BAB 2
LANDASAN TEORI

2.1 Kanker Prostat

Kanker prostat merupakan pertumbuhan sel yang tidak terkontrol di kelenjar prostat, yang umumnya dipicu oleh kombinasi berbagai macam faktor, bukan disebabkan oleh satu faktor tunggal saja. Organ prostat adalah organ yang sebagian besar terdiri dari jaringan kelenjar yang menghasilkan cairan penyusun sekitar 30–35% dari semen. Kanker prostat biasanya dimulai dari mutasi pada sel kelenjar akibat peradangan kronis yang terjadi pada sel basal di bagian luar prostat. Sel-sel tersebut akan terus berkembang secara tidak terkendali dan membentuk benjolan yang dapat terlihat secara kasat mata [18]. Menentukan stadium kanker prostat, digunakan sistem klasifikasi TNM (Tumor, Node, Metastasis) yang dikeluarkan oleh *American Joint Committee on Cancer* (AJCC) dan *Union Internationale Contre le Cancer* (UICC) pada tahun 2010. Sistem ini mengintegrasikan skor gleason sebagai metode penilaian yang dianjurkan, serta mempertimbangkan faktor-faktor prognostik seperti nilai Skor Gleason dan kadar antigen spesifik prostat (PSA) sebelum operasi dalam proses pengelompokan stadium [19]. Tabel 2.1 menunjukkan penilaian yang digunakan dalam sistem TNM untuk menentukan stadium kanker prostat.

Tabel 2.1. Pengelompokan stadium prostat AJCC dan UICC [19]

Stage	T	N	M	PSA (ng/ml)	Gleason score
I	T1a–c	N0	M0	<10	≤ 6
	T2a	N0	M0	<10	≤ 6
	T1–2a	N0	M0	X	X
IIA	T1a–c	N0	M0	≤ 20	7
	T1a–c	N0	M0	≥ 10 and <20	≤ 6
	T2a	N0	M0	<20	7
	T2b	N0	M0	≤ 20	≤ 7
	T2b	N0	M0	X	X
IIB	T2c	N0	M0	Any	Any
	T1–2	N0	M0	≥ 20	Any
	T1–2	N0	M0	Any	≥ 8
III	T3a–b	N0	M0	Any	Any
IV	T4	N0	M0	Any	Any
	Any	N1	M0	Any	Any

2.2 Ekspresi Gen RNA Sequence

Ekspresi gen merujuk pada proses di mana sel mengaktifkan atau menonaktifkan gen tertentu untuk mengatur fungsi biologis. Proses ini melibatkan transkripsi informasi genetik dari DNA menjadi RNA, yang kemudian dapat diterjemahkan menjadi protein [20]. Ekspresi gen telah menjadi alat yang sangat penting dalam mengidentifikasi biomarker diagnostik dan prognostik pada kanker [21]. Salah satu metode terkini yang digunakan untuk memprofilkan ekspresi RNA adalah RNA sequencing (RNA-seq), yang berbasis pada teknologi Next-Generation Sequencing (NGS) dan telah menggantikan penggunaan microarray dalam studi ekspresi gen [22].

RNA sequence memungkinkan pengukuran dan perbandingan pola ekspresi gen dengan resolusi yang sangat tinggi. Dalam proses ini, jutaan fragmen pendek RNA, yang dapat disebut sebagai *reads*, diambil secara acak dari input RNA. Hasilnya kemudian dipetakan secara komputasional ke referensi genom untuk membentuk peta transkripsi, yang mencerminkan tingkat ekspresi setiap gen berdasarkan jumlah reads yang sejajar dengannya [23].

2.3 MikroRNA

MikroRNA (miRNA) adalah molekul RNA non-koding pendek yang berperan dalam mengatur ekspresi gen dengan menghambat translasi atau mendegradasi mRNA [24]. miRNA berfungsi dalam berbagai proses seluler, seperti pembelahan, diferensiasi, kematian sel, dan metabolisme [25]. miRNA memiliki potensi besar sebagai biomarker dalam penelitian kanker prostat karena dapat diekstraksi dari berbagai sampel biologis, stabil, dan tahan terhadap berbagai kondisi penyimpanan, serta memiliki kemampuan untuk terdeteksi dalam cairan tubuh tanpa memerlukan biopsi invasif, bahkan telah ditemukan dalam urin dan semen [26]. Mengingat pentingnya fungsi miRNA, analisis ekspresi miRNA menjadi fokus dalam riset dasar maupun klinis, terutama pada berbagai jenis kanker. Sebagian besar studi masih mengandalkan teknologi mikroarray dan PCR real-time kuantitatif; namun, *Next-Generation Sequencing* (NGS) mulai menggantikan metode konvensional tersebut karena menawarkan sensitivitas yang lebih tinggi, rentang deteksi yang lebih luas, dan akurasi yang lebih baik dalam melakukan analisis ekspresi [27].

2.4 Analisis DEG

Analisis *Differentially Expressed Gene* (DEG) adalah teknik yang umum digunakan dalam biologi molekuler untuk mengevaluasi perbedaan tingkat ekspresi gen antara dua atau lebih kelompok sampel yang berbeda. Tujuan dari analisis ini adalah untuk mengidentifikasi gen-gen yang mengalami perubahan ekspresi secara signifikan pada kondisi yang dibandingkan. Informasi ini digunakan untuk memahami peran gen dalam berbagai proses biologis, jalannya suatu penyakit, serta respons sel terhadap perlakuan tertentu. Gen-gen yang teridentifikasi melalui analisis ini juga dapat dijadikan kandidat biomarker yang berpotensi digunakan dalam diagnosis, penentuan prognosis, maupun evaluasi efektivitas suatu terapi atau pengobatan [28]. Analisis DEG umumnya dilakukan dengan bantuan alat statistik seperti DESeq2 dan limma, yang digunakan untuk menganalisis data ekspresi gen dari teknologi RNA-Seq dan mikroarray [29].

DESeq2 memodelkan data ekspresi gen sebagai distribusi negatif binomial, lalu menghitung perbedaan ekspresi antar kondisi menggunakan *log2 fold change* yang distabilkan dengan pendekatan *empirical Bayes* dan diuji secara statistik menggunakan *Wald test* [30]. Sementara itu, limma mengasumsikan data mengikuti distribusi normal, menggunakan model linear dan penyusutan variansi antar gen dengan pendekatan *empirical Bayes*, lalu menguji signifikansi menggunakan *moderated t-test* [31]. Keduanya menghasilkan daftar gen yang terekspresi secara signifikan, yang dapat dimanfaatkan dalam studi biologis lanjutan atau sebagai kandidat biomarker.

2.5 Seleksi Fitur

Seleksi fitur adalah proses penting dalam machine learning untuk mengurangi dimensi data dengan cara menghapus fitur yang tidak relevan atau redundan. Tujuannya adalah meningkatkan akurasi model, efisiensi komputasi, dan interpretabilitas tanpa menghilangkan informasi penting dalam data [32]. Metode seleksi fitur umumnya dibagi menjadi tiga kategori: filter, wrapper, dan embedded [33]. Metode filter memilih fitur berdasarkan karakteristik statistik data secara independen dari model. Metode wrapper menggunakan performa model untuk mengevaluasi subset fitur, namun memerlukan komputasi tinggi. Sementara itu, metode embedded melakukan seleksi selama pelatihan model, menggabungkan keunggulan filter dan wrapper secara efisien.

2.5.1 Analysis of Variance (ANOVA)

ANOVA adalah metode statistik yang digunakan untuk mengevaluasi dan membandingkan rata-rata dari dua atau lebih kelompok atau kondisi yang diuji. Teknik ini memberikan dasar bagi peneliti untuk menyimpulkan apakah perbedaan yang diamati di antara kelompok-kelompok tersebut signifikan secara statistik. Metode ini sangat berguna dalam studi eksperimental maupun observasional yang bertujuan untuk menilai efek suatu faktor terhadap berbagai kelompok, menggunakan hasil statistik F, yang merupakan rasio antara varians antar kelompok dan varians dalam kelompok [34]. Pada penerapan pemilihan fitur, ANOVA menggunakan statistik F untuk mengukur sejauh mana setiap fitur berhubungan signifikan dengan variabel target, sehingga memungkinkan identifikasi dan prioritisasi fitur-fitur yang paling relevan. Rumus untuk statistik F dapat diuraikan pada Persamaan 2.1.

$$F = \frac{\text{MSB}}{\text{MSW}} \tag{2.1}$$

MSB (*Mean Square Between-groups*) merupakan rata-rata kuadrat varians antar kelompok, yang dapat dihitung dengan Persamaan 2.2

$$MSB = \frac{\text{SSB}}{\text{DFB}} \tag{2.2}$$

SSB (*sum of squares between groups*) mengukur variasi rata-rata antar kelompok dibandingkan dengan rata-rata keseluruhan yang dapat dihitung dengan Persamaan 2.3

$$SSB = \sum_{i=1}^{k} n_i(\bar{X}_i - \bar{X})^2 \tag{2.3}$$

DFB *Degrees of Freedom Between* mengacu pada nilai bebas yang digunakan untuk menghitung varians antar kelompok yang dapat dihitung dengan Persamaan 2.4.

$$\text{DFB} = k - 1 \tag{2.4}$$

MSW (*Mean Square Within-groups*) menilai varians rata-rata di dalam masing-masing kelompok yang dihitung dengan Persamaan 2.5.

$$\text{MSW} = \frac{SSW}{DFW} \tag{2.5}$$

SSW (*Sum of Squares Within-groups*) dihitung berdasarkan variasi data dalam kelompok terhadap rata-rata kelompoknya, sebagaimana ditunjukkan pada Persamaan 2.6.

$$\text{SSW} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2 \tag{2.6}$$

DFW (Degrees of Freedom Within-groups) merepresentasikan derajat kebebasan di dalam kelompok yang dihitung dengan Persamaan 2.7.

$$\text{DFW} = N - k \tag{2.7}$$

X_{ij} adalah nilai observasi ke-j pada kelompok ke-i, $\bar{X}_i$ adalah rata-rata kelompok ke-i, dan $\bar{X}$ adalah rata-rata keseluruhan. Variabel n_i menyatakan jumlah observasi dalam kelompok ke-i, k adalah jumlah total kelompok, dan N merupakan total seluruh observasi [35].

2.5.2 Minimum Redundancy Maximum Relevance (MRMR)

MRMR merupakan metode *filter* yang memilih fitur berdasarkan relevansi terhadap label dan redundansi antar fitur. Fitur yang terpilih akan menunjukkan keseimbangan yang baik antara relevansi maksimum dan redundansi minimum [36]. Relevansi maksimum bertujuan menemukan fitur dengan korelasi tinggi terhadap label. Menghitung relevansi maksimum, digunakan nilai rata-rata *mutual information* (MI) dari setiap fitur X_i terhadap label C. Rumus MI antara variabel X dan Y dapat dijabarkan dengan Persamaan 2.8.

$$\text{MI}(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right) \tag{2.8}$$

Nilai relevansi maksimum V_i dari fitur X_i dalam himpunan S terhadap label C dapat dihitung dengan Persamaan 2.9

$$V_i(S) = \frac{1}{|S|} \sum_{X_i \in S} \text{MI}(X_i, C) \tag{2.9}$$

Untuk menghindari ketergantungan antar fitur, relevansi maksimum perlu diimbangi dengan redundansi minimum. Jika hanya relevansi yang diperhitungkan, ada kemungkinan besar ketergantungan antar fitur akan meningkat. Oleh karena itu, redundansi minimum harus diterapkan tanpa mengurangi relevansi yang tersedia.

Nilai redundansi minimum W_i dihitung dengan MI antara fitur X_i dengan fitur-fitur lainnya X_j dalam himpunan T, yang dapat dihitung dengan Persamaan 2.10.

$$W_i(T) = \frac{1}{|T|} \sum_{X_j \in T} \mathrm{MI}(X_i, X_j) \tag{2.10}$$

Kriteria metode mRMR untuk pemilihan fitur adalah dengan memaksimalkan relevansi dan meminimalkan redundansi, yang dihitung menggunakan Persamaan 2.11

$$\max_i \left(V_i(S) - W_i(T)\right) \tag{2.11}$$

V_i adalah nilai relevansi maksimum antara fitur X_i dan variabel label C, dengan S merupakan himpunan fitur-fitur kandidat yang mengandung X_i. Sementara itu, W_i adalah nilai redundansi rata-rata antara X_i dan fitur-fitur lainnya X_j dalam himpunan T [37].

2.5.3 Recursive Feature Elimination (RFE)

Metode RFE merupakan strategi seleksi fitur berbasis *wrapper* yang mengeliminasi fitur secara bertahap melalui proses rekursif. Pseudo kode dari RFE dapat dilihat pada Algoritma 1 [38].

Algorithm 1 Recursive Feature Elimination (RFE)

1: **Input:** Dataset T dengan fitur $F = \{f_1, f_2, ..., f_n\}$, target y, model M, dan jumlah fitur yang diinginkan k
2: **Output:** Subset fitur terpilih F'
3: Inisialisasi $F' \leftarrow F$
4: **while** $|F'| > k$ **do**
5: Latih model M menggunakan fitur F'
6: Hitung skor kepentingan tiap fitur $f \in F'$
7: Hapus fitur dengan kepentingan terendah dari F'
8: **end while**
9: **return** F'

Pada setiap iterasi, model dilatih menggunakan seluruh fitur yang tersisa, lalu fitur-fitur tersebut diberi peringkat berdasarkan tingkat kepentingannya terhadap performa model. Fitur yang memiliki kontribusi paling rendah kemudian dihapus. Proses ini diulangi sampai jumlah fitur yang diinginkan tercapai atau tidak ada lagi fitur yang perlu dihilangkan. Karena kontribusi fitur bergantung

pada kombinasi yang digunakan, pendekatan rekursif memastikan hanya fitur paling relevan yang dipertahankan [38].

2.5.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah algoritma *supervised learning* yang digunakan untuk tugas klasifikasi dan regresi [39]. Konsep dasar SVM adalah membangun sebuah *hyperplane* sebagai batas pemisah untuk mengklasifikasikan data ke dalam dua kelompok. Posisi *hyperplane* ini ditentukan oleh titik data yang disebut sebagai *support vectors*, yaitu data yang paling dekat dengan *hyperplane*. Titik-titik ini membentuk *margin*, yaitu jarak antara *hyperplane* dan data dari masing-masing kelas [40]. Dalam kasus klasifikasi dua kelas, *hyperplane* ditentukan berdasarkan teori Vapnik [41, 42]. Misalkan data fitur berupa $\mathbf{x}_i \in \mathbb{R}^N$ untuk $i = 1, 2, \ldots, m$, dan kelas $y_i \in \{1, -1\}$. Persamaan yang menggambarkan dua *hyperplane* yang memisahkan kedua kelas dituliskan pada Persamaan 2.12.

$$\begin{aligned} \mathbf{w}^T \mathbf{x_i} + b = 1 \quad \text{jika } y = 1 \\ \mathbf{w}^T \mathbf{x_i} + b = -1 \quad \text{jika } y = -1 \end{aligned} \tag{2.12}$$

Formulasi mencari *hyperplane* yang memenuhi kondisi pemisahan untuk Persamaan 2.12 dapat dirumuskan dalam bentuk Pertidaksamaan 2.13.

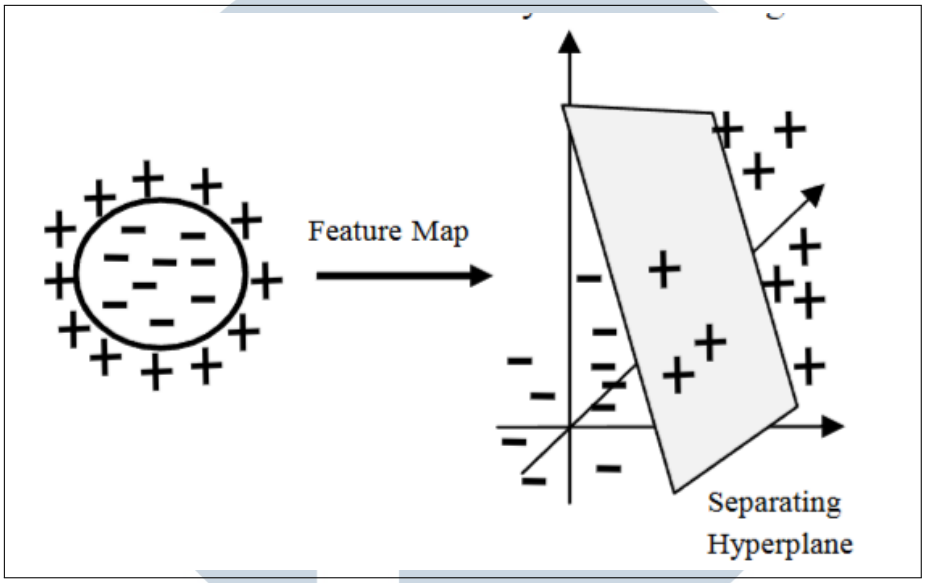
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i = 1, 2, \ldots, m \tag{2.13}$$

$\mathbf{w}$ adalah vektor bobot yang menentukan arah *hyperplane*, dan b adalah bias yang menggeser posisi *hyperplane* terhadap titik asal. Pendekatan ini dikenal sebagai pendekatan *hard margin* yang mengasumsikan bahwa data dapat dipisahkan secara linear tanpa kesalahan klasifikasi [43]. Namun, data dalam dunia nyata sering kali tidak dapat dipisahkan secara sempurna akibat adanya *noise* dan *outlier*. Pendekatan *soft margin* mengatasi masalah tersebut dengan memperbolehkan adanya kesalahan klasifikasi melalui variabel slack ξ_i, yang menggambarkan tingkat kesalahan klasifikasi pada setiap sampel [44]. Variabel *slack* ini memodifikasi pertidaksamaan pada Persamaan 2.13, sebagaimana ditunjukkan dalam Persamaan 2.14.

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1, 2, \ldots, m \tag{2.14}$$

Tujuan utama dari SVM adalah mencari parameter $\mathbf{w}$ dan b yang memaksimalkan margin antara dua kelas, yang dikenal sebagai *maximum-margin hyperplane*.

Maximum-margin hyperplane merupakan garis pemisah yang memaksimalkan jarak terhadap *support vectors* dari masing-masing kelas [45]. Ilustrasi konsep ini ditunjukkan pada Gambar 2.1.

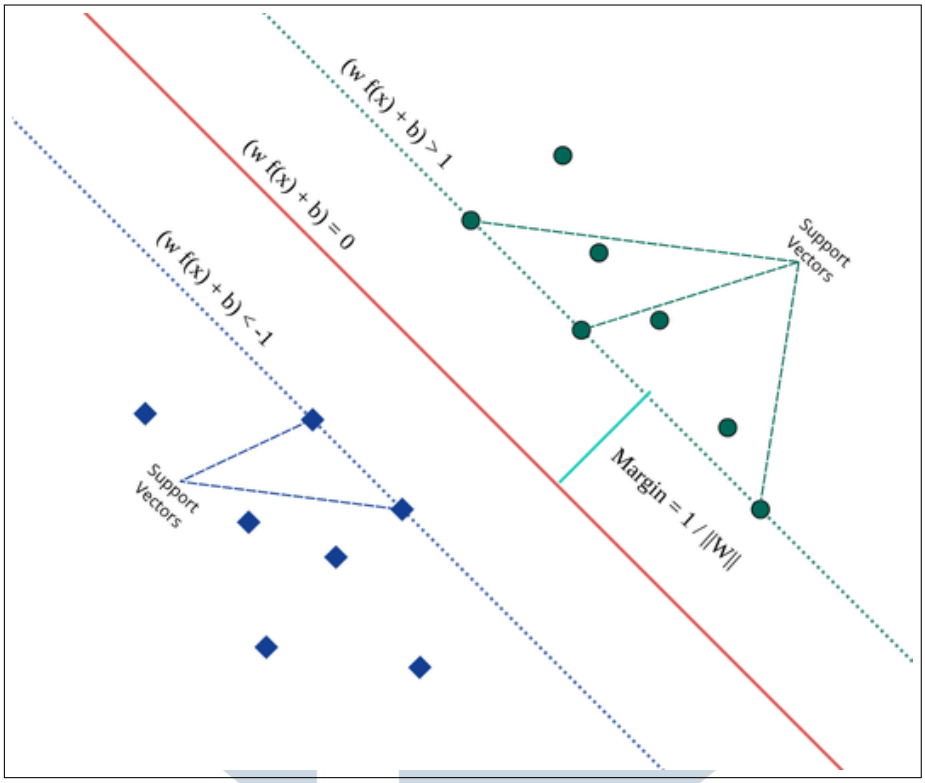


Gambar 2.1. Ilustrasi pemisahan kelas oleh *hyperplane* pada SVM [45]

Untuk memperoleh *maximum-margin hyperplane*, SVM perlu menghitung nilai margin antara dua kelas. Secara matematis, margin tersebut dapat dinyatakan melalui Persamaan 2.15.

$$\text{Margin} = \frac{1}{\|\mathbf{w}\|} \tag{2.15}$$

Support vector adalah titik data yang terletak tepat pada batas margin, yaitu titik yang apabila dimasukkan ke dalam Persamaan 2.13 menghasilkan nilai 1. Titik-titik ini memiliki peran penting dalam menentukan posisi dan orientasi vektor **w**, serta mempengaruhi ukuran margin yang dibentuk oleh *hyperplane*. Visualisasi SVM dalam mencari nilai margin dapat dilihat pada Gambar 2.2.



Gambar 2.2. Ilustrasi pencarian margin pada SVM [44]

SVM bertujuan untuk memaksimalkan margin antara dua kelas. Permasalahan optimisasi ini disebut sebagai masalah *primal* pada SVM diformulasikan dalam dua aturan utama: pertama, meminimalkan $|\mathbf{w}||^2$ untuk memperbesar margin, sebagaimana ditunjukkan pada Persamaan 2.16.

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.16}$$

Konstanta $\frac{1}{2}$ digunakan untuk konvensi matematis. Aturan kedua, memastikan seluruh data terklasifikasi dengan benar, sesuai fungsi pertidaksamaan pada Persamaan 2.13. Masalah optimisasi *primal* dapat diselesaikan secara optimal dengan mengubahnya ke dalam bentuk *dual* menggunakan metode Lagrange [46]. Metode ini memperkenalkan variabel Lagrange α_i yang membuat fungsi objektif

SVM dituliskan di Persamaan 2.17.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{dengan syarat} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad \text{untuk } i = 1, \ldots, m \end{aligned} \tag{2.17}$$

Persamaan 2.17 ini bertujuan meningkatkan nilai dari variabel Lagrange α_i untuk setiap data pelatihan. Variabel α_i menggambarkan bobot atau kontribusi setiap data terhadap margin optimal, sementara y_i adalah label kelas dari data. Vektor $\mathbf{x}_i$ dan $\mathbf{x}_y$ merepresentasikan fitur data, dan C adalah parameter regulasi yang mengontrol keseimbangan antara margin yang lebih besar dan kesalahan klasifikasi.

Setelah memperoleh nilai Lagrange α_i yang optimal, parameter model seperti bobot $\mathbf{w}$ dan bias b dapat dihitung. Kedua parameter ini digunakan dalam fungsi keputusan SVM, seperti dituliskan pada Persamaan 2.18.

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \right) \tag{2.18}$$

Nilai bias b diperoleh dari data yang memenuhi kondisi $0 < \alpha_i < C$, sebagaimana ditunjukkan pada Persamaan 2.19

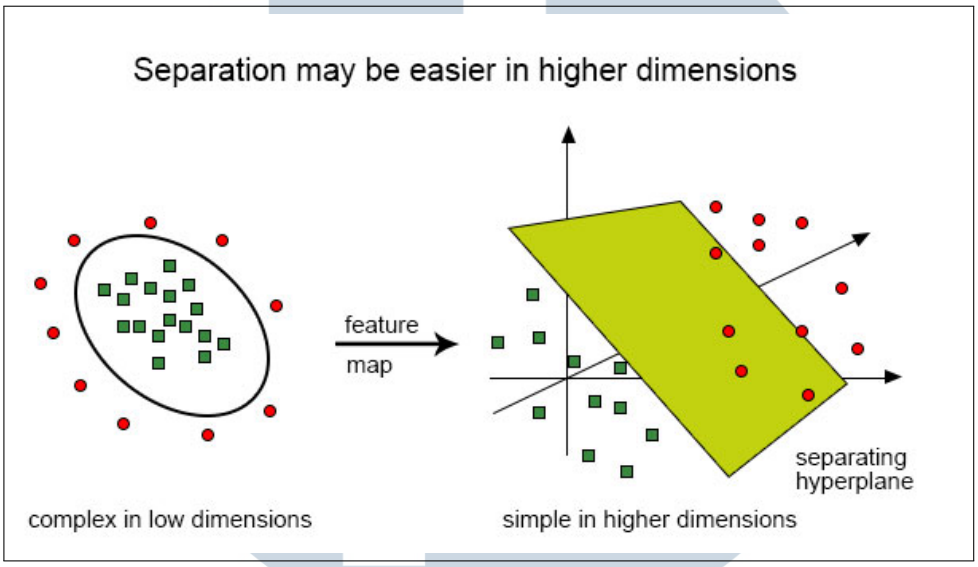
$$b = y_k - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_k \tag{2.19}$$

Fungsi keputusan pada Persamaan 2.18 berlaku hanya untuk data yang dapat dipisahkan secara linear. Namun untuk data yang tidak linear, SVM menggunakan *kernel trick* yang memetakan data ke ruang dimensi yang lebih tinggi tanpa melakukan transformasi secara eksplisit [47]. Salah satu kelebihan dari formulasi *dual* adalah kemampuannya untuk mengintegrasikan fungsi kernel secara langsung. Fungsi kernel didefinisikan sebagai berikut dalam Persamaan 2.20.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \tag{2.20}$$

Fungsi kernel menggantikan operasi *dot product* dalam formulasi SVM untuk mengukur kesamaan antara dua titik data $\mathbf{x}_i$ dan $\mathbf{x}_j$ yang telah dipetakan ke ruang

fitur berdimensi lebih tinggi melalui fungsi $\phi(\mathbf{x})$. Teknik ini dikenal sebagai *kernel trick*, yang memungkinkan SVM melakukan klasifikasi *non-linear* tanpa perlu menghitung transformasi secara eksplisit. Gambar 2.3 menunjukkan bagaimana data yang tidak terpisahkan secara *linear* menjadi separable setelah pemetaan.



Gambar 2.3. Ilustrasi *kernel trick* untuk transformasi ke ruang fitur berdimensi lebih tinggi [48].

Fungsi keputusan SVM menggunakan *kernel trick* dituliskan pada Persamaan 2.21.

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{2.21}$$

Beberapa fungsi kernel yang sering diterapkan dalam metode SVM antara lain meliputi:

- Kernel Linear:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j \tag{2.22}$$

  Kernel ini digunakan untuk data yang dapat dipisahkan secara linier dan umumnya menjadi pilihan standar dalam implementasi SVM karena sifatnya yang sederhana dan efisien.

- Kernel Polinomial:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d \tag{2.23}$$

  Kernel ini memetakan data ke ruang fitur berdimensi polinomial, dengan c sebagai konstanta dan d sebagai derajat yang menentukan tingkat

kelengkungan *hyperplane*.

- Kernel *Radial Basis Function* (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \tag{2.24}$$

Kernel ini mengukur kesamaan antara dua titik berdasarkan jarak *euclidean*. Parameter σ mengatur sejauh mana data pelatihan terdekat mempengaruhi keputusan klasifikasi. Nilai σ yang kecil membuat keputusan lebih sensitif terhadap titik data terdekat.

2.5.5 L2-penalized Logistic Regression

Algoritma regresi logistik semakin banyak diterapkan dalam berbagai bidang, khususnya dalam kedokteran dan ilmu pengetahuan alam [49]. Fungsi utama dari regresi logistik adalah untuk memprediksi probabilitas bahwa suatu data termasuk ke dalam salah satu dari dua kelas dalam masalah klasifikasi biner berdasarkan hubungan antara fitur-fitur input dan masing-masing kelas. Proses ini dilakukan dengan memetakan kombinasi linier dari fitur input ke dalam nilai probabilitas antara 0 dan 1 menggunakan fungsi sigmoid [50, 51]. Misalkan data fitur berupa $\mathbf{x}_i \in \mathbb{R}^N$ untuk $i = 1, 2, \ldots, m$, dan kelas $y_i \in \{1, -1\}$. Fungsi linier yang digunakan dalam regresi logistik dituliskan pada Persamaan 2.25.

$$z = \mathbf{w}^\top \mathbf{x}_i + b \tag{2.25}$$

$\mathbf{w}$ adalah vektor bobot dan b adalah bias. Nilai z kemudian dimasukkan ke dalam fungsi sigmoid untuk menghitung probabilitas bahwa data $\mathbf{x}_i$ termasuk dalam kelas positif. Fungsi sigmoid yang dimaksud diberikan pada Persamaan 2.26.

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x}_i + b)}} \tag{2.26}$$

Probabilitas dari data $\mathbf{x}_i$ termasuk ke dalam kelas positif $y_i = 1$ dan negatif

$y_i = 0$ dirangkum melalui Persamaan 2.27.

$$\begin{aligned} P(y_i = 1 \mid \mathbf{x}_i) &= \sigma(z) \\ P(y_i = -1 \mid \mathbf{x}_i) &= 1 - \sigma(z) \end{aligned} \tag{2.27}$$

Fungsi kerugian atau *log loss* digunakan untuk mengevaluasi seberapa baik model mengklasifikasikan data masukan sesuai dengan label yang sebenarnya. Dalam regresi logistik, fungsi kerugian yang umum dipakai adalah *log loss* (*binary cross-entropy*), yang dapat dilihat pada rumus di Persamaan 2.28.

$$\mathscr{L}(\mathbf{w}, b) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{2.28}$$

Fungsi kerugian ini menghitung perbedaan antara probabilitas yang diprediksi $\hat{y}_i$ dengan label yang sebenarnya y_i untuk setiap sampel. Jika model memberikan prediksi yang sangat dekat dengan nilai label, maka nilai kerugian akan kecil dan sebaliknya.

Untuk menghindari *overfitting* selama pelatihan, regresi logistik sering dilengkapi dengan regularisasi L2 (juga dikenal sebagai *Ridge regularization*) [52]. Pendekatan ini menambahkan penalti terhadap besar nilai bobot model ke dalam fungsi kerugian, sehingga fungsi kerugian dengan regularisasi L2 dituliskan pada Persamaan 2.29:

$$\mathscr{L}(\mathbf{w}, b) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\mathbf{w}\|_2^2 \tag{2.29}$$

λ sebagai hiperparameter regulasi yang mengontrol besarnya penalti, dan $\|\mathbf{w}\|_2^2$ merupakan norma L2 dari bobot. Regularisasi ini mendorong nilai bobot tetap kecil, sehingga meningkatkan generalisasi model terhadap data yang tidak terlihat.

Tujuan regresi logistik adalah meminimalkan fungsi kerugian dengan regularisasi L2, yang dapat dicapai menggunakan fungsi optimisasi *gradient descent* untuk menemukan nilai optimal dari parameter model $\mathbf{w}$ dan b [53]. Rumus *gradient descent* dapat dirumuskan melalui Persamaan 2.30 dan 2.31

$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathscr{L}(\mathbf{w}, b) \tag{2.30}$$

$$b := b - \eta \nabla_b \mathscr{L}(\mathbf{w}, b) \tag{2.31}$$

Learning rate (η) mengontrol besar langkah pembaruan pada setiap iterasi *gradient descent*. Gradien fungsi kerugian terhadap $\mathbf{w}$ dan b, masing-masing ditulis sebagai $\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w},b)$ dan $\nabla_b\mathscr{L}(\mathbf{w},b)$ yang ditunjukkan pada Persamaan 2.32 dan 2.33.

$$\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)\mathbf{x}^{(i)} + \frac{\lambda}{m}\mathbf{w} \tag{2.32}$$

$$\nabla_b\mathscr{L}(\mathbf{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right) \tag{2.33}$$

$\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w},b)$ dan dan $\nabla_{\mathbf{b}}\mathscr{L}(\mathbf{w},b)$ dihitung dengan menurunkan fungsi kerugian dari Persamaan 2.28 terhadap $\mathbf{w}$ dan b. Kedua gradien ini memberikan arah dan besar perubahan yang harus diterapkan pada $\mathbf{w}$ dan b untuk meminimalkan fungsi kerugian dalam proses pelatihan.

2.5.6 Model Evaluation

Evaluasi model digunakan untuk mengukur sejauh mana performa model yang dikembangkan dalam menghasilkan prediksi yang akurat serta menilai kemampuannya dalam melakukan generalisasi terhadap data di dunia nyata [54]. Metrik yang umum digunakan dalam klasifikasi meliputi akurasi, presisi, recall, AUC-ROC, dan F1-score. Seluruh metrik ini dapat dihitung dari *confusion matrix*, yang merepresentasikan hasil prediksi model terhadap data aktual dalam empat kategori, seperti ditunjukkan pada Tabel 2.2 [55].

Tabel 2.2. *Confusion matrix* untuk klasifikasi biner

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

True Positive (TP) adalah jumlah prediksi positif yang benar, di mana baik prediksi model maupun label data aktual keduanya positif. *False Positive* (FP) menunjukkan jumlah kasus di mana model memprediksi positif, namun data aktual negatif. *False Negative* (FN) mencatat jumlah prediksi negatif yang salah, di mana data aktual sebenarnya positif. *True Negative* (TN) adalah jumlah prediksi negatif yang benar, di mana model dan data aktual keduanya negatif.

Metrik akurasi mengukur rasio antara jumlah prediksi yang benar dengan total prediksi yang dilakukan oleh model. Metrik ini memberikan gambaran umum tentang seberapa sering model membuat prediksi yang benar. Rumus untuk menghitung akurasi dapat dituliskan pada Persamaan 2.34.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.34}$$

Metrik presisi mengukur rasio antara jumlah prediksi positif yang benar dengan total prediksi positif yang dilakukan oleh model. Metrik ini memberikan gambaran tentang seberapa akurat prediksi positif yang dihasilkan oleh model. Rumus untuk menghitung presisi dapat dituliskan pada Persamaan 2.35.

$$\text{Presisi} = \frac{TP}{TP+FP} \tag{2.35}$$

Metrik *recall* mengukur rasio antara jumlah prediksi positif yang benar dengan jumlah data aktual yang positif. Metrik ini memberikan gambaran tentang seberapa baik model dalam mendeteksi data positif yang sebenarnya. Rumus untuk menghitung *recall* dapat dituliskan pada Persamaan 2.36.

$$Recall = \frac{TP}{TP+FN} \tag{2.36}$$

Metrik F1-score adalah rata-rata harmonik antara presisi dan recall. F1-score memberikan gambaran tentang keseimbangan antara keduanya, di mana nilai F1-score yang tinggi menunjukkan bahwa model memiliki performa yang baik dalam hal presisi maupun recall. Rumus untuk menghitung F1-score dapat dituliskan pada Persamaan 2.37.

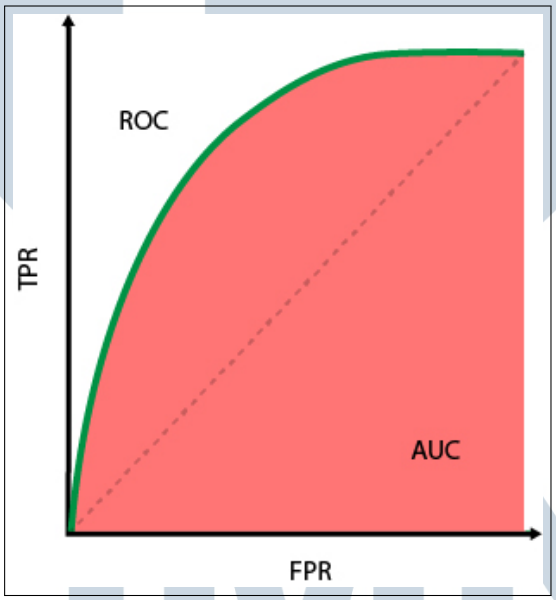
$$\text{F1-score} = 2 \times \frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}} \tag{2.37}$$

Metrik *AUC-ROC* (*Area Under the Receiver Operating Curve*) memvisualisasikan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) pada berbagai ambang keputusan. Rumus untuk menghitung TPR dan

FPR dituliskan pada Persamaan 2.38.

$$\text{TPR} = \frac{TP}{TP+FN}$$
$$\text{FPR} = \frac{FP}{FP+TN} \tag{2.38}$$

TPR, atau *recall*, mengukur proporsi sampel positif yang berhasil diklasifikasikan dengan benar, sedangkan FPR menunjukkan proporsi sampel negatif yang salah diklasifikasikan sebagai positif. Kurva ROC-AUC menunjukkan hubungan antara kedua metrik ini dan dapat dilihat pada Gambar 2.4.



Gambar 2.4. kurva ROC-AUC pada berbagai nilai ambang keputusan [56]

Nilai AUC (Area Under the Curve) dihitung sebagai luas area di bawah kurva tersebut dan merepresentasikan sejauh mana model mampu membedakan antara kelas positif dan negatif secara keseluruhan [57]. Rumus AUC ditulis pada Persamaan 2.39.

$$AUC = \sum_{i=1}^{n-1} \frac{(FPR_i - FPR_{i-1}) \cdot (TPR_i + TPR_{i-1})}{2} \tag{2.39}$$

AUC diperoleh dengan menjumlahkan hasil perkalian perubahan FPR dan rata-rata TPR antara dua ambang keputusan berturut-turut. FPR_i dan FPR_{i-1} merujuk pada

nilai FPR pada ambang keputusan ke-i dan ke-$i-1$, sedangkan TPR_i dan TPR_{i-1} merujuk pada nilai TPR pada ambang keputusan ke-i dan ke-$i-1$.