BAB 2
LANDASAN TEORI

Terdapat beberapa teori yang digunakan dalam penelitian, yaitu Kanker Paru-Paru, *Ensemble Learning*, *Decision Tree*, *Random Forest*, dan *Confusion Matrix*.

2.1 Kanker Paru-Paru

Kanker paru-paru merupakan salah satu jenis kanker yang membahayakan di seluruh dunia. Kanker ini terjadi ketika sel-sel di paru-paru tumbuh tidak terkendali dan membentuk tumor ganas [1, 8]. Tumor ini dapat mengganggu fungsi normal paru-paru, mengganggu kemampuan organ untuk menghasilkan oksigen dan menghilangkan karbon dioksida dari tubuh [38, 39].

Penyebab utama kanker paru-paru adalah merokok, baik secara aktif maupun pasif [3, 19]. Paparan terhadap asap rokok dapat merusak DNA dalam sel-sel paru-paru, menyebabkan pertumbuhan sel yang tidak terkendali [13, 40]. Selain itu, faktor risiko lainnya termasuk paparan terhadap polusi udara juga menjadi salah satu penyumbang risiko mengidap kanker paru-paru [5, 41].

Gejala kanker paru-paru sering kali tidak terdeteksi pada tahap awal, namun gejala yang mungkin muncul ketika kanker paru-paru berkembang adalah batuk terus menerus, sesak napas, nyeri dada, batuk darah, kehilangan nafsu makan, penurunan berat badan, dan mudah lelah [2, 42].

Pemeriksaan dini dan pengobatan yang tepat sangat penting dalam penanganan kanker paru-paru. Metode diagnosis yang umum digunakan medis adalah *CT scan* dan biopsi jaringan paru-paru [7, 14]. Pengobatan kanker paru-paru dapat meliputi pembedahan, kemoterapi, radioterapi, terapi target, dan imunoterapi, tergantung pada tahapan kanker paru-paru dan kondisi kesehatan pasien [43].

Meskipun ada upaya dan perkembangan teknologi dalam bidang medis, kanker paru-paru tetap menjadi penyakit yang berbahaya yang perlu diwaspadai [44, 45]. Oleh karena itu, penelitian terus dilakukan untuk mengembangkan metode klasifikasi dini untuk meningkatkan peluang kesembuhan pasien dengan kanker paru-paru [9, 22].

2.2 Ensemble Learning

Ensemble Learning adalah pembelajaran mesin yang menggabungkan hasil dari beberapa model atau algoritma yang berbeda [46, 47]. Dengan mengumpulkan berbagai hasil dari berbagai model, *Ensemble Learning* bertujuan untuk menciptakan hasil yang lebih konsisten daripada model tunggal [48, 49]. Berikut ini beberapa jenis metode *Ensemble Learning* [50–53].

1. *Bagging*

   Bagging menggunakan beberapa model kemudian menggabungkan hasil keputusan dari semua model dengan cara *voting* untuk menghasilkan hasil keputusan akhir. Proses pelatihan model-model tersebut dilakukan secara paralel dan mandiri sehingga berbagai model tersebut dapat saling melengkapi kelemahan satu sama lain. Contoh model *Bagging* adalah Random Forest.

2. *Boosting*

   Boosting adalah metode yang berfokus pada mengoreksi kesalahan hasil keputusan model sebelumnya. Model-model ini diurutkan berdasarkan kinerja sebelumnya, dan setiap model berusaha untuk memperbaiki kesalahan yang dilakukan oleh model sebelumnya. Model *AdaBoost (Adaptive Boosting)* adalah contoh dari metode *Boosting*. Dalam *AdaBoost*, setiap model memperhatikan contoh data yang salah diklasifikasikan oleh model sebelumnya.

3. *Stacking*

   Stacking adalah metode yang menggunakan beberapa model untuk membuat hasil keputusan, dan kemudian menggabungkan hasil-hasil tersebut untuk dijadikan sebagai fitur untuk melatih model tambahan yang disebut sebagai *meta-learner*. Sebuah contoh penerapan *Stacking* adalah dengan menggunakan beberapa model seperti *Decision Trees*, *Support Vector Machines*, dan *Neural Networks* sebagai model-model dasar, kemudian menggabungkan hasil keputusan dari model-model ini sebagai fitur untuk melatih model meta-learner seperti *Logistic Regression*.

2.3 Decision Tree

Decision Tree adalah algoritma dalam *machine learning* yang digunakan untuk mengkategorikan atau membuat prediksi dari sekumpulan *dataset* [54]. *Decision Tree* digunakan untuk membantu menyelesaikan masalah yang berkaitan dengan regresi dan klasifikasi [28, 54]. Struktur *decision tree* dimulai dari simpul akar (*root node*), cabang, simpul internal (*internal node/decision node*), dan terakhir simpul daun (*leaf node/terminal node*) [55]. Simpul akar (*root node*) mewakili pertanyaan atau masalah yang ingin dipecahkan. Kemudian nantinya akan mengarah ke beberapa keputusan atau *internal node* [55, 56].

Setiap decision tree bisa memiliki beberapa *internal node* sebagai alternatif jawaban atau keputusan [56]. *Internal node* juga bisa memiliki cabang node lain yaitu *leaf node*, yang akan mewakili keputusan akhir [57]. Salah satu metode yang digunakan untuk membangun *decision tree* adalah ID3 (*interactive dichotomizer-3*) [55, 56]. ID3 (*interactive dichotomizer-3*) adalah metode yang paling banyak dikenal dan digunakan untuk membangun pohon keputusan pada tahap awal [55]. ID3 digunakan untuk memilih fitur terbaik yang memisahkan data berdasarkan *entropy* dan *information gain* [55–57].

Berikut adalah rumus ID3 untuk membangun pohon keputusan [55].

$$Ent(D) = -\sum_{i=1}^{n} P_i log_2(P_i) \tag{2.1}$$

$$Ent(D|a) = \sum_{v=1}^{v} \frac{|D_v|}{D} * Ent(D^v) \tag{2.2}$$

$$Gain(D,a) = Ent(D) - Ent(D|a) \tag{2.3}$$

Keterangan:

D = Data pelatihan

$Ent(D)$ = *Entropy* dari data pelatihan

$\sum_{i=1}^{n}$ = Penjumlahan dari i sama dengan 1 hingga n.

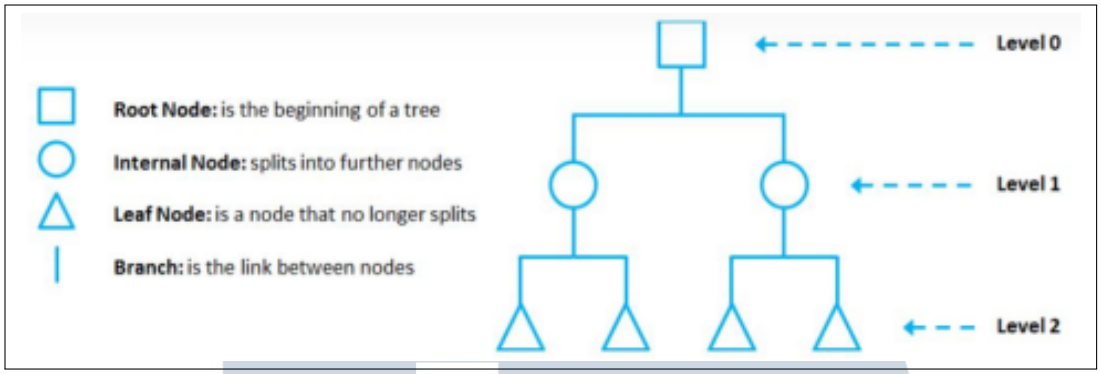
P_i = Proporsi muncul sampel kelas i dari data pelatihan

a = atribut yang digunakan untuk mempartisi *dataset* D

$Ent(D|a)$ = *Entropy* fitur a dari data pelatihan

D_v = Total sampel yang dimiliki fitur a_v

$Ent(D^v)$ = *Entropy* sampel dari fitur a_v



Gambar 2.1. *Decision Tree*

2.4 Random Forest

Algoritma *Random Forest* adalah salah satu algoritma dalam *machine learning* untuk menyelesaikan masalah klasifikasi dan regresi [58, 59]. Algoritma ini terdiri dari sejumlah pohon keputusan yang dibuat menggunakan *subset dataset* pelatihan yang dipilih secara acak [60, 61]. Algoritma *Random Forest* secara umum bekerja dengan langkah-langkah berikut [24–27].

1. Pembentukan Pohon Keputusan (*Decision Trees*)

   Setiap pohon dalam *Random Forest* dibuat secara terpisah. Masing-masing pohon ini adalah pohon keputusan yang akan mengelompokkan data dari data pelatihan seperti yang ditunjukkan pada Gambar 2.2. Proses utama pada algoritma *Random Forest* adalah membangun banyak pohon keputusan yang digunakan untuk membuat keputusan tentang bagaimana mengklasifikasikan data.

2. Pembuatan Sampel Acak (*Bootstrapping*)

   Sebelum pembuatan setiap pohon, dilakukan proses *bootstrapping*, yaitu teknik pengambilan sampel acak dengan penggantian dari data pelatihan. Ini berarti beberapa contoh data digunakan untuk melatih setiap pohon untuk menciptakan variasi hasil keputusan dari setiap pohon.

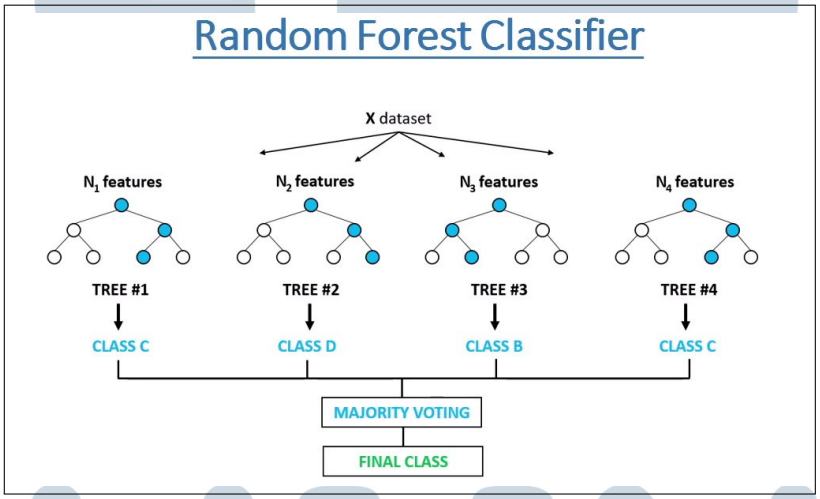
3. Pemilihan Fitur Acak (*Random Feature Selection*)

   Ketika membuat setiap pohon, hanya beberapa fitur dari *dataset* yang digunakan dalam pembuatan pohon keputusan. Ini membantu mengurangi korelasi antara pohon-pohon dan mencegah *overfitting*.

4. Pemungutan Hasil Keputusan Terbanyak (*Majority Voting*)

   Setelah semua pohon dibuat, hasil keputusan dari setiap pohon digabungkan. Dalam kasus klasifikasi, ini bisa berupa *majority voting*, di mana kelas yang paling banyak dipilih oleh pohon-pohon menjadi hasil keputusan akhir seperti yang ditunjukkan pada Gambar 2.2. Dalam kasus regresi, hasil keputusan dari setiap pohon dapat diambil rata-rata untuk memberikan hasil keputusan akhir.

Dengan begitu, *Random Forest* mampu mengatasi masalah *overfitting* dan data dimensi tinggi (*high-dimensional data*) yang artinya algoritma ini mampu menangani *dataset* yang memiliki banyak fitur atau atribut [28–32].



Gambar 2.2. Visualisasi Algoritma *Random Forest*

2.5 Confusion Matrix

Confusion Matrix merupakan suatu metode untuk mengukur performa model yang digunakan pada serangkaian data uji [62, 63]. *Confusion matrix* menunjukan hasil klasifikasi model dalam bentuk tabel dari kombinasi nilai dari nilai prediksi dan nilai aktual [64–67]. Jumlah baris dan kolom dalam *confusion matrix* ditentukan oleh jumlah kelas yang digunakan. Misalnya, jika ada tiga kelas, maka *confusion matrix* akan memiliki tiga baris dan tiga kolom seperti yang ditunjukkan pada Tabel 2.1 [68, 69].

Tabel 2.1. Bentuk Umum *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi			
		A	**B**	**C**	
Nilai Aktual	**A**	TP_A	E_{AB}	E_{AC}	N_A
	B	E_{BA}	TP_B	E_{BC}	N_B
	C	E_{CA}	E_{CB}	TP_C	N_C
		P_A	P_B	P_C	

Jumlah data yang diuji adalah

$$N_A + N_B + N_C = P_A + P_B + P_C = N \tag{2.4}$$

Sampel kelas dari nilai aktual adalah

$$N_A = TP_A + E_{AB} + E_{AC} = TP_A + FN_A \tag{2.5}$$

$$N_B = TP_B + E_{BA} + E_{BC} = TP_B + FN_B \tag{2.6}$$

$$N_C = TP_C + E_{CA} + E_{CB} = TP_C + FN_C \tag{2.7}$$

Sampel kelas dari nilai prediksi adalah

$$P_A = TP_A + E_{BA} + E_{CA} = TP_A + FP_A \tag{2.8}$$

$$P_B = TP_B + E_{AB} + E_{CB} = TP_B + FP_B \tag{2.9}$$

$$P_C = TP_C + E_{AC} + E_{BC} = TP_C + FP_C \tag{2.10}$$

Ukuran pertama yang perlu diperhatikan adalah keakuratan model klasifikasi secara keseluruhan [70]. Akurasi model didapatkan dari jumlah sampel yang diklasifikasikan (TP_A, TP_B, TP_C) dengan benar yang dibagi dengan jumlah total sampel yang diuji (N) [67].

$$Accuracy = \frac{TP_A + TP_B + TP_C}{N} \tag{2.11}$$

Tabel 2.2. Bentuk *Confusion Matrix* 2x2 Kelas A dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi	
		A	**Bukan A**
Nilai Aktual	**A**	TP_A	$\text{FN}_A = E_{AB} + E_{AC}$
	Bukan A	$\text{FP}_A = E_{BA} + E_{CA}$	$\text{TN}_A = TP_B + E_{BC} + E_{CB} + TP_C$

Tabel 2.3. Gambaran *Confusion Matrix* 3x3 Kelas A dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi		
		A	**B**	**C**
Nilai Aktual	**A**	TP_A	FN_A	FN_A
	B	FP_A	TN_A	TN_A
	C	FP_A	TN_A	TN_A

Tabel 2.4. Bentuk *Confusion Matrix* 2x2 Kelas B dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi	
		B	**Bukan B**
Nilai Aktual	**B**	TP_B	$\text{FN}_B = E_{BA} + E_{BC}$
	Bukan B	$\text{FP}_B = E_{AB} + E_{CB}$	$\text{TN}_B = TP_A + E_{AC} + E_{CA} + TP_C$

Tabel 2.5. Gambaran *Confusion Matrix* 3x3 Kelas B dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi		
		A	**B**	**C**
Nilai Aktual	**A**	TN_B	FP_B	TN_B
	B	FN_B	TP_B	FN_B
	C	TN_B	FP_B	TN_B

Tabel 2.6. Bentuk *Confusion Matrix* 2x2 Kelas C dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi	
		C	**Bukan C**
Nilai Aktual	**C**	TP_C	$\text{FN}_C = E_{CA} + E_{CB}$
	Bukan C	$\text{FP}_C = E_{AC} + E_{BC}$	$\text{TN}_C = TP_A + E_{AB} + E_{BA} + TP_B$

Tabel 2.7. Gambaran *Confusion Matrix* 3x3 Kelas C dari *Confusion Matrix* 3x3

Confusion Matrix		Nilai Prediksi		
		A	**B**	**C**
Nilai Aktual	**A**	TN_C	TN_C	FP_C
	B	TN_C	TN_C	FP_C
	C	FN_C	FN_C	TP_C

Pada Tabel 2.2 - 2.7 merupakan bentuk *confusion matrix* 2 x 2 dan gambaran *confusion matrix* 3 x 3 untuk mendapatkan kombinasi nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (FN) dari masing-masing kelas [67, 71]. Kombinasi nilai masing-masing kelas tersebut akan menjadi dasar perhitungan untuk *accuracy*, *precision*, *recall*, dan *F1-Score* [69, 72]. Berikut cara untuk menghitung *precision*, *recall* dan *F1 score* dari masing-masing kelas dan secara keseluruhan [65–67].

1. *Precision*

   Precision menggambarkan seberapa banyak dari nilai prediksi positif yang sebenarnya benar. Ini mengukur tingkat ketepatan dari prediksi positif yang dilakukan oleh model. Berikut merupakan rumus untuk mendapatkan nilai *precision* dari masing-masing kelas dan secara keseluruhan.

$$Precision_A = \frac{TP_A}{TP_A + FP_A} = \frac{TP_A}{P_A} \tag{2.12}$$

$$Precision_B = \frac{TP_B}{TP_B + FP_B} = \frac{TP_B}{P_B} \tag{2.13}$$

$$Precision_C = \frac{TP_C}{TP_C + FP_C} = \frac{TP_C}{P_C} \tag{2.14}$$

$$Precision = \frac{Precision_A + Precision_B + Precision_C}{3} \tag{2.15}$$

2. *Recall*

   Recall menggambarkan seberapa banyak dari nilai aktual positif yang sebenarnya berhasil diprediksi oleh model. Ini mengukur seberapa baik model dalam menemukan semua *instance* (data) yang sebenarnya masuk ke

dalam nilai aktual positif. Berikut merupakan rumus untuk mendapatkan nilai *recall* dari masing-masing kelas dan secara keseluruhan.

$$Recall_A = \frac{TP_A}{TP_A + FN_A} = \frac{TP_A}{N_A} \tag{2.16}$$

$$Recall_B = \frac{TP_B}{TP_B + FN_B} = \frac{TP_B}{N_B} \tag{2.17}$$

$$Recall_C = \frac{TP_C}{TP_C + FN_C} = \frac{TP_C}{N_C} \tag{2.18}$$

$$Recall = \frac{Recall_A + Recall_B + Recall_C}{3} \tag{2.19}$$

3. *F1 Score*

F1 Score adalah nilai gabungan dari *precision* dan *recall* yang memberikan gambaran tentang keseimbangan antara keduanya. Jika *F1 Score* punya hasil yang baik menggambarkan bahwa model klasifikasi yang digunakan mempunyai *precision* dan *recall* yang baik. Berikut merupakan rumus untuk mendapatkan nilai *F1 score* dari masing-masing kelas dan secara keseluruhan.

$$F1 - Score_A = \frac{2 * Precision_A * Recall_A}{Precision_A + Recall_A} \tag{2.20}$$

$$F1 - Score_B = \frac{2 * Precision_B * Recall_B}{Precision_B + Recall_B} \tag{2.21}$$

$$F1 - Score_C = \frac{2 * Precision_C * Recall_C}{Precision_C + Recall_C} \tag{2.22}$$

$$F1 - Score = \frac{F1 - Score_A + F1 - Score_B + F1 - Score_C}{3} \tag{2.23}$$