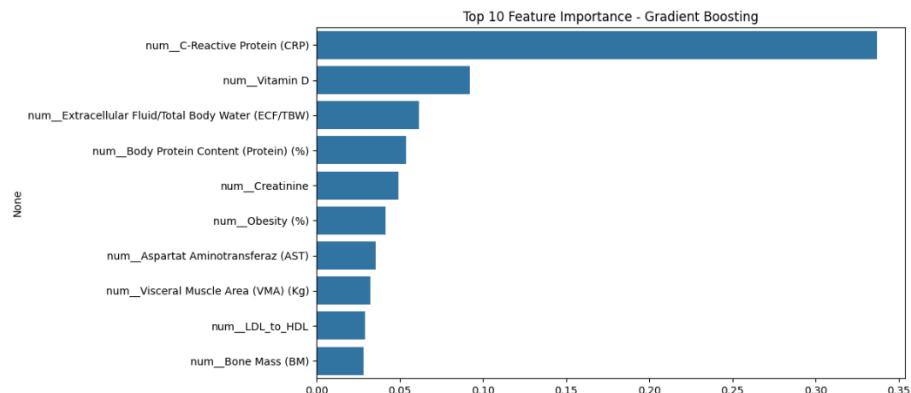
#### **Metrik:**

- Accuracy: 0.7656
- Precision: 0.8148
- Recall: 0.6875
- F1-Score: 0.7458
- ROC-AUC: 0.8545
- Training Time: 0.5943 detik

#### **Confusion Matrix / Visualization:**



#### Feature Importance (jika applicable):



Model Gradient Boosting mencapai accuracy 76.56% dan ROC-AUC 0.8545, dengan kemampuan klasifikasi yang cukup baik. Namun, berdasarkan confusion matrix terlihat masih terdapat cukup banyak false negative, yang menyebabkan nilai recall lebih rendah (68.75%). Selain itu, meskipun lebih kompleks, performanya tidak menunjukkan peningkatan signifikan dibandingkan Logistic Regression pada dataset ini.

Jika dataset diperbesar dan dilakukan hyperparameter tuning lebih lanjut, performa Gradient Boosting berpotensi meningkat di masa mendatang.

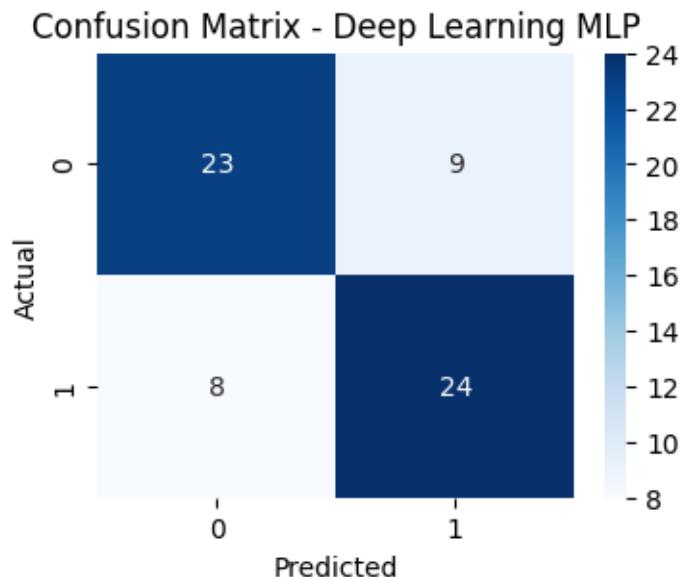
#### 7.2.3. Model 3 MLP (Deep Learning)

##### Metrik:

- Accuracy: 0.7344
- Precision: 0.7273
- Recall: 0.7500
- F1-Score: 0.7385

- ROC-AUC: 0.8184
- Training Time: 4.9746 detik

### Confusion Matrix / Visualization:



Berdasarkan confusion matrix Deep Learning MLP:

- True Negative (TN) = 23  
(Model berhasil mengklasifikasikan 23 pasien tanpa risiko batu empedu secara benar).
- False Positive (FP) = 9  
(Sebanyak 9 pasien diprediksi berisiko, padahal sebenarnya tidak).
- False Negative (FN) = 8  
(Model gagal mendeteksi 8 pasien yang sebenarnya berisiko batu empedu).
- True Positive (TP) = 24  
(Model berhasil mendeteksi 24 pasien berisiko dengan benar).

Nilai False Negative yang masih cukup tinggi menjadi perhatian penting dalam konteks medis, karena pasien berisiko yang tidak terdeteksi dapat berdampak pada keterlambatan penanganan. Meskipun demikian, nilai recall yang mencapai 75% menunjukkan bahwa model masih mampu mendeteksi sebagian besar pasien berisiko.

### Training History:

```
Epoch 1/25
16/16 ━━━━━━━━━━ 1s 18ms/step -
accuracy: 0.4427 - loss: 0.8100 - val_accuracy: 0.5938 -
val_loss: 0.7132
Epoch 2/25
16/16 ━━━━━━━━━━ 0s 6ms/step -
accuracy: 0.5261 - loss: 0.7528 - val_accuracy: 0.5781 -
val_loss: 0.7035
```

```
Epoch 3/25
16/16 _____ 0s 6ms/step -
accuracy: 0.5617 - loss: 0.7182 - val_accuracy: 0.5469 -
val_loss: 0.6962
Epoch 4/25
16/16 _____ 0s 6ms/step -
accuracy: 0.5695 - loss: 0.7138 - val_accuracy: 0.5781 -
val_loss: 0.6895
Epoch 5/25
16/16 _____ 0s 7ms/step -
accuracy: 0.6215 - loss: 0.6936 - val_accuracy: 0.5781 -
val_loss: 0.6817
Epoch 6/25
16/16 _____ 0s 7ms/step -
accuracy: 0.6223 - loss: 0.6897 - val_accuracy: 0.6094 -
val_loss: 0.6756
Epoch 7/25
16/16 _____ 0s 9ms/step -
accuracy: 0.6383 - loss: 0.6787 - val_accuracy: 0.6719 -
val_loss: 0.6704
Epoch 8/25
16/16 _____ 0s 7ms/step -
accuracy: 0.6670 - loss: 0.6620 - val_accuracy: 0.6562 -
val_loss: 0.6658
Epoch 9/25
16/16 _____ 0s 6ms/step -
accuracy: 0.6628 - loss: 0.6600 - val_accuracy: 0.7031 -
val_loss: 0.6606
Epoch 10/25
16/16 _____ 0s 6ms/step -
accuracy: 0.6824 - loss: 0.6383 - val_accuracy: 0.7031 -
val_loss: 0.6539
Epoch 11/25
16/16 _____ 0s 6ms/step -
accuracy: 0.7385 - loss: 0.6256 - val_accuracy: 0.7188 -
val_loss: 0.6466
Epoch 12/25
16/16 _____ 0s 6ms/step -
accuracy: 0.7755 - loss: 0.6199 - val_accuracy: 0.7188 -
val_loss: 0.6447
Epoch 13/25
16/16 _____ 0s 6ms/step -
accuracy: 0.7328 - loss: 0.6364 - val_accuracy: 0.7344 -
val_loss: 0.6384
Epoch 14/25
16/16 _____ 0s 6ms/step -
accuracy: 0.7499 - loss: 0.6044 - val_accuracy: 0.7344 -
val_loss: 0.6335
Epoch 15/25
16/16 _____ 0s 7ms/step -
accuracy: 0.7215 - loss: 0.5970 - val_accuracy: 0.7188 -
val_loss: 0.6262
Epoch 16/25
16/16 _____ 0s 7ms/step -
accuracy: 0.7613 - loss: 0.5711 - val_accuracy: 0.7188 -
val_loss: 0.6203
```

```

Epoch 17/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.7932 - loss: 0.5561 - val_accuracy: 0.7188 -
val_loss: 0.6125
Epoch 18/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.7298 - loss: 0.5640 - val_accuracy: 0.7344 -
val_loss: 0.6073
Epoch 19/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.7973 - loss: 0.5329 - val_accuracy: 0.7344 -
val_loss: 0.5946
Epoch 20/25
16/16 ━━━━━━━━ 0s 7ms/step -
accuracy: 0.7970 - loss: 0.5338 - val_accuracy: 0.7188 -
val_loss: 0.5892
Epoch 21/25
16/16 ━━━━━━━━ 0s 7ms/step -
accuracy: 0.8211 - loss: 0.5097 - val_accuracy: 0.7188 -
val_loss: 0.5830
Epoch 22/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.8011 - loss: 0.5065 - val_accuracy: 0.7188 -
val_loss: 0.5769
Epoch 23/25
16/16 ━━━━━━━━ 0s 7ms/step -
accuracy: 0.8655 - loss: 0.4641 - val_accuracy: 0.7344 -
val_loss: 0.5696
Epoch 24/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.8370 - loss: 0.4704 - val_accuracy: 0.7344 -
val_loss: 0.5652
Epoch 25/25
16/16 ━━━━━━━━ 0s 6ms/step -
accuracy: 0.8370 - loss: 0.4532 - val_accuracy: 0.7344 -
val_loss: 0.5631
2/2 ━━━━━━━━ 0s 19ms/step

```

### Test Set Predictions:

| Contoh Prediksi Test Set (MLP): |        |           |             |
|---------------------------------|--------|-----------|-------------|
|                                 | Actual | Predicted | Probability |
| 0                               | 1      | 1         | 0.621816    |
| 1                               | 1      | 1         | 0.787154    |
| 2                               | 0      | 0         | 0.269085    |
| 3                               | 0      | 0         | 0.470697    |
| 4                               | 0      | 0         | 0.008350    |
| 5                               | 0      | 0         | 0.035263    |
| 6                               | 1      | 1         | 0.760281    |
| 7                               | 1      | 1         | 0.662971    |
| 8                               | 1      | 1         | 0.649968    |
| 9                               | 1      | 1         | 0.803860    |

Contoh prediksi pada data test menunjukkan bahwa model menghasilkan probabilitas prediksi yang cukup konsisten:

- Sampel kelas positif (Actual = 1) umumnya memiliki probabilitas di atas 0,6–0,8.

- Sampel kelas negatif (Actual = 0) memiliki probabilitas yang relatif rendah, meskipun beberapa berada di sekitar nilai ambang (contoh di atas 0,47).

Hal ini menunjukkan bahwa:

- Kesalahan prediksi sebagian besar terjadi pada kasus borderline, yaitu pasien dengan karakteristik klinis yang berada di antara kelas positif dan negatif.
- Fenomena ini mencerminkan adanya overlap biologis antar kelas, di mana nilai fitur klinis pasien berisiko dan tidak berisiko saling tumpang tindih.

### 7.3. Perbandingan Ketiga Model

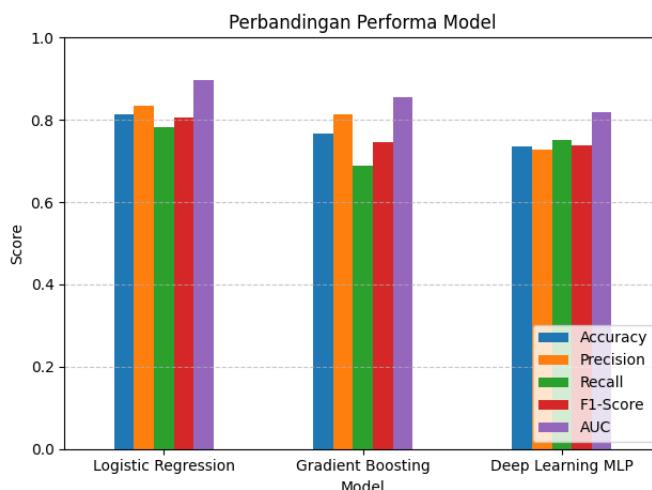
**Tabel Perbandingan:**

| == Perbandingan akhir model == |          |           |        |          |        |   |
|--------------------------------|----------|-----------|--------|----------|--------|---|
|                                | Accuracy | Precision | Recall | F1-Score | AUC    | \ |
| Logistic Regression            | 0.8125   | 0.8333    | 0.7812 | 0.8065   | 0.8965 |   |
| Gradient Boosting              | 0.7656   | 0.8148    | 0.6875 | 0.7458   | 0.8545 |   |
| Deep Learning MLP              | 0.7344   | 0.7273    | 0.7500 | 0.7385   | 0.8184 |   |

|                     | Training Time (s) |
|---------------------|-------------------|
| Logistic Regression | 0.0180            |
| Gradient Boosting   | 0.5943            |
| Deep Learning MLP   | 4.9746            |

**Visualisasi Perbandingan:**



Logistic Regression menunjukkan performa terbaik secara keseluruhan dengan stabilitas yang baik dan waktu training yang sangat singkat.

### 7.4. Analisis Hasil

#### Interpretasi:

- 1) **Model Terbaik:** Model terbaik berdasarkan hasil eksperimen adalah Logistic Regression.

Model Logistic Regression dipilih sebagai model terbaik karena menunjukkan performa paling stabil dan unggul dibandingkan model lainnya

pada data uji. Model ini mencapai accuracy sebesar 81.25%, F1-score sebesar 0.8065, serta AUC tertinggi sebesar 0.8965, yang menunjukkan kemampuan diskriminasi kelas yang sangat baik. Selain itu, Logistic Regression memiliki keseimbangan yang baik antara precision (0.8333) dan recall (0.7812), sehingga mampu melakukan klasifikasi dengan tingkat kesalahan yang relatif rendah pada kedua kelas.

Meskipun Deep Learning MLP memiliki kemampuan pembelajaran non-linear, pada dataset ini performanya masih berada di bawah Logistic Regression, yang mengindikasikan bahwa kompleksitas model yang lebih tinggi belum tentu menghasilkan performa yang lebih baik pada dataset berukuran terbatas.

### 2) Perbandingan dengan Baseline:

- Logistic Regression sebagai model baseline sudah mampu memberikan hasil klasifikasi yang kuat dengan waktu pelatihan yang sangat cepat (0.0180 detik) serta performa evaluasi yang tinggi. Hal ini menunjukkan bahwa hubungan antara fitur klinis dan target klasifikasi relatif dapat dimodelkan secara linear.
- Gradient Boosting tidak menunjukkan peningkatan performa dibandingkan baseline, dengan accuracy sebesar 76.56% dan F1-score sebesar 0.7458. Hal ini kemungkinan disebabkan oleh ukuran dataset yang terbatas, sehingga model ensemble tidak dapat memaksimalkan keunggulannya dalam menangkap pola kompleks.
- Deep Learning MLP menunjukkan performa paling rendah dibandingkan dua model lainnya, dengan accuracy sebesar 73.44% dan F1-score sebesar 0.7385. Meskipun demikian, model ini tetap mampu belajar dengan baik, namun kompleksitas arsitektur belum sepenuhnya terkompensasi oleh jumlah data yang tersedia.

### 3) Trade-off:

| Aspek             | Logistic Regression | Gradient Boosting | Deep Learning MLP |
|-------------------|---------------------|-------------------|-------------------|
| Kompleksitas      | Rendah              | Sedang            | Tinggi            |
| Training Time     | Sangat cepat        | Sedang            | Paling lama       |
| Interpretabilitas | Tinggi              | Sedang            | Rendah            |
| Performa          | Terbaik             | Cukup             | Terendah          |

Trade-off utama yang terlihat adalah antara kompleksitas model dan performa aktual. Logistic Regression menawarkan kombinasi terbaik antara performa, kecepatan pelatihan, dan interpretabilitas. Sebaliknya, Deep Learning MLP membutuhkan waktu pelatihan paling lama, namun tidak memberikan peningkatan performa yang sebanding.

### 4) Error Analysis:

Berdasarkan confusion matrix masing-masing model, kesalahan prediksi umumnya terjadi pada pasien dengan nilai klinis yang berada di sekitar batas

pemisahan antar kelas (borderline). Pasien dengan kombinasi faktor risiko yang tidak dominan atau tidak ekstrem cenderung lebih sulit diklasifikasikan secara tepat oleh semua model.

Logistic Regression menunjukkan kesalahan prediksi paling sedikit, yang mengindikasikan bahwa pola data relatif konsisten dan dapat dipisahkan secara cukup baik menggunakan pendekatan linear. Gradient Boosting dan Deep Learning MLP masih mengalami kesulitan dalam beberapa kasus dengan karakteristik yang saling tumpang tindih antar kelas, yang mencerminkan adanya overlap biologis pada data klinis pasien batu empedu.

### 5) Overfitting/Underfitting:

Secara umum, tidak ditemukan indikasi underfitting pada ketiga model, karena seluruh model mampu mencapai akurasi di atas baseline acak. Namun, pada model Deep Learning MLP terlihat adanya overfitting ringan, yang ditunjukkan oleh meningkatnya training accuracy hingga lebih dari 80% sementara validation accuracy cenderung stagnan di kisaran 71–73%.

Overfitting tersebut berhasil dikendalikan melalui penerapan regularisasi L2, early stopping, dan standarisasi fitur, sehingga tidak menyebabkan penurunan performa yang drastis pada data validasi.

## 8. Conclusion

### 8.1. Kesimpulan Utama

#### Model Terbaik: Logistic Regression

**Alasan:** Model ini memberikan performa terbaik secara keseluruhan dengan keseimbangan yang baik antara accuracy, F1-score, dan AUC, serta memiliki keunggulan dalam hal interpretabilitas dan efisiensi komputasi. Hasil ini menunjukkan bahwa pada dataset klinis berukuran terbatas, model sederhana yang terkalibrasi dengan baik dapat mengungguli model yang lebih kompleks.

#### Pencapaian Goals:

Seluruh goals pada Section 3.2 berhasil tercapai, yaitu:

1. Pengembangan tiga model klasifikasi (Logistic Regression, Gradient Boosting, dan Deep Learning MLP).
2. Perbandingan performa antar model menggunakan metrik evaluasi yang relevan.
3. Evaluasi menyeluruh terhadap kemampuan generalisasi model.

### 8.2. Key Insights

#### Insight dari Data:

Dataset relatif seimbang sehingga tidak memerlukan teknik penanganan imbalance. Fitur klinis dan bioimpedansi memiliki variasi biologis yang informatif, serta outlier mencerminkan kondisi medis nyata yang tidak sebaiknya dihapus secara sembarangan.

### **Insight dari Modeling:**

Model sederhana seperti Logistic Regression terbukti sangat kompetitif pada dataset kecil. Model yang lebih kompleks tidak selalu menghasilkan performa yang lebih baik tanpa dukungan data yang cukup. Deep Learning memerlukan kontrol regularisasi yang ketat agar tidak mengalami overfitting.

### **8.3. Kontribusi Proyek**

#### **Manfaat praktis:**

Secara praktis, model yang dikembangkan dalam penelitian ini dapat digunakan sebagai sistem pendukung keputusan awal (decision support system) untuk melakukan skrining risiko batu empedu. Model ini membantu tenaga medis dalam mengidentifikasi pasien yang berisiko secara lebih cepat dan efisien berdasarkan data klinis dan bioimpedansi, sehingga dapat mendukung proses pengambilan keputusan awal sebelum dilakukan pemeriksaan lanjutan yang lebih kompleks atau invasif.

#### **Pembelajaran yang didapat:**

Dari sisi akademik, proyek ini memberikan pemahaman yang mendalam mengenai pentingnya pemilihan model machine learning yang sesuai dengan karakteristik data. Hasil penelitian menunjukkan bahwa model dengan kompleksitas rendah seperti Logistic Regression dapat memberikan performa yang optimal pada dataset berukuran terbatas. Selain itu, penelitian ini menekankan pentingnya evaluasi performa model secara komprehensif menggunakan berbagai metrik, seperti accuracy, precision, recall, F1-score, dan AUC, untuk menilai kemampuan generalisasi model secara lebih objektif.

## **9. Future Work (Opsional)**

Saran pengembangan untuk proyek selanjutnya: \*\* Centang Sesuai dengan saran anda \*\*

#### **Data:**

- Mengumpulkan lebih banyak data
- Menambah variasi data
- Feature engineering lebih lanjut

#### **Model:**

- Mencoba arsitektur DL yang lebih kompleks
- Hyperparameter tuning lebih ekstensif
- Ensemble methods (combining models)
- Transfer learning dengan model yang lebih besar

#### **Deployment:**

- Membuat API (Flask/FastAPI)

- Membuat web application (Streamlit/Gradio)
- Containerization dengan Docker
- Deploy ke cloud (Heroku, GCP, AWS)

#### **Optimization:**

- Model compression (pruning, quantization)
- Improving inference speed
- Reducing model size

## **10. Reproducibility (Wajib)**

### **10.1. GitHub Repository**

**Link Repository:** <https://github.com/widyawuland/Klasifikasi-Status-Batu-Empedu.git>

Repository harus berisi:

- Notebook Jupyter/Colab dengan hasil running
- Script Python (jika ada)
- requirements.txt atau environment.yml
- README.md yang informatif
- Folder structure yang terorganisir
- .gitignore (jangan upload dataset besar)

### **10.2. Environment & Dependencies**

**Python Version:** 3.12

**Main Libraries & Versions:**

```
numpy==1.24.3
pandas==2.0.3
scikit-learn==1.3.0
matplotlib==3.7.2
seaborn==0.12.2

# Deep Learning Framework
tensorflow==2.14.0
```