BAB II
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Dalam melakukan penelitian mengenai pembuatan model *credit scoring* menggunakan algoritma *eXtreme Gradient Boosting* (XGBoost), peneliti menggunakan penelitian-penelitian sebelunya yang bersumber dari berbagai jurnal sebagai informasi mengenai *credit scoring*. berikut ini adalah penjelasan mengenai penelitian sebelumnya secara detail.

Fan, S., Shen, Y., & Peng, S. [19] telah melakukan penelitian mengenai Credit Scoring dengan menggunakan metode XGBoost, Logistic Regression, Support Vector Machine (SVM), dan Group Method of Data Handling (GMDH). Dari keempat algoritma yang digunakan, nilai Area Under the Curve (AUC) yang didapatkan oleh XGBoost adalah yang terbaik. Dilihat dari nilai akurasinya, XGBoost mendapatkan akurasi sebesar 90.1%, Logistic Regression sebesar 70.1%, Support Vector Machine sebesar 77.4%, dan Group Method of Data Handling (GMDH) sebesar 75.1%.

Niu, B., Ren, J., & Li, X. [20] telah melakukan penelitian mengenai Credit Scoring dengan menggunakan metode LightGBM, AdaBoost, dan Random Forest. Algoritma LightGBM memiliki nilai klasifikasi yang terbaik, terlepas dari akurasi atau F1 Score dan AUC. Sebelum dikombinasikan dengan Social Network Information, nilai AUC model asli adalah 0.692, akurasinya adalah 65.50% dan F1 scorenya adalah 0.649. Setelah dikombinasikan dengan Social Network Information, nilai AUC-nya meningkat menjadi 0.711, akurasinya menjadi 66.22%, dan F1 score menjadi 0.659. Untuk algoritma AdaBoost, sebelum dikombinasikan dengan Social Network Information, nilai AUC, akurasi, dan F1 score masing-masing sebesar 0.681, 63.91%, dan 0.642. Setelah dikombinasikan dengan Social Network Information, nilai AUC, akurasi, dan F1 score masing-masing sebesar 0.697, 64.40%, dan 0.651. Untuk algoritma Random Forest, sebelum dikombinasikan dengan Social Network Information, nilai AUC, akurasi, dan F1 score masing-masing sebesar 0.674, 63.57%, dan 0.635. Setelah

dikombinasikan dengan Social Network Information, nilai AUC, akurasi, dan F1 score masing-masing sebesar 68,9%, 63.92%, dan 64,4%.

Yotsawat, W., Wattuya, P., & Srivihok, A. [21] telah melakukan penelitian mengenai Credit Scoring dengan menggunakan metode Cost-sensitive Neural Network Ensemble (CS-NNE). Pendekatan CS-NNE yang diusulkan meningkatkan kinerja prediksi dari *Neural Network* berdasarkan dataset kredit Thailand, dengan mencapai nilai *Area Under the Curve* (AUC), *Default Detection Rate* (DDR), dan G-Mean (GM) masing-masing sebesar 1.36%, 15.67%, dan 6.11%. Pendekatan CS-NNE dapat secara efektif memecahkan masalah ketidakseimbangan data telah mengungguli banyak model yang ada.

Li, H., Cao, Y., Li, S., Zhao, J., & Sun, Y. [25] telah melakukan penelitian mengenai evaluasi kredit dengan menggunakan 5 model yang berbeda yaitu XGBoost, Logistic Regression, Decision Tree, Random Forest, dan GBDT. Dari kelima model tersebut, XGBoost memperoleh nilai akurasi sebesar 93,70%, Logistic Regression 89,21%, Decision Tree 88,01%, Random Forest 92,49%, dan GBDT memiliki akurasi 92,79%. Dari kelima model tersebut, pada dataset yang sama model XGBoost memiliki akurasi nilai yang paling tinggi.

Hayashi, Y. [26]. Telah melakukan penelitian mengenai deep learning untuk Credit Scoring. Dengan menggunakan metode utama yaitu XGBoost. Dari penelitian tersebut, model XGBoost mendapatkan akurasi sebesar 89,35% untuk dataset Australia dan mendapatkan akurasi sebesar 77,5% untuk dataset jerman.

Tabel 2. 1 *Literatur Review*

Penulis	Tahun	Judul	Metode	Perbedaan
Fan, S., Shen, Y., & Peng, S. [19]	2020	*Improved ML-Based Technique for Credit Card Scoring in Internet Financial Risk Control*	XGBoost, *Logistic Regression, Support Vector Machine* (SVM), dan *Group Method of Data Handling* (GMDH)	Penelitian tersebut dengan penelitian yang dilakukan memiliki perbedaan yang terdapat pada dataset yang digunakan. Nilai parameter XGBoost yang digunakan, dan juga nilai akhir yang dihasilkan oleh metode

Penulis	Tahun	Judul	Metode	Perbedaan
				yang ada pada penelitian ini.
Niu, B., Ren, J., & Li, X. [20]	2019	*Credit Scoring Using Machine Learning by Combing Social Network Information: Evidence from Peer-to-Peer Lending*	LightGBM, AdaBoost, dan *Random Forest*	Penelitian ini memiliki perbedaan dengan penelitian yang dilakukan yaitu perbedaan pada metode yang digunakan dan juga perbedaan pada dataset yang digunakan
Yotsawat, W., Wattuya, P., & Srivihok, A. [21]	2021	*A Novel Method for Credit Scoring based on Cost-sensitive Neural Network Ensemble*	*Cost-sensitive Neural Network Ensemble* (CS-NNE)	Penelitian ini memiliki perbedaan dengan penelitian yang dilakukan yaitu perbedaan pada metode yang digunakan dan juga perbedaan pada dataset yang digunakan
Li, H., Cao, Y., Li, S., Zhao, J., & Sun, Y [25].	2020	*XGBoost Model and Its Application to Personal Credit Evaluation*	*XGBoost, Logistic Regression, Decision Tree, Random Forest, dan GBDT*	Penelitian tersebut dengan penelitian yang dilakukan memiliki perbedaan yang terdapat pada dataset yang digunakan. Nilai parameter XGBoost yang digunakan, dan juga nilai akhir yang dihasilkan oleh metode yang ada pada penelitian ini.
Hayashi, Y. [26]	2022	*Emerging Trends in Deep Learning for Credit Scoring: A*	*XGBoost*	Penelitian tersebut dengan penelitian yang dilakukan memiliki perbedaan yang terdapat pada dataset yang digunakan. Nilai

Penulis	Tahun	Judul	Metode	Perbedaan
		Review.		parameter XGBoost yang digunakan, dan juga nilai akhir yang dihasilkan oleh metode yang ada pada penelitian ini.

Pada studi literatur yang sudah dilakukan, yaitu dengan 5 jurnal referensi yang berbeda. Algoritma XGBooost yang digunakan pada penelitian tersebut memiliki akurasi yang tertinggi dibandingkan dengan algoritma yang lain seperti *Logistic Regression, Decision Tree, Random Forest, GBDT, Neural Network* (NN), *LightGBM, Support Vector Machine* (SVM). XGBoost memiliki akurasi yang paling baik untuk masalah *credit scoring*. Hal ini memperkuat alasan penulis untuk menggunakan algoritma XGBoost untuk menentukan *credit scoring*.

Perbedaan penelitian tersebut dengan yang dilakukan ada pada parameter XGBoost yang digunakan seperti *max_depth* dan *n_estimator*. Terdapat perbedaan juga pada dataset yang digunakan, sehingga akan berpengaruh juga pada nilai akurasi yang akan dihasilkan nantinya.

2.2 Dasar Teori

2.2.1 *Credit Scoring*

Credit scoring adalah suatu metode yang digunakan oleh lembaga keuangan, seperti bank dan lembaga peminjam lainnya, untuk menilai risiko kredit yang dimiliki oleh calon peminjam. Tujuan dari *credit scoring* adalah untuk mengukur seberapa kemungkinan seseorang akan membayar kembali pinjaman mereka secara tepat waktu berdasarkan data dan informasi yang tersedia. Metode ini sangat penting karena membantu lembaga keuangan dalam mengambil keputusan yang lebih memiliki informasi dan objektif dalam memberikan pinjaman, serta menentukan suku bunga yang sesuai.

Proses *credit scoring* melibatkan analisis mendalam terhadap berbagai faktor yang berkaitan dengan profil kredit calon peminjam. Faktor-faktor ini dapat mencakup riwayat pembayaran kredit sebelumnya, jumlah pinjaman

yang dimiliki, lama histori kredit, jenis-jenis kredit yang dimiliki, dan faktor lain seperti pendapatan dan status pekerjaan. Metode ini menggunakan algoritma matematis yang kompleks untuk mengolah data-data tersebut dan menghasilkan skor kredit yang mencerminkan risiko peminjam. Semakin tinggi skor kredit, semakin rendah risiko *default* atau keterlambatan pembayaran.

Metode *credit scoring* telah mengalami evolusi sejak diperkenalkan, dengan pemanfaatan teknologi dan data yang semakin berkembang. Awalnya, metode ini menggunakan pendekatan manual dan terbatas pada data yang terbatas pula. Namun, dengan kemajuan teknologi informasi dan perkembangan dunia digital, lembaga keuangan dapat mengakses lebih banyak data dan menerapkan algoritma yang lebih kompleks. Hal ini memungkinkan penilaian risiko kredit yang lebih akurat dan efisien.

Secara keseluruhan, *credit scoring* adalah alat penting bagi lembaga keuangan dalam mengelola risiko kredit. Ini membantu mereka dalam membuat keputusan yang lebih terinformasi dan berkurangnya risiko terhadap *default* atau keterlambatan pembayaran. Melalui analisis yang cermat terhadap profil kredit calon peminjam, metode ini memainkan peran sentral dalam membentuk sistem peminjaman yang berkelanjutan dan aman bagi semua pihak yang terlibat [22].

2.2.2 Pandas

Pandas adalah salah satu library paling populer dalam bahasa pemrograman Python yang digunakan untuk analisis data. Library ini memberikan alat yang kuat dan fleksibel untuk mengolah data, yang sangat diperlukan dalam ilmu data, analisis statistik, dan pengolahan data lainnya. Pandas menyediakan dua struktur data utama: Series dan DataFrame. Series adalah struktur data satu dimensi yang dapat menyimpan berbagai jenis data, seperti bilangan bulat, float, string, dan sebagainya. DataFrame adalah struktur data dua dimensi yang mirip dengan tabel dalam database atau spreadsheet, yang terdiri dari baris dan kolom yang dapat diberi label. Salah

satu keunggulan utama Pandas adalah kemampuannya untuk membaca dan menulis data dari berbagai format, seperti CSV, Excel, SQL databases, dan lainnya. Setelah data dimuat ke dalam DataFrame, Pandas memungkinkan pengguna untuk melakukan berbagai operasi seperti pemilihan kolom, pemfilteran baris, penggabungan, pengelompokan, dan agregasi data. Misalnya, DataFrame dapat digabungkan berdasarkan kolom tertentu, menghitung statistik ringkasan, dan membuat visualisasi data [23].

2.2.3 NumPy

Numpy adalah sebuah library dalam pemrograman Python yang dirancang untuk bekerja dengan array dan matriks secara efisien. Nama "Numpy" merupakan singkatan dari "Numerical Python". Pustaka ini memberikan dukungan untuk operasi matematika dan manipulasi data berukuran besar dengan cepat dan efisien. Numpy merupakan komponen utama dalam ekosistem ilmiah Python dan banyak digunakan dalam bidang ilmu data, pemrosesan gambar, pemodelan matematika, serta penelitian ilmiah.

Numpy memperkenalkan objek array multi-dimensi, yang disebut ndarray (n-dimensional array), yang memiliki beberapa keunggulan penting dibandingkan dengan struktur data list biasa di Python. Ndarray menyediakan operasi vektorisasi yang memungkinkan pengolahan array secara paralel dan efisien. Fitur ini memungkinkan pemrogram untuk melakukan operasi matematika kompleks pada array dengan sintaks yang lebih ringkas dan performa yang lebih baik.

Selain itu, Numpy juga menawarkan berbagai fungsi matematika, seperti trigonometri, statistik, linear algebra, transformasi Fourier, dan lainnya, yang mempermudah analisis data dan pemodelan matematika. Kombinasi kemampuan operasi array dan fungsi matematika membuat Numpy sangat berguna dalam berbagai aplikasi, termasuk pengolahan gambar, analisis statistik, pengolahan sinyal, dan simulasi numerik [23].

2.2.4 Matplotlib

Matplotlib adalah sebuah pustaka atau library pada bahasa pemrograman Python yang digunakan untuk membuat visualisasi data secara interaktif maupun statis. Tujuannya adalah untuk membantu pengguna dalam menghasilkan grafik, plot, diagram, dan berbagai jenis visualisasi lainnya dengan mudah. Matplotlib sangat populer di kalangan ilmuwan data, analis, peneliti, dan pengembang perangkat lunak karena kemampuannya yang kuat dalam menghasilkan visualisasi yang informatif dan menarik. Matplotlib menyediakan beragam jenis plot seperti scatter plot, line plot, bar plot, histogram, pie chart, dan lainnya. Pustaka ini memberikan kontrol yang detail terhadap berbagai elemen visual seperti warna, gaya garis, tampilan label, dan anotasi, sehingga memungkinkan pengguna untuk menyesuaikan visualisasi sesuai kebutuhan. Matplotlib juga mendukung pelabelan sumbu, penyusunan legenda, dan penambahan judul pada plot, sehingga membantu dalam memberikan konteks dan penjelasan yang lebih baik terhadap data yang ditampilkan. Salah satu fitur menarik dari Matplotlib adalah kemampuannya untuk menghasilkan visualisasi interaktif melalui pengintegrasian dengan pustaka lain seperti Jupyter Notebook. Pengguna dapat membuat plot yang dapat diubah secara dinamis, memberikan pandangan yang lebih mendalam terhadap data, serta memungkinkan eksplorasi yang lebih baik. [23].

2.2.5 *Scikit-Learn*

Sklearn, atau scikit-learn, merupakan sebuah pustaka (library) Python yang populer dalam bidang pembelajaran mesin (machine learning). Pustaka ini menyediakan beragam algoritma dan alat untuk mempermudah pengembangan model pembelajaran mesin, analisis data, dan eksplorasi data. Nama "scikit-learn" sendiri merupakan singkatan dari "Scientific Kit Learn," mencerminkan fokus pustaka ini pada ilmu pengetahuan dan pembelajaran mesin.

Sklearn menyediakan berbagai algoritma pembelajaran mesin seperti regresi, klasifikasi, pengelompokan, dan lain-lain. Ini termasuk algoritma populer seperti Regresi Linier, Support Vector Machine (SVM), Random Forest, K-Means, Principal Component Analysis (PCA), dan banyak lagi. Pustaka ini juga menawarkan alat untuk melakukan pra-pemrosesan data seperti pemrosesan skala, seleksi fitur, dan transformasi data.

Salah satu kelebihan utama Sklearn adalah antarmuka yang konsisten dan mudah digunakan. Ini memungkinkan pengguna untuk dengan cepat menguji berbagai algoritma dan mengoptimalkan parameter model. Selain itu, Sklearn menyediakan dokumentasi yang kaya, contoh kode, dan panduan yang membantu pengguna memahami konsep dan implementasi algoritma secara lebih mendalam.

Sklearn juga cocok untuk pemula dan praktisi berpengalaman dalam pembelajaran mesin. Pustaka ini membantu dalam memahami konsep dasar pembelajaran mesin melalui kode sumber yang terbuka dan dokumentasi yang baik. Bagi para peneliti dan praktisi, Sklearn menyediakan alat untuk mengimplementasikan model pembelajaran mesin dalam proyek-proyek yang lebih besar. [23].

2.2.6 XGBoost

XGBoost (Extreme Gradient Boosting) adalah sebuah algoritma machine learning yang terkenal karena kinerjanya yang unggul dalam berbagai tugas seperti klasifikasi, regresi, dan ranking. XGBoost merupakan implementasi yang ditingkatkan dari metode Boosting, yang merupakan teknik ensemble di mana model lemah digabungkan untuk membentuk model yang lebih kuat.

Salah satu ciri utama XGBoost adalah kemampuannya mengatasi masalah bias-variance trade-off dalam machine learning. Bias mengacu pada kesalahan akibat penyederhanaan model, sementara variance mengacu pada sensitivitas model terhadap variasi dalam data pelatihan. XGBoost mengatasi masalah ini dengan menggabungkan banyak pohon keputusan

yang relatif sederhana untuk membentuk model yang lebih kompleks. Proses ini mengurangi bias sambil membatasi varian sehingga mencegah overfitting.

XGBoost menggunakan pohon keputusan sebagai model dasarnya. Pohon keputusan adalah struktur pohon yang mengambil keputusan berdasarkan serangkaian aturan if-else pada fitur-fitur data. Namun, XGBoost menggabungkan pohon-pohon ini secara cerdas untuk meningkatkan performa. Proses ini dimulai dengan pohon pertama yang dibangun untuk memprediksi target. Kemudian, kesalahan dari prediksi ini digunakan untuk membangun pohon kedua. Proses ini berlanjut dengan membangun pohon-pohon tambahan yang masing-masing mencoba mengoreksi kesalahan prediksi sebelumnya.

XGBoost memiliki beberapa fitur kunci yang menyumbang pada keunggulannya. Pertama, algoritma ini menggabungkan pohon-pohon keputusan secara paralel dan memiliki penanganan yang baik terhadap missing values dalam data. Selain itu, XGBoost menerapkan regularisasi yang disesuaikan (customizable regularization) untuk mengurangi kompleksitas model dan mencegah overfitting. Fitur lainnya adalah penggunaan fungsi objektif yang dapat disesuaikan, yang memungkinkan penyesuaian algoritma untuk tugas-tugas khusus.

XGBoost juga memiliki pendekatan yang unik dalam meminimalkan fungsi kerugian (loss function). Fungsi kerugian ini mengukur sejauh mana prediksi model dari nilai sebenarnya. XGBoost mengoptimalkan fungsi kerugian ini dengan pendekatan gradient boosting, di mana setiap iterasi menyesuaikan model berdasarkan gradien fungsi kerugian terhadap prediksi sebelumnya. Pendekatan ini secara efektif mengarahkan model menuju solusi yang lebih baik seiring berjalannya iterasi [15].

XGBoost digunakan untuk maslah supervised learning di mana menggunakan data latih dengan fitur X_i untuk memprediksi variable target yaitu y_i. Model dan parameter model dalam supervised learning biasanya mengacu pada struktur matematika di mana prediksi y_i dibuat dari inputan

X_i. Contoh umum adalah model linear, di mana prediksi diberikan sebagai $\hat{y}_i = \sum_j \theta_j X_{ij}$, kombinasi linier fitur masukan bobot. Nilai prediksi dapat memiliki interpretasi yang berbeda, tergantung pasa tugasnya, yaitu regresi atau klasifikasi [28].

Dengan pilihan bijak untuk $\mathcal{Y}i$ dapat mengekspresikan berbagai tugas, seperti regresi, klasifikasi dan peringkat. Tugas melatih model adalah menemukan parameter terbaik θ yang paling sesuai dengan data pelatihan $\mathcal{X}i$ dan label $\mathcal{Y}i$. Untuk melatih model, perlu mendefinisikan fungsi tujuan untuk mengukur seberapa cocok model tersebut dengan data pelatihan. Karakteristik penting dari fungsi objektif adalah bahwa mereka terdiri dari dua bagian: training loss dan regularization:

$$obj(\theta) = L(\theta) + \Omega(\theta) \tag{2.1}$$

Di mana L adalah fungsi training loss dan **Ω** adalah istilah dalam regulization. Training loss mengukur seberapa prediktif model terkait dengan data training. L adalah kesalahan rata-rata, yang diberikan oleh:

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \tag{2.2}$$

Fungsi kerugian lain yang umum digunakan adalah loss logistic, yang akan digunakan untuk regresi logistic:

$$L(\theta) = [y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i) \ln\left(1 + e^{-\hat{y}_i}\right)] \tag{2.3}$$

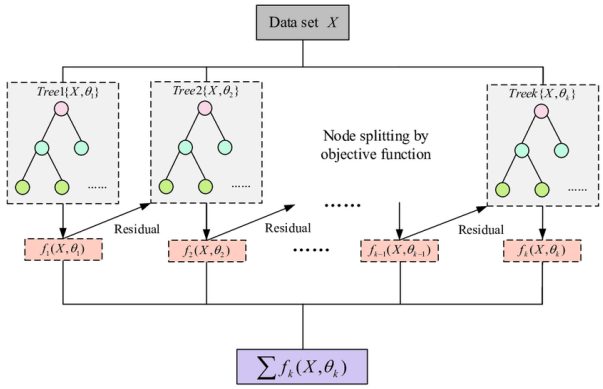
Istilah regularization adalah istilah yang biasanya lupa ditambahkan. Istilah regularization mengontrol kompleksitas model, yang membantu untuk menghindari overfitting.

Pada gambar 2. 1 adalah contoh flowchart dari XGBoost. Skor prediksi dari semua pohon dijumlahkan untuk mendapatkan skor akhir. $\{R_1(X_iY_i), \dots, R_k(X_iY_i)\}$ dan klasifikasi C di mana X_i dan Y_i adalah label data training dan class. Skor prediksi dievaluasi dengan menggunakan fungsi aditif k sebagai berikut:

$$\hat{y} = \sum_{k=1}^{c} f_k(x_i), f_k \in F \tag{2.4}$$

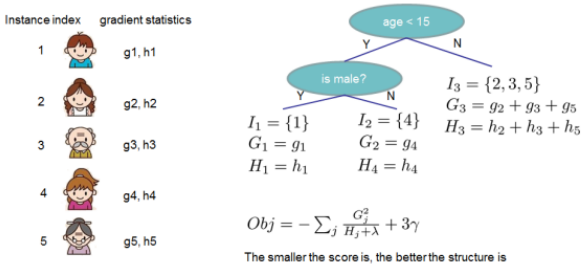
Di mana f_k adalah struktur pohon independent dengan skor daun dan $F = \{f(x) = w_{q(x)}\}(q : R^m \rightarrow T, w \in R^T$ adalah ruang pohon regresi

juga dikenal sebagai CART. Di sini q mewakili struktur setiap pohon yang memetakann contoh ke index daun yang sesuai. T adalah jumlah daun di pohon. Setiap f_x sesuai dengan struktur pohon independent q dan bobot daun w. Tidak seperti decision tree, setiap pohon regresi berisi skor terus menerus pada setiap daun, menggunakan w_i untuk mewakili skor pada daun ke-1. Sebagai contoh, kita akan menggunakan sturan keputusan di pohon (diberikan oleh q) untuk mengklasifikasikan.



Gambar 2. 1 Flowchart Algoritma XGBoost [15]

Pada gambar 2. 2 adalah perhitungan skor struktur. Dengan menjumlahkan gradien dan statistic gradien uritan kedua pada setiap daun, lalu menerapkan rumus penilaian untuk mendapatkan skor kualitas.



Gambar 2. 2 Struktur skor kualitas [28]

Dari gambar 2. 2 di atas dapat dilihat bagaimana skor dapat dihitung. Pada dasarnya, untuk struktur pohon tertentu, mendorong statistika g_i dan h_i ke daunnya, menjumlahkan statistikanya dan menggunakan rumus untuk menghitung seberapa bagus pohon tersebut. Skor ini seperti ukuran ketidakmurnian dalam decision tree, kecuali bahwa itu juga

memperhitungkan kompleksitas model. Akhirnya mendapatkan informasi dari fungsi objektif setelah setiap pemisahan adalah:

$$L_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda}\right] - Y \quad (2.5)$$

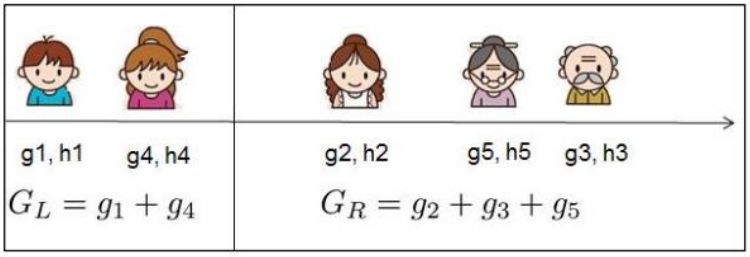
Seperti yang dapat di lihat pada (2.5), untuk menekan pertumbuhan pohon dan mencegah model overfitting, ditambahkan ambang pemisah γ. Node daun diizinkan untuk membagi jika dan hanya jika informasi yang diperoleh lebih besar dari γ. Ini sama dengan pra-penetapan nilai pohon sambil mengoptimalkan fungsi objektif, Selain itu, terdapat dua teknik terbaik XGBoost berikut untuk menghindari overfitting dalam percobaan:

1. Jika semua bobot sampel pada node daun kurang dari ambang batas, pemisah dihentikan. Ini mencegah model dari mempelajari sampel pelatihan khusus.
2. Fitur sampel secara acak ketika membangun setiap pohon.

Semua metode ini membuat XGBoost lebih di generalisasikan dan mendapatkan kinerja yang lebih baik dalam aplikasi praktis. Sekarang memiliki cara untuk mengukur seberapa bagus pohon itu, idealnya akan menghitung semua pohon yang mungkin dan memilih yang terbaik. Dalam praktiknya, hal ini sulit dilakukan, jadi mencoba mengoptimalkan satu tingkat pohon pada satu waktu. Secara khusus dengan mencoba membelah daun menjadi dua daun, dan skor yang diperolehnya adalah:

$$Gain = \frac{1}{2}\left[\frac{{G^2}_L}{H_L+\lambda} + \frac{{G^2}_R}{H_R+\lambda} + \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - Y \quad (2.6)$$

Rumus ini dapat diuraikan sebagai 1) skor pada daun kiri baru 2) skor pada daun kanan baru 3) skor pada daun asli 4) regularisasi pada daun tambahan. dapat melihat fakta penting di sini: jika gain lebih kecil dari γ, sebaiknya tidak menambahkan cabang tersebut. Inilah teknik pemangkasan dalam model berbasis pohon dengan menggunakan prinsip-prinsip supervised learning. Untuk data bernilai nyata, biasanya dicari pemisahan yang optimal.



Gambar 2. 3 Struktur Blok [28]

Untuk melakukannya secara efisien, dengan menempatkan semua kelas data dalam urutan yang diurutkan, seperti Gambar 2. 3 skema pemilihan pemisahan terbaik pemindaian kiri ke kanan cukup untuk menghitung skor struktur dari semua solusi pemisahan yang mungkin dan dapat menemukan pemisahan terbaik secara efisien.

Algoritma XGBoost memiliki hyperparameter yang membutuhkan penyetelan untuk kinerja algoritma yang optimal. Dampak dari penggunaan optimasi hyperparameter pada kinerja model klasifikasi sudah terbukti keduanya secara teoritis dan empiris oleh banyak penelitian yang terdapat dalam literatur. Beberapa parameter utama untuk meningkatkan elgoritma XGBoost yaitu:

1. *n_estimator*

   n_estimator adalah jumlah iterasi dalam training. *n_estimator* yang terlalu kecil dapat menyebabkan *underfitting*, yang membuat model tidak sepenuhnya melakukan kemampuan pelatihannya. Namun, *n_estimator* yang terlalu besar juga biasanya tidak bagus karena dapat menyebabkan *overfitting*.

2. *learning_rate*

   learning_rate adalah parameter yang sangat penting yang perlu di sesuaikan, juga dalam XGBoost. Ini sangat mempengaruhi kinerja model. Kita bias mengurangi bobot setiap langkah untuk membuat model lebih kuat.

3. *max_depth*

   max_depth adalah kedalaman *maximum* pohon. Semakin besar kedalaman pohon, semakin kompleks model pohonnya, dan semakin

kuat kemampuan pemasangannya, tetapi pada saat yang sama, modelnya jauh lebih mudah untuk dikenakan.

4. *min_child_weight*

   min_child_weight untuk menentukan sampel node daun terkecil untuk mencegah terjadinya *overfitting*.

Grid search adalah salah satu metode yang paling umum digunakan untuk optimasi hyperparameter, ini adalah jenis teknik brute force. Idenya adalah untuk mencari melalui subset dari hyperparameter untuk algoritma machine learning yang bersangkutan. Sehingga ruang pencarian ditentukan dan tujuannya adalah untuk mengoptimalkan kinerja algoritma machine learning. Ini adalah metode yang sangat sederhana tetapi dengan peningkatan hyperparameter, biaya komputasi meningkat secara eksponensial. Beberapa keuntungan lainnya dari metode grid search adalah kemudahan implementasi dan paralelisme [29].

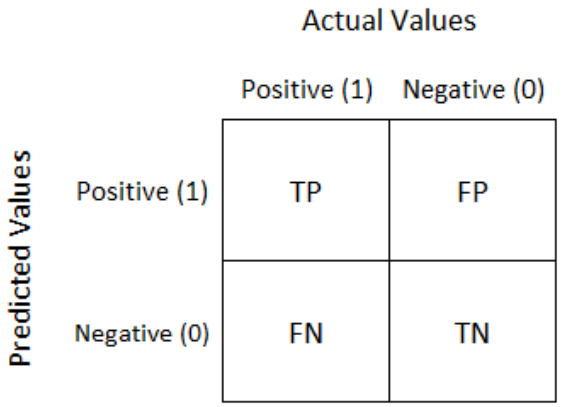
2.2.7 *Confusion Matrix*

Confusion matrix adalah alat penting dalam evaluasi kinerja model klasifikasi dalam pembelajaran mesin. Ini adalah tabel yang digunakan untuk menggambarkan performa prediksi model dengan membandingkan hasil prediksi dengan kelas sebenarnya dari data uji. Confusion matrix membantu kita memahami sejauh mana model dapat mengklasifikasikan data dengan benar dan menganalisis jenis kesalahan yang dibuat oleh model tersebut [24]. Confusion matrix terdiri dari empat istilah utama:

1. True Positive (TP): Ini mengacu pada jumlah instance yang diklasifikasikan dengan benar sebagai positif oleh model, ketika kelas sebenarnya adalah positif.
2. True Negative (TN): Ini merujuk pada jumlah instance yang diklasifikasikan dengan benar sebagai negatif oleh model, ketika kelas sebenarnya adalah negatif.

3. False Positive (FP): Ini adalah jumlah instance yang salah diklasifikasikan sebagai positif oleh model, padahal seharusnya negatif. Juga dikenal sebagai kesalahan jenis I.
4. False Negative (FN): Ini adalah jumlah instance yang salah diklasifikasikan sebagai negatif oleh model, padahal seharusnya positif. Juga dikenal sebagai kesalahan jenis II.

Pada Gambar 2. 2 memperlihatkan ilustrasi dari confusion matrix.



Gambar 2. 4 Ilustrasi *confusion matrix* [24]

Akurasi merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan rumus akurasi pada persamaan 1 sebagai berikut:

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (2.7)$$

Recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Dengan rumus recall pada persamaan 2 sebagai berikut:

$$Recall = \frac{TP}{TP+FN} \quad (2.8)$$

Precision merupakan rasio perdiksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Dengan rumus precision pada persamaan 3 sebagai berikut:

$$Precision = \frac{TP}{TP+FP} \quad (2.9)$$

F1-Score merupakan perbandingan rata-rata presisi dan recall yang dibobotkan. Dengan rumus F1-Score pada persamaan 4 sebagai berikut:

$$F1Score = \frac{2*(Recall*Precision)}{(Recall+Precision)} \quad (2.10)$$

2.2.8 *Class Imbalance*

Ketidakseimbangan kelas (*class imbalance*) adalah kondisi di mana distribusi jumlah sampel di antara kelas-kelas yang berbeda dalam sebuah dataset tidak proporsional atau seimbang. Dalam konteks klasifikasi, kelas mayoritas memiliki jumlah sampel yang jauh lebih banyak daripada kelas minoritas. Fenomena ini umum terjadi dalam berbagai bidang seperti deteksi penipuan, deteksi penyakit langka, analisis sentimen, dan lain-lain. Namun, ketidakseimbangan kelas dapat menyebabkan masalah serius dalam kinerja model pembelajaran mesin.

Penyebab ketidakseimbangan kelas bisa bervariasi. Misalnya, dalam aplikasi deteksi penyakit langka, jumlah pasien yang sehat mungkin jauh lebih banyak daripada pasien yang sebenarnya terinfeksi. Hal ini mengakibatkan model cenderung lebih memilih untuk mengklasifikasikan sampel sebagai kelas mayoritas, mengabaikan kelas minoritas. Dampak negatif dari ketidakseimbangan ini termasuk penurunan akurasi, presisi, dan recall pada kelas minoritas [27]. Pada kasus dataset dengan dua kelas, perhitungan Imbalance Ratio (IR) bisa dijabarkan sebagai berikut:

$$IR = \frac{Jumlah\ sampel\ kelas\ mayoritas}{Jumlah\ sampel\ kelas\ minoritas} \quad (2.11)$$

Tabel 2. 2 Kategori Tingkat Ketidakseimbangan Data

Derajat Ketidak Seimbangan Data (IR)	Proporsi Kelas Minoritas
Extreme	<1% dari seluruh data
Moderate	1-20% dari seluruh data
Mild	20-50% dari seluruh data
Slight	>50% dari seluruh data