BAB 2
LANDASAN TEORI

2.1 Home Mortgage Disclosure Act

Home Mortgage Disclosure Act (HMDA) adalah undang-undang di Amerika Serikat yang mewajibkan lembaga keuangan untuk mengumpulkan, mencatat, dan melaporkan data tertentu tentang aplikasi, asal, dan pembelian pinjaman perumahan kepada publik. Tujuan utama HMDA adalah untuk meningkatkan transparansi dalam pasar kredit perumahan dan membantu dalam mengidentifikasi potensi diskriminasi dalam pemberian pinjaman [10]. Dengan adanya regulasi ini, data mengenai karakteristik pemohon, jumlah pinjaman, dan keputusan kredit dapat diakses untuk analisis lebih lanjut. Hal ini memungkinkan regulator dan masyarakat untuk mengamati pola pemberian kredit dan menilai kepatuhan lembaga keuangan terhadap kebijakan keadilan kredit.

Implementasi HMDA terkandung dalam Regulasi C yang ditetapkan oleh The Bureau of Consumer Financial Protection sebagai agensi pemerintahan Amerika Serikat yang melindungi konsumen dari ketidakadilan praktik keuangan. Regulasi C mengatur persyaratan bagi institusi keuangan terkait dengan pengumpulan, pencatatan, dan pelaporan data pinjaman perumahan [11]. Regulasi ini menetapkan standar pelaporan untuk memastikan bahwa informasi yang dikumpulkan dapat digunakan dalam analisis tren kredit dan kepatuhan terhadap regulasi anti-diskriminasi. Selain itu, Regulasi C mewajibkan institusi keuangan untuk menyusun *Loan/Application Register* (LAR) setiap kuartal guna memastikan akurasi dan transparansi data yang dilaporkan kepada otoritas pengawas.

Setiap institusi keuangan yang masuk dalam cakupan Regulasi C diwajibkan untuk mencatat data pinjaman hipotek mereka dalam LAR dalam waktu 30 hari setelah akhir setiap kuartal kalender [12]. Data yang harus dilaporkan mencakup informasi tentang identitas institusi keuangan, tahun kalender pelaporan, informasi kontak untuk pertanyaan terkait pelaporan, otoritas pengawas federal yang relevan, jumlah total entri dalam pengajuan data, *Taxpayer Identification Number* (TIN) institusi keuangan, dan *Legal Entity Identifier* (LEI) institusi. Dengan mekanisme ini, regulator dapat memantau apakah institusi keuangan telah memberikan pinjaman secara adil dan sesuai dengan kebijakan yang berlaku, serta mengidentifikasi kemungkinan adanya diskriminasi dalam akses kredit.

Catatan, informasi, dan laporan dari berbagai institusi keuangan dikumpulkan menjadi satu data HMDA, yang berisi detail informasi peminjam, jumlah pinjaman, jenis properti, serta faktor demografis seperti ras, jenis kelamin, dan pendapatan. Keunggulan data HMDA adalah kelengkapannya dalam mencakup berbagai faktor yang digunakan oleh bank dan lembaga keuangan dalam menentukan kelayakan kredit pemohon. Data ini digunakan secara luas dalam penelitian keuangan dan kecerdasan buatan untuk mengkaji tren pinjaman dan mengidentifikasi potensi diskriminasi dalam keputusan kredit.

2.2 Min-Max Scaling

Min-max scaling merupakan metode normalisasi data yang mengubah rentang nilai setiap fitur ke interval tertentu, umumnya [0, 1]. Proses ini bertujuan agar setiap nilai x direpresentasikan relatif terhadap nilai minimum dan maksimum dalam suatu fitur [13]. Teknik ini sangat efektif dalam mengurangi bias yang muncul akibat perbedaan skala antar fitur. Cara perhitungannya dapat disajikan melalui formula berikut.

Rumus 2.1 menunjukkan cara perhitungan nilai *Min-max scaling*.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.1}$$

di mana:

x : Nilai asli suatu fitur

x_{max} : Nilai maksimum dalam suatu fitur

x_{min} : Nilai minimum dalam suatu fitur

2.3 Standard Scaling

Standard scaling, atau *z-score normalization*, adalah metode normalisasi yang mengubah distribusi data sehingga memiliki rata-rata nol dan standar deviasi satu [13]. Metode ini berguna terutama ketika data mendekati distribusi normal, sehingga meminimalkan dominasi fitur dengan rentang nilai yang besar dalam proses pelatihan model. Rumus yang digunakan adalah sebagai berikut.

Rumus 2.2 menunjukkan cara perhitungan nilai *Standard scaling*.

$$Z = \frac{X - \mu}{\sigma} \tag{2.2}$$

di mana:

x : Nilai asli suatu fitur

μ : Nilai rata-rata suatu fitur

σ : Nilai simpangan baku suatu fitur

2.4 Recursive Feature Elimination

Recursive feature elimination (RFE) merupakan salah satu pendekatan seleksi fitur berbasis model yang tujuan utamanya adalah penyingkiran fitur dengan bobot terendah melalui proses iterasi [14]. Disebut berbasis model karena eliminasi fitur bergantung pada skor prediksi model dalam setiap iterasi, beserta pengecekan peringkat bobot fitur dalam pengaruhnya terhadap prediksi. Skor prediksi model bisa dipilih tergantung metrik evaluasi yang ingin dipakai, misalnya akurasi prediksi. Berikut adalah langkah utama dalam pendekatan RFE.

1. Melatih model yang dipakai dengan seluruh fitur yang tersedia.
2. Menghitung bobot atau skor pengaruh tiap fitur.
3. Mengurutkan bobot fitur dari nilai tertinggi dan kemudian mengeliminasi nilai-nilai terendahnya.
4. Latih ulang model yang dipakai dengan fitur terbaru setelah eliminasi.
5. Ulangi langkah ke-3 dan ke-4 hingga mendapatkan fitur optimal sesuai bobot pengaruhnya.

2.5 Information Value

Information value (IV) merupakan salah satu metode yang sering digunakan dalam proses seleksi fitur, khususnya untuk kasus klasifikasi biner. Metode ini bekerja dengan cara mengevaluasi seberapa baik suatu variabel prediktor dalam membedakan antara dua kelas target. Dalam praktiknya, pendekatan ini menentukan kontribusi setiap fitur dengan cara menganalisis distribusi data

terhadap target biner [14]. IV dihitung dengan membagi variabel x dalam beberapa interval atau bin, kemudian membandingkan proporsi kejadian antara kelas positif ($y = 1$) dan kelas negatif ($y = 0$) pada setiap bin. Rumus yang digunakan untuk menghitung IV adalah sebagai berikut.

Rumus 2.3 menunjukkan cara perhitungan nilai IV.

$$IV = \sum_{i=1}^{k} (p_i - q_i) \times \ln\left(\frac{p_i}{q_i}\right) \tag{2.3}$$

di mana:

k : Jumlah total bin

p_i : proporsi observasi dengan $y = 1$ pada bin ke-i

q_i : proporsi observasi dengan $y = 0$ pada bin ke-i

Interpretasi nilai IV menunjukkan bahwa semakin besar nilai IV suatu fitur, maka semakin kuat kemampuannya dalam memisahkan antara kelas positif dan negatif 2.1. IV sering dijadikan sebagai indikator dalam menilai kekuatan prediktif suatu fitur. Dengan demikian, IV dapat menjadi alat yang efektif untuk menyederhanakan model dengan mempertahankan hanya fitur-fitur yang memiliki pengaruh nyata terhadap variabel target. Berikut tabel yang menunjukkan nilai ambang batas kekuatan prediktif dari skor IV.

Tabel 2.1. Nilai ambang batas untuk skor IV.

Nilai IV	**Kekuatan Prediktif**
$< 0{,}02$	Tidak berguna
0,02 hingga 0,1	Lemah
0,1 hingga 0,3	Sedang
0,3 hingga 0,5	Kuat
$> 0{,}5$	Sangat kuat namun mencurigakan

2.6 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) adalah algoritma *ensemble* berbasis pohon keputusan yang dirancang untuk meningkatkan akurasi prediksi melalui pendekatan *boosting* [15]. Setiap pohon keputusan baru dibangun secara bertahap untuk mengoreksi kesalahan dari pohon sebelumnya. Prediksi akhir merupakan hasil penjumlahan dari semua pohon dalam model.

Rumus 2.4 menunjukkan perhitungan prediksi agregat XGBoost untuk sampel data ke-i.

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathscr{F} \tag{2.4}$$
$$\text{dengan } \mathscr{F} = \{f(x) = w_{q(x)}\}\{q : \mathbb{R}^m \longrightarrow T, w \in \mathbb{R}^T\}$$

di mana:

K : Jumlah total pohon

$f_k(x_i)$: Prediksi pohon keputusan ke-k untuk sampel x_i

Dalam meningkatkan akurasi prediksi dan mengendalikan kompleksitas model, XGBoost memperkenalkan fungsi objektif teregularisasi. Fungsi ini tidak hanya mengukur kesalahan antara prediksi model dengan label sebenarnya melalui fungsi *loss*, tetapi juga menambahkan penalti terhadap kompleksitas struktur model melalui komponen regularisasi. Pendekatan ini memungkinkan model untuk belajar secara aditif dan bertahap mengoreksi *residual* dengan tetap menjaga generalisasi. Fungsi *loss*, misalnya *mean squared error* untuk regresi atau *log loss* untuk klasifikasi, dihitung berbasis gradien dan digunakan untuk menentukan arah dan besar pembaruan bobot pada setiap iterasi. Fungsi objektif pada XGBoost dirumuskan sebagai berikut.

Rumus 2.5 menunjukkan fungsi objektif XGBoost.

$$\mathscr{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.5}$$

di mana:

n : Jumlah sampel dalam data

$l(\hat{y}_i, y_i)$: Fungsi *loss* yang mengukur selisih antara nilai aktual y_i dan prediksi $\hat{y}_i$

K : Jumlah total pohon

$\Omega(f_k)$: Fungsi regularisasi untuk pohon keputusan ke-k

2.7 1D Convolutional Neural Network

1D Convolutional Neural Network (1DCNN) adalah arsitektur *deep learning* yang dirancang untuk memproses data sekuensial, seperti deret waktu, sinyal audio, atau data spektral, dengan mengekstraksi fitur lokal melalui operasi

konvolusi. 1DCNN beroperasi pada vektor satu dimensi yang menjadikannya efisien untuk menangkap pola lokal dan dependensi temporal [16]. Arsitektur 1DCNN yang umum terdiri dari beberapa lapisan kunci, yaitu lapisan *convolutional 1D*, *max pooling 1D*, *average pooling 1D*, *flatten*, *dense* atau *fully connected*, dan *output*, yang masing-masing memiliki peran spesifik dalam mengekstraksi dan memproses fitur untuk menghasilkan prediksi yang akurat. Setiap lapisan memiliki peran dan karakteristik khusus dalam pemrosesan informasi, yang akan dijelaskan melalui poin-poin berikut.

- **Lapisan convolutional 1D**

  Lapisan *convolutional 1D* bertugas mengekstraksi pola lokal dari deret waktu input menggunakan *kernel* [17]. Prosesnya melibatkan operasi konvolusi antara bobot *kernel* dan input dari lapisan sebelumnya, lalu ditambahkan bias [18]. Secara matematis, proses konvolusi untuk neuron ke-k pada lapisan l dirumuskan sebagai berikut.

  Rumus 2.6 menunjukkan cara perhitungan lapisan *convolutional 1D*.

$$x_k^l = \sum_{i=1}^{N_{l-1}} con1D\left(w_{ik}^{l-1}, s_i^{l-1}\right) + b_k^l \tag{2.6}$$

  di mana:

  N_{l-1} : Jumlah neuron di lapisan $l-1$
  w_{ik}^{l-1} : *Kernel* dari neuron ke-i lapisan $l-1$ ke neuron ke-k di lapisan l
  s_i^{l-1} : Keluaran dari neuron ke-i di lapisan $l-1$
  b_k^l : Bias untuk neuron ke-k di layer l

- **Lapisan pooling**

  Lapisan *pooling* bertujuan melakukan reduksi dimensi dengan cara mengambil nilai tertentu dari setiap jendela input yang telah ditentukan sehingga mengurangi kompleksitas komputasi [18]. Dua metode *pooling* (*downsampling*) yang utama adalah *max* dan *average pooling* [14]. Proses ini dinotasikan sebagai berikut.

  Rumus 2.7 menunjukkan cara perhitungan lapisan *pooling*.

$$s_k^l = y_k^l \downarrow ss \tag{2.7}$$

di mana:

y_k^l : Input dari lapisan sebelumnya

ss : Metode *downsampling*

- **Lapisan flatten**

  Lapisan *flatten* berfungsi sebagai jembatan antara lapisan ekstraksi fitur (konvolusi dan *pooling*) dengan lapisan klasifikasi atau regresi pada akhir CNN [16]. Lapisan ini mengubah data multidimensi, misalnya keluaran dengan bentuk (5, 4) menjadi vektor satu dimensi 20 elemen tanpa mengubah nilai-nilai aslinya, sehingga seluruh informasi fitur tetap terjaga.

- **Lapisan dense**

  Lapisan *dense* (atau *fully connected*) merupakan lapisan di mana setiap neuron menerima input dari seluruh neuron di lapisan sebelumnya melalui bobot terhubung penuh. Proses komputasinya dilakukan dengan mengalikan nilai keluaran dari lapisan sebelumnya dengan bobot (w_{ij}), menjumlahkannya, lalu menambahkan bias dan melewatkannya ke fungsi aktivasi nonlinier [14]. Lapisan ini berfungsi mengintegrasikan fitur yang telah diekstrak dari lapisan sebelumnya dan biasanya diletakkan sebelum lapisan *output*.

  Rumus 2.8 menunjukkan cara perhitungan neuron ke-i.

$$y_i = f\left(\sum_{j=1}^{M} w_{ij}\, x_j\right) \tag{2.8}$$

  di mana:

  f : Fungsi aktivasi yang dipakai

  M : Jumlah neuron di lapisan sebelumnya

  w_{ij} : Bobot dari neuron ke-j ke neuron ke-i

  x_j : Keluaran dari neuron ke-j di lapisan sebelumnya

- **Lapisan output**

  Lapisan *output* adalah komponen final yang bertanggung jawab menghasilkan

prediksi atau klasifikasi berdasarkan fitur yang telah diproses sebelumnya. Untuk masalah klasifikasi biner, fungsi aktivasi *sigmoid* digunakan untuk memetakan keluaran ke rentang [0, 1] [16]. Berikut rumus fungsi aktivasi *sigmoid* yang diterapkan dalam lapisan *output*.

Rumus 2.9 menunjukkan cara perhitungan fungsi aktivasi *sigmoid* dalam lapisan *output* untuk klasifikasi biner.

$$\sigma(z) = \frac{1}{1+e^{-z}}, \text{ dengan } z = w\,x + b \tag{2.9}$$

di mana:

w : Bobot dari lapisan *dense* sebelumnya

x : Keluaran dari lapisan *dense* sebelumnya

b : Bias dari lapisan *dense* sebelumnya

Dalam tugas klasifikasi biner, fungsi *loss* seperti *binary cross-entropy* dipakai untuk mengukur perbedaan antara distribusi probabilitas yang diprediksi dan label sebenarnya. Pemakaiannya dalam proses *backpropagation* dapat memperbarui bobot jaringan untuk mengurangi kesalahan [19]. Dengan demikian, lapisan ini memastikan bahwa model memberikan hasil akhir yang jelas dalam bentuk klasifikasi sesuai dengan kebutuhan data input yang diolah.

Rumus 2.10 menunjukkan cara perhitungan fungsi *binary cross-entropy* dalam lapisan *output* untuk klasifikasi biner.

$$Loss = -y_j\,log(p_j) + (1-y_j)\,log(1-p_j) \tag{2.10}$$

di mana:

y_j : Label sebenarnya untuk sampel ke-j

p_j : Probabilitas prediksi untuk sampel ke-j

2.8 Model Hibrida 1DCNN-XGBoost

Model hibrida 1DCNN-XGBoost merupakan pendekatan inovatif yang menggabungkan keunggulan dari dua metode pembelajaran mesin dengan pendekatan *generalized stacking* untuk meningkatkan performa prediksi atau klasifikasi, yang salah satunya dalam sektor keuangan. Pada tahap awal, model

1DCNN berfungsi sebagai model dasar untuk mengekstrak fitur-fitur lokal dari data sekuensial. Proses ekstraksi fitur dilambangkan dengan *y* pada teori 1DCNN sebelumnya.

Fungsi dari 1DCNN ini adalah untuk menangkap pola *spasio-temporal* dalam data sehingga menghasilkan representasi fitur yang informatif. Fitur-fitur yang diekstrak kemudian digunakan sebagai input bagi model meta, yakni XGBoost, yang memiliki kemampuan unggul dalam mengatasi interaksi nonlinier antar fitur. Dengan demikian, integrasi kedua model ini menawarkan solusi yang dapat meningkatkan akurasi prediksi secara teoritis [2].

Mekanisme *generalized stacking* yang dipakai memungkinkan XGBoost untuk belajar dari *residual* atau kesalahan prediksi yang dihasilkan oleh model dasar 1DCNN. Secara matematis, prediksi akhir dapat dinyatakan sebagai simbol $\hat{y}_i$. Pendekatan ini mengombinasikan keunggulan representasi fitur lokal yang dihasilkan oleh 1DCNN dengan kemampuan XGBoost dalam memodelkan interaksi kompleks, sehingga menghasilkan model yang lebih kokoh dan dapat diandalkan.

2.9 Metrik Evaluasi Model

Evaluasi performa model merupakan langkah penting dalam menilai efektivitas suatu algoritma dalam melakukan klasifikasi atau prediksi. Berbagai metrik digunakan untuk mengukur seberapa baik model dalam melakukan klasifikasi atau prediksi data dengan benar, terutama dalam kasus data yang tidak seimbang. Beberapa metrik yang umum digunakan adalah *accuracy*, *precision*, *recall*, *f1-score*, *area under the curve* (AUC), *macro average*, *weighted average*, dan *h-score*. Masing-masing metrik memiliki keunggulan dan fokus yang berbeda dalam menilai kinerja model.

2.9.1 Precision

Precision mengukur jumlah benar dalam klasifikasi kelas positif [2]. Nilai *precision* yang tinggi menunjukkan bahwa model memiliki sedikit *false positive*. *Precision* sangat penting dalam sektor keuangan ini, di mana *false positive* dapat memiliki dampak besar dan perlu dipertimbangkan dengan hati-hati. Formula *precision* dinyatakan sebagai berikut.

Rumus 2.11 menunjukkan cara perhitungan nilai *precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2.11}$$

di mana:

TP : *True positive*, yakni jumlah klasifikasi kelas positif yang benar

FP : *False positive*, yakni jumlah klasifikasi kelas positif yang salah

2.9.2 Recall

Recall berguna untuk mengukur kemampuan model dalam menemukan semua sampel positif yang ada [2]. Metrik ini penting ketika mengutamakan *true positive*, sedangkan *false negative* perlu ditekan serendah mungkin karena dampaknya yang sangat merugikan atau bahkan fatal, seperti diagnosis penyakit. Nilai *recall* yang tinggi menandakan bahwa model mampu menangkap sebagian besar sampel positif dengan baik. Formula *recall* disajikan sebagai berikut.

Rumus 2.12 menunjukkan cara perhitungan nilai *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.12}$$

di mana:

TP : *True positive*, yakni jumlah klasifikasi kelas positif yang benar

FN : *False negative*, yakni jumlah klasifikasi kelas negatif yang salah

2.9.3 F1-score

F1-score adalah rata-rata harmonik antara *precision* dan *recall*, yang digunakan untuk menyeimbangkan kedua metrik tersebut [2]. *F1-score* juga berguna ketika data yang dipakai memiliki distribusi kelas yang tidak seimbang. *F1-score* yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara *precision* dan *recall* sehingga cocok dijadikan parameter utama dalam kasus keduanya perlu dipertimbangkan, misalnya deteksi spam email. Rumusnya adalah sebagai berikut.

Rumus 2.13 menunjukkan cara menghitung *f1-score*.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.13}$$

2.9.4 Area Under the Curve

Area under the curve (AUC) berguna untuk mengukur performa model berdasarkan kurva *receiver operating characteristic* (ROC). AUC dihitung sebagai luas di bawah kurva ROC, yang menggambarkan *trade-off* antara *true positive rate* (TPR atau *sensitivity*) dan *false positive rate* (FPR atau $1 - specificity$) [20]. AUC dirumuskan sebagai berikut [2].

Rumus 2.14 menunjukkan cara perhitungan nilai AUC.

$$AUC = \frac{1}{2}\left(1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}\right) \tag{2.14}$$

di mana:

TP : *True positive*, yakni jumlah klasifikasi kelas positif yang benar

TN : *True negative*, yakni jumlah klasifikasi kelas negatif yang benar

FP : *False positive*, yakni jumlah klasifikasi kelas positif yang salah

FN : *False megative*, yakni jumlah klasifikasi kelas negatif yang salah

2.9.5 Macro Average

Macro average digunakan untuk mengukur rata-rata metrik (*precision*, *recall*, *f1-score*) di setiap kelas tanpa memperhatikan jumlah sampel di setiap kelas. Rumus *macro average* adalah sebagai berikut [2].

Rumus 2.15 menunjukkan cara perhitungan nilai *macro average*.

$$macro - average = \frac{\sum_k SCORE_k}{N} \tag{2.15}$$

di mana:

$SCORE$: Metrik evaluasi (*precision*, *recall*, atau *f1-score*)

N : Jumlah kelas dalam data

2.9.6 Weighted Average

Weighted average merupakan metrik evaluasi yang mengukur rata-rata metrik (*precision*, *recall*, *f1-score*) di setiap kelas dengan mempertimbangkan

proporsi jumlah sampel di setiap kelas. Formula *weighted average* adalah sebagai berikut [2].

Rumus 2.16 menunjukkan cara perhitungan nilai *weighted average*.

$$weighted - average = \sum_{k} SCORE_k \; w_k \tag{2.16}$$

di mana:

$SCORE$: Metrik evaluasi (*precision*, *recall*, atau *f1-score*)

w_i : Proporsi jumlah sampel untuk kelas ke-i

2.9.7 H-score

H-score, atau *h-measure*, merupakan sebuah ukuran statistik yang diperkenalkan oleh Hand di tahun 2009 sebagai alternatif koheren dari AUC untuk mengevaluasi performa model klasifikasi biner [2]. *H-score* dirancang untuk mengatasi kelemahan mendasar AUC, yaitu faktor inkonsistensi dalam distribusi biaya kesalahan klasifikasi yang bergantung pada model klasifikasi itu sendiri. Berbeda dengan AUC, yang menggunakan distribusi biaya berbeda untuk setiap model (sehingga tidak adil dalam perbandingan), *h-score* menggunakan distribusi biaya tetap berbentuk Beta simetris dengan parameter $\alpha = \beta = 2$ atau $Beta(2,2)$ sebagai standar untuk memastikan semua model dievaluasi dengan metrik yang sama [20].

H-score mengintegrasikan minimum *loss* dari kesalahan klasifikasi dengan mempertimbangkan *trade-off* antara *sensitivity* dan *specificity*, lalu menerapkan standarisasi terhadap kerugian maksimum yang mungkin terjadi. Nilainya berada pada rentang 0 hingga 1, di mana 1 menunjukkan performa sempurna (tanpa kesalahan klasifikasi), sedangkan 0 menunjukkan kinerja setara dengan tebakan acak. Perhitungannya melibatkan pembobotan distribusi biaya yang konsisten, sehingga lebih mencerminkan konteks nyata di mana keseriusan kesalahan klasifikasi bersifat eksternal (tergantung masalah), bukan ditentukan oleh model. *H-score* dihitung menggunakan fungsi berikut.

Rumus 2.17 menunjukkan cara perhitungan *h-score*.

$$H = 1 - \frac{EC(hb_{classifier})}{EC(rd_{classifier})} \tag{2.17}$$

di mana:

$EC(hb_{classifier})$: Biaya ekspektasi dari model yang digunakan

$EC(rd_{classifier})$: Biaya ekspektasi dari model acak (*random classifier*)

2.10 SHapley Additive exPlanations

SHapley Additive exPlanations (SHAP) merupakan metode XAI yang diperkenalkan oleh Lundberg dan Lee pada tahun 2017 [21]. Metode ini dirancang untuk memberikan penjelasan baik secara lokal (tiap-tiap prediksi) maupun global (keseluruhan perilaku model) terhadap keluaran model pembelajaran mesin. Penjelasan melalui SHAP dapat memudahkan memahami kontribusi setiap fitur secara adil berdasarkan prinsip nilai *shapley* dari teori permainan [9].

SHAP menguraikan keluaran model dengan pendekatan *additive feature attribution*. Pendekatan ini berupaya menyederhanakan pemahaman hubungan input dan keluaran dari model pembelajaran mesin yang kompleks dengan mereduksi model tersebut menjadi bentuk linier yang mudah diinterpretasi [22]. Ide dasarnya adalah menyederhanakan model nonlinier (dianggap kotak hitam) menjadi model penjelasan sederhana.

Pendekatan ini memberikan solusi penjelasan tunggal yang menjamin bahwa penjelasan model yang dihasilkan bersifat adil, konsisten, dan dapat diinterpretasi. Kualitas interpretasi tersebut dipenuhi oleh SHAP melalui tiga properti penting dari *additive feature attribution*, yaitu *local accuracy*, *missingness*, dan *consistency* [9]. Berikut merupakan penjelasan untuk masing-masing properti yang dipenuhi oleh SHAP.

1. **Local accuracy**

   Local accuracy adalah prinsip yang menyatakan bahwa model penjelas (yaitu model sederhana yang digunakan untuk menjelaskan prediksi) harus menghasilkan nilai prediksi yang sama persis dengan model asli untuk input tertentu yang sedang dijelaskan [9].

2. **Missingness**

   Missingness merupakan properti yang menyatakan bahwa fitur yang tidak

tersedia atau tidak digunakan dalam input harus memiliki kontribusi nol terhadap prediksi [23]. Dalam konteks SHAP, jika sebuah fitur dianggap hilang atau tidak aktif dalam input yang disederhanakan, maka nilai kontribusi dalam model penjelas wajib bernilai nol.

3. **Consistency**

   Consistency adalah properti yang menjamin bahwa jika kontribusi suatu fitur terhadap prediksi meningkat atau tetap saat model berubah, maka nilai atribusi dari fitur tersebut dalam model penjelas tidak boleh menurun [9]. Dalam konteks SHAP, ini berarti bahwa ketika kita membandingkan dua model berbeda, misalnya model lama dan model baru, dan kita tahu bahwa suatu fitur memiliki pengaruh yang lebih besar dalam model baru, maka penjelasan yang diberikan untuk fitur tersebut harus mencerminkan peningkatan pengaruh tersebut [23]. Dengan demikian, properti ini mencegah adanya inkonsistensi logis dalam interpretasi, dan menjaga agar hasil penjelasan tetap mencerminkan perubahan nyata dalam perilaku model.

Ketiga properti tersebut menegaskan bahwa model penjelas benar-benar merepresentasikan cara kerja model prediksi yang sesungguhnya. Dengan begitu, akan didapat secara eksplisit perhitungan nilai *shapley* untuk setiap fitur dengan mengintegrasikan seluruh *subset* fitur yang mungkin dengan menggunakan koefisien berbasis faktor-faktorial sebagai bobot [22]. Berikut merupakan formula akhir SHAP.

Rumus 2.18 menunjukkan perhitungan nilai *shapley* untuk setiap fitur.

$$\phi_i(f,x) = \sum_{z' \subseteq X'} \frac{|z'|!\,(M-|z'|-1)!}{M!} \left[f_x(z') - f_x(z' \setminus i) \right] \tag{2.18}$$

di mana:

z' : *Subset* dari fitur yang berkontribusi

M : Total jumlah fitur

$f_x(z')$: Prediksi model dengan *subset* fitur z'

$f_x(z' \setminus i)$: Prediksi model tanpa fitur ke-i

$\phi_i(f,x)$: Nilai kontribusi fitur ke-i untuk prediksi $f(x)$