BAB 2

TINJAUAN PUSTAKA

2.1 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 dari 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).

2.2 Analisis Sepak Bola

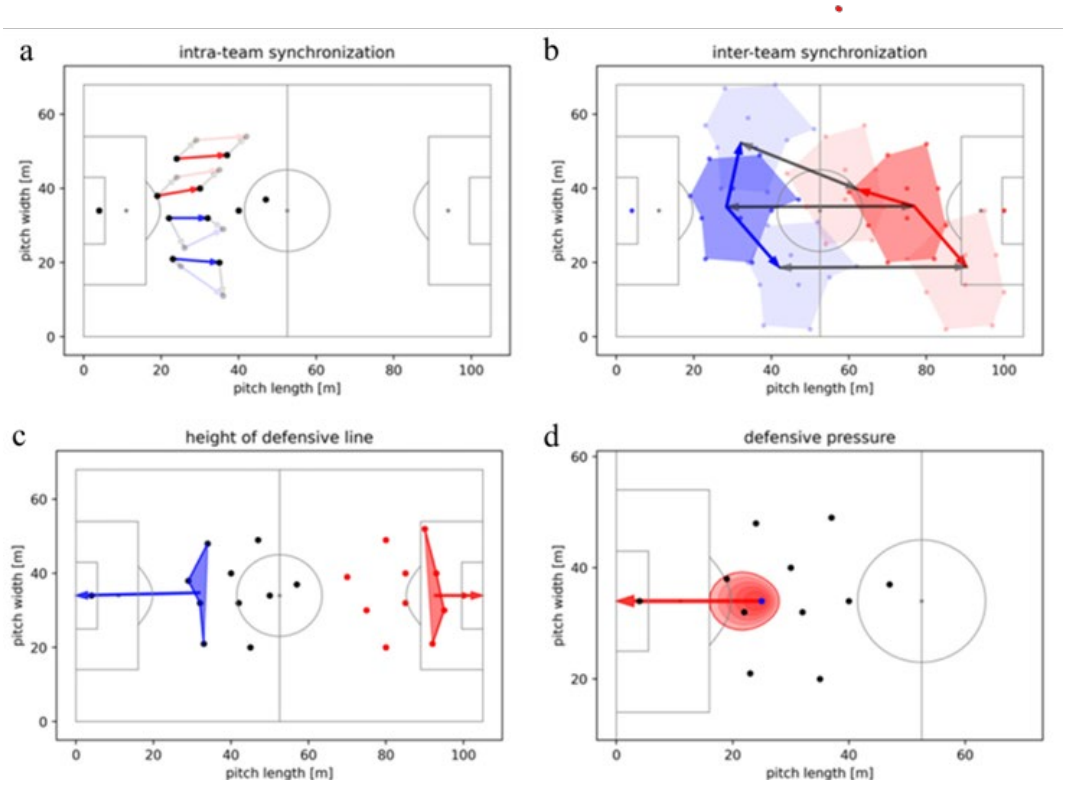
Analisis sepak bola merupakan proses yang kompleks dan melibatkan berbagai aspek dari permainan yang saling terkait. Secara mendasar, analisis ini mencakup pengukuran komunikasi antar pemain, kemampuan adaptasi, tempo permainan, serta evaluasi taktik penyerangan dan pertahanan (Mclean *et al*., 2017).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

Gambar 2.1 menunjukkan contoh visualisasi hasil analisis menggunakan *data tracking*, yang menggambarkan indikator-indikator kinerja utama dalam permainan bertahan.



Gambar 2.1 Contoh Visualisasi pada Analisis Sepak Bola (Forcher *et al*., 2022)

Pada Gambar 2.1 bagian (a) merupakan visualisasi yang menunjukkan tingkat sinkronisasi intra-tim di mana perilaku gerakan yang sinkron digambarkan pada warna merah dan perilaku yang asinkron pada warna biru. Bagian (b) menggambarkan sinkronisasi gerakan antar tim melalui pusat massa (*centroid*) dari tim-tim yang berlawanan. Pada bagian (c), visualisasi ini menunjukkan tinggi garis pertahanan (biru) yang digunakan sebagai pengukur posisi bertahan. Terakhir, bagian (d) menunjukkan tekanan bertahan yang dilakukan oleh dua pemain bertahan (hitam) terhadap pemain penyerang (biru), dengan arah ancaman ke gawang yang ditunjukkan oleh panah merah. Visualisasi ini memberikan gambaran yang jelas tentang dinamika taktik bertahan dalam sepak bola (Forcher *et al*., 2022).

2.3 *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. Model ini memiliki potensi besar untuk diintegrasikan sebagai salah satu komponen inti dalam sebuah Sistem Pendukung Keputusan (SPK) modern. Dengan menyediakan *output* yang terukur dan objektif untuk menilai kualitas peluang, SPK yang memanfaatkan xG dapat mengurangi bias subjektif dalam pengambilan keputusan strategis klub, terutama pada area vital seperti analisis performa ofensif, evaluasi pemain, dan optimalisasi taktik.

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 2.2, yang merupakan Visualisasi xG pada pertandingan langsung, memperlihatkan bagaimana setiap faktor dihitung untuk menghasilkan prediksi xG yang mendalam dan akurat.

Gambar 2.2 Visualisasi xG pada Pertandingan Langsung (Statsbomb, 2024)

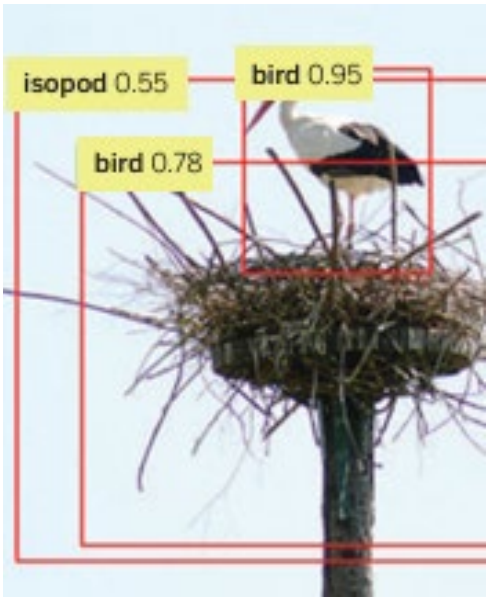
2.4 *Machine learning*

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.

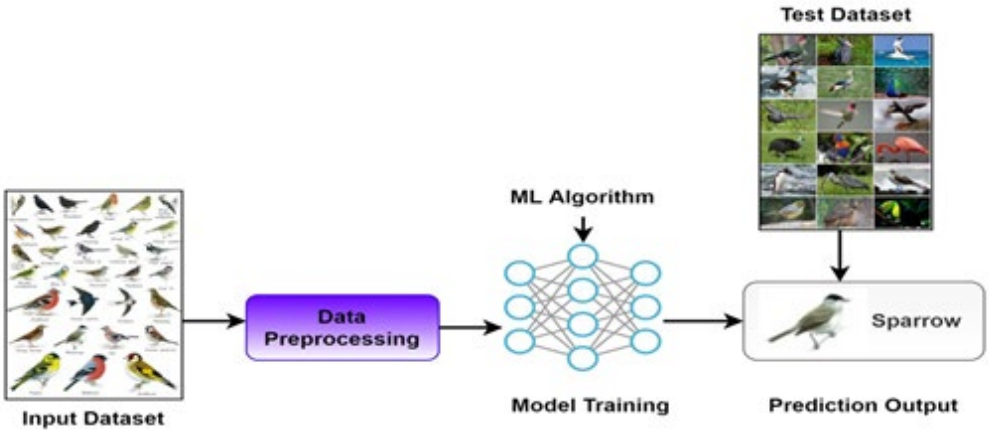
ML memiliki penerapan yang luas, salah satunya adalah *visual recognition*, yang memungkinkan pengenalan objek atau wajah dalam gambar secara otomatis. Dalam Gambar 2.3, ditampilkan implementasi ML pada aplikasi *visual recognition*, di mana teknologi ini mengenali dan mengklasifikasikan objek secara *real-time* berdasarkan data visual. Penerapan ini memanfaatkan kemampuan ML untuk belajar dari data visual guna membangun model yang mampu mendeteksi pola, bentuk, atau warna tertentu, serta mengenali objek-objek spesifik secara akurat dan efisien.



Gambar 2.3 Contoh Implementasi ML (Jordan & Mitchell, 2015)

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). Dalam Gambar 2.4, 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 2.4 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 2.1 menunjukkan bagaimana setiap model baru (x) ditambahkan secara bertahap dengan bobot pada iterasi ke-m, yang bertujuan untuk mengurangi kesalahan prediksi dari model sebelumnya.

$$F_m(\mathrm{x}) = F_{m-1}(\mathrm{x}) + \rho_m h_m(\mathrm{x}) \tag{2.1}$$

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 2.2.

$$F_0(\mathrm{x}) = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \alpha) \tag{2.2}$$

Persamaan 2.2 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 2.3.

$$\left(\rho_m, h_m(\mathrm{x})\right) = \underset{\rho,h}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_m - 1(\mathrm{x}_i) + \rho h(\mathrm{x}_i)\right) \tag{2.3}$$

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 2.4.

$$\mathrm{r}_{mi} = \left[\frac{\partial L\left(Y_i, F(\mathrm{x})\right)}{\partial L(\mathrm{x})}\right]_{F(\mathrm{x})=F_{m-1(\mathrm{x})}} \tag{2.4}$$

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, Thompson, Gu, & Binder, 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.

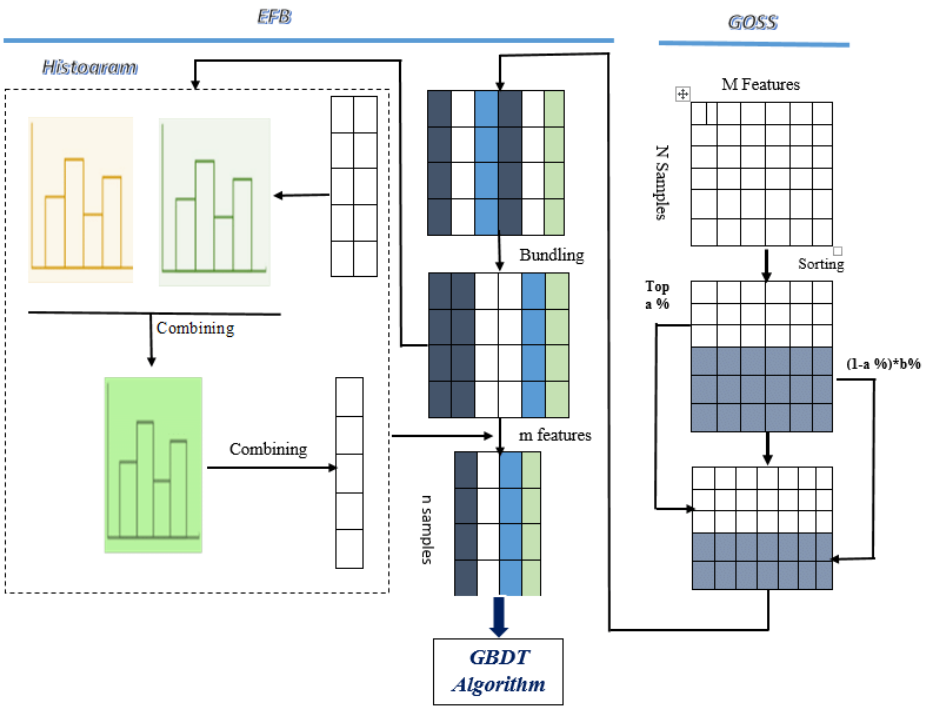
Sebagai algoritma *ensemble learning* yang semakin berkembang, telah terbukti unggul dalam meningkatkan prediksi dibandingkan dengan model lain, seperti *artificial neural network*, terutama dalam konteks pemodelan dinamis *bioprocess* (Mowbray *et al*., 2020). Dalam penerapan ini, *gradient boosting* 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 adalah kerangka kerja yang dirancang untuk mengimplementasikan algoritma *Gradient Boosting Decision Tree* (GBDT). 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. *Framework* ini 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 2.5 adalah arsitektur peningkatan algoritma GBDT dengan EFB dan GOSS.



Gambar 2.5 Arsitektur GOSS dan EFB

Pada Gambar 2.5 disajikan mengilustrasikan 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

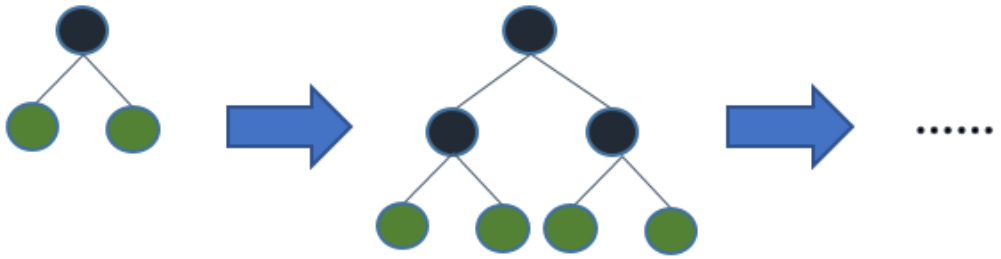
dengan gradien besar (*top* α%) dipertahankan secara utuh karena mereka berkontribusi paling signifikan terhadap *error* model, sedangkan sampel dengan gradien kecil diambil secara acak pada laju (1−α%)×*b*%. Pendekatan ini memungkinkan algoritma GBDT untuk fokus pada sampel yang paling informatif, mempercepat 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 menunjukkan kegunaan yang sangat luas dalam berbagai bidang dan masalah. Dalam masalah penugasan tugas *multi*-UAV (*Unmanned Aerial Vehicle*), model LightGBM memberikan solusi yang lebih baik dan cakupan solusi yang lebih luas dibandingkan algoritma lainnya. Hal ini menunjukkan kemampuannya dalam menangani masalah kompleks yang melibatkan banyak variabel dan pengambilan keputusan secara bersamaan (Wang & Zhang, 2023).

Dalam pemuliaan tanaman yang memanfaatkan data genom, LightGBM terbukti menghasilkan prediksi yang lebih akurat, model yang lebih stabil, dan proses komputasi yang lebih cepat, misalnya pada data sebanyak 50.000 sampel dan 10.000 SNP (*Single Nucleotide Polymorphism*) LightGBM hanya memerlukan delapan menit pelatihan dan 20 GB memori, sementara rrBLUP (*ridge regression Best Linear Unbiased Prediction*) memakan waktu lebih dari tujuh belas jam pelatihan dan memerlukan 116 GB memori, sehingga mempercepat proses seleksi sifat unggul berbasis genomik (Yan *et al*., 2021).

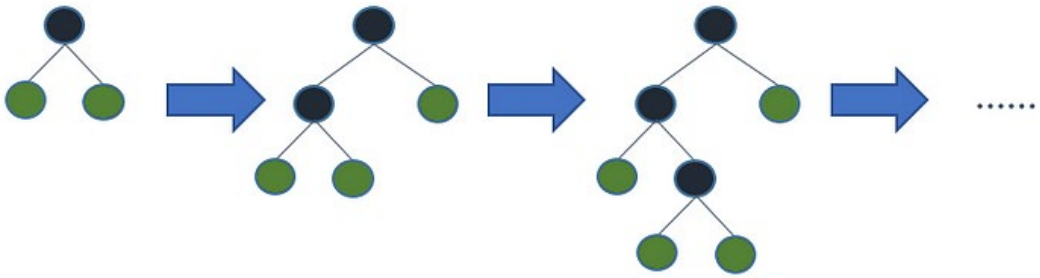
Dalam prediksi beban termal bangunan, LightGBM terbukti lebih unggul dibandingkan dengan algoritma *Random Forest* (RF) dan *Long Short-Term Memory* (LSTM) dalam hal akurasi prediksi dan efisiensi komputasi. Sebagai contoh, dalam studi oleh Chen *et al*. (2023), LightGBM mencapai nilai koefisien variasi dari *root mean squared error* (CVRMSE) sebesar 5,25 persen dan koefisien determinasi (R^2) sebesar 0,9959, dengan waktu komputasi hanya 7 detik. Sebaliknya, RF memiliki CVRMSE sebesar 18,54 persen, R^2 sebesar 0,9482, dan waktu komputasi 44,6 detik, sedangkan LSTM menunjukkan CVRMSE sebesar 22,06 persen, R^2 sebesar 0,9267, dan waktu komputasi 758,8 detik. Hasil ini menunjukkan bahwa LightGBM tidak hanya memberikan prediksi yang lebih akurat tetapi juga memerlukan waktu komputasi yang jauh lebih singkat, menjadikannya pilihan yang sangat efisien untuk aplikasi di bidang konstruksi dan manajemen energi bangunan.

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 2.6 (LightGBM, 2024).



Gambar 2.6 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 2.7. 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 2.7 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.

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.

Dalam pengembangan model LightGBM, kemampuan interpretasi dan keterbukaan model merupakan aspek penting, terutama untuk memahami alasan di balik prediksi yang dihasilkan. Teknik interpretasi seperti *Permutation Feature Importance* (PFI) dan *Shapley additive explanations* (SHAP) menjadi metode yang sangat berguna untuk menjelaskan kontribusi setiap fitur dalam model terhadap prediksi akhir (Chaibi *et al*., 2021). PFI, misalnya, menilai pentingnya setiap fitur dengan mengevaluasi dampak perubahan nilai fitur terhadap akurasi model, sementara SHAP memberikan nilai yang menunjukkan pengaruh masing-masing fitur pada setiap prediksi. Dengan menggunakan teknik ini, pengguna dapat lebih memahami dan meningkatkan model yang mereka bangun.

Nilai SHAP, khususnya, dapat digunakan pada model LightGBM untuk memastikan kemampuan interpretasi prediksi dengan tingkat keterbukaan yang lebih tinggi. Dengan mengaplikasikan nilai SHAP, model dapat meningkatkan performa inferensi serta mempercepat waktu pelatihan, terutama pada *dataset* yang kompleks. Selain itu, penggunaan SHAP dapat mengurangi kecenderungan model untuk "*fit-to-noise*" atau penyesuaian yang terlalu sensitif terhadap data acak, yang sering kali menjadi masalah dalam analisis data berukuran besar (Bugaj *et al*., 2021). Hal ini membuat SHAP menjadi alat interpretasi yang sangat efektif dalam membangun model LightGBM yang andal dan terbuka terhadap evaluasi.

Selain PFI dan SHAP, kemampuan interpretasi dan keterbukaan dalam LightGBM dapat ditingkatkan melalui metode pembelajaran yang lebih adaptif seperti *personalized interpretability estimation* (ML-PIE). Dengan pendekatan ini, pengguna dapat mengarahkan proses sintesis model berdasarkan preferensi kemampuan interpretasi yang di personalisasi, melalui algoritma evolusi *bi-objektif* yang mempertimbangkan kemampuan interpretasi bersama dengan akurasi. Metode ML-PIE ini memungkinkan pengguna untuk menentukan prioritas interpretasi dalam pengembangan model, sehingga menghasilkan model LightGBM yang tidak hanya efisien tetapi juga mudah diinterpretasi, sesuai dengan kebutuhan spesifik dari pengguna atau lingkungan aplikasinya (Virgolin *et al*., 2021).

2.7 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 2.5.

$$Brier\ Score\ = (f_t - \ o_t)^2 \tag{2.5}$$

Brier Score digunakan untuk menghitung selisih kuadrat antara nilai prediksi dan nilai aktual, sebagaimana terlihat pada Persamaan 2.5. 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.

2.8 *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 2.6.

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR}), d(\mathrm{FPR}) \tag{2.6}$$

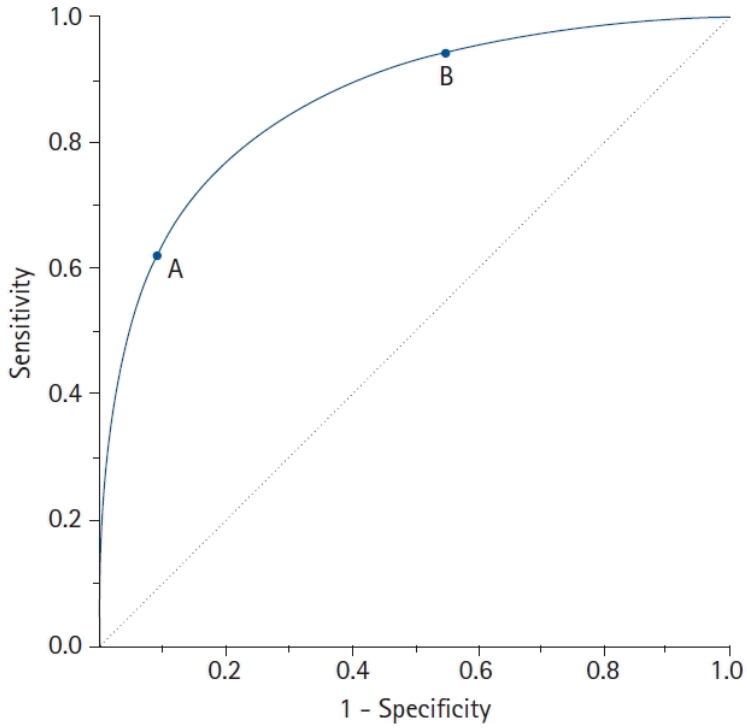
Dalam praktiknya, karena kurva ROC terdiri dari sejumlah titik diskrit, AUC sering dihitung menggunakan aturan trapesium. Parameter TPR dan FPR sendiri dihitung berdasarkan nilai dari *confusion matrix* dengan persamaan 2.7 dan 2.8.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.7}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.8}$$

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 2.8, 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 2.8 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-ROC sangat bermanfaat dalam seleksi model yang lebih baik dan pemahaman tentang ruang yang dapat dioptimalkan untuk meningkatkan kinerja klasifikasi (Tafvizi *et al*., 2022).

2.9 Presisi (*Precision*)

Dalam evaluasi model klasifikasi, metrik Presisi memberikan wawasan penting tentang kualitas prediksi positif yang dihasilkan oleh model. Untuk memahaminya, diperlukan pemahaman mengenai *confusion matrix*.

Confusion matrix adalah sebuah tabel yang merangkum performa model dengan membandingkan kelas aktual dengan kelas yang diprediksi. Tabel ini terbagi menjadi empat komponen utama, seperti ditunjukkan pada 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)

Presisi mengukur proporsi dari prediksi positif yang benar-benar merupakan kasus positif (Tharwat, 2021). Metrik ini secara esensial mengevaluasi seberapa andal prediksi positif yang dibuat oleh model. Dengan demikian, presisi menjadi

metrik yang sangat penting dalam situasi di mana biaya dari False Positive tinggi. Sebagai contoh, dalam sistem penyaringan email, kesalahan mengklasifikasikan email penting sebagai spam (False Positive) dapat menyebabkan pengguna kehilangan informasi krusial. Nilai presisi dihitung menggunakan persamaan 2.9 (Powers, 2011).

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

2.10 *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 (Powers, 2011). Esensi dari metrik ini adalah untuk mengevaluasi tingkat kelengkapan (*completeness*) dari prediksi positif yang dihasilkan.

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)

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

2.11 *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).

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

2.12 *Log-Loss* (*Logarithmic Loss*)

Log-Loss, atau dikenal juga sebagai *cross-entropy loss*, adalah metrik evaluasi yang mengukur performa model klasifikasi di mana *output* prediksinya adalah nilai probabilitas antara 0 dan 1. Berbeda dengan metrik yang hanya menilai benar atau salahnya sebuah klasifikasi, *Log-Loss* juga memperhitungkan tingkat "keyakinan" (*confidence*) dari prediksi tersebut.

Metrik ini memberikan penalti yang besar pada prediksi yang sangat yakin namun salah. Sebagai contoh, jika model memprediksi probabilitas 0.95 untuk

sebuah tembakan yang ternyata tidak menjadi gol, penalti yang diberikan akan jauh lebih tinggi daripada jika model memprediksi probabilitas 0.55. Model yang sempurna akan memiliki nilai *Log-Loss* 0, sedangkan nilai yang lebih tinggi menunjukkan performa yang lebih buruk.

Untuk masalah klasifikasi biner, *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*, y_i adalah label aktual untuk sampel ke-*i* (dengan nilai 1 untuk kelas positif dan 0 untuk kelas negatif), dan p_i adalah probabilitas yang diprediksi oleh model bahwa sampel ke-*i* termasuk dalam kelas positif. Oleh karena kemampuannya dalam mengevaluasi akurasi probabilistik, *Log-Loss* menjadi metrik standar untuk mengevaluasi dan membandingkan model klasifikasi probabilistik.

2.13 *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.

Proses KDD terdiri dari beberapa tahapan yang saling berinteraksi secara iteratif. Secara umum, tahapan tersebut mencakup (Chaudhary & Kishore, 2017):

a. Pembersihan data (penghapusan kebisingan dan data yang tidak konsisten),
b. Integrasi data (penggabungan beberapa sumber data),
c. Pemilihan data (pengambilan data yang relevan untuk tugas analisis),
d. Transformasi data (pengolahan atau konsolidasi data untuk memudahkan *mining*),
e. Data *mining* (aplikasi metode cerdas untuk mengekstraksi pola),
f. Evaluasi pola (penilaian pola yang menarik berdasarkan ukuran keterminatan), dan
g. Presentasi pengetahuan (penggunaan teknik visualisasi untuk menyajikan pengetahuan yang ditemukan kepada pengguna).

Data *mining* adalah bagian penting dalam proses KDD yang melibatkan berbagai alat dan teknik untuk menemukan pola yang berguna dalam basis data yang sangat besar (Chaudhary & Kishore, 2017). Salah satu bentuk terbaru dari data

mining adalah ringkasan linguistik, yang bertujuan untuk memberikan deskripsi verbal yang dihasilkan oleh komputer mengenai pengetahuan yang tersembunyi dalam *database*, sering kali dalam bentuk aturan '*if-then*' yang menyerupai granula pengetahuan *fuzzy*. Selain itu, teknik *text mining* digunakan untuk mengekstraksi pola dari dokumen teks, yang melibatkan analisis teks untuk mengubah dokumen yang tidak terstruktur menjadi sekumpulan fitur yang sesuai, lalu menerapkan teknik data *mining* untuk ekstraksi pola (Chaudhary & Kishore, 2017).

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.

2.14 *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.

2.15 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*.

2.16 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*.

2.17 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 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.

2.19 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.

2.20 Penelitian Sejenis

Penelitian sejenis yang digunakan pada penelitian ini ditunjukkan pada Tabel 2.1.

Tabel 2.2 Penelitian Sejenis

No	**Sumber**	**Judul dan Penulis**	**Metode dan *Tools***	**Data Set dan Hasil**	**Kelebihan dan Kekurangan**
1	*Asian Conference on Intelligent Information and Database Systems* (2023)	*Improving the Expected Goal Value in Football Using Multilayer Perceptron Networks* (Méndez, Montero, & Núñez, 2023)	*Multilayer Perceptron Neural Network* dan Python, Keras	Sumber: StatsBomb *event* data (> 12 000 tembakan dari berbagai liga top) Input: lokasi tembakan, situasi permainan, tipe *assist* Hasil: MLP ungguli regresi logistik dengan	Kelebihan: Menangkap pola non-linear kompleks Kekurangan: Membutuhkan *tuning* parameter teliti; rawan *overfitting* pada *dataset*

				peningkatan akurasi hingga 6%; ROC AUC = 0,87	kecil; susah diinterpretasi langsung
2	SSAC 8th *Annual* MIT *Sloan Sports Analytics Conference* (2015)	*Quality vs Quantity: Improved Shot Prediction in Soccer Using Strategic Features from Spatiotemporal Data* (Lucey, Bialkowski, Monfort, Carr & Matthews, 2015)	*Conditional Random Fields, Custom code*	Sumber: Prozone/Stats Perform data (~ 9 732 tembakan + 10 s video pra-tembakan)Fitur: fase permainan, kedekatan & interaksi pemain, kecepatan Hasil: EGV lebih akurat daripada model lokasi-tunggal; peningkatan ROC-AUC vs *baseline*	Kelebihan: Menangkap konteks temporal & strategi tim Kekurangan: Asumsi probabilistik simplistik, kompleksitas inferensi tinggi; skalabilitas terbatas pada *dataset* besar
3	*Journal of Sports Analytics* 4(3) (2018)	*Spatial Analysis of Shots in MLS: A Model for Expected Goals and Fractal Dimensionality* (Fairchild, Pelechrinis & Kokkodis, 2018)	Regresi Logistik, Python, SciPy/Statsmodels	Sumber: 1 115 tembakan *non-penalty* dari 99 pertandingan MLS 2016 (koordinat x,y ditag manual) Hasil: kalibrasi kuat (*leave-one-out*); ROC AUC = 0,80	Kelebihan: Sederhana, mudah diinterpretasi, hemat komputasi, Kekurangan: Linieritas membatasi interaksi non-linear & konteks spasial-temporal
4	StatsBomb *Conference*	*"Estimated Player*	*Generalised Linear Mixed*	Sumber: StatsBomb *event* data (580	Kelebihan: Menangkap

	Proceedings (2022)	*Impact" (EPI): Quantifying the Effects of Individual Players on Football Actions Using Hierarchical Statistical Models* (Tureen & Olthof, 2022)	*Models* (GLMM), R (lme4)	*Premier League* + 326 *Women's Super League*) *Output*: skor EPI individual; *random player effects* jelaskan variabilitas *outcome*	struktur hierarkis antar pertandingan & pemain, Kekurangan: Asumsi linearitas & distribusi *residual* membatasi non-linear; komputasi skala besar berat; interpretasi *random effect* sulit
5	*Journal of the Operational Research Society* 76(1) (2022)	*Explainable Expected Goals* (Cavus & Biecek, 2022)	XGBoost, *Random Forest*, LightGBM, CatBoost via Forester AutoML, SHAP untuk interpretasi	Sumber: Understat *event* data (315 430 tembakan dari 7 musim Top 5 Eropa) Hasil: *Random Forest* terbaik pada ROC AUC (*cross-validation*)	Kelebihan: AutoML memudahkan eksplorasi model; SHAP tingkatkan transparansi, Kekurangan: Risiko *overfitting* pada *forest* besar; waktu komputasi tinggi; SHAP masih terbatas dalam

					interpretasi interaksi fitur
6	IEEE MLDM *for Sports Analytics Workshop* (2016)	*Expected Goals in Soccer: Explaining Match Results Using Predictive Analytics* (Eggels, van Elk & Pechenizkiy, 2016)	*Logistic Regression*, *Decision Tree*, *Random Forest*, AdaBoost, Python (scikit-learn)	Sumber: ORTEC & Inmotio *event*/*tracking* + EA Sports *attributes* (~20 000 tembakan, 2 musim) Hasil: ROC AUC: AdaBoost ~0,84, *Random Forest* ~0,82, *Logistic* ~0,78, Tree ~0,74	Kelebihan: *Ensemble methods* menangkap interaksi fitur kompleks, Kekurangan: Model kompleks rawan *overfitting*, mahal komputasi, dan sulit diinterpretasi
7	PLoS *ONE* 18(4): e0282295	*Expected Goals in Football: Improving Model Performance and Demonstrating Value* (Mead, O'Hare &	*Logistic Regression*, *Random Forest*, AdaBoost, XGBoost, Python (scikit-learn), *cross-validation*	Sumber: Wyscout (*Soccer_match_event_dataset,* ~250 000 tembakan), Fitur: lokasi, tipe tembakan, nilai pemain, ELO *rating*, penonton, Hasil: ROC AUC tertinggi: *Random*	Kelebihan: *Ensemble* model tangkap interaksi kompleks dan tingkatkan akurasi, Kekurangan: *Overfitting*,

		McMenemy, 2023)		*Forest* = 0,91; LogReg = 0,85	*tuning* intensif, regresi logistik terlalu sederhana
8	Franklin *Open* 4 (2023)	*A Machine learning Approach for Player and Position Adjusted Expected Goals in Football* (Hewitt & Karakuş, 2023)	*Logistic Regression*, *Gradient Boosted Trees*, scikit-learn, SHAP	Sumber: StatsBomb *event* data (~15 574 tembakan, 1 887 gol) Hasil: Korelasi: LogReg = 0.887 (xG = 1 866), GBT = 0.902 (xG = 1 870); Uji pada Messi: *uplift* +347 xG	Kelebihan: GBT menangkap dinamika non-linear, SHAP bantu interpretasi fitur, Kekurangan: *Overfitting* dan interpretasi sulit pada GBT, regresi logistik terlalu terbatas
9	Universitat Politècnica de Catalunya (2020)	*Creating a Model for Expected Goals in*	*Logistic Regression*, XGBoost, *Artificial*	Sumber: OPTA (~20 000 tembakan, 5 liga top Eropa) + atribut	Kelebihan: Model non-linear tangkap

		Football Using Qualitative Player Information (P. M. Pardo, 2020)	*Neural Network* (ANN), scikit-learn, Keras	pemain (740 pemain, FIFA) Hasil: ROC AUC LogReg = 0,78; XGBoost = 0,85; ANN = 0,88. RMSE: 0,32; 0,27; 0,25	interaksi atribut kualitatif, Kekurangan: *Overfitting* pada XGBoost; ANN butuh *tuning* intensif & data besar; LogReg terlalu sederhana
10	PLoS *ONE* 19(10): e0312278	*Predicting Goal Probabilities with Improved* xG *Models Using Event Sequences* (Bandara *et al*., 2024)	*Random Forest* (100 *estimators*), scikit-learn	Sumber: StatsBomb Open Data; fitur: urutan 3 *event* (termasuk "*advancement factor*", posisi pemain) Hasil: ROC AUC validasi = 0,833, ROC AUC uji Euro 2020 = 0,826, lebih tinggi dari model *shot*-tunggal	Kelebihan: Fitur temporal meningkatkan akurasi prediksi xG, Kekurangan: Bergantung pada *random under-sampling*; RF rawan *overfitting* &

					interpretasi sulit
11	The Statistician (1997)	*Measuring the effectiveness of playing strategies at soccer* (Pollard & Reep)	*Logistic Regression*	22 pertandingan, 489 tembakan Hasil: Menemukan bahwa peluang yang diambil lebih dekat dan lebih sentral ke gawang memiliki probabilitas gol yang lebih tinggi.	Kelebihan: Merupakan salah satu studi perintis dalam kuantifikasi xG Kekurangan: Keterbatasan pada variabel dasar (lokasi, sudut), pengumpulan data manual.
12	JORS 76(1)	*A New xG Model for Football Analytics* (Cefis & Carpita, 2024)	*Logistic Regression*, R (glm)	Understat (~50 000 shots), SoFIFA, Math&Sport, ROC AUC = 0,81 dengan fitur: tekanan, *rating* pemain, kualitas lawan	Kelebihan: Sederhana, mudah diinterpretasi, Kekurangan: Sulit tangkap interaksi non-linear dan konteks kompleks

13	arXiv:2101.02104	*A Probabilistic Model for Predicting Shot Success in Football* (Wheatcroft & Sienkiewicz, 2021)	*Parametric probabilistic model,* Python (SciPy optimize)	>1 juta *shot* dari 22 liga (football-data.co.uk) Perbaikan *log-score* ~0.03 vs *equal-probability baseline*	Kelebihan: Sederhana, cocok untuk *pipeline* prediksi, Kekurangan: Abaikan lokasi/situasi *shot*, risiko propagasi *error*
14	*World congress on science and football (2005*	*Applications of Logistic Regression to Shots at Goal in Association Football (Ensum, Pollard, & Taylor)*	*Logistic Regression*	Sumber: 48 pertandingan, 1.099 tembakan Hasil: Mengonfirmasi dan memperluas temuan awal dengan *dataset* yang lebih besar, mengkuantifikasi berbagai faktor tembakan.	Kelebihan: Peningkatan ukuran *dataset* dari studi sebelumnya. Kekurangan: Masih menggunakan model linier yang membatasi analisis interaksi non-linear.

15	*ESANN 2015 proceedings* (2015)	*Measuring scoring efficiency through goal expectancy estimation* (Ruiz *et al*.)	*Multilayer Perceptron* (MLP	Sumber: EPL 2013/14, Prozone, 10.318 tembakan Hasil: Mengidentifikasi pemain yang lebih efisien dalam mencetak gol dari peluang yang ada.	Kelebihan: Menggunakan model non-linier (MLP) yang dapat menangkap pola kompleks. Kekurangan: Sulit diinterpretasikan secara langsung (bersifat *black box*).

Berdasarkan Tabel 2.1, terdapat lima belas penelitian yang mengkaji perhitungan metrik xG dalam analisis sepak bola, yang dapat dikelompokkan berdasarkan kompleksitas metodologinya. Kelompok pertama mencakup penelitian yang mengandalkan model statistik yang lebih sederhana dan mudah diinterpretasi. Studi perintis oleh Pollard & Reep (1997) menerapkan Regresi Logistik untuk mengukur efektivitas strategi permainan. Pendekatan ini dilanjutkan oleh Ensum, Pollard, & Taylor (2005) dengan *dataset* yang lebih besar. Penelitian lain seperti oleh Fairchild, Pelechrinis & Kokkodis (2018) menggunakan Regresi Logistik untuk analisis spasial tembakan di MLS, dan Cefis & Carpita (2024) yang turut memakai Regresi Logistik dengan tambahan fitur *rating* pemain dan kualitas lawan.

Terdapat pula pendekatan yang menggunakan model probabilistik parametrik seperti yang dilakukan oleh Wheatcroft & Sienkiewicz (2021).

Sebagai pendekatan alternatif dari model statistik sederhana, banyak penelitian yang memanfaatkan *machine learning* dan *deep learning* untuk menangkap pola non-linear yang lebih kompleks dalam data tembakan. Beberapa dari penelitian ini berfokus pada perbandingan komprehensif berbagai algoritma untuk menemukan model yang paling optimal. Contohnya, Eggels, van Elk & Pechenizkiy (2016) menguji *Random Forest* dan AdaBoost, sementara Cavus & Biecek (2022) menggunakan kerangka kerja AutoML untuk mengeksplorasi XGBoost dan *Random Forest*. Serupa dengan itu, Mead, O'Hare & McMenemy (2023) juga melakukan perbandingan dan menemukan bahwa *Random Forest* merupakan model dengan ROC AUC tertinggi.

Model jaringan saraf juga menjadi pilihan populer karena kemampuannya memodelkan hubungan yang rumit. *Multilayer Perceptron* (MLP) digunakan secara efektif oleh Méndez, Montero, & Núñez (2023) dan Ruiz *et al*. P. M. Pardo (2020) turut membandingkan XGBoost dengan *Artificial Neural Network* (ANN) untuk menganalisis pengaruh informasi kualitatif pemain terhadap kualitas peluang.

Selain itu, terdapat penelitian yang mengembangkan model dengan memanfaatkan fitur-fitur yang lebih canggih. Lucey *et al*. menerapkan *Conditional Random Fields* untuk memasukkan konteks temporal dan strategi tim sebelum tembakan terjadi, sedangkan Bandara *et al*. menggunakan *Random Forest* dengan fitur sekuensial dari beberapa kejadian terakhir. Untuk mengakomodasi struktur data yang hierarkis, Tureen & Olthof (2022) menerapkan *Generalised Linear Mixed*

Models (GLMM) yang dapat memodelkan efek acak dari pemain dan pertandingan. Seiring meningkatnya kompleksitas model, muncul pula fokus pada interpretabilitas, seperti yang ditunjukkan oleh Hewitt & Karakuş (2023) yang melengkapi model *Gradient Boosted Trees* mereka dengan analisis SHAP untuk menjelaskan kontribusi setiap fitur.

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

1. Fokus pada penggunaan LightGBM sebagai metode utama untuk perhitungan xG, yang dikenal karena efisiensi dan kecepatan komputasinya.
2. 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.

BAB 3

METODOLOGI PENELITIAN

3.1 Pendekatan Penelitian

Penelitian ini melihat kinerja metode LGBM 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 Tempat dan Waktu Penelitian

3.2.1 Tempat Penelitian

Penelitian ini dilakukan menggunakan *open-data* dari StatsBomb melalui repositori GitHub dengan proses analisis yang dilakukan menggunakan Python dan platform Google Colaboratory.

3.2.2 Waktu Penelitian

Rencana waktu pelaksanaan penelitian ditunjukkan pada Tabel 3.1.

Tabel 3.1 Waktu Pelaksanaan Penelitian

No.	Tahapan	Februari 2025	Maret 2025	April 2025	Mei 2025	Juni 2025	Juli 2025
1	Landasan Teori						

2	Pengumpulan Data						
3	Analisis Data						
4	Interpretasi						
5	Pembuatan Laporan						

3.3 Metodologi Pengumpulan Data

3.3.1 Studi Literatur

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.

3.3.2 Pengambilan Data

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.

3.4 Pengembangan Model

3.4.1 Metode KDD

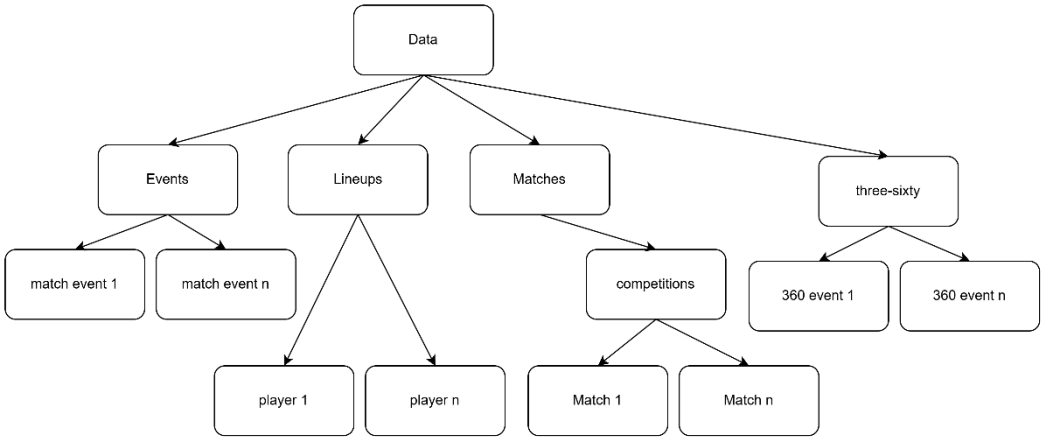
Penelitian ini menggunakan pendekatan *Knowledge Discovery in Databases* (KDD) dalam proses pengembangan model. Metode KDD memiliki keunggulan dalam membantu mengidentifikasi pola tersembunyi dari kumpulan data yang kompleks sehingga dapat menghasilkan informasi yang lebih mudah dipahami. Proses KDD 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.

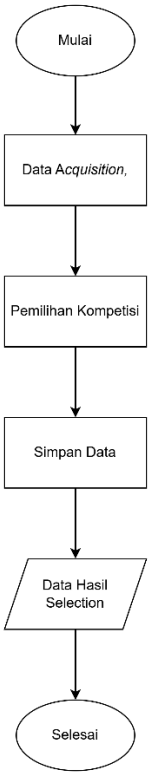
Untuk mengatasi masalah ini, saat ini hanya situasi gol dan kartu (kuning/merah) yang diikutkan sebagai *event* terkait dalam pemrosesan data StatsBomb. Dengan begitu, kita tetap menjaga konteks penting seperti *assist* sebelum gol atau pelanggaran yang berujung kartu tanpa menumpuk terlalu banyak duplikasi. 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-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.*



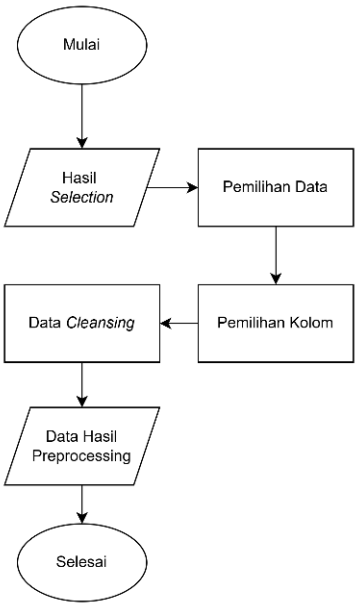
Gambar 3.2 *Flowchart Selection*

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.

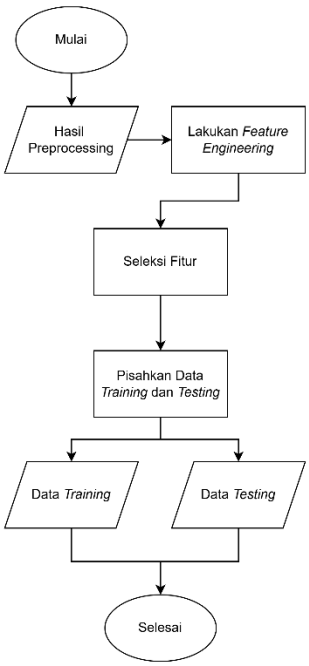
Langkah terakhir adalah data *cleansing*, yang meliputi pemeriksaan nilai kosong dan duplikat. Namun, karena data StatsBomb memiliki format yang sangat terstruktur dan tiap peristiwa dalam pertandingan bersifat unik, data yang diperoleh relatif bersih dan tidak memerlukan proses pembersihan lanjutan. 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 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.

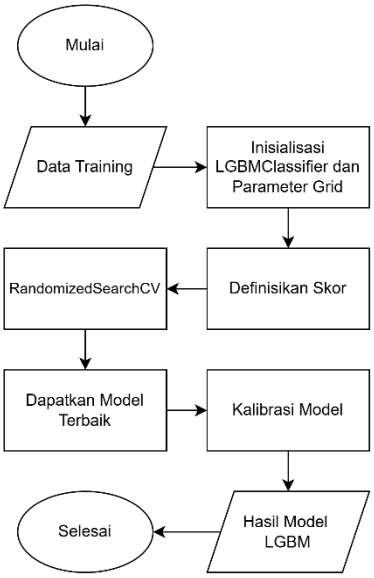


Gambar 3.4 *Flowchart Transformation*

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.



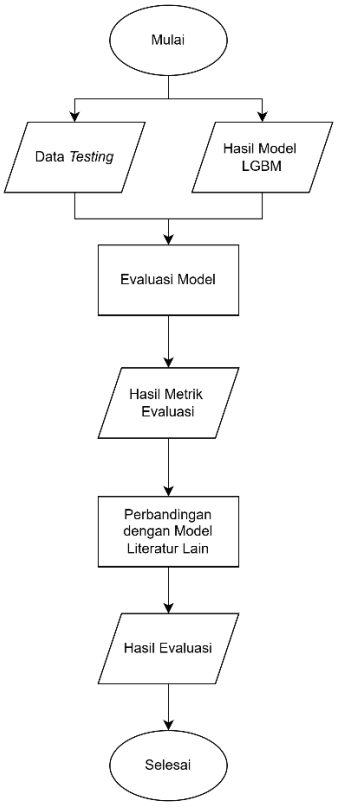
Gambar 3.5 *Flowchart* Data *Mining*

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*

3.4.2 Permodelan 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.

3.5 Analisis Data dan Intrepretasi Hasil

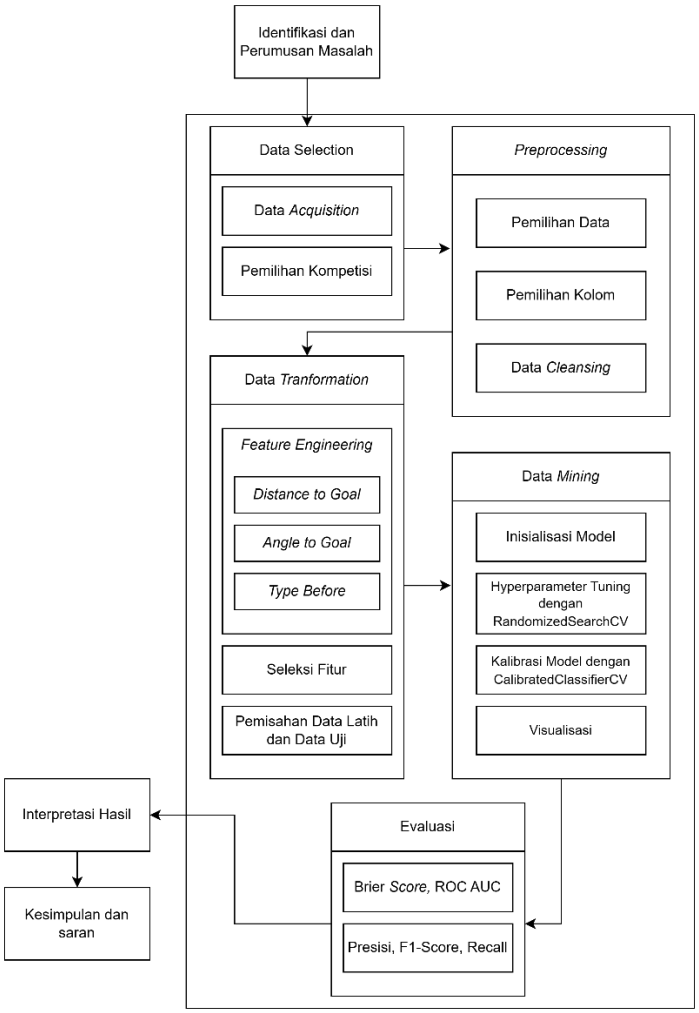
Analisis data dalam penelitian ini dilakukan berdasarkan pendekatan KDD (*Knowledge Discovery in Database*) yang mencakup lima tahapan utama: data *selection, preprocessing, transformation,* data *mining*, dan *evaluation*. Proses analisis dimulai dari tahap data *selection*, yaitu dengan menyiapkan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

3.6 Tahapan Penelitian

Tahapan penelitian yang dilakukan dapat dilihat pada Gambar 3.7.



Gambar 3.7 Tahapan Penelitian

Tahapan pertama dalam penelitian ini adalah identifikasi dan perumusan masalah yang dilakukan dengan mengkaji literatur yang relevan untuk memahami permasalahan aktual dan peluang riset di bidang analisis pertandingan sepak bola menggunakan teknik pembelajaran mesin. Literatur yang digunakan berasal dari

jurnal ilmiah, artikel konferensi, serta laporan penelitian terkini. Setelah permasalahan dirumuskan, penelitian dilanjutkan dengan mengikuti alur metode KDD yang terdiri dari tahapan data *selection, preprocessing, transfor*mation, data *mining*, dan *evaluation*.

Pada tahap data *selection*, data dikumpulkan dengan mengunduh *dataset* publik dari GitHub yang berisi catatan pertandingan sepak bola, lalu difokuskan pada data *event* yang relevan untuk keperluan prediksi. Selanjutnya, dilakukan *preprocessing* berupa pemilihan variabel yang akan digunakan, pembersihan data dari nilai yang hilang dan duplikat, proses *encoding* untuk fitur kategorial, serta normalisasi data numerik.

Tahap berikutnya adalah *transformation* yang mencakup rekayasa fitur (*feature engineering*), analisis korelasi antar variabel, seleksi fitur penting, dan pembagian data menjadi set pelatihan dan pengujian. Pada tahap data *mining*, digunakan algoritma LightGBM dengan proses *tuning hyperparameter* melalui *RandomizedSearchCV*, serta dilanjutkan dengan proses kalibrasi model agar hasil prediksi probabilistik menjadi lebih akurat.

Tahap terakhir adalah *evaluation*, yaitu evaluasi performa model secara komprehensif menggunakan serangkaian metrik yang mencakup ROC AUC, *Brier Score*, presisi, *recall*, F1-Score, dan *Log-Loss* untuk menilai berbagai aspek kinerja model mulai dari kemampuan diskriminatif hingga akurasi probabilistik. Setelah seluruh tahapan metode KDD selesai, penelitian diakhiri dengan analisis hasil dan penarikan kesimpulan serta saran untuk penelitian selanjutnya.

3.7 Perangkat Penelitian

Penelitian ini menggunakan perangkat keras (*hardware*) dan perangkat lunak (*software*) dengan spesifikasi yang dijelaskan pada Tabel 3.2.

Tabel 3.2 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