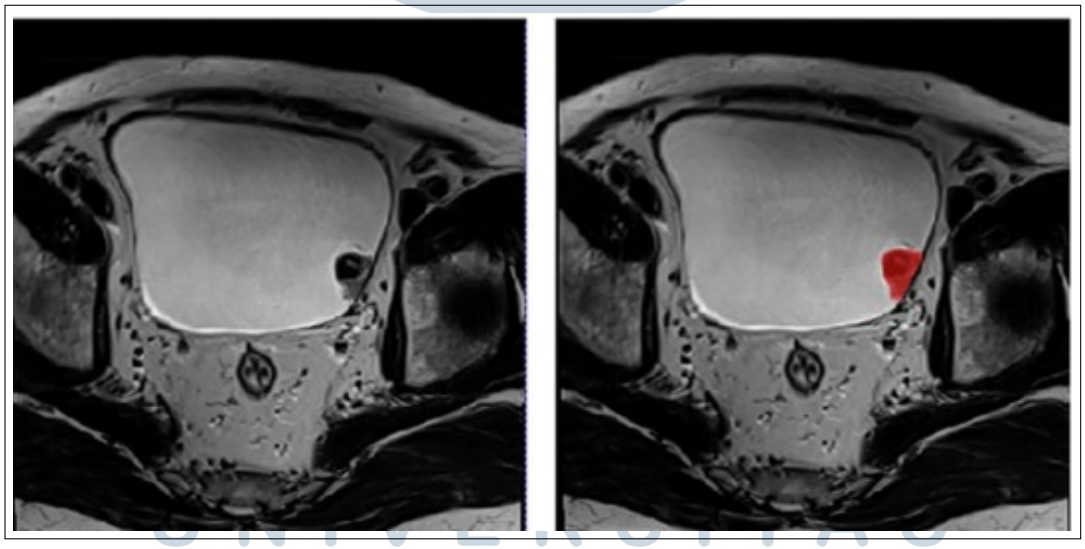
BAB 2
LANDASAN TEORI

2.1 Kanker Kandung Kemih

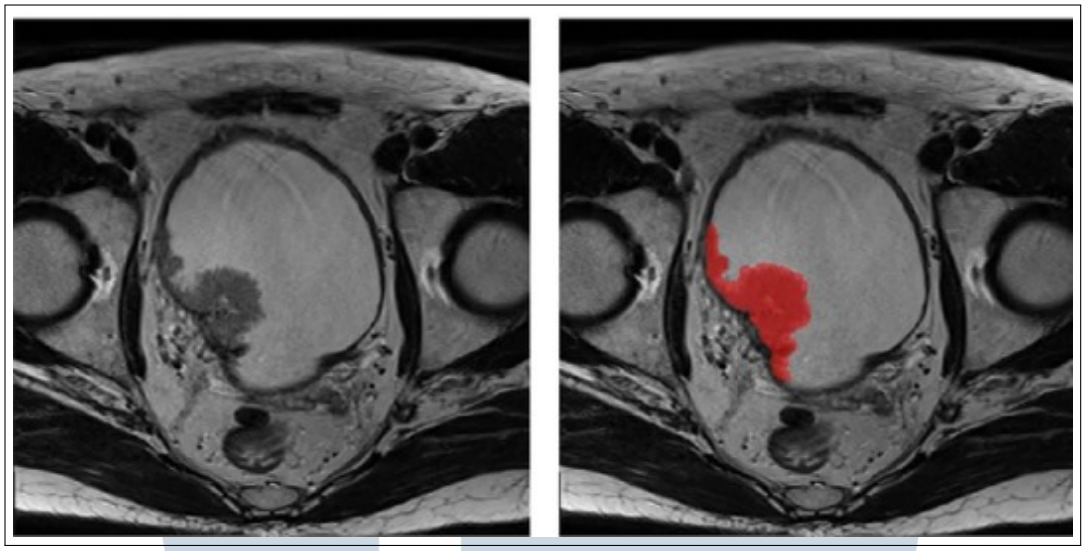
Kanker kandung kemih adalah keganasan yang bermula dari sel-sel di lapisan dalam kandung kemih [13]. Penentuan stadium sangat penting karena menggambarkan sejauh mana kanker telah menyebar, yang menjadi dasar utama dalam perencanaan terapi [5],[6]. Secara medis, dokter membaginya menjadi dua kelompok besar berdasarkan kedalaman akarnya:

1. *NMIBC (Non-Muscle Invasive)* Ini adalah jenis yang akarnya dangkal. Sel kanker hanya tumbuh di lapisan kulit dalam atau lapisan penopang di bawahnya, tetapi belum menyentuh daging atau otot utama dinding kandung kemih. Jenis ini lebih sering ditemukan namun mudah kambuh [5]. Visualisasi contoh citra MRI pasien dengan NMIBC beserta segmentasinya dapat dilihat pada Gambar 2.1



Gambar 2.1. Contoh citra MRI dan segmentasi tumor kasus NMIBC (diadaptasi dari [14])

2. *MIBC (Muscle Invasive)*: Ini adalah jenis yang akarnya dalam. Sel kanker sudah menembus masuk lapisan otot yang tebal. Jenis ini jauh lebih berbahaya karena pembuluh darah di otot sangat banyak, sehingga sel kanker mudah menyebar (*metastasis*) ke organ lain [6]. Visualisasi contoh citra MRI pasien dengan MIBC beserta segmentasinya dapat dilihat pada Gambar 2.2



Gambar 2.2. Contoh citra MRI dan segmentasi tumor kasus MIBC (diadaptasi dari [14])

2.2 Rekayasa Fitur (*Feature Engineering*)

Rekayasa fitur adalah proses dalam pembelajaran mesin yang bertujuan untuk mengubah data mentah menjadi fitur-fitur informatif yang dapat merepresentasikan masalah dengan lebih baik kepada model prediktif. Dalam konteks citra medis, data mentahnya adalah piksel gambar, yang perlu diubah menjadi fitur kuantitatif sebelum dapat digunakan oleh algoritma klasifikasi. Proses ini terdiri dari dua tahap utama: ekstraksi fitur dan seleksi fitur [15].

2.3 Ekstraksi Fitur (*Radiomics*)

Radiomics adalah bidang ilmu yang berfokus pada ekstraksi sejumlah besar fitur kuantitatif dari citra medis, seperti *CT Scan* dan *Magnetic Resonance Imaging* (MRI) [16]. Fitur-fitur ini, yang sering kali tidak terlihat oleh mata manusia, dapat menangkap karakteristik tumor seperti bentuk, ukuran, dan heterogenitas internal. Proses ini dimulai dengan segmentasi *Region of Interest* (*ROI*), yaitu area tumor yang menjadi fokus analisis. Setelah *ROI* ditentukan, citra dan area tumor diproses menggunakan perangkat lunak *radiomics* untuk menghitung berbagai karakteristik kuantitatif.

Secara garis besar, proses transformasi citra MRI menjadi fitur *radiomics* terdiri atas beberapa tahapan. Tahap pertama adalah akuisisi citra MRI, yang dilanjutkan dengan segmentasi tumor untuk memperoleh *ROI*. Selanjutnya dilakukan tahap prapemrosesan citra, seperti normalisasi intensitas dan *resampling*

voxel, untuk meningkatkan konsistensi karakteristik citra. Setelah itu, perangkat lunak *radiomics*, seperti PyRadiomics, menghitung berbagai fitur kuantitatif berdasarkan nilai intensitas dan hubungan spasial antar *voxel* di dalam *ROI*. Hasil akhir dari proses tersebut berupa sekumpulan nilai numerik yang merepresentasikan karakteristik biologis tumor dan dapat digunakan sebagai masukan bagi algoritma *machine learning* untuk proses klasifikasi.

Setelah *ROI* ditentukan, fitur-fitur *radiomics* diekstraksi dan dikelompokkan ke dalam beberapa kelas utama sesuai standar dari *Image Biomarker Standardisation Initiative* (*IBSI*) [16]. Kelompok fitur tersebut meliputi fitur bentuk (*Shape*), statistik orde pertama (*First Order Statistics*), serta fitur tekstur yang dihitung menggunakan matriks seperti *Gray Level Co-occurrence Matrix* (GLCM), *Gray Level Run Length Matrix* (GLRLM), *Gray Level Size Zone Matrix* (GLSZM), *Gray Level Dependence Matrix* (GLDM), dan *Neighbouring Gray Tone Difference Matrix* (NGTDM). Selain itu, fitur juga dapat diekstraksi dari citra hasil transformasi, seperti *Wavelet* dan *Laplacian of Gaussian* (LoG), untuk menangkap karakteristik tumor pada berbagai skala dan frekuensi [16], [17].

- Fitur Bentuk (*Shape-based Features*)
  Fitur ini mengkuantifikasi properti geometris dari *ROI* dan hanya bergantung pada hasil segmentasi, bukan pada intensitas piksel di dalamnya. Fitur ini menggambarkan ukuran dan kompleksitas bentuk tumor.

- Fitur Orde Pertama (*First-Order Statistics*)
  Fitur ini menggambarkan distribusi intensitas *voxel* di dalam ROI berdasarkan histogram, tanpa mempertimbangkan hubungan spasial antar *voxel*.

- *Gray Level Co-occurrence Matrix (GLCM)*
  textitGLCM adalah matriks yang mengukur hubungan spasial antara pasangan *voxel* dengan intensitas tertentu pada jarak dan arah yang telah ditentukan. Matriks ini menjadi dasar untuk fitur tekstur yang menggambarkan homogenitas dan kontras lokal. Beberapa fitur yang diekstraksi dari *GLCM* antara lain *Contrast*, *Correlation*, *Energy*, dan *Homogeneity* [16].

- *Gray Level Run Length Matrix (GLRLM)*
  textitGLRLM mengkuantifikasi deretan *voxel* berurutan yang memiliki tingkat keabuan yang sama dalam arah tertentu. Fitur ini berguna untuk

mengidentifikasi tekstur kasar atau halus. Matriks ini menghitung jumlah *run* (deretan) dengan panjang tertentu untuk setiap tingkat keabuan [16].

- *Gray Level Size Zone Matrix (GLSZM)*
  GLSZM mengkuantifikasi zona-zona *voxel* yang terhubung dan memiliki tingkat keabuan yang sama. Sebuah zona adalah area 3D di mana semua *voxel* memiliki intensitas yang sama dan saling terhubung. Fitur ini menggambarkan homogenitas regional dan tidak bergantung pada arah [16].

- *Gray Level Dependence Matrix (GLDM)*
  textitGLDM mengkuantifikasi dependensi tingkat keabuan di dalam sebuah lingkungan. Sebuah *voxel* dianggap dependen pada *voxel* pusat jika perbedaan intensitasnya berada di bawah ambang batas tertentu. Matriks ini menghitung jumlah dependensi untuk setiap tingkat keabuan [16].

- *Neighbouring Gray Tone Difference Matrix (NGTDM)*
  textitNGTDM mengukur perbedaan antara intensitas sebuah *voxel* dengan rata-rata intensitas tetangganya. Matriks ini menghasilkan fitur seperti *Coarseness*, *Contrast*, dan *Busyness*, yang menggambarkan seberapa cepat intensitas berubah dalam sebuah lingkungan lokal [16].

Selain diekstraksi dari citra asli (*original image*), fitur radiomics pada penelitian ini juga dihitung dari beberapa transformasi citra untuk menangkap karakteristik tumor pada perspektif spasial dan frekuensi yang berbeda. Pendekatan ini dikenal sebagai ekstraksi fitur multi-skala dan direkomendasikan dalam kerangka kerja *radiomics* modern karena mampu meningkatkan sensitivitas terhadap pola heterogenitas jaringan yang kompleks.

Transformasi citra yang digunakan meliputi:

- Transformasi *Wavelet*
  Fitur *wavelet* memanfaatkan transformasi *wavelet*, yaitu suatu metode matematis yang mendekomposisi citra ke dalam komponen frekuensi pada berbagai skala dan orientasi. Secara matematis, transformasi *wavelet* pada citra volumetrik MRI dapat dinyatakan sebagai:

$$I_{\text{wavelet}} = W_{3D}(I_{MRI}(x,y,z)) \tag{2.1}$$

di mana $I_{MRI}(x,y,z)$ merupakan citra MRI tiga dimensi, W_{3D} adalah operator transformasi *wavelet* tiga dimensi, dan I_{wavelet} adalah hasil dekomposisi citra ke dalam beberapa sub-band frekuensi.

Proses dekomposisi ini menghasilkan representasi yang lebih komprehensif terhadap struktur internal citra, yang umumnya terdiri dari komponen frekuensi rendah dan tinggi seperti *LLL*, *LLH*, *HHL*, dan *HHH*. Fitur yang diturunkan dari analisis *wavelet*, seperti energi dan entropi *wavelet*, menyediakan perspektif multiresolusi terhadap data *radiomics*. Dengan demikian, informasi tidak hanya diperoleh pada resolusi asli citra, tetapi juga pada berbagai tingkat detail, sehingga memungkinkan pemahaman yang lebih mendalam terhadap pola tekstur dan karakteristik struktural yang mendasarinya. Integrasi fitur *wavelet* dalam analisis *radiomics* membantu meningkatkan sensitivitas dalam mendeteksi serta mengarakterisasi perubahan struktural yang halus, yang berpotensi berkaitan dengan kondisi patologis atau variasi fisiologis tertentu, sehingga dapat memperkuat nilai diagnostik dan prognostik pencitraan medis [18].

Selain itu, transformasi *wavelet* telah banyak digunakan dalam berbagai penelitian *radiomics* sebelumnya dan terbukti mampu meningkatkan kemampuan diskriminatif fitur tekstur. Beberapa studi melaporkan bahwa fitur berbasis *wavelet* memberikan performa klasifikasi yang sebanding bahkan lebih tinggi dibandingkan fitur dari citra asli (*original image*) [19], [20]. Hal ini menunjukkan bahwa integrasi transformasi *wavelet* tidak hanya memperkaya representasi fitur, tetapi juga berkontribusi terhadap peningkatan stabilitas dan akurasi model prediktif.

- *Laplacian of Gaussian (LoG)* atau *log-sigma*
  Filter *Laplacian of Gaussian* (LoG) merupakan metode filtrasi berbasis konvolusi yang menggabungkan fungsi Gaussian dan operator *Laplacian*. Secara matematis, fungsi Gaussian dua dimensi dinyatakan sebagai:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.2}$$

  Filter LoG kemudian didefinisikan sebagai penerapan operator Laplacian terhadap hasil konvolusi antara kernel Gaussian dan citra MRI, yaitu:

$$LoG(x,y) = \nabla^2 \left[G(x,y,\sigma) * I_{MRI}(x,y) \right] \tag{2.3}$$

di mana $I_{MRI}(x,y)$ merupakan intensitas citra MRI, ∇^2 adalah operator Laplacian, dan $*$ menyatakan operasi konvolusi.

Secara komputasional, filter ini diterapkan dengan menggeser kernel (matriks numerik) secara sistematis ke seluruh citra melalui proses konvolusi, sehingga menghasilkan citra terfilter yang menonjolkan karakteristik spasial tertentu. Tahap Gaussian berfungsi untuk melakukan penghalusan awal, sedangkan operator *Laplacian* menekankan perubahan intensitas lokal dengan memperkuat area yang mengalami perbedaan sinyal yang signifikan, seperti batas lesi atau daerah dengan peningkatan kontras [17].

Parameter σ mengatur tingkat penghalusan sebelum proses deteksi perubahan intensitas dilakukan, sehingga secara langsung menentukan skala struktur yang disorot. Nilai σ yang kecil mempertahankan detail halus, sedangkan nilai yang lebih besar menekankan pola struktural yang lebih luas. Melalui mekanisme ini, filter LoG menghasilkan citra yang dapat memperjelas dan melokalisasi karakteristik penting pada area tertentu sebelum proses ekstraksi fitur *radiomics* dilakukan.

Dalam konteks *radiomics*, filter LoG telah digunakan sebagai bagian dari tahap prapemrosesan sebelum ekstraksi fitur dilakukan [21]. Hal ini menunjukkan bahwa transformasi LoG dapat menghasilkan representasi tekstur yang stabil untuk analisis kuantitatif lebih lanjut.

- Transformasi Intensitas *Square*
  Transformasi *square* merupakan operasi non-linear berbasis *voxel-wise* yang mengkuadratkan setiap nilai intensitas pada citra. Secara matematis, transformasi ini dapat dinyatakan sebagai:

$$I_{\text{square}}(x,y,z) = \left(I_{MRI}(x,y,z)\right)^2 \tag{2.4}$$

di mana $I_{MRI}(x,y,z)$ merupakan intensitas citra MRI tiga dimensi, dan $I_{\text{square}}(x,y,z)$ adalah hasil transformasi kuadrat dari setiap *voxel*.

Transformasi ini memperbesar perbedaan nilai intensitas dengan memberikan peningkatan yang lebih signifikan pada voxel dengan intensitas tinggi dibandingkan intensitas rendah. Akibatnya, distribusi intensitas menjadi

lebih terentang dan area dengan sinyal kuat semakin dominan. Dengan memodifikasi dinamika rentang intensitas tersebut, transformasi ini dapat meningkatkan kontras lokal serta menonjolkan karakteristik tekstur yang sebelumnya kurang terlihat pada citra asli. Dalam konteks *radiomics*, pendekatan ini berpotensi memperkaya representasi fitur dengan menekankan variasi intensitas yang berkaitan dengan heterogenitas jaringan tumor [21].

Melalui kombinasi citra asli dan berbagai transformasi tersebut, fitur *radiomics* tidak hanya merepresentasikan karakteristik morfologi dan statistik intensitas, tetapi juga variasi tekstur multi-skala yang mencerminkan kompleksitas biologis tumor secara lebih komprehensif.

2.4 Seleksi Fitur

Setelah proses ekstraksi, seringkali dihasilkan ratusan atau ribuan fitur. Banyak dari fitur ini mungkin tidak relevan atau redundan, yang dapat menurunkan performa model. Seleksi fitur adalah proses untuk memilih sub-set fitur yang paling informatif dan relevan untuk tugas prediksi. Dalam penelitian ini, digunakan algoritma *Boruta*.

Boruta adalah algoritma seleksi fitur yang bekerja berdasarkan prinsip semua relevan (*all-relevant*), yang bertujuan untuk menemukan semua fitur yang memiliki hubungan prediktif dengan variabel target [22]. Cara kerjanya secara konseptual adalah sebagai berikut:

1. Pembuatan Fitur Bayangan (*Shadow Features*): *Boruta* membuat salinan dari seluruh dataset fitur, lalu mengacak nilai di setiap kolom salinan tersebut. Fitur-fitur acak ini disebut fitur bayangan (*shadow features* dan berfungsi sebagai referensi acak.

2. Pelatihan Model: *Dataset* asli digabungkan dengan fitur bayangan, dan sebuah model *classifier* (*Random Forest*) dilatih pada *dataset* gabungan ini untuk menghitung *feature importance* dari semua fitur (asli dan bayangan).

3. Perbandingan dan Keputusan: Kepentingan setiap fitur asli dibandingkan dengan kepentingan fitur bayangan terbaik. Sebuah fitur asli dianggap penting jika kepentingannya secara konsisten lebih tinggi daripada fitur bayangan terbaik setelah melalui beberapa iterasi. Fitur yang kepentingannya lebih rendah akan ditolak.

2.5 Voxel dan Peran Isotropic Resampling pada MRI untuk *radiomics*

Dalam pencitraan medis dan pemrosesan citra tiga dimensi, *voxel* (*volume element*) merupakan unit dasar yang digunakan untuk merepresentasikan informasi spasial dalam ruang tiga dimensi. *Voxel* dapat dipandang sebagai ekstensi piksel pada citra dua dimensi, di mana setiap *voxel* merepresentasikan nilai intensitas atau atribut tertentu pada suatu posisi di dalam volume citra. Representasi berbasis voxel memungkinkan struktur anatomi dan objek medis dimodelkan secara diskret dalam ruang 3D, sehingga mendukung analisis visual maupun kuantitatif pada berbagai modalitas pencitraan medis seperti *Computed Tomography* (CT) dan *Magnetic Resonance Imaging* (MRI) [23], [24]. Ukuran voxel secara langsung menentukan resolusi spasial citra dan tingkat detail informasi yang dapat direpresentasikan dalam analisis medis.

Pencitraan medis berbasis *MRI*, adalah citra tiga dimensi tersusun atas elemen volumetrik yang disebut *voxel*, yang merepresentasikan resolusi spasial pada arah sumbu x, y, dan z. Perbedaan parameter akuisisi citra, jenis pemindai, serta protokol pencitraan antar pusat layanan kesehatan dapat menyebabkan variasi ukuran *voxel*, khususnya pada dataset *MRI multi-center* [25]. Variasi resolusi *voxel* ini berpotensi menimbulkan heterogenitas spasial yang dapat memengaruhi konsistensi dan stabilitas fitur *radiomics* yang diekstraksi. Penelitian terdahulu menunjukkan bahwa perubahan parameter akuisisi yang secara langsung memengaruhi ukuran *voxel*, seperti *pixel spacing* dan *slice thickness*, dapat menyebabkan variasi signifikan pada nilai serta *repeatability* fitur *radiomics*, terutama pada fitur tekstur tiga dimensi [26].

Isotropic voxel resampling merupakan salah satu teknik praproses citra yang bertujuan untuk menyeragamkan ukuran *voxel* pada seluruh citra dengan menjadikan resolusi spasial sama pada setiap dimensi. Dengan menerapkan *resampling* isotropik, citra MRI dengan resolusi awal yang berbeda dapat distandarisasi ke ukuran *voxel* tertentu, sehingga mengurangi bias akibat perbedaan resolusi dan meningkatkan keterbandingan antar citra. Pendekatan ini banyak digunakan dalam studi *radiomics* untuk meningkatkan reprodusibilitas dan keandalan fitur yang dihasilkan [11].

Dalam konteks *radiomics*, resampling *voxel* umumnya dilakukan sebelum proses ekstraksi fitur, karena banyak fitur tekstur dan bentuk sangat sensitif terhadap perubahan resolusi spasial. Standarisasi ukuran *voxel* diharapkan dapat meningkatkan stabilitas fitur serta kinerja model pembelajaran mesin yang

dibangun berdasarkan fitur-fitur tersebut. Oleh karena itu, *isotropic voxel resampling* menjadi tahapan penting dalam pipeline *radiomics*, khususnya pada penelitian yang melibatkan data *MRI* dari berbagai pusat dengan variasi protokol pencitraan.

2.6 Algoritma Pembelajaran Mesin (*Machine Learning*)

Penelitian ini menerapkan tiga algoritma utama untuk klasifikasi status invasi otot, yang masing-masing mewakili paradigma pembelajaran yang berbeda: geometris (*Support Vector Machine*), *ensemble* (*eXtreme Gradient Boosting*), dan koneksionis (*MLP (Multilayer Perceptron)/Neural Network*). Pemilihan ketiga algoritma ini didasarkan pada efektivitasnya yang terbukti dalam studi *radiomics* kanker kandung kemih terkini [27], [28].

2.6.1 Support Vector Machine (SVM)

Support Vector Machine (*SVM*) adalah algoritma *supervised learning* yang bekerja dengan prinsip pencarian *hyperplane* (bidang pemisah) optimal yang memisahkan dua kelas data dengan margin terbesar. *SVM* dikenal memiliki kemampuan generalisasi yang baik pada dataset dengan dimensi tinggi namun jumlah sampel terbatas, seperti yang sering ditemukan dalam *radiomics* [29]. Efektivitas SVM dalam berbagai tugas klasifikasi telah ditunjukkan dalam penelitian-penelitian sebelumnya [30], [31]. Cara kerjanya secara konseptual adalah sebagai berikut:

1. Konsep *Hyperplane* dan *Margin*
   Diberikan sekumpulan data latih (x_i, y_i) di mana x_i adalah vektor fitur dan $y_i \in \{-1, +1\}$ adalah label kelas. SVM mencari vektor bobot w dan bias b yang memenuhi persamaan *hyperplane*:

   $$w \cdot x + b = 0 \tag{2.5}$$

   Tujuan utama SVM adalah menemukan *hyperplane* optimal yang mampu memisahkan dua kelas dengan margin terbesar. Margin merupakan jarak antara *hyperplane* dengan titik data terdekat yang disebut sebagai *support vectors*. Untuk memaksimalkan margin tersebut, SVM dirumuskan sebagai masalah optimasi berikut:

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{2.6}$$

Dengan batasan (*constraint*):

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, 2, ..., n \tag{2.7}$$

Pada kasus data yang tidak dapat dipisahkan secara linier, digunakan pendekatan *soft margin* dengan menambahkan variabel slack ξ_i untuk mengakomodasi kesalahan klasifikasi:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \tag{2.8}$$

Dengan constraint:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{2.9}$$

Persamaan tersebut memastikan bahwa setiap data latih berada pada sisi yang benar dari *hyperplane* dan memiliki jarak minimum terhadap batas keputusan. Titik-titik data yang berada paling dekat dengan *hyperplane* disebut sebagai *support vectors* dan berperan penting dalam menentukan posisi *hyperplane* optimal. Parameter C berfungsi untuk mengontrol *trade-off* antara margin yang lebih besar dan kesalahan klasifikasi. Nilai C yang besar cenderung menghasilkan margin yang lebih sempit dengan kesalahan yang lebih kecil, sedangkan nilai C yang kecil menghasilkan margin yang lebih lebar dengan kemungkinan kesalahan yang lebih besar.

```
Input:
    Training data D = {(x1, y1), (x2, y2), ..., (xn, yn)}
    where yi in {+1, -1}

Parameters:
    Learning rate alpha
    Regularization parameter lambda
    Number of iterations T

-----------------------------------------------
Process: Training SVM Linear
-----------------------------------------------
```

```
Initialize w = 0
Initialize b = 0

For t = 1 to T:
    For each (xi, yi) in D:

        c = yi * (w . xi + b)

        If c >= 1:
            w = w - alpha * (2 * lambda * w)
        Else:
            w = w - alpha * (2 * lambda * w - yi * xi)
            b = b + alpha * yi

-----------------------------------------------
Process: Prediction
-----------------------------------------------
For each test sample x:

    score = w . x + b

    If score >= 0:
        y_pred = +1
    Else:
        y_pred = -1

-----------------------------------------------
Output:
    Predicted label y_pred
```

Kode 2.1: Pseudocode Alur SVM Linear

2. *Kernel Trick*

   Pada banyak kasus, data tidak dapat dipisahkan secara linier dalam ruang fitur asli. Oleh karena itu, SVM menggunakan *kernel trick* untuk memetakan data ke ruang berdimensi lebih tinggi. Transformasi ini dinyatakan sebagai fungsi pemetaan:

$$\phi(x) : x \rightarrow \mathscr{H} \tag{2.10}$$

   Dalam bentuk dual, fungsi optimasi SVM dapat dituliskan sebagai:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{2.11}$$

Dengan constraint:

$$0 \leq \alpha_i \leq C \tag{2.12}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.13}$$

Fungsi keputusan SVM kemudian dapat dituliskan sebagai:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{2.14}$$

Salah satu kernel yang umum digunakan adalah *Radial Basis Function* (RBF) yang didefinisikan sebagai:

$$K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right) \tag{2.15}$$

Selain RBF, beberapa kernel lain yang umum digunakan adalah:

Kernel Linear

$$K(x_i, x_j) = x_i^T x_j \tag{2.16}$$

Kernel Polynomial

$$K(x_i, x_j) = (x_i^T x_j + 1)^d \tag{2.17}$$

Parameter γ mengontrol seberapa jauh pengaruh satu sampel pelatihan, sedangkan parameter C (regularisasi) mengontrol *trade-off* antara kesalahan klasifikasi dan lebar *margin*. Optimasi parameter ini sangat penting untuk meningkatkan performa model dan mencegah *overfitting*.

```
Input:
    Training data D = {(x1, y1), (x2, y2), ..., (xn, yn)}
    where yi in {+1, -1}

```

```
Parameters:
    Learning rate alpha
    Regularization parameter lambda
    Number of iterations T
    RBF Kernel parameter gamma

-----------------------------------------------
Process: Initialization & Kernel Function
-----------------------------------------------
Initialize array beta of size n with all 0s
Initialize b = 0

// Definisi Fungsi Kernel RBF
Function K(xi, xj):
    Return exp(-gamma * ||xi - xj||^2)

-----------------------------------------------
Process: Training SVM RBF Kernel
-----------------------------------------------
For t = 1 to T:
    For i = 1 to n:

        kernel_sum = 0
        // Optimasi: Hanya hitung data yang beta-nya tidak nol
        For j = 1 to n:
            If absolute_value(beta[j]) > 0:
                kernel_sum = kernel_sum + beta[j] * K(xj, xi)

        score = kernel_sum + b
        c = yi * score

        // Menerapkan penalti regularisasi ke semua bobot beta
        For j = 1 to n:
            beta[j] = beta[j] * (1 - 2 * alpha * lambda)

        // Cek margin (Gradient Update)
        If c < 1:
            beta[i] = beta[i] + alpha * yi
            b = b + alpha * yi

-----------------------------------------------
```

```
Process: Extract Support Vectors
-------------------------------------------------
Initialize list SV
For i = 1 to n:
    // Ekstraksi hanya data yang menjadi Support Vector
    If absolute_value(beta[i]) > 0:
        Add (xi, beta[i]) to SV

-------------------------------------------------
Process: Prediction
-------------------------------------------------
For each test sample x:

    kernel_sum = 0

    // Prediksi hanya dibandingkan dengan list Support Vector
    For each (x_sv, beta_sv) in SV:
        kernel_sum = kernel_sum + beta_sv * K(x_sv, x)

    score = kernel_sum + b

    If score >= 0:
        y_pred = +1
    Else:
        y_pred = -1

-------------------------------------------------
Output:
    Predicted label y_pred
```

Kode 2.2: Pseudocode Alur SVM Kernel RBF

2.6.2 eXtreme Gradient Boosting (XGBoost)

XGBoost adalah pengembangan dari algoritma *Gradient Boosting Decision Tree* (*GBDT*) yang dirancang untuk kecepatan dan performa tinggi pada data tabular. Dalam studi *radiomics* kanker kandung kemih, *XGBoost* sering menunjukkan performa superior karena kemampuannya menangani interaksi fitur non-linier dan data yang tidak seimbang [32]. Cara kerjanya secara konseptual adalah sebagai berikut:

1. Prinsip Boosting

   Berbeda dengan model tunggal, *XGBoost* membangun model secara bertahap (*additive*). Model memprediksi hasil dengan menjumlahkan skor dari sekumpulan pohon keputusan (f_k):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{2.18}$$

   Di mana f_k merupakan pohon keputusan ke-k, dan K adalah jumlah total pohon yang dibangun. Setiap pohon baru ditambahkan secara iteratif untuk memperbaiki kesalahan (*residual*) dari model sebelumnya. Proses pembelajaran dilakukan menggunakan metode *gradient boosting*, di mana model baru dilatih berdasarkan gradien dari fungsi *loss*. Secara umum, pembaruan model pada iterasi ke-t dapat dituliskan sebagai:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{2.19}$$

   Persamaan tersebut menunjukkan bahwa setiap pohon baru berfungsi untuk memperbaiki kesalahan dari model sebelumnya, sehingga performa model meningkat secara bertahap.

2. Fungsi Objektif dan Regularisasi

   Keunggulan utama *XGBoost* dibanding *GBDT* biasa adalah adanya komponen regularisasi (Ω) dalam fungsi objektifnya untuk mencegah *overfitting*, yang sangat krusial pada *dataset radiomics* dengan jumlah fitur yang besar. Fungsi objektif yang diminimalkan adalah:

$$\mathscr{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.20}$$

   Di mana $l(y_i, \hat{y}_i)$ adalah fungsi *loss* yang mengukur kesalahan prediksi, seperti *log-loss* untuk klasifikasi biner, dan $\Omega(f_k)$ adalah fungsi regularisasi yang mengontrol kompleksitas model. Fungsi regularisasi pada *XGBoost* didefinisikan sebagai:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{2.21}$$

Di mana T adalah jumlah daun (*leaf nodes*) pada pohon, w adalah bobot masing-masing daun, γ adalah parameter penalti jumlah daun, dan λ adalah parameter regularisasi bobot. Regularisasi ini bertujuan untuk mengontrol kompleksitas model agar tidak terlalu kompleks sehingga dapat meningkatkan kemampuan generalisasi model.

Dengan kombinasi pendekatan *gradient boosting* dan regularisasi, *XGBoost* mampu menangkap pola non-linier yang kompleks sekaligus menjaga stabilitas model, sehingga sangat efektif digunakan pada dataset *radiomics* dengan dimensi tinggi [32].

```
Input:
    Training data D = {(x1, y1), (x2, y2), ..., (xn, yn)}

Parameters:
    Number of trees T
    Learning rate eta

----------------------------------------------------
Process: Training XGBoost
----------------------------------------------------
Initialize prediction y_pred(i) = 0 for all samples

For t = 1 to T:

    For each sample (xi, yi):
        Compute gradient (residual):
            g_i = dL(yi, y_pred(i))

    Train decision tree h_t(x) using {g_i}

    For each sample (xi):
        Update prediction:
            y_pred(i) = y_pred(i) + eta * h_t(xi)

----------------------------------------------------
Process: Prediction
----------------------------------------------------
For each test sample x:

    y_pred = 0

```

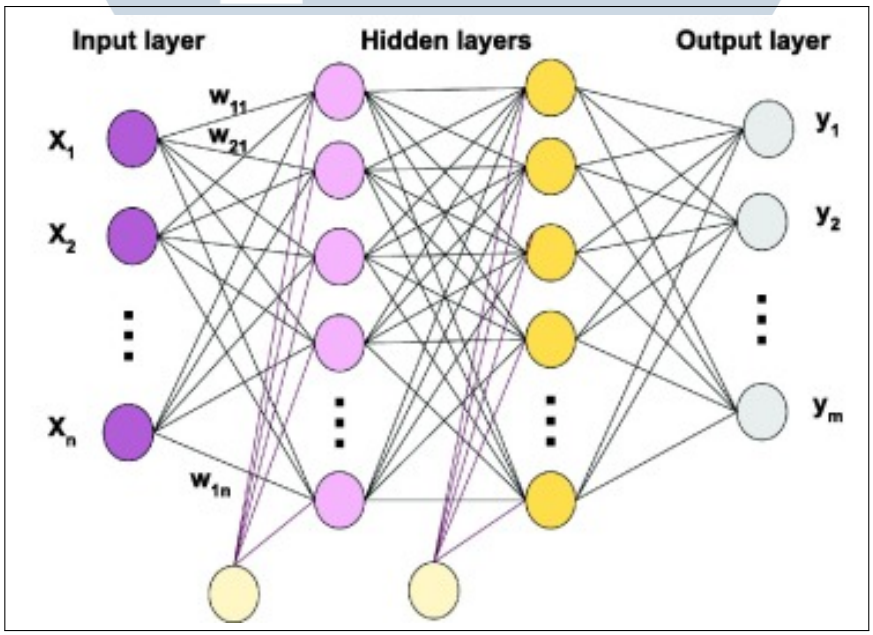
```
    For t = 1 to T:
        y_pred = y_pred + eta * h_t(x)

-------------------------------------------------
Output:
    Predicted value or class label
```

Kode 2.3: Pseudocode Alur XGBoost

2.6.3 Multilayer Perceptron - MLP (Neural Network)

Multilayer Perceptron (MLP) adalah jenis jaringan saraf tiruan yang terdiri dari lapisan input, satu atau lebih lapisan tersembunyi (*hidden layers*), dan lapisan output. *MLP* telah digunakan dalam model *Deep Radiomics* untuk menangkap representasi fitur abstrak yang mendalam dari citra *CT* dan *MRI* [27].



Gambar 2.3. Arsitektur Multilayer Perceptron (diadaptasi dari [33])

MLP bekerja dengan melakukan transformasi bertingkat melalui beberapa lapisan neuron. Setiap neuron dalam lapisan tersembunyi melakukan transformasi linier diikuti oleh fungsi aktivasi non-linier. Jika x adalah input, W adalah bobot, dan b adalah bias, maka output neuron pada lapisan tersembunyi (h) dihitung sebagai:

$$h = f(Wx + b) \tag{2.22}$$

Di mana $f(\cdot)$ merupakan fungsi aktivasi non-linier yang memungkinkan jaringan saraf mempelajari hubungan kompleks antar fitur. Penelitian ini menggunakan fungsi aktivasi *ReLU (Rectified Linear Unit)* pada lapisan tersembunyi yang didefinisikan sebagai:

$$f(x) = \max(0, x) \tag{2.23}$$

Fungsi ReLU membantu mempercepat proses konvergensi dan mengurangi masalah *vanishing gradient* pada jaringan saraf dalam. Setelah melalui lapisan tersembunyi, output akhir dihitung menggunakan fungsi aktivasi *Sigmoid* untuk menghasilkan probabilitas klasifikasi biner:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.24}$$

Output akhir model MLP untuk klasifikasi biner dapat dirumuskan sebagai:

$$\hat{y} = \sigma(W_o h + b_o) \tag{2.25}$$

Di mana W_o dan b_o masing-masing merupakan bobot dan bias pada lapisan output. Nilai $\hat{y}$ yang dihasilkan berada pada rentang 0 hingga 1 dan diinterpretasikan sebagai probabilitas kelas. Jika $\hat{y} \geq 0.5$ maka data diklasifikasikan ke kelas positif (MIBC), sedangkan jika $\hat{y} < 0.5$ maka diklasifikasikan ke kelas negatif (NMIBC).

Dengan struktur berlapis dan fungsi aktivasi non-linier, MLP mampu mempelajari hubungan kompleks antar fitur radiomics sehingga dapat meningkatkan performa klasifikasi pada dataset dengan dimensi tinggi [27].

```
Input:
    Training data D = {(x1, y1), (x2, y2), ..., (xn, yn)}

Parameters:
    Learning rate alpha
    Number of epochs E

------------------------------------------------
Process: Training MLP
------------------------------------------------
Initialize weights W1, W_o and biases b1, b_o randomly

For epoch = 1 to E:
    For each sample (xi, yi):

```

```
        Forward propagation:

            Hidden layer:
                z1 = W1 . xi + b1
                h = ReLU(z1)            // f(x) = max(0, x)

            Output layer:
                z2 = W_o . h + b_o
                y_hat = sigmoid(z2)   //    (x) = 1 / (1 + e^-x)

        Compute loss:
            L = loss(yi, y_hat)

        Backpropagation:
            Compute gradients of L w.r.t W1, W_o, b1, b_o

        Update parameters:
            W1 = W1 - alpha * gradient
            W_o = W_o - alpha * gradient
            b1 = b1 - alpha * gradient
            b_o = b_o - alpha * gradient

------------------------------------------------
Process: Prediction
------------------------------------------------
For each test sample x:

    z1 = W1 . x + b1
    h = ReLU(z1)

    z2 = W_o . h + b_o
    y_hat = sigmoid(z2)

    If y_hat >= 0.5:
        y_pred = 1
    Else:
        y_pred = 0

------------------------------------------------
Output:
    Predicted class label y_pred
```

Kode 2.4: Multilayer Perceptron (MLP) for Binary Classification

2.6.4 Hyperparameter Tuning dengan RandomizedSearchCV

Hyperparameter tuning merupakan tahapan penting dalam pengembangan model pembelajaran mesin karena berpengaruh langsung terhadap performa serta kemampuan generalisasi model. Berbagai metode penalaan telah dikembangkan, seperti *grid search*, *random search*, optimisasi *Bayesian*, dan *meta-learning* [34]. Pada penelitian ini digunakan metode *RandomizedSearchCV* karena mampu mengeksplorasi ruang hyperparameter yang luas dengan biaya komputasi yang lebih efisien dibandingkan metode *GridSearchCV*.

RandomizedSearchCV bekerja dengan memilih sejumlah kombinasi parameter secara acak dari distribusi hyperparameter yang telah ditentukan. Pendekatan ini memungkinkan eksplorasi parameter yang lebih luas tanpa harus mengevaluasi seluruh kombinasi parameter yang tersedia. Hal ini sangat penting pada dataset *radiomics* yang memiliki dimensi fitur tinggi dan kompleksitas model yang besar.

Proses evaluasi dilakukan menggunakan *5-fold cross-validation*, di mana dataset pelatihan dibagi menjadi lima subset. Setiap subset digunakan secara bergantian sebagai data validasi, sedangkan subset lainnya digunakan sebagai data pelatihan. Nilai performa akhir diperoleh dari rata-rata performa pada seluruh fold.

Secara matematis, *RandomizedSearchCV* bertujuan untuk mencari konfigurasi hyperparameter optimal θ^* dari sejumlah sampel parameter acak yang meminimalkan nilai fungsi loss rata-rata pada seluruh fold validasi:

$$\theta^* = \arg \min_{\theta_j \in \Theta_{sample}} \frac{1}{K} \sum_{k=1}^{K} L_k(\theta_j) \tag{2.26}$$

di mana Θ_{sample} merupakan subset parameter yang dipilih secara acak dari ruang pencarian Θ, K adalah jumlah fold *cross-validation*, dan $L_k(\theta_j)$ adalah nilai fungsi loss pada fold ke-k untuk kombinasi parameter ke-j.

Pada penelitian ini, jumlah iterasi pencarian ditetapkan sebanyak 100 kombinasi parameter (*n_iter = 100*) dengan metrik evaluasi menggunakan *accuracy*. Pendekatan ini digunakan untuk mengoptimalkan hyperparameter pada setiap algoritma klasifikasi sebagai berikut:

1. Support Vector Machine (SVM)

   Hyperparameter yang dioptimalkan pada model SVM meliputi parameter regularisasi C, parameter *kernel* γ, serta jenis *kernel* yang digunakan seperti

linear, *rbf*, *polynomial*, dan *sigmoid*. Selain itu, model juga menggunakan *class weight balancing* untuk mengatasi ketidakseimbangan data.

2. eXtreme Gradient Boosting (XGBoost)
   Pada model *XGBoost*, hyperparameter yang dituning mencakup jumlah pohon (*n_estimators*), kedalaman maksimum pohon (*max_depth*), *learning rate*, rasio *subsample*, parameter γ, rasio pemilihan fitur (*colsample_bytree*), serta parameter regularisasi L1 (*reg_alpha*) dan L2 (*reg_lambda*). Selain itu, parameter *scale_pos_weight* juga digunakan untuk menangani ketidakseimbangan kelas pada dataset.

3. Multilayer Perceptron (MLP)
   Pada model *Multilayer Perceptron*, hyperparameter yang dioptimalkan meliputi konfigurasi *hidden layer* (*hidden_layer_sizes*), fungsi aktivasi (*relu*, *tanh*, dan *logistic*), parameter regularisasi α, serta skema *learning rate*. Selain itu, model juga menggunakan *early stopping* dan jumlah iterasi maksimum sebesar 2000 untuk meningkatkan stabilitas proses pelatihan dan mencegah *overfitting*.

Pendekatan *RandomizedSearchCV* ini memastikan bahwa setiap algoritma dievaluasi menggunakan konfigurasi hyperparameter terbaiknya masing-masing, sehingga hasil perbandingan performa model menjadi lebih objektif dan reliabel.

2.7 Stacking Model

Metode *stacking* merupakan salah satu teknik *ensemble learning* yang bertujuan meningkatkan performa prediksi dengan menggabungkan beberapa model pembelajaran mesin melalui proses pembelajaran bertingkat. Berbeda dengan metode *ensemble* sederhana seperti *bagging* atau *boosting*, *stacking* tidak hanya menggabungkan hasil prediksi, tetapi juga melatih sebuah model tambahan (*meta-learner*) untuk mempelajari cara terbaik dalam mengombinasikan keluaran dari beberapa model dasar [35].

Pada pendekatan *stacking*, beberapa algoritma klasifikasi yang berbeda terlebih dahulu dilatih secara independen menggunakan *dataset* yang sama. Setiap model dasar menghasilkan prediksi yang kemudian digunakan sebagai fitur baru pada tingkat kedua. Selanjutnya, *meta-learner* dilatih menggunakan keluaran tersebut untuk menghasilkan prediksi akhir yang diharapkan lebih akurat dan stabil

dibandingkan model tunggal. Struktur ini memungkinkan *stacking* memanfaatkan keunggulan masing-masing algoritma sekaligus mengurangi kelemahan individual model.

Secara matematis, misalkan terdapat M model dasar yang dilatih menggunakan dataset X, maka masing-masing model menghasilkan prediksi sebagai berikut:

$$h_1(x), h_2(x), h_3(x), ..., h_M(x) \tag{2.27}$$

Pada penelitian ini digunakan tiga model dasar yaitu *Support Vector Machine* (SVM), *eXtreme Gradient Boosting* (XGBoost), dan *Multilayer Perceptron* (MLP), sehingga prediksi masing-masing model dapat dituliskan sebagai:

$$z_i = [h_{SVM}(x_i), h_{XGB}(x_i), h_{MLP}(x_i)] \tag{2.28}$$

Vektor z_i kemudian digunakan sebagai input bagi model tingkat kedua (*meta-learner*). Pada penelitian ini, model *meta-learner* yang digunakan adalah *Logistic Regression*. Model ini mempelajari kombinasi linear dari prediksi model dasar untuk menghasilkan prediksi akhir:

$$\hat{y}_i = \sigma(w_1 h_{SVM}(x_i) + w_2 h_{XGB}(x_i) + w_3 h_{MLP}(x_i) + b) \tag{2.29}$$

di mana w_1, w_2, w_3 merupakan bobot yang dipelajari oleh *meta-learner*, b adalah bias, dan σ merupakan fungsi sigmoid yang digunakan untuk mengubah output linear menjadi probabilitas klasifikasi biner. Fungsi sigmoid tersebut telah dijelaskan sebelumnya pada Subbab 2.6.3 (*Multilayer Perceptron*).

Nilai $\hat{y}_i$ yang dihasilkan merupakan probabilitas klasifikasi akhir. Jika $\hat{y}_i \geq 0.5$ maka data diklasifikasikan sebagai kelas positif (MIBC), sedangkan jika $\hat{y}_i < 0.5$ maka diklasifikasikan sebagai kelas negatif (NMIBC).

Secara konseptual, *stacking* bekerja berdasarkan penggabungan prediksi dari beberapa model untuk menghasilkan keputusan yang lebih andal dibandingkan model individual. Integrasi keluaran dari berbagai model dasar memungkinkan pengurangan variansi prediksi serta meningkatkan stabilitas model akhir. Dengan mempelajari kombinasi prediksi dari berbagai model, *meta-learner* dapat memperbaiki kesalahan yang tidak dapat ditangani oleh satu model saja [36].

Metode *stacking* juga telah banyak diterapkan pada berbagai penelitian

klasifikasi kanker untuk meningkatkan performa prediksi dibandingkan model tunggal [37], [38], [39]. Pada studi berbasis data citra medis maupun data klinis, pendekatan *stacking ensemble* mampu mengombinasikan keunggulan beberapa algoritma pembelajaran mesin sehingga menghasilkan peningkatan akurasi, sensitivitas, serta stabilitas model klasifikasi. Keberhasilan penerapan tersebut menunjukkan bahwa metode *stacking* memiliki kemampuan generalisasi yang baik dalam menangani karakteristik data kanker yang kompleks dan berdimensi tinggi.

2.7.1 Logistic Regression

Logistic Regression merupakan salah satu algoritma pembelajaran mesin yang umum digunakan untuk permasalahan klasifikasi, khususnya klasifikasi biner. Berbeda dengan regresi linear yang digunakan untuk memprediksi nilai kontinu, Logistic Regression digunakan untuk memprediksi probabilitas suatu data termasuk ke dalam kelas tertentu. Algoritma ini memanfaatkan fungsi logistik atau sigmoid untuk memetakan nilai prediksi ke dalam rentang probabilitas antara 0 dan 1.

Secara matematis, Logistic Regression menggunakan fungsi sigmoid yang dinyatakan sebagai berikut:

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}} \tag{2.30}$$

di mana $P(y=1|x)$ merupakan probabilitas suatu data termasuk ke dalam kelas positif, β_0 merupakan intercept, $\beta_1, \beta_2, \ldots, \beta_n$ merupakan koefisien fitur, dan $x_1, x_2, \ldots, x_n$ merupakan fitur input.

Dalam penelitian ini, Logistic Regression digunakan sebagai *meta-learner* pada metode *stacking*. Model ini berfungsi untuk menggabungkan hasil prediksi dari beberapa model dasar (*base learner*) seperti Support Vector Machine (SVM), XGBoost, dan Multilayer Perceptron (MLP). Output berupa probabilitas dari masing-masing model dasar kemudian digunakan sebagai fitur masukan bagi Logistic Regression untuk menghasilkan prediksi akhir.

Penggunaan Logistic Regression sebagai *meta-learner* didasarkan pada karakteristiknya sebagai algoritma linear yang relatif sederhana dan memiliki kompleksitas komputasi yang rendah. Hal ini menjadi penting karena model *stacking* telah melibatkan beberapa algoritma pembelajaran mesin pada tahap *base learner*, sehingga penggunaan algoritma yang lebih sederhana pada tahap *meta-learner* dapat membantu mengurangi beban komputasi serta menjaga

stabilitas model secara keseluruhan. Selain itu, Logistic Regression juga telah banyak digunakan dalam pendekatan *stacking ensemble* pada berbagai penelitian sebelumnya karena kemampuannya dalam menggabungkan output probabilitas dari beberapa model menjadi satu prediksi akhir yang terintegrasi [40], [41].

2.8 Metrik Evaluasi

Untuk mengevaluasi performa model secara terpisah, digunakan beberapa metrik yang dapat menggambarkan kinerja model dari berbagai aspek, terutama dalam menganalisis pola kesalahan yang terjadi.

Confusion Matrix adalah sebuah tabel yang digunakan untuk memvisualisasikan performa dari sebuah algoritma klasifikasi. Dalam konteks klasifikasi multi-kelas dengan N kelas, matriks ini akan berukuran $N \times N$. Setiap baris merepresentasikan kelas aktual, sementara setiap kolom merepresentasikan kelas yang diprediksi oleh model [42]. Struktur matriks ini memungkinkan analisis mendalam terhadap pola kesalahan prediksi dan menjadi dasar untuk menghitung metrik-metrik evaluasi lainnya.

Untuk mengukur seberapa baik model klasifikasi membedakan antara kelas *NMIBC* dan *MIBC*, digunakan *Confusion Matrix*. Matriks ini merepresentasikan perbandingan antara prediksi model dengan label kebenaran dasar (*ground truth*). Berdasarkan matriks ini, empat parameter dasar dapat dihitung:

1. *True Positive* (TP): Jumlah kasus positif yang diprediksi dengan benar (misal: MIBC terdeteksi sebagai MIBC).
2. *True Negative* (TN): Jumlah kasus negatif yang diprediksi dengan benar (misal: NMIBC terdeteksi sebagai NMIBC).
3. *False Positive* (FP): Jumlah kasus negatif yang salah diprediksi sebagai positif (Error Tipe I).
4. *False Negative* (FN): Jumlah kasus positif yang salah diprediksi sebagai negatif (Error Tipe II).

Berdasarkan parameter tersebut, kinerja model diukur menggunakan metrik-metrik berikut:

Accuracy mengukur proporsi total prediksi yang benar terhadap seluruh jumlah sampel data. Metrik ini memberikan gambaran umum mengenai seberapa

sering model membuat keputusan yang tepat secara global. Rumus akurasi didefinisikan sebagai berikut:

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

Meskipun akurasi adalah metrik yang paling intuitif, penggunaannya pada data medis perlu diperhatikan. Dalam dataset yang tidak seimbang (*imbalanced dataset*), di mana jumlah pasien satu kelas jauh lebih banyak dari kelas lainnya, akurasi yang tinggi bisa menipu (*misleading*). Model dapat mencapai akurasi tinggi hanya dengan memprediksi kelas mayoritas saja, namun gagal mendeteksi kasus minoritas yang mungkin justru lebih klinis dan kritis [42].

Precision atau disebut juga *Positive Predictive Value*, mengukur seberapa andal model saat memprediksi kelas positif. Metrik ini menghitung dari semua pasien yang diprediksi mengidap kanker ganas (*MIBC*) berapa persen yang benar-benar mengidapnya.

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

Nilai presisi yang tinggi mengindikasikan tingkat *False Positive* (FP) yang rendah. Dalam konteks medis, presisi sangat penting untuk meminimalkan alarm palsu, di mana pasien yang sebenarnya kondisinya ringan tidak perlu menjalani prosedur lanjutan yang invasif dan memicu kecemasan yang tidak perlu [42].

Recall dalam dunia medis lebih dikenal sebagai Sensitivitas (*Sensitivity*), mengukur kemampuan model untuk menemukan semua kasus positif yang ada. Metrik ini menghitung dari seluruh pasien yang benar-benar mengidap kanker ganas, berapa persen yang berhasil dideteksi oleh sistem.

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

Dalam diagnosis kanker, *Recall* sering dianggap sebagai metrik yang paling krusial. Nilai *Recall* yang rendah berarti banyak terjadi *False Negative* (FN), yaitu pasien yang sakit tidak terdeteksi. Kesalahan ini dapat berakibat fatal karena pasien kehilangan kesempatan untuk mendapatkan penanganan dini yang menyelamatkan nyawa [42].

F1-Score adalah rata-rata harmonik dari Presisi dan *Recall*. Berbeda dengan rata-rata aritmatika biasa, rata-rata harmonik memberikan bobot lebih pada nilai yang rendah, sehingga *F1-Score* hanya akan bernilai tinggi jika kedua komponen (Presisi dan *Recall*) seimbang dan tinggi.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.34}$$

F1-Score menjadi metrik pilihan utama dalam penelitian klasifikasi medis, terutama ketika *dataset* memiliki distribusi kelas yang tidak seimbang. Metrik ini memberikan penilaian yang lebih objektif dibandingkan akurasi karena menuntut model untuk memiliki kinerja yang baik dalam meminimalisir kesalahan positif palsu maupun negatif palsu secara bersamaan [42].