BAB 2

TINJAUAN PUSTAKA

2.1 *Data Mining*

Data dalam jumlah besar yang terus terakumulasi sering kali menyimpan informasi dan pola tersembunyi yang sangat berharga. Namun, besarnya volume data membuat analisis manual menjadi tidak mungkin dilakukan. Di sinilah peran *data mining* menjadi sangat penting. *Data mining* adalah proses penemuan pola, anomali, dan korelasi yang menarik dan bermanfaat dari kumpulan data berskala besar untuk memprediksi hasil di masa depan (Han *et al*., 2022).

Tujuan utama dari *data mining* adalah untuk mengubah data mentah (*raw data*) menjadi pengetahuan yang dapat ditindaklanjuti (*actionable knowledge*). Proses ini tidak hanya sekadar mengekstraksi data, tetapi juga melibatkan penggunaan teknik dari disiplin ilmu lain seperti statistika, kecerdasan buatan (*artificial intelligence*), dan *machine learning* untuk mengidentifikasi tren yang sebelumnya tidak diketahui (Tan *et al*., 2019). Dengan menemukan pola-pola tersebut, sebuah organisasi dapat memperoleh wawasan strategis, meningkatkan efisiensi operasional, dan membuat keputusan yang lebih baik berdasarkan bukti data.

Secara fungsional, tugas-tugas dalam *data mining* dapat dikategorikan menjadi dua jenis utama: prediktif dan deskriptif. Tugas prediktif bertujuan untuk memprediksi nilai dari suatu atribut tertentu berdasarkan nilai dari atribut lainnya, sedangkan tugas deskriptif bertujuan untuk menemukan pola yang menggambarkan

data dan dapat ditafsirkan oleh manusia (Tan *et al*., 2019). Beberapa tugas utama dalam *data mining* yaitu:

a. Klasifikasi (*Classification*)

Klasifikasi adalah salah satu tugas prediktif yang paling umum. Klasifikasi bertujuan untuk membangun sebuah model yang dapat memetakan suatu objek data ke dalam salah satu dari beberapa kelas yang telah ditentukan sebelumnya. Model ini dibangun berdasarkan analisis dari sekumpulan data latih yang label kelasnya sudah diketahui. Contoh penerapannya adalah mengklasifikasikan email sebagai "*spam*" atau "bukan *spam*", atau menentukan apakah seorang nasabah bank berisiko tinggi atau rendah untuk kredit macet (Han *et al*., 2022).

b. Regresi/Prediksi (*Regression*/*Prediction*)

Serupa dengan klasifikasi, regresi juga merupakan tugas prediktif. Perbedaannya terletak pada *output* yang dihasilkan. Jika klasifikasi memprediksi label kelas yang bersifat kategoris (diskrit), maka regresi memprediksi nilai yang bersifat kontinu (numerik). Contohnya adalah memprediksi harga sebuah rumah berdasarkan luas bangunan, jumlah kamar, dan lokasi, atau meramalkan angka penjualan produk pada kuartal berikutnya (Zaqy *et al*., 2023).

c. *Clustering*

Berbeda dengan klasifikasi dan regresi, *clustering* atau klasterisasi adalah tugas deskriptif. Tujuannya adalah untuk mengelompokkan sekumpulan objek data ke dalam beberapa grup atau *cluster* sedemikian rupa sehingga objek-objek dalam satu *cluster* memiliki tingkat kemiripan yang tinggi, sementara objek-

objek di *cluster* yang berbeda memiliki tingkat kemiripan yang rendah. Dalam *clustering*, label kelas dari data tidak diketahui sebelumnya. Contoh aplikasinya adalah segmentasi pelanggan berdasarkan pola pembelian untuk strategi pemasaran yang lebih tertarget (Tan *et al*., 2019).

2.2 Klasifikasi Probabilistik (*Probabilistic Classification*)

Klasifikasi probabilistik adalah salah satu pendekatan inti dalam *machine learning* yang bertujuan untuk memetakan data masukan ke label kelas berdasarkan probabilitas (Murphy, 2022). Berbeda dari klasifikasi deterministik yang hanya menghasilkan satu label kelas tunggal sebagai *output*, klasifikasi probabilistik bekerja dengan mengestimasi distribusi probabilitas untuk seluruh kelas yang memungkinkan bagi setiap data masukan (Vaddella & Hosseinzadeh, 2021). Pendekatan ini secara fundamental tidak hanya menjawab pertanyaan "data ini termasuk kelas apa?", tetapi juga "seberapa besar kemungkinan data ini termasuk dalam setiap kelas?". Kemampuan untuk mengukur ketidakpastian (*quantifying uncertainty*) ini merupakan keunggulan utamanya, yang menjadikannya sangat krusial dalam domain yang memerlukan pengambilan keputusan berbasis risiko, seperti pada diagnosis medis, penyaringan spam, dan penilaian kredit (Murphy, 2022).

Fondasi matematis dari sebagian besar model klasifikasi probabilistik adalah Teorema Bayes. Teorema ini menyediakan cara untuk memperbarui keyakinan (*belief*) kita terhadap sebuah hipotesis berdasarkan bukti (*evidence*) baru yang diperoleh. Dalam konteks klasifikasi, tujuannya adalah untuk menemukan

probabilitas kelas *y* setelah kita mengamati data masukan *x*, yang dikenal sebagai probabilitas posterior $P(y \mid x)$. Rumus Teorema Bayes dinyatakan pada persamaan (2.1) (Russell & Norvig, 2020).

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} \tag{2.1}$$

Fondasi dari klasifikasi probabilistik terletak pada perhitungan probabilitas posterior $P(y \mid x)$, yaitu probabilitas sebuah data *x* benar-benar termasuk dalam kelas *y* setelah data tersebut diamati. Untuk menghitungnya, Teorema Bayes memanfaatkan tiga komponen utama. Pertama adalah *likelihood* $P(x \mid y)$, yang merepresentasikan probabilitas mengamati data *x* dengan asumsi berasal dari kelas *y*, di mana komponen ini dimodelkan dari data latih. Kedua adalah probabilitas *prior* $P(y)$, yang merupakan probabilitas awal dari setiap kelas sebelum data diobservasi, sering kali didasarkan pada frekuensi kelas dalam data latih. Komponen terakhir adalah *evidence* $P(x)$, yang berfungsi sebagai faktor normalisasi untuk memastikan total probabilitas posterior di semua kelas berjumlah satu.

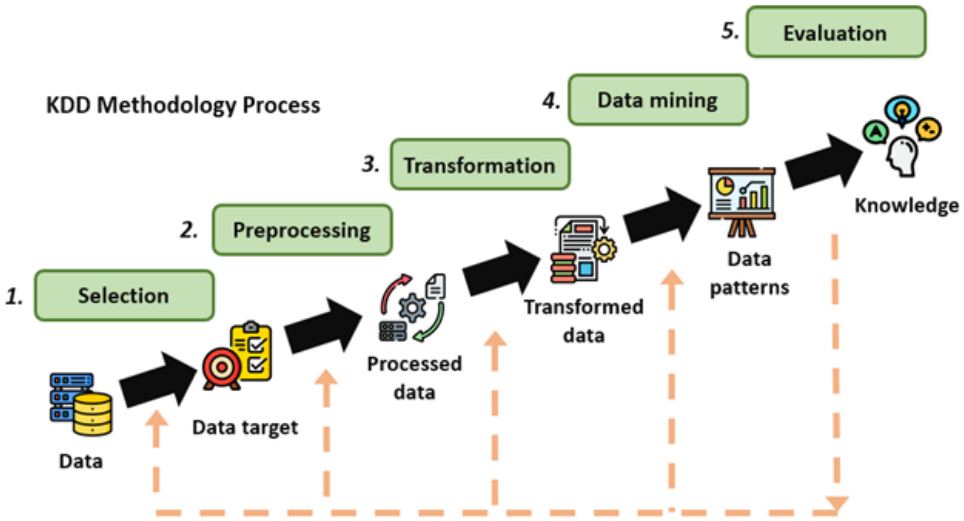
Berdasarkan cara pemodelan komponen-komponen tersebut, pendekatan klasifikasi probabilistik dapat dibagi menjadi dua paradigma utama: generatif dan diskriminatif. Model generatif secara eksplisit memodelkan *likelihood* $P(x \mid y)$ dan *prior* $P(y)$, sehingga secara konseptual mampu "membangkitkan" data baru untuk setiap kelas. Karena faktor *evidence* $P(x)$ bernilai sama untuk semua kelas, prediksi seringkali disederhanakan menjadi $argmax_y P(x \mid y)P(y)$. Sebaliknya, model diskriminatif tidak memodelkan distribusi data asli, melainkan langsung

memodelkan probabilitas posterior $P(y \mid x)$ sebagai fungsi dari data input x. Pendekatan ini berfokus langsung pada penentuan batas keputusan antar kelas (Ng & Jordan, 2002).

2.3 *Knowledge Discovery in Databases* (KDD)

Knowledge Discovery in Databases atau KDD adalah proses yang bertujuan untuk mengekstraksi informasi yang dapat dipahami, menarik, dan bernilai dari data yang tidak terstruktur (Solanki & Sharma, 2021). Proses ini digunakan di berbagai bidang, seperti ilmu kehidupan, perdagangan, keuangan, dan kedokteran, untuk mengidentifikasi pola-pola yang tersembunyi dalam data yang besar dan kompleks (Solanki & Sharma, 2021). Proses ini mencakup berbagai teknik dan metode yang dapat digunakan untuk menggali wawasan dari data yang belum terorganisir.

KDD merupakan suatu bidang yang mengandalkan metode cerdas dalam *data mining* untuk menemukan pola-pola yang menjadi inti pengetahuan (Atloba, Balkir, & El-Mouadib, 2021). Pola-pola ini memungkinkan pengguna untuk memahami informasi yang terkandung dalam *dataset* besar, memberikan wawasan yang dapat diterapkan untuk pengambilan keputusan yang lebih baik dalam berbagai disiplin ilmu, dan alur prosesnya diilustrasikan pada Gambar 2.1.



Gambar 2.1 Proses KDD (Paucar & Andrade-Arenas, 2025)

Proses KDD, seperti yang diilustrasikan pada Gambar 2.1, merupakan alur kerja sistematis yang terdiri dari beberapa tahapan penting untuk mengubah data mentah menjadi pengetahuan yang berguna (Paucar & Andrade-Arenas, 2025). Tahapan-tahapan tersebut adalah:

a. Seleksi (*Selection*): Tahap awal di mana *subset* data yang relevan dengan tujuan analisis dipilih dari kumpulan data yang besar. Ini melibatkan pemilihan variabel dan sampel data yang akan menjadi fokus utama dalam proses penemuan pengetahuan.

b. Pra-pemrosesan (*Preprocessing*): Pada tahap ini, data yang telah dipilih dibersihkan untuk memastikan kualitasnya. Aktivitas yang dilakukan meliputi penanganan data yang hilang (*missing values*), penghapusan *noise* atau data yang tidak konsisten, dan perbaikan kesalahan data.

c. Transformasi (*Transformation*): Data yang sudah bersih diubah atau dikonsolidasikan ke dalam format yang sesuai untuk proses penambangan data. Ini bisa mencakup normalisasi data, agregasi, atau pembuatan atribut baru (dikenal sebagai *feature engineering*) untuk meningkatkan efektivitas analisis.

d. Penambangan Data (*Data Mining*): Ini adalah tahap inti di mana berbagai metode dan algoritma cerdas (seperti klasifikasi, *clustering*, atau analisis asosiasi) diterapkan pada data yang telah ditransformasi untuk mengidentifikasi pola-pola yang berpotensi menarik dan bermanfaat.

e. Evaluasi dan Interpretasi (*Evaluation*): Pola-pola yang ditemukan pada tahap sebelumnya dievaluasi untuk memverifikasi validitasnya dan diinterpretasikan untuk menjadi pengetahuan. Hanya pola yang dianggap signifikan, baru, dan dapat ditindaklanjuti yang akan disajikan kepada pengguna sebagai hasil akhir.

Dalam KDD, *machine learning* berperan penting untuk menganalisis data, mengenali korelasi, dan memprediksi hasil yang akan terjadi (Kodati & Selvaraj, 2021). Teknik-teknik *machine learning* digunakan untuk melatih model dalam mengidentifikasi pola-pola yang ada dalam data, yang kemudian dapat digunakan untuk membuat prediksi yang lebih akurat dalam berbagai aplikasi, seperti analisis kesehatan atau analisis perilaku konsumen.

Aplikasi KDD sangat luas, salah satunya adalah dalam bidang kesehatan, di mana KDD digunakan untuk mengembangkan sistem medis yang dapat mendeteksi dan memberikan saran pengobatan untuk penyakit dengan upaya minimal

(Nwankwo, Ngene, & Onuora, 2023). Selain itu, KDD berbasis metode *gradient boosting machine* juga diterapkan dalam prediksi energi listrik, memberikan referensi praktis bagi aplikasi KDD pada sektor energi lainnya (Xie *et al*., 2022).

KDD juga memiliki keterkaitan yang erat dengan analisis olahraga, khususnya sepak bola, di mana pendekatan KDD yang komprehensif memungkinkan persiapan data yang tepat untuk prediksi hasil pertandingan olahraga, termasuk hasil pertandingan sepak bola (Głowania, Kozak, & Juszczuk, 2023). Dengan menggunakan teknik KDD, analisis yang lebih mendalam dapat dilakukan terhadap data pertandingan untuk mengidentifikasi faktor-faktor yang mempengaruhi hasil akhir pertandingan.

2.4 *Machine Learning*

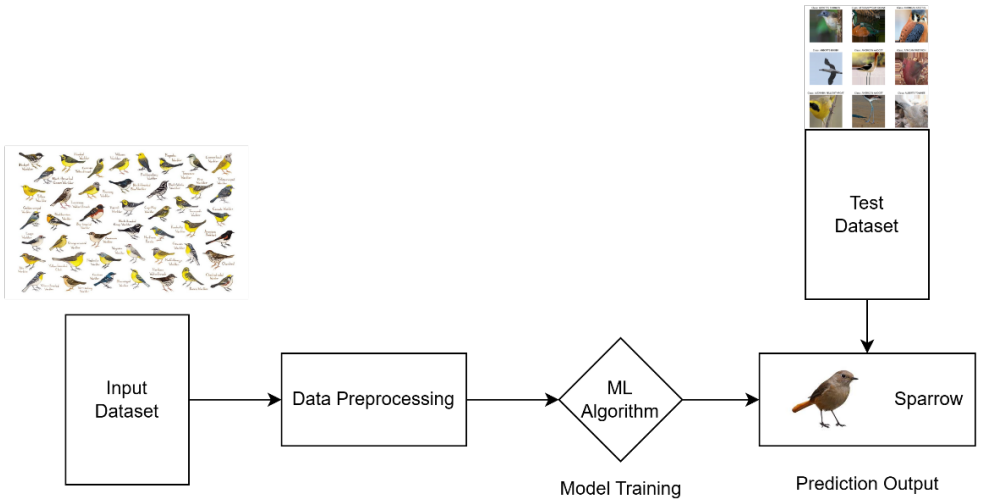
Machine Learning (ML) merupakan kemampuan suatu sistem untuk belajar dari data pelatihan yang spesifik terhadap masalah tertentu, dengan tujuan untuk mengotomatisasi proses pembangunan model analitik serta memecahkan tugas-tugas terkait. Dalam konteks ini, ML memungkinkan sistem komputer untuk mengidentifikasi pola dalam data tanpa campur tangan manual yang intensif, sehingga memungkinkan solusi otomatis terhadap berbagai masalah kompleks berbasis data (Janiesch *et al*., 2021).

Secara lebih mendalam, ML dapat dilihat sebagai bentuk kecerdasan buatan (AI) yang memanfaatkan data untuk melatih komputer dalam melakukan berbagai tugas tertentu, menggunakan algoritma untuk membangun serangkaian aturan secara otomatis. Proses ini memungkinkan sistem untuk secara mandiri mengenali

pola serta membuat keputusan berdasarkan data tanpa perlu diinstruksikan secara eksplisit, yang pada akhirnya meningkatkan ketepatan dan efisiensi sistem dalam memecahkan masalah kompleks (Schneider & Guo, 2018).

ML berbeda dari *data mining* dan statistik tradisional, baik dalam aspek filosofis maupun metodologis. Terdapat tiga pendekatan utama dalam ML yang membedakannya, yaitu statistika klasik, teori pembelajaran statistik Vapnik, serta teori pembelajaran komputasional (Kodama *et al*., 2023). Ketiga pendekatan ini menyediakan dasar yang berbeda untuk pengembangan algoritma, dimana ML fokus pada kemampuan untuk terus memperbaiki kinerja model berdasarkan data pelatihan, dibandingkan hanya melakukan analisis data historis sebagaimana dalam statistik tradisional.

Terdapat berbagai kategori dalam ML, meliputi *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Masing-masing pendekatan ini memiliki teknik-teknik unik, seperti *zero-shot learning, active learning, contrastive learning, self-supervised learning*, dan *semi-supervised learning* (Mahadevkar *et al*., 2022). Pada Gambar 2.2, ditunjukkan contoh implementasi *supervised learning*, di mana model dilatih menggunakan data berlabel untuk dapat mengklasifikasikan atau memprediksi berdasarkan pola yang telah dikenali. Teknik-teknik ini memperkaya cara sistem mempelajari data visual, baik dengan data yang memiliki label atau tanpa label.



Gambar 2.2 Contoh Implementasi *Supervised Learning* (Mahadevkar *et al*., 2022)

Algoritma dasar dalam ML sangat beragam, mencakup *decision tree, Random Forest, artificial neural network, support vector machine* (SVM), serta algoritma *boosting* dan *bagging*, yang membantu dalam meningkatkan kinerja model dengan menggabungkan prediksi dari beberapa model. Selain itu, algoritma *backpropagation* (BP) berperan penting dalam *neural networks* untuk mengoptimalkan bobot model berdasarkan kesalahan yang dihasilkan pada prediksi awal, sehingga meningkatkan kemampuan sistem dalam memprediksi hasil dengan lebih akurat (Jin, 2020).

Dalam ML, metrik evaluasi adalah instrumen logis dan matematis yang digunakan untuk mengukur seberapa dekat hasil prediksi model terhadap nilai aktualnya. Metrik evaluasi memungkinkan analisis kinerja model secara mendalam, sehingga aspek seperti akurasi, kesalahan, dan ketepatan dalam memprediksi dapat diukur secara kuantitatif. Hal ini penting untuk memahami performa model dan

menentukan langkah-langkah penyempurnaan lebih lanjut dalam pengembangan model (Plevris *et al*., 2022).

Beberapa metrik evaluasi yang paling sering digunakan dalam ML mencakup *Root Mean Squared Error* (RMSE), *Mean Absolute Error* (MAE), *Pearson Correlation Coefficient*, dan *Coefficient of Determination* (R^2) (Plevris *et al.*, 2022). Metrik-metrik ini membantu dalam mengukur seberapa akurat dan presisi prediksi model terhadap data yang diujikan, sehingga para praktisi dapat memilih metrik evaluasi yang paling relevan dengan konteks data dan tujuan analisis mereka.

2.5 *Gradient Boosting*

Gradient boosting merupakan teknik *machine learning* yang sangat efektif dan sering digunakan untuk menangani tugas dengan fitur heterogen serta data yang cenderung *noise*. Teknik ini bekerja dengan menggabungkan prediksi dari sejumlah model sederhana atau *weak learners* untuk menghasilkan prediksi yang kuat. Dalam klasifikasi, *Gradient boosting* menghasilkan distribusi pada label kelas, sementara dalam regresi, model ini memberikan prediksi nilai tunggal atau *point prediction* untuk mendekati hasil yang diinginkan. Kemampuan *gradient boosting* dalam menghadapi variasi pada fitur dan ketidakpastian dalam data menjadikannya alat yang sangat kuat dalam berbagai aplikasi *machine learning* (Ustimenko, Prokhorenkova, & Malinin, 2021).

Proses *gradient boosting* dimulai dengan mengombinasikan *weak learners*, yaitu model yang performanya sedikit lebih baik dari prediksi acak, untuk

membentuk *strong learner* secara iteratif. *gradient boosting* merupakan algoritma *boosting* yang dirancang khusus untuk masalah regresi.

Dalam algoritma ini, diberikan kumpulan data pelatihan $D = \{ x_i, y_i\}_1^N$, dengan tujuan utama mencari aproksimasi $\hat{F}(x)$ dari fungsi $F^*(x)$, yang memetakan instance x ke nilai output y, melalui minimisasi nilai ekspektasi dari fungsi loss tertentu $L(y, F(x))$. *Gradient boosting* membangun aproksimasi tambahan dari $F^*(x)$ sebagai jumlah berbobot dari sejumlah fungsi, sehingga memungkinkan model meningkatkan akurasi prediksi melalui iterasi yang berfokus pada mengurangi kesalahan residu (Bentéjac, Csörgő, & Martínez-Muñoz, 2020).

Pada persamaan (2.2) menunjukkan bagaimana setiap model baru (x) ditambahkan secara bertahap dengan bobot pada iterasi ke-m, yang bertujuan untuk mengurangi kesalahan prediksi dari model sebelumnya.

$$F_m(x) = F_{m-1}(x) + \rho_m h_m(x) \tag{2.2}$$

Dalam proses iteratif *gradient boosting*, ρ_m adalah bobot yang diberikan pada fungsi ke-m, yaitu $h_m(x)$**.** Fungsi-fungsi ini merupakan model-model dalam *ensemble*, seperti *decision tree*. Aproksimasi dari $F^*(x)$ dibangun secara bertahap, dimulai dengan mendapatkan aproksimasi konstan untuk $F^*(x)$ pada iterasi pertama. Hal ini dicapai dengan meminimalkan nilai *loss function* $L(y_i, \alpha)$ untuk setiap data pelatihan, dengan α adalah parameter konstanta yang mengoptimalkan fungsi tersebut. Pada iterasi pertama, aproksimasi ini diberikan oleh persamaan (2.3).

$$F_0(x) = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \alpha) \tag{2.3}$$

Persamaan (2.3) menunjukkan bahwa pada awalnya, model menghasilkan prediksi yang didasarkan pada nilai konstanta α yang meminimalkan kesalahan prediksi keseluruhan, $L(y_i, \alpha)$, di seluruh *dataset* . Pendekatan ini digunakan untuk membangun dasar dari model *gradient boosting* sebelum melanjutkan ke iterasi selanjutnya, di mana model-model tambahan (seperti *decision tree*) akan berfungsi untuk memperbaiki prediksi dari model sebelumnya (Bentéjac, Csörgő, & Martínez-Muñoz, 2020).

Pada iterasi selanjutnya, model yang dibangun diharapkan dapat meminimalkan fungsi pada persamaan (2.4).

$$\left(\rho_m, h_m(x)\right) = \underset{\rho,h}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_m - 1(x_i) + \rho h(x_i)\right) \tag{2.4}$$

Namun, alih-alih menyelesaikan masalah optimisasi ini secara langsung, setiap model h_m dapat dipandang sebagai langkah *greedy* dalam optimisasi menggunakan metode *gradient descent* untuk F^*. Untuk itu, setiap model h_m dilatih menggunakan *dataset* baru $D = \{x_i, r_{mi}\}_{i=1}^{N}$, di mana *residual* palsu r_{mi} dihitung berdasarkan turunan dari fungsi *loss* $L(y, F(x))$ terhadap $F(x)$, yang dievaluasi pada $F(x) = F_{m-1(x)}$, dengan rumus yang ditunjukkan pada persamaan (2.5).

$$\mathrm{r}_{mi} = \left[\frac{\partial L\left(Y_i, F(x)\right)}{\partial L(x)}\right]_{F(x)=F_{m-1(x)}} \tag{2.5}$$

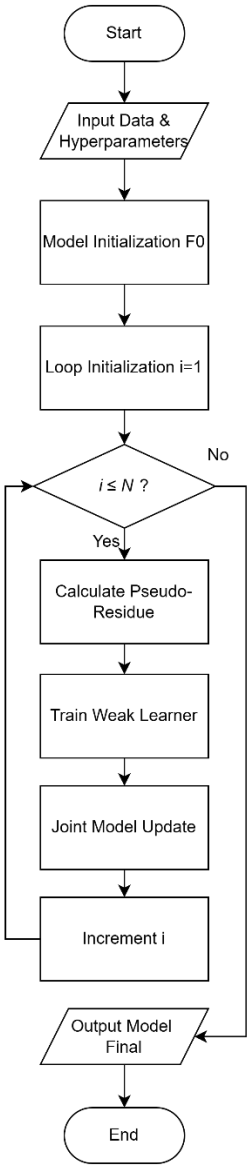
Nilai dari ρ_m kemudian dihitung dengan menyelesaikan masalah optimisasi pencarian garis. Proses ini, meskipun sangat efektif, dapat mengalami *overfitting* jika langkah-langkah iteratif tidak diatur dengan benar. Beberapa fungsi *loss* (misalnya *loss* kuadratik) dapat menyebabkan *residual* palsu menjadi nol pada

iterasi berikutnya jika model h_m sangat cocok dengan *residual* palsu, yang akan menyebabkan proses tersebut berhenti terlalu cepat. Untuk mengatasi masalah ini dan mengontrol proses penambahan dalam *gradient boosting*, beberapa parameter regularisasi dipertimbangkan. Salah satu cara alami untuk meredakan *overfitting* adalah dengan menerapkan *shrinkage*, yang berfungsi untuk mengurangi setiap langkah *gradient descent* (Bentéjac, Csörgő, & Martínez-Muñoz, 2020).

Gradient boosting membedakan dirinya dari metode *boosting* lainnya dengan menggabungkan konsep-konsep dari teori klasifikasi untuk estimasi dan seleksi efek prediktor dalam model regresi. Dalam hal ini, *gradient boosting* mempertimbangkan efek acak dan menawarkan pendekatan pemodelan yang lebih organik dan tidak bias. Berbeda dengan algoritma *boosting* lainnya yang mungkin mengasumsikan hubungan linier atau terlalu bergantung pada keputusan acak dalam tahap pemilihan model, *gradient boosting* memastikan bahwa estimasi prediktor disesuaikan secara cermat dengan data, meningkatkan akurasi model secara keseluruhan (Griesbach, Säfken, & Waldmann, 2020).

Selain itu, *gradient boosting* juga menawarkan kemampuan untuk menghasilkan perbaikan pada model non-konstan, dengan menggabungkan pengetahuan sebelumnya atau wawasan fisik terkait proses yang menghasilkan data (Wozniakowski *et al.*, 2021). Ini menjadi keunggulan lain dari *gradient boosting*, karena ia tidak hanya mengandalkan data murni, tetapi juga dapat memanfaatkan pengetahuan domain atau pemahaman fisik tentang bagaimana data tersebut terbentuk. Dengan pendekatan ini, *gradient boosting* dapat meningkatkan prediksi dalam konteks yang lebih luas, termasuk dalam situasi di mana model yang lebih

sederhana mungkin gagal. Alur kerja dari algoritma *gradient boosting* tersebut diilustrasikan secara *flowchart* pada Gambar 2.3.



Gambar 2.3 *Flowchart Gradient Boosting* (Zhang *et al*., 2023)

Gambar 2.3 menyajikan *flowchart* dari algoritma *Gradient Boosting*. Proses diawali dengan tahap persiapan, di mana data latih (*training data*) beserta konfigurasi *hyperparameter* (seperti jumlah pohon *N*) dimasukkan ke dalam sistem.

Selanjutnya, model diinisialisasi dengan membuat sebuah prediksi awal yang menjadi dasar bagi iterasi berikutnya.

Alur kerja kemudian memasuki sebuah perulangan utama yang dikontrol oleh kondisi $i \leq N$, yang memastikan proses akan berjalan sebanyak N kali. Pada setiap iterasi, langkah pertama adalah menghitung pseudo-residu, yaitu selisih atau kesalahan (*error*) dari prediksi model gabungan saat ini terhadap nilai target sebenarnya. Residu ini kemudian menjadi target pembelajaran bagi sebuah *Weak Learner* baru yang akan dilatih. Setelah model lemah tersebut terbentuk, ia ditambahkan untuk memperbarui (*update*) model gabungan, sehingga secara bertahap memperbaiki akurasi. Terakhir, penghitung iterasi (i) dinaikkan satu tingkat.

Proses iteratif ini terus berlanjut hingga target jumlah *Weak Learner* (sebanyak N) terpenuhi. Setelah keluar dari perulangan, alur kerja akan menghasilkan sebuah Model Final, yang merupakan ansambel (gabungan) kuat dari seluruh model lemah yang telah dilatih secara sekuensial. Dengan demikian, proses ini pun berakhir.

Sebagai algoritma *ensemble learning* yang semakin berkembang, telah terbukti unggul dalam meningkatkan prediksi dibandingkan dengan model lain, seperti *artificial neural network*, terutama dalam konteks pemodelan dinamis *bioprocess* (Mowbray *et al*., 2020). Dalam penerapan ini, *gradient boosting* menggabungkan beberapa model pembelajaran yang lemah untuk menghasilkan prediksi yang lebih akurat, menunjukkan keunggulannya dalam memodelkan dan

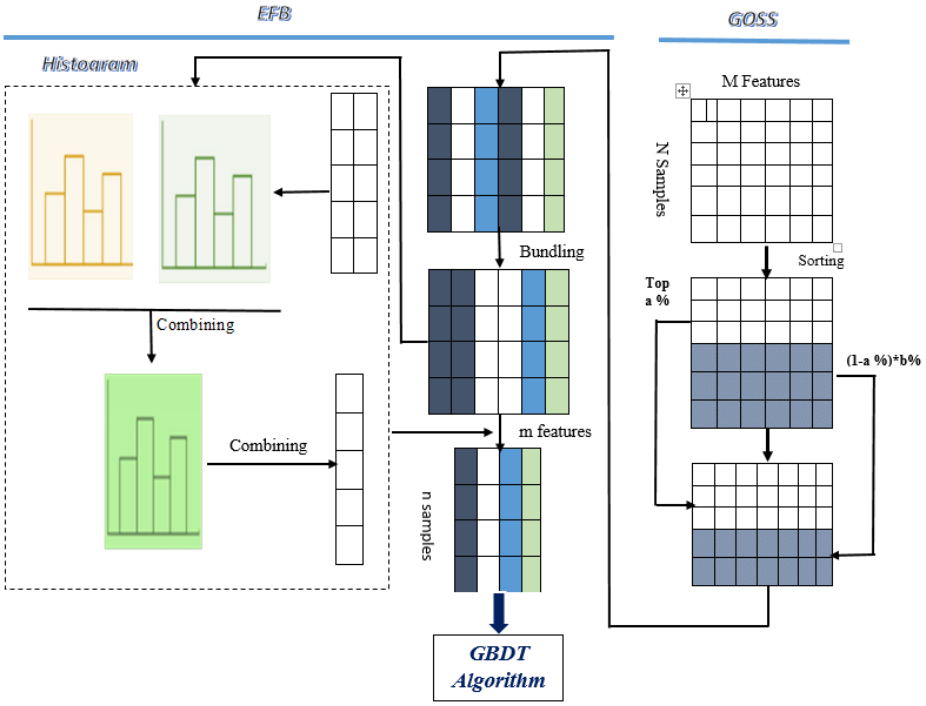
memprediksi proses yang dinamis dan kompleks, serta mampu mengatasi variasi yang ada dalam data yang digunakan.

Beberapa parameter dalam *gradient boosting*, seperti jumlah *node*, kedalaman maksimum, dan tingkat pembelajaran, dapat disesuaikan berdasarkan kinerja model pada *testing* set (Hu *et al*., 2023). Pengaturan parameter ini penting untuk memastikan model tidak hanya memberikan prediksi yang akurat, tetapi juga menghindari *overfitting*. Menyesuaikan parameter-parameter tersebut memungkinkan pemodel untuk mengoptimalkan performa model sesuai dengan karakteristik data yang digunakan, menjadikannya lebih fleksibel dan dapat diandalkan dalam berbagai jenis aplikasi.

2.6 *Light Gradient Boosting Machine*

Light Gradient Boosting Machine (LightGBM) adalah sebuah implementasi *gradien boosting* berkinerja tinggi yang didasarkan pada algoritma *Gradient Boosting Decision Tree* (GBDT), namun disempurnakan dengan teknik-teknik inovatif untuk meningkatkan kecepatan dan efisiensi. LightGBM memiliki beberapa keunggulan, termasuk kecepatan pelatihan yang lebih tinggi, penggunaan memori yang lebih rendah, akurasi yang lebih baik, serta dukungan untuk distribusi data dalam jumlah besar. LightGBM dikembangkan untuk mengatasi keterbatasan dalam GBDT tradisional, khususnya dalam hal kinerja dan efisiensi komputasi, sehingga memungkinkan pelatihan model pada *dataset* yang lebih besar dengan waktu yang lebih singkat (Huang & Chen, 2023).

LightGBM pertama kali dikembangkan pada tahun 2016 oleh tim peneliti di Microsoft sebagai peningkatan atas model GBDT yang populer, yaitu XGBoost. LightGBM diperkenalkan untuk meningkatkan efisiensi dan kecepatan yang lebih tinggi dari XGBoost, yang sering mengalami kendala kecepatan pada data berukuran besar. Dalam pengembangan LightGBM, tim peneliti memperkenalkan dua teknik baru: *Gradient-based One-Side Sampling* (GOSS) dan *Exclusive Feature Bundling* (EFB). Teknik ini dirancang untuk mengurangi jumlah sampel data dan fitur yang perlu diproses dalam pelatihan GBDT, sehingga mengatasi tantangan komputasi yang terkait dengan pemrosesan *dataset* besar (Kriuchkova, Toloknova, & Drin, 2024). Gambar 2.4 adalah arsitektur peningkatan algoritma GBDT dengan EFB dan GOSS.



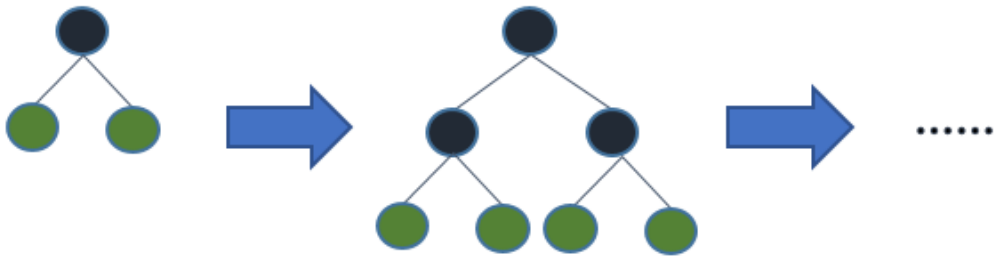
Gambar 2.4 Arsitektur GOSS dan EFB

Pada Gambar 2.4 disajikan arsitektur dan aliran data dalam kerangka kerja GBDT pada LightGBM yang ditingkatkan, mengintegrasikan teknik EFB dan GOSS. EFB bertujuan untuk mengurangi dimensi fitur dengan menggabungkan fitur-fitur yang jarang aktif bersamaan ke dalam bundel tunggal, sehingga menghasilkan matriks fitur yang lebih ringkas (m fitur dari M fitur awal). Proses ini melibatkan pembentukan histogram untuk setiap fitur dan kemudian menggabungkannya, yang secara efektif mengurangi kompleksitas komputasi tanpa mengorbankan informasi signifikan. Matriks fitur yang telah dibundel kemudian disatukan dengan sampel-sampel yang telah diseleksi oleh GOSS.

Sementara itu, GOSS mengatasi tantangan jumlah sampel yang besar dengan secara selektif mempertahankan instansi berdasarkan gradiennya. Sampel dengan gradien besar (*top* α%) dipertahankan secara utuh karena mereka berkontribusi paling signifikan terhadap *error* model, sedangkan sampel dengan gradien kecil diambil secara acak pada laju $(1-\alpha\%)\times b\%$. Pendekatan ini memungkinkan algoritma GBDT untuk fokus pada sampel yang paling informatif, mempercepat proses pelatihan sambil menjaga akurasi model. Kombinasi EFB dan GOSS secara sinergis mengurangi dimensi fitur dan jumlah sampel, secara substansial meningkatkan efisiensi komputasi dari algoritma GBDT tanpa mengorbankan kinerja, menjadikannya sangat efektif untuk *dataset* skala besar.

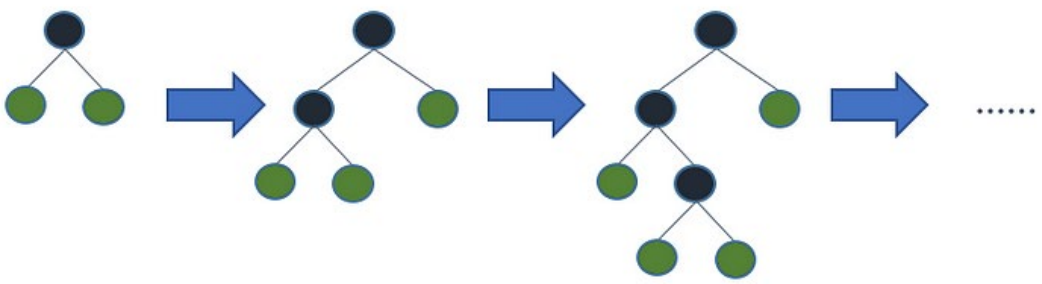
LightGBM menggunakan pendekatan yang berbeda dalam *decision tree learning* dibandingkan algoritma *decision tree* tradisional yang biasanya tumbuh berdasarkan tingkat atau kedalaman pohon (*depth-wise*). Dalam metode tradisional ini, semua *node* pada tingkat yang sama dianggap sama pentingnya, dan pohon

bertumbuh secara berjenjang untuk mencakup setiap *node* pada tingkat tertentu, seperti yang ditunjukkan pada Gambar 2.5 (LightGBM, 2024).



Gambar 2.5 Ilustrasi *Level-wise Tree Growth* (LightGBM, 2024)

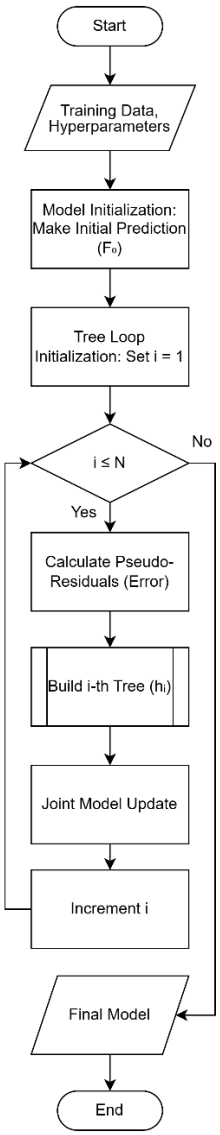
Namun, LightGBM mengadopsi strategi pertumbuhan pohon berbasis daun atau *leaf-wise*, yang hanya membagi daun yang diharapkan memberikan peningkatan terbesar terhadap akurasi model, seperti pada Gambar 2.6. Dengan fokus pada daun yang paling berpotensi untuk meningkatkan performa model, LightGBM membangun pohon secara lebih selektif dan efisien. Strategi *leaf-wise* ini bertujuan untuk memaksimalkan akurasi model dengan sumber daya yang lebih minimal, dibandingkan dengan metode tradisional yang sering kali menghasilkan cabang-cabang pohon yang tidak diperlukan dan memperlambat proses pelatihan (LightGBM, 2024).



Gambar 2.6 Ilustrasi *Leaf-wise Tree Growth* (LightGBM, 2024)

Pendekatan *leaf-wise* dalam LightGBM sering disebut juga sebagai pertumbuhan "*greedy growth*," yang memungkinkan algoritma untuk menemukan dan membagi daun dengan dampak terbesar terhadap akurasi model tanpa harus mempertimbangkan semua cabang secara merata pada setiap tingkat (LightGBM, 2024). Hal ini dapat diibaratkan seperti memangkas cabang-cabang yang tidak perlu, dengan fokus pada jalur yang paling bermanfaat. Sebagai akibat dari pendekatan yang selektif ini, struktur pohon dalam LightGBM menjadi asimetris, di mana beberapa cabang tumbuh lebih dalam daripada cabang lainnya, karena tujuan utamanya bukan simetri, melainkan peningkatan akurasi model.

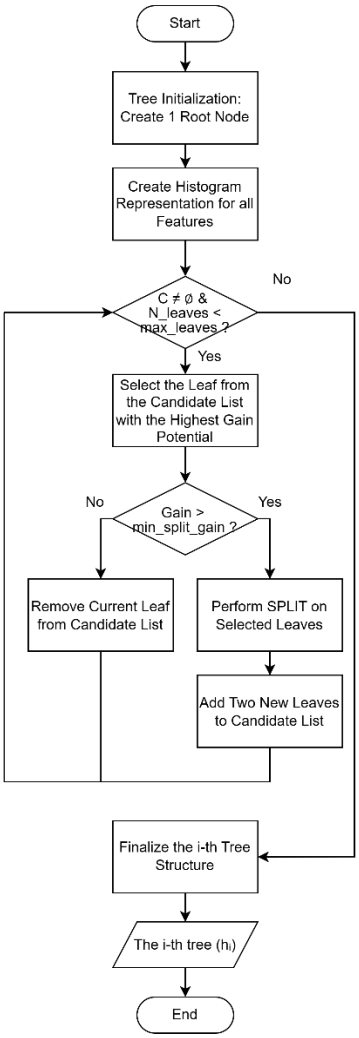
Manfaat dari strategi pertumbuhan berbasis daun ini adalah dalam hal kecepatan dan akurasi (LightGBM, 2024). Dari segi kecepatan, LightGBM menjadi sangat efisien karena metode *leaf-wise* hanya membagi daun yang memberikan dampak signifikan pada model, sehingga menghindari pengembangan sub-pohon yang tidak berkontribusi banyak terhadap peningkatan akurasi. Selain itu, pertumbuhan *leaf-wise* ini cenderung menghasilkan model dengan tingkat kesalahan (*loss*) yang lebih rendah dan akurasi yang lebih tinggi, karena algoritma dapat lebih terfokus pada bagian data yang paling informatif. Hal ini menjadikan LightGBM sebagai algoritma yang unggul dalam hal efisiensi dan ketepatan dalam menangani *dataset* yang besar dan kompleks. Gambar 2.7 menunjukkan *flowchart* dari LightGBM.



Gambar 2.7 *Flowchart* LightGBM (Ke *et al.*, 2017)

Gambar 2.7 menyajikan diagram alur kerja rinci dari algoritma LightGBM. Proses diawali dengan masukan (*input*) berupa data latih (*training data*) beserta konfigurasi *hyperparameter* yang telah ditentukan. Selanjutnya, algoritma melakukan tahap inisialisasi, di mana model dasar (F_0) dibentuk dengan membuat prediksi awal dan sebuah penghitung iterasi (i) diatur untuk memulai perulangan.

Setelah inisialisasi, proses memasuki *loop* iteratif utama yang berjalan selama jumlah pohon (*N*) yang ditargetkan belum tercapai. Pada setiap iterasi, langkah pertama adalah menghitung nilai *pseudo-residual*, yaitu selisih antara nilai target aktual dengan hasil prediksi model dari iterasi sebelumnya. Nilai *residual* ini kemudian digunakan sebagai target baru untuk melatih sebuah model lemah (*weak learner*), yang dalam hal ini adalah satu pohon keputusan. Setelah pohon keputusan yang baru berhasil dibangun, model gabungan diperbaharui dengan menambahkan kontribusi dari pohon baru tersebut yang telah diskalakan oleh *learning rate*. Proses ini diulang secara terus-menerus hingga kondisi berhenti terpenuhi dan menghasilkan sebuah model final yang kuat. Adapun rincian mengenai alur kerja sub-proses untuk membangun satu pohon keputusan secara detail disajikan pada Gambar 2.8.



Gambar 2.8 *Flowchart* Sub-Proses Membangun Satu Pohon Keputusan

Gambar 2.8 menyajikan diagram alur kerja untuk sub-proses pembangunan satu pohon keputusan, yang merupakan langkah detail dari alur utama LightGBM. Proses ini diawali dengan inisialisasi pohon yang terdiri dari satu *root node*. Untuk efisiensi komputasi, seluruh fitur pada data kemudian diubah ke dalam representasi berbasis histogram.

Selanjutnya, algoritma memasuki *loop* utama untuk menumbuhkan pohon secara *leaf-wise*. Pada setiap iterasi, dari semua daun yang menjadi kandidat,

algoritma akan memilih satu daun yang memiliki potensi *information gain* tertinggi. Setelah kandidat daun terbaik ditentukan, dilakukan pengecekan apakah nilai gain dari pemecahan (*split*) tersebut melebihi ambang batas *min_split_gain*. Jika gain mencukupi, daun tersebut akan dipecah menjadi dua daun baru, dan keduanya ditambahkan ke dalam daftar kandidat untuk iterasi selanjutnya. Sebaliknya, jika gain tidak mencukupi, daun tersebut akan dihapus dari daftar kandidat. Perulangan ini terus berjalan hingga tidak ada lagi kandidat yang bisa dipecah atau jumlah daun telah mencapai batas *max_leaves*, kemudian struktur pohon yang sudah final dikembalikan ke alur utama.

Untuk mengimplementasikan LightGBM, *library* utama yang diperlukan adalah LightGBM itu sendiri, yang dapat diinstal melalui pengelola paket sesuai bahasa pemrograman yang digunakan, seperti *Python* atau R (LightGBM, 2024). Selain *library* utama tersebut, ada beberapa dependensi lain yang juga dibutuhkan, seperti CMake untuk membangun lingkungan pengembangan dan *library* CUDA jika ingin memanfaatkan akselerasi GPU untuk mempercepat proses komputasi. Dengan adanya dukungan GPU, LightGBM dapat menangani data dalam jumlah besar dengan lebih efisien, mempercepat pelatihan model secara signifikan.

2.7 Sepak Bola

Sepak bola merupakan olahraga tim yang dimainkan secara global, menuntut pemain untuk menguasai berbagai kemampuan teknis, taktis, dan fisik dalam lingkungan yang dinamis dan kompetitif. Permainan ini pada dasarnya melibatkan dua tim yang saling berhadapan, di mana setiap tim berusaha untuk

mencetak gol dengan memasukkan bola ke gawang lawan menggunakan bagian tubuh mana pun selain tangan atau lengan (Sarmento *et al*., 2014). Sifat permainan yang kompleks dan interaktif ini menjadikan sepak bola sebagai subjek yang kaya untuk dianalisis dari berbagai perspektif ilmiah, mulai dari fisiologi hingga analisis data performa.

Sebuah pertandingan sepak bola standar dimainkan dalam dua babak yang masing-masing berdurasi 45 menit, dengan tujuan utama untuk mencetak skor lebih tinggi dari tim lawan. Setiap tim terdiri atas pemain yang menempati posisi-posisi strategis, seperti penjaga gawang yang bertugas melindungi gawang, pemain bertahan yang menghalau serangan lawan, pemain tengah yang mengatur alur permainan, serta penyerang yang berfokus untuk menciptakan peluang dan mencetak gol. Keberhasilan sebuah tim tidak hanya ditentukan oleh kemampuan individu, tetapi juga oleh kohesi dan koordinasi kolektif dalam menjalankan strategi permainan di bawah kerangka aturan yang diawasi oleh wasit (Carling, Williams, & Reilly, 2005).

2.8 Analisis Sepak Bola

Analisis sepak bola merupakan proses yang kompleks dan melibatkan berbagai aspek dari permainan yang saling terkait. Secara mendasar, analisis ini mencakup pengukuran komunikasi antar pemain, kemampuan adaptasi, tempo permainan, serta evaluasi taktik penyerangan dan pertahanan (Mclean *et al*., 2017). Analisis ini memperhitungkan dimensi sosial dan teknis dalam sepak bola, di mana

pemahaman akan sistem permainan sangat penting dalam mengoptimalkan kinerja tim secara keseluruhan.

Lebih jauh, analisis dalam sepak bola tidak hanya fokus pada aspek teknis dan taktis, tetapi juga memperhatikan variabel fisik yang relevan dalam konteks permainan sepak bola pria dewasa. Di samping itu, terdapat variabel situasional yang perlu diperhatikan seperti lokasi pertandingan, kualitas lawan, dan status pertandingan yang berpengaruh pada performa tim (Sarmento *et al*., 2014). Faktor-faktor ini menambah kompleksitas analisis dan menekankan pentingnya pendekatan menyeluruh yang mempertimbangkan kondisi dinamis permainan.

Dalam upaya meningkatkan performa pemain dan mengembangkan aktivitas pelatih, analisis sepak bola juga mengarah pada aspek-aspek mendetail seperti performa dalam situasi bola mati, perilaku sistem kolektif, komunikasi tim, dan profil aktivitas pemain. Fokus ini bertujuan untuk memberikan wawasan yang lebih dalam mengenai pola-pola permainan serta interaksi pemain di lapangan, yang pada akhirnya membantu pelatih dalam menyesuaikan strategi berdasarkan analisis berbasis data yang komprehensif (Sarmento *et al*., 2018).

Salah satu contoh penerapan analisis sepak bola yang semakin populer adalah penggunaan *data tracking* pemain. Data ini memungkinkan analisis yang lebih mendalam terhadap struktur permainan dengan memberikan wawasan mengenai performa tim, terutama dalam strategi bertahan. Implementasi analisis ini mengidentifikasi karakteristik permainan defensif yang berhasil, ditandai dengan tekanan tinggi, sinkronisasi gerakan antar pemain, keseimbangan pertahanan, serta organisasi pertahanan yang kompak dan terkoordinasi. Melalui *data tracking*,

pelatih dan analis dapat memahami pola pertahanan yang efektif dan mengoptimalkan strategi tim berdasarkan perilaku lapangan yang terukur (Forcher *et al*., 2022).

2.9 *Expected Goals* (xG)

Expected Goals atau xG adalah salah satu metrik yang semakin digunakan dalam analisis sepak bola modern untuk menilai peluang terjadinya gol berdasarkan kualitas dan lokasi tembakan yang dilakukan (Mead, O'Hare, & McMenemy, 2023). Metrik ini memberikan prediksi probabilitas yang lebih akurat dibandingkan statistik konvensional dalam memperkirakan keberhasilan suatu tim di masa mendatang. Dalam hal ini, xG membantu memberikan pandangan yang lebih obyektif dan berbasis data mengenai kemungkinan pencapaian gol yang dihasilkan dari berbagai jenis tembakan selama pertandingan.

Metrik xG dirancang untuk memberikan skor probabilistik pada setiap tembakan, dengan nilai yang berkisar antara 0 dan 1, di mana 0 menunjukkan tidak ada peluang mencetak gol, dan 1 menunjukkan kepastian terjadinya gol. Penilaian ini memungkinkan xG untuk menangani unsur ketidakpastian dalam sepak bola dengan lebih baik dibandingkan metrik berbasis gol konvensional. Karena tembakan jauh lebih sering terjadi daripada gol, pendekatan ini memungkinkan analisis yang lebih stabil dan realistis dalam memahami efektivitas tim dan pemain di lapangan (Mead, O'Hare, & McMenemy, 2023).

Selain berguna untuk analisis taktis yang mendukung peningkatan performa di lapangan, xG juga memainkan peran penting dalam keputusan finansial klub.

Metrik ini membantu dalam keputusan seperti perekrutan pemain dan negosiasi kontrak dengan memberikan wawasan yang lebih akurat mengenai kontribusi pemain. Dengan demikian, xG tidak hanya membantu klub dalam memaksimalkan performa di lapangan tetapi juga dalam mengelola sumber daya finansial secara lebih efisien (Mead, O'Hare, & McMenemy, 2023).

Penerapan xG memberikan keuntungan strategis bagi klub sepak bola dengan memperluas pemahaman terkait kualitas peluang yang dihasilkan. Hal ini memungkinkan klub untuk mengevaluasi kinerja pemain secara lebih mendalam dan membantu dalam pengembangan strategi permainan yang berbasis pada kualitas dan efektivitas peluang (Mead, O'Hare, & McMenemy, 2023). Oleh karena itu, xG melampaui perannya sebagai metrik statistik.

Di dalam konsepnya, perhitungan xG dapat dianggap sebagai permasalahan klasifikasi, karena melibatkan penentuan probabilitas tembakan menghasilkan gol berdasarkan berbagai faktor. Untuk menghitung probabilitas ini, metode *machine learning* dan statistika sering diterapkan, termasuk *logistic regression*, *gradient boosting*, *neural networks*, *support vector machines*, serta algoritma klasifikasi *tree-based*. Beragam pendekatan ini memungkinkan xG untuk memanfaatkan data historis dan pola dalam data tembakan untuk memodelkan kemungkinan gol secara lebih akurat, yang berguna dalam memberikan penilaian yang lebih detail tentang kualitas peluang tembakan (Herbinet, 2018).

Model xG dapat memiliki tingkat akurasi yang berbeda tergantung pada jumlah faktor yang dimasukkan ke dalam perhitungannya. Sebagai contoh, model

xG biasanya memperhitungkan jarak tembakan ke gawang, sudut tembakan, bagian tubuh yang digunakan, dan jenis umpan yang mendahului tembakan.

Berdasarkan faktor-faktor tersebut, sebuah tembakan mungkin diberi nilai 0,30 xG. namun model yang lebih presisi, seperti Statsbomb xG, mempertimbangkan informasi tambahan seperti posisi kiper, status kiper, posisi pemain bertahan dan penyerang, serta tinggi dampak tembakan. Dalam kondisi kiper yang tidak berada di posisinya, model ini mungkin memberikan nilai yang lebih tinggi, misalnya 0,65 xG, untuk menggambarkan kualitas peluang yang lebih tinggi (Statsbomb, 2024).

Visualisasi dari model ini pada Gambar 2.9, yang merupakan visualisasi xG pada pertandingan langsung, memperlihatkan bagaimana setiap faktor dihitung untuk menghasilkan prediksi xG yang mendalam dan akurat.

Gambar 2.9 Visualisasi xG pada Pertandingan Langsung (Statsbomb, 2024)

2.10 *Brier Score*

Brier Score merupakan metrik evaluasi yang mengukur ketepatan dalam pemodelan prediksi, dengan cara membagi prediksi ke dalam beberapa kelompok atau "*bins*" berdasarkan kesamaan nilai prediksi (Foster & Hart, 2022). Metrik ini memadukan skor kalibrasi dan skor penyempurnaan (*refinement*) untuk mengukur keahlian dalam pemodelan prediktif. Dengan menggabungkan aspek kalibrasi, yang menunjukkan seberapa baik prediksi sejalan dengan hasil aktual, dan aspek penyempurnaan, yang melihat kemampuan model dalam memisahkan atau membedakan hasil yang berbeda, *Brier Score* memberikan gambaran komprehensif mengenai performa model dalam memberikan prediksi probabilistik.

Penggunaan *Brier Score* dalam evaluasi model probabilitas penting karena metrik ini dapat mengukur kemampuan diskriminasi dan performa prediktif secara keseluruhan. Dengan kata lain, *Brier Score* tidak hanya melihat akurasi dari prediksi probabilitas tetapi juga sejauh mana model dapat membedakan antara kejadian yang mungkin terjadi dengan yang tidak (Dimitriadis *et al*., 2023). Hal ini membuat *Brier Score* menjadi pilihan yang baik untuk mengevaluasi performa model probabilistik, khususnya ketika diperlukan pemahaman yang lebih dalam mengenai kualitas prediksi yang bersifat probabilistik. Fungsi Brier Score ditunjukan pada persamaan (2.6).

$$Brier\ Score\ = (f_t - o_t)^2 \tag{2.6}$$

Brier Score digunakan untuk menghitung selisih kuadrat antara nilai prediksi dan nilai aktual, sebagaimana terlihat pada persamaan (2.6). Dalam konteks ini, f_t merepresentasikan nilai probabilitas yang diprediksi untuk suatu

peristiwa, sedangkan o_t adalah nilai aktual dari peristiwa tersebut (biasanya 1 jika terjadi dan 0 jika tidak terjadi). *Brier Score* memiliki rentang nilai antara 0 hingga 1, di mana nilai yang lebih rendah menunjukkan prediksi yang lebih akurat karena mendekati hasil aktual (Steyerberg *et al*., 2010).

Brier Score diperkenalkan oleh Glenn W. Brier pada tahun 1950 sebagai alat untuk menilai akurasi prediksi probabilitas (Foster & Hart, 2022). Skor ini menghitung selisih antara nilai prediksi dan realisasi aktual, di mana hasil perhitungan *Brier Score* memperlihatkan seberapa dekat prediksi tersebut dengan hasil aktual menggunakan formula *mean squared error* standar.

Sejak pertama kali diperkenalkan yaitu pada evaluasi ramalan cuaca, *Brier Score* telah berkembang menjadi metode yang diakui untuk mengukur akurasi model probabilitas dalam berbagai bidang, termasuk bisnis dan aplikasi lainnya (Petropoulos *et al*., 2022). Penerapan awalnya pada meteorologi menunjukkan bagaimana metode ini dapat memberikan wawasan yang lebih mendalam terhadap ketepatan perkiraan, yang kemudian menjadikan *Brier Score* sebagai standar dalam penilaian akurasi probabilitas di berbagai disiplin ilmu.

2.11 *Receiver Operating Characteristic Area Under Curve* (ROC AUC)

Receiver Operating Characteristic (ROC) adalah alat statistik yang digunakan untuk menilai kinerja model klasifikasi dengan menggambarkan hubungan antara dua parameter, yaitu *True Positive Rate* (TPR) dan *False Positive Rate* (FPR). Analisis ROC dapat dilakukan dengan memanfaatkan distribusi prior dan algoritma *elicitation* untuk memilih prior yang tepat, yang selanjutnya

digunakan untuk menarik inferensi mengenai AUC (*Area Under the Curve*) dan karakteristik error model (Labadi *et al*., 2022).

ROC juga digunakan untuk mengevaluasi kinerja perangkat pengujian dan algoritma klasifikasi dalam menilai kepatuhan terhadap kriteria tertentu (Pendrill *et al*., 2023). Dengan demikian, ROC menjadi alat yang penting untuk perbandingan dan evaluasi relatif dari berbagai sistem klasifikasi dalam konteks yang berbeda.

Kurva ROC menggambarkan kinerja model klasifikasi pada berbagai ambang batas klasifikasi dengan memplot dua parameter utama, yaitu TPR dan FPR. Salah satu kelemahan dari kurva ROC adalah kesulitan dalam menginterpretasi kinerja model jika terdapat banyak titik keputusan, karena setiap titik mewakili *trade-off* antara TPR dan FPR, yang dapat membuat sulit untuk menentukan titik terbaik yang mencerminkan kinerja keseluruhan model (Chen *et al*., 2023). ROC AUC mengukur luas dua dimensi di bawah kurva ROC, dimulai dari titik (0,0) hingga (1,1). Semakin tinggi nilai ROC AUC, semakin baik model dalam membedakan antara kelas positif dan negatif.

Secara matematis, AUC dari kurva ROC dihitung dengan mengintegralkan fungsi ROC. Mengingat kurva ROC memplot *True Positive Rate* (TPR) sebagai fungsi dari *False Positive Rate* (FPR), AUC dapat didefinisikan dengan persamaan integral (Fawcett, 2006) seperti pada persamaan (2.7).

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR}), d(\mathrm{FPR}) \tag{2.7}$$

Dalam praktiknya, karena kurva ROC terdiri atas sejumlah titik diskrit, AUC sering dihitung menggunakan aturan trapesium. Parameter TPR dan FPR

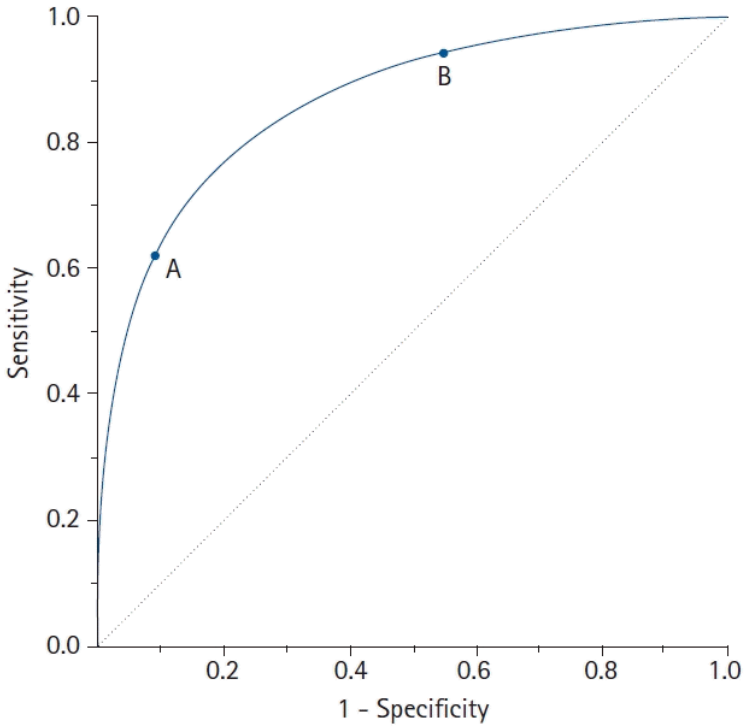
sendiri dihitung berdasarkan nilai dari *confusion matrix* dengan persamaan (2.8) dan (2.9).

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.8}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.9}$$

Di mana TP adalah *True Positive*, FN adalah *False Negative*, FP adalah *False Positive*, dan TN adalah *True Negative*. Nilai AUC juga dapat diinterpretasikan sebagai probabilitas bahwa model akan memberikan skor yang lebih tinggi untuk sampel positif yang dipilih secara acak dibandingkan dengan sampel negatif yang dipilih secara acak (Fawcett, 2006).

Pada Gambar 2.10, menampilkan kurva ROC AUC, di mana sumbu *x* menunjukkan nilai 1 - spesifisitas (*false positive rate*) dan sumbu *y* menunjukkan sensitivitas pada semua nilai *cut-off* yang diukur dari hasil pengujian (Nahm, 2022). Ketika nilai *cut-off* yang lebih ketat diterapkan, titik pada kurva akan bergerak ke bawah dan ke kiri (Titik A). Sebaliknya, saat *cut-off* lebih longgar diterapkan, titik pada kurva bergerak ke atas dan ke kanan (Titik B). Garis diagonal 45° pada grafik ini berfungsi sebagai garis referensi, yang merepresentasikan kurva ROC dari klasifikasi acak.



Gambar 2.10 Contoh ROC AUC (Nahm, 2022)

ROC AUC memiliki peran penting dalam evaluasi model karena mampu mengukur kinerja model dalam berbagai kelompok risiko yang diprediksi (Carrington *et al*., 2021). Ini memberikan informasi yang lebih mendalam yang dapat digunakan dalam pengambilan keputusan, memungkinkan pemahaman yang lebih komprehensif tentang bagaimana model berperforma di berbagai titik potong dan kelompok risiko.

Lebih lanjut, ROC AUC juga memungkinkan perbandingan yang wajar antar model dan membantu mengidentifikasi batas keputusan yang optimal serta potensi peningkatan ROC AUC. Ini membuat ROC AUC sangat bermanfaat dalam seleksi model yang lebih baik dan pemahaman tentang ruang yang dapat dioptimalkan untuk meningkatkan kinerja klasifikasi (Tafvizi *et al*., 2022).

Secara fundamental, fungsi utama dari ROC AUC adalah untuk menyediakan satu nilai tunggal yang merangkum kinerja keseluruhan model klasifikasi di semua kemungkinan ambang batas (*threshold*). Alih-alih harus menganalisis setiap titik pada kurva ROC, yang dapat menyulitkan interpretasi jika terdapat banyak titik keputusan (Chen *et al*., 2023), ROC AUC menyederhanakan evaluasi dengan mengukur total area dua dimensi di bawah kurva tersebut. Nilai AUC ini dapat diartikan sebagai probabilitas bahwa model akan memberikan skor prediksi yang lebih tinggi untuk sampel kelas positif yang dipilih secara acak daripada sampel kelas negatif yang dipilih secara acak. Oleh karena itu, AUC berfungsi sebagai metrik yang agregat dan tidak bergantung pada ambang batas tertentu, di mana nilai yang mendekati 1.0 menunjukkan kemampuan diskriminasi yang sangat baik antara kelas positif dan negatif, sementara nilai mendekati 0.5 mengindikasikan kinerja yang tidak lebih baik dari tebakan acak (Fawcett, 2006).

2.12 *Confusion Matrix*

Evaluasi kinerja sebuah model klasifikasi dimulai dari pemahaman terhadap *confusion matrix*. *Confusion matrix* adalah alat visualisasi fundamental dalam bentuk tabel kontingensi yang merangkum dan membandingkan hasil prediksi model dengan kelas aktual dari data uji. Struktur ini memberikan gambaran yang jelas tidak hanya tentang seberapa sering model benar, tetapi juga tentang jenis kesalahan yang dibuatnya (Tharwat, 2021).

Untuk masalah klasifikasi biner, *confusion matrix* biasanya disajikan dalam format 2×2. Matriks ini memiliki empat komponen utama yang mendeskripsikan

hasil prediksi. Pertama, *True Positive* (TP), yang mewakili kasus di mana model dengan tepat memprediksi kelas positif; sebagai contoh, sebuah *email spam* berhasil diidentifikasi sebagai spam. Kedua, *True Negative* (TN), yaitu kasus di mana model secara benar memprediksi kelas negatif, seperti email penting yang tidak diklasifikasikan sebagai spam. Dua komponen lainnya menggambarkan kesalahan model. *False Positive* (FP), atau *Error* Tipe I, terjadi ketika model salah memprediksi kelas positif untuk *instance* yang sebenarnya negatif, misalnya email penting yang keliru ditandai sebagai spam. Terakhir, *False Negative* (FN), atau *Error* Tipe II, terjadi saat model salah memprediksi kelas negatif untuk *instance* yang sebenarnya positif, seperti *email spam* yang lolos dari filter dan masuk ke kotak masuk utama. Visualisasi dari keempat komponen tersebut disajikan dalam struktur Tabel 2.1.

Tabel 2.1 *Confusion Matrix*

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

2.13 Presisi (*Precision*)

Setelah memahami komponen *confusion matrix*, metrik Presisi dapat didefinisikan secara spesifik. Presisi menjawab pertanyaan: "Dari semua *instance* yang diprediksi oleh model sebagai kelas positif, berapa persen yang benar-benar positif?" Metrik ini mengukur tingkat keandalan atau ketepatan dari prediksi positif yang dibuat oleh model (Powers, 2011).

Secara matematis, presisi dihitung dengan membagi jumlah *True Positive* dengan total jumlah prediksi positif yang dihasilkan oleh model (*True Positive* ditambah *False Positive*), seperti yang ditunjukkan pada persamaan (2.10) (Powers, 2011).

$$\text{Presisi} = \frac{TP}{TP + FP} \tag{2.10}$$

Presisi menjadi metrik yang sangat krusial dalam skenario di mana biaya dari *False Positive* (FP) sangat tinggi. Sebagai contoh, dalam sistem penyaringan email, kesalahan mengklasifikasikan email penting dari atasan sebagai spam (sebuah FP) dapat menyebabkan pengguna kehilangan informasi yang sangat krusial. Dalam kasus deteksi penyakit, mendiagnosis orang sehat sebagai penderita penyakit (FP) dapat menyebabkan kecemasan, biaya pengobatan yang tidak perlu, dan tes lebih lanjut yang invasif. Oleh karena itu, model dengan presisi tinggi lebih disukai dalam situasi-situasi tersebut karena ia cenderung tidak salah dalam melabeli sesuatu sebagai positif (Tharwat, 2021).

2.14 *Recall*

Recall, yang juga dikenal sebagai sensitivitas atau *true positive rate* (TPR), adalah metrik yang mengukur kemampuan sebuah model untuk menemukan kembali semua sampel positif yang relevan dalam sebuah *dataset*. Dengan kata lain, *recall* merepresentasikan proporsi dari kasus positif aktual yang berhasil diidentifikasi dengan benar oleh model. Esensi dari metrik ini adalah untuk mengevaluasi tingkat kelengkapan (*completeness*) dari prediksi positif yang dihasilkan (Powers, 2011).

Recall sangat krusial dalam domain di mana biaya dari *False Negative* sangat tinggi. Misalnya, dalam diagnosis medis, gagal mendeteksi adanya penyakit (*False Negative*) pada pasien bisa berakibat fatal. Oleh karena itu, *recall* yang tinggi lebih diutamakan dalam konteks tersebut. *Recall* dapat dihitung dengan persamaan (2.11) (Powers, 2011).

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

2.15 *F1-Score*

F1-Score adalah metrik yang menggabungkan presisi dan *recall* ke dalam satu skor tunggal dengan menghitung rata-rata harmonik dari keduanya. Rata-rata harmonik cenderung lebih dekat ke nilai yang lebih kecil, sehingga *F1-Score* memberikan bobot yang seimbang pada kedua metrik tersebut (Sasaki, 2007).

Metrik ini sangat berguna ketika terjadi ketidakseimbangan kelas (*imbalanced class*), di mana jumlah sampel pada satu kelas jauh lebih dominan daripada kelas lainnya. Dalam kasus seperti itu, akurasi saja bisa menyesatkan, sedangkan *F1-Score* memberikan gambaran yang lebih representatif mengenai performa model. Nilai *F1-Score* yang tinggi menunjukkan bahwa model memiliki performa yang baik dalam hal presisi maupun *recall*, menjadikannya metrik evaluasi yang komprehensif. Persamaan untuk *F1-Score* ditunjukkan pada persamaan (2.12) (Sasaki, 2007).

$$F1 - Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall} \tag{2.12}$$

2.16 *Log-Loss* (*Cross-Entropy Loss*)

Log-Loss, yang secara formal dikenal sebagai *cross-entropy loss*, adalah metrik evaluasi fundamental untuk model klasifikasi yang menghasilkan *output* probabilitas (Bishop, 2006). Berakar dari teori informasi, metrik ini mengukur "jarak" antara distribusi probabilitas yang diprediksi model dengan distribusi aktualnya. Berbeda dengan akurasi yang hanya menilai kebenaran prediksi, *Log-Loss* memberikan evaluasi yang lebih mendalam dengan mengukur seberapa baik kalibrasi dan tingkat keyakinan (*confidence*) dari setiap prediksi (Hastie *et al*. 2009). Hal ini menjadikannya sangat berharga dalam aplikasi berbasis risiko, di mana mengetahui probabilitas suatu hasil jauh lebih penting daripada sekadar klasifikasi benar atau salah.

Mekanisme utama *Log-Loss* adalah memberikan penalti yang besar pada prediksi yang sangat yakin namun ternyata salah. Sebagai contoh, prediksi dengan probabilitas 0.95 untuk kelas yang salah akan dihukum jauh lebih berat daripada prediksi 0.55 untuk kasus yang sama. Dengan demikian, metrik ini mendorong model untuk tidak hanya akurat, tetapi juga menghasilkan probabilitas yang terkalibrasi dengan baik. Nilai *Log-Loss* yang ideal adalah 0, yang menandakan model sempurna, dan nilai yang semakin tinggi menunjukkan performa model yang semakin buruk.

Untuk masalah klasifikasi biner (di mana hasil akhirnya adalah 1 atau 0), *Log-Loss* dihitung menggunakan persamaan (2.13) (Bishop, 2006).

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \qquad (2.13)$$

Dalam persamaan (2.13), *N* merepresentasikan jumlah total sampel dalam *dataset*. Perhitungan kerugian dilakukan secara iteratif untuk setiap sampel, dari sampel pertama (*i*=1) hingga terakhir (*i*=*N*), yang dilambangkan oleh operator sigma (∑). Hasil penjumlahan total kerugian kemudian dinormalisasi dengan cara dibagi oleh N, sehingga menghasilkan nilai kerugian rata-rata. Normalisasi ini memastikan bahwa performa model dapat dibandingkan secara adil tanpa terpengaruh oleh ukuran *dataset*.

Setiap komponen dalam kurung siku menghitung kerugian untuk satu sampel individual. Di sini, y_i adalah label kelas aktual dari sampel ke-*i,* yang memiliki nilai 1 untuk kelas positif dan 0 untuk kelas negatif. Sementara itu, p_i adalah probabilitas yang dihasilkan oleh model, yang menunjukkan prediksi peluang sampel ke-*i* untuk masuk ke dalam kelas positif (nilai antara 0 dan 1). Fungsi logaritma natural, *log*(), menjadi inti dari mekanisme penalti dalam rumus ini.

Logika perhitungan kerugian dapat dipahami dengan menganalisis dua kondisi berdasarkan nilai y_i. Pertama, ketika label aktualnya adalah 1 (y_i=1), suku kedua dalam penjumlahan, yaitu $(1-y_i)log(1-p_i)$, akan menjadi nol. Dengan demikian, kerugian untuk sampel ini hanya dihitung dari suku pertama, $y_i\ log(p_i)$ atau $log(p_i)$. Jika model memprediksi probabilitas (p_i) yang mendekati 1 (sangat yakin benar), nilai $log(p_i)$ akan mendekati 0, yang berarti kerugian sangat kecil. Sebaliknya, jika model salah prediksi dengan p_i mendekati 0, nilai $log(p_i)$ akan

menuju negatif tak terhingga, yang setelah dikalikan dengan tanda negatif di awal rumus, akan menghasilkan nilai kerugian yang sangat besar.

Kedua, ketika label aktualnya adalah 0 (y_i=0), suku pertama, $y_i\ log(p_i)$, akan menjadi nol. Perhitungan kerugian kini bergantung pada suku kedua, $(1-y_i)log(1-p_i)$ atau $log(1-p_i)$. Suku $1-p_i$ merepresentasikan probabilitas sampel untuk masuk ke kelas negatif. Jika model memprediksi p_i mendekati 0 (sehingga $1-p_i$ mendekati 1), maka model sangat yakin bahwa sampel ini adalah kelas negatif. Dalam kasus ini, nilai $log(1-p_i)$ akan mendekati 0, dan kerugiannya pun kecil. Namun, jika model salah besar dengan memprediksi p_i mendekati 1, maka $1-p_i$ akan mendekati 0, dan kerugian (*loss*) yang dihasilkan akan sangat besar.

2.17 *Feature Engineering*

Feature engineering adalah proses rekayasa data secara cerdas untuk meningkatkan kinerja model *machine learning* dengan cara meningkatkan akurasi dan kemampuan interpretasinya (Verdonck *et al*., 2024). Proses ini dilakukan melalui penyesuaian fitur yang telah ada atau dengan mengekstraksi fitur baru yang lebih bermakna dari berbagai sumber data. Teknik ini bertujuan untuk menciptakan representasi data yang lebih informatif, sehingga model dapat memahami hubungan yang lebih kompleks di dalam data. *Feature engineering* tidak hanya membantu dalam memperbaiki akurasi prediksi, tetapi juga memungkinkan pengguna untuk memahami bagaimana setiap fitur memengaruhi hasil akhir, menjadikannya langkah penting dalam pengembangan model *machine learning* yang lebih efektif dan dapat diandalkan.

Feature engineering memungkinkan pengguna untuk membuat fitur-fitur baru secara mandiri yang lebih relevan dengan permasalahan yang sedang dianalisis (Das *et al*., 2022). Fitur-fitur ini kemudian dapat digunakan untuk meningkatkan proses penerapan algoritma *machine learning* dalam membuat prediksi yang lebih akurat. Dengan menciptakan fitur yang disesuaikan dengan kebutuhan analisis, pengguna dapat membantu model *machine learning* mengenali pola-pola penting yang sebelumnya tidak terdeteksi, sehingga hasil prediksi menjadi lebih optimal dan bermakna.

Teknik-teknik esensial dalam *feature engineering* berperan penting dalam meningkatkan kinerja model prediksi di berbagai bidang. Teknik-teknik ini mencakup (Katya, 2023):

a. *Feature Selection*

Feature Selection merupakan proses memilih fitur-fitur yang paling relevan dan informatif dari kumpulan data yang tersedia. Dengan menyaring fitur yang tidak signifikan atau *redundant*, proses ini membantu mengurangi *noise* dan kompleksitas data. Hal tersebut sangat penting untuk mencegah *overfitting* dan memastikan bahwa model hanya menggunakan informasi yang benar-benar berkontribusi terhadap variabel target. Dengan demikian, model prediksi dapat bekerja lebih efisien dan menghasilkan akurasi yang lebih tinggi.

b. *Dimensionality Reduction*

Dimensionality reduction adalah teknik yang bertujuan untuk mengurangi jumlah fitur dalam *dataset* tanpa mengorbankan informasi penting yang

terkandung di dalamnya. Teknik ini menyederhanakan struktur data, sehingga memudahkan proses analisis dan meningkatkan performa model. Metode seperti *Principal Component Analysis* (PCA) mengubah fitur asli menjadi komponen baru yang lebih ringkas, tetapi tetap merepresentasikan variasi data secara keseluruhan. Pendekatan ini tidak hanya mempercepat proses pelatihan model, tetapi juga meningkatkan kemampuan interpretasi hasil.

c. *Interaction Term Creation*

Interaction term creation adalah proses menciptakan fitur baru dengan mengombinasikan dua atau lebih fitur yang ada. Teknik ini dirancang untuk menangkap interaksi atau hubungan sinergis antar fitur yang mungkin tidak terlihat saat dianalisis secara individual. Dengan menggabungkan fitur-fitur tersebut, model dapat lebih sensitif terhadap pola-pola kompleks yang berpengaruh terhadap hasil akhir, sehingga meningkatkan keakuratan prediksi.

Secara keseluruhan, penerapan teknik-teknik ini dalam *feature engineering* membantu mengoptimalkan data *input* sehingga algoritma *machine learning* dapat menghasilkan prediksi yang lebih akurat dan interpretasi yang lebih mendalam. Teknik-teknik tersebut berperan penting dalam menyederhanakan, menyoroti, dan memperkaya informasi yang terkandung dalam data, yang pada akhirnya berkontribusi terhadap peningkatan kinerja model di berbagai aplikasi.

2.18 *Tools* Penelitian

Pelaksanaan penelitian ini didukung oleh beberapa perangkat lunak esensial yang digunakan untuk pemrosesan dan analisis data. Setiap *tool* memiliki peran spesifik yang berkontribusi pada pencapaian tujuan penelitian.

2.18.1 Python

Python adalah bahasa pemrograman tingkat tinggi bersifat *object-oriented*, dikembangkan oleh Guido van Rossum, bahasa ini dirancang untuk menjadi mudah dipahami dan digunakan sehingga cocok baik untuk pemula yang sedang mempelajari dasar-dasar pemrograman maupun untuk para profesional yang mengerjakan proyek pemrograman di dunia nyata (Srinath, 2017). Python menawarkan *syntax* yang sederhana dan intuitif, sehingga memungkinkan pengguna menulis kode dengan lebih cepat dan efisien. Selain itu, Python memiliki dukungan pustaka yang sangat luas serta komunitas yang aktif, menjadikannya pilihan populer untuk berbagai kebutuhan, mulai dari pengembangan web, analisis data, *machine learning*, hingga komputasi ilmiah dan otomatisasi sistem.

Python menawarkan keseimbangan antara kejelasan *syntax* dan fleksibilitas dalam pengembangan alat-alat penelitian komputasi, sehingga sangat mendukung dalam menciptakan solusi untuk berbagai jenis permasalahan yang kompleks. Bahasa ini dirancang untuk menangani beragam tantangan yang melibatkan pengolahan *dataset* berukuran besar, penerapan algoritma yang rumit, serta pengembangan sistem komputasi (Pérez, Granger & Hunter, 2011). Kemampuan Python untuk berintegrasi dengan berbagai pustaka dan *framework* membuatnya menjadi pilihan utama dalam penelitian berbasis data dan pengembangan teknologi

inovatif. Dengan ekosistem yang luas, Python memungkinkan peneliti dan pengembang untuk membangun, menguji, serta mengimplementasikan solusi secara efisien dan *scalable*.

2.18.2 Pandas

Pandas adalah pustaka Python berperforma tinggi yang dirancang khusus untuk manipulasi, analisis, dan eksplorasi data. Pustaka ini banyak digunakan oleh peneliti data, analis, dan pengembang karena kemampuannya yang unggul dalam mengolah data secara efisien (Molin & Jee, 2021). Pandas menyediakan berbagai fungsi yang memudahkan proses pembersihan, transformasi, serta analisis data dalam berbagai format, seperti tabel, *file* CSV, dan *database*. Selain itu, Pandas juga mendukung integrasi dengan pustaka visualisasi seperti Matplotlib dan Seaborn, sehingga memungkinkan pengguna untuk membuat visualisasi data yang informatif dan menarik. Kemudahan penggunaan serta fleksibilitas Pandas menjadikannya salah satu alat utama dalam analisis data modern dan pengembangan aplikasi berbasis data.

Salah satu kekuatan utama dari pustaka ini adalah penggunaan data *frame* dan *series*, yang menjadi inti dalam proses manipulasi, perhitungan, serta analisis data (Nelli, 2015). Data *frame* adalah struktur data berbentuk tabel dengan label pada baris dan kolom, mirip dengan tabel pada *database* atau *spreadsheet*, sehingga memudahkan pengolahan data dalam jumlah besar. Sementara itu, *series* merupakan struktur data satu dimensi yang berfungsi seperti *array*, tetapi dilengkapi dengan indeks yang memungkinkan akses data lebih fleksibel. Kombinasi dari dua struktur data ini memungkinkan pengguna untuk melakukan

berbagai operasi analisis secara efisien, seperti pengolahan data numerik, transformasi data, serta agregasi hasil analisis dengan *syntax* yang sederhana namun *powerful*.

2.18.3 Scikit-learn

Scikit-learn merupakan pustaka Python yang menyediakan antarmuka standar untuk mengimplementasikan berbagai algoritma *machine learning*. Pustaka ini dirancang agar mudah digunakan, sehingga memudahkan pengguna dari berbagai latar belakang untuk mengembangkan model *machine learning* dengan lebih efisien. Selain mendukung algoritma untuk klasifikasi, regresi, dan *clustering*, Scikit-learn juga dilengkapi dengan berbagai fungsi penting lainnya, seperti data *preprocessing*, *resampling*, evaluasi model, serta pencarian *hyperparameter*. Fungsi-fungsi tersebut membantu memastikan bahwa proses pengolahan data, pelatihan model, hingga evaluasi dapat dilakukan secara menyeluruh dan sistematis (Bisong, 2019).

2.18.4 Matplotlib

Matplotlib adalah pustaka Python yang digunakan untuk pembuatan grafik dan visualisasi data. Pustaka ini menyediakan berbagai fitur yang memungkinkan pengguna untuk membuat beragam jenis grafik dan diagram, mulai dari grafik garis (*line plot*), grafik sebar (*scatter plot*), peta panas (*heatmap*), diagram batang (*bar chart*), diagram lingkaran (*pie chart*), hingga visualisasi data dalam bentuk tiga dimensi (3D plot) (Hunt, 2019). Kemampuan Matplotlib dalam menghasilkan visualisasi yang informatif dan berkualitas tinggi menjadikannya salah satu alat

utama bagi peneliti dan analis data. Selain itu, pustaka ini mendukung kustomisasi penuh pada setiap elemen grafik, seperti warna, label, dan sumbu, sehingga memudahkan pengguna untuk menyajikan data secara lebih menarik dan sesuai dengan kebutuhan analisis.

2.18.5 Seaborn

Seaborn adalah pustaka Python yang dirancang untuk membuat visualisasi grafik statistik dengan cara yang lebih mudah dan estetis. Pustaka ini menyediakan antarmuka tingkat tinggi untuk Matplotlib, sehingga memungkinkan pengguna membuat grafik kompleks dengan sedikit kode (Waskom, 2021). Seaborn juga terintegrasi erat dengan Pandas, sehingga pengguna dapat langsung memvisualisasikan data dari struktur data *frame* tanpa perlu konversi tambahan. Dengan berbagai fitur bawaan, seperti pembuatan grafik hubungan antar variabel, distribusi data, serta anotasi statistik, Seaborn membantu dalam menyajikan visualisasi data yang informatif dan menarik. Kemudahan penggunaan serta desain visual yang lebih elegan membuat Seaborn menjadi pilihan utama bagi analis data dan ilmuwan data yang ingin meningkatkan kualitas visualisasi mereka.

2.19 Penelitian Sejenis

Penelitian sejenis yang digunakan pada penelitian ini ditunjukkan pada Tabel 2.2.

Tabel 2.2 Penelitian Sejenis

No	Penulis	Domain Riset	Metode dan *Tools*	*Dataset* (Populasi, Sampel)	Kontribusi	Hasil
1	Pollard & Reep (1997)	Pemodelan kualitas tembakan dan cikal bakal metrik xG	Regresi Logistik	22 pertandingan, 489 tembakan	Salah satu studi perintis dalam kuantifikasi xG, membuktikan pentingnya lokasi tembakan.	• Persamaan probabilitas gol (regresi logistik) • Koefisien Jarak (*X*): -0.096 (semakin jauh, peluang turun) • Koefisien Sudut (*A*): -1.037 (semakin menyamping, peluang turun)
2	Ensum, Pollard, & Taylor (2005)	Pengembangan model probabilitas gol melalui analisis multivariat kontekstual	Regresi Logistik	48 pertandingan, 1.099 tembakan (*FA Premier League* & *World Cup*)	Konfirmasi dan perluasan temuan awal dengan *dataset* yang lebih besar.	• Persamaan probabilitas gol (regresi logistik) • Koefisien Jarak: -0.16 • Koefisien Sudut: -1.24 • Koefisien Sundulan: -0.73 • Koefisien Kaki Lemah: -0.63 • Koefisien Tendangan Voli: -0.27 (Semua koefisien negatif menunjukkan penurunan probabilitas gol dibandingkan kondisi ideal).
3	Lucey *et al.* (2015)	Analitika video *spasiotemporal* untuk pemodelan peluang gol	*Conditional Random Fields*	Data Prozone/Stats Perform (~9.732 tembakan + 10 detik video pra-tembakan)	Menggabungkan fitur strategis & *spasiotemporal* (fase permainan, interaksi pemain) dari data video.	ROC AUC: • AUC Model *Baseline* (Hanya Lokasi): 0.75 • AUC Model *Spasiotemporal* (EGV): 0.81

No	Penulis	Domain Riset	Metode dan *Tools*	*Dataset* (Populasi, Sampel)	Kontribusi	Hasil
4	Ruiz *et al*. (2015)	*Machine learning* untuk evaluasi kemampuan *finishing* pemain	*Multilayer Perceptron* (MLP)	Data Prozone (EPL 2013/14, 10.318 tembakan)	Aplikasi model non-linear (MLP) untuk mengidentifikasi efisiensi pemain secara individual.	*p-value*: • Tingkat Signifikansi Sangat Tinggi (p 0.01): - Sergio Agüero - Wayne Rooney - Luis Suárez • Tingkat Signifikansi Tinggi (p 0.05) - Daniel Sturridge - Yaya Touré Nilai *p-value* ini menunjukkan bahwa kemampuan individu para pemain ini dalam mencetak gol secara signifikan melebihi ekspektasi model xG standar.
5	Eggels *et al.* (2016)	Analitik prediktif untuk hasil pertandingan	• Regresi Logistik • *Decision Tree* • *Random Forest* • AdaBoost • Python (scikit-learn)	Data *event*/*tracking* ORTEC & *Inmotio* + atribut EA Sports (~20.000 tembakan)	Perbandingan komprehensif berbagai model *machine learning* untuk prediksi xG.	ROC AUC: AdaBoost = 0,84 *Random Forest* = 0,82 Regresi Logistik = 0,78 *Decision Tree* = 0,74
6	Fairchild *et al.* (2018)	Analitika terapan pemodelan xG yang praktis dan terinterpretasi	• Regresi Logistik • Python • SciPy/Statsmodels	1.115 tembakan non-penalti dari 99 pertandingan MLS 2016 (data *tag* manual)	Aplikasi model sederhana yang dapat diinterpretasikan pada *dataset* yang lebih kecil	Kalibrasi model yang kuat dengan validasi silang ROC AUC = 0,80.
7	Pardo (2020)	Pengayaan model xG dengan atribut pemain dari data eksternal	• Regresi Logistik • XGBoost • ANN • *scikit-learn*	Data OPTA (~20.000 tembakan) + atribut pemain FIFA (740 pemain)	Integrasi atribut kualitatif pemain (dari *game* FIFA) ke dalam model xG.	ROC AUC: • ANN = 0,88 • XGBoost = 0,85 • Regresi Logistik = 0,78. RMSE:

No	Penulis	Domain Riset	Metode dan *Tools*	*Dataset* (Populasi, Sampel)	Kontribusi	Hasil
			• Keras			• ANN = 0,25 • XGBoost = 0,27 • Regresi Logistik = 0,32.
8	Wheatcroft & Sienkiewicz (2021)	Pemodelan probabilistik kesuksesan tembakan	• Model probabilistik parametrik • Python (SciPy *optimize*)	>1 juta tembakan dari 22 liga (football-data.co.uk)	Pengembangan model yang sangat sederhana dan cepat untuk *pipeline* prediksi.	Peningkatan *Log-Score* (*log-likelihood per shot*) dibandingkan model dasar: • *Log-Score* Model Mereka: -0.312 • *Log-Score Baseline Naive*: -0.342 • Peningkatan vs. *Baseline Naive*: +0.030 • Peningkatan vs. *Baseline* Pollard & Reep: +0.004
9	Cavus & Biecek (2022)	*Explainable AI* (XAI) untuk interpretabilitas model xG kompleks	XGBoost, RF, LightGBM, CatBoost (via AutoML)	Data *event* Understat (315.430 tembakan)	Pionir dalam menerapkan AutoML untuk eksplorasi model dan SHAP untuk interpretabilitas model kompleks.	*Random Forest* menjadi model terbaik: • ROC AUC (*Random Forest*): 0.875 • Brier Score (*Random Forest*): 0.072 Performa model lain: • AUC (LightGBM): 0.873 • AUC (CatBoost): 0.872 • AUC (XGBoost): 0.869
10	Méndez *et al*. (2023)	Peningkatan nilai xG dengan jaringan saraf	• *Multilayer Perceptron* (MLP) • Python • Keras	Data *event* StatsBomb (>12.000 tembakan)	Menunjukkan superioritas MLP dalam menangkap pola non-linear dibandingkan regresi logistik.	MLP secara konsisten mengungguli Regresi Logistik (LR) di semua metrik: • ROC AUC: 0.87 (MLP) vs. 0.82 (LR) • Akurasi: 90.04% (MLP) vs. 89.28% (LR)

No	Penulis	Domain Riset	Metode dan *Tools*	*Dataset* (Populasi, Sampel)	Kontribusi	Hasil
						• Brier Score: 0.076 (MLP) vs. 0.081 (LR) • Log-Loss: 0.262 (MLP) vs. 0.280 (LR) • F1-Score: 0.41 (MLP) vs. 0.34 (LR) Arsitektur MLP: 2 *hidden layers* (16 dan 8 neuron, aktivasi *ReLU*)
11	Mead *et al.* (2023)	Peningkatan performa & demonstrasi nilai model xG	• Regresi Logistik, • RF, • AdaBoost, • XGBoost, • Python (scikit-learn)	Data Wyscout (~250.000 tembakan)	Menunjukkan peningkatan performa signifikan dengan fitur tambahan (nilai pemain, *rating* ELO).	Perbandingan ROC AUC yang menunjukkan keunggulan *Random Forest* dengan fitur yang diperkaya: • *Random Forest* (Fitur Diperkaya): 0.910 • *Random Forest* (Fitur Dasar): 0.891 • Regresi Logistik (Fitur Dasar): 0.852
12	Hewitt & Karakuş (2023)	Pemodelan xG kontekstual berbasis identitas dan peran pemain	• *Regresi Logistik,* • *Gradient Boosted Trees (GBT),* • *scikit-learn,*	Data *event* StatsBomb (15.574 tembakan) dari 5 liga top Eropa musim 2021/22	Mengembangkan model yang menyesuaikan nilai xG dengan kemampuan pemain dan posisi.	• Korelasi Pearson antara xG dan gol lebih tinggi pada GBT (0,208) dibandingkan Regresi Logistik (0,188). • Analisis pada 347 tembakan Lionel Messi menunjukkan model ini memberikan nilai xG +3,74 lebih tinggi daripada model dasar.
13	Bandara *et al.* (2024)	Pemodelan xG sekuensial berbasis	• *Random Forest* (100 *estimators*), • scikit-learn	Data dari 990 pertandingan (StatsBomb Open	Inovasi dengan menggunakan fitur dari 3 urutan *event* sebelum	• Model berbasis sekuens mencapai ROC AUC 0,833 pada set validasi.

No	Penulis	Domain Riset	Metode dan *Tools*	*Dataset* (Populasi, Sampel)	Kontribusi	Hasil
		aliran *event* pra-tembakan		Data), mencakup kompetisi seperti Piala Dunia, Euro, Liga *Champions*, dll.	tembakan, bukan hanya data tembakan tunggal.	• Pada pengujian dengan data Euro 2020, model mencapai ROC AUC 0,826, mengungguli model tembakan tunggal (*baseline*).
14	Cefis & Carpita (2024)	Pemodelan statistik xG dengan pengayaan fitur multi-sumber	• Regresi Logistik • R (fungsi glm)	Data dari 3 sumber: Understat, SoFIFA, Math&Sport. Total 49.872 tembakan dari 5 liga top Eropa musim 2022/23.	Mengintegrasikan fitur-fitur baru seperti tekanan pada penembak, *rating* pemain (SoFIFA), dan kekuatan lawan ke dalam model linier (Regresi Logistik).	• ROC AUC = 0,812 • *Brier Score* = 0,078 Hasil ini menunjukkan bahwa model linier yang lebih sederhana masih bisa mencapai performa yang kuat jika diperkaya dengan fitur-fitur yang relevan dan inovatif.
15	Xu *et al.* (2025)	*Computer vision* dan pemodelan xG berbasis analisis pose tubuh	• Jaringan Saraf Konvolusional (CNN) • Regresi Logistik • Analisis *Pose* (OpenPose)	• Data Publik: 477 tembakan dari *dataset* publik. • Data SoccerNet-v2: 927 tembakan dari 500 pertandingan.	• Pionir dalam menggunakan data pose/kerangka tubuh pemain (*skeleton* data) secara langsung untuk estimasi xG. • Memperkenalkan model Skor-xG yang mengintegrasikan orientasi tubuh pemain dan posisi kiper.	• ROC AUC: Model Skor-xG mencapai 0,845, mengungguli model Regresi Logistik (0,791). • Brier Score: Skor-xG mendapatkan 0,068, lebih baik dari Regresi Logistik (0,075). (Nilai lebih rendah lebih baik).

Berdasarkan Tabel 2.2, terdapat lima belas penelitian yang mengkaji perhitungan metrik *Expected Goals* (xG) dalam analisis sepak bola. Penelitian-penelitian ini dapat dikelompokkan berdasarkan metodologi yang digunakan, mulai dari model statistik sederhana hingga pendekatan *deep learning* yang kompleks.

Kelompok pertama mencakup penelitian yang mengandalkan model statistik yang lebih mudah diinterpretasi. Studi perintis oleh Pollard & Reep (1997) dan kelanjutannya oleh Ensum, Pollard, & Taylor (2005) menerapkan Regresi Logistik untuk mengukur efektivitas tembakan. Pendekatan serupa juga digunakan oleh Fairchild *et al*. (2018) untuk analisis spasial di liga MLS, serta Cefis & Carpita (2024) yang memperkaya model Regresi Logistik dengan fitur inovatif seperti tekanan pada penembak dan kekuatan lawan. Selain itu, terdapat pendekatan model probabilistik parametrik yang efisien seperti yang ditunjukkan oleh Wheatcroft & Sienkiewicz (2021).

Sebagai alternatif, banyak penelitian memanfaatkan *machine learning* untuk menangkap pola non-linear yang lebih kompleks. Beberapa di antaranya berfokus pada perbandingan komprehensif berbagai algoritma. Contohnya, Eggels *et al*. (2016) menguji empat model berbeda dan menemukan AdaBoost memiliki performa terbaik, sementara Cavus & Biecek (2022) menggunakan AutoML yang menobatkan *Random Forest* sebagai model paling optimal. Mead *et al*. (2023) juga mengonfirmasi keunggulan *Random Forest*, terutama setelah diperkaya dengan fitur tambahan seperti nilai pemain dan rating ELO.

Model jaringan saraf juga menjadi pilihan populer. Ruiz *et al*. (2015) dan Méndez *et al*. (2023) secara efektif menggunakan *Multilayer Perceptron* (MLP)

untuk menunjukkan superioritas model non-linear dalam menangkap pola kompleks dibandingkan regresi logistik. Pardo (2020) turut membandingkan *Artificial Neural Network* (ANN) dengan XGBoost untuk menganalisis pengaruh informasi kualitatif pemain terhadap kualitas peluang.

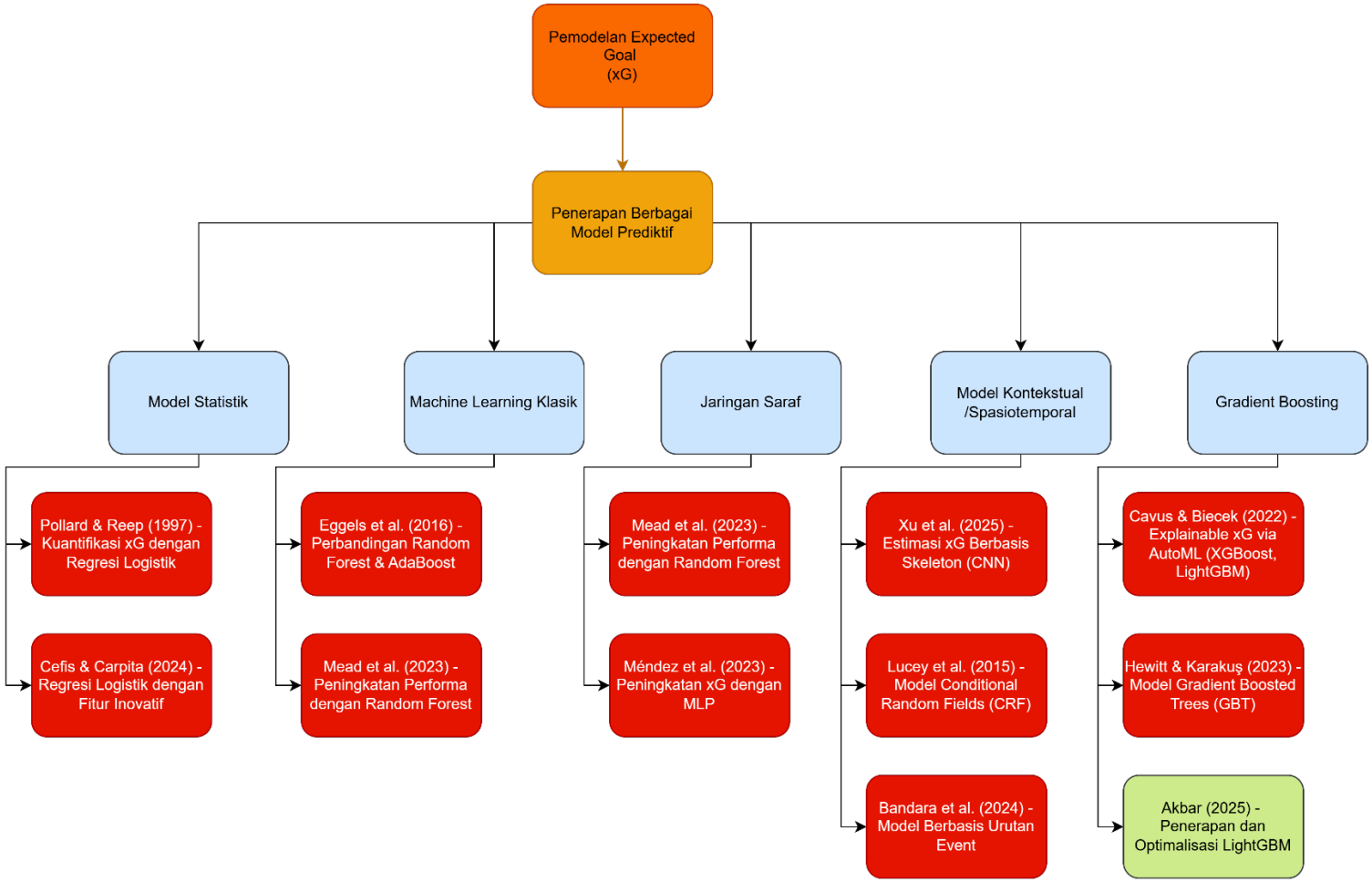
Lebih lanjut, terdapat penelitian yang mengembangkan model dengan fitur-fitur yang lebih canggih dan kontekstual. Lucey *et al*. (2015) menerapkan *Conditional Random Fields* untuk memasukkan informasi spasiotemporal, sementara Bandara *et al*. (2024) menggunakan *Random Forest* dengan fitur sekuensial dari tiga kejadian terakhir sebelum tembakan. Hewitt & Karakuş (2023) mengembangkan model *Gradient Boosted Trees* yang dapat menyesuaikan nilai xG dengan kemampuan individu pemain. Inovasi paling mutakhir datang dari Xu *et al*. (2025) yang memelopori penggunaan data pose tubuh pemain (*skeleton data*) dengan *Convolutional Neural Network* (CNN) untuk menghasilkan estimasi xG yang lebih akurat.

Penelitian ini bertujuan untuk menerapkan LightGBM untuk perhitungan metrik xG dalam analisis sepak bola. Perbedaan penelitian ini dengan penelitian sebelumnya yaitu:

a. Fokus pada penggunaan LightGBM sebagai metode utama untuk perhitungan xG, yang dikenal karena efisiensi dan kecepatan komputasinya.
b. Mengeksplorasi kemampuan LightGBM dalam menangani data sepak bola, yang sering kali memiliki interaksi fitur kompleks dan non-linear, dibandingkan dengan metode yang lebih sederhana seperti regresi logistik.

2.20 Ranah Penelitian

Pada tahap ini, menggambarkan ranah penelitian sejenis yang dilakukan penulis berdasarkan literatur yang dibandingkan, dimana berbagai metode mulai dari statistik hingga *machine learning* digunakan pada bidang analisis olahraga. Berdasarkan penelitian sejenis yang identik dengan ranah penelitian sebelumnya, maka ranah penelitian penulis berkaitan dengan bidang analisis sepak bola. Gambar 2.11 adalah ilustrasi dari ranah penelitian.



Gambar 2.11 Ranah Penelitian

Ranah pada penelitian ini adalah berfokus pada penerapan dan optimalisasi algoritma LightGBM untuk prediksi nilai xG. Data yang digunakan dalam penelitian ini adalah data *event* pertandingan sepak bola yang bersumber dari StatsBomb *Open Data*, yang mencakup berbagai kompetisi ternama di dunia. Hal yang membedakan penelitian ini dengan penelitian-penelitian sebelumnya adalah fokusnya yang mendalam pada optimalisasi algoritma LightGBM. Jika penelitian sebelumnya hanya menyertakan LightGBM sebagai salah satu pembanding dalam kerangka kerja AutoML, penelitian ini secara spesifik melakukan proses *hyperparameter tuning* dan kalibrasi untuk menggali potensi akurasi dan efisiensi komputasi yang sesungguhnya dari model. Selain itu, evaluasi model dilakukan secara komprehensif menggunakan serangkaian metrik holistik untuk memberikan gambaran kinerja yang utuh.

BAB 3

METODOLOGI PENELITIAN

3.1 Pendekatan Penelitian

Penelitian ini menginvestigasi kinerja metode LightGBM dalam memprediksi nilai xG dari data tembakan pada pertandingan sepak bola dengan menggunakan pendekatan kuantitatif. Penelitian ini menggunakan bahasa pemrograman Python dan platform Google Colaboratory untuk proses pengambilan, pembersihan, dan pemodelan data. *Dataset* diambil dari repositori terbuka StatsBomb yang tersedia di GitHub. Microsoft Word digunakan untuk penyusunan laporan penelitian.

3.2 Sumber Data

Penelitian ini menggunakan pendekatan kuantitatif dengan memanfaatkan data sekunder yang bersifat publik (*publicly available*). Data yang digunakan bersumber dari StatsBomb Open Data, yang disediakan secara resmi dan gratis melalui repositori GitHub untuk mendorong riset serta inovasi di bidang analisis sepak bola.

Dalam konteks penelitian ini, populasi merujuk pada keseluruhan data peristiwa (*event data*) yang tersedia dalam repositori tersebut, sedangkan sampel adalah *subset* data yang dipilih secara spesifik untuk tujuan membangun dan mengevaluasi model prediksi xG.

3.2.1 Populasi dan Sampel Penelitian

Populasi data dalam penelitian ini adalah keseluruhan data peristiwa dari ribuan pertandingan sepak bola yang tersedia dalam StatsBomb Open Data. Populasi ini mencakup berbagai kompetisi elit di seluruh dunia:

a. Liga-liga top Eropa (Contoh: La Liga, Serie A, Bundesliga).
b. Kompetisi antarklub Eropa (Contoh: UEFA *Champions League*).
c. Turnamen internasional (Contoh: FIFA *World Cup*, UEFA Euro).

Dari populasi yang luas tersebut, sampel penelitian diambil menggunakan metode *purposive sampling* (pengambilan sampel bertujuan). Metode ini diterapkan untuk memilih data yang paling relevan dengan tujuan pemodelan xG. Sampel akhir yang terkumpul adalah data peristiwa tembakan (*shot*) yang memenuhi serangkaian kriteria seleksi yang ketat.

3.2.2 Kriteria Seleksi Sampel

Proses seleksi sampel dilakukan untuk memastikan homogenitas data, karena probabilitas gol dari situasi yang berbeda memiliki karakteristik yang sangat berbeda dan sering kali dimodelkan secara terpisah. Kriteria seleksi yang diterapkan adalah sebagai berikut:

a. Jenis Peristiwa (*event type*): Hanya peristiwa dengan jenis tembakan (*shot*) yang dimasukkan ke dalam *dataset*. Peristiwa lain seperti umpan, tekel, atau dribel akan diekstraksi untuk menghasilkan fitur turunan namun tidak menjadi bagian dari sampel utama.
b. Situasi Permainan (*play pattern*): Sampel dibatasi hanya pada tembakan yang terjadi dari situasi permainan terbuka (*open play*). Tembakan yang berasal dari

situasi bola mati seperti penalti (*penalty*), tendangan bebas langsung (*direct free-kick*), atau situasi setelah tendangan sudut (*corner*) tidak diikutsertakan. Hal ini dilakukan untuk menjaga homogenitas data, karena probabilitas gol dari situasi bola mati memiliki karakteristik yang sangat berbeda dan sering dimodelkan secara terpisah.

Setelah melalui proses seleksi ini, terkumpul puluhan ribu data tembakan yang menjadi sampel final penelitian, yang kemudian dibagi menjadi data latih (training data) dan data uji (testing data) untuk keperluan pemodelan.

3.3 Perangkat Penelitian

Penelitian ini menggunakan perangkat keras (*hardware*) dan perangkat lunak (*software*) dengan spesifikasi yang dijelaskan pada Tabel 3.1.

Tabel 3.1 Spesifikasi *Hardware* dan *Software*

Hardware	Laptop Lenovo ADA 11	AMD Athlon Gold 3150U *with* Radeon Graphics 2.40 GHz
		12.0 GB RAM
		256 GB SSD
		Monitor 15 Inch
Software	Sistem Operasi	Windows 11 Home
	Tools	Google Colaboratory
	Bahasa Pemrograman	Python

3.4 Pengumpulan Data

Data yang digunakan dalam penelitian ini terdiri atas data primer dan sekunder. Data primer diperoleh dari *dataset* terbuka yang disediakan oleh StatsBomb melalui repositori GitHub. Sementara itu, data sekunder diperoleh dari

berbagai jurnal ilmiah, buku, dan sumber internet yang relevan dengan topik penelitian, khususnya yang berkaitan dengan analisis xG, pemodelan prediktif, dan algoritma LightGBM.

Pengambilan data dilakukan dengan mengunduh *dataset* event pertandingan sepak bola dari repositori *open-source* StatsBomb di GitHub. Proses ini dilakukan menggunakan skrip Python di platform Google Colaboratory. *Dataset* yang digunakan mencakup data tembakan dalam pertandingan, termasuk informasi seperti lokasi, jarak, sudut tembakan, serta atribut kontekstual lainnya yang mendukung perhitungan nilai xG.

3.5 Pengembangan Model

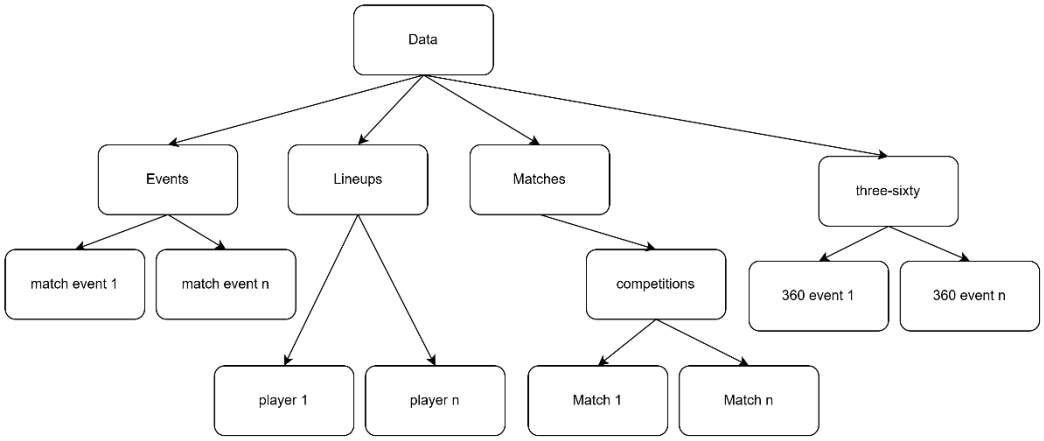
3.5.1 *Knowledge Discovery in Databases*

Penelitian ini menggunakan pendekatan *Knowledge Discovery in Databases* (KDD) dalam proses pengembangan model. Metode KDD memiliki keunggulan dalam membantu mengidentifikasi pola tersembunyi dari kumpulan data yang kompleks sehingga dapat menghasilkan informasi yang lebih mudah dipahami. Proses KDD terdiri dari beberapa tahapan, yaitu: *preprocessing* data, pemilihan data (*data selection*), transformasi data, proses *data mining*, dan evaluasi pengetahuan yang diperoleh (*knowledge evaluation*) (Ramos *et al*., 2021).

a. *Data Selection*

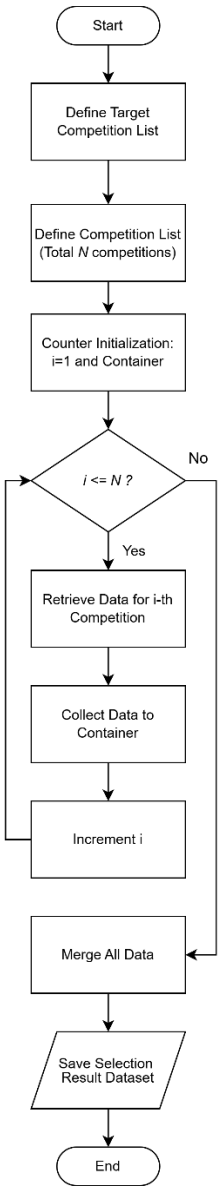
   Data dari StatsBomb *open-data* diambil dengan mengakses repositori resmi di GitHub. Pertama, kita perlu mengidentifikasi kompetisi apa saja yang tersedia dalam *dataset*. Setiap kompetisi kemudian terdiri dari beberapa musim (edisi),

dan masing-masing musim ini mewakili rentang waktu berlangsungnya pertandingan yang terdokumentasi. Di dalam setiap musim terdapat fase-fase pertandingan: untuk kompetisi sistem gugur biasanya meliputi babak perempat final, semi final, final, dan seterusnya, sedangkan untuk liga reguler umumnya hanya ada satu fase liga utama, dengan beberapa kompetisi seperti, Piala FA yang juga memiliki babak *play-off*. Setelah fase-fase ditentukan, barulah kita mengakses data pertandingan. Dalam konteks StatsBomb, satu pertandingan terdiri dari serangkaian *event*, dan masing-masing *event* ini dapat memiliki *event* terkait. Misalnya, sebuah tusukan (*dribble*) bisa jadi dipicu oleh operan rekan tim yang sebelumnya dieksekusi operan tersebut, kemudian tercatat sebagai *event* terkait. Namun, karena operan juga tercatat sebagai *event* utama, jika kita menarik semua *event* terkait tanpa seleksi, kita akan mendapati banyak duplikasi operan tercatat dua kali, sekali sebagai *event* utama dan sekali lagi sebagai *event* terkait. Sebaliknya, jika kita sama sekali mengabaikan *event* terkait, kita bisa kehilangan jejak kronologi aksi yang sebenarnya terjadi di lapangan. Untuk mengatasi masalah ini, saat ini hanya situasi gol dan kartu (kuning/merah) yang diikutkan sebagai *event* terkait dalam pemrosesan data StatsBomb. Dengan begitu, kita tetap menjaga konteks penting seperti *assist* sebelum gol atau pelanggaran yang berujung kartu tanpa menumpuk terlalu banyak duplikasi. Pada Gambar 3.1 dijelaskan struktur data yang dimiliki oleh StatsBomb *open-data.*



Gambar 3.1 Struktur Data StatsBomb *open-data.*

Data *event* dari StatsBomb disediakan dalam format JSON pada repositori GitHub mereka. Karena pengambilan data langsung dari GitHub juga memakan waktu, biasanya *file* JSON tersebut diunduh sekali saja lalu dikonversi dan disimpan dalam format *Parquet* untuk penggunaan selanjutnya. Dengan cara ini, analisis bisa dilakukan lebih cepat tanpa perlu terus-menerus mengunduh data mentah. Gambar 3.2 menunjukkan *flowchart* dari *data selection.*



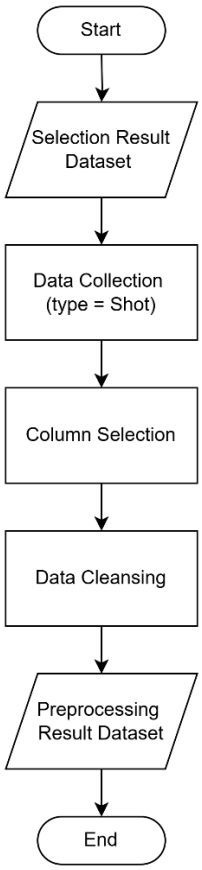
Gambar 3.2 *Flowchart Data Selection*

b. *Data Preprocessing*

Tahap data *preprocessing* bertujuan untuk menyiapkan data hasil seleksi agar dapat dianalisis secara optimal dan digunakan dalam proses pelatihan model. Proses ini disesuaikan dengan karakteristik data *event* sepak bola yang diperoleh dari StatsBomb, yang memiliki struktur sangat baik dan konsisten

sehingga mempermudah proses pembersihan dan pengolahan data. Langkah pertama adalah pemilihan data, yaitu dengan mengambil hanya *event* yang bertipe *Shot* dan berasal dari situasi permainan terbuka (*open play*), karena jenis tembakan ini paling relevan dalam konteks prediksi *expected goals*. Setelah itu, dilakukan pemilihan kolom dengan memilih fitur-fitur yang berpotensi mendukung prediksi, seperti posisi tembakan, bagian tubuh yang digunakan, tekanan lawan, serta pola permainan. Kolom-kolom yang bersifat administratif atau tidak relevan terhadap tujuan model, seperti nama pemain dan identifikasi pertandingan, tidak disertakan.

Langkah terakhir adalah data *cleansing*, yang meliputi pemeriksaan nilai kosong dan duplikat. Namun, karena data StatsBomb memiliki format yang sangat terstruktur dan tiap peristiwa dalam pertandingan bersifat unik, data yang diperoleh relatif bersih dan tidak memerlukan proses pembersihan lanjutan. Tahapan *preprocessing* ini menghasilkan *dataset* yang konsisten, bebas duplikasi, dan siap untuk dianalisis lebih lanjut pada tahap transformasi dan pemodelan. Pada Gambar 3.2 dijelaskan *flowchart* dari tahap *preprocessing.*

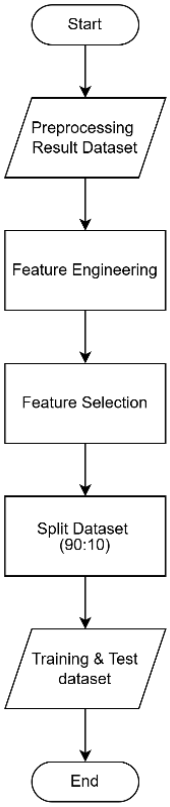


Gambar 3.3 *Flowchart Data Preprocessing*

c. *Data Transformation*

Tahap ini bertujuan untuk memperkaya representasi data agar dapat meningkatkan performa model pada tahapan *data mining*. Pertama, dilakukan proses *feature engineering* untuk menciptakan fitur-fitur baru yang merepresentasikan dinamika permainan secara lebih mendalam. Transformasi ini memungkinkan data mentah memberikan wawasan yang lebih bermakna dan relevan dalam konteks prediksi performa tembakan. Fitur-fitur seperti jarak dan sudut tembakan ke gawang serta segmentasi waktu pertandingan ditambahkan untuk memperkaya informasi spasial dan temporal. Setelah fitur baru

ditambahkan, data kemudian dibagi menjadi data latih dan data uji agar proses pelatihan dan evaluasi model dapat dilakukan secara terpisah. Alur tahapan *transformation* ditunjukkan pada Gambar 3.3.

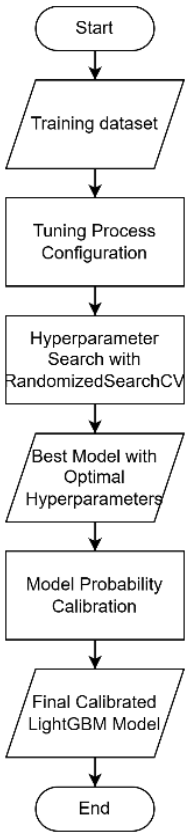


Gambar 3.4 *Flowchart Data Transformation*

d. *Data Mining*

Pada tahapan ini pemodelan xG dilakukan menggunakan algoritma LightGBM. Namun sebelum model dilatih, terdapat beberapa proses penting yang harus dilakukan, yaitu pencarian *hyperparameter* terbaik dan kalibrasi probabilitas. Pencarian *hyperparameter* dilakukan dengan menggunakan *RandomizedSearchCV* sebanyak 100 iterasi, yang mengevaluasi berbagai kombinasi parameter dengan 5-*fold cross-validation*. Proses ini menggunakan

metrik skor *roc_auc* sebagai acuan untuk menentukan kombinasi parameter terbaik dan secara otomatis melakukan *refit* pada model dengan skor tersebut. Setelah memperoleh model dengan konfigurasi terbaik, dilakukan kalibrasi probabilitas menggunakan *CalibratedClassifierCV* untuk memastikan bahwa prediksi probabilitas dari model merefleksikan tingkat kepercayaannya secara akurat (Davis & Robberechts, 2024). Selain pelatihan dan kalibrasi, tahap ini juga mencakup analisis fitur untuk memahami kontribusi tiap variabel dalam proses prediksi. Gambar 3.4 menunjukkan alur dari tahapan data *mining* dalam penelitian ini.

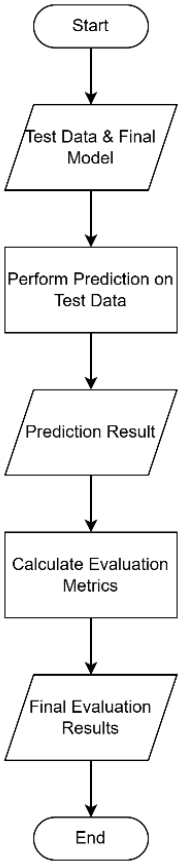


Gambar 3.5 *Flowchart Data Mining*

e. *Evaluation*

Setelah proses *data mining* selesai, tahap selanjutnya adalah evaluasi terhadap model yang telah dibuat. Evaluasi ini bertujuan untuk mengukur performa model secara komprehensif terhadap data uji. Sesuai dengan batasan masalah, evaluasi kinerja model akan menggunakan serangkaian metrik yang mencakup ROC AUC, *Brier Score*, presisi, *recall*, F1-Score, dan *Log-Loss*.

ROC AUC digunakan untuk menilai kemampuan diskriminatif model, yaitu kemampuannya dalam membedakan antara kelas positif dan negatif secara keseluruhan tanpa terikat pada ambang batas klasifikasi tertentu. Untuk mengukur akurasi dari prediksi probabilistik, digunakan *Brier Score* yang menghitung rata-rata selisih kuadrat antara probabilitas prediksi dengan hasil aktual, sehingga efektif dalam menilai kalibrasi model. Serupa dengan itu, *Log-Loss* juga memberikan penalti untuk prediksi yang tingkat keyakinannya tidak sesuai dengan hasil aktual. Terakhir, untuk evaluasi yang lebih bernuansa pada tugas klasifikasi, digunakan presisi, *recall*, dan F1-Score yang menganalisis keseimbangan antara keandalan prediksi positif (Presisi) dan kelengkapan dalam mengidentifikasi kasus positif (*recall*). *Flowchart* dari tahapan evaluasi model ditunjukkan pada Gambar 3.5.



Gambar 3.6 *Flowchart Evaluation*

3.5.2 Pemodelan LightGBM

Pada penelitian ini, metode yang digunakan adalah LightGBM (*Light Gradient Boosting Machine*) untuk membangun model prediksi. LightGBM dirancang untuk menangani data berukuran besar dengan efisiensi tinggi melalui dua teknik utama: *Gradient-based One-Side Sampling* (GOSS) dan *Exclusive Feature Bundling* (EFB).

Teknik GOSS berfokus pada efisiensi pelatihan model dengan mempertahankan seluruh data yang memiliki nilai gradien besar yang mengandung lebih banyak informasi dan secara acak mengambil sebagian dari data dengan

gradien kecil (Ke *et al*., 2017). Namun, karena proses ini dapat mengubah distribusi data asli, LightGBM memperkenalkan pengali konstan saat menghitung *information gain* untuk data dengan gradien kecil guna menyeimbangkan kontribusi antara dua kelompok data tersebut. Pendekatan ini memungkinkan model untuk tetap fokus pada sampel yang paling berpengaruh terhadap pembaruan model tanpa kehilangan akurasi secara signifikan.

Sementara itu, teknik EFB dirancang untuk mengatasi tantangan ketika terdapat banyak fitur yang bersifat saling eksklusif, yaitu fitur-fitur yang tidak pernah aktif secara bersamaan. Algoritma ini menggabungkan fitur-fitur eksklusif tersebut ke dalam fitur padat (*dense feature*) dalam jumlah yang jauh lebih sedikit, sehingga mengurangi dimensi data dan beban komputasi (Ke *et al*., 2017). Selain itu, LightGBM juga mengoptimalkan algoritma histogram dasar dengan cara mengabaikan nilai nol pada fitur, yakni dengan mencatat hanya nilai-nilai non-nol menggunakan struktur data khusus. Kombinasi dari GOSS dan EFB menjadikan LightGBM sangat efisien dan *scalable* dalam membangun model prediksi dari *dataset* dengan jumlah *instance* dan fitur yang sangat besar.

3.6 Analisis Data dan Intrepretasi Hasil

Analisis data dalam penelitian ini dilakukan berdasarkan pendekatan KDD (*Knowledge Discovery in Database*) yang mencakup lima tahapan utama: data *selection, preprocessing, transformation,* data *mining*, dan *evaluation*. Proses analisis dimulai dari tahap data *selection*, yaitu dengan menyiapkan *dataset* yang relevan untuk membangun model prediksi. Tahap selanjutnya adalah *preprocessing*

yang meliputi pembersihan data, penanganan *missing value*, penghapusan duplikasi.

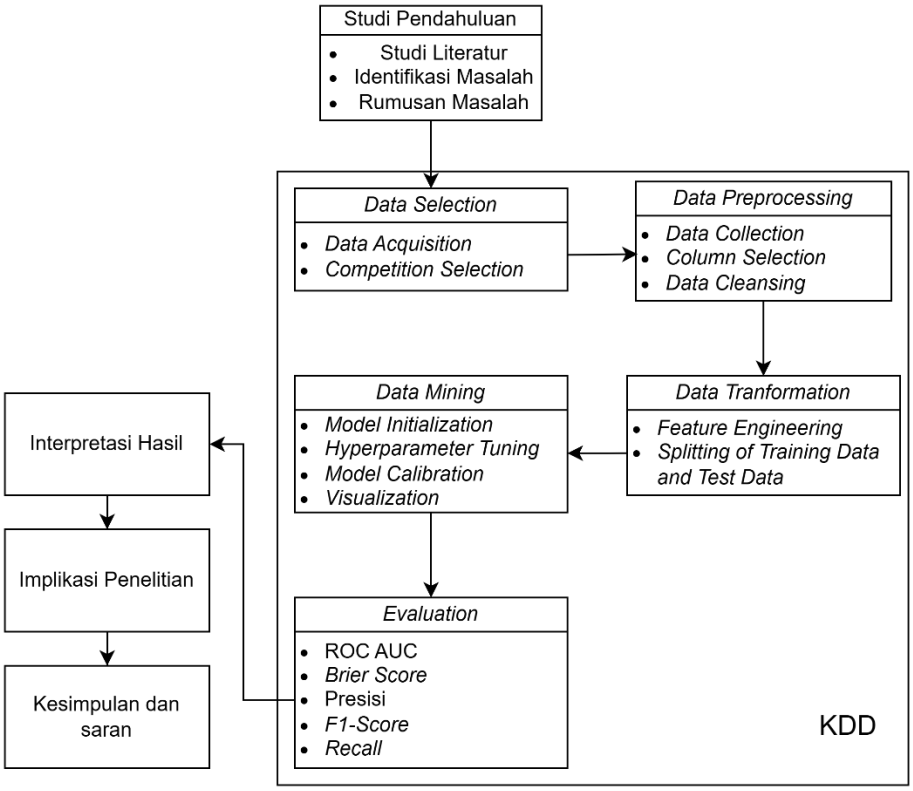
Pada tahap *transformation*, dilakukan pembagian data menjadi data latih dan data uji, serta dilakukan transformasi fitur agar sesuai dengan kebutuhan algoritma yang digunakan. Tahap data *mining* dilakukan dengan membangun model prediksi menggunakan algoritma LightGBM, serta melakukan *hyperparameter tuning* menggunakan *RandomizedSearchCV* untuk memperoleh kombinasi parameter terbaik berdasarkan nilai skor ROC AUC.

Kemudian, pada tahap *evaluation*, performa model diukur secara komprehensif menggunakan serangkaian metrik. Metrik-metrik tersebut meliputi ROC AUC, *Brier Score*, presisi, *recall*, *F1-Score*, dan *Log-Loss* untuk menilai performa model dari berbagai aspek, mulai dari kemampuan diskriminatif hingga akurasi probabilistik. Hasil dari seluruh tahapan analisis ini serta interpretasi terhadap performa model akan dijelaskan secara rinci pada Bab 4.

3.7 Tahapan Penelitian

Penelitian ini diawali dengan studi literatur mendalam terhadap sumber akademis untuk memahami kondisi terkini (*state-of-the-art*) dan merumuskan masalah, yang menjadi landasan bagi penerapan kerangka kerja *Knowledge Discovery in Database* (KDD). Proses KDD dimulai dengan tahap *data selection* yaitu pengumpulan *dataset* publik dari GitHub yang berisi catatan peristiwa pertandingan sepak bola, yang dilanjutkan dengan *preprocessing* komprehensif meliputi pemilihan variabel, pembersihan data dari nilai hilang dan duplikat,

encoding fitur kategorial, serta normalisasi data numerik. Selanjutnya, pada tahap *transformation*, dilakukan rekayasa fitur (*feature engineering*), analisis korelasi, seleksi fitur penting, dan pembagian data menjadi set pelatihan dan pengujian. Tahap inti *data mining* berfokus pada penggunaan algoritma LightGBM yang dioptimalkan melalui proses *tuning hyperparameter* dengan *RandomizedSearchCV* dan diperkuat oleh kalibrasi model untuk memastikan akurasi prediksi probabilistik. Pada tahap akhir, *evaluation* dilakukan secara komprehensif menggunakan serangkaian metrik mencakup ROC AUC, *Brier Score*, presisi, *recall*, *F1-Score*, dan *Log-Loss* untuk menilai kinerja model dari aspek diskriminatif hingga akurasi probabilistik, sebelum penelitian ditutup dengan interpretasi hasil, analisis implikasi, penarikan kesimpulan, dan saran untuk riset selanjutnya. Tahapan penelitian yang dilakukan dapat dilihat pada Gambar 3.7.



Gambar 3.7 Tahapan Penelitian

3.8 Jadwal Penelitian

Rencana waktu pelaksanaan penelitian ditunjukkan pada Tabel 3.2.

Tabel 3.2 Waktu Pelaksanaan Penelitian

No.	Tahapan	Maret 2025	April 2025	Mei 2025	Juni 2025	Juli 2025	Agustus 2025
1	Studi Pendahuluan & Landasan Teori						
2	Pengumpulan & Seleksi Data						
3	Pra-pemrosesan & Transformasi Data						
4	Pemodelan, Tuning, & Kalibrasi Model						
5	Evaluasi Model & Interpretasi Hasil						
6	Penyusunan Laporan & Kesimpulan						