# BAB 2
# LANDASAN TEORI

## 2.1 Heart Failure

*Heart Failure* (HF) atau gagal jantung dipahami sebagai sindrom klinis multivariat kompleks yang terpicu saat kapabilitas pompa jantung tidak lagi adekuat dalam mendistribusikan pasokan darah untuk menyokong kebutuhan metabolisme jaringan tubuh [7]. Kelainan ini umumnya berakar dari kerusakan struktural atau degradasi fungsional organ jantung yang mengganggu dinamika pengisian ataupun ejeksi darah pada bagian ventrikel. Di ranah medis, HF dipetakan bersandarkan proporsi *Left Ventricular Ejection Fraction* (LVEF), yaitu persentase ejeksi volume darah dari ventrikel kiri pada tiap denyut kontraksi. Berdasarkan indeks LVEF tersebut, gangguan ini dikategorikan menjadi *Heart Failure with Reduced Ejection Fraction* (HFrEF) dengan rentang LVEF di bawah 40%, *Heart Failure with Mildly Reduced Ejection Fraction* (HFmrEF) dengan LVEF berkisar 40–49%, dan *Heart Failure with Preserved Ejection Fraction* (HFpEF) dengan capaian LVEF sama atau melebihi 50% [8]. Secara klinis, pasien HF diidentifikasi lewat manifestasi simptomatik seperti dispnea (sesak napas), letargi, penurunan toleransi aktivitas fisik, serta retensi cairan yang memicu edema perifer [9].

$$LVEF(\%) = \frac{EDV - ESV}{EDV} \times 100 \tag{2.1}$$

Persamaan tersebut menunjukkan bahwa *Left Ventricular Ejection Fraction* (LVEF) merupakan rasio antara volume darah yang dipompa keluar dari ventrikel kiri selama fase sistolik (*stroke volume*) terhadap volume darah yang berada di ventrikel kiri pada akhir fase diastolik (*End-Diastolic Volume* atau EDV). Nilai *stroke volume* diperoleh dari selisih antara *End-Diastolic Volume* (EDV) dan *End-Systolic Volume* (ESV), yaitu volume darah yang tersisa di ventrikel setelah kontraksi. Oleh karena itu, apabila nilai EDV dan ESV telah diketahui melalui pemeriksaan pencitraan medis, persentase LVEF dapat dihitung menggunakan Persamaan 2.1.

## 2.2 Data Acquisition

*Data acquisition* merupakan tahapan pengumpulan serta seleksi data yang akan dimanfaatkan dalam proses penelitian. Dalam penelitian yang menerapkan *machine learning*, kualitas data menjadi faktor yang sangat penting karena berpengaruh langsung terhadap kinerja model yang dibangun. Data yang tidak lengkap, mengandung ketidakkonsistenan, atau kurang merepresentasikan kondisi sebenarnya dapat menurunkan tingkat akurasi serta kemampuan model dalam melakukan generalisasi terhadap data yang belum pernah digunakan sebelumnya. Oleh sebab itu, penggunaan data yang relevan, representatif, dan berkualitas tinggi diperlukan agar model dapat mempelajari pola-pola yang sesuai dengan karakteristik permasalahan yang diteliti [10].

Penelitian ini menggunakan dataset *Medical Information Mart for Intensive Care IV* (MIMIC-IV), yaitu basis data rekam medis elektronik yang dikembangkan oleh MIT Laboratory for Computational Physiology dan tersedia secara publik untuk keperluan penelitian. Dataset ini berisi data pasien yang menjalani perawatan di *Beth Israel Deaconess Medical Center* (BIDMC), Boston, Amerika Serikat, dengan cakupan periode tahun 2008 hingga 2019 [11]. MIMIC-IV merupakan pengembangan dari MIMIC-III yang sebelumnya mencakup data pasien pada periode tahun 2001 hingga 2012 [12]. Dibandingkan dengan MIMIC-III, MIMIC-IV memiliki cakupan data yang lebih baru, struktur basis data yang lebih terorganisasi, serta pemisahan modul data rumah sakit (*hospital module*) dan unit perawatan intensif (*ICU module*) yang meningkatkan efisiensi proses ekstraksi dan analisis data.

MIMIC-IV menyediakan informasi klinis yang komprehensif, meliputi data demografis, diagnosis penyakit, riwayat rawat inap, hasil pemeriksaan laboratorium, tanda vital, prosedur medis, hingga data perawatan intensif. Kelengkapan dan keragaman informasi tersebut menjadikan MIMIC-IV sebagai salah satu sumber data yang banyak dimanfaatkan dalam penelitian kesehatan berbasis data, termasuk pengembangan model *machine learning* untuk prediksi luaran klinis dan *hospital readmission* [11]. Penggunaan MIMIC-IV dalam penelitian ini dipilih karena menyediakan data klinis yang lebih mutakhir dibandingkan MIMIC-III, sehingga diharapkan dapat menghasilkan model prediksi *readmission* yang lebih representatif terhadap kondisi pelayanan kesehatan yang lebih terkini.

## 2.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) merupakan pendekatan analisis data yang bertujuan untuk mengeksplorasi dan memahami karakteristik data sebelum dilakukan analisis lebih lanjut. Menurut [13], EDA berfokus pada proses *discovery* dan *exploration* untuk menemukan pola, hubungan, serta fenomena yang terdapat dalam data tanpa dibatasi oleh hipotesis yang telah ditentukan sebelumnya. Melalui EDA, peneliti dapat memperoleh pemahaman yang lebih mendalam mengenai struktur dan kualitas data yang digunakan dalam penelitian.

Dalam penelitian *machine learning*, EDA berperan penting dalam mengidentifikasi distribusi data, nilai yang hilang (*missing values*), pencilan (*outliers*), ketidakseimbangan kelas, serta hubungan antar variabel yang dapat memengaruhi performa model [13]. Oleh karena itu, EDA digunakan sebagai tahap awal untuk memastikan kualitas data dan mendukung proses *preprocessing*, *feature selection*, serta pembangunan model prediksi yang lebih efektif.

## 2.4 Data Preprocessing

*Data preprocessing* merupakan tahapan awal yang dilakukan untuk mempersiapkan data sebelum digunakan dalam proses pembangunan model *machine learning*. Tujuan utama dari tahap ini adalah meningkatkan kualitas dan konsistensi data dengan menangani berbagai permasalahan yang berpotensi memengaruhi performa model, seperti banyaknya kategori yang kompleks, keberadaan nilai yang hilang, serta format data yang belum sesuai untuk proses analisis. Dalam penelitian ini, tahapan *data preprocessing* mencakup *category simplification*, *missing value imputation*, dan *categorical encoding* guna menghasilkan dataset yang lebih terorganisasi, konsisten, dan siap digunakan pada proses pelatihan model.

### 2.4.1 Category Simplyfication

*Category simplification* merupakan proses penyederhanaan variabel kategorikal dengan cara menggabungkan kategori-kategori yang memiliki karakteristik serupa atau frekuensi kemunculan yang rendah ke dalam kelompok yang lebih umum. Proses ini bertujuan untuk mengurangi kompleksitas data, menurunkan dimensionalitas yang dihasilkan setelah proses encoding, serta meningkatkan efisiensi dan kemampuan generalisasi model *machine learning*.

Penyederhanaan kategori umumnya diterapkan pada variabel kategorikal dengan jumlah kategori yang besar (*high-cardinality categorical features*) sehingga representasi data menjadi lebih ringkas dan mudah dipelajari oleh model [14].

*Missing value imputation* adalah teknik yang digunakan untuk menangani data yang memiliki nilai hilang dengan mengisi nilai tersebut menggunakan pendekatan tertentu sehingga dataset tetap dapat dimanfaatkan secara efektif dalam proses pemodelan. Pada penelitian ini, metode yang diterapkan adalah *median imputation*, yaitu menggantikan setiap nilai yang hilang dengan nilai median yang diperoleh dari seluruh data valid pada fitur yang bersangkutan. Pemilihan metode ini didasarkan pada kemudahannya dalam implementasi serta kemampuannya yang lebih baik dalam mengurangi pengaruh nilai ekstrem (*outlier*) dibandingkan metode pengisian yang menggunakan nilai rata-rata (*mean*) [15].

Meskipun tidak terdapat batas persentase *missing value* yang bersifat universal untuk menentukan apakah suatu fitur harus diimputasi atau dieliminasi, fitur dengan proporsi data hilang yang sangat tinggi perlu dipertimbangkan secara hati-hati karena proses imputasi berpotensi menghasilkan estimasi yang kurang merepresentasikan kondisi sebenarnya [16]. Oleh karena itu, keputusan penanganan *missing value* dalam penelitian ini tidak hanya mempertimbangkan persentase data yang hilang, tetapi juga mempertimbangkan kualitas informasi yang dapat dipertahankan setelah proses imputasi.

### 2.4.2 Categorical Encoding

*Categorical Encoding* merupakan tahapan konversi variabel kategorikal ke dalam bentuk numerik sehingga dapat diolah oleh algoritma *machine learning*. Salah satu teknik yang banyak digunakan adalah *One-Hot Encoding*, yaitu metode yang merepresentasikan setiap kategori sebagai variabel biner tersendiri dengan nilai 0 atau 1. Melalui pendekatan ini, setiap kategori diperlakukan secara independen tanpa membentuk urutan atau tingkatan tertentu yang sebenarnya tidak ada pada data. Oleh karena itu, *One-Hot Encoding* dinilai efektif untuk merepresentasikan data kategorikal pada dataset tabular, khususnya dalam permasalahan klasifikasi [17].

Sebagai ilustrasi, fitur *gender* yang memiliki dua kategori, yaitu *Male* dan *Female*, dapat ditransformasikan menjadi dua kolom biner sebagaimana ditunjukkan pada Tabel 2.1.

Tabel 2.1. Contoh Transformasi One-Hot Encoding

| Gender | Gender_Female | Gender_Male |
|---|---|---|
| Female | 1 | 0 |
| Male | 0 | 1 |

## 2.5 Data Splitting dan Cross Validation

Integrasi tahapan *data splitting* bersama *cross validation* memegang peranan krusial dalam siklus hidup *machine learning* untuk menyajikan kerangka kerja evaluasi yang objektif dan mencerminkan kinerja model sesungguhnya terhadap data luar. *Data splitting* mengeksekusi pemisahan fisik antara data latihan dan data uji, sementara *cross validation* disuntikkan pada kompartemen latihan untuk menguji ketahanan generalisasi dan konsistensi arsitektur sepanjang eksperimen. Sinergi kedua teknik ini efektif meminimalkan dampak buruk *overfitting* sekaligus memberikan kalkulasi performa yang representatif sebelum model diverifikasi final menggunakan data uji independen [18].

### 2.5.1 Train Test Split

Teknik *Train-test split* adalah metode partisi dataset menjadi dua kompartemen independen, yaitu data pelatihan (*training set*) dan data pengujian (*testing set*), yang umum dipakai dalam rekayasa *machine learning*. Data pelatihan dioptimalkan untuk mengekstrak relasi pola internal data ke dalam model, sementara data pengujian disiapkan secara terpisah untuk mengukur reliabilitas model saat mengonstruksi prediksi dari sampel asing. Penerapan partisi ini bertujuan untuk menegakkan objektivitas uji performa serta menjamin kemampuan model dalam menggeneralisasi data baru [19].

### 2.5.2 Cross Validation

Prosedur *Cross validation* mengacu pada teknik evaluasi berulang yang dimanfaatkan demi memvalidasi konsistensi stabilitas performa arsitektur model menggunakan porsi data pelatihan secara bergantian. Pada skema *K-Fold Cross Validation*, dataset didistribusikan ke dalam beberapa partisi (*fold*), di mana setiap *fold* bertindak sebagai data uji validasi secara bergantian sedangkan sisa *fold* lainnya berperan sebagai data pelatihan. Pendekatan ini menjamin seluruh porsi

data digunakan baik dalam fase pelatihan maupun validasi, memberikan estimasi performa yang jauh lebih objektif daripada partisi tunggal [20].

## 2.6 Feature Selection

*Feature selection* ditujukan untuk memetakan dan menyeleksi variabel-variabel yang menyumbang kontribusi informatif paling signifikan bagi penentuan kelas target pada model *machine learning*. Prosedur ini krusial untuk memangkas dimensi data yang berlebihan, membuang fitur yang tidak relevan maupun duplikatif, sekaligus mempermudah akselerasi komputasi dan kualitas generalisasi model. Melalui retensi fitur dominan tersebut, model dilatih secara terarah untuk mengenali pola data, memicu akurasi tinggi, serta mempermudah aspek interpretasi [21].

### 2.6.1 Feature Importance

*Feature Importance* merupakan metode seleksi fitur yang mengukur tingkat kontribusi setiap fitur terhadap proses prediksi model. Pada model berbasis pohon seperti *Random Forest*, nilai kepentingan fitur dihitung berdasarkan akumulasi penurunan impuritas yang dihasilkan oleh fitur tersebut pada seluruh pohon keputusan [22].

$$\mathrm{Imp}(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t) \Delta i(s_t, t) \tag{2.2}$$

Persamaan tersebut menunjukkan bahwa nilai kepentingan suatu fitur ($\mathrm{Imp}(X_m)$) diperoleh dari rata-rata total penurunan impuritas yang dihasilkan oleh fitur $X_m$ pada seluruh pohon dalam *ensemble*. Nilai $N_T$ menyatakan jumlah pohon, $p(t)$ merupakan proporsi sampel yang mencapai node $t$, sedangkan $\Delta i(s_t, t)$ menunjukkan besarnya penurunan impuritas yang dihasilkan oleh pemisahan pada node tersebut. Semakin besar nilai penurunan impuritas yang dihasilkan suatu fitur, semakin tinggi pula tingkat kepentingannya dalam proses prediksi [22].

### 2.6.2 Extra Trees

*Extra Trees* (*Extremely Randomized Trees*) merupakan metode seleksi fitur berbasis *ensemble* pohon keputusan yang membangun banyak pohon dengan

pemilihan titik pemisah (*split point*) secara acak. Nilai kepentingan fitur diperoleh dari akumulasi penurunan impuritas yang dihasilkan oleh suatu fitur pada seluruh pohon dalam *ensemble*, sehingga fitur dengan kontribusi terbesar dianggap paling relevan terhadap variabel target [23].

$$\text{Imp}(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t) \Delta I(s_t, t) \tag{2.3}$$

Persamaan tersebut menunjukkan bahwa nilai kepentingan suatu fitur ($\text{Imp}(X_m)$) dihitung berdasarkan rata-rata total penurunan impuritas yang dihasilkan oleh fitur $X_m$ pada seluruh pohon dalam model. Nilai $N_T$ menyatakan jumlah pohon, $p(t)$ merupakan proporsi sampel yang mencapai node $t$, sedangkan $\Delta I(s_t, t)$ menunjukkan besarnya penurunan impuritas yang dihasilkan oleh fitur pada proses pemisahan data. Semakin besar akumulasi penurunan impuritas yang dihasilkan, semakin tinggi tingkat kepentingan fitur tersebut dalam proses prediksi [23].

### 2.6.3 Mutual Information

*Mutual Information* (MI) merupakan metode seleksi fitur berbasis teori informasi yang digunakan untuk mengukur tingkat ketergantungan antara suatu fitur dan variabel target. Semakin besar nilai *Mutual Information*, semakin banyak informasi yang diberikan oleh fitur tersebut dalam memprediksi variabel target, termasuk untuk hubungan yang bersifat nonlinier [24].

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right) \tag{2.4}$$

Persamaan tersebut menghitung nilai *Mutual Information* antara fitur $X$ dan target $Y$ berdasarkan distribusi probabilitas gabungan $p(x,y)$ serta distribusi probabilitas marginal $p(x)$ dan $p(y)$. Nilai $I(X;Y)$ akan bernilai nol apabila kedua variabel saling independen, sedangkan nilai yang lebih besar menunjukkan bahwa fitur $X$ memiliki informasi yang lebih relevan untuk menjelaskan atau memprediksi variabel target $Y$ [24].

### 2.6.4 LASSO

*Least Absolute Shrinkage and Selection Operator* (LASSO) merupakan metode regularisasi dan seleksi fitur yang bekerja dengan memberikan batasan (*constraint*) pada jumlah nilai absolut parameter model. Melalui mekanisme penalti norma $L_1$ ini, LASSO melakukan proses penyusutan (*shrinking*) koefisien regresi dari fitur-fitur yang dinilai redundan atau tidak informatif hingga mencapai nilai tepat sama dengan nol, sehingga fitur tersebut secara otomatis tereliminasi dari model [25].

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \left( \frac{\|Y - X\beta\|_2^2}{n} + \lambda \|\beta\|_1 \right) \tag{2.5}$$

Persamaan tersebut menunjukkan bahwa estimasi koefisien LASSO diperoleh dengan meminimalkan jumlah kuadrat sisaan (*sum of squared errors*) yang dibagi dengan jumlah sampel $n$, lalu dikombinasikan dengan fungsi penalti nilai absolut dari vektor koefisien $\|\beta\|_1$. Pengendalian kekuatan proses seleksi fitur sepenuhnya diatur oleh *tuning parameter* $\lambda$ (lambda); di mana peningkatan nilai $\lambda$ akan memperketat batasan geometris berbentuk belah ketupat (*diamond constraint region*) pada ruang solusi koordinat. Akibatnya, fungsi optimasi akan memotong titik pojok bersudut lancip dari wilayah batasan tersebut, yang memaksa banyak koefisien variabel menyusut menjadi nol ($\hat{\beta}_j(\lambda) = 0$) dan hanya menyisakan subset fitur paling signifikan yang mempertahankan nilai koefisien tidak sama dengan nol [25].

### 2.6.5 ANOVA

*Analysis of Variance* (ANOVA) merupakan metode seleksi fitur statistik yang digunakan untuk mengukur kemampuan suatu fitur dalam membedakan kelas yang berbeda berdasarkan perbedaan nilai rata-rata antar kelompok. Pada proses seleksi fitur, ANOVA menggunakan nilai *F-statistic* untuk mengevaluasi tingkat relevansi setiap fitur terhadap variabel target, di mana fitur dengan nilai F yang lebih tinggi dianggap memiliki kemampuan diskriminatif yang lebih baik [26].

$$F = \frac{MS_{between}}{MS_{within}} \tag{2.6}$$

Persamaan tersebut menunjukkan bahwa nilai F diperoleh dari rasio antara *Mean Square Between Groups* ($MS_{between}$) dan *Mean Square Within Groups* ($MS_{within}$). Nilai $MS_{between}$ merepresentasikan variasi antar kelompok kelas, sedangkan $MS_{within}$ merepresentasikan variasi data di dalam kelompok yang sama. Semakin besar nilai F, semakin besar perbedaan antar kelas dibandingkan variasi di dalam kelas, sehingga fitur tersebut dianggap lebih informatif dan relevan untuk digunakan dalam proses prediksi [26].

## 2.7 Machine Learning

*Machine learning* merupakan salah satu cabang dari kecerdasan buatan (*Artificial Intelligence*/AI) yang memungkinkan komputer mempelajari pola dari data dan menghasilkan prediksi tanpa memerlukan aturan yang diprogram secara eksplisit. Dalam bidang kesehatan, teknologi ini telah banyak dimanfaatkan untuk mengolah data klinis serta mendukung pengambilan keputusan medis, termasuk dalam prediksi risiko penyakit dan *readmission* pasien. Meskipun demikian, penerapannya masih menghadapi berbagai tantangan, seperti data yang tidak lengkap, distribusi kelas yang tidak seimbang, serta kompleksitas hubungan antarvariabel [27, 28, 29].

## 2.8 Random Forest

*Random Forest* merupakan algoritma *Ensemble Learning* yang menerapkan teknik *Bootstrap Aggregating* (Bagging) untuk membangun sekumpulan *decision tree*. Pada metode ini, setiap pohon dilatih menggunakan sampel bootstrap yang diambil secara acak dari data pelatihan, sedangkan pada setiap proses pemisahan (split) dipilih subset fitur secara acak. Kombinasi kedua mekanisme tersebut menghasilkan keragaman antar pohon, sehingga mampu mengurangi variansi model, meningkatkan kemampuan generalisasi, serta meminimalkan risiko overfitting. Selanjutnya, setiap *decision tree* melakukan proses klasifikasi menggunakan kriteria *Gini Impurity* untuk menentukan pemisahan terbaik pada setiap node, sedangkan prediksi akhir diperoleh melalui mekanisme *majority voting*, yaitu kelas yang memperoleh suara terbanyak dari seluruh pohon dipilih sebagai hasil klasifikasi akhir [30].

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{2.7}$$

dengan:

- $p_i$ adalah proporsi sampel pada kelas ke-$i$.
- $C$ adalah jumlah kelas pada dataset.

Persamaan Gini digunakan untuk mengukur tingkat ketidakmurnian (*impurity*) pada suatu node. Nilai Gini yang semakin kecil menunjukkan bahwa distribusi kelas pada node semakin homogen sehingga pemisahan data dianggap semakin baik [30].

$$RF = \arg \max_{j \in \{1,2,\ldots,C\}} \sum_{i=1}^{I} DT_{i,j} \tag{2.8}$$

dengan:

- $DT_{i,j}$ adalah prediksi pohon ke-$i$ terhadap kelas ke-$j$.
- $I$ adalah jumlah total *decision tree*.
- $C$ adalah jumlah kelas target.
- arg max menunjukkan kelas dengan jumlah suara terbanyak.

Persamaan tersebut menunjukkan mekanisme *majority voting* pada Random Forest. Setiap *decision tree* menghasilkan prediksi terhadap kelas target, kemudian kelas yang memperoleh jumlah suara terbanyak dari seluruh pohon dipilih sebagai hasil prediksi akhir model [30].

```
Input  : Dataset pelatihan D, jumlah decision tree I
Output : Kelas hasil prediksi

for i = 1 hingga I do

    Bentuk bootstrap sample Di dari dataset pelatihan D

    Pilih subset fitur secara acak

    Bangun sebuah decision tree menggunakan Di

```

```
    while kriteria penghentian belum terpenuhi do

        Hitung nilai Gini Impurity untuk setiap kandidat
    split

        Pilih split dengan nilai Gini Impurity terkecil

        Lakukan pemisahan (split) pada node

    end while

end for

for setiap data uji do

    Peroleh hasil prediksi dari seluruh decision tree

    Tentukan hasil prediksi akhir menggunakan Majority
    Voting

end for

return Kelas hasil prediksi
```

Kode 2.1: Pseudocode Random Forest

Pseudocode pada kode 2.1 menunjukkan alur kerja Random Forest yang diawali dengan pembentukan sejumlah *bootstrap sample* dari data pelatihan menggunakan teknik *bagging*. Setiap sampel digunakan untuk membangun sebuah *decision tree* dengan pemilihan subset fitur secara acak pada setiap proses pemisahan. Kualitas pemisahan ditentukan menggunakan nilai *Gini Impurity*, di mana nilai Gini terkecil dipilih sebagai *split* terbaik. Setelah seluruh *decision tree* selesai dibangun, setiap pohon memberikan prediksi terhadap data uji, kemudian hasil akhir ditentukan menggunakan mekanisme *majority voting*.

## 2.9 Extreme Gradient Boosting

*Extreme Gradient Boosting* (XGBoost) ialah salah satu algoritma *ensemble learning* turunan dari kerangka *gradient boosting*, di mana model diuji dan

dibangun secara sekuensial lewat tumpukan pohon keputusan demi mengevaluasi sisa kekeliruan prediksi model terdahulu. Pada proses kalkulasi percabangan pohon, XGBoost menggunakan parameter informasi gradien serta matriks Hessian untuk menentukan pemisahan (*split*) node paling optimal, sehingga menghasilkan model berakurasi tinggi dengan durasi komputasi yang efisien [31].

$$\text{Gain}_{\text{XGBoost}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{2.9}$$

Persamaan tersebut digunakan untuk menghitung nilai *gain* yang dihasilkan oleh suatu kandidat pemisahan node pada pohon keputusan. Nilai $g_i$ dan $h_i$ masing-masing merepresentasikan gradien orde pertama dan Hessian orde kedua dari fungsi kerugian, sedangkan $I_L$ dan $I_R$ menunjukkan himpunan data pada node kiri dan kanan setelah pemisahan. Parameter $\lambda$ dan $\gamma$ berfungsi sebagai regularisasi untuk mengontrol kompleksitas model. Semakin besar nilai *gain* yang diperoleh, semakin baik kualitas pemisahan yang dihasilkan sehingga kandidat *split* tersebut lebih diprioritaskan dalam proses pembentukan pohon [31].

```
Input  : Dataset pelatihan D, jumlah pohon T
Output : Kelas hasil prediksi

Inisialisasi prediksi awal model

FOR t = 1 HINGGA T DO

    Hitung nilai gradient orde pertama (gi)

    Hitung nilai Hessian orde kedua (hi)

    Bangun sebuah decision tree

    WHILE kriteria penghentian belum terpenuhi DO

        Hitung nilai Gain untuk setiap kandidat split

        Pilih split dengan nilai Gain terbesar

        Lakukan pemisahan (split) pada node

```

```
    END WHILE

    Perbarui model dengan menambahkan decision tree yang baru

END FOR

Hasilkan prediksi akhir

RETURN Kelas hasil prediksi
```

Kode 2.2: Pseudocode Extreme Gradient Boosting (XGBoost)

Pseudocode pada kode 2.2 menggambarkan proses pembentukan model XGBoost yang dilakukan secara sekuensial. Pada setiap iterasi, model terlebih dahulu menghitung nilai gradien orde pertama dan Hessian orde kedua dari fungsi kerugian untuk mengukur kesalahan prediksi. Selanjutnya dibangun sebuah *decision tree* baru dengan memilih kandidat pemisahan yang menghasilkan nilai *Gain* terbesar. Pohon yang terbentuk kemudian ditambahkan ke dalam model untuk memperbaiki kesalahan prediksi pada iterasi sebelumnya. Proses tersebut diulang hingga jumlah pohon yang ditentukan tercapai sehingga menghasilkan model prediksi akhir.

## 2.10 Light Gradient Boosting Machine

*Light Gradient Boosting Machine* (LightGBM) merepresentasikan varian arsitektur *ensemble learning* berbasis struktur *Gradient Boosting Decision Tree* (GBDT) yang dirancang khusus untuk meminimalkan beban komputasi pada basis data skala besar. Algoritma ini memprioritaskan ekspansi pohon lewat pendekatan secara *leaf-wise* serta menerapkan skema *Gradient-based One-Side Sampling* (GOSS) guna mengisolasi sampel yang bernilai informatif tinggi semasa pelatihan, memicu keaslian akurasi model dengan durasi waktu pelatihan yang lebih singkat [32].

$$\tilde{V}_j(d) = \frac{1}{2n}\left[\frac{\left(\sum_{i\in A_L} g_i + \frac{1-a}{b}\sum_{i\in B_L} g_i\right)^2}{N_L^j(d)} + \frac{\left(\sum_{i\in A_R} g_i + \frac{1-a}{b}\sum_{i\in B_R} g_i\right)^2}{N_R^j(d)}\right] \tag{2.10}$$

Persamaan tersebut digunakan untuk menghitung nilai *variance gain* pada mekanisme *Gradient-based One-Side Sampling* (GOSS) yang diterapkan oleh LightGBM. Himpunan $A$ merepresentasikan sampel dengan nilai gradien besar yang dipertahankan seluruhnya, sedangkan himpunan $B$ merupakan sampel dengan gradien kecil yang dipilih secara acak dan diberi faktor penyesuaian $\frac{1-a}{b}$ untuk mempertahankan distribusi data. Nilai $g_i$ menunjukkan gradien dari fungsi kerugian, sementara $N_L^j(d)$ dan $N_R^j(d)$ menyatakan jumlah sampel pada node kiri dan kanan setelah pemisahan. Semakin besar nilai $\tilde{V}_j(d)$ yang diperoleh, semakin baik kualitas pemisahan yang dihasilkan sehingga kandidat *split* tersebut lebih diprioritaskan dalam proses pembentukan pohon [32].

```
Input  : Dataset pelatihan D, jumlah decision tree T
Output : Kelas hasil prediksi

Inisialisasi prediksi awal model

FOR t = 1 HINGGA T DO

    Hitung nilai gradient dari fungsi kerugian

    Terapkan Gradient-based One-Side Sampling (GOSS)

    Bangun sebuah decision tree secara leaf-wise

    WHILE kriteria penghentian belum terpenuhi DO

        Hitung nilai Variance Gain untuk setiap kandidat split

        Pilih split dengan nilai Variance Gain terbesar

        Lakukan pemisahan (split) pada node daun yang memberikan
    peningkatan terbesar

    END WHILE

    Perbarui model dengan menambahkan decision tree yang baru

END FOR

Hasilkan prediksi akhir

```

```
RETURN Kelas hasil prediksi
```

Kode 2.3: Pseudocode Light Gradient Boosting Machine (LightGBM)

Pseudocode pada kode 2.3 menunjukkan proses pembentukan model LightGBM yang dilakukan secara bertahap (*gradient boosting*). Pada setiap iterasi, model terlebih dahulu menghitung nilai *gradient* dari fungsi kerugian, kemudian menerapkan mekanisme *Gradient-based One-Side Sampling* (GOSS) untuk mempertahankan sampel dengan gradien besar dan mengambil sebagian sampel dengan gradien kecil. Selanjutnya dibangun sebuah *decision tree* menggunakan strategi pertumbuhan pohon secara *leaf-wise*, di mana node daun yang memberikan peningkatan (*gain*) terbesar diprioritaskan untuk dikembangkan. Kualitas setiap kandidat *split* dievaluasi menggunakan nilai *Variance Gain*, kemudian pohon yang terbentuk ditambahkan ke dalam model untuk memperbaiki kesalahan prediksi pada iterasi sebelumnya. Proses tersebut diulang hingga jumlah *decision tree* yang ditentukan tercapai sehingga menghasilkan model prediksi akhir.

## 2.11 Ensemble Learning

Pendekatan *Ensemble Learning* dalam domain *machine learning* beroperasi dengan mengombinasikan beberapa model prediktif untuk melahirkan keputusan klasifikasi yang lebih robust, presisi, dan konsisten daripada performa satu model mandiri. Strategi ini bekerja memadukan output dari deretan *base learner*, yang secara simultan berimplikasi pada penguatan daya generalisasi sistem serta penekanan laju margin kesalahan [33].

### 2.11.1 Logistic Regression

*Logistic Regression* adalah algoritma klasifikasi yang digunakan untuk memperkirakan probabilitas terjadinya suatu peristiwa biner berdasarkan satu atau lebih variabel prediktor. Dalam penelitian ini, *Logistic Regression* tidak berperan sebagai model klasifikasi utama, melainkan digunakan sebagai *meta-learner* pada metode *Stacked Ensemble*. Sebagai *meta-learner*, model ini mempelajari pola hubungan antara prediksi yang dihasilkan oleh model-model dasar (*base learners*) dengan variabel target untuk menghasilkan prediksi akhir yang lebih optimal. Pemilihan *Logistic Regression* didasarkan pada kesederhanaan, efisiensi komputasi, serta kemampuannya dalam mengintegrasikan keluaran dari berbagai model sehingga dapat meningkatkan kemampuan generalisasi model ensemble [34].

### 2.11.2 Stacked Ensemble

Paradigma *Stacked Ensemble* merupakan salah satu manifestasi dari *ensemble learning* yang mengintegrasikan luaran prediksi dari jajaran model basis (*base learners*) lewat penambahan satu model penengah yang bertindak sebagai *meta-learner*. Mekanisme kerja arsitektur ini diformulasikan untuk mengeksploitasi kekuatan unik tiap algoritma dasar, sehingga mampu memicu akurasi prediksi yang lebih superior dibandingkan model tunggal konvensional [33].

$$\hat{y} = f_{\text{meta}}(f_1(x), f_2(x), \ldots, f_n(x)) \tag{2.11}$$

Persamaan tersebut menunjukkan bahwa prediksi akhir ($\hat{y}$) diperoleh dari model *meta-learner* ($f_{meta}$) yang menerima masukan berupa hasil prediksi dari beberapa model dasar ($f_1, f_2, \ldots, f_n$). Dalam penelitian ini, model dasar yang digunakan terdiri atas Random Forest, XGBoost, dan LightGBM, sedangkan Logistic Regression berperan sebagai *meta-learner* untuk menghasilkan prediksi akhir berdasarkan kombinasi keluaran dari ketiga model tersebut [33].

```
Input  : Dataset pelatihan D
Output : Kelas hasil prediksi

Latih model Random Forest

Latih model XGBoost

Hasilkan probabilitas prediksi dari masing-masing base learner

Bentuk dataset baru menggunakan probabilitas prediksi sebagai
    fitur masukan

Latih model Logistic Regression sebagai meta-learner

FOR setiap data uji DO

    Peroleh probabilitas prediksi dari Random Forest

    Peroleh probabilitas prediksi dari XGBoost

    Gunakan kedua probabilitas tersebut sebagai masukan ke meta-
    learner
```

```
21
22      Hasilkan prediksi akhir
23
24  END FOR
25
26  RETURN Kelas hasil prediksi
```

Kode 2.4: Pseudocode Stacked Ensemble

Pseudocode pada kode 2.4 menunjukkan proses pembentukan model *Stacked Ensemble*. Tahap pertama dilakukan dengan melatih dua *base learner*, yaitu Random Forest dan XGBoost menggunakan dataset pelatihan. Selanjutnya, kedua model menghasilkan probabilitas prediksi yang digunakan sebagai fitur masukan untuk membentuk dataset baru. Dataset tersebut kemudian digunakan untuk melatih Logistic Regression sebagai *meta-learner*. Pada tahap prediksi, probabilitas yang dihasilkan oleh Random Forest dan XGBoost kembali diberikan sebagai masukan ke *meta-learner* untuk menghasilkan prediksi akhir.

## 2.12 Hyperparameter Tuning

*Hyperparameter tuning* mengacu pada serangkaian prosedur pencarian kombinasi parameter paling ideal demi mendongkrak daya prediksi model komputasi. Studi ini mengadopsi teknik *RandomizedSearchCV* untuk eksekusi optimasi tersebut, di mana metode ini menguji sampel acak dari sekumpulan ruang parameter yang telah ditentukan sebelumnya, kemudian memvalidasi efisiensi masing-masing kombinasi lewat mekanisme *cross-validation* [35].

$$\theta^* = \arg\max_{\theta \in \Theta} Score(\theta) \tag{2.12}$$

Persamaan tersebut menunjukkan bahwa proses *hyperparameter tuning* bertujuan mencari kombinasi hyperparameter optimal $\theta^*$ dari ruang pencarian $\Theta$ yang menghasilkan nilai evaluasi terbaik. Pada metode *Random Search*, sejumlah kombinasi hyperparameter dipilih secara acak untuk dievaluasi, sehingga proses pencarian dapat menjelajahi ruang parameter yang lebih luas dengan biaya komputasi yang lebih rendah dibandingkan *Grid Search* [35].

## 2.13 Threshold Tuning

*Threshold tuning* merupakan proses penentuan nilai ambang (*threshold*) yang digunakan untuk mengubah probabilitas prediksi menjadi kelas biner. Nilai *threshold* yang optimal dapat meningkatkan performa model karena nilai default 0.5 tidak selalu memberikan hasil klasifikasi terbaik, terutama pada dataset dengan distribusi kelas yang tidak seimbang [36].

$$\hat{y} = \begin{cases} 1, & \text{jika } P(y=1|x) \geq \tau \\ 0, & \text{jika } P(y=1|x) < \tau \end{cases} \tag{2.13}$$

Persamaan tersebut menunjukkan proses konversi probabilitas prediksi $P(y=1|x)$ menjadi kelas biner menggunakan nilai ambang $\tau$. Pada penelitian ini, nilai *threshold* ditentukan berdasarkan nilai F1-Score tertinggi yang diperoleh dari berbagai kandidat *threshold*, sehingga menghasilkan keseimbangan yang lebih baik antara *precision* dan *recall* [36].

## 2.14 Evaluation Metrics

Penilaian kinerja model klasifikasi ditujukan untuk menguji kompetensi arsitektur dalam memproyeksikan estimasi pada sampel data baru yang diisolasi dari proses latih. Dalam riset ini, efektivitas model dikuantifikasi bersandarkan metrik *Confusion Matrix*, *Accuracy*, *Precision*, *Recall*, *F1-Score*, serta *Area Under the Receiver Operating Characteristic Curve* (ROC-AUC). Kombinasi instrumentasi evaluasi tersebut diaplikasikan guna menyajikan perspektif komprehensif mengenai validitas dan kualitas prediksi yang diproduksi model [37].

### 2.14.1 Confusion Matrix

*Confusion Matrix* merupakan instrumen tabulasi yang diterapkan untuk menyandingkan luaran prediksi model algoritma terhadap label kelas orisinalnya. Kerangka matriks ini tersusun atas empat parameter fundamental, meliputi *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN), yang bertindak sebagai basis kalkulasi fundamental bagi rumpun metrik penilaian performa pada pemodelan klasifikasi [37].

### 2.14.2 Accuracy

Metrik *Accuracy* bertugas mengalkulasi rasio total prediksi yang teridentifikasi secara tepat dari keseluruhan populasi sampel data yang diuji. Kendati mampu menyajikan estimasi global terkait reliabilitas model, parameter ini cenderung memberikan penilaian yang bias atau kurang akurat jika diimplementasikan pada kondisi data yang mengalami ketidakseimbangan kelas (*class imbalance*) [37].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

### 2.14.3 Precision

Sementara itu, *Precision* merefleksikan persentase ketepatan antara data yang benar-benar positif dibandingkan dengan akumulasi seluruh prediksi positif yang diputuskan oleh model. Nilai ini memegang peranan krusial pada skenario riset yang mewajibkan minimalisasi munculnya kekeliruan klasifikasi berupa *false positive* [37].

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

### 2.14.4 Recall

Di sisi lain, *Recall* atau *Sensitivity* menguji kompetensi model dalam menjaring kembali seluruh observasi yang secara faktual berstatus positif. Perolehan indeks *recall* yang tinggi mengonfirmasi keandalan sistem dalam mengidentifikasi sebagian besar entitas target yang ada di lapangan [37].

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

### 2.14.5 F1-Score

Sebagai jembatan penyeimbang, *F1-Score* diformulasikan lewat perhitungan rata-rata harmonik yang menyatukan metrik *precision* dan *recall* ke dalam satu parameter tunggal. Indikator ini kerap diandalkan dalam membedah problem klasifikasi dengan data timpang karena mampu memproyeksikan evaluasi kinerja yang jauh lebih objektif dan proporsional [37].

$$F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{2.17}$$

### 2.14.6 ROC-AUC

*Receiver Operating Characteristic* (ROC) merupakan representasi grafis yang memetakan interaksi antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) di sepanjang spektrum nilai *threshold* yang berbeda. Selaras dengan itu, metrik *Area Under the Curve* (AUC)—yang merepresentasikan cakupan luas wilayah di bawah kurva ROC tersebut—diadopsi sebagai parameter instrumen untuk mengukur signifikansi kapabilitas model dalam memisahkan antara kelas positif dan negatif. Pemerolehan skor AUC yang semakin mendekati angka satu mengindikasikan tingkat diskriminasi dan akurasi prediksi model yang kian superior [37].

$$TPR = \frac{TP}{TP + FN} \tag{2.18}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.19}$$

*True Positive Rate* (TPR) mengukur rasio keberhasilan model dalam mengidentifikasi seluruh sampel positif secara akurat, sementara *False Positive Rate* (FPR) merepresentasikan proporsi data negatif yang salah diidentifikasi ke dalam kelas positif. Konstruksi grafik ROC didasarkan pada plotting titik-titik koordinat pasangan nilai TPR dan FPR yang dihasilkan dari variasi nilai ambang batas (*threshold*), di mana nilai kuantitatif ROC-AUC diperoleh dari kalkulasi total luas wilayah di bawah kurva tersebut. Pemerolehan skor ROC-AUC yang semakin tinggi dan mendekati nilai mutlak 1 mengonfirmasi keandalan performa model yang

kian presisi dalam mengklasifikasikan kedua label kelas [37].

## 2.15 Explainable Artificial Intelligence (XAI) SHAP

*SHapley Additive exPlanations* (SHAP) merupakan salah satu metode dalam *Explainable Artificial Intelligence* (XAI) yang digunakan untuk menginterpretasikan kontribusi masing-masing fitur terhadap hasil prediksi model. Metode ini mengadaptasi konsep nilai Shapley dari teori permainan untuk menghitung kontribusi setiap fitur secara adil terhadap prediksi yang dihasilkan, sehingga dapat membantu meningkatkan transparansi dan interpretabilitas model [38].

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f(S \cup \{i\}) - f(S)] \tag{2.20}$$

Persamaan tersebut digunakan untuk menghitung nilai Shapley ($\phi_i$) dari fitur ke-$i$. Himpunan $F$ menyatakan seluruh fitur yang digunakan dalam model, sedangkan $S$ merupakan subset fitur yang tidak mengandung fitur ke-$i$. Nilai $f(S \cup \{i\}) - f(S)$ menunjukkan perubahan prediksi yang dihasilkan ketika fitur ke-$i$ ditambahkan ke dalam subset fitur $S$. Semakin besar nilai Shapley yang diperoleh, semakin besar kontribusi fitur tersebut terhadap hasil prediksi model [38].

## 2.16 Penelitian Terdahulu

Berbagai penelitian telah dilakukan untuk memprediksi risiko *readmission* pada pasien *Heart Failure* menggunakan pendekatan *machine learning*. Penelitian-penelitian tersebut memanfaatkan data rekam medis elektronik (*Electronic Health Records*), fitur klinis, maupun data dari basis data MIMIC dengan berbagai metode klasifikasi dan strategi seleksi fitur. Ringkasan penelitian terdahulu yang relevan dengan penelitian ini disajikan pada Tabel 2.2.

Tabel 2.2. Ringkasan penelitian terdahulu terkait prediksi readmission pasien Heart Failure

| No | Penelitian | Sampel | Sumber Data | Fitur | Metode dan Hasil |
|---|---|---|---|---|---|
| 1 | *Rajkumar et al. (2023)* | 93.354 pasien | DAD dan NACRS (Kanada) | Demografi, komorbiditas, klinis (prosedur, resep obat), dan biaya ekonomi. | Membandingkan RF, XGBoost, LightGBM, Ensemble Model. Ensemble Model menghasilkan *precision* 0.49 dan *recall* 0.68 dalam memprediksi readmission 30 hari. |
| 2 | *Sharma et al. (2022)* | 9.845 pasien | Data Administratif Rumah Sakit (Alberta Health Serv ices) | Data administratif, demografi, kode diagnosis ICD-9/10, dan komorbiditas. | Menguji LR, RF, XGBoost, Grad-Boost, dll. Model XGBoost memberikan hasil terbaik dengan AUC sebesar 0.654 untuk prediksi readmission dalam kurun waktu 30 hari. |
| 3 | *Song et al. (2024)* | 6687 pasien | Sichuan Provincial People's Hospital | Demografi, data laboratorium, ekokardiografi, dan terapi obat. | Membandingkan SVM, RF, KNN, AdaBoost, dll. Model XGBoost memberikan hasil terbaik dengan akurasi 0.81 dan *precision* 0.37. |

| No | Penelitian | Sampel | Sumber Data | Fitur | Metode dan Hasil |
|---|---|---|---|---|---|
| 4 | *Zheng et al. (2024)* | 746 pasien | Fifth Affiliated Hospital of Sun Yat-sen University | Demografi, tanda vital, tes laboratorium, ekokardiografi, dan komorbiditas. | Menerapkan XGBoost, RF, NN, LR. Model XGBoost menghasilkan AUC sebesar 0.722 untuk prediksi 90 hari. |
| 5 | *Shi et al. (2025)* | 2137 pasien | Sichuan Provincial People'sHospital | 29 variabel meliputi demografi, status klinis, pemeriksaan lab, dan pengobatan. | Menggunakan algoritma XGBoost tunggal memberikan hasil AUROC 0.678 dalam prediksi 1 *year unplanned readmission*. |
| 6 | *Luo et al. (2024)* | 718 pasien | Affiliated Lu'an Hospital of Anhui Medical University | Fitur klinis dasar, hasil laboratorium, tingkat keparahan lesi koroner, dan obat. | Membandingkan LR, BP, KNN, DT, dan XGBoost. Model XGBoost menghasilkan AUC tertinggi dengan 0.891 untuk prediksi 3 *year readmission*. |

| No | Penelitian | Sampel | Sumber Data | Fitur | Metode dan Hasil |
|---|---|---|---|---|---|
| 7 | *Pishgar et al. (2022)* | 3411 pasien | MIMIC-III | Informasi demografis, skor keparahan, riwayat medis, dan runtun waktu *event logs*. | Menggunakan pendekatan *process mining* (algoritma DREAM) gabungan dengan *Neural Network* (NN). Model yang diusulkan menghasilkan AUROC tertinggi sebesar 0.930 |