BAB 2
LANDASAN TEORI

2.1 Kanker Payudara

Kanker payudara merupakan salah satu jenis kanker paling umum yang menyerang wanita di seluruh dunia, ditandai dengan pertumbuhan sel abnormal secara tidak terkendali pada jaringan payudara, yang dapat berasal dari berbagai bagian seperti lobulus, duktus, atau jaringan ikat lainnya [17]. Deteksi dini sangat krusial karena meskipun tidak semua benjolan bersifat ganas, sel kanker memiliki potensi menyebar ke organ lain melalui sistem limfatik dan aliran darah, terutama ke paru-paru, hati, atau tulang [18, 19], sehingga metode pencitraan seperti mammografi banyak digunakan untuk skrining awal [20]. Dalam konteks pengobatan dan prognosis, penentuan stadium kanker menjadi aspek penting karena stadium awal cenderung memiliki tingkat kesembuhan lebih tinggi, sedangkan stadium lanjut memiliki risiko metastasis yang lebih besar [21].

Sistem klasifikasi TNM (*Tumor, Node, Metastasis*) digunakan secara luas untuk menggambarkan sejauh mana kanker telah menyebar, dengan kombinasi ukuran tumor primer (T), keterlibatan kelenjar getah bening regional (N), dan adanya metastasis jauh (M). Berdasarkan sistem ini, stadium klinis kanker payudara diklasifikasikan dari stadium 0 hingga IV, mengacu pada AJCC *Cancer Staging Manual*, 8th *Edition*, sebagaimana dijelaskan dalam Tabel 2.1. Stadium 0 menunjukkan keberadaan karsinoma in situ, belum dianggap sebagai kanker, namun memiliki potensi untuk berkembang menjadi kanker. Stadium I mengindikasikan kanker yang masih terlokalisasi. Stadium II menunjukkan kanker stadium lanjut lokal pada tahap awal, sementara stadium III menunjukkan kanker stadium lanjut lokal pada tahap akhir, yang biasanya ditandai dengan keterlibatan kelenjar getah bening. Stadium IV menunjukkan kondisi kanker metastasis, yaitu ketika sel kanker telah menyebar ke organ tubuh lain [22].

Tabel 2.1. Pengelompokan stadium kanker payudara menurut AJCC edisi ke-8

Stage	Tumor (T)	Node (N)	Metastasis (M)
0	Tis	N0	M0
IA	T1	N0	M0
IB	T0	N1mi	M0
	T1	N1mi	M0
IIA	T0	N1	M0
	T1	N1	M0
	T2	N0	M0
IIB	T2	N1	M0
	T3	N0	M0
IIIA	T0	N2	M0
	T1	N2	M0
	T2	N2	M0
	T3	N1	M0
	T3	N2	M0
IIIB	T4	N0	M0
	T4	N1	M0
	T4	N2	M0
IIIC	Any T	N3	M0
IV	Any T	Any N	M1

Sumber: [16]

2.2 Ekspresi Gen (RNA-seq)

Ekspresi gen melalui sekuensing RNA (RNA-seq) dengan teknologi sekuensing generasi berikutnya (NGS) merupakan metode untuk mempelajari transkriptom, yaitu keseluruhan molekul RNA yang ditranskripsi dari gen-gen sel dalam kondisi tertentu [23]. RNA-seq memanfaatkan teknologi NGS untuk menghasilkan jutaan fragmen RNA pendek yang disebut ”*reads*” yang kemudian dipetakan ke genom referensi untuk mengukur tingkat ekspresi gen. Pendekatan ini memberikan profil resolusi tinggi dengan menghitung jumlah *reads* yang selaras dengan setiap gen, menghasilkan ukuran digital dari tingkat ekspresinya, yang umumnya dinormalisasi dalam unit seperti *Counts*, TPM (*transcript per million*), RPKM (*reads per kilobase of transcript per million reads mapped*), dan FPKM (*fragments per kilobase of transcript per million reads mapped*) [24].

2.3 Ekspresi miRNA (stem-loop)

MicroRNA (miRNA) adalah molekul RNA non-coding yang berperan dalam regulasi gen dengan mengikat mRNA target untuk menghambat translasi atau memicu degradasi mRNA. Struktur stem-loop miRNA, yang terbentuk dari transkrip prekursor (pre-miRNA), adalah karakteristik utama yang mengindikasikan pemrosesan menjadi miRNA matang [25]. Dengan teknologi *Next-Generation Sequencing* (NGS) pada platform Illumina, ekspresi miRNA dapat dianalisis secara masif dan sensitif, menghasilkan data profil ekspresi dengan resolusi tinggi. Data miRNA digunakan sebagai acuan untuk mencari biomarker baru karena miRNA menunjukkan pola ekspresi spesifik pada berbagai penyakit, seperti kanker, dan memiliki stabilitas tinggi di jaringan serta cairan tubuh. Platform Illumina memfasilitasi deteksi miRNA dalam jumlah rendah sekalipun, menjadikannya alat untuk mengidentifikasi miRNA sebagai biomarker diagnostik atau prognostik.

2.4 *Differentially Expressed Genes* (DEG)

Perkembangan teknologi sekuensing generasi baru (Next-Generation Sequencing/NGS), khususnya RNA sequencing (RNA-seq), telah merevolusi pendekatan dalam studi ekspresi gen. Salah satu output utama dari analisis data RNA-seq adalah identifikasi *Differentially Expressed Genes* (DEG), yaitu gen-gen yang menunjukkan perbedaan tingkat ekspresi yang signifikan antar kondisi biologis, seperti antara jaringan sehat dan kanker. DEG mampu mengungkap gen-gen kunci yang berperan dalam patogenesis suatu penyakit atau respons terhadap terapi. Seiring dengan kemajuan RNA-seq, microRNA (miRNA) juga menjadi fokus dalam analisis DEG karena miRNA berperan dalam regulasi pascatranskripsi dan dapat memengaruhi ekspresi gen target secara luas. Deteksi perubahan ekspresi miRNA melalui RNA-seq dapat memberikan informasi tambahan mengenai mekanisme regulasi molekuler yang mendasari suatu kondisi patologis [26]. Untuk menganalisis DEG dari data RNA-seq, digunakan berbagai metode statistik dan bioinformatika, salah satunya adalah paket *limma* dari Bioconductor. Awalnya dikembangkan untuk data mikroarray, *limma* kini telah diperluas untuk menganalisis data RNA-seq. Keunggulan utama *limma* adalah kemampuannya menangani desain eksperimental kompleks serta memberikan estimasi statistik yang stabil, bahkan pada dataset dengan jumlah sampel kecil [27]. Dengan

memadukan pendekatan statistik yang kuat dan kemampuan menangani data RNA-seq dan miRNA, *limma* menjadi salah satu alat yang banyak digunakan dalam studi transkriptomik dan penemuan biomarker berbasis ekspresi gen. Prosedur analisis DEG menggunakan *limma* secara umum dapat direpresentasikan secara ringkas menggunakan pseudocode berikut, disusun berdasarkan fungsi-fungsi utama dari paket *limma* dalam dokumentasi Bioconductor [28].

Algorithm 1 *Differential Expression Analysis using Limma*

```
function LIMMA_DEG_ANALYSIS(expression_data, labels)
    Input:
        expression_data: matriks ekspresi gen/miRNA (fitur x sampel)
        labels: vektor label kelas ("early", "late")
    Output:
        deg_results: tabel gen/miRNA yang terekspresi secara berbeda

    Transpose expression_data agar fitur di baris dan sampel di kolom
    Buat desain model dengan model.matrix( labels)
    Jalankan lmFit() untuk memodelkan ekspresi terhadap label
    Terapkan eBayes() untuk memperbaiki estimasi variansi
    Ambil hasil DEG menggunakan topTable() dengan threshold yang sesuai
    return deg_results
end function
```

2.5 *Feature Selection*

Feature selection atau seleksi fitur merupakan tahap krusial dalam pengolahan data berdimensi tinggi, terutama pada bidang bioinformatika dan pembelajaran mesin. Pada umumnya, dataset yang digunakan memiliki jumlah fitur jauh lebih besar dibanding jumlah sampel. Ketidakseimbangan ini dapat menyebabkan model prediksi menjadi rentan terhadap *overfitting*, serta memperlambat proses pelatihan dan inferensi model [29].

Secara umum, seleksi fitur merupakan teknik praproses yang bertujuan untuk menyaring fitur-fitur penting dari kumpulan data dengan cara menghilangkan fitur yang redundan dan tidak relevan. Pendekatan ini tidak hanya bertujuan mengurangi dimensi data, tetapi juga berkontribusi terhadap peningkatan performa algoritma klasifikasi maupun regresi, seperti peningkatan akurasi, efisiensi komputasi, dan penyederhanaan struktur model [30]. Dengan mengidentifikasi himpunan fitur optimal, proses pembelajaran mesin menjadi lebih efektif dan dapat menghasilkan model yang lebih general.

2.5.1 *Analysis of Variance* (ANOVA)

Analysis of Variance (ANOVA) merupakan metode statistik yang digunakan dalam proses pemilihan fitur dengan mengukur nilai F untuk setiap fitur terhadap target [31]. Pendekatan ini bertujuan untuk mengevaluasi sejauh mana rata-rata nilai fitur berbeda antar kelompok kelas dalam data. Semakin besar nilai F yang dihasilkan, semakin besar pula indikasi bahwa fitur tersebut memiliki perbedaan yang signifikan antar kelas target [32]. Dengan demikian, ANOVA membantu dalam mengidentifikasi fitur numerik yang memiliki keterkaitan kuat dengan variabel target dan relevan untuk digunakan dalam proses klasifikasi. Proses perhitungan F-statistik dijabarkan sebagai berikut:

Langkah pertama adalah menghitung *Sum of Squares Between* (SSB) untuk mengukur variabilitas antar kelompok yang dihitung dengan rumus:

$$\text{SSB} = \sum_{i=1}^{k} n_i \left(\bar{x}_i - \bar{x}\right)^2 \tag{2.1}$$

Keterangan:

- k: jumlah kelompok atau kelas.
- n_i: jumlah sampel pada kelompok ke-i.
- $\bar{x}_i$: rata-rata nilai fitur pada kelompok ke-i.
- $\bar{x}$: rata-rata keseluruhan (*grand mean*) dari semua data.

Setelah mendapatkan nilai SSB, selanjutnya menghitung MSB yang dimana MSB adalah SSB yang dinormalkan dengan derajat kebebasan antar kelompok yang dihitung dengan rumus:

$$\text{MSB} = \frac{\text{SSB}}{k-1} \tag{2.2}$$

Dimana $k-1$ adalah derajat kebebasan untuk varians antar kelompok.

Selanjutnya menghitung *Sum of Squares Within* (SSW) untuk mengukur variabilitas dalam kelompok yang dihitung dengan rumus:

$$\text{SSW} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(x_{ij} - \bar{x}_i\right)^2 \tag{2.3}$$

Keterangan:

- x_{ij}: nilai data ke-j dalam kelompok ke-i.

Setelah mendapatkan nilai SSW, selanjutnya menghitung MSW yang dimana MSW adalah SSW yang dinormalkan dengan derajat kebebasan dalam kelompok yang dihitung dengan rumus:

$$\text{MSW} = \frac{\text{SSW}}{N-k} \tag{2.4}$$

Keterangan:

- MSW: *Mean Square Within*, yaitu varians rata-rata di dalam kelompok.
- SSW: *Sum of Squares Within*, total varians dalam kelompok.
- N: jumlah total sampel dari semua kelompok.

Setelah MSB dan MSW dihitung, F-statistik dihitung dengan rumus:

$$F = \frac{\text{MSB}}{\text{MSW}} \tag{2.5}$$

Keterangan:

- MSB (*Mean Square Between*): Varians antar kelompok, yang mengukur seberapa jauh rata-rata setiap kelompok dari rata-rata keseluruhan.
- MSW (*Mean Square Within*): Varians dalam kelompok, yang mengukur seberapa besar variasi data dalam setiap kelompok.

2.5.2 *Recursive Feature Elimination* (RFE)

Recursive Feature Elimination (RFE) adalah metode seleksi fitur berbasis pembelajaran mesin yang digunakan untuk memilih subset fitur paling relevan dari data berdimensi tinggi. Prosedur ini bekerja secara bertahap dengan cara menghapus fitur yang kontribusinya terhadap performa model dianggap paling rendah [33]. Tujuan utama dari RFE adalah memperoleh kumpulan fitur optimal yang mampu menghasilkan model prediktif yang efisien dan akurat [34]. RFE mengevaluasi fitur secara berulang berdasarkan bobot atau koefisien yang diperoleh dari model pelatih. Dalam setiap iterasi, fitur dengan pengaruh terkecil dieliminasi hingga jumlah fitur yang ditentukan tercapai. Teknik ini sangat berguna dalam data yang memiliki dimensi sangat besar dan jumlah sampel terbatas. Proses seleksi

fitur menggunakan RFE secara umum dapat direpresentasikan dengan pseudocode berikut [34]:

Algorithm 2 *Recursive Feature Elimination* (RFE)

```
function FEATURE_SELECTION_RFE(X, y, n_features)
    Input:
        X: DataFrame berisi fitur-fitur
        y: Series berisi label kelas
        n_features: Jumlah fitur yang ingin dipilih
    Output:
        X_new: DataFrame dengan fitur terpilih
        selected_features: Daftar nama fitur terpilih

    Bangun model pembelajaran mesin
    Inisialisasi RFE dengan model dan jumlah fitur target
    for setiap iterasi hingga jumlah fitur tersisa sama dengan n_features do
        Latih model untuk menentukan kepentingan fitur
        Hapus satu fitur dengan kepentingan terendah
        Perbarui daftar fitur yang tersisa
    end for
    Simpan fitur terpilih ke selected_features dari daftar fitur yang tersisa
    Buat X_new dengan fitur terpilih
    return X_new, selected_features
end function
```

2.6 *Logistic Regression* dengan L2 *Ridge Regularization*

Logistic Regression merupakan algoritma klasifikasi yang digunakan untuk memprediksi probabilitas suatu data termasuk ke dalam salah satu dari dua kelas [35]. Model ini bekerja dengan mencari hubungan linear antara fitur masukan dan variabel target menggunakan fungsi linear seperti pada regresi linier, kemudian menerapkan fungsi aktivasi *sigmoid* untuk mengubah nilai tersebut menjadi probabilitas dalam rentang antara 0 dan 1 [36]. Proses ini dimulai dengan membentuk fungsi linear sebagai berikut:

$$z = \mathbf{w}^T \mathbf{x} + b \tag{2.6}$$

Keterangan:

- z: nilai kombinasi linear antara fitur dan parameter.

- $\mathbf{w}$: vektor bobot (*weights*) yang akan dioptimalkan.
- $\mathbf{x}$: vektor fitur input.
- b: bias (*intercept*).

Nilai z kemudian dipetakan ke dalam probabilitas menggunakan fungsi aktivasi *sigmoid*:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.7}$$

Keterangan:

- $\sigma(z)$: probabilitas bahwa instance termasuk kelas positif ($y = 1$).
- e: bilangan eksponensial.

Untuk mengukur kesalahan prediksi, digunakan fungsi *log loss* (juga dikenal sebagai *cross-entropy loss*), yang dalam konteks regularisasi L2 (*Ridge*) dimodifikasi dengan menambahkan penalti terhadap nilai parameter yang besar:

$$J(\mathbf{w},b) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\sigma(z_i)) + (1-y_i)\log(1-\sigma(z_i))] + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.8}$$

Keterangan:

- N: jumlah total data.
- λ: parameter regularisasi.
- $\|\mathbf{w}\|^2$: norma L2 dari bobot (tidak termasuk bias).

Regularisasi L2 membantu mencegah *overfitting* dengan mengurangi kompleksitas model melalui penalti terhadap bobot yang besar.

Untuk meminimalkan fungsi loss $J(\mathbf{w},b)$, digunakan algoritma *gradient descent* dengan turunan parsial terhadap parameter:

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{1}{N}\sum_{i=1}^{N}(\sigma(z_i) - y_i)\mathbf{x}_i + \lambda \mathbf{w} \tag{2.9}$$

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(\sigma(z_i) - y_i) \tag{2.10}$$

Keterangan:

- $(\sigma(z_i) - y_i)$: selisih antara prediksi dan label sebenarnya.
- $\lambda \mathbf{w}$: turunan dari komponen regularisasi L2.

Pembaruan parameter dilakukan secara iteratif berdasarkan nilai gradien:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial J}{\partial \mathbf{w}} \tag{2.11}$$

$$b = b - \eta \frac{\partial J}{\partial b} \tag{2.12}$$

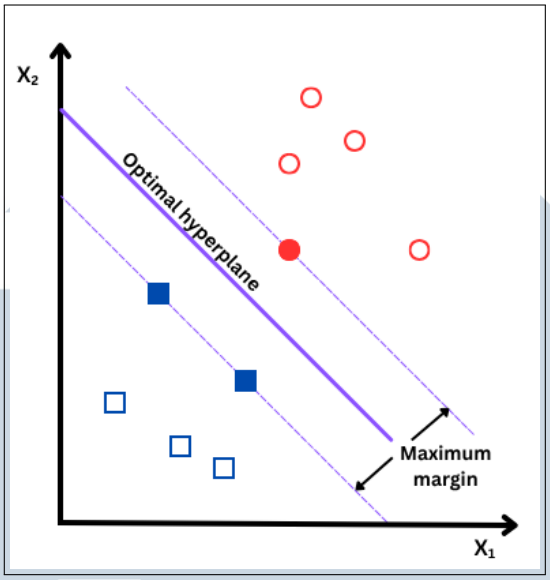
Keterangan:

- η: *learning rate* atau laju pembelajaran.

Proses ini diulang hingga nilai fungsi loss mencapai konvergensi atau hingga jumlah iterasi maksimum tercapai.

2.7 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) merupakan metode *supervised learning* yang digunakan untuk menyelesaikan permasalahan klasifikasi maupun regresi, baik yang bersifat linear maupun non-linear [37]. Konsep dasar SVM bertumpu pada teori margin dan *hyperplane*, dengan tujuan utama mencari *hyperplane* terbaik yang mampu memisahkan dua kelas data secara optimal [38]. *Hyperplane* ini menjadi batas pemisah yang ideal, yang ditentukan berdasarkan margin maksimum, yaitu jarak antara *hyperplane* dan titik data terdekat dari masing-masing kelas, yang disebut sebagai *support vectors*. Dalam kasus data non-linear, SVM menggunakan pendekatan *kernel* untuk memetakan data ke ruang berdimensi lebih tinggi, di mana pemisahan linear menjadi dilakukan [37]. *Hyperplane* optimal berada di tengah margin yang memisahkan dua kelas secara maksimal, terlihat pada Gambar 2.1.



Gambar 2.1. Ilustrasi optimal *hyperplane* dengan margin maksimum

Sumber: [39]

Hyperplane merupakan batas keputusan dalam ruang berdimensi d yang memisahkan dua kelas. Untuk kasus data linear, *hyperplane* dirumuskan sebagai:

$$\mathbf{w}^\top \mathbf{x} + b = 0 \tag{2.13}$$

Keterangan:

- $\mathbf{w}$ adalah vektor bobot,
- $\mathbf{x}$ adalah vektor fitur input,
- b adalah bias (intersep).

Untuk memenuhi syarat klasifikasi yang benar, setiap data harus berada pada sisi yang sesuai dari *hyperplane*, yaitu:

$$\mathbf{w}^\top \mathbf{x}_i + b \geq 1, \quad \text{jika } y_i = +1 \tag{2.14}$$

$$\mathbf{w}^\top \mathbf{x}_i + b \leq -1, \quad \text{jika } y_i = -1 \tag{2.15}$$

Keterangan:

- y_i adalah label kelas dari data ke-i, dengan nilai $+1$ untuk satu kelas dan -1 untuk kelas lainnya.

Support vector adalah titik-titik data yang berada paling dekat dengan *hyperplane* dan memiliki peran penting dalam menentukan posisi *hyperplane* optimal. Titik-titik ini memenuhi:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1 \tag{2.16}$$

Data dengan jarak minimum ke *hyperplane*, dan hanya titik-titik ini yang berkontribusi terhadap pembentukan *hyperplane*. Titik dengan $\alpha_i > 0$ dari hasil optimasi adalah *support vector*.

Margin adalah jarak antara *hyperplane* dan titik data *support vector* terdekat. SVM bertujuan untuk memaksimalkan margin ini, yang dirumuskan sebagai:

$$\text{Margin} = \frac{2}{||\mathbf{w}||} \tag{2.17}$$

Keterangan:

- $||\mathbf{w}||$ adalah norm (panjang) dari vektor bobot $\mathbf{w}$.
- Margin kecil cenderung menyebabkan overfitting karena model terlalu kompleks dan sensitif terhadap noise.
- Margin besar meningkatkan kemampuan generalisasi model ke data yang belum dilihat.

Untuk kasus data yang secara linear dapat dipisahkan, digunakan Linear SVM. Tujuannya adalah memaksimalkan margin, yang setara dengan meminimalkan norm dari vektor bobot:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||^2 \tag{2.18}$$

$$\text{dengan syarat:} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i \tag{2.19}$$

Fungsi objektif ini meminimalkan kompleksitas model sambil memastikan semua data berada di sisi yang benar dari margin.

Untuk menyelesaikan optimasi tersebut, dibentuklah fungsi *Lagrangian* sebagai:

$$\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right] \tag{2.20}$$

Lagrangian ini digunakan untuk menggabungkan fungsi objektif dengan constraint melalui penggunaan multiplier α_i.
Keterangan:

- $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha})$ adalah fungsi Lagrangian yang akan dioptimalkan.
- $||\mathbf{w}||^2$ adalah kuadrat dari norma vektor bobot yang ingin diminimalkan untuk memaksimalkan margin.
- α_i adalah *Lagrange multiplier* untuk data ke-i, digunakan untuk menggabungkan constraint ke dalam fungsi objektif.
- $\mathbf{x}_i$ adalah vektor fitur dari data ke-i.
- n adalah jumlah total data latih.

Dengan menurunkan fungsi Lagrangian terhadap $\mathbf{w}$ dan b, diperoleh kondisi optimal:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{2.21}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.22}$$

Hasil ini kemudian disubstitusikan kembali ke fungsi *Lagrangian* untuk membentuk fungsi dual, sehingga optimasi menjadi:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{2.23}$$

$$\text{dengan syarat: } \sum i = 1^n \alpha_i y_i = 0, \quad \alpha_i \geq 0 \tag{2.24}$$

Keterangan:

- y_i, y_j adalah label kelas dari data ke-i dan ke-j, bernilai $+1$ atau -1,
- $\mathbf{x}_i^\top \mathbf{x}_j$ adalah hasil perkalian dalam (dot product) antara dua vektor fitur.

- Syarat $\sum_{i=1}^{n} \alpha_i y_i = 0$ memastikan keseimbangan kelas.
- Syarat $\alpha_i \geq 0$ menunjukkan bahwa multipliers harus bernilai non-negatif.

Jika data tidak sepenuhnya dapat dipisahkan secara linear (*non-separable*), maka digunakan pendekatan *Soft Margin*. Pendekatan ini memperbolehkan beberapa data melanggar margin menggunakan *slack variable* ξ_i:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i \tag{2.25}$$

$$\text{dengan syarat:} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{2.26}$$

Keterangan:

- ξ_i adalah variabel slack yang mengukur pelanggaran terhadap margin oleh data ke-i.
- C adalah parameter regulasi yang mengontrol kompromi antara margin maksimum dan jumlah kesalahan klasifikasi.

Sebaliknya, *Hard Margin* digunakan ketika data benar-benar dapat dipisahkan tanpa kesalahan, dengan syarat ketat:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i \tag{2.27}$$

Namun, pendekatan ini tidak cocok jika data mengandung outlier atau noise karena tidak ada toleransi terhadap pelanggaran margin.

Untuk data yang tidak dapat dipisahkan secara linear, digunakan fungsi *kernel* untuk memetakan data ke ruang fitur berdimensi lebih tinggi:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \tag{2.28}$$

Keterangan:

- $K(\mathbf{x}_i, \mathbf{x}_j)$ adalah fungsi kernel yang mengukur kesamaan antara dua data $\mathbf{x}_i$ dan $\mathbf{x}_j$.
- $\phi(\cdot)$ adalah fungsi transformasi non-linear yang memetakan data ke ruang fitur berdimensi lebih tinggi.

- Fungsi kernel memfasilitasi perhitungan dot product di ruang fitur tinggi tanpa eksplisit menghitung $\phi(\mathbf{x})$ *kernel trick*.

  Linear Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j \tag{2.29}$$

Keterangan:

- Kernel linear digunakan ketika data masih dapat dipisahkan secara linear di ruang asli.
- Kernel ini identik dengan tidak melakukan transformasi, yaitu $\phi(\mathbf{x}) = \mathbf{x}$.

  RBF Kernel (*Radial Basis Function*):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2) \tag{2.30}$$

Keterangan:

- γ adalah parameter skala yang mengontrol seberapa jauh pengaruh satu titik data menjangkau. Nilai besar dari γ menyebabkan model lebih fokus pada titik-titik yang sangat dekat, sementara nilai kecil membuat model lebih toleran terhadap data yang jauh.
- Kernel RBF sangat efektif dalam menangani data nonlinear karena memproyeksikan ke ruang fitur tak hingga dimensi secara implisit.

2.8 *Confusion Matrix*

Confusion matrix adalah tabel yang digunakan untuk mengevaluasi performa model klasifikasi biner berdasarkan hasil prediksi dan label aktual [40]. Tabel ini terdiri dari empat komponen utama, yaitu *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN), sebagaimana ditunjukkan pada Tabel 2.2.

Tabel 2.2. Struktur *Confusion Matrix* untuk Klasifikasi Biner

Aktual / Prediksi	**Positif (1)**	**Negatif (0)**
Positif (1)	True Positive (TP)	False Negative (FN)
Negatif (0)	False Positive (FP)	True Negative (TN)

Sumber: [40]

Penjelasan dari masing-masing elemen pada Tabel 2.2 adalah sebagai berikut:

- ***True Positive* (TP)**: Jumlah data aktual yang termasuk dalam kelas positif dan berhasil diprediksi sebagai positif oleh model.
- ***False Negative* (FN)**: Jumlah data aktual yang termasuk dalam kelas positif namun keliru diprediksi sebagai negatif.
- ***False Positive* (FP)**: Jumlah data aktual yang termasuk dalam kelas negatif namun keliru diprediksi sebagai positif.
- ***True Negative* (TN)**: Jumlah data aktual yang termasuk dalam kelas negatif dan berhasil diprediksi sebagai negatif oleh model.

Berdasarkan nilai-nilai yang terdapat pada *Confusion Matrix*, beberapa metrik evaluasi dapat dihitung. Penjelasan dan rumus dari masing-masing metrik dijabarkan sebagai berikut:

- ***Accuracy*** merupakan proporsi dari jumlah prediksi yang benar terhadap keseluruhan jumlah data. Metrik ini memberikan gambaran umum mengenai seberapa sering model melakukan prediksi yang tepat. Rumus perhitungannya disajikan pada Persamaan (2.31).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.31}$$

- ***Precision*** mengukur ketepatan prediksi terhadap kelas positif, yakni proporsi data yang benar-benar positif dari seluruh data yang diprediksi sebagai positif. Rumus perhitungannya ditunjukkan pada Persamaan (2.32).

$$Precision = \frac{TP}{TP + FP} \tag{2.32}$$

- ***Recall*** atau *Sensitivity* menunjukkan kemampuan model dalam mengidentifikasi data yang benar-benar termasuk dalam kelas positif. Metrik ini sangat relevan dalam konteks di mana kegagalan mendeteksi

kelas positif memiliki konsekuensi besar. Rumusnya dapat dilihat pada Persamaan (2.33).

$$Recall = \frac{TP}{TP + FN} \tag{2.33}$$

- ***Specificity*** mengukur proporsi data dari kelas negatif yang berhasil diklasifikasikan dengan benar sebagai negatif. Metrik ini digunakan untuk menilai sejauh mana model menghindari kesalahan dalam mendeteksi kelas negatif. Rumus perhitungannya ditunjukkan pada Persamaan (2.34).

$$Specificity = \frac{TN}{TN + FP} \tag{2.34}$$

- **F1-Score** merupakan rata-rata harmonis dari nilai *precision* dan *recall*. Metrik ini memberikan penilaian yang seimbang terhadap ketepatan dan keberhasilan deteksi kelas positif, terutama pada data yang tidak seimbang. Ditunjukkan pada Persamaan (2.35).

$$\text{F1-Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.35}$$

- **ROC *Curve* (*Receiver Operating Characteristic*)** merupakan representasi visual dari kinerja model klasifikasi biner pada berbagai ambang batas keputusan. Kurva ini menunjukkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR). ROC curve sangat berguna terutama dalam konteks data yang tidak seimbang [41]. Ditunjukkan pada Persamaan (2.36).

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN} \tag{2.36}$$

- **AUC (*Area Under the Curve*)** menunjukkan sejauh mana model mampu membedakan antara kelas positif dan negatif. Nilai AUC berada dalam rentang 0 hingga 1, dengan nilai yang lebih tinggi menunjukkan performa klasifikasi yang lebih baik [42]. Secara umum, nilai AUC dapat diinterpretasikan sebagaimana ditunjukkan pada Tabel 2.3.

Tabel 2.3. Interpretasi Nilai AUC dalam Studi Klinis

Nilai AUC	**Interpretasi**
$0.9 \leq \text{AUC} \leq 1.0$	Sangat baik
$0.8 \leq \text{AUC} < 0.9$	Cukup baik
$0.7 \leq \text{AUC} < 0.8$	Cukup
$0.6 \leq \text{AUC} < 0.7$	Kurang
$0.5 \leq \text{AUC} < 0.6$	Gagal

Sumber: [42]