BAB 2
LANDASAN TEORI

2.1 Kanker Payudara (Breast Cancer)

Kanker payudara merupakan salah satu jenis kanker paling umum dan penyebab utama kematian terkait kanker pada perempuan di seluruh dunia. Pada tahun 2022, dilaporkan terdapat sekitar 20 juta kasus baru kanker secara global. Di antara berbagai jenis kanker, kanker payudara merupakan salah satu yang paling sering didiagnosis dan menjadi penyebab utama kematian akibat kanker, dengan jumlah kasus mencapai 2,3 juta dan sekitar 685.000 kematian yang ditimbulkannya [1]. Penyakit ini diklasifikasikan menjadi beberapa subtipe molekuler utama berdasarkan ekspresi reseptor, yaitu luminal A, luminal B, HER2-positif, dan triple-negative breast cancer (TNBC), yang masing-masing memiliki karakteristik biologis dan respons terapi yang berbeda. Faktor risiko utama meliputi faktor genetik (seperti mutasi BRCA1/2), hormonal (paparan estrogen jangka panjang), serta faktor lingkungan dan gaya hidup [2].

Kanker payudara diklasifikasikan ke dalam beberapa tahapan berdasarkan penyebaran tumor dan keterlibatan jaringan sekitarnya atau organ lain. Stadium 0, juga dikenal sebagai karsinoma in situ (*carsinoma in situ* merupakan kanker non-invasif di mana sel abnormal masih terbatas pada lokasi asalnya tanpa menyerang jaringan di sekitarnya. Stadium 1 menandai awal kanker invasif, terbagi menjadi 1A (tumor $\geq$ 2cm, tanpa keterlibatan kelenjar getah bening) dan 1B (kelompok kecil sel kanker di kelenjar getah bening). Stadium 2 menunjukkan tumor yang lebih besar (2-5 cm) atau keterlibatan kelenjar getah bening terbatas (2A: kelenjar getah bening terkena tetapi tidak menyebar ke dada; 2B: tumor ukuran lebih dari 5 cm tanpa penyebaran ke kelenjar getah bening aksila). Stadium 3 menandakan kanker yang telah menyebar secara lokal, dengan subkategori 3A (menyebar ke 4-9 kelenjar getah bening), 3B (tumor menyerang kulit dada atau menyebabkan peradangan), dan 3C (penyebaran luas ke kelenjar getah bening, termasuk tulang selangka atau aksila). Stadium 4 adalah kanker payudara metastatik, di mana penyakit telah menyebar ke organ jauh seperti paru-paru, hati, tulang, atau otak, sehingga pengobatannya lebih kompleks [3].

Terdapat juga sistem pengelompokan stadium kanker payudara dengan klasifikasi TNM. Sistem TNM dari American Joint Committee on Cancer (AJCC)

yang paling sering digunakan untuk *stage labeling* kanker payudara. Berikut merupakan tabel klasifikasinya.

Tabel 2.1. Stage Groupings Based on the TNM classification of the Breast Cancer

Stage	T level	N level	M level
Stage 0	Tis	N0	M0
Stage I	T1	N0	M0
Stage IIA	T0	N1	M0
	T1	N1	M0
	T2	N0	M0
Stage IIB	T2	N1	M0
	T3	N0	M0
Stage IIIA	T0	N2	M0
	T1	N2	M0
	T2	N2	M0
	T3	N1	M0
	T3	N2	M0
Stage IIIB	T4	Any N	M0
Stage IIIC	Any T	N3	M0
Stage IV	Any T	Any N	M1

Sumber: [12]

Klasifikasi TNM pada kanker payudara digunakan untuk menggambarkan stadium penyakit berdasarkan tiga parameter utama, yaitu ukuran tumor (*Tumor Size*), status kelenjar getah bening (*Lymph Node Status*), dan metastasis jauh (Distant Metastasis). Pada kategori *Tumor Size* (T), klasifikasi dimulai dari Tx, yang menunjukkan bahwa ukuran tumor tidak dapat diukur atau tidak ditemukan. T0 berarti tidak terdapat bukti adanya tumor primer. Tis mengindikasikan kanker in situ, yaitu kanker yang belum menyebar ke jaringan payudara di sekitarnya. Kategori T1 sampai T4 mengacu pada ukuran tumor: T1 untuk tumor berukuran kurang dari 2 cm, T2 untuk ukuran 2-5 cm, T3 untuk tumor lebih dari 5 cm, dan T4 untuk tumor dengan ukuran berapa pun yang telah menyusup ke kulit atau menempel ke dinding dada. Untuk kategori *Lymph Node Status* (N), Nx menunjukkan bahwa status kelenjar getah bening tidak dapat diukur, sedangkan N0 berarti tidak ada penyebaran ke kelenjar getah bening. N1 mengacu pada metastasis ke kelenjar getah bening aksila di sisi yang sama dan masih bisa digerakkan.

N2 berarti kelenjar getah bening tersebut menyatu satu sama lain atau dengan struktur lain. N3 menandakan penyebaran ke kelenjar getah bening di bawah tulang dada (*internal mammary nodes*). Pada parameter *Distant Metastasis* (M), Mx menunjukkan bahwa keberadaan metastasis jauh tidak dapat ditentukan. M0 berarti tidak ditemukan metastasis jauh, sedangkan M1 menandakan bahwa metastasis jauh telah ditemukan [12].

2.2 Ekspresi Gen RNA-seq

Ekspresi gen merupakan proses biologis di mana informasi genetik yang tersimpan dalam DNA digunakan untuk mensintesis produk fungsional, seperti RNA atau protein [13]. Kumpulan RNA yang ditranskripsi pada kondisi dan waktu tertentu mencerminkan keadaan sel saat itu, dan dapat mengungkap mekanisme patologis yang mendasari suatu penyakit. Data ekspresi gen dapat dihasilkan melalui metode RNA sequencing (RNA-Seq). Data ekspresi gen RNA-Seq adalah jenis data ekspresi gen yang diperoleh secara khusus menggunakan teknologi RNA sequencing berbasis Next-Generation Sequencing (NGS). RNA-Seq memungkinkan pengukuran ekspresi gen dengan tingkat ketelitian dan jangkauan dinamis yang tinggi [14]. Proses terbentuknya data ekspresi gen RNA-Seq terdiri dari beberapa tahapan penting. Pertama, RNA dipotong menjadi fragmen kecil dan ditranskripsi balik menjadi DNA komplemen (cDNA). Fragmen cDNA ini selanjutnya diperkuat melalui proses amplifikasi dan diberi adaptor agar dapat dibaca oleh mesin sekuensing. Proses sekuensing menggunakan platform seperti Illumina akan menghasilkan jutaan potongan pendek RNA (*reads*), yang kemudian di-mapping ke genom atau transkriptom. Setelah mapping, jumlah reads yang terpetakan ke masing-masing gen dihitung untuk mendapatkan nilai ekspresi (biasanya dalam bentuk count). Nilai ini kemudian dinormalisasi untuk menghilangkan bias teknis seperti perbedaan panjang gen atau kedalaman sekuensing. Hasil akhirnya adalah data kuantitatif yang merepresentasikan tingkat ekspresi masing-masing gen [14].

2.3 Differentially Expressed Genes (DEG)

Analisis DEG (*Differential Expression Gene*) adalah teknik dalam biologi molekuler yang digunakan untuk membandingkan tingkat ekspresi gen antara dua atau lebih kelompok sampel, misalnya antara jaringan sehat dan jaringan

penyakit, atau sel yang mendapatkan perlakuan berbeda. Tujuan utama dari analisis DGE adalah untuk mengidentifikasi gen-gen yang menunjukkan perbedaan ekspresi yang signifikan antara kelompok yang dibandingkan. Teknik ini sangat berguna untuk menemukan gen-gen yang terlibat dalam proses biologis tertentu, penyakit, atau respon terhadap pengobatan. Dengan demikian, DEG dapat memberi informasi penting tentang regulasi gen dan mekanisme biologis yang mendasarinya. Analisis ini umumnya dilakukan dalam penelitian penyakit untuk menemukan penanda biologis (biomarker) yang dapat digunakan untuk diagnosis, prognosis, atau mengevaluasi efektivitas suatu pengobatan [15]. Dalam melakukan analisis DEG, dapat dilakukan dengan menggunakan berbagai *package R* yang telah dikembangkan dan disediakan, seperti *edgeR*, *Deseq2*, dan *limma*.

Identifikasi DEG guna mendapatkan gen-gen yang memiliki nilai ekspresi yang signifikan satu antar lainnya pada data RNA-seq dapat dimulai dari tahap normalisasi dan pra-proses data. Tahap ini dilakukan untuk mengurangi *noise* atau gangguan dengan cara menghilangkan variabilitas yang terkait dengan masalah teknis, sehingga hasil antar sampel menjadi lebih mudah dibandingkan. Setelah data bersih dan konsisten, langkah berikutnya adalah memilih model analisis yang paling sesuai dengan karakteristik data. Seperti metode limma, metode ini menawarkan pendekatan yang kuat dan fleksibel [16]. Lalu selanjutnya, menganalisis data yang sudah diproses untuk menemukan gen-gen yang memang menunjukkan perbedaan ekspresi yang signifikan. Gen-gen ini bisa memberikan informasi penting mengenai mekanisme penyakit, target obat yang potensial, atau bahkan sebagai penanda untuk diagnosis dan prognosis [15]. Identifikasi DEG merupakan proses krusial yang membutuhkan pemilihan *cutoff* yang cermat. Pemilihan ukuran signifikansi statistik, seperti nilai *p* (*p-value*), dan nilai *p* yang telah disesuaikan (*adjusted p-value*) dengan berbagai teknik seperti *false discovery rate* (FDR), serta besarnya perbedaan yang diwakili oleh *log2 fold change* (*logFC*), sangat penting karena secara langsung memengaruhi keandalan hasil dan interpretasi data. Nilai *p* yang terlalu tinggi dapat menyebabkan *false negative*, sementara nilai yang terlalu rendah dapat menghasilkan *false positive*. Namun, pemilihan *cutoff* untuk *logFC* dan *p-value* sebaiknya disesuaikan dengan spesifikasi data dan kondisi eksperimen. Ini berarti bahwa tidak ada nilai *cutoff* yang bersifat universal, melainkan harus ditentukan secara spesifik untuk setiap kasus. Kombinasi kedua kriteria *cutoff* (*logFC* dan *p-value/adjusted p-value*) memberikan keunggulan yang lebih dalam mengidentifikasi DEG.

Proses analisis DEG dengan menggunakan paket limma dapat

diinterpretasikan dengan menggunakan Algoritma 1 berikut ini :

Algorithm 1 *Differential Expression Analysis using Limma Package*

function DEG_ANALYSIS_WITH_LIMMA(data ekspresi, label)
    Input:
        data ekspresi: matriks ekspresi gen (fitur × sampel)
        label: label kelas (“early”, “late”)
    Output:
        hasil DEG: tabel gen yang terekspresi secara berbeda

    Ambil label kelas dari kolom pertama
    Ambil data ekspresi (selain kolom label)
    Konversi nilai menjadi numerik dan transposisikan
    Buat desain model: `model.matrix(labels)`
    Fit model: `lmFit()`
    Moderasi variansi: `eBayes()`
    Ambil hasil DEG: `topTable()`
    Simpan hasil ke file `.csv`
end function

2.4 Feature Selection

Feature selection adalah proses mengidentifikasi dan memilih fitur-fitur yang paling relevan dari sebuah dataset, sambil membuang fitur-fitur yang redundan atau tidak relevan. Proses ini merupakan langkah krusial dalam pipeline machine learning, terutama saat menangani dataset berdimensi tinggi. Dengan mengurangi dimensi data, feature selection dapat meningkatkan performa model, menurunkan biaya komputasi, dan meningkatkan interpretabilitas. Tujuan utama dari feature selection adalah memastikan bahwa model machine learning fokus pada variabel-variabel yang paling informatif, sehingga menghasilkan generalisasi dan akurasi prediksi yang lebih baik. Selain itu, feature selection juga menyederhanakan model, membuatnya lebih mudah dipahami dan diimplementasikan, serta mengurangi risiko overfitting [17]. Terdapat berbagai macam method machine learning yang sudah dikembangkan untuk melakukan seleksi fitur. berikut penjelasan terkait method-method yang digunakan dalam penelitian ini.

2.4.1 Analysis of Variance (ANOVA)

Analysis of Variance / ANOVA merupakan metode statistik yang digunakan untuk menentukan apakah terdapat perbedaan yang signifikan antara rata-rata dari dua atau lebih kelompok. Dalam seleksi fitur (feature selection) untuk pengklasifikasian, ANOVA digunakan untuk mengukur sejauh mana sebuah fitur dapat membedakan antar kelas [18]. Untuk mengetahui sejauh mana sebuah fitur dapat dengan baik membedakan antar kelasnya adalah dengan membandingkan variansi antar kelas (*between-class variance*) dan variansi dalam kelas (*within-class variance*). Fitur yang memiliki variansi antar kelas yang besar dibandingkan dengan variansi dalam kelas dianggap lebih informatif atau diskriminatif, karena fitur tersebut lebih mampu membedakan kelas-kelas yang berbeda.

$$\mathrm{SSB}_j = \sum_{m=1}^{k} n_m \left(\bar{x}_{j,m} - \bar{x}_j\right)^2 \tag{2.1}$$

Nilai variansi antar kelas (*between-class variance*) dapat dihitung dengan Rumus 2.1. Rumus tersebut menghasilkan nilai yang menunjukkan seberapa jauh rata-rata tiap kelas menyimpang dari rata-rata keseluruhan.

$$\mathrm{SSW}_j = \sum_{m=1}^{k} (n_m - 1) s_{j,m}^2 \tag{2.2}$$

Nilai variansi dalam kelas (*within-class variance*) dapat dihitung dengan Rumus 2.2. Rumus tersebut mengukur variansi data dalam masing-masing kelas.

$$F_j = \frac{\mathrm{SSB}_j/(k-1)}{\mathrm{SSW}_j/(n-k)} \tag{2.3}$$

Rumus 2.3 menunjukkan perhitungan nilai F (F-statistic) untuk fitur ke-*j*. Nilai ini digunakan untuk menentukan seberapa baik fitur *j* dalam membedakan antar kelas.

Keterangan Notasi:

- k : jumlah kelas
- n_m : jumlah sampel dalam kelas ke-m

- $\bar{x}_{j,m}$: rata-rata fitur ke-j dalam kelas ke-m
- $\bar{x}_j$: rata-rata keseluruhan dari fitur ke-j
- $s^2_{j,m}$: varians fitur ke-j dalam kelas ke-m
- n : total seluruh sampel dari semua kelas

Pada ANOVA *F-value* yang tinggi menunjukan bahwa fitur tersebut memiliki variansi antar kelas lebih besar dibandingkan dengan variansi dalam kelasnya yang berarti fitur tersebut baik dalam membedakan kelas. Sebaliknya, jika *F-value* rendah maka fitur tersebut kurang efektif dalam membedakan kelas karena variansi antar kelas hampir sama atau lebih kecil dari variansi dalam kelas [18].

2.4.2 Recursive Feature Elimination (RFE)

Recursive Feature Elimination merupakan salah satu metode *wrapper* yang melibatkan pelatihan dan evaluasi model machine learning secara berulang untuk menentukan subset fitur yang optimal. Metode ini melakukan seleksi fitur dengan cara menghapus secara bertahap fitur-fitur yang dianggap paling tidak penting, kemudian mengevaluasi kembali performa model pada setiap langkahnya. Proses ini dimulai dengan seluruh fitur yang tersedia, lalu secara iteratif mengeliminasi fitur-fitur dengan kontribusi paling kecil terhadap model, yang ditentukan berdasarkan metrik seperti koefisien fitur atau skor pentingnya fitur [17]. RFE sangat efektif ketika digunakan bersama dengan algoritma klasifikasi seperti *Support Vector Machines* (SVM) dan *Logistic Regression*, dimana bobot dari model klasifikasi digunakan sebagai kriteria untuk menentukan kepentingan fitur. Proses eliminasi fitur dilakukan secara rekursif hingga diperoleh subset fitur optimal yang memberikan performa klasifikasi terbaik. Proses seleksi fitur dengan menggunakan *recursive feature selection* dapat diinterpretasikan dengan menggunakan pseudocode Algoritma 2 berikut ini [19]:

2.5 Logistic Regression (LR)

Logistic Regression adalah model klasifikasi berbasis probabilistik yang digunakan untuk memprediksi kemungkinan suatu data termasuk dalam salah satu dari dua kelas (misalnya: positif atau negatif). Model ini merupakan discriminative classifier, artinya model langsung mempelajari hubungan antara fitur

Algorithm 2 *Recursive Feature Elimination (RFE)*

function RFE_FEATURE_SELECTION(X, y, n_features)
    Input:
        X: DataFrame berisi fitur
        y: Label kelas dalam bentuk Series
        $n\_features$: Jumlah fitur akhir yang diinginkan
    Output:
        $X\_rfe\_selected$: DataFrame hanya dengan fitur terpilih
        $selected\_features\_name$: Nama fitur yang dipilih

    Buat model klasifikasi yang sesuai
    Inisialisasi algoritma RFE dengan model dan jumlah fitur yang diinginkan
    for hingga jumlah fitur = $n\_features$ **do**
        Latih model untuk menghitung skor kepentingan fitur
        Hapus fitur dengan kontribusi terendah
        Perbarui subset fitur
    end for
    Simpan nama fitur terpilih sebagai $selected\_features\_name$
    Bentuk ulang $X\_rfe\_selected$ menggunakan fitur-fitur terpilih
    return $X\_rfe\_selected$, $selected\_features\_name$
end function

dan label (tanpa perlu memodelkan distribusi fitur secara eksplisit seperti pada Naive Bayes). *Logistic regression* memetakan output ke dalam rentang probabilitas [0, 1] menggunakan fungsi sigmoid [20]. Untuk mendapatkan nilai probabilitas tersebut, dimulai dengan menghitung skor linear (*logit*) pada masing-masing fitur dengan Persamaan 2.4 berikut:

$$z = w \cdot x + b \tag{2.4}$$

Keterangan:

- w: bobot model untuk setiap fitur.
- x: nilai fitur dari satu sampel.
- b: bias atau *intercept* (konstanta).
- z: skor prediksi (bukan probabilitas), bisa negatif atau positif.

Kemudian, hasil dari *logit* (z) tersebut digunakan pada Persamaan 2.5 untuk Mengubah skor z menjadi nilai antara 0 dan 1 agar bisa ditafsirkan sebagai

probabilitas.

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}} = P(y = 1 \mid x) \tag{2.5}$$

Keterangan:

- $\hat{y}$: probabilitas bahwa input x termasuk ke kelas 1.
- $\sigma(z)$: fungsi sigmoid.

Setelah mendapatkan hasil probabilitas, suatu fitur baru dapat dikategorikan/diklasifikasikan. Kriteria keputusan kelas dapat dilihat pada persamaan 2.6. Ketika $\hat{y} \geq 0.5$ maka akan dikategorikan sebagai kelas positif dan jika bernilai < 0.5 maka akan dikategorikan sebagai kelas negatif [21].

$$\text{decision}(x) = \begin{cases} 1(Positif) & \text{jika } P(y = 1 \mid x) \geq 0.5 \\ 0(Negatif) & \text{jika } P(y = 1 \mid x) < 0.5 \end{cases} \tag{2.6}$$

Selanjutnya untuk mengukur seberapa baik prediksi model dari kelas/label sebenarnya, digunakan fungsi *log loss* (*cross-entropy* yang dapat dilihat pada Persamaan 2.7 sebagai berikut:

$$L(\hat{y}, y) = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})] \tag{2.7}$$

Keterangan:

- y: label sebenarnya dari data (0 atau 1).
- $\hat{y}$: probabilitas hasil prediksi.
- Jika $y = 1$, hanya bagian $\log(\hat{y})$ yang aktif.
- Jika $y = 0$, hanya bagian $\log(1-\hat{y})$ yang aktif.

Ketika suatu model *Logistic Regression* menggunakan regulasi L1 maka akan berdampak pada fungsi *log loss* untuk mencegah overfitting dan memaksa beberapa bobot menjadi nol dan sehingga menghasilkan model yang lebih sederhana. Persamaan 2.8 merupakan total *loss* dengan L1.

$$\text{Total Loss} = L(\hat{y}, y) + \alpha \sum_{j=1}^{n} |w_j| \tag{2.8}$$

Keterangan:

- α: parameter regularisasi (*hyperparameter*) yang mengontrol kekuatan penalti.
- $\sum_{j=1}^{n} |w_j|$: jumlah absolut semua bobot (norma L1).
- Tidak termasuk bias b dalam penalti.

$$\frac{\partial L}{\partial w_j} = (\hat{y} - y) \cdot x_j + \alpha \cdot \text{sign}(w_j) \tag{2.9}$$

Keterangan:

- $(\hat{y} - y) \cdot x_j$: komponen dari turunan *loss cross-entropy*.
- $\text{sign}(w_j)$: fungsi tanda, bernilai:

  $+1$ jika $w_j > 0$
  -1 jika $w_j < 0$
  0 jika $w_j = 0$

- Tambahan $\alpha \cdot \text{sign}(w_j)$ berasal dari turunan regularisasi L1.

Persamaan 2.9 ini menunjukkan gradien turunan fungsi loss terhadap bobot pada *Logistic Regression* dengan penalti L1. Gradien ini digunakan dalam algoritma gradient descent untuk memperbarui bobot selama proses pelatihan (dapat merujuk ke Persamaan 2.10.

$$w_j \leftarrow w_j - \eta \cdot \frac{\partial L}{\partial w_j} \tag{2.10}$$

Dengan:

- η: *learning rate* (kecepatan pembelajaran).

2.6 Random Forest

Random forest adalah teknik pembelajaran mesin yang menggabungkan banyak pohon keputusan untuk mengurangi korelasi di antara data fitur. Salah satu keunggulan random forest adalah akurasi tingginya dibandingkan dengan pendekatan lain seperti bagging dan boosting. Random forest juga berfungsi secara efektif pada database besar dan dapat mengakomodasi banyak variabel, memungkinkan kita untuk menganalisis ribuan variabel input tanpa perlu menghapus satu pun input. Random Forest (RF) mengurangi korelasi antara pohon keputusan dengan menggunakan pemilihan sampel dan fitur secara acak. Awalnya, sejumlah data yang setara dipilih secara acak dari sampel pelatihan dalam data pelatihan asli. Selanjutnya, subset dari fitur-fitur dipilih secara acak untuk membangun pohon keputusan. Penggunaan kedua bentuk randomisasi ini menghasilkan penurunan korelasi antara setiap pohon keputusan, sehingga mengurangi potensi kesalahan yang disebabkan oleh overfitting dan meningkatkan akurasi model [22].

Untuk klasifikasi biner, prediksi akhir pada data baru x dilakukan berdasarkan hasil mayoritas dari seluruh pohon keputusan:

$$f(x) = \arg\max_{y} \sum_{j=1}^{J} \mathbb{I}(h_j(x) = y) \tag{2.11}$$

dengan:

- $h_j(x)$: prediksi dari pohon ke-j untuk input x
- $\mathbb{I}$: fungsi indikator, bernilai 1 jika argumen benar, dan 0 jika salah

Tujuan utama Random Forest adalah meminimalkan rata-rata error terhadap distribusi sebenarnya dari data. Dalam konteks klasifikasi biner, digunakan *zero-one loss*, yaitu:

- *Loss* = 0 jika prediksi benar
- *Loss* = 1 jika prediksi salah

Gini Index atau *Gini Impurity* digunakan untuk mengukur ketidakhomogenan suatu node dalam pohon keputusan. Nilai Gini digunakan sebagai kriteria untuk memilih split terbaik di setiap node.

Untuk sebuah node dengan K kelas, Gini index dihitung dengan rumus berikut:

$$\text{Gini} = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.12}$$

dengan:

- p_k: proporsi data yang termasuk kelas ke-k pada node tersebut.

Dalam kasus klasifikasi biner ($K = 2$), misalnya untuk dua kelas *positif* dan *negatif*, Gini index menjadi:

$$\text{Gini} = 2 \cdot p \cdot (1 - p) \tag{2.13}$$

di mana p adalah proporsi salah satu kelas (misalnya kelas positif) pada node tersebut.

Keterangan:

- Gini = 0: semua data dalam node berasal dari satu kelas (pure).
- Gini mendekati 0.5: distribusi kelas seimbang (misalnya 50%-50%).

Selama proses pembuatan pohon, algoritma akan memilih split yang menghasilkan penurunan Gini paling besar dari parent ke child node.

Proses klasifikasi dengan menggunakan Random Forest dapat diinterpretasikan dengan menggunakan pseudocode Algoritma 3 berikut ini [23]:

2.7 Extreme Gradient Boosting (XGBoost)

XGBoost merupakan algoritma pembelajaran mesin ensemble yang bergantung pada pohon keputusan dan menggunakan metode gradient-boosting. XGBoost adalah metode gradient tree boosting yang dioptimalkan yang menghasilkan pohon keputusan secara berurutan. Algoritma ini mampu melakukan perhitungan terkait dengan sangat cepat di semua lingkungan komputasi [24]. XGBoost berkembang dari pohon keputusan tunggal menjadi ensemble melalui teknik bagging yang menggabungkan prediksi dari multiple pohon menggunakan voting mayoritas dan seleksi fitur acak. Algoritma ini meningkatkan performa

Algorithm 3 *Random Forest - Klasifikasi Biner*

function RANDOM_FOREST(Data $\mathscr{D}$, jumlah_pohon J, jumlah_fitur_acak m)
    Input:
        $\mathscr{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$: Dataset pelatihan
        J: Jumlah pohon dalam hutan
        m: Jumlah fitur acak per node (misal: $\sqrt{M}$)
    Output:
        H: Kumpulan semua pohon $h_1, \ldots, h_J$

    for $j = 1$ hingga J **do**
        Ambil sampel bootstrap $\mathscr{D}_j$ dari $\mathscr{D}$ (ukuran N)
        Bangun pohon h_j dengan:
            a. Mulai dari root (semua data)
            b. Selama node belum memenuhi kondisi berhenti:
                i. Pilih acak m fitur dari M fitur total
                ii. Cari split terbaik berdasarkan impurity (misal: Gini)
                iii. Split node menjadi dua child node
            c. Pohon dibuat tanpa pruning
    end for
    return $H = \{h_1, h_2, \ldots, h_J\}$
end function

function PREDICT_CLASS(x, H)
    Kumpulkan prediksi dari semua pohon: $y_j = h_j(x)$ untuk $j = 1$ hingga J
    Hitung jumlah vote untuk setiap kelas
    return kelas dengan vote terbanyak (mayoritas)
end function

dengan meminimalkan error secara berurutan menggunakan *gradient descent*. XGBoost mengoptimalkan gradient boosting tradisional melalui tiga cara utama: menangani *missing values*, mencegah *overfitting* dengan pemrosesan paralel, dan meningkatkan efisiensi sistem melalui *parallelization*, *tree pruning*, dan optimasi hardware.

XGBoost membangun model secara iteratif dengan menambahkan pohon keputusan (*decision tree*) baru yang dilatih untuk memperbaiki kesalahan prediksi dari model sebelumnya. Proses ini mengikuti pendekatan *additive boosting* [25, 26]. Model prediksi pada iterasi ke-p dapat dituliskan dengan:

$$\hat{f}_i^{(p)} = \hat{f}_i^{(p-1)} + f_p(x_i) \tag{2.14}$$

dengan:

- $\hat{f}_i^{(p)}$: prediksi model pada iterasi ke-p,
- $f_p(x_i)$: prediksi dari pohon ke-p untuk input x_i.

Fungsi objektif pada XGBoost terdiri dari dua komponen: fungsi *loss* dan fungsi regularisasi, ditulis sebagai:

$$\mathscr{L}^{(p)} = \sum_{i=1}^{n} l(y_i, \hat{y}i^{(p)}) + \sum k = 1^p \Omega(f_k) \tag{2.15}$$

di mana l adalah fungsi loss (misalnya *log loss* untuk klasifikasi biner), dan $\Omega(f_k)$ adalah fungsi regularisasi. Fungsi regularisasi dirumuskan sebagai:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda |w|^2 \tag{2.16}$$

- T: jumlah leaf dalam pohon,
- w: skor output pada setiap leaf,
- γ: penalti untuk jumlah leaf (mengontrol kompleksitas struktur pohon),
- λ: parameter regularisasi.

Untuk mengoptimalkan fungsi objektif, XGBoost menggunakan pendekatan *second-order Taylor expansion*:

$$\mathscr{L}^{(p)} \approx \sum_{i=1}^{n} \left[g_i f_p(x_i) + \frac{1}{2} h_i f_p(x_i)^2 \right] + \Omega(f_p) \tag{2.17}$$

dengan:

- $g_i = \frac{\partial l(y_i, \hat{y}_i)}{\partial \hat{y}_i}$: gradien,
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$: hessian.

XGBoost mencari split terbaik pada node dengan memaksimalkan gain, yang dirumuskan sebagai:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \tag{2.18}$$

dengan:

- G_L, G_R: jumlah gradien pada cabang kiri dan kanan,
- H_L, H_R: jumlah hessian pada cabang kiri dan kanan.

Gain digunakan untuk menentukan apakah pemisahan node layak dilakukan. Jika nilai gain lebih kecil dari γ, maka pemisahan tidak dilakukan untuk mencegah overfitting.

Untuk melakukan klasifikasi biner, hasil akhir dari pohon-pohon keputusan akan dijumlahkan, lalu dilewatkan ke fungsi sigmoid:

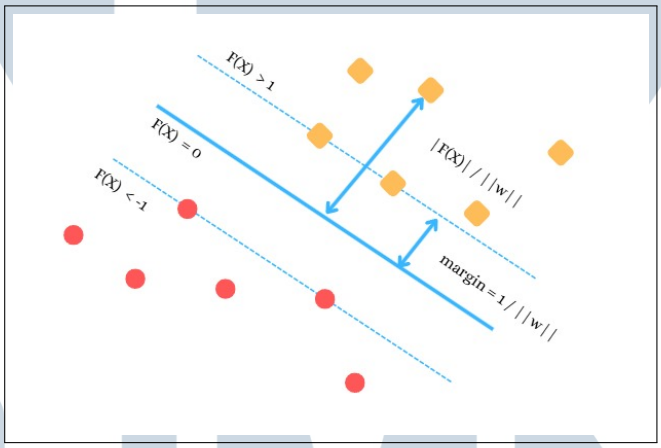
$$P(y = 1 \mid x) = \hat{y} = \frac{1}{1 + e^{-\sum_{k=1}^{p} f_k(x)}} \tag{2.19}$$

Jika probabilitas prediksi $\hat{y} > 0.5$, maka input diklasifikasikan ke dalam kelas positif, dan sebaliknya jika $\hat{y} < 0.5$ maka input akan diklasifikasikan kedalam kelas negatif.

2.8 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah salah satu metode supervised learning yang digunakan dalam klasifikasi (Support Vector Classification) dan regresi (Support Vector Regression) yang menggabungkan teori-teori komputasi selama

beberapa dekade, seperti konsep margin hyperplane. Konsep dasar SVM dapat dijelaskan secara sederhana sebagai pencarian hyperplane terbaik yang merupakan pemisah optimal antara dua kelas data, yang ditentukan dengan mengukur titik maksimum dari margin hyperplane. Margin adalah jarak antara hyperplane dan data terdekat di setiap kelas, dimana data yang paling dekat dengan hyperplane terbaik disebut support vector. SVM memiliki keunggulan dibandingkan teknik klasifikasi lainnya karena dapat menyelesaikan masalah klasifikasi dan regresi baik yang bersifat linear maupun non-linear. Untuk masalah non-linear, SVM menggunakan konsep kernel dalam ruang berdimensi tinggi, dimana pada ruang dimensional tersebut akan dicari hyperplane terbaik dengan memaksimalkan jarak antar kelas, dan proses pencarian hyperplane terbaik inilah yang menjadi inti dari proses SVM [27, 28].



Gambar 2.1. Ilustrasi hyperplnae optimal yang memaksimalkan margin dalam dua ruang dimensi

Sumber: [28]

Pada SVM, klasifikasi dilakukan dengan mencari optimal *hyperplane* yang dapat memisahkan dua kelas data secara maksimal. *Hyperplane* adalah suatu bidang dalam ruang berdimensi *n* dapat dirumuskan sebagai berikut:

$$f(x) = w \cdot x - b = 0 \tag{2.20}$$

keterangan:

- w: vektor bobot (*weight vector*) yang tegak lurus terhadap hyperplane.
- b: bias / *intercept* yang mengatur posisi hyperplane terhadap asal koordinat.

Fungsi $f(x)$ digunakan untuk menentukan di sisi mana sebuah titik data x berada terhadap hyperplane. Berdasarkan hasil evaluasi dari fungsi tersebut, proses klasifikasi dilakukan sebagai berikut:

$$\begin{cases} f(x_i) > 0 \Rightarrow y_i = +1 \\ f(x_i) < 0 \Rightarrow y_i = -1 \end{cases} \tag{2.21}$$

Persamaan tersebut berarti bahwa jika sebuah titik x_i berada di sisi yang benar dari hyperplane sesuai label kelasnya y_i. Misalnya fungsi $f(x_i)$ harus menghasilkan nilai > 0 untuk dikategorikan dalam kelas positif dan harus menghasilkan nilai < 0 untuk dikategorikan dalam negatif.

Agar klasifikasi dilakukan dengan benar dan jarak margin antara dua kelas dapat dimaksimalkan, maka Support Vector Machine (SVM) berusaha mencari vektor bobot w dan bias b yang meminimalkan norma dari vektor w. Hal ini dapat dilakukan dengan Persamaan 2.22

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{2.22}$$

Dengan syarat:

$$y_i(w \cdot x_i - b) \geq 1, \quad \forall i \tag{2.23}$$

Jika data tidak dapat dipisahkan secara sempurna menggunakan hyperplane linear maka digunakan pendekatan *soft-margin SVM*. Pendekatan ini memperbolehkan adanya sedikit kesalahan klasifikasi, diinterpretasikan sebagai *slack* ξ_i untuk setiap titik data. Tujuan dari *soft-margin SVM* adalah tetap memaksimalkan margin, tetapi juga meminimalkan jumlah kesalahan klasifikasi. Untuk persamaannya dapat dilihat pada Persamaan 2.24

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \tag{2.24}$$

Dengan syarat:

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{2.25}$$

Di mana parameter $C > 0$ berfungsi sebagai pengatur trade-off antara dua tujuan, yaitu:

- Margin yang besar ($\|w\|^2$ kecil)
- Jumlah kesalahan klasifikasi yang rendah ($\sum \xi_i$ kecil)

Untuk menyelesaikan masalah optimasi pada SVM, digunakan pendekatan *Lagrange Multiplier* α_i. Proses ini memungkinkan kita beralih dari bentuk *primal* (langsung mengoptimasi w, b) ke bentuk *dual*. Persamaannya sebagai berikut:

$$\mathscr{J}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i - b) - 1] \tag{2.26}$$

keterangan:

- w: vektor bobot (*weight vector*).
- b: bias atau *intercept*.
- α_i: Lagrange multiplier untuk data ke-i.
- x_i: vektor fitur dari data ke-i.
- y_i: label kelas dari data ke-i, bernilai $+1$ atau -1.
- m: jumlah total data pelatihan.

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i, \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.27}$$

keterangan:

- Turunan pertama: w dituliskan sebagai kombinasi linier dari data pelatihan.
- Turunan kedua: kendala agar hasil klasifikasi konsisten dengan label.

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{2.28}$$

Persamaan 2.28 merupakan fungsi yang dioptimasi dalam bentuk dual. Hanya melibatkan produk skalar $x_i \cdot x_j$, yang dapat digantikan dengan kernel $K(x_i, x_j)$ untuk SVM non-linear.

$$0 \leq \alpha_i \leq C \tag{2.29}$$

Dengan:

- C: parameter regularisasi untuk mengontrol toleransi kesalahan.
- Kendala ini memungkinkan soft margin, di mana beberapa titik boleh melanggar margin dengan penalti.

Ketika data tidak bisa dipisahkan secara linear, digunakan Radial Basis Function untuk mentransformasikan data ke ruang berdimensi tinggi tanpa eksplisit menghitung transformasi:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{2.30}$$

Beberapa jenis Kernel pada SVM:

Tabel 2.2. Fungsi-fungsi Kernel yang Umum Digunakan dalam SVM

Kernel	Rumus
Linear	$K(x_i, x_j) = x_i \cdot x_j$
Polynomial	$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$
RBF (Radial Basis Function)	$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
Sigmoid	$K(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$

Sumber: [29]

Dengan penggunaan kernel, maka fungsi klasifikasi dapat dirumuskan pada persamaan 2.31

$$F(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i, x) - b \tag{2.31}$$

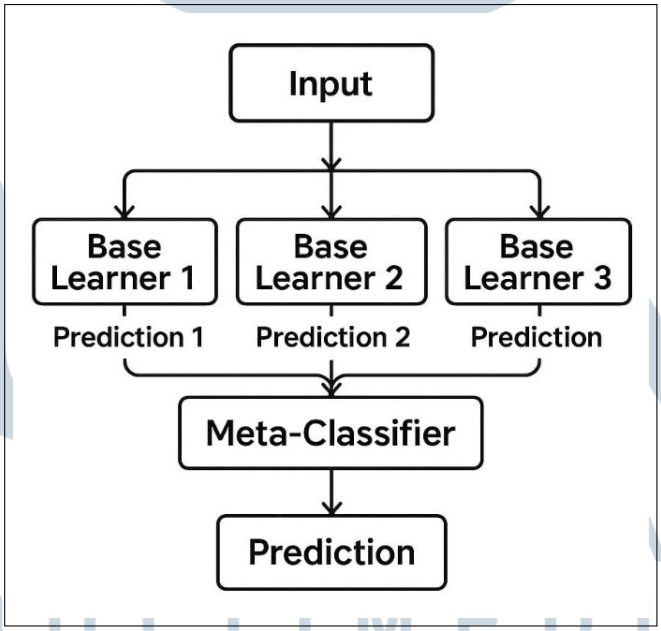
keterangan:

- $K(x_i, x)$: hasil fungsi kernel antara data pelatihan x_i dan data uji x.

- α_i: Lagrange multiplier untuk data ke-i.
- y_i: label kelas dari data ke-i.
- b: bias (intercept).
- $F(x)$: skor klasifikasi. Jika $F(x) > 0$ maka kelas $+1$, jika $F(x) < 0$ maka kelas -1.

2.9 Stacking Ensembel Learning

Stacking Ensembel Learning (Stacking Classifier) merupakan metode yang menggunakan gabungan dari berbagai jenis algoritma klasifikasi yang berbeda (contohnya: Random Forest, Logistic Regression, XGBoost, dan lainnya). Dengan menggabungkan beberapa model dasar (base classifier) yang memiliki cara kerja berbeda, stacking dapat menghasilkan model akhir yang lebih kuat dan mampu membuat prediksi yang lebih akurat. Metode ini bekerja dalam dua tahap utama, seperti yang diperlihatkan pada Gambar 2.2.



Gambar 2.2. Arsitektur Stacking Classifier

Sumber: [30]

Pertama, dataset dilatih menggunakan beberapa model awal (*base learners*). Hasil prediksi dari masing-masing model ini kemudian dikumpulkan dan digunakan

sebagai input untuk model utama (*meta classifier*). *Meta classifier* ini akan mempelajari pola dari hasil prediksi sebelumnya untuk menghasilkan keputusan akhir. Dengan mempertimbangkan kemampuan belajar dari model-model awal dan model akhir, stacking mampu meningkatkan kinerja klasifikasi secara keseluruhan secara signifikan [30]. Proses klasifikasi menggunakan Stacking Classifier dapat diinterpretasikan dengan menggunakan pseudocode Algoritma 4 berikut ini [31]:

Algorithm 4 *Stacking Ensemble Model*

procedure STACKINGMODELEVALUATION
    Input: Training Data $T_{train} = \{X_i, y_i\}$, $\forall i$ and Testing Data T_{test}
    Output: Ensemble Classifier *STACK*

    Let $base\_learners(B) = \{B_1, ..., B_n\}$
    for each B_i in B **do**
        Train B_i on T_{train}
    end for

    for each B_i **do**
        Evaluate B_i on T_{test}
    end for
    Kirimkan hasil prediksi ke meta-classifier (*STACK*) sebagai input
    Train *STACK* pada T_{train}
    Evaluate *STACK* pada T_{test}
    Return *STACK*
end procedure

2.10 Metrik Evaluasi (Evaluation Metrix)

Metrik evaluasi bertujuan untuk mengevaluasi dan membandingkan berbagai model klasifikasi / metode *machine learning*. Terdapat banyak metrik yang berguna untuk menguji kemampuan suatu model klasifikasi, beberapa diantaranya sebagai berikut:

2.10.1 Confusion Matrix

Confusion matrix merupakan tabel yang digunakan untuk mencatat dan membandingkan hasil prediksi model dengan data yang sebenarnya. Dalam tabel ini, kolom menunjukkan hasil prediksi model, sedangkan baris menunjukkan kondisi sebenarnya dari data. Tabel ini terbagi menjadi beberapa bagian utama,

yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN) [32].

Tabel 2.3. Confusion matrix

Classes	Predictive Positive	Predictive Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Sumber: [32]

2.10.2 Akurasi (Accuracy)

Rumus Akurasi adalah dengan membandingan jumlah *true positive* dan *true negative* dengan jumlah *true positive*, *true negative*, *false positive*, dan *false negative*. Persamaannya dapat dilihat pada Persamaan 2.32. Akurasi memberikan ukuran menyeluruh tentang seberapa baik model memprediksi secara benar pada seluruh kumpulan data.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.32}$$

Unit dasar dari metrik ini adalah individu tunggal dalam dataset: setiap unit memiliki bobot yang sama dan berkontribusi secara setara terhadap nilai Akurasi. Namun, terdapat kondisi dimana akan ada kelas dengan jumlah unit yang besar dan kelas lain dengan jumlah unit yang sedikit. Dalam situasi ini, kelas yang lebih besar akan memiliki bobot yang lebih besar dibandingkan kelas yang lebih kecil

2.10.3 Presisi (Precision)

Precision merupakan perbandingan dari jumlah *true positive* dengan *true positive* dan *false positive*. Persamaan *precision* dapat dilihat pada Persamaan 2.33

$$Precision = \frac{TP}{TP + FP} \tag{2.33}$$

Precision menunjukkan proporsi unit yang diprediksi sebagai positif oleh model dan memang benar-benar positif. Dengan kata lain, Precision dapat

dijadikan sebagai acuan seberapa besar suatu model dapat dipercaya ketika model memprediksi seseorang/unit sebagai Positif.

2.10.4 Recall

Recall merupakan rasio antara *true positive* dengan *true positive* dan *false negative*. Rumus 2.34 menunjukkan cara perhitungan *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.34}$$

Recall menilai seberapa baik model dalam mengidentifikasi data yang termasuk dalam kelas positif. Secara sederhana, metrik ini mencerminkan seberapa efektif model dalam menangkap semua instance positif yang terdapat dalam dataset.

2.10.5 F1-score

Rumus F1-score dapat diartikan sebagai rata-rata tertimbang antara Precision dan Recall, di mana nilai terbaik dari F1-score adalah 1 dan nilai terburuknya adalah 0. Kontribusi Precision dan Recall dianggap sama pentingnya dalam perhitungan F1-score, sehingga penggunaan rata-rata harmonik memungkinkan tercapainya keseimbangan terbaik di antara keduanya. Persamaannya dapat dilihat pada Persamaan 2.35.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.35}$$

Untuk memahami cara kerja F1-score, dapat dengan melihat bagaimana rata-rata harmonik digunakan. Rata-rata ini lebih memperhatikan keseimbangan nilai, jadi jika Precision atau Recall-nya jauh lebih kecil dari yang lain, nilai F1-score juga akan ikut turun. Karena itu, F1-score lebih menghargai model yang seimbang, yaitu yang punya Precision dan Recall yang sama baiknya.

2.10.6 ROC (Receiver Operating Characteristic) Curve

ROC (Receiver Operating Characteristic) Curve merupakan grafik yang digunakan untuk mengevaluasi kinerja metode klasifikasi diagnostik biner. Kurva ini menggambarkan hubungan antara true positive rate (sensitivitas) di sumbu y

dan false positive rate (1 - spesifisitas) di sumbu x, berdasarkan berbagai nilai ambang (cut-off) dari hasil tes. ROC bermanfaat untuk menunjukkan trade-off antara sensitivitas dan spesifisitas pada berbagai titik ambang, sehingga membantu dalam memilih nilai cut-off optimal. ROC juga berguna untuk membandingkan kinerja beberapa tes diagnostik tanpa dipengaruhi oleh prevalensi penyakit dalam populasi [33].

2.10.7 Area Under the Curve (AUC)

Area Under the Curve (AUC) merupakan ukuran kuantitatif dari luas di bawah kurva ROC yang menunjukkan kemampuan keseluruhan suatu tes diagnostik dalam membedakan antara kondisi positif dan negatif. Nilai AUC berkisar antara 0 hingga 1, di mana AUC = 1.0 menunjukkan kinerja sempurna, sedangkan AUC = 0.5 mencerminkan kinerja acak seperti melempar koin. Dalam praktik klinis, AUC $>$ 0.8 dianggap sebagai tes yang dapat diterima, dan semakin mendekati 1, semakin baik akurasi diagnostiknya. AUC sering disertai confidence interval (CI) 95% untuk menggambarkan ketidakpastian statistik dari estimasi tersebut [33].

Tabel 2.4. Interpretasi Area Under the Curve (AUC)

***Area Under the Curve* (AUC)**	**Interpretasi**
$0.9 \leq$ AUC	Sangat Baik
$0.8 \leq$ AUC < 0.9	Baik
$0.7 \leq$ AUC < 0.8	Cukup
$0.6 \leq$ AUC < 0.7	Buruk
$0.5 \leq$ AUC < 0.6	Gagal

Sumber: [33]