BAB II
KAJIAN LITERATUR

2.1 Skor Kredit

Skor kredit adalah nilai numerik yang diberikan oleh lembaga keuangan untuk mengukur tingkat risiko kredit seorang peminjam berdasarkan riwayat keuangan, perilaku pembayaran, dan faktor-faktor lain [9] Skor ini digunakan oleh pasar sebagai cara untuk menilai sifat tersembunyi seseorang yang tidak bisa diamati langsung, yaitu seberapa sabar individu tersebut dalam membayar utang [2]. Skor kredit memengaruhi syarat pinjaman, seperti besarnya suku bunga, limit pinjaman, dan kemudahan persetujuan. Skor ini juga tidak tetap, melainkan berubah seiring penggunaan kredit, pelunasan tepat waktu, atau keterlambatan pembayaran [9]. Semakin baik riwayat pembayaran seseorang, semakin tinggi skor kreditnya, dan semakin menguntungkan syarat pinjaman yang ditawarkan [9].

Penilaian skor kredit (*credit scoring*) adalah proses yang digunakan oleh lembaga keuangan untuk mengelola risiko kredit potensial peminjam [2]. *Credit scoring* memungkinkan pembedaan otomatis dan akurat antara peminjam baik dan buruk [2]. Proses ini mengurangi asimetri informasi antara peminjam dan pemberi pinjaman, sehingga menurunkan harga pinjaman bagi rumah tangga yang memiliki hubungan perbankan kuat [10]. Sebelum adanya skor kredit, bank memiliki monopoli atas riwayat kredit calon peminjam dan dapat membebankan suku bunga yang lebih tinggi [10]. Dengan diperkenalkannya skor kredit, masalah *hold-up* (penahanan informasi oleh bank) berkurang secara signifikan [10].

Secara keseluruhan, skor kredit berperan sebagai mekanisme dinamis yang mengintegrasikan pembagian informasi antar lembaga keuangan [2]. Penilaian skor kredit memungkinkan terciptanya sistem keuangan yang lebih transparan dan efisien karena setiap peminjam dinilai berdasarkan perilaku kredit historisnya [10]. Selain itu, skor kredit memberikan manfaat bagi peminjam dengan memberikan peluang akses kredit yang lebih adil serta mendorong perilaku pembayaran yang disiplin agar skor tetap baik di masa depan [2]. Dengan demikian, penilaian skor kredit (*credit scoring*) menjadi alat utama dalam manajemen risiko kredit yang modern dan efektif di berbagai negara [10].

2.2 Ketidakseimbangan Data

Ketidakseimbangan data (*imbalanced data*) terjadi ketika distribusi kelas dalam suatu dataset tidak merata, yaitu satu kelas (kelas mayoritas/*majority class*) memiliki jumlah sampel yang jauh lebih banyak dibandingkan kelas lain (kelas minoritas/*minority class*) [11]. *Data imbalance* merujuk pada kondisi di mana sampel dalam setiap kategori tidak seimbang, sehingga dataset dapat dibagi menjadi *majority class* (*negative class*) dan *minority class* (*positive class*) berdasarkan ukuran sampel [12]

Dalam klasifikasi kredit skor, ketidakseimbangan ini sangat umum terjadi. Kelas "*High*" atau nasabah tidak bermasalah biasanya mendominasi, sementara kelas "*Low*" atau nasabah berisiko tinggi jumlahnya jauh lebih sedikit [11]. Ketidakseimbangan ini menyebabkan model *machine learning* cenderung bias terhadap kelas mayoritas [11].Karena algoritma klasifikasi tradisional bertujuan memaksimalkan akurasi keseluruhan dan dibangun berdasarkan asumsi distribusi kategori yang relatif seimbang, maka tingkat kesalahan klasifikasi (*misclassification rate*) pada sampel minoritas menjadi tinggi [12]

Akibatnya, model sering kali mengabaikan kelas minoritas meskipun akurasi keseluruhan tampak tinggi [12] Hal ini menimbulkan masalah serius karena kelas minoritas (misalnya peminjam berisiko tinggi atau transaksi *fraud*) justru merupakan kasus yang paling penting untuk dideteksi dengan akurat [12] Dalam praktiknya, ketidakseimbangan data menyebabkan peningkatan *false negative* pada kelas minoritas, sehingga model gagal mengidentifikasi peminjam berisiko atau transaksi *fraud*, yang berpotensi menimbulkan kerugian finansial yang besar bagi lembaga keuangan [11].

Selain itu, ketidakseimbangan juga memperburuk performa metrik evaluasi standar seperti *precision, recall, dan F1-score* khususnya pada kelas minoritas [11], [12], [13]. Dalam domain *credit card fraud detection*, jumlah aktivitas *fraud* sangat minim dibandingkan transaksi legal, sehingga menciptakan masalah *class imbalance* yang signifikan [11], [12], [13]. Kondisi ini menyebabkan model sulit mendeteksi *fraud* secara akurat dan meningkatkan risiko *false positive* maupun *false negative* [11], [12], [13].

Dengan demikian, ketidakseimbangan data bukan hanya masalah distribusi sampel semata, melainkan sumber utama bias model yang dapat menurunkan akurasi prediksi risiko kredit dan deteksi *fraud* secara keseluruhan [11], [12], [13]. Masalah ini menjadi salah satu tantangan utama dalam *machine learning* yang memerlukan penanganan khusus agar model dapat bekerja secara andal pada data dunia nyata yang jarang seimbang [11].

2.3 *Preprocessing*

Preprocessing merupakan proses pengolahan data awal yang digunakan untuk mengubah data mentah yang diperoleh dari berbagai sumber menjadi informasi yang lebih bersih dan dapat digunakan untuk analisis lebih lanjut [14]. Data mentah yang dikumpulkan sering kali mengandung masalah kualitas berupa ketidaklengkapan, ketidakkonsistenan, *noise*, redundansi (pengulangan) , serta pencatatan duplikat. Jika masalah-masalah tersebut tidak diatasi, informasi yang diekstrak dari data pun akan keliru. Oleh karena itu, *preprocessing* menjadi tahap yang sangat penting dalam proses analisis data [14].

Pada tahap ini, data target terlebih dahulu dipilih dari seluruh himpunan data melalui proses pemilihan data. Selanjutnya, metode *preprocessing* diterapkan untuk memperbaiki kesalahan dan meningkatkan kualitas data. Pada bagian akhir tahap ini, data diubah menjadi bentuk yang sesuai untuk algoritma *data mining* (penambangan data), sehingga proses *mining* dapat berjalan lebih efisien dan model yang dihasilkan lebih mudah dipahami [14]. Dengan demikian, *preprocessing* memastikan akurasi dan konsistensi dari data yang digunakan dalam analisis selanjutnya [14].

Selain itu, sebelum memasuki tahap pelatihan, *dataset* dibagi menjadi *data training* (80%) dan *data testing* (20%) menggunakan *stratified sampling*. *Stratified random sampling* membagi populasi menjadi subkelompok atau strata berdasarkan karakteristik target, kemudian mengambil sampel secara acak dari setiap stratum untuk memastikan proporsi kelas target (*Good*, *Standard*, *Poor*) tetap terjaga pada kedua himpunan data [15]. Rasio 80% *training* dan 20% *testing* merupakan proporsi yang umum digunakan dalam *machine learning* karena memberikan keseimbangan yang baik antara data untuk melatih model dan data untuk evaluasi [16]. Untuk menjaga konsistensi hasil penelitian, pembagian data dilakukan dengan menetapkan nilai *random_seed*. Pengaturan ini memastikan bahwa proses pengacakan data selalu menghasilkan pembagian yang sama setiap kali kode dijalankan ulang, sehingga eksperimen dapat direproduksi dengan hasil yang identik [17]

2.3.1 *Missing Value Imputation*

Missing value imputation merupakan salah satu permasalahan utama dalam *preprocessing data*. Nilai untuk satu atau beberapa atribut dari beberapa sampel dapat hilang dalam *dataset* mentah. Situasi ini disebut sebagai masalah *missing value* dan memiliki berbagai penyebab, seperti kesalahan *measuring device* (perangkat pengukur), kesalahan pencatatan oleh manusia, serta kesalahan jaringan [14].

Ketika dataset yang mengandung *missing value* diberikan langsung kepada algoritma pembelajaran, baik tingkat akurasi model akan menurun atau bahkan model tidak dapat terbentuk sama sekali karena algoritma gagal berjalan. Oleh karena itu, untuk dapat mengekstrak pengetahuan dari *dataset*, data tersebut harus dibersihkan dan dipersiapkan terlebih dahulu untuk proses *data mining* (penambangan data) [14].

Pada dasarnya terdapat tiga jenis data yang hilang, yaitu *Missing Completely at Random* (MCAR), *Missing at Random* (MAR), dan *Missing Data Not at Random* (MNAR). MCAR terjadi ketika tidak ada hubungan sama sekali antara nilai yang hilang dengan nilai lain yang teramati maupun yang hilang dalam dataset. MAR berarti ketidakhadiran suatu data tidak acak dan tidak berhubungan dengan data yang hilang itu sendiri, melainkan dengan beberapa data lain yang teramati dalam dataset. Sedangkan jenis yang tidak termasuk MCAR maupun MAR disebut MNAR [14].

Dengan demikian, *missing value imputation* menjadi langkah penting dalam preprocessing untuk mengatasi masalah ini sebelum melanjutkan ke tahap analisis data selanjutnya.

2.3.2 *Matrix Correlation Analysis*

Matriks korelasi merupakan hal yang sangat mendasar dalam banyak studi ilmiah yang melibatkan banyak variabel, dan visualisasi koefisien korelasi serta matriks korelasi dapat dilakukan dengan berbagai cara [18]. Salah satu visualisasi populer adalah *corrgram* yang menggunakan warna dan *shading* pada tampilan tabular untuk merepresentasikan entri matriks korelasi, mirip dengan *heatmap* [18].

Analisis korelasi dapat digunakan sebagai alat untuk mengidentifikasi fitur redundan yang didefinisikan oleh ketergantungan linier [19]. Fitur yang memiliki tingkat korelasi kuat (misalnya ukuran *Pearson* lebih besar dari 0,9) berpotensi menyebabkan *multikolinearitas* yaitu fitur yang terlalu mirip atau berkorelasi tinggi yang merusak kestabilan dan *interpretabilitas* model linier [19].

Oleh karena itu, analisa matriks korelasi melalui visualisasi yang tepat merupakan langkah krusial dalam *preprocessing* data untuk mendeteksi pola hubungan serta redundansi antar variabel, sehingga memastikan kualitas data yang lebih baik sebelum memasuki tahap pemodelan selanjutnya [18], [19].

2.3.3 *Data Cleansing*

Data cleansing merupakan tugas utama dalam persiapan data yang terdiri dari mendeteksi dan menghilangkan kesalahan data [20]. *Data cleansing* adalah proses mendeteksi atau memperbaiki catatan yang rusak, duplikat, tidak lengkap, tidak akurat, atau *noisy* dari dataset, dengan tujuan meningkatkan kualitas dataset [20]. Proses ini sangat penting karena kinerja model *machine learning* sangat bergantung pada kualitas data yang digunakan untuk pelatihan [20].

1. Konversi Tipe Data

   Konversi tipe data merupakan bagian penting dari *structural standardization* dalam *preprocessing* data [21]. *Structural standardization* bertujuan untuk menciptakan representasi data yang konsisten dan dapat digunakan dalam hal format dan tipe data [21]. Proses ini mencakup konversi tipe data, seperti mengubah tipe *boolean* menjadi kategori atau numerik, serta mengubah tipe kompleks menjadi bentuk yang lebih mudah diinterpretasikan, misalnya mengubah tanggal dalam format *string* menjadi format tanggal/waktu atau numerik [21].

2. Penanganan *Placeholder*

   Operator sering kali memasukkan *placeholder values* seperti angka atau teks sembarang ketika pengguna gagal menyediakan informasi yang diperlukan [22]. Nilai-nilai *surrogate* (pengganti) ini sebenarnya merupakan data yang hilang dan harus diidentifikasi serta ditangani selama tahap *preprocessing* [22]. Penanganan *placeholder* dilakukan untuk membersihkan entri-entri tidak valid tersebut sehingga tidak mengganggu proses *encoding* dan pelatihan model [22].

3. Penanganan Nilai Negatif

   Penanganan nilai negatif pada *dataset* dilakukan dengan mengidentifikasi nilai yang berada di luar rentang yang dapat diterima untuk pertanyaan tersebut, termasuk nilai negatif pada bidang yang hanya boleh memiliki nilai positif (misalnya harga roti) [23]. Beberapa nilai data jelas secara logis atau biologis tidak mungkin, seperti pria tidak dapat hamil atau harga roti tidak dapat negatif [23]. Pemeriksaan rentang memastikan bahwa setiap variabel dalam survei hanya berisi data dalam *domain* nilai valid yang terbatas [23]. Variabel numerik harus berada dalam nilai minimum dan maksimum yang telah ditentukan, seperti 0 hingga 120 tahun untuk usia, dan harus selalu dinyatakan sebagai *integer* tahun dengan aturan pembulatan naik atau turun yang sesuai untuk bayi [23]. Proses ini dilakukan melalui pemeriksaan bahwa nilai numerik berada dalam rentang yang telah ditentukan sebelumnya [23].

2.3.4 *Data Encoding*

Algoritma *machine learning* memerlukan input numerik, sehingga diperlukan teknik untuk mengubah variabel kategorikal menjadi bentuk numerik yang sesuai [24]. Label *Encoding* melibatkan konversi setiap nilai dalam kolom kategorikal menjadi angka [24]. *One-Hot Encoding* melibatkan pembuatan kolom baru yang menunjukkan kehadiran atau ketiadaan setiap nilai unik dalam data asli [24]. Proses *encoding* ini memastikan variabel kategorikal dapat diintegrasikan ke dalam model *machine learning* tanpa menimbulkan kesalahan atau performa yang buruk [24].

Variabel kategorikal merepresentasikan jenis data yang dapat dibagi ke dalam kelompok-kelompok [24]. Karena algoritma machine learning membutuhkan input numerik, maka diperlukan teknik untuk mengubah variabel kategorikal tersebut menjadi bentuk numerik yang sesuai [24]. Salah satu teknik yang digunakan adalah Label Encoding, yaitu dengan mengonversi setiap nilai dalam kolom kategorikal menjadi sebuah angka [24]. Teknik lain yang sering digunakan adalah One-Hot Encoding, yang membuat kolom-kolom baru untuk menunjukkan kehadiran atau ketiadaan setiap nilai unik dalam data asli [24].

Selain itu, terdapat juga Target Encoding yang dikenal sebagai mean encoding [24]. Metode ini mengubah variabel kategorikal berdasarkan nilai rata-rata dari variabel target [24]. Target Encoding sangat berguna untuk fitur dengan kardinalitas tinggi karena One-Hot Encoding pada kondisi tersebut dapat menyebabkan konsumsi memori yang tinggi [24]. Cara kerjanya adalah dengan menghitung nilai rata-rata variabel target untuk setiap kategori, kemudian mengganti kategori tersebut dengan nilai rata-rata yang telah dihitung [24]

2.3.5 *Data Scaling*

Feature scaling merupakan teknik pemetaan dalam tahap *preprocessing* yang bertujuan memberikan bobot yang sama pada semua atribut [25]. Proses ini dilakukan untuk memastikan bahwa fitur dengan rentang nilai yang lebih besar tidak mendominasi fitur lainnya dalam perhitungan jarak atau bobot [26]. Dalam beberapa aplikasi, transformasi data ini dapat meningkatkan performa model *machine learning* secara signifikan [25].

Penerapan metode scaling tanpa evaluasi yang cermat terhadap kesesuaiannya dengan masalah dan model tertentu tidak disarankan karena dapat berdampak negatif terhadap hasil [25]. Praktik tersebut berisiko mengurangi validitas klaim mengenai

performa model dan dapat meningkatkan risiko overfitting, yaitu ketika model menyesuaikan diri terlalu dekat dengan noise atau karakteristik khusus dalam data pelatihan sehingga menurunkan kemampuan generalisasi [25].

Seiring dengan semakin luasnya penerapan metode machine learning, penting untuk menilai dan menjustifikasi setiap langkah dalam pipeline pemodelan, termasuk feature scaling, agar menghasilkan temuan yang robust dan dapat direplikasi [25].

2.4 *Machine Learning*

Machine learning adalah subbidang kecerdasan buatan yang memungkinkan komputer belajar dari data tanpa pemrograman eksplisit [27].

Teknik *machine learning* dibagi menjadi 4 bagian, yaitu :

1. *Supervised Learning*

   Supervised learning menggunakan data dengan hasil yang sudah diketahui untuk melatih model agar dapat mengklasifikasikan atau memprediksi hasil baru pada data yang belum pernah dilihat [27]. *Supervised learning* sangat umum digunakan dalam masalah klasifikasi karena tujuannya adalah agar komputer mempelajari sistem klasifikasi yang telah dibuat [28].Klasifikasi adalah teknik *supervised learning* yang melibatkan pengkategorian data ke dalam kelas-kelas yang berbeda [29]. Klasifikasi merupakan teknik yang digunakan untuk memprediksi informasi serupa berdasarkan nilai variable dari kelas kategorikal target [29]. Klasifikasi digunakan untuk memprediksi hasil suatu masalah berdasarkan fitur *input* dan bertujuan untuk mengalokasikan pola yang tidak diketahui ke kelas yang diketahui [29]. Bidang klasifikasi mencakup empat jenis tugas utama, yaitu klasifikasi *biner, multi-kelas, multi-label*, dan klasifikasi tidak seimbang [29]. Klasifikasi melibatkan penggunaan data pelatihan yang dikategorikan untuk menetapkan label pada dataset yang sesuai [29]. Klasifikasi dapat diterapkan pada data terstruktur maupun tidak terstruktur dan kelas-kelas umumnya dikenal sebagai target, label, atau kategori [29]. Pemahaman teknik klasifikasi sangat penting untuk mencapai hasil yang dapat diandalkan di berbagai domain [29].

2. *Unsupervised Learning*

   Unsupervised learning bekerja pada data tanpa label [28]. *Unsupervised learning* memiliki 2 teknik utama, yaitu *clustering* dan *association* [30]. *Clustering* adalah proses mengelompokkan objek atau data yang memiliki kemiripan sifat menjadi

beberapa kelompok. Sedangkan, *association* adalah teknik unsupervised learning yang digunakan untuk menemukan hubungan antar variabel dalam dataset besar [30],

3. *Semi-Supervised Learning*

   Semi-supervised learning dapat meningkatkan performa model *deep neural network* secara signifikan dengan memanfaatkan data tidak berlabel ketika data berlabel sangat terbatas [31]. *Semi-supervised learning* merupakan paradigma (kerangka berpikir) yang dapat meningkatkan performa pembelajaran dengan sedikit data berlabel dengan menggunakan contoh tambahan yang tidak berlabel sebagai pendukung dibandingkan dengan *supervised learning* [31]. *Semi-supervised learning* menyediakan cara untuk mengeksplorasi pola laten (tersembunyi) dari contoh tidak berlabel tambahan, sehingga mengurangi kebutuhan akan jumlah label yang besar [31].

4. *Reinforcement Learning*

   Reinforcement learning merupakan pendekatan di mana agen belajar melalui percobaan dan kesalahan dengan menggunakan sistem *reward* [32]. *Reward* positif diberikan ketika agen berhasil mengklasifikasikan sampel dengan benar, dan *reward* negatif diberikan jika sebaliknya [32]. Hal ini sangat penting, sebagai contoh dalam konteks tugas klinis di mana tujuannya adalah melatih model yang mampu menggeneralisasi dengan baik pada berbagai hasil data pasien [32].

2.5 *Ensemble Learning*

Ensemble learning merupakan teknik yang digunakan untuk menggabungkan dua atau lebih algoritma *machine learning* guna mendapatkan performa yang lebih unggul dibandingkan ketika algoritma konstituen (komponen penyusun) digunakan secara individual [33]. *Ensemble learning* bertujuan untuk mengurangi kesalahan generalisasi prediksi model [34]. Metode *ensemble* dapat dibagi menjadi *bagging*, *boosting*, *stacking*, dan *voting* [33].

1. *Bagging*

   Bagging adalah agregasi (menyatukan) prediksi dari banyak pohon keputusan yang telah disesuaikan dengan sampel berbeda dari dataset yang sama [33]. *Bagging* mengurangi varians tanpa meningkatkan bias [33]. *Bagging* mencapai keragaman dengan pengambilan sampel ulang data pelatihan [35]. *Bagging* melatih model yang berbeda secara independen dan menggabungkan prediksi mereka menggunakan aturan mayoritas [35].

2. *Boosting*

Boosting adalah teknik *machine learning* yang mampu mengubah pembelajar lemah menjadi pengklasifikasi kuat [33]. *Boosting* mengurangi bias [33]. *Boosting* melatih model secara iterative (berulang) sehingga model pada setiap iterasi belajar untuk memperbaiki kesalahan yang dibuat oleh model sebelumnya [35]. *Boosting* menggunakan pembelajaran sekuensial (berurut) sehingga model pada setiap iterasi belajar untuk memperbaiki kesalahan yang dibuat oleh model sebelumnya [35].

3. *Stacking*

   Stacking adalah metode *ensemble* di mana satu atau lebih pengklasifikasi tingkat dasar ditumpuk dengan pengklasifikasi *meta-learner* [36]. *Stacking* melibatkan penyesuaian beberapa jenis model berbeda pada data yang sama dan kemudian menggunakan model berbeda untuk menentukan cara menggabungkan hasil secara paling efisien [34]. *Stacking* menggunakan meta-learner untuk menggabungkan keluaran dari *base learner* [36]. *Base learner* adalah model-model sederhana seperti *Decision Tree*, SVM, atau KNN yang bekerja secara independent [34]. *Meta-learner* adalah model tingkat kedua (biasanya *Logistic Regression* atau *Neural Network*) yang belajar cara terbaik menggabungkan prediksi dari semua *base learner* agar menghasilkan prediksi akhir yang lebih akurat [34].

4. *Voting*

   Voting adalah strategi pengklasifikasi *ensemble* yang menggabungkan prediksi dari beberapa model independen untuk membuat prediksi akhir [34]. *Voting* menggunakan konsep kebijaksanaan kerumunan untuk menghasilkan prediksi yang lebih akurat dengan mempertimbangkan penilaian agregat (penggabungan) dari beberapa model daripada bergantung pada satu model saja [34]. *Voting* terdiri dari dua jenis yaitu *hard voting* (*modus*) dan *soft voting* (*mean)* [34].

Dengan demikian, *ensemble learning* melatih beberapa pembelajar dasar dan menggabungkan prediksi mereka untuk mendapatkan performa yang lebih baik serta kemampuan generalisasi yang lebih tinggi daripada pembelajar individual [33].

2.6 *Random Forest*

Random Forest adalah teknik *ensemble learning* yang menggabungkan beberapa *decision tree* untuk menghasilkan prediksi akhir melalui *majority vote* pada klasifikasi atau rata-rata pada regresi [6]. *Random Forest* merupakan metode yang dibangun berdasarkan metode *bagging* dengan menggunakan *random subset data* dan *random subset* fitur untuk setiap pohon keputusan [6]. *Random Forest* menggunakan *random*

feature selection & *sampling* sehingga setiap pohon dilatih secara independen dan mengurangi korelasi antar pohon [37]. Prediksi akhir *Random Forest* pada klasifikasi dihitung menggunakan *majority vote* dari semua *decision tree* [6]:

$C(t) = max_p \sum_{i=1}^{K}(c_i(T) = P)$ ..... (2.1)

Penjelasan :

$C(t)$: Variabel Hasil (output)

max_p : Nilai Maksimum P

K : Total Populasi

i : Indeks Data

$c_i(T)$: Data Spesifik

Random Forest mampu mengatasi o*verfitting* dengan melatih banyak *decision tree* pada *subset* data acak sehingga meningkatkan kemampuan generalisasi terhadap data baru [37]. *Random Forest* dapat menangani data tidak seimbang dan variabel dengan nilai hilang tanpa *preprocessing* tambahan yang rumit [37]. *Random Forest* memiliki keunggulan dalam menangani dataset besar dengan ribuan variabel tanpa perlu menghapus variabel apapun [37]. *Random Forest* juga memberikan ukuran pentingnya variabel (*variable importance*) secara otomatis jika menggunakan pohon keputusan berbasis *Gini Impurity* melalui *Gini importance* [6]:

$i(t) = 1 - f_1^2 - f_0^2$ ..... (2.2)

Penjelasan :

$i(t)$: *Gini Impurity*

1 : Nilai Maksimum Ideal

f_1^k : Frekuensi Relatif Kelas 1

f_0^k : Frekuensi Relatif Kelas 0

Decision Tree merupakan model prediksi yang menggunakan struktur pohon dengan *root node, internal nodes, dan leaf nodes* [37]. *Decision Tree* terbagi menjadi dua jenis utama: *classification trees* (untuk variabel target diskrit/kategorikal) dan *regression trees* (untuk variabel target kontinu/numerik) [37]. *Decision Tree* dibangun menggunakan algoritma CART (*Classification and Regression Tree*) yang membagi data secara *rekursif* (mengulang) berdasarkan aturan "*if-then-else*" hingga mencapai *leaf node* yang menyimpan hasil prediksi [37]. *Decision Tree* bersifat nonparametric (fleksibel), tidak memerlukan asumsi distribusi data, dan dapat menangani variabel kategorikal maupun numerik sekaligus [37].

2.7 XGBoost

XGBoost (*eXtreme Gradient Boosting*) adalah algoritma *gradient boosting* yang dirancang agar lebih cepat dan lebih akurat dibandingkan metode *gradient boosting* konvensional. Cara kerjanya adalah membangun model secara bertahap (*additive*) dengan menambahkan pohon keputusan (*decision tree*) satu per satu. Setiap pohon baru yang ditambahkan bertugas memperbaiki kesalahan (*residual*/*error*) dari prediksi pohon-pohon sebelumnya [38].

Prediksi akhir XGBoost diperoleh dengan menjumlahkan kontribusi dari semua pohon keputusan yang dibangun secara bertahap, sehingga untuk setiap sampel ke-i hasil akhirnya dapat dinyatakan sebagai [38] :

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \quad \ldots\ldots (2.3)$$

Penjelasan :

$\hat{y}_i$ = prediksi akhir untuk sampel ke-i

f_k = fungsi (pohon keputusan *CART*) ke-k

K = jumlah total pohon yang dibangun

1. Fungsi Objektif (Objective Function)

XGBoost menggunakan informasi kelengkungan (*curvature*) dari *loss function* agar optimasi menjadi lebih tepat. Dengan melakukan ekspansi *Taylor* hingga orde kedua pada fungsi objektif [38]:

$$\text{Obj} = \sum_i^K I(y_i \hat{y}_i) + \sum_k^K \Omega(f_k) \quad \ldots\ldots (2.4)$$

Di mana:

$I(y_i \hat{y}_i)$ = *loss function* (misalnya *logistic loss* untuk klasifikasi, *squared error* untuk regresi) → mengukur seberapa jauh prediksi dari nilai sebenarnya.

Loss function biasanya didefinisikan sebagai [38] :

$$l(y, \hat{y}) = -\left[y \log(\hat{y})(1-y)\log(1-\hat{y}) \right] \quad \ldots\ldots (2.5)$$

$\Omega(f_k)$ = regularisasi (*penalty*) untuk pohon ke-k → mencegah pohon terlalu kompleks (*overfitting*).

Regularisasi Ω biasanya didefinisikan sebagai [38] :

$$\Omega(f) = \gamma T \frac{1}{2} \lambda \|w\|^2 \quad \ldots\ldots (2.6)$$

Di mana:

T = jumlah *leaf node* (semakin banyak *leaf* → semakin kompleks)

w = bobot (*weight/score*) pada setiap *leaf*

γ dan λ = *hyperparameter* untuk mengontrol kompleksitas.

2. Proses Pembelajaran Bertahap (*Additive Training*)

   XGBoost membangun pohon secara *sequential* [38] :

   a. Mulai dengan model awal

   b. Hitung *residual*

   c. Pohon baru dilatih untuk memprediksi *residual*

   d. Tambahkan dengan *learning rate*

$$\eta: \hat{y}_i^{(new)} = \hat{y}_i^{(ad)} \quad \eta \cdot f_k(x_i) \quad \ldots\ldots (2.7)$$

   e. Ulangi sampai K optimal.

3. Pendekatan *Taylor Order*-2

   XGBoost menggunakan *second-order Taylor expansion* [38] :

$$\text{Obj}^{(i)} \approx \sum_{i=1}^{n} [I(y_i, \hat{y}_i^{(i-1)}) g_i f_i(x_i) \frac{1}{2} h_i f_i^2(x_i)] \Omega(f_i) \quad \ldots\ldots\ldots (2.8)$$

   di mana:

$$g_i^{(k)} = \hat{p}_i^{(k)} - y_i^{(k)} \quad (\textit{Gradient / first derivative}) \ldots\ldots\ldots (2.9)$$

$$h_i^{(k)} = \hat{p}_i^{(k)} \times (1 - \hat{p}_i^{(k)}) \quad (\textit{Hessian / second derivative}) \ldots\ldots\ldots (2.10)$$

4. Struktur Pohon dan Perhitungan *Gain Split*

   Untuk setiap pohon baru:

   a. XGBoost mendefinisikan kompleksitas pohon dengan [38] :

$$\Omega(f) = \gamma T \frac{1}{2} \wedge \sum_{j=1}^{T} w_j^2 \quad \ldots\ldots\ldots (2.11)$$

   Di mana:

   T = jumlah *leaf*

   w_j = *skor/bobot leaf ke-j*

   b. Bobot optimal pada setiap *leaf* dihitung secara analitis (*closed-form*):

$$w_j^* = -\frac{g_i}{h_i + \lambda} \quad \ldots\ldots\ldots (2.12)$$

   Di mana:

   I_j = *himpunan sampel yang jatuh ke leaf ke-j)*

   c. *Gain split* (keuntungan membagi *node*) dihitung sebagai:

$$Gain = \frac{1}{2}\left[\frac{(\sum g_L)^2 (\sum g_R)^2}{\sum h_L \lambda \ \sum h_R \lambda} - \frac{(\sum g_J)^2}{\sum h_I \lambda}\right] - \gamma \quad \ldots\ldots\ldots (2.13)$$

   Gain ini digunakan untuk memilih fitur dan titik *split* terbaik pada setiap *node* [38].

2.8 *Hybrid*

Hybrid merupakan pendekatan *machine learning* yang menggabungkan dua atau lebih algoritma atau teknik dari bidang yang berbeda untuk menyelesaikan suatu masalah atau meningkatkan hasil algoritma tunggal [39]. Pendekatan ini memanfaatkan kelebihan masing-masing metode sehingga dapat mengatasi kelemahan yang ada pada algoritma tunggal, seperti mudah terjebak di solusi yang kurang optimal, serta meningkatkan kemampuan mencari solusi yang lebih baik dan lebih cepat [39].

Menurut Azevedo et al. (2024), tidak ada algoritma yang sempurna; setiap algoritma memiliki keterbatasan tertentu yang dapat dikurangi atau dihilangkan melalui kombinasi metodologi yang berbeda [39]. Metode *Hybrid* memungkinkan integrasi antara teknik optimisasi (seperti *swarm intelligence* dan *evolutionary algorithms*) dengan *machine learning* (*supervised*, *unsupervised*, *semi-supervised learning* atau *reinforcement learning*) [9]. Kombinasi ini menghasilkan kerangka kerja yang lebih kuat dan efisien, terutama dalam menghadapi masalah kompleks yang melibatkan data besar, seperti *clustering* dan *classification* [39].

Penerapan metode *hybrid* telah menunjukkan potensi besar dalam mengatasi kelemahan algoritma murni [39]. Misalnya, algoritma optimisasi *bio-inspired* seperti *Particle Swarm Optimization* (PSO) dan *Genetic Algorithm* (GA) sering digabungkan dengan algoritma *machine learning* seperti Su*pport Vector Machine* (SVM), *k-Nearest Neighbors* (k-NN), atau *Decision Tree* untuk meningkatkan akurasi klasifikasi, mengurangi *overfitting*, serta mempercepat proses pencarian solusi optimal [39]. Dengan demikian, pendekatan *hybrid* tidak hanya memperkuat kelebihan masing-masing metode, tetapi juga membuka peluang baru untuk aplikasi di berbagai bidang yang memerlukan solusi akurat dan cepat [39].

2.9 *Stacking*

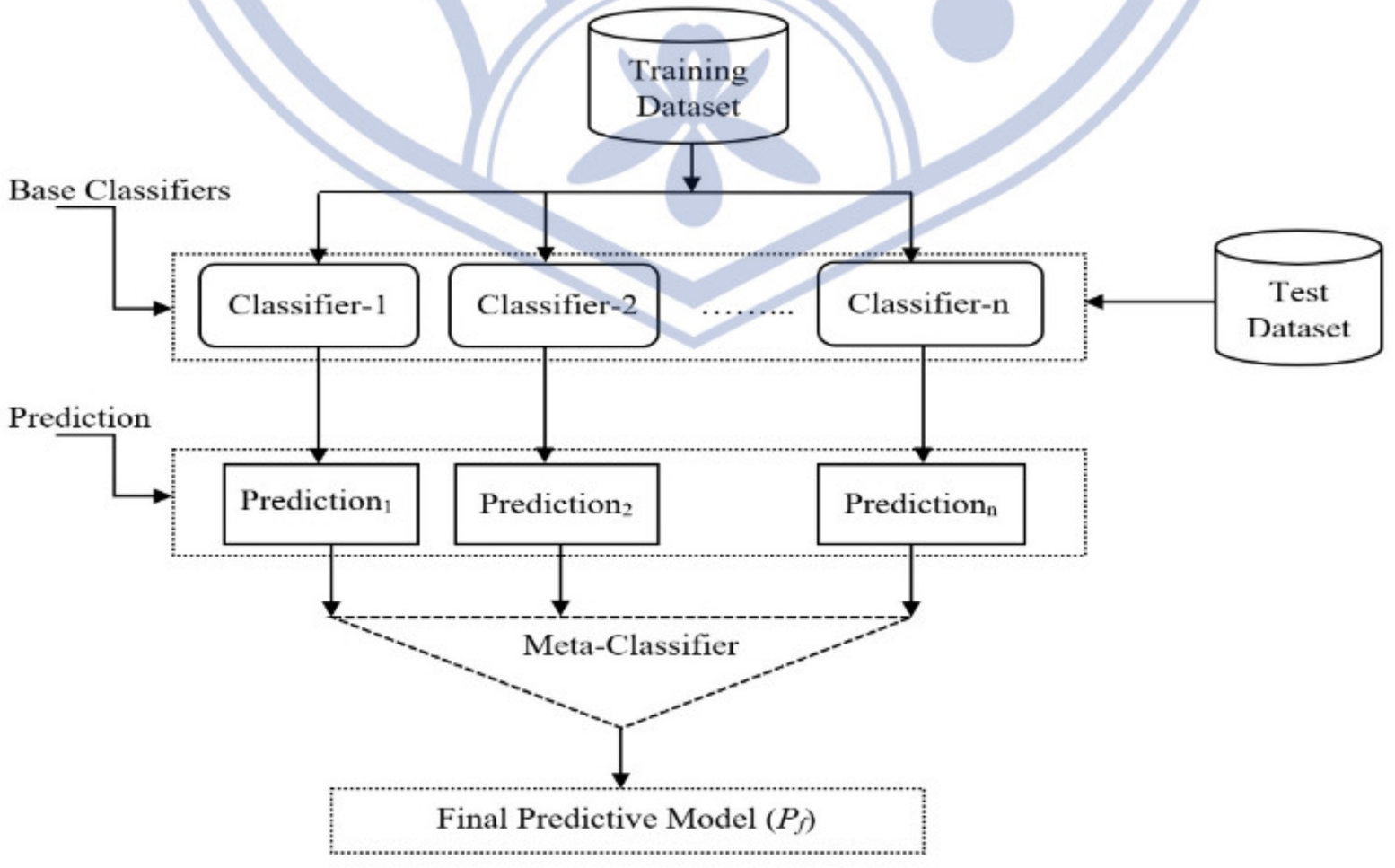
Stacking (atau *Stacked Generalization*) adalah teknik *ensemble learning* yang menggabungkan beberapa model dasar (*base learners*) dengan satu model *meta-learner* untuk menghasilkan prediksi akhir yang lebih akurat dan stabil [40]. Berbeda dengan *bagging* atau *boosting* yang menggabungkan prediksi secara sederhana (*voting* atau *weighted average*), stacking menggunakan *meta-learner* yang dilatih secara khusus untuk mempelajari cara terbaik menggabungkan prediksi dari *base learners* [40]. Pendekatan ini memanfaatkan kelebihan masing-masing *base learner* sekaligus mengurangi

kelemahan individu, sehingga menghasilkan performa yang lebih baik, terutama pada data yang kompleks atau tidak seimbang [40].

Mekanisme teknis *stacking* terdiri dari dua tahap utama [40]. Pada tahap pelatihan, data pelatihan digunakan untuk melatih beberapa *base learners* yang berbeda [40]. Untuk menghindari *overfitting*, *metadata* (prediksi dari setiap *base learner*) dihasilkan menggunakan metode *k-fold cross-validation*: data pelatihan dibagi menjadi k lipatan, setiap *base learner* dilatih pada k-1 lipatan dan memprediksi lipatan yang tersisa [40]. Hasil prediksi dari semua *base learners* digabungkan menjadi satu matriks *metadata* Z (dengan dimensi $n \times b$, di mana n adalah jumlah sampel dan b adalah jumlah *base learners*) [40]. *Metadata Z* inilah yang kemudian digunakan untuk melatih *meta-learner* [40].

Pada tahap pengujian, data uji diprediksi terlebih dahulu oleh semua *base learners* yang sudah dilatih [40]. Prediksi-prediksi tersebut digabungkan dan dimasukkan ke *meta-learner* yang telah dilatih sebelumnya [40]. Keluaran dari *meta-learner* inilah yang menjadi prediksi akhir dari keseluruhan *stacked ensemble* [40]. Dengan demikian, *meta-learner* bertindak sebagai "penilai akhir" yang mempelajari pola kesalahan dan kekuatan setiap *base learner,* sehingga menghasilkan keputusan yang lebih cerdas daripada sekadar rata-rata atau *voting* biasa [40].

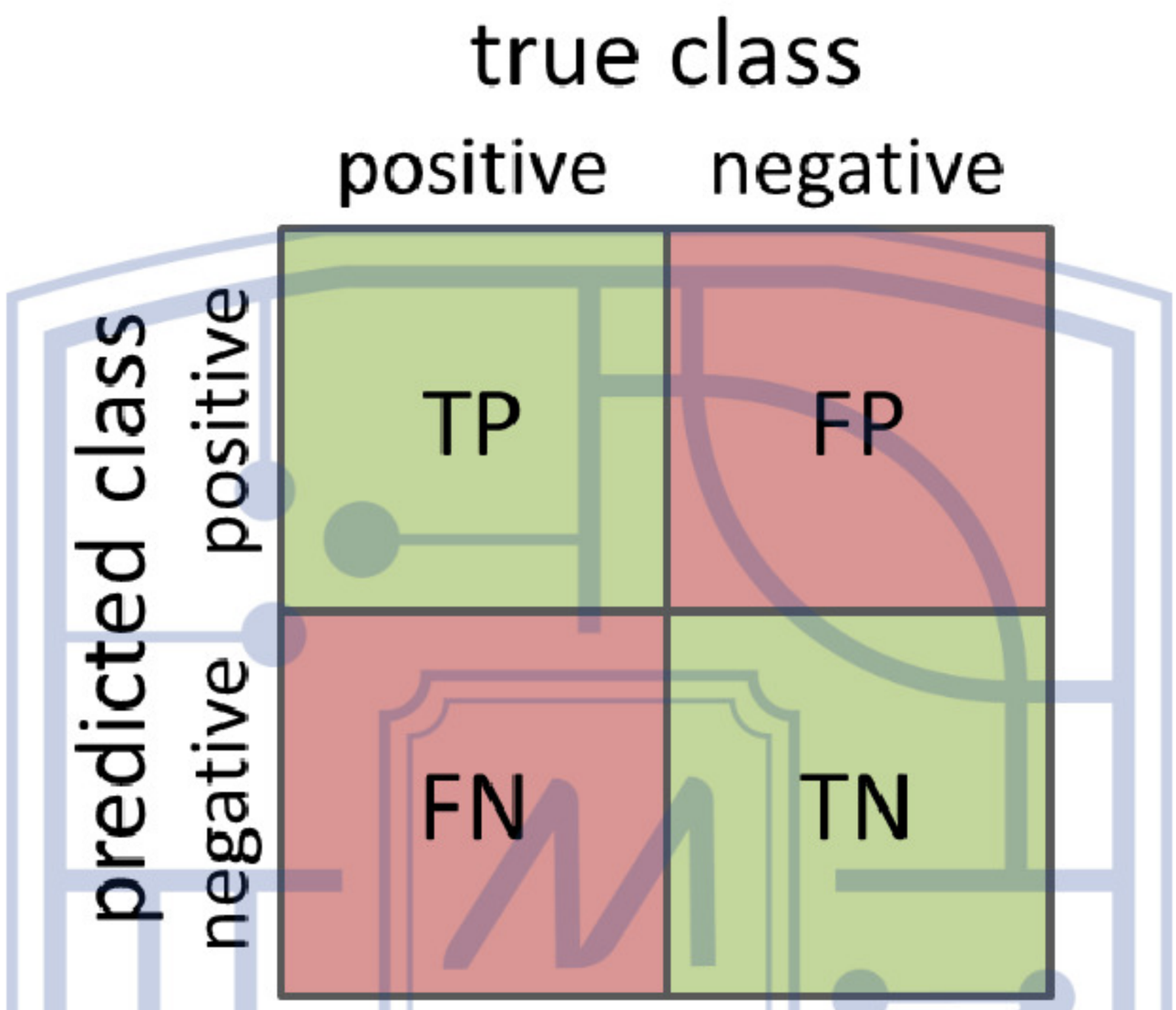
Gambar 2.1 di bawah ini mengilustrasikan konsep *stacking* secara keseluruhan. Diagram tersebut menunjukkan hubungan antara *base learners* pada tahap pertama dan *meta-learner* pada tahap kedua, serta alur pembentukan *metadata* melalui *cross-validation* hingga menghasilkan prediksi akhir.



Gambar 2.1 Konsep *Stacking* [40]

2.10 *Confusion Matrix*

Confusion matrix (error matrix) merupakan metode visualisasi yang memecah jumlah *instance ground truth* dari kelas tertentu terhadap jumlah *instance* kelas yang diprediksi [41]. Dalam *confusion matrix* untuk klasifikasi biner, kolom



Gambar 2.2 *Confusion Matrix* [42]

merepresentasikan nilai aktual (*ground truth*) sementara baris merepresentasikan nilai yang diprediksi [41]. Pada Gambar 2.2, kotak kiri atas menunjukkan *true positives (TP)*, kotak kanan atas *false positives (FP)*, kotak kiri bawah *false negatives (FN*), dan kotak kanan bawah *true negatives (TN)* [41].

True Positive (TP) berarti model memprediksi hasil positif dengan benar [11]. *True Negative* (TN) berarti model memprediksi hasil negatif dengan benar [41]. *False Positive (FP)* berarti model memprediksi hasil positif secara salah [41]. *False Negative (FN)* berarti model memprediksi hasil negatif secara salah [41]. Berdasarkan nilai-nilai dalam *confusion matrix* tersebut, metrik evaluasi kinerja model dapat dihitung secara langsung [41].

Berbagai metrik performa dapat dihitung langsung dari *confusion matrix* untuk menilai kualitas model, yaitu [41] :

1. *Accuracy* adalah rasio prediksi yang benar terhadap total instance

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(2.14)}$$

2. *Precision* adalah rasio prediksi positif yang benar terhadap semua prediksi positif

$$Precision = \frac{TP}{TP+FP} \quad \text{.....(2.15)}$$

3. *Recall* atau *sensitivity* adalah rasio prediksi positif yang benar terhadap semua instance aktual positif

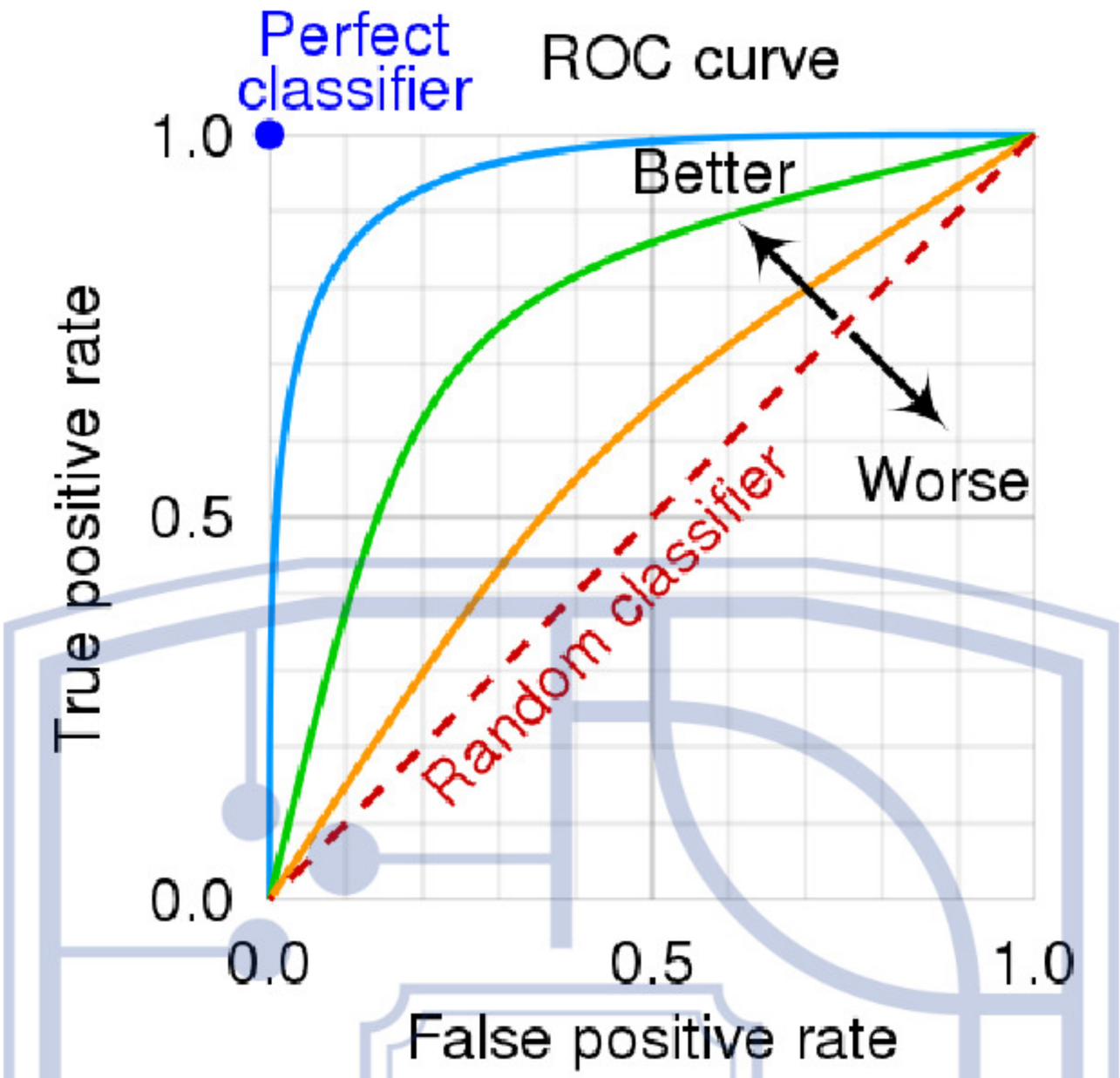
$$Recall = \frac{TP}{TP+FN} \quad \text{.....(2.16)}$$

4. *F1-score* merupakan rata-rata harmonik antara *precision* dan *recall*

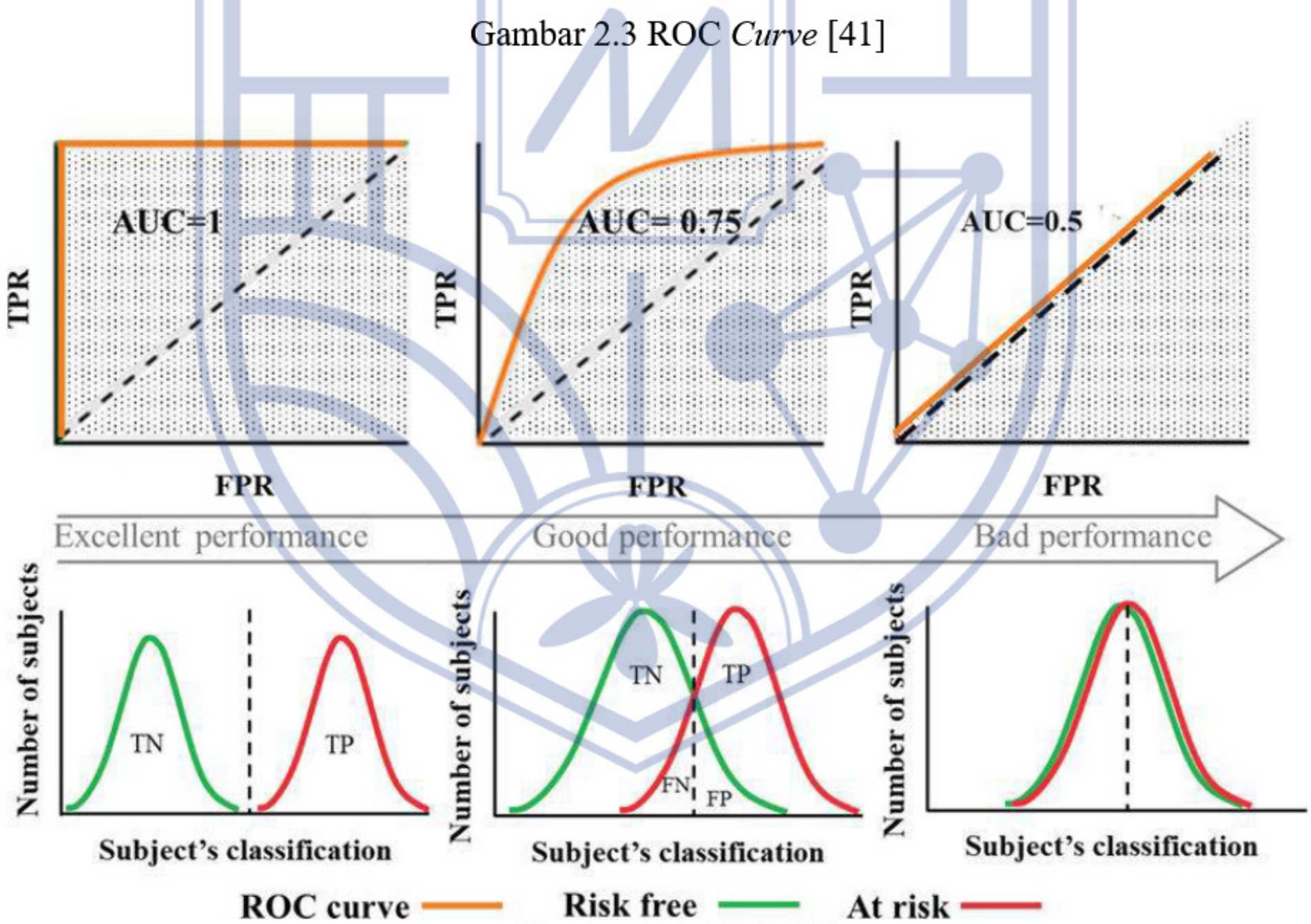
$$F1 - Score = 2 \times \frac{\text{Precision} \times \text{Recal}}{\text{Precision+Recal}} = \frac{2TP}{2TP+FP+FN} \quad \text{.....(2.17)}$$

Selain metrik di atas, evaluasi model juga dapat dilakukan melalui kurva *Receiver Operating Characteristic (*ROC*)* dan *Area Under the Curve (*AUC*)* [40]. ROC *curve* adalah grafik yang menunjukkan hubungan antara *True Positive Rate (sensitivity)* dan *False Positive Rate* pada berbagai nilai ambang batas [41]. AUC adalah luas area di bawah kurva ROC dan memberikan ukuran keseluruhan kemampuan model dalam membedakan antara kelas positif dan negatif [41]. Semakin mendekati 1 nilai AUC, semakin baik performa model, sedangkan nilai 0,5 menunjukkan model yang tidak lebih baik daripada tebakan acak [38].

Gambar 2.3 dan Gambar 2.4 berikut ini menyajikan visualisasi kurva ROC serta *Area Under the Curve* (AUC). Gambar 2.3 menampilkan kurva *Receiver Operating Characteristic* (ROC) yang menggambarkan *trade-off* antara *True Positive Rate* dan *False Positive Rate* pada berbagai tingkat performa model, mulai dari *perfect classifier* hingga *random classifier*. Sedangkan Gambar 2.4 mengilustrasikan nilai AUC beserta interpretasinya, termasuk perbandingan performa model (*excellent, good*, dan *bad performance*) serta distribusi kelas pada subjek.



Gambar 2.3 ROC *Curve* [41]



Gambar 2.4 AUC [41]