BAB 2
LANDASAN TEORI

2.1 Penyakit Jantung (Heart Disease)

Penyakit jantung merupakan salah satu penyebab utama kematian di seluruh dunia. Berdasarkan laporan dari *World Health Organization* (WHO), sekitar 17,9 juta orang meninggal setiap tahunnya akibat penyakit kardiovaskular, yang mencakup penyakit jantung koroner, penyakit serebrovaskular, dan penyakit jantung rematik [1]. Penyakit ini tidak hanya menimbulkan beban besar bagi individu, tetapi juga bagi sistem pelayanan kesehatan secara global.

Secara umum, penyakit jantung adalah kondisi yang memengaruhi struktur dan fungsi jantung. Salah satu bentuk paling umum dari penyakit jantung adalah penyakit jantung koroner (coronary heart disease), yang terjadi akibat penyempitan atau penyumbatan pembuluh darah koroner karena penumpukan plak kolesterol, yang dikenal sebagai aterosklerosis. Kondisi ini dapat menyebabkan angina (nyeri dada), serangan jantung, atau bahkan kematian mendadak jika tidak ditangani dengan baik.

Faktor risiko utama yang dapat meningkatkan kemungkinan seseorang mengalami penyakit jantung antara lain:

- Tekanan darah tinggi (hipertensi)
- Kadar kolesterol tinggi
- Diabetes atau kadar gula darah tinggi
- Merokok
- Obesitas
- Riwayat keluarga dengan penyakit jantung
- Gaya hidup tidak aktif (kurang olahraga)

2.2 Pembelajaran Mesin

Pembelajaran mesin (*machine learning*) adalah cabang kecerdasan buatan (*artificial intelligence*) yang memungkinkan sistem belajar secara otomatis dari

data tanpa pemrograman eksplisit. ML berfokus pada pengembangan algoritma yang dapat mengidentifikasi pola, membuat prediksi, atau mengambil keputusan berdasarkan data historis [14]. Dalam konteks medis, ML telah menjadi alat kritis untuk analisis data kompleks seperti rekam medis, citra diagnostik, atau sinyal biologis [15]. Pembelajaran mesin dapat dibagi menjadi beberapa pendekatan utama, sebagaimana dijelaskan berikut ini.

2.2.1 Supervised Learning

Supervised learning adalah metode pembelajaran di mana model dilatih menggunakan data yang sudah diberi label. Model akan belajar untuk memetakan fitur input terhadap label output berdasarkan data historis yang ada [16]. Contoh aplikasinya dalam dunia medis adalah klasifikasi apakah pasien memiliki penyakit jantung atau tidak berdasarkan fitur-fitur seperti tekanan darah, kolesterol, usia, dan detak jantung.

2.2.2 Unsupervised Learning

Unsupervised learning adalah metode pembelajaran di mana data tidak memiliki label, dan model bertujuan untuk menemukan pola tersembunyi di dalam data [17]. Teknik ini sering digunakan untuk *clustering* (pengelompokan) dan *dimensionality reduction*. Dalam bidang kesehatan, *unsupervised learning* dapat digunakan untuk menemukan segmen pasien dengan karakteristik risiko tertentu tanpa label diagnosis sebelumnya.

2.2.3 Reinforcement Learning

Reinforcement learning merupakan pendekatan di mana agen belajar dari interaksi dengan lingkungan melalui umpan balik berupa *reward* atau *penalty*. Teknik ini meniru cara manusia belajar melalui pengalaman.

2.3 Decision Tree

Decision tree adalah algoritma pembelajaran mesin berbasis pohon keputusan yang digunakan untuk tugas klasifikasi dan regresi. Model ini meniru cara manusia mengambil keputusan secara logis berdasarkan kondisi tertentu. Struktur dari *decision tree* menyerupai diagram alir yang terdiri atas *root node*

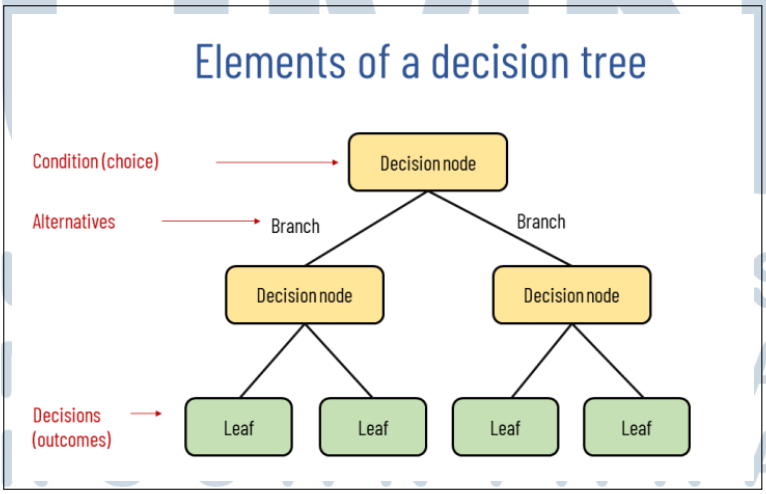
(simpul akar), *internal node* (simpul cabang), dan *leaf node* (simpul daun) yang merepresentasikan keputusan akhir atau kelas output [18].

Proses pembentukan *decision tree* dimulai dari pemilihan fitur terbaik untuk dijadikan *root node*. Pemilihan ini didasarkan pada kriteria tertentu seperti *information gain*, *gain ratio*, atau *Gini index*, yang mengukur seberapa baik suatu fitur membagi data ke dalam kelas yang homogen. Setiap percabangan membagi data ke dalam subset yang lebih kecil hingga mencapai kondisi berhenti seperti: semua data dalam satu node memiliki kelas yang sama, atau kedalaman pohon telah mencapai batas tertentu.

Berikut adalah karakteristik utama *decision tree*:

- Mampu menangani data numerik dan kategorikal.
- Dapat memodelkan hubungan non-linier antar fitur.
- Tidak memerlukan penskalaan fitur atau normalisasi.
- Mudah diinterpretasi dan divisualisasikan.

Namun, *decision tree* juga memiliki kekurangan yaitu rentan terhadap *overfitting*, terutama ketika pohon terlalu dalam dan terlalu menyesuaikan data latih. Oleh karena itu, digunakan teknik *pruning* untuk memangkas bagian pohon yang tidak memberikan kontribusi signifikan terhadap generalisasi model. Gambar 2.1 merepresentasikan struktur dari *decision tree*.



Gambar 2.1. Struktur dasar decision tree

2.4 Random Forest

Random Forest adalah salah satu algoritma *ensemble learning* yang termasuk dalam kategori *supervised learning*. Algoritma ini merupakan pengembangan dari *decision tree*, namun dengan kinerja dan akurasi yang lebih tinggi. Hal ini dikarenakan Random Forest bekerja dengan membangun beberapa *decision tree* sekaligus, lalu menggabungkan hasil prediksinya melalui proses voting untuk klasifikasi atau rata-rata untuk regresi [19].

Prinsip utama dari Random Forest adalah membentuk sekelompok *decision tree* yang dilatih secara independen dengan variasi data dan fitur, sehingga dapat mengurangi variansi dan mencegah *overfitting*. Setiap pohon dalam Random Forest dibuat dari data pelatihan yang diambil secara acak dengan teknik *bootstrap*, dan pada setiap percabangan pohon hanya dipilih subset fitur tertentu secara acak. Berikut adalah tahapan pembentukan model Random Forest

1. Menentukan parameter model, seperti jumlah pohon yang akan dibuat (n), kedalaman maksimum pohon, dan jumlah fitur yang dipertimbangkan di setiap split.

2. Melakukan proses *bootstrap sampling*, yaitu mengambil sampel acak dari dataset pelatihan (dengan pengembalian) untuk membentuk dataset pelatihan baru untuk masing-masing pohon.

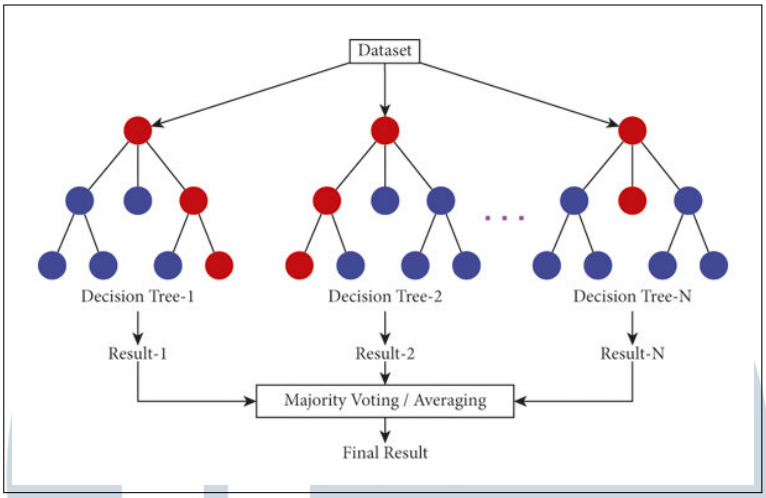
3. Untuk setiap dataset hasil *bootstrap*, bangun sebuah *decision tree* dengan memilih subset fitur secara acak pada setiap node split, lalu lakukan pemisahan berdasarkan kriteria terbaik seperti *Gini index* atau *information gain*.

4. Ulangi proses pembuatan pohon untuk setiap sampel hingga mencapai jumlah pohon yang telah ditentukan (n pohon).

5. Setelah seluruh pohon terbentuk, lakukan proses prediksi dengan meminta setiap pohon memberikan hasil prediksinya terhadap data uji.

6. Gabungkan seluruh prediksi dari masing-masing pohon:

    - Untuk klasifikasi: pilih kelas yang paling banyak dipilih oleh pohon (*majority voting*).
    - Untuk regresi: hitung rata-rata hasil prediksi dari seluruh pohon.



Gambar 2.2. Cara kerja algoritma random forest

Setelah seluruh pohon selesai dibangun, proses prediksi dilakukan dengan menggabungkan hasil prediksi dari masing-masing pohon. Untuk tugas klasifikasi, hasil akhir ditentukan berdasarkan mayoritas suara atau disebut juga dengan *majority voting*, yaitu kelas yang paling sering dipilih oleh seluruh pohon yang ada. Konsep ini memastikan bahwa prediksi akhir lebih stabil dan tidak bergantung pada satu pohon saja.

$$\hat{C}_{rf}(x) = \text{majority vote}\{\hat{C}_b(x)\}_{b=1}^{B} \tag{2.1}$$

Di mana $\hat{C}_b(x)$ adalah hasil klasifikasi dari pohon ke-b, dan B adalah jumlah total pohon yang dibangun. Walaupun memiliki performa tinggi, Random Forest sangat bergantung pada pemilihan kombinasi nilai hyperparameter yang tepat, seperti jumlah pohon (*n_estimators*), kedalaman maksimum (*max_depth*), dan jumlah fitur yang dipertimbangkan di setiap split (*max_features*). Oleh karena itu, diperlukan proses *hyperparameter tuning* seperti *randomized search* untuk memperoleh kombinasi optimal guna meningkatkan performa model.

2.5 Randomized Search untuk Optimasi Hyperparameter

Dalam pembangunan model pembelajaran mesin, pemilihan nilai hyperparameter yang optimal sangat penting untuk meningkatkan performa model. Hyperparameter adalah parameter yang tidak dipelajari langsung dari data, melainkan ditentukan sebelum proses pelatihan dimulai. Contoh hyperparameter

pada algoritma Random Forest antara lain jumlah pohon (*n_estimators*), kedalaman maksimum pohon (*max_depth*), dan jumlah fitur yang dipertimbangkan pada setiap split (*max_features*).

Randomized Search adalah salah satu metode yang digunakan untuk melakukan optimasi hyperparameter secara efisien. Tidak seperti *Grid Search* yang mencoba seluruh kombinasi parameter dalam ruang pencarian, *Randomized Search* hanya mencoba sejumlah kombinasi parameter yang dipilih secara acak. Hal ini membuat proses pencarian lebih hemat waktu dan sumber daya, terutama ketika jumlah kombinasi sangat besar [20].

Langkah-langkah dalam Randomized Search:

1. Menentukan ruang pencarian (*search space*) untuk setiap hyperparameter.
2. Menentukan jumlah iterasi (jumlah kombinasi acak yang akan diuji).
3. Pada setiap iterasi, memilih satu kombinasi nilai hyperparameter secara acak.
4. Melatih model menggunakan kombinasi yang dipilih dan mengevaluasi performanya menggunakan teknik validasi silang (*cross-validation*).
5. Menyimpan kombinasi hyperparameter dengan skor evaluasi terbaik.
6. Menggunakan kombinasi terbaik tersebut sebagai konfigurasi akhir model.

2.6 Feature Importance

Dalam proses pelatihan model Random Forest, salah satu keunggulan yang ditawarkan adalah kemampuannya untuk mengukur tingkat kepentingan atau kontribusi dari setiap fitur terhadap hasil prediksi. Nilai ini dikenal sebagai *feature importance*. Informasi ini sangat berguna dalam analisis data karena dapat membantu menentukan fitur mana yang paling berpengaruh dalam klasifikasi, serta mengidentifikasi fitur yang dapat diabaikan untuk menyederhanakan model tanpa menurunkan akurasi secara signifikan [21].

Feature importance dihitung berdasarkan seberapa besar peningkatan kinerja pemisahan kelas (misalnya berdasarkan *Gini index* atau *entropy*) ketika suatu fitur digunakan untuk melakukan pemisahan pada node di dalam pohon. Dalam Random Forest, nilai kontribusi ini diakumulasi dari semua pohon yang dibentuk, lalu dirata-ratakan untuk menghasilkan skor akhir masing-masing fitur. Tahapan penghitungan feature importance:

1. Selama proses pelatihan, setiap kali sebuah fitur digunakan untuk memisahkan node pada pohon keputusan, dicatat seberapa besar peningkatan nilai metrik impuritas (seperti *Gini gain*) yang terjadi.
2. Nilai kontribusi dari fitur tersebut diakumulasi untuk seluruh pohon dalam Random Forest.
3. Akumulasi nilai dari setiap fitur kemudian dinormalisasi sehingga total keseluruhan skor menjadi 1 (atau 100%).

Fitur dengan skor tertinggi dianggap paling penting, karena memberikan kontribusi terbesar dalam mengurangi ketidakmurnian data dan meningkatkan kemampuan klasifikasi. Sebaliknya, fitur dengan skor mendekati nol dianggap tidak terlalu relevan, dan dapat dipertimbangkan untuk dieliminasi.

Sebagai contoh, jika fitur 'usia' memiliki nilai *feature importance* sebesar 0,25, artinya sekitar 25% kontribusi terhadap akurasi prediksi berasal dari fitur tersebut. Informasi ini juga dapat divisualisasikan dalam bentuk grafik batang atau horizontal bar chart untuk mempermudah interpretasi dan analisis lanjutan.

Dengan menganalisis nilai *feature importance*, peneliti dapat melakukan seleksi fitur secara lebih sistematis. Hal ini tidak hanya membantu mengurangi kompleksitas model, tetapi juga dapat meningkatkan kecepatan pelatihan serta menghindari *overfitting* akibat dimensi fitur yang terlalu tinggi.

2.7 Confusion Matrix

Dalam konteks klasifikasi, *confusion matrix* merupakan salah satu alat evaluasi yang sangat penting. Matriks ini digunakan untuk menggambarkan performa model klasifikasi berdasarkan hasil prediksi dan nilai sebenarnya. Dengan menggunakan *confusion matrix*, kita dapat menghitung berbagai metrik evaluasi seperti *accuracy*, *precision*, *recall*, *f1-score*, dan *ROC AUC* [22].

Confusion matrix berbentuk tabel dua dimensi yang membandingkan antara kelas prediksi dan kelas aktual, biasanya digunakan dalam kasus klasifikasi biner. Tabel ini memiliki empat komponen utama:

- **True Positive (TP)**: Jumlah kasus positif yang diprediksi benar sebagai positif.
- **False Positive (FP)**: Jumlah kasus negatif yang salah diprediksi sebagai positif.

- **False Negative (FN)**: Jumlah kasus positif yang salah diprediksi sebagai negatif.
- **True Negative (TN)**: Jumlah kasus negatif yang diprediksi benar sebagai negatif.

Struktur umum *confusion matrix* ditunjukkan sebagai berikut:

Tabel 2.1. Struktur confusion matrix klasifikasi biner

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

Dari matriks ini, beberapa metrik evaluasi dapat dihitung dengan rumus berikut:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.4}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

Selain metrik di atas, evaluasi juga dapat dilakukan menggunakan *Receiver Operating Characteristic* (ROC) dan *Area Under Curve* (AUC). Nilai *ROC AUC* mengukur kemampuan model dalam membedakan antara kelas positif dan negatif, di mana nilai AUC mendekati 1 menunjukkan model yang sangat baik.

Pemilihan metrik yang tepat sangat penting terutama dalam kasus ketidakseimbangan kelas. Misalnya, ketika jumlah kasus positif jauh lebih sedikit dibandingkan negatif, metrik seperti *recall* dan *f1-score* lebih relevan dibandingkan hanya mengandalkan *accuracy*.

2.7.1 Confusion Matrix untuk Klasifikasi Multikelas

Pada kasus klasifikasi multikelas, *confusion matrix* berbentuk tabel persegi berukuran $n \times n$, di mana n adalah jumlah total kelas. Setiap baris mewakili kelas aktual, dan setiap kolom mewakili kelas yang diprediksi oleh model.
Contoh struktur confusion matrix 3 kelas:

Tabel 2.2. Confusion Matrix untuk Klasifikasi Multikelas

	Prediksi Kelas A	Prediksi Kelas B	Prediksi Kelas C
Aktual Kelas A	a	b	c
Aktual Kelas B	d	e	f
Aktual Kelas C	g	h	i

Pada klasifikasi multikelas, metrik *precision*, *recall*, dan *f1-score* dapat dihitung menggunakan pendekatan rata-rata berikut:

- **Macro average**: Menghitung metrik untuk setiap kelas, kemudian dirata-rata tanpa mempertimbangkan jumlah sampel per kelas.
- **Micro average**: Menggabungkan kontribusi dari semua kelas untuk menghitung metrik keseluruhan.
- **Weighted average**: Sama seperti macro, tetapi setiap metrik kelas diberi bobot sesuai proporsi jumlah sampelnya.

Evaluasi multikelas sangat penting untuk memahami performa model secara menyeluruh, khususnya saat distribusi kelas tidak seimbang. Penggunaan pendekatan rata-rata ini memungkinkan interpretasi yang adil dan informatif terhadap hasil klasifikasi.

Akurasi dalam klasifikasi multikelas dihitung dengan menjumlahkan semua prediksi benar (nilai diagonal dari matriks) dan membaginya dengan total jumlah seluruh sampel. Rumus akurasinya adalah sebagai berikut:

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} M_{ii}}{\sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij}} \tag{2.6}$$

Di mana M_{ii} adalah jumlah prediksi benar untuk kelas ke-i, dan M_{ij} adalah elemen baris ke-i dan kolom ke-j dari matriks. Jumlah total prediksi yang benar terletak di diagonal utama matriks, sedangkan penyebutnya adalah jumlah seluruh elemen dalam matriks.