BAB 2
LANDASAN TEORI

2.1 Kanker Lambung (*Gastric Cancer*)

Kanker lambung merupakan salah satu jenis keganasan yang berkembang pada lapisan mukosa hingga dinding lambung, dan masih menjadi salah satu penyebab utama kematian akibat kanker secara global [1]. Penyakit ini sering kali terdiagnosis pada stadium lanjut karena gejala awal yang tidak spesifik, sehingga tingkat mortalitasnya masih relatif tinggi dibandingkan beberapa jenis kanker lainnya.

Secara biologis, perkembangan kanker lambung tidak terjadi secara tiba-tiba, melainkan melalui proses bertahap yang dikenal sebagai *Correa cascade* [15]. Tahapan ini dimulai dari *gastritis kronis*, kemudian berkembang menjadi *atrophic gastritis*, dilanjutkan dengan *metaplasia intestinal*, *displasia*, hingga akhirnya menjadi *karsinoma invasif*. Proses ini menunjukkan bahwa kanker lambung merupakan hasil dari akumulasi perubahan genetik dan epigenetik yang berlangsung dalam jangka waktu panjang.

Dari sudut pandang molekuler, perkembangan kanker lambung dipengaruhi oleh akumulasi mutasi genetik yang terjadi secara progresif pada sel epitel lambung [4, 14]. Mutasi ini dapat dipicu oleh berbagai faktor, seperti infeksi *Helicobacter pylori*, inflamasi kronis, faktor lingkungan, serta predisposisi genetik. Akumulasi mutasi tersebut menyebabkan gangguan pada mekanisme regulasi sel, termasuk proliferasi, apoptosis, dan perbaikan DNA, yang pada akhirnya mendorong transformasi sel normal menjadi sel kanker.

Untuk merepresentasikan proses tersebut secara sederhana, dapat digunakan model matematis yang menggambarkan pertumbuhan jumlah mutasi terhadap waktu sebagai berikut:

$$M(t) = M_0 + \lambda t \tag{2.1}$$

Persamaan tersebut menunjukkan bahwa jumlah mutasi pada waktu tertentu merupakan hasil dari jumlah mutasi awal (M_0) yang telah ada, ditambah dengan akumulasi mutasi baru yang terjadi secara linear terhadap waktu dengan laju λ. Model ini mengasumsikan bahwa laju mutasi bersifat konstan dalam rentang waktu tertentu, sehingga setiap peningkatan waktu akan meningkatkan jumlah mutasi secara proporsional.

Dalam konteks biologis, model ini dapat merepresentasikan akumulasi kerusakan DNA yang terus berlangsung akibat paparan faktor internal maupun eksternal, seperti stres oksidatif, paparan karsinogen, serta kegagalan mekanisme *DNA repair* [16]. Meskipun demikian, dalam kenyataannya laju mutasi bersifat dinamis dan dapat berubah seiring perkembangan mikroenvironment tumor, interaksi imunologis, serta heterogenitas genetik antar sel kanker.

Walaupun bersifat sederhana, model linear ini tetap memberikan pendekatan awal yang berguna untuk memahami dinamika progresi kanker. Model ini dapat digunakan sebagai dasar konseptual dalam analisis komputasional, khususnya dalam pengembangan model prediktif berbasis data *gene expression*. Dengan demikian, model ini tidak hanya berfungsi sebagai representasi matematis, tetapi juga sebagai jembatan antara konsep biologis dan pendekatan machine learning dalam analisis kanker.

Dimana $M(t)$ adalah jumlah mutasi pada waktu t, M_0 adalah jumlah mutasi awal, dan λ adalah laju akumulasi mutasi.

2.2 *Microsatellite Instability* (MSI)

Microsatellite adalah sekuens DNA pendek yang terdiri dari pengulangan basa nukleotida. Ketidakstabilan pada sekuens ini dikenal sebagai *Microsatellite Instability* (MSI) [5, 7].

Dalam proses replikasi DNA, kesalahan dapat terjadi akibat ketidaksempurnaan mekanisme penyalinan materi genetik [16]. Untuk mengukur tingkat kesalahan tersebut, digunakan pendekatan matematis berupa rasio antara jumlah kesalahan dengan total proses replikasi.

$$E = \frac{N_{error}}{N_{total}} \tag{2.2}$$

Persamaan ini menyatakan bahwa tingkat kesalahan replikasi DNA dihitung sebagai perbandingan antara jumlah kesalahan yang terjadi dengan total basa DNA yang direplikasi. Nilai error rate yang tinggi mengindikasikan adanya gangguan pada sistem mismatch repair (MMR) [6], yang berkontribusi terhadap terjadinya MSI dan peningkatan akumulasi mutasi. Secara biologis, peningkatan kesalahan ini dapat menyebabkan perubahan panjang pada sekuens microsatellite yang berulang, sehingga memicu ketidakstabilan genomik. Kondisi ini sangat penting dalam perkembangan kanker karena menyebabkan mutasi pada gen-gen penting yang mengatur pertumbuhan sel. Oleh karena itu, error rate menjadi indikator penting dalam analisis genetik dan sering digunakan sebagai parameter dalam penelitian kanker.

dimana E adalah *error rate*, N_{error} jumlah kesalahan replikasi, dan N_{total} total basa DNA yang direplikasi.

Untuk mengukur tingkat mutasi dalam suatu tumor secara keseluruhan, digunakan konsep *Tumor Mutation Burden* (TMB).

$$TMB = \frac{\text{Jumlah Mutasi}}{\text{Ukuran Genom}} \tag{2.3}$$

Persamaan ini menunjukkan bahwa TMB dihitung sebagai rasio antara jumlah mutasi dengan ukuran genom. Nilai TMB yang tinggi menunjukkan banyaknya perubahan genetik dalam sel tumor, yang sering dikaitkan dengan peningkatan respons imun terhadap kanker [6]. Hal ini terjadi karena mutasi yang tinggi menghasilkan neoantigen yang dapat dikenali oleh sistem imun sebagai target asing. Oleh karena itu, TMB sering digunakan sebagai biomarker dalam menentukan efektivitas terapi imun, khususnya pada pasien dengan status MSI-High [7]. Selain itu, nilai TMB juga dapat memberikan gambaran tentang kompleksitas mutasi dan tingkat agresivitas tumor.

2.3 *Gene Expression*

Gene expression merupakan proses biologis di mana informasi genetik yang tersimpan dalam DNA digunakan untuk menghasilkan produk fungsional berupa RNA maupun protein [16]. Proses ini menjadi dasar utama dalam memahami bagaimana suatu gen berperan dalam kondisi sel normal maupun penyakit, termasuk kanker. Perubahan tingkat ekspresi gen dapat mencerminkan aktivitas biologis sel dan sering digunakan sebagai indikator penting dalam analisis bioinformatika

modern.

Dalam analisis data *RNA sequencing* (RNA-Seq), jumlah pembacaan (*read counts*) yang diperoleh dari setiap gen tidak dapat langsung dibandingkan karena adanya bias teknis, seperti panjang gen dan kedalaman sequencing. Oleh karena itu, diperlukan metode normalisasi agar nilai ekspresi antar gen dan antar sampel dapat dibandingkan secara adil. Salah satu metode yang umum digunakan adalah *Transcripts Per Million* (TPM).

$$TPM_i = \frac{\frac{R_i}{L_i}}{\sum_{j=1}^{N} \frac{R_j}{L_j}} \times 10^6 \tag{2.4}$$

Pada persamaan tersebut, R_i merepresentasikan jumlah *read count* yang dipetakan ke gen ke-i, sedangkan L_i adalah panjang gen tersebut. Rasio $\frac{R_i}{L_i}$ disebut sebagai *reads per kilobase* yang bertujuan untuk mengoreksi bias akibat perbedaan panjang gen, karena gen yang lebih panjang cenderung menghasilkan lebih banyak *read* meskipun tingkat ekspresinya sama.

Selanjutnya, nilai tersebut dinormalisasi terhadap total seluruh gen dalam satu sampel, yaitu $\sum_{j=1}^{N} \frac{R_j}{L_j}$. Proses normalisasi ini memastikan bahwa total ekspresi seluruh gen dalam satu sampel memiliki skala yang sama, kemudian dikalikan dengan 10^6 untuk memudahkan interpretasi dalam satuan per juta (*per million scale*). Dengan demikian, TPM merepresentasikan proporsi relatif ekspresi suatu gen terhadap keseluruhan ekspresi gen dalam sampel tersebut.

Metode TPM sangat penting dalam analisis RNA-Seq [12] karena mampu mengurangi bias teknis dan memungkinkan perbandingan yang lebih konsisten antar sampel, baik pada kondisi normal maupun patologis. Hal ini menjadikan TPM sebagai salah satu metode normalisasi yang paling banyak digunakan dalam studi ekspresi gen diferensial dan analisis kanker berbasis data genomik.

Selain normalisasi, transformasi data juga diperlukan untuk meningkatkan kualitas distribusi data sebelum digunakan dalam model machine learning. Salah satu transformasi yang umum digunakan adalah transformasi logaritmik.

$$X' = \log_2(X + 1) \tag{2.5}$$

Pada persamaan tersebut, X merupakan nilai ekspresi gen asli, sedangkan X' adalah nilai setelah transformasi. Penambahan konstanta 1 bertujuan untuk menghindari nilai logaritma dari nol yang tidak terdefinisi secara matematis. Transformasi log basis 2 digunakan karena lebih mudah diinterpretasikan dalam konteks perubahan lipat (*fold change*) pada ekspresi gen.

Secara statistik, transformasi logaritmik berfungsi untuk mengurangi *skewness* atau kemencengan distribusi data, yang umumnya terjadi pada data ekspresi gen karena sebagian besar gen memiliki nilai ekspresi rendah, sementara sebagian kecil memiliki nilai sangat tinggi. Selain itu, transformasi ini juga membantu menstabilkan variansi data sehingga lebih sesuai untuk metode statistik dan algoritma machine learning [17].

Dengan demikian, kombinasi normalisasi TPM dan transformasi logaritmik merupakan tahap penting dalam preprocessing data *gene expression* untuk memastikan bahwa data yang digunakan dalam analisis memiliki skala yang konsisten, distribusi yang lebih stabil, serta lebih representatif untuk proses pemodelan klasifikasi berbasis machine learning.

2.4 *Data Acquisition*

Tahap *data acquisition* merupakan proses awal dalam penelitian yang bertujuan untuk mengumpulkan data dari berbagai sumber yang relevan. Dalam penelitian bioinformatika, data molekuler umumnya diperoleh dari repositori publik seperti *Gene Expression Omnibus* (GEO) dan *The Cancer Genome Atlas* (TCGA), yang menyediakan berbagai jenis data genomik untuk mendukung penelitian kanker [18, 19].

Pada penelitian ini, data diperoleh melalui platform *UCSC Xena Browser*, yaitu sebuah portal berbasis web yang mengintegrasikan berbagai dataset genomik dari TCGA dan sumber data kanker lainnya sehingga memudahkan proses eksplorasi, visualisasi, serta pengunduhan data untuk keperluan analisis [20]. Dataset yang digunakan adalah *TCGA Stomach Adenocarcinoma* (TCGA-STAD) dengan data ekspresi gen *IlluminaHiSeq Percentile UNC*. Dataset tersebut berisi nilai ekspresi gen hasil RNA-Seq yang telah dinormalisasi dalam bentuk *percentile rank*, di mana setiap gen memiliki nilai antara 0 hingga 100. Nilai yang lebih tinggi menunjukkan tingkat ekspresi gen yang relatif lebih tinggi dibandingkan gen lain pada sampel yang sama.

Data ekspresi gen yang diperoleh memiliki jumlah fitur yang sangat besar sehingga memerlukan representasi yang terstruktur agar dapat diproses secara efisien menggunakan metode *machine learning*. Untuk memudahkan proses pengolahan dan analisis, dataset direpresentasikan dalam bentuk matriks numerik sebagai berikut:

$$X \in \mathbb{R}^{n \times p} \tag{2.6}$$

Persamaan tersebut menunjukkan bahwa data direpresentasikan sebagai sebuah matriks berdimensi $n \times p$, di mana n menyatakan jumlah sampel atau observasi, sedangkan p menyatakan jumlah fitur atau variabel. Dalam penelitian ini, setiap baris merepresentasikan satu sampel pasien kanker lambung, sedangkan setiap kolom merepresentasikan tingkat ekspresi suatu gen [21]. Representasi ini memungkinkan data diproses secara sistematis menggunakan berbagai algoritma *machine learning* yang memerlukan masukan dalam bentuk numerik [17].

Selain itu, representasi matriks ini mempermudah proses manipulasi data seperti penyaringan fitur (*feature filtering*), normalisasi, serta transformasi data yang merupakan bagian penting dari tahap *preprocessing* [22]. Dengan struktur data yang terorganisir, proses analisis dapat dilakukan secara lebih efisien, konsisten, dan terukur sehingga mendukung keseluruhan tahapan penelitian.

2.5 *Exploratory Data Analysis* (EDA)

Exploratory Data Analysis (EDA) merupakan suatu pendekatan analitis yang bertujuan untuk menginvestigasi, meringkas, dan memahami karakteristik mendasar dari suatu dataset sebelum menerapkan algoritma pemodelan prediktif formal. Dalam koridor data transkriptomik berdimensi tinggi, tahapan EDA bertindak sebagai instrumen validasi krusial untuk mengidentifikasi keberadaan anomali teknis, mengevaluasi distribusi probabilitas fitur, serta mendeteksi struktur laten fungsional yang tersembunyi di dalam matriks ekspresi gen [23]. Melalui pemanfaatan visualisasi grafis dan ringkasan statistik deskriptif, peneliti dapat memperoleh pemahaman intuitif mengenai tingkat kompleksitas pemisahan kelas target serta potensi bias yang melekat pada data mentah medis [24].

2.5.1 Distribusi Kelimpahan Ekspresi Gen

Pada umumnya, data mentah kelimpahan hitungan (*raw counts*) hasil penjajaran sekuensing RNA-Seq memiliki karakteristik distribusi yang sangat condong ke kanan (*highly right-skewed*) akibat dominasi nilai ekspresi rendah pada mayoritas gen [25]. Oleh karena itu, pemeriksaan profil distribusi secara makroskopis diperlukan untuk mengevaluasi keberhasilan transformasi matematis non-linear seperti transformasi $\log_2(x+1)$ [26]. Keberhasilan transformasi ini ditandai dengan terbentuknya kurva simetris yang mendekati distribusi normal (*bell-shaped curve*), sehingga mampu mengurangi heteroskedastisitas (variansi yang tidak konstan), menstabilkan variansi, serta menekan derau teknis yang berasal dari proses pengukuran ekspresi gen [22].

2.5.2 Distribusi Variabilitas Fitur Gen

Analisis sebaran variansi global bertindak sebagai proses penyaringan fitur awal tanpa memanfaatkan informasi kelas (*unsupervised filtering*) untuk memetakan tingkat variabilitas ekspresi setiap gen pada seluruh sampel [21]. Dalam data ekspresi gen, sebagian besar gen merupakan *housekeeping genes* yang memiliki tingkat ekspresi relatif stabil dengan variansi yang sangat rendah karena berperan dalam mempertahankan fungsi dasar sel. Gen-gen tersebut umumnya tidak memberikan informasi yang cukup untuk membedakan kondisi biologis atau fenotipe penyakit, seperti status *Microsatellite Instability* (MSI) [19]. Oleh karena itu, visualisasi distribusi variansi digunakan untuk membantu menentukan nilai ambang (*threshold*) dalam menghilangkan fitur yang kurang informatif sehingga dapat mengurangi dimensi data dan meminimalkan dampak *curse of dimensionality* pada proses pembelajaran mesin [17].

2.5.3 Matriks Densitas Dua Dimensi dan *Hierarchical Clustering*

Visualisasi matriks intensitas dua dimensi atau *heatmap* merupakan teknik penyajian data transkriptomik yang menggabungkan gradasi warna numerik dengan algoritma pengelompokan hierarki (*hierarchical clustering*) [21]. Melalui pendekatan ini, tingkat kedekatan jarak Euclidean antar profil ekspresi sampel maupun gen diukur secara simultan dan direpresentasikan dalam bentuk pohon dendrogram [17]. *Heatmap* memungkinkan peneliti untuk mengidentifikasi keberadaan partisi blok warna kontras yang searah atau ko-ekspresi fungsional (*functional co-expression*). Fenomena ko-ekspresi ini mengindikasikan adanya keterikatan jalur fungsional biokimia tertentu yang aktif bekerja secara sistemik pada klaster subtipe tumor tertentu [19].

2.5.4 Fenomena Ketidakseimbangan Kelas (*Class Imbalance*)

Kuantifikasi terhadap distribusi proporsi label target merupakan aspek krusial dalam EDA untuk mendeteksi keberadaan fenomena ketidakseimbangan kelas (*class imbalance*) [22]. Dalam ranah onkologi medis, ketimpangan populasi sampel antara subtipe tumor langka (seperti status MSI-H) dan subtipe umum (Non-MSI-H) merupakan manifestasi klinis riil yang sangat lumrah ditemukan [19]. Namun, dari sudut pandang komputasi, ketimpangan jumlah sampel yang ekstrem dapat membiaskan fungsi objektif dari pengklasifikasi standar, di mana model cenderung

mengoptimalkan prediksi pada kelas mayoritas demi mengamankan nilai akurasi global [17]. Fenomena ini memicu lahirnya akurasi semu (accuracy paradox), sehingga identifikasi awal terhadap ketimpangan kelas ini menjadi landasan metodologis wajib untuk mengintegrasikan algoritma penyeimbangan data sintetis pada pipa pemodelan [23].

2.6 *Data Preprocessing*

Tahap *data preprocessing* merupakan salah satu komponen penting dalam pipeline machine learning, khususnya pada data *gene expression* yang memiliki karakteristik berdimensi tinggi dan rentan terhadap noise. Proses ini bertujuan untuk meningkatkan kualitas data sebelum dilakukan proses pemodelan, sehingga model yang dihasilkan menjadi lebih stabil, efisien, dan memiliki kemampuan generalisasi yang lebih baik.

2.6.1 *Variance Filtering*

Dalam tahap awal *data preprocessing*, salah satu permasalahan utama pada data *gene expression* adalah tingginya jumlah fitur (gen) yang tidak semuanya memberikan informasi yang relevan terhadap proses klasifikasi. Banyak gen memiliki nilai ekspresi yang hampir sama pada seluruh sampel sehingga tidak mampu membedakan kelas yang akan diprediksi. Oleh karena itu, dilakukan reduksi dimensi menggunakan metode *Variance Filtering*, yaitu teknik seleksi fitur yang menghapus fitur berdasarkan tingkat variasi nilainya. Metode ini termasuk ke dalam kelompok *filter feature selection*, yaitu teknik seleksi fitur yang bekerja secara independen tanpa melibatkan algoritma pembelajaran (*learning algorithm*), sehingga sangat efisien diterapkan pada dataset berdimensi tinggi seperti data ekspresi gen [27, 28].

Variansi suatu fitur dihitung menggunakan Persamaan 2.7.

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2.7}$$

dengan:

- x_i : nilai observasi ke-i
- $\bar{x}$: rata-rata seluruh nilai fitur
- n : jumlah sampel

Nilai variansi menunjukkan seberapa besar penyebaran data terhadap nilai rata-ratanya. Semakin kecil nilai variansi, semakin sedikit perubahan nilai fitur antar sampel sehingga fitur tersebut dianggap kurang informatif. Sebaliknya, fitur yang memiliki variansi lebih tinggi cenderung mengandung informasi yang lebih berguna dalam membedakan kelas pada proses klasifikasi [29].

Proses Algoritma Variance Filtering

Secara umum, algoritma *Variance Filtering* bekerja melalui tahapan berikut.

1. Menghitung nilai rata-rata setiap fitur.

2. Menghitung nilai variansi masing-masing fitur menggunakan Persamaan 2.7.
3. Membandingkan nilai variansi setiap fitur dengan nilai ambang batas (*threshold*) yang telah ditentukan.
4. Menghapus fitur yang memiliki nilai variansi lebih kecil dari *threshold*.
5. Mempertahankan fitur yang memiliki nilai variansi lebih besar dari *threshold* sebagai fitur yang digunakan pada proses analisis selanjutnya.

Melalui proses tersebut, fitur yang hampir konstan pada seluruh sampel dapat dieliminasi sehingga hanya fitur yang memiliki variasi ekspresi yang cukup besar yang digunakan pada tahap pemodelan. Pendekatan ini mampu mengurangi kompleksitas data tanpa memerlukan informasi mengenai label kelas sehingga proses komputasi menjadi lebih efisien [28].

Kelebihan dan Kekurangan

Kelebihan metode *Variance Filtering* adalah sebagai berikut.

- Sederhana dan cepat diterapkan pada dataset berdimensi tinggi.
- Mengurangi jumlah fitur secara signifikan.
- Mempercepat proses komputasi model.
- Mengurangi risiko *overfitting*.

Adapun kekurangan metode ini adalah sebagai berikut.

- Tidak mempertimbangkan hubungan antara fitur dengan label kelas.
- Berpotensi menghapus fitur yang memiliki variansi rendah tetapi sebenarnya penting untuk proses klasifikasi.

Alasan Digunakan pada Penelitian

Pada penelitian ini, *Variance Filtering* digunakan sebagai tahap seleksi fitur awal untuk menghilangkan gen-gen yang memiliki tingkat variasi ekspresi sangat rendah. Langkah ini bertujuan mengurangi dimensi data, mempercepat proses komputasi, serta menghasilkan dataset yang lebih efisien sebelum dilakukan proses seleksi fitur menggunakan LIMMA, LASSO, ANOVA, maupun RFE. Penggunaan metode ini sesuai untuk data ekspresi gen yang memiliki puluhan ribu fitur sehingga proses analisis selanjutnya menjadi lebih efektif [28].

2.6.2 *Data Scaling*

Setelah dilakukan reduksi dimensi menggunakan *Variance Filtering*, tahap berikutnya adalah melakukan *Data Scaling*. Proses ini bertujuan menyeragamkan skala nilai setiap fitur sehingga seluruh fitur memiliki kontribusi yang seimbang dalam proses pembelajaran model.

Tahapan ini merupakan salah satu proses *preprocessing* yang penting karena banyak algoritma *machine learning* sensitif terhadap perbedaan skala antar fitur [29, 30].

Pada data *gene expression*, setiap gen dapat memiliki rentang nilai ekspresi yang berbeda. Apabila tidak dilakukan normalisasi skala, fitur dengan nilai yang lebih besar dapat mendominasi proses pembelajaran terutama pada algoritma yang menggunakan perhitungan jarak maupun proses optimisasi.

Salah satu metode yang umum digunakan adalah standardisasi (*Z-score Standardization*) yang dihitung menggunakan Persamaan 2.8.

$$Z = \frac{x - \mu}{\sigma} \tag{2.8}$$

dengan:

- x : nilai asli
- μ : rata-rata fitur
- σ : standar deviasi fitur

Transformasi ini menghasilkan data yang memiliki rata-rata mendekati nol dan standar deviasi sebesar satu. Standardisasi banyak digunakan karena mampu mengurangi pengaruh perbedaan skala antar fitur sehingga algoritma berbasis optimisasi maupun perhitungan jarak, seperti Support Vector Machine (SVM), Logistic Regression, dan K-Nearest Neighbor (KNN), dapat bekerja secara lebih optimal [30].

Selain standardisasi, metode lain yang banyak digunakan adalah Min-Max Scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.9}$$

dengan:

- X : nilai asli
- X_{min} : nilai minimum fitur
- X_{max} : nilai maksimum fitur

Transformasi ini mengubah rentang nilai setiap fitur menjadi interval 0 sampai 1 tanpa mengubah bentuk distribusi data. Min-Max Scaling banyak diterapkan pada algoritma berbasis jaringan saraf tiruan (*Artificial Neural Network*) karena mampu mempercepat proses konvergensi selama pelatihan model [31].

Proses Algoritma Data Scaling

Secara umum, proses standardisasi dilakukan melalui tahapan berikut.

1. Menghitung nilai rata-rata setiap fitur.
2. Menghitung standar deviasi setiap fitur.
3. Mengurangi setiap nilai fitur dengan rata-ratanya.

4. Membagi hasil pengurangan tersebut dengan standar deviasi menggunakan Persamaan 2.8.
5. Menghasilkan dataset baru yang memiliki rata-rata nol dan standar deviasi satu.

Sedangkan tahapan Min-Max Scaling dilakukan sebagai berikut.

1. Menentukan nilai minimum setiap fitur.
2. Menentukan nilai maksimum setiap fitur.
3. Mengurangi setiap nilai fitur dengan nilai minimum.
4. Membagi hasil pengurangan tersebut dengan selisih nilai maksimum dan minimum menggunakan Persamaan 2.9.
5. Menghasilkan dataset baru dengan rentang nilai antara 0 hingga 1.

Kelebihan dan Kekurangan

Kelebihan *Data Scaling* adalah sebagai berikut.

- Menyamakan skala seluruh fitur.
- Mempercepat proses optimisasi model.
- Meningkatkan stabilitas numerik selama proses pelatihan.
- Sangat penting pada algoritma berbasis jarak seperti SVM dan KNN.

Adapun kekurangannya adalah sebagai berikut.

- Standardisasi sensitif terhadap nilai pencilan (*outlier*).
- Min-Max Scaling sangat dipengaruhi oleh nilai minimum dan maksimum sehingga perubahan pada data baru dapat memengaruhi hasil transformasi.

Alasan Digunakan pada Penelitian

Pada penelitian ini, proses *Data Scaling* dilakukan untuk menyeragamkan rentang nilai ekspresi gen sebelum dilakukan proses pelatihan model *machine learning*. Tahapan ini bertujuan agar setiap gen memiliki kontribusi yang seimbang selama proses pembelajaran sehingga algoritma seperti Support Vector Machine (SVM), Logistic Regression, maupun metode seleksi fitur dapat bekerja secara optimal dan menghasilkan performa klasifikasi yang lebih baik [30, 29].

2.6.3 *Data Splitting*

Setelah proses *data preprocessing* selesai dilakukan, tahap berikutnya adalah membagi dataset menjadi data pelatihan (*training set*) dan data pengujian (*testing set*). Tujuan utama pembagian data adalah untuk mengevaluasi kemampuan model dalam melakukan prediksi terhadap data baru yang belum pernah digunakan selama proses pelatihan. Dengan demikian, performa model

yang diperoleh dapat menggambarkan kemampuan generalisasi (*generalization*) model secara lebih objektif [32, 29].

Pada penelitian klasifikasi, proses pembagian data perlu dilakukan secara hati-hati, terutama ketika distribusi kelas tidak seimbang. Apabila pembagian dilakukan secara acak tanpa mempertimbangkan proporsi setiap kelas, maka terdapat kemungkinan salah satu kelas memiliki jumlah sampel yang jauh lebih sedikit pada data pelatihan maupun data pengujian. Kondisi tersebut dapat menyebabkan model mengalami bias terhadap kelas mayoritas dan menghasilkan evaluasi yang kurang representatif [33].

Untuk mengatasi permasalahan tersebut digunakan metode *Stratified Sampling*, yaitu teknik pembagian data yang mempertahankan proporsi setiap kelas pada data pelatihan maupun data pengujian agar tetap sama dengan distribusi kelas pada dataset asli. Pendekatan ini banyak digunakan pada dataset yang memiliki distribusi kelas tidak seimbang karena mampu menghasilkan proses pelatihan dan evaluasi yang lebih adil [33].

Proses Algoritma Data Splitting

Secara umum, proses pembagian data dilakukan melalui tahapan berikut.

1. Menentukan proporsi pembagian data pelatihan dan data pengujian.
2. Mengidentifikasi distribusi setiap kelas pada dataset.
3. Membagi data menggunakan metode *Stratified Sampling* sehingga proporsi setiap kelas tetap terjaga.
4. Menyimpan data pelatihan untuk proses pembangunan model.
5. Menyimpan data pengujian sebagai data yang tidak digunakan selama pelatihan dan hanya digunakan untuk evaluasi akhir model.

Melalui proses tersebut, distribusi kelas pada data pelatihan dan data pengujian tetap konsisten sehingga hasil evaluasi model menjadi lebih representatif terhadap data sebenarnya [33].

Stratified K-Fold Cross Validation

Selain melakukan pembagian data menjadi *training set* dan *testing set*, penelitian ini juga menggunakan metode *Stratified K-Fold Cross Validation*. Metode ini merupakan teknik validasi yang membagi data pelatihan menjadi K lipatan (*fold*) dengan tetap mempertahankan proporsi setiap kelas pada masing-masing lipatan. Dengan demikian, distribusi kelas MSI-H dan Non-MSI-H pada setiap *fold* tetap seimbang sehingga proses pelatihan dan validasi menjadi lebih representatif [34, 29].

Pada setiap iterasi, satu *fold* digunakan sebagai data validasi, sedangkan $K - 1$ *fold* lainnya digunakan sebagai data pelatihan. Proses ini dilakukan sebanyak K kali sehingga setiap *fold* memperoleh kesempatan menjadi data validasi tepat satu kali. Nilai performa akhir diperoleh dari rata-rata hasil evaluasi pada seluruh iterasi. Pendekatan ini mampu memberikan estimasi performa model yang lebih stabil dibandingkan hanya menggunakan satu kali pembagian data [34].

Pada penelitian ini digunakan nilai $K = 3$, sehingga data pelatihan dibagi menjadi tiga *fold*. Pada iterasi pertama, *fold* pertama digunakan sebagai data validasi dan dua *fold* lainnya digunakan sebagai data pelatihan. Iterasi berikutnya dilakukan dengan mengganti *fold* validasi hingga seluruh *fold* pernah digunakan sebagai data validasi sebanyak satu kali. Pendekatan ini memungkinkan seluruh data pelatihan dimanfaatkan secara bergantian untuk proses validasi sehingga estimasi performa model menjadi lebih stabil dibandingkan hanya menggunakan satu kali pembagian data [32].

Pembagian Data 80:20 dan 70:30

Dalam pengembangan model *machine learning*, dua rasio pembagian data yang paling umum digunakan adalah 80:20 dan 70:30.

Pembagian data sebesar 80:20 berarti sebanyak 80% data digunakan sebagai data pelatihan, sedangkan 20% sisanya digunakan sebagai data pengujian. Rasio ini memberikan jumlah data pelatihan yang lebih besar sehingga model memiliki kesempatan mempelajari pola data secara lebih optimal. Oleh karena itu, pembagian 80:20 banyak digunakan pada dataset yang memiliki jumlah sampel relatif terbatas karena mampu memberikan keseimbangan antara proses pembelajaran dan evaluasi model [33, 32].

Sementara itu, pembagian data sebesar 70:30 menggunakan 70% data sebagai data pelatihan dan 30% sebagai data pengujian. Rasio ini memberikan jumlah data pengujian yang lebih besar sehingga evaluasi performa model dilakukan pada jumlah sampel yang lebih banyak. Namun demikian, jumlah data pelatihan menjadi lebih sedikit dibandingkan pembagian 80:20 sehingga informasi yang dipelajari model juga berkurang [29].

Secara umum, pembagian 80:20 lebih sering dipilih karena mampu memberikan keseimbangan antara jumlah data pelatihan yang cukup besar dan jumlah data pengujian yang masih memadai untuk mengevaluasi kemampuan generalisasi model [32].

Kelebihan dan Kekurangan

Kelebihan penggunaan *Stratified Data Splitting* dan *Stratified K-Fold Cross Validation* adalah sebagai berikut.

- Mempertahankan distribusi kelas pada setiap pembagian data.
- Mengurangi bias akibat ketidakseimbangan kelas.
- Menghasilkan estimasi performa model yang lebih stabil.
- Memanfaatkan seluruh data pelatihan sebagai data validasi secara bergantian.
- Mengurangi risiko *overfitting* terhadap satu pembagian data tertentu.

Adapun kekurangannya adalah sebagai berikut.

- Waktu komputasi lebih lama dibandingkan satu kali pembagian data.
- Membutuhkan proses pelatihan model sebanyak jumlah *fold* yang digunakan.

- Tidak cocok digunakan apabila ukuran dataset sangat besar dan waktu komputasi menjadi kendala.

Alasan Digunakan pada Penelitian

Pada penelitian ini digunakan pembagian data sebesar 80% sebagai data pelatihan dan 20% sebagai data pengujian menggunakan metode *Stratified Sampling*. Pembagian ini dipilih karena mampu menyediakan jumlah data pelatihan yang cukup besar untuk membangun model, sementara data pengujian tetap disisihkan sebagai data independen untuk mengevaluasi kemampuan generalisasi model [33].

Selanjutnya, pada data pelatihan diterapkan *Stratified 3-Fold Cross Validation*. Pemilihan tiga *fold* dilakukan untuk memperoleh estimasi performa model yang lebih stabil dibandingkan satu kali pembagian data, sekaligus menjaga efisiensi waktu komputasi. Selain itu, penggunaan metode stratifikasi memastikan bahwa proporsi kelas MSI-H dan Non-MSI-H tetap terjaga pada setiap *fold*, sehingga proses pelatihan dan validasi dapat dilakukan secara lebih representatif dan mengurangi bias akibat ketidakseimbangan distribusi kelas [34, 32].

2.7 *Data Balancing*

Ketidakseimbangan data (*class imbalance*) merupakan salah satu permasalahan yang sering dijumpai dalam tugas klasifikasi, terutama pada bidang medis dan bioinformatika, di mana jumlah sampel antar kelas tidak seimbang. Kondisi ini dapat menyebabkan model pembelajaran lebih banyak mempelajari karakteristik kelas mayoritas sehingga performa prediksi terhadap kelas minoritas menjadi rendah. Akibatnya, model dapat menghasilkan nilai akurasi yang tinggi tetapi memiliki kemampuan yang buruk dalam mengidentifikasi sampel dari kelas minoritas [35, 36].

Dalam penelitian klasifikasi status MSI-H dan Non-MSI-H, distribusi data umumnya didominasi oleh kelas Non-MSI-H sehingga diperlukan teknik penyeimbangan data sebelum proses pelatihan model dilakukan. Salah satu metode yang paling banyak digunakan adalah *Synthetic Minority Over-sampling Technique* (SMOTE) yang diperkenalkan oleh Chawla *et al.* [37]. Berbeda dengan metode *random oversampling* yang hanya menduplikasi sampel pada kelas minoritas, SMOTE menghasilkan sampel sintetis baru melalui proses interpolasi antar sampel minoritas sehingga distribusi data menjadi lebih beragam.

Proses pembentukan sampel sintetis pada SMOTE dihitung menggunakan Persamaan 2.10.

$$x_{new} = x_i + \delta(x_{nn} - x_i) \tag{2.10}$$

dengan:

- x_i : sampel pada kelas minoritas.
- x_{nn} : salah satu tetangga terdekat (*nearest neighbor*) dari x_i.
- δ : bilangan acak dengan nilai $0 \leq \delta \leq 1$.

Persamaan tersebut menunjukkan bahwa sampel sintetis dibentuk pada suatu titik di antara dua sampel kelas minoritas. Nilai δ menentukan posisi titik baru pada garis yang menghubungkan

kedua sampel tersebut. Dengan demikian, data sintetis yang dihasilkan masih berada pada distribusi ruang fitur kelas minoritas sehingga lebih representatif dibandingkan sekadar melakukan duplikasi data [37].

Proses Algoritma SMOTE

Secara umum, algoritma SMOTE bekerja melalui tahapan berikut.

1. Mengidentifikasi seluruh sampel yang termasuk ke dalam kelas minoritas.
2. Menentukan sejumlah tetangga terdekat (*k-nearest neighbors*) untuk setiap sampel minoritas menggunakan perhitungan jarak, umumnya Euclidean Distance.
3. Memilih salah satu tetangga terdekat secara acak.
4. Menghasilkan sampel sintetis menggunakan Persamaan 2.10 dengan melakukan interpolasi antara sampel asli dan tetangga terdekat yang dipilih.
5. Mengulangi proses hingga jumlah sampel kelas minoritas mencapai distribusi yang diinginkan.

Melalui proses tersebut, SMOTE menghasilkan sampel baru yang tersebar di sepanjang ruang fitur kelas minoritas sehingga memperluas wilayah representasi kelas tersebut. Pendekatan ini membantu model mempelajari karakteristik kelas minoritas secara lebih baik dibandingkan metode oversampling konvensional yang hanya menduplikasi data [37, 35].

Kelebihan dan Kekurangan

Kelebihan metode SMOTE adalah sebagai berikut.

- Mengurangi ketidakseimbangan distribusi kelas.
- Menghasilkan sampel sintetis sehingga mengurangi risiko *overfitting* akibat duplikasi data.
- Meningkatkan kemampuan model dalam mengenali kelas minoritas.
- Meningkatkan metrik evaluasi seperti Recall, F1-Score, dan ROC-AUC pada dataset yang tidak seimbang.

Adapun kekurangan metode SMOTE adalah sebagai berikut.

- Berpotensi menghasilkan sampel sintetis yang berada pada area tumpang tindih (*overlap*) antar kelas.
- Sensitif terhadap keberadaan *outlier* pada kelas minoritas.
- Dapat meningkatkan risiko *overfitting* apabila digunakan secara berlebihan atau tanpa dikombinasikan dengan proses seleksi fitur.

Alasan Digunakan pada Penelitian

Pada penelitian ini, SMOTE digunakan untuk mengatasi ketidakseimbangan jumlah sampel antara kelas MSI-H dan Non-MSI-H. Proses oversampling dilakukan hanya pada data pelatihan (*training set*) di setiap iterasi *Stratified Cross Validation*, sedangkan data validasi dan data pengujian tidak mengalami proses oversampling untuk menghindari terjadinya *data leakage*. Penggunaan SMOTE bertujuan agar model memperoleh jumlah sampel yang lebih seimbang selama proses pembelajaran sehingga mampu mengenali karakteristik kelas MSI-H secara lebih baik dan menghasilkan performa klasifikasi yang lebih optimal, khususnya pada metrik Recall dan F1-Score [37, 35, 36].

2.8 *Feature Selection*

Feature selection merupakan salah satu tahap penting dalam analisis data *gene expression*, khususnya pada data berdimensi tinggi seperti data genomik. Tujuan utama dari proses ini adalah untuk memilih subset fitur (gen) yang paling relevan terhadap variabel target, sehingga dapat meningkatkan performa model, mengurangi kompleksitas komputasi, serta meningkatkan interpretabilitas hasil analisis [38]. Selain itu, feature selection juga berperan dalam mengurangi noise dan menghindari fenomena *curse of dimensionality* yang umum terjadi pada data berdimensi tinggi.

2.8.1 *Differential Gene Expression* (LIMMA)

Salah satu pendekatan statistik yang umum digunakan dalam feature selection adalah *Differential Gene Expression* (DGE) menggunakan metode LIMMA (*Linear Models for Microarray Data*) [26]. Metode ini berbasis pada model regresi linear yang digunakan untuk mengidentifikasi gen yang menunjukkan perbedaan ekspresi signifikan antara dua atau lebih kondisi biologis.

$$Y = X\beta + \varepsilon \tag{2.11}$$

Pada persamaan tersebut, Y merepresentasikan nilai ekspresi gen yang diamati, X adalah matriks desain yang menggambarkan kondisi eksperimen, β adalah parameter koefisien yang menunjukkan kontribusi setiap variabel, dan ε merupakan error atau noise yang tidak dapat dijelaskan oleh model.

Dalam konteks analisis ekspresi gen, LIMMA bekerja dengan mengestimasi perbedaan ekspresi antar kelompok (misalnya normal dan kanker) melalui pendekatan *empirical Bayes*, sehingga menghasilkan estimasi yang lebih stabil meskipun ukuran sampel relatif kecil. Gen yang memiliki nilai perbedaan ekspresi signifikan kemudian dipilih sebagai kandidat fitur penting atau biomarker potensial [26]. Metode ini sangat efektif dalam studi bioinformatika karena mampu menggabungkan kekuatan model linear dengan stabilisasi varians antar gen

Proses Algoritma LIMMA

Secara umum, algoritma LIMMA bekerja melalui tahapan berikut.

1. Menyusun matriks desain (*design matrix*) yang merepresentasikan kelompok sampel yang akan dibandingkan.
2. Membangun model regresi linear untuk setiap gen menggunakan Persamaan 2.11.
3. Mengestimasi variansi setiap gen.
4. Menerapkan pendekatan *Empirical Bayes* untuk menstabilkan estimasi variansi antar gen.
5. Menghitung statistik uji dan *p-value* setiap gen.
6. Melakukan koreksi *multiple testing* menggunakan metode Benjamini-Hochberg untuk memperoleh nilai *Adjusted p-value*.
7. Memilih gen yang memiliki nilai *Adjusted p-value* di bawah nilai signifikansi yang telah ditentukan sebagai *Differentially Expressed Genes* (DEGs).

Pendekatan *Empirical Bayes* memungkinkan LIMMA menghasilkan estimasi variansi yang lebih stabil dibandingkan metode statistik konvensional, terutama ketika jumlah sampel relatif sedikit tetapi jumlah gen sangat besar [26, 39].

Kelebihan dan Kekurangan

Kelebihan LIMMA adalah sebagai berikut.

- Sangat efektif untuk data berdimensi tinggi.
- Memiliki performa baik pada jumlah sampel yang relatif kecil.
- Menghasilkan estimasi variansi yang stabil melalui pendekatan *Empirical Bayes*.
- Banyak digunakan sebagai standar analisis ekspresi gen.

Adapun kekurangannya adalah sebagai berikut.

- Mengasumsikan hubungan linear antar variabel.
- Tidak mempertimbangkan hubungan non-linear antar gen.
- Sensitif terhadap kualitas preprocessing data.

Alasan Digunakan pada Penelitian

Pada penelitian ini, LIMMA digunakan sebagai tahap awal seleksi fitur untuk mengidentifikasi gen-gen yang memiliki perbedaan ekspresi signifikan antara kelompok MSI-H dan Non-MSI-H. Gen yang memenuhi kriteria signifikansi selanjutnya digunakan sebagai kandidat fitur pada proses seleksi berikutnya menggunakan LASSO, ANOVA, dan RFE. Penggunaan LIMMA dipilih karena mampu menghasilkan analisis ekspresi gen yang stabil meskipun jumlah sampel relatif sedikit dibandingkan jumlah fitur [26].

2.8.2 *LASSO*

Selain pendekatan statistik, metode berbasis machine learning juga banyak digunakan dalam feature selection, salah satunya adalah LASSO (*Least Absolute Shrinkage and Selection Operator*) [40]. LASSO merupakan metode regresi yang menambahkan regularisasi L1 untuk melakukan seleksi fitur secara otomatis.

$$\min_{\beta} \left(\sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right) \tag{2.12}$$

Pada persamaan tersebut, bagian pertama merepresentasikan error kuadrat antara nilai prediksi dan nilai aktual, sedangkan bagian kedua merupakan penalti L1 yang dikendalikan oleh parameter λ. Penalti ini mendorong beberapa koefisien β_j menjadi nol, sehingga fitur yang tidak relevan secara otomatis dieliminasi dari model.

Keunggulan utama LASSO adalah kemampuannya dalam melakukan seleksi fitur dan regularisasi secara simultan. Hal ini tidak hanya mengurangi kompleksitas model, tetapi juga meningkatkan generalisasi serta mengurangi risiko *overfitting*, terutama pada data dengan jumlah fitur yang jauh lebih besar dibanding jumlah sampel.

Proses Algoritma LASSO

Secara umum, algoritma LASSO bekerja melalui tahapan berikut.

1. Menginisialisasi seluruh koefisien model.
2. Menghitung nilai prediksi menggunakan model regresi.
3. Menghitung fungsi objektif yang terdiri atas error prediksi dan penalti L1 pada Persamaan 2.12.
4. Melakukan optimisasi untuk memperoleh nilai koefisien yang meminimalkan fungsi objektif.
5. Mengurangi nilai koefisien yang kurang penting hingga bernilai nol.
6. Mempertahankan fitur yang memiliki koefisien tidak sama dengan nol sebagai fitur terpilih.

Melalui proses regularisasi tersebut, LASSO mampu melakukan seleksi fitur dan pelatihan model secara bersamaan sehingga menghasilkan model yang lebih sederhana namun tetap memiliki performa yang baik [40, 29].

Kelebihan dan Kekurangan

Kelebihan LASSO adalah sebagai berikut.

- Melakukan seleksi fitur dan regularisasi secara simultan.
- Mengurangi risiko overfitting.
- Menghasilkan model yang lebih sederhana.

- Sangat sesuai untuk data berdimensi tinggi.

Adapun kekurangannya adalah sebagai berikut.

- Sensitif terhadap fitur yang saling berkorelasi tinggi.
- Pemilihan parameter regularisasi λ sangat memengaruhi hasil seleksi fitur.

Alasan Digunakan pada Penelitian

Pada penelitian ini, LASSO digunakan untuk memilih gen-gen yang paling berkontribusi terhadap proses klasifikasi MSI-H dan Non-MSI-H. Kemampuan LASSO dalam menghilangkan fitur yang kurang relevan melalui regularisasi L1 sangat sesuai untuk data ekspresi gen yang memiliki jumlah fitur jauh lebih banyak dibandingkan jumlah sampel [40].

2.8.3 *Analysis of Variance* (ANOVA)

Metode lain yang sering digunakan dalam feature selection adalah ANOVA (*Analysis of Variance*) [41]. ANOVA digunakan untuk menguji apakah terdapat perbedaan rata-rata yang signifikan antara dua atau lebih kelompok data.

$$F = \frac{\text{Variansi antar kelompok}}{\text{Variansi dalam kelompok}} \tag{2.13}$$

Nilai statistik F yang tinggi menunjukkan bahwa variasi antar kelompok lebih besar dibandingkan variasi dalam kelompok. Hal ini mengindikasikan bahwa fitur tersebut memiliki kemampuan diskriminatif yang baik dalam membedakan kelas.

Dalam konteks data *gene expression*, ANOVA digunakan untuk mengidentifikasi gen yang menunjukkan perbedaan ekspresi signifikan antar kondisi, seperti jaringan normal dan kanker. Dengan demikian, ANOVA dapat digunakan sebagai metode seleksi fitur berbasis uji statistik untuk menyaring gen-gen yang paling relevan terhadap variabel target.

Proses Algoritma ANOVA

Secara umum, algoritma ANOVA bekerja melalui tahapan berikut.

1. Mengelompokkan data berdasarkan kelas target.
2. Menghitung rata-rata setiap kelompok.
3. Menghitung variasi antar kelompok (*between-group variance*).
4. Menghitung variasi di dalam kelompok (*within-group variance*).
5. Menghitung nilai statistik F menggunakan Persamaan 2.13.
6. Memilih fitur yang memiliki nilai statistik F terbesar sebagai fitur yang paling mampu membedakan kelas.

Semakin besar nilai statistik F, semakin besar pula kemampuan fitur dalam membedakan kelompok data [41, 32].

Kelebihan dan Kekurangan

Kelebihan ANOVA adalah sebagai berikut.

- Proses komputasi cepat.
- Mudah diimplementasikan.
- Efektif untuk memilih fitur yang memiliki perbedaan rata-rata signifikan.

Adapun kekurangannya adalah sebagai berikut.

- Hanya mempertimbangkan hubungan satu fitur terhadap label.
- Tidak mempertimbangkan interaksi antar fitur.
- Mengasumsikan distribusi data tertentu.

Alasan Digunakan pada Penelitian

Pada penelitian ini, ANOVA digunakan sebagai metode filter untuk memilih gen yang memiliki perbedaan ekspresi paling signifikan antara kelas MSI-H dan Non-MSI-H berdasarkan nilai statistik F. Metode ini dipilih karena memiliki proses komputasi yang cepat sehingga sesuai digunakan pada data ekspresi gen berdimensi tinggi [32].

2.8.4 *Recursive Feature Elimination* (RFE)

Recursive Feature Elimination (RFE) merupakan metode feature selection berbasis model yang bekerja dengan cara melakukan eliminasi fitur secara iteratif berdasarkan tingkat kepentingannya [42]. Metode ini termasuk dalam kategori *wrapper method* karena menggunakan model pembelajaran mesin untuk mengevaluasi kontribusi setiap fitur.

Secara umum, proses RFE dilakukan melalui langkah-langkah berikut:

1. Melatih model menggunakan seluruh fitur yang tersedia
2. Menghitung tingkat kepentingan setiap fitur berdasarkan parameter model (misalnya koefisien atau feature importance)
3. Menghapus fitur dengan kontribusi paling kecil terhadap performa model
4. Mengulangi proses secara iteratif hingga jumlah fitur yang diinginkan tercapai

RFE sering dikombinasikan dengan algoritma seperti Support Vector Machine (SVM) atau regresi linear untuk menghasilkan subset fitur yang optimal. Keunggulan utama metode ini adalah kemampuannya dalam mempertimbangkan interaksi antar fitur secara tidak langsung melalui proses pelatihan model berulang.

Dalam konteks data *gene expression*, RFE sangat efektif untuk mengurangi dimensi data yang sangat tinggi sekaligus mempertahankan fitur-fitur yang paling berkontribusi terhadap klasifikasi. Hal ini menjadikannya salah satu metode yang sangat cocok untuk aplikasi bioinformatika dan analisis kanker berbasis data genomik.

Kelebihan dan Kekurangan

Kelebihan metode RFE adalah sebagai berikut.

- Mempertimbangkan kontribusi fitur terhadap performa model.
- Menghasilkan subset fitur yang optimal.
- Dapat digunakan pada berbagai algoritma machine learning.
- Mempertimbangkan interaksi antar fitur melalui proses pelatihan model.

Adapun kekurangannya adalah sebagai berikut.

- Membutuhkan waktu komputasi yang relatif tinggi.
- Hasil seleksi sangat bergantung pada model yang digunakan.
- Kurang efisien apabila jumlah fitur sangat besar.

Alasan Digunakan pada Penelitian

Pada penelitian ini, RFE digunakan untuk memperoleh subset gen yang paling optimal berdasarkan kontribusinya terhadap performa model klasifikasi. Berbeda dengan metode filter, RFE melakukan seleksi fitur menggunakan proses pelatihan model secara iteratif sehingga mampu mempertimbangkan hubungan antar fitur selama proses seleksi. Oleh karena itu, RFE diharapkan menghasilkan subset fitur yang lebih representatif untuk membedakan kelas MSI-H dan Non-MSI-H [42].

2.9 *Hyperparameter Tuning*

Hyperparameter tuning merupakan proses optimasi nilai hyperparameter yang ditentukan sebelum proses pelatihan model dilakukan. Berbeda dengan parameter model yang dipelajari secara otomatis selama proses pembelajaran, hyperparameter harus ditentukan terlebih dahulu oleh peneliti dan memiliki pengaruh yang besar terhadap performa model. Pemilihan hyperparameter yang kurang tepat dapat menyebabkan model mengalami underfitting maupun overfitting sehingga kemampuan generalisasi model menjadi kurang optimal [43, 33].

Salah satu metode yang paling banyak digunakan untuk melakukan optimasi hyperparameter adalah *Grid Search*. Metode ini bekerja dengan mengevaluasi seluruh kombinasi hyperparameter yang telah ditentukan pada ruang pencarian (*search space*). Setiap kombinasi hyperparameter digunakan untuk melatih model, kemudian performanya dievaluasi menggunakan metode validasi silang (*cross validation*). Kombinasi hyperparameter yang menghasilkan performa terbaik dipilih sebagai hyperparameter optimal [29, 32].

Secara matematis, proses optimasi hyperparameter menggunakan Grid Search dapat dinyatakan pada Persamaan 2.14.

$$\theta^* = \arg\max_{\theta \in \Theta} Score(\theta) \tag{2.14}$$

dengan:

- Θ : himpunan seluruh kombinasi hyperparameter yang diuji.
- θ : satu kombinasi hyperparameter.
- $Score(\theta)$: nilai evaluasi model menggunakan kombinasi hyperparameter θ.
- θ^* : kombinasi hyperparameter terbaik.

Persamaan tersebut menunjukkan bahwa Grid Search akan mencari kombinasi hyperparameter yang menghasilkan nilai evaluasi tertinggi di antara seluruh kombinasi yang tersedia pada ruang pencarian.

Untuk memperoleh hasil evaluasi yang lebih stabil, Grid Search umumnya dikombinasikan dengan metode *k-Fold Cross Validation*. Pada metode ini, data pelatihan dibagi menjadi k bagian (*fold*) yang memiliki ukuran hampir sama. Setiap fold digunakan secara bergantian sebagai data validasi, sedangkan fold lainnya digunakan sebagai data pelatihan. Nilai performa akhir diperoleh dari rata-rata seluruh hasil validasi [34].

Rata-rata nilai evaluasi dihitung menggunakan Persamaan 2.15.

$$CV_{score} = \frac{1}{k} \sum_{i=1}^{k} Score_i \tag{2.15}$$

dengan:

- $Score_i$: nilai evaluasi pada fold ke-i.
- k : jumlah fold yang digunakan.
- CV_{score} : rata-rata performa seluruh fold.

Nilai rata-rata tersebut digunakan sebagai dasar pemilihan kombinasi hyperparameter terbaik. Semakin tinggi nilai rata-rata evaluasi yang diperoleh, semakin baik kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya.

Proses Algoritma Grid Search

Secara umum, algoritma Grid Search bekerja melalui tahapan berikut.

1. Menentukan hyperparameter yang akan dioptimasi beserta rentang nilainya.
2. Membentuk seluruh kombinasi hyperparameter dari ruang pencarian (*search space*).
3. Melatih model menggunakan satu kombinasi hyperparameter.
4. Mengevaluasi performa model menggunakan *Cross Validation*.
5. Mengulangi proses hingga seluruh kombinasi hyperparameter selesai dievaluasi.
6. Membandingkan seluruh nilai evaluasi yang diperoleh.
7. Memilih kombinasi hyperparameter dengan nilai evaluasi tertinggi sebagai model terbaik.

Melalui proses tersebut, seluruh kombinasi hyperparameter dievaluasi secara sistematis sehingga diperoleh konfigurasi model yang menghasilkan performa terbaik pada data pelatihan [33].

Kelebihan dan Kekurangan

Kelebihan metode Grid Search adalah sebagai berikut.

- Mampu mengevaluasi seluruh kombinasi hyperparameter secara sistematis.
- Mudah diimplementasikan menggunakan berbagai pustaka machine learning.
- Memberikan hasil optimasi yang optimal apabila ruang pencarian tidak terlalu besar.
- Mengurangi subjektivitas dalam pemilihan hyperparameter.

Adapun kekurangan metode Grid Search adalah sebagai berikut.

- Membutuhkan waktu komputasi yang cukup lama apabila jumlah kombinasi hyperparameter sangat banyak.
- Seluruh kombinasi harus dievaluasi meskipun sebagian besar menghasilkan performa yang buruk.
- Kurang efisien pada model yang memiliki banyak hyperparameter.

Alasan Digunakan pada Penelitian

Pada penelitian ini, Grid Search digunakan untuk memperoleh kombinasi hyperparameter terbaik pada algoritma Support Vector Machine (SVM), Random Forest, dan Extreme Gradient Boosting (XGBoost). Proses optimasi dilakukan menggunakan *Stratified 3-Fold Cross Validation* sehingga setiap kombinasi hyperparameter dievaluasi secara konsisten pada distribusi kelas yang seimbang.

Metrik evaluasi yang digunakan selama proses optimasi adalah *F1-Score* karena dataset yang digunakan memiliki distribusi kelas yang tidak seimbang antara MSI-H dan Non-MSI-H. Pemilihan F1-Score bertujuan memperoleh model yang memiliki keseimbangan antara Precision dan Recall sehingga mampu mengklasifikasikan kelas minoritas dengan lebih baik. Melalui proses optimasi ini diharapkan model yang dihasilkan memiliki kemampuan generalisasi yang lebih baik ketika diterapkan pada data baru [32, 29, 33].

2.10 *Machine Learning*

Machine Learning merupakan salah satu cabang dari kecerdasan buatan (*Artificial Intelligence*/AI) yang berfokus pada pengembangan algoritma yang mampu mempelajari pola dari data dan menghasilkan prediksi tanpa diprogram secara eksplisit [17, 44]. Berbeda dengan pemrograman konvensional yang menggunakan aturan-aturan yang ditentukan secara manual, machine learning membangun model berdasarkan data sehingga mampu mengenali hubungan antar variabel dan menghasilkan keputusan secara otomatis.

Pada proses pembelajaran, algoritma menerima sekumpulan data pelatihan (*training data*) yang terdiri atas fitur (*feature*) dan target (*label*). Selanjutnya algoritma mempelajari hubungan antara fitur dan target tersebut untuk membentuk suatu fungsi prediksi yang dapat digunakan pada data baru yang belum pernah dilihat sebelumnya.

Secara matematis, proses pembelajaran machine learning dapat direpresentasikan sebagai fungsi pemetaan berikut.

$$f : X \rightarrow y \tag{2.16}$$

dengan:

- X : himpunan fitur atau variabel masukan.
- y : target atau kelas yang akan diprediksi.
- f : fungsi yang dipelajari oleh model machine learning.

Persamaan tersebut menunjukkan bahwa tujuan utama machine learning adalah mempelajari fungsi f yang mampu memetakan data masukan menjadi keluaran yang sesuai. Fungsi tersebut dibangun melalui proses optimasi sehingga kesalahan prediksi terhadap data pelatihan menjadi sekecil mungkin.

Berdasarkan jenis data yang digunakan, machine learning dibagi menjadi beberapa kategori, yaitu *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning* [45].

Pada penelitian ini digunakan pendekatan *supervised learning*, yaitu metode pembelajaran yang menggunakan data berlabel sehingga model dapat mempelajari hubungan antara fitur dan kelas target. Pendekatan ini dipilih karena setiap sampel pada dataset telah memiliki label MSI-H maupun Non-MSI-H sehingga proses klasifikasi dapat dilakukan secara terarah.

Kemampuan terpenting dari suatu model machine learning adalah kemampuan generalisasi (*generalization*), yaitu kemampuan model untuk memberikan prediksi yang akurat terhadap data baru yang tidak digunakan selama proses pelatihan. Model dengan kemampuan generalisasi yang baik diharapkan mampu memberikan performa yang stabil ketika diterapkan pada data nyata [21].

Proses Algoritma Machine Learning

Secara umum, proses pembelajaran machine learning dilakukan melalui tahapan berikut.

1. Mengumpulkan dan mempersiapkan dataset.
2. Melakukan preprocessing seperti pembersihan data, normalisasi, dan seleksi fitur.
3. Membagi dataset menjadi data pelatihan dan data pengujian.
4. Melatih model menggunakan data pelatihan.
5. Mengoptimalkan parameter maupun hyperparameter model.
6. Mengevaluasi performa model menggunakan data validasi atau data pengujian.
7. Menggunakan model untuk melakukan prediksi terhadap data baru.

Melalui tahapan tersebut, model diharapkan mampu mempelajari pola dari data pelatihan sehingga dapat melakukan klasifikasi dengan tingkat akurasi yang tinggi pada data yang belum pernah dilihat sebelumnya.

Kelebihan dan Kekurangan

Kelebihan machine learning adalah sebagai berikut.

- Mampu mempelajari pola yang kompleks dari data.
- Dapat menangani data berdimensi tinggi seperti data ekspresi gen.
- Memiliki kemampuan generalisasi terhadap data baru.
- Banyak digunakan dalam berbagai bidang seperti bioinformatika, kesehatan, dan pengolahan citra.

Adapun kekurangannya adalah sebagai berikut.

- Membutuhkan jumlah data yang cukup besar untuk menghasilkan model yang baik.
- Rentan mengalami overfitting apabila model terlalu kompleks.
- Memerlukan proses tuning hyperparameter agar memperoleh performa optimal.
- Interpretasi beberapa algoritma machine learning relatif sulit dibandingkan metode statistik klasik.

Alasan Digunakan pada Penelitian

Pada penelitian ini, machine learning digunakan untuk membangun model klasifikasi status MSI-H dan Non-MSI-H berdasarkan data ekspresi gen. Pendekatan ini dipilih karena mampu mempelajari hubungan kompleks antar ribuan gen secara otomatis sehingga dapat menghasilkan model prediksi yang memiliki kemampuan generalisasi yang baik. Selain itu, machine learning juga mampu menangani karakteristik data genomik yang memiliki dimensi sangat tinggi, sehingga sesuai digunakan dalam penelitian bioinformatika.

subsection*Random Forest Classification*

Random Forest Classification merupakan salah satu algoritma *ensemble learning* yang digunakan untuk menyelesaikan permasalahan klasifikasi dengan menggabungkan banyak *decision tree*. Algoritma ini pertama kali diperkenalkan oleh Breiman [46] dan bekerja berdasarkan prinsip *bagging* (*Bootstrap Aggregating*), yaitu membangun sejumlah pohon keputusan menggunakan subset data yang dipilih secara acak melalui teknik *bootstrap sampling*. Selain itu, pada setiap proses pemisahan node, hanya sebagian fitur yang dipilih secara acak untuk dipertimbangkan dalam menentukan percabangan terbaik. Kombinasi kedua mekanisme tersebut menghasilkan model yang lebih stabil, memiliki variansi yang lebih rendah, serta mampu mengurangi risiko *overfitting* dibandingkan penggunaan satu *decision tree*.

Pada proses klasifikasi, setiap *decision tree* menghasilkan prediksi kelas secara independen. Selanjutnya, seluruh hasil prediksi digabungkan menggunakan mekanisme *majority voting*, yaitu kelas yang memperoleh jumlah suara terbanyak akan menjadi prediksi akhir model. Secara matematis, proses tersebut dapat dinyatakan pada Persamaan 2.17.

$$\hat{y} = \text{mode}\{f_1(x), f_2(x), \ldots, f_T(x)\} \tag{2.17}$$

dengan:

- $\hat{y}$: hasil prediksi akhir.
- T : jumlah *decision tree*.
- $f_t(x)$: prediksi kelas dari pohon ke-t.
- mode : kelas yang paling banyak dipilih oleh seluruh pohon.

Persamaan tersebut menunjukkan bahwa hasil klasifikasi Random Forest diperoleh berdasarkan kelas yang memperoleh suara terbanyak dari seluruh *decision tree*. Pendekatan ini membuat prediksi menjadi lebih stabil dibandingkan menggunakan satu pohon keputusan saja.

Pada setiap *decision tree*, pemilihan fitur terbaik dilakukan menggunakan ukuran impuritas. Salah satu ukuran yang paling umum digunakan adalah *Gini Impurity*. Nilai Gini menunjukkan tingkat ketidakmurnian suatu node, di mana semakin kecil nilai Gini maka node tersebut semakin homogen. Persamaan Gini Impurity diberikan pada Persamaan 2.18.

$$Gini = 1 - \sum_{i=1}^{C} p_i^2 \tag{2.18}$$

dengan:

- C : jumlah kelas.
- p_i : proporsi sampel pada kelas ke-i.

Node dengan nilai Gini terkecil dipilih sebagai pemisahan terbaik karena mampu menghasilkan kelompok data yang lebih homogen.

Proses Algoritma Random Forest Classification

Secara umum, algoritma Random Forest Classification bekerja melalui tahapan berikut.

1. Mengambil sejumlah sampel dari dataset menggunakan metode *Bootstrap Sampling*.
2. Membangun satu *Decision Tree* menggunakan data hasil bootstrap.
3. Pada setiap node, memilih sejumlah fitur secara acak (*Random Feature Selection*).
4. Menghitung nilai *Gini Impurity* setiap fitur kandidat.
5. Memilih fitur yang menghasilkan nilai Gini terkecil sebagai pemisah node.
6. Mengulangi proses hingga pohon terbentuk sepenuhnya.
7. Mengulangi langkah 1 sampai 6 sebanyak jumlah pohon yang telah ditentukan.
8. Melakukan prediksi menggunakan seluruh pohon.
9. Menentukan hasil klasifikasi akhir menggunakan mekanisme *Majority Voting*.

Melalui proses tersebut, setiap pohon memiliki struktur yang berbeda karena dibangun menggunakan data dan fitur yang berbeda. Variasi antar pohon inilah yang membuat Random Forest memiliki kemampuan generalisasi yang lebih baik dibandingkan satu *Decision Tree*.

Kelebihan dan Kekurangan

Kelebihan Random Forest Classification adalah sebagai berikut.

- Memiliki akurasi yang tinggi pada berbagai jenis data.
- Mengurangi risiko *overfitting* dibandingkan *Decision Tree*.
- Mampu menangani data berdimensi tinggi seperti data ekspresi gen.
- Tidak sensitif terhadap *noise* maupun *outlier*.
- Dapat menghitung tingkat kepentingan fitur (*Feature Importance*).

Adapun kekurangan Random Forest Classification adalah sebagai berikut.

- Membutuhkan waktu komputasi yang lebih lama dibandingkan satu *Decision Tree*.
- Memerlukan memori yang lebih besar karena membangun banyak pohon.
- Interpretasi model relatif lebih sulit dibandingkan *Decision Tree*.

Alasan Digunakan pada Penelitian

Pada penelitian ini, Random Forest Classification digunakan sebagai salah satu algoritma klasifikasi untuk membedakan status MSI-H dan Non-MSI-H berdasarkan data ekspresi gen. Algoritma ini dipilih karena mampu menangani data berdimensi tinggi, memiliki ketahanan terhadap *noise*, serta mampu mengurangi risiko *overfitting* melalui mekanisme *bagging* dan *random feature selection*. Selain itu, Random Forest juga mampu menghasilkan nilai *feature importance* yang dapat digunakan untuk mengevaluasi kontribusi masing-masing gen terhadap proses klasifikasi, sehingga sangat sesuai untuk analisis data bioinformatika [46, 33, 32].

2.10.1 *Extreme Gradient Boosting* (XGBoost)

Extreme Gradient Boosting (XGBoost) merupakan salah satu algoritma *ensemble learning* berbasis *gradient boosting* yang dikembangkan oleh Chen dan Guestrin [47]. Berbeda dengan metode *bagging* seperti Random Forest yang membangun setiap pohon secara independen, XGBoost membangun *decision tree* secara berurutan (*sequential*), di mana setiap pohon baru bertugas memperbaiki kesalahan (*residual error*) yang dihasilkan oleh pohon sebelumnya. Pendekatan ini memungkinkan model belajar secara bertahap sehingga menghasilkan prediksi yang lebih akurat.

Pada setiap iterasi, model baru tidak dibangun dari awal, tetapi difokuskan pada sampel yang masih memiliki kesalahan prediksi tinggi. Dengan demikian, setiap pohon memberikan kontribusi untuk memperbaiki kelemahan model sebelumnya hingga diperoleh model akhir yang memiliki performa optimal.

Prediksi akhir pada XGBoost diperoleh dari penjumlahan seluruh prediksi yang dihasilkan oleh setiap *decision tree*. Secara matematis, proses tersebut dinyatakan pada Persamaan 2.19.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{2.19}$$

dengan:

- $\hat{y}_i$: hasil prediksi sampel ke-i.
- K : jumlah *decision tree*.
- $f_k(x_i)$: prediksi yang dihasilkan oleh pohon ke-k.

Persamaan tersebut menunjukkan bahwa prediksi akhir merupakan hasil akumulasi kontribusi seluruh pohon keputusan yang dibangun secara bertahap.

Untuk memperoleh model yang optimal, XGBoost meminimalkan fungsi objektif yang terdiri atas fungsi kesalahan (*loss function*) dan komponen regularisasi. Fungsi objektif tersebut diberikan pada Persamaan 2.20.

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

dengan:

- $l(y_i, \hat{y}_i)$: fungsi kesalahan antara nilai aktual dan hasil prediksi.
- $\Omega(f_k)$: fungsi regularisasi untuk mengontrol kompleksitas pohon.
- K : jumlah pohon.

Komponen regularisasi pada XGBoost berfungsi untuk mengurangi kompleksitas model sehingga mampu menekan risiko *overfitting*. Selain itu, XGBoost juga menerapkan teknik optimisasi menggunakan turunan pertama dan kedua (*first-order* dan *second-order gradient*) sehingga proses pelatihan menjadi lebih cepat dan akurat dibandingkan algoritma *gradient boosting* konvensional [47].

Selain regularisasi, XGBoost memiliki parameter penting yaitu *learning rate* (η). Parameter ini mengatur besarnya kontribusi setiap pohon baru terhadap model akhir. Nilai *learning rate* yang kecil menyebabkan proses pembelajaran berlangsung lebih lambat namun menghasilkan model yang lebih stabil, sedangkan nilai yang terlalu besar dapat mempercepat proses pelatihan tetapi meningkatkan risiko *overfitting*. Oleh karena itu, pemilihan nilai *learning rate* yang tepat sangat penting dalam menghasilkan model dengan kemampuan generalisasi yang baik.

Proses Algoritma XGBoost

Secara umum, algoritma XGBoost bekerja melalui tahapan berikut.

1. Menginisialisasi model dengan prediksi awal.
2. Menghitung nilai kesalahan (*residual*) antara prediksi dan data aktual.
3. Membangun satu *decision tree* baru untuk mempelajari residual tersebut.
4. Menghitung fungsi objektif yang terdiri atas *loss function* dan regularisasi.
5. Memperbarui prediksi model dengan menambahkan kontribusi pohon baru menggunakan *learning rate*.

6. Mengulangi proses hingga jumlah pohon yang diinginkan atau kondisi penghentian tercapai.
7. Menggabungkan seluruh pohon untuk menghasilkan model akhir.
8. Menggunakan model akhir untuk melakukan prediksi terhadap data baru.

Melalui proses tersebut, setiap pohon selalu memperbaiki kesalahan yang masih dilakukan oleh model sebelumnya sehingga performa klasifikasi meningkat secara bertahap pada setiap iterasi.

Kelebihan dan Kekurangan

Kelebihan XGBoost adalah sebagai berikut.

- Memiliki akurasi yang sangat tinggi pada berbagai permasalahan klasifikasi.
- Menggunakan regularisasi sehingga mampu mengurangi risiko *overfitting*.
- Mendukung pemrosesan data berdimensi tinggi.
- Memiliki efisiensi komputasi yang tinggi melalui optimisasi paralel.
- Mampu menghitung tingkat kepentingan fitur (*Feature Importance*).
- Dapat menangani nilai yang hilang (*missing value*) secara otomatis.

Adapun kekurangan XGBoost adalah sebagai berikut.

- Memiliki jumlah hyperparameter yang cukup banyak sehingga memerlukan proses *hyperparameter tuning*.
- Membutuhkan waktu komputasi yang lebih lama dibandingkan algoritma sederhana.
- Interpretasi model relatif lebih sulit dibandingkan satu *decision tree*.

Alasan Digunakan pada Penelitian

Pada penelitian ini, XGBoost digunakan sebagai salah satu algoritma klasifikasi untuk membedakan status MSI-H dan Non-MSI-H berdasarkan data ekspresi gen. Algoritma ini dipilih karena memiliki kemampuan yang sangat baik dalam menangani data berdimensi tinggi, mampu mempelajari hubungan non-linear antar fitur, serta dilengkapi dengan mekanisme regularisasi yang efektif untuk mengurangi risiko *overfitting*. Selain itu, XGBoost juga mampu menghasilkan nilai *feature importance* sehingga dapat membantu mengidentifikasi gen-gen yang memiliki kontribusi terbesar terhadap proses klasifikasi. Karakteristik tersebut menjadikan XGBoost sangat sesuai untuk analisis data genomik dan bioinformatika yang umumnya memiliki jumlah fitur jauh lebih banyak dibandingkan jumlah sampel [47, 33, 32].

2.11 *Model Evaluation*

Evaluasi model merupakan tahap penting dalam machine learning yang bertujuan untuk mengukur sejauh mana performa model dalam melakukan klasifikasi terhadap data uji. Proses ini tidak hanya digunakan untuk mengetahui tingkat akurasi model, tetapi juga untuk mengevaluasi kemampuan generalisasi model terhadap data baru yang belum pernah dilihat sebelumnya. Dalam konteks data medis dan bioinformatika, evaluasi model menjadi sangat krusial karena kesalahan prediksi dapat berdampak pada interpretasi biologis maupun keputusan klinis. Oleh karena itu, diperlukan beberapa metrik evaluasi yang dapat memberikan gambaran performa model secara lebih komprehensif [48].

A *Accuracy*

Accuracy merupakan metrik evaluasi yang paling sederhana dan paling umum digunakan dalam masalah klasifikasi. Metrik ini mengukur seberapa besar proporsi prediksi yang benar dibandingkan dengan seluruh data yang diuji. Accuracy sering digunakan karena mudah dipahami dan memberikan gambaran umum mengenai performa model secara keseluruhan. Namun demikian, interpretasi accuracy perlu dilakukan dengan hati-hati terutama pada dataset yang tidak seimbang.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.21}$$

Pada persamaan tersebut, TP (True Positive) dan TN (True Negative) merepresentasikan jumlah prediksi yang benar, sedangkan FP (False Positive) dan FN (False Negative) merupakan kesalahan prediksi. Accuracy mengukur rasio prediksi yang benar terhadap seluruh data yang diuji. Meskipun mudah dipahami, metrik ini kurang representatif pada dataset yang tidak seimbang. Hal ini disebabkan karena model dapat memiliki nilai accuracy tinggi hanya dengan memprediksi kelas mayoritas secara dominan [48].

B *Precision*

Precision merupakan metrik yang digunakan untuk mengukur tingkat ketepatan model dalam memprediksi kelas positif. Metrik ini menunjukkan seberapa banyak prediksi positif yang benar-benar merupakan kondisi positif yang sesungguhnya. Precision menjadi sangat penting dalam kasus di mana kesalahan positif palsu harus diminimalkan. Dalam bidang medis, precision yang tinggi dapat mengurangi risiko diagnosis yang tidak tepat.

$$Precision = \frac{TP}{TP + FP} \tag{2.22}$$

Pada metrik ini, fokus utama adalah meminimalkan False Positive. Precision menunjukkan tingkat keandalan model ketika memberikan prediksi positif. Nilai precision yang tinggi menunjukkan bahwa sebagian besar prediksi positif yang dihasilkan model adalah benar. Oleh karena itu, metrik ini sangat penting dalam aplikasi yang membutuhkan tingkat kepastian tinggi

terhadap hasil positif [48].

C *Recall*

Recall merupakan metrik evaluasi yang digunakan untuk mengukur kemampuan model dalam menemukan seluruh data positif yang sebenarnya ada. Metrik ini menunjukkan seberapa sensitif model dalam mendeteksi kelas positif. Recall menjadi sangat penting ketika kegagalan mendeteksi kasus positif dapat memiliki dampak yang serius. Dalam konteks medis, recall yang tinggi sangat dibutuhkan untuk memastikan tidak adanya kasus penting yang terlewat.

$$Recall = \frac{TP}{TP + FN} \tag{2.23}$$

Pada metrik ini, fokus utama adalah meminimalkan False Negative. Recall mengukur kemampuan model dalam menangkap seluruh kasus positif yang sebenarnya. Nilai recall yang tinggi menunjukkan bahwa model memiliki sensitivitas yang baik terhadap kelas positif. Oleh karena itu, recall sering digunakan sebagai metrik penting dalam aplikasi medis dan deteksi penyakit [48].

D *F1-Score*

F1-Score merupakan metrik evaluasi yang menggabungkan precision dan recall dalam satu nilai tunggal. Metrik ini menggunakan rata-rata harmonik untuk menyeimbangkan kedua nilai tersebut. F1-Score sangat berguna ketika dataset memiliki ketidakseimbangan kelas. Dengan demikian, metrik ini dapat memberikan evaluasi yang lebih adil dibandingkan accuracy.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.24}$$

F1-Score memberikan keseimbangan antara precision dan recall dalam satu nilai evaluasi. Metrik ini mempertimbangkan kedua jenis kesalahan, yaitu False Positive dan False Negative secara bersamaan. Oleh karena itu, F1-Score lebih representatif digunakan pada dataset yang tidak seimbang. Dalam penelitian ini, F1-Score menjadi metrik utama karena distribusi kelas yang tidak seimbang antara MSI-H dan Non-MSI-H [48].

2.12 *Model Interpretation*

Interpretasi model merupakan salah satu aspek penting dalam pengembangan sistem machine learning, terutama pada domain yang bersifat kritis seperti kesehatan dan bioinformatika. Hal ini disebabkan karena banyak model machine learning modern, seperti ensemble dan model berbasis kernel, bersifat *black-box* sehingga sulit untuk dipahami secara langsung. Dalam konteks penelitian medis, interpretasi model menjadi sangat penting untuk memastikan bahwa keputusan yang dihasilkan dapat dijelaskan secara transparan dan dapat dipertanggungjawabkan secara ilmiah. Oleh karena itu, diperlukan metode khusus yang mampu menjelaskan kontribusi masing-masing fitur terhadap hasil prediksi model.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f(S \cup \{i\}) - f(S)] \tag{2.25}$$

Persamaan tersebut merupakan formulasi dari SHAP (*SHapley Additive exPlanations*) yang digunakan untuk menghitung kontribusi setiap fitur terhadap output prediksi model. Dalam persamaan ini, ϕ_i menunjukkan kontribusi fitur ke-i, sedangkan F adalah himpunan seluruh fitur yang digunakan dalam model. Fungsi $f(S)$ merepresentasikan output model ketika hanya subset fitur S yang digunakan. Dengan mempertimbangkan semua kemungkinan kombinasi fitur, SHAP menghitung kontribusi rata-rata marginal dari suatu fitur terhadap seluruh kemungkinan skenario.

Pendekatan ini didasarkan pada teori permainan kooperatif (*cooperative game theory*), di mana setiap fitur dipandang sebagai "pemain" yang memberikan kontribusi terhadap hasil akhir model. Dengan cara ini, SHAP mampu memberikan nilai kontribusi yang adil, konsisten, dan dapat dibandingkan antar fitur. Keunggulan utama metode ini adalah kemampuannya dalam memberikan interpretasi lokal (untuk satu sampel) maupun global (untuk keseluruhan model). Oleh karena itu, SHAP banyak digunakan untuk menjelaskan model machine learning yang kompleks, terutama dalam bidang medis [49].