BAB I
PENDAHULUAN

1.1 Latar Belakang

Investasi adalah proses penempatan sejumlah modal oleh individu atau kelompok pada suatu aset dengan tujuan untuk mendapatkan keuntungan di masa depan. Menurut data *Single Investor Identification* (SID), lebih dari 600 ribu investor baru tercatat sejak tahun 2023 hingga 9 Agustus 2024. Peningkatan tersebut memperlihatkan bahwa investasi saham merupakan salah satu investasi dengan jumlah peminat yang terus meningkat. (Ningrum, 2023) representasikan makna saham sebagai instrumen keuangan yang bersifat kepemilikan atas perusahaan terbuka. Harga saham bersifat fluktuatif, yang berarti nilainya dapat berubah secara dinamis seiring waktu. Fluktuasi menawarkan peluang keuntungan yang lebih besar dibandingkan dengan jenis investasi lainnya, sebaliknya juga membawa risiko kerugian yang besar. Pergerakan harga saham yang tidak teratur mampu merugikan investor, maka muncul berbagai penelitian untuk memperkirakan pergerakan harga saham, sehingga risiko kerugian berkurang dan potensi keuntungan meningkat.

Pembelajaran mesin adalah salah satu metode penelitian yang terbukti berhasil digunakan dalam memprediksi pergerakan harga saham. (Purnama and Sofana, 2021) menjelaskan pembelajaran mesin sebagai bagian dari kecerdasan buatan (*Artificial Intelligence*) yang melatih sistem untuk mengembangkan sistem secara otomatis dengan memanfaatkan inputan data atau pengalaman masa lalu. Terdapat berbagai penelitian yang menerapkan pembelajaran mesin untuk mengembangkan prediksi harga saham seperti *Regresi*, *Random Forest*, *Boosting Algorithm*, *Deep Learning*, dan berbagai metode tunggal serta metode gabungan lainnya. Seluruh metode memiliki kelebihan dan kekurangan masing-masing. Namun penelitian (Yang et al., 2021) menemukan metode *Gradient Boosting* seperti LightGBM dan XGBoost memiliki tingkat akurasi yang lebih baik daripada *single method* dan *Neural Network* dengan kondisi data besar yang mencakup data abnormal serta *missing value*. (Gumelar et al., 2020) juga melakukan penelitian dengan membandingkan XGBoost dan LSTM terkait harga penutupan saham 25 perusahaan, XGBoost menunjukan kinerja terbaik dengan 99% akurasi. Penelitian lain dilakukan oleh (Li Jiabao, 2022) yang membandingkan metode LSTM, LightGBM, dan CNN dalam memprediksi persediaan volatilitas, variabel yang menunjukan tingkat fluktuasi harga suatu aset, hasil ditemukan bahwa LightGBM yang paling cocok dalam memprediksi persediaan volatilitas dalam jangka pendek.

Metode LightGBM dan XGBoost semakin popular seiring dengan banyaknya penelitian yang membuktikan tingginya tingkat akurasi dari kedua metode tersebut. Hal ini menarik perhatian peneliti lain untuk membandingkan keduanya. (Rizky et al., 2022) dalam penelitiannya membandingkan kedua metode menggunakan data dengan kelas tidak seimbang. Hasil penelitian menemukan bahwa kinerja metode XGBoost dalam segi akurasi dan sensitivitas lebih baik dibandingkan LightGBM di hampir seluruh data. Sedangkan kemampuan dalam menebak kelas minoritas (spesifisitas), metode LightGBM lebih baik dibandingkan XGBoost.

Data harga saham membutuhkan akurasi tinggi sehingga model perlu dirancang sebaik mungkin dalam menangkap pola atau tren dari data harga saham yang bersifat time series. Pemilihan hyperparameter memiliki peran penting karena model seperti LightGBM dan XGBoost sangat bergantung pada pengaturan *learning rate*,

max_depth, *subsample*, dan parameter lainnya yang berfungsi dalam mengembangkan model hingga menghasilkan prediksi yang akurat. (Suhartono dan Lewi, 2022) dalam penelitiannya mengatakan bahwa penggunaan metode *hyperparameter optimization* seperti Grid Search diperlukan agar model dapat mencari konfigurasi *hyperparameter* terbaik, tapi Grid Search sendiri memiliki kelemahan dalam waktu pemrosesan yang cukup lama. Penelitian (Lim, 2022) menjelaskan bahwa optuna mampu mencoba kombinasi hyperparameter secara berulang dan memberhentikan proses pelatihan apabila tidak menunjukkan peningkatan signifikan. Hal ini membuktikan bahwa Optuna menghemat waktu dan lebih efektif dibandingkan metode tradisional seperti Grid Search.

Mcdonald's Corporation adalah salah satu perusahaan makanan cepat saji terbesar di dunia yang beroperasi sejak 1948 dan telah membuka 35.000+ cabang di berbagai negara tercatat tahun 2018. McDonald's dengan merek yang hampir dikenal dengan adaptasinya dalam pembuatan menu baru dan penyesuaian rasa di setiap negara membuat McDonald's tetap unggul dalam pasar internasional. Sistem manajemen yang terbuka dan kinerja keuangan yang konsisten stabil, tercatat dalam website resminya, menarik banyak investor untuk memilih saham McDonald's, namun terdapat berbagai cabang McDonald's di negara-negara pro-Palestina menghadapi boikot publik, termasuk Indonesia (Hanathasia dan Lestari, 2023). Akibat kondisi pasar yang tidak menentu, saham McDonald's menjadi subjek yang menarik untuk diprediksi karena investor perlu mempertimbangkan risiko yang ada sebelum mengambil keputusan investasi.

Berdasarkan penjabaran latar belakang di atas, maka peneliti tertarik untuk melakukan penelitian dengan judul “Perbandingan Metode LightGBM dan XGBoost dalam Peramalan Harga Saham Menggunakan *Hyperparameter Optimization Optuna*” sebuah studi kasus untuk harga saham McDonald’s.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini sebagai berikut:

1. Bagaimana perbandingan akurasi hasil prediksi model LightGBM dan XGBoost dalam memprediksi harga saham?
2. Bagaimana *hyperparameter tuning* dengan *Optuna Hyperparameter Optimization* mengembangkan kinerja model LightGBM dan XGBoost untuk peramalan harga saham?

1.3 Batasan Penelitian

Batasan masalah dalam penelitian ini sebagai berikut:

1. Penelitian ini menggunakan algoritma LightGBM dan XGBoost dalam membangun model peramalan.
2. *Hyperparameter tuning* menggunakan metode *Optuna Hyperparameter Optimization.*
3. *Tools* bahasa pemrograman yang digunakan dalam penelitian ini adalah *Python*.
4. Data yang digunakan dalam penelitian ini adalah data harga saham McDonald's. yang diambil dari November 2019 sampai November 2024.

5. Data *input* yang digunakan adalah data historis harga saham yang terdiri dari harga *low, open, close, high*, dan *volume*.
6. Parameter yang digunakan untuk mengevaluasi akurasi hasil peramalan harga saham adalah *Mean Absolute Percentage Error* (MAPE), *Mean Square Error* (MSE), *Root Mean Square Error* (RMSE), dan *R-Squared Score*

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan dari penelitian ini sebagai berikut:

1. Memperoleh hasil akurasi kinerja model peramalan algoritma LightGBM dan XGBoost dalam memprediksi harga saham.
2. Menganalisis pengaruh *hyperparameter tuning* dengan *Optuna Hyperparameter Optimization* dalam mengembangkan kinerja model LightGBM dan XGBoost.

1.5 Manfaat Penelitian

Penelitian ini bermanfaat dalam pengembangan pengetahuan terhadap peramalan harga saham menggunakan metode LightGBM dan XGBoost dan memberikan proyeksi model dari harga saham, agar para investor punya perencanaan yang matang sebelum melakukan investasi saham.

1.6 Landasan Teori

1.6.1 Investasi Saham

Investasi Saham menurut (Ricky, 2022) merupakan penanaman modal atau uang pada suatu perusahaan yang ditandai bukti kepemilikan modal dengan tujuan untuk memperoleh keuntungan berupa dividen dan kenaikan harga. Saham menjadi salah satu investasi dengan jumlah peminat yang besar, terbukti dengan meningkatnya 600 ribu investor baru tercatat sejak tahun 2023 hingga 9 agustus 2024 berdasarkan data dari *Single Investor Identification* (SID). Salah satu alasan yang menjadikan investasi saham lebih unggul dibandingkan dengan jenis investasi lain atau menyimpan uang di bank adalah kenaikan harga saham (*capital gain*) yang tinggi dan mampu imbangi harga inflasi. Investasi saham dengan banyak keunggulan, tentu juga memiliki risiko yang lebih tinggi akibat fluktuasi sehingga mampu memberikan kerugian yang besar apabila tidak dianalisis dengan baik.

Harga saham bersifat fluktuasi atau *non linear* tergantung kondisi perekonomian dan prestasi individual perusahaan. Kondisi perekonomian seperti inflasi, perubahan kebijakan pemerintah, ketegangan geopolitik, atau bahkan bencana alam dapat menyebabkan ketidakpastian yang memengaruhi pasar saham (Sumarjo, Mangantar, & Rumokoy, 2022). Hal ini merupakan faktor eksternal yang tidak dapat diprediksi melalui data, sehingga dibandingkan fokus terhadap faktor eksternal, terdapat faktor internal berupa prestasi individual perusahaan yang lebih mudah diprediksi. Faktor internal terbagi dua berdasarkan jenis data yang dianalisis, yaitu analisis fundamental dan analisis *technical*. Analisis fundamental adalah analisis

yang berfokus pada laporan keuangan dan bisnis perusahaan, sedangkan analisis *technical* adalah analisis yang berorientasi harga data historis dari pasar investasi (Agungtya et al., 2021).

Analisis fundamental mengarah pada penilaian nilai intrinsik suatu perusahaan dengan menilai kinerja keuangan berdasarkan berbagai indikator, seperti profitabilitas, likuiditas, solvabilitas, dan efisiensi operasional (Tandelilin, 2017). Investor menggunakan pendekatan ini dengan menganalisis data seperti pendapatan, laba bersih, arus kas, rasio keuangan, serta strategi bisnis perusahaan untuk menentukan apakah harga saham di pasar modal sudah mencerminkan nilai sebenarnya. Semakin baik kinerja keungan perusahaan dalam mengelola dana, maka semakin dinilai mampu mengoptimalkan nilai perusahaan (Putri et al., 2022). Dengan demikian, analisis ini menjadi alat penting bagi investor untuk menilai kelayakan investasi, khususnya dalam jangka panjang.

Selain analisis fundamental, investor juga perlu mempertimbangkan analisis *technical*, yaitu pendekatan yang berfokus pada perilaku harga saham di pasar modal untuk memprediksi pergerakan harga di masa mendatang. Analisis ini mencakup informasi seperti data harga pasar (*open, close, high*, dan *low*) serta *volume* perdagangan, yang mencerminkan kebiasaan para pelaku pasar, sehingga dapat membantu investor membuat keputusan yang lebih terinformasi (Maulana, 2021). Analisis *technical* bertujuan untuk mengidentifikasi dan memperhitungkan sentimen pasar, yang digambarkan melalui berbagai kondisi tren. Contohnya adalah tren naik (*bullish*) yang mencerminkan optimisme pasar, tren turun (*bearish*) yang mencerminkan pesimisme, serta tren mendatar (*sideways range trading*) yang menunjukkan ketidakpastian terhadap arah harga di masa depan (Rockefeller, 2020). Dengan demikian, analisis *technical* menjadi alat yang penting bagi *trader* atau investor jangka pendek untuk mengambil keputusan berdasarkan dinamika pasar secara *real-time*.

1.6.2 Peramalan

Peramalan adalah seni ukur yang memperkirakan sesuatu kejadian di masa depan dengan melibatkan analisa pola menggunakan data masa lalu. Peramalan bertujuan agar perhitungan yang telah diperkirakan bisa meminimumkan kesalahan peramalan, artinya perbedaan antara kenyataan dengan ramalan tidak begitu jauh (Mulyani et al., 2021).

(Makridakis et al., 1999) pertama kali klasifikasikan peramalan ke dalam dua kategori utama, yaitu peramalan kuantitatif dan kualitatif. Metode peramalan kualitatif digunakan ketika data historis terbatas atau tidak relevan sehingga peramalan didasari oleh opini ahli, intuisi, atau survei. Sebaliknya metode peramalan kuantitatif mengandalkan data historis numerik dan model matematis sehingga peramalan didasari pola pada data masa lalu yang dapat diukur dan diproyeksikan.

Metode peramalan kuantitatif terbagi menjadi dua jenis, yaitu metode regresi (kausal) dan metode *time series* (runtun waktu). Peramalan *time series* merupakan pendekatan kuantitatif untuk memperkirakan nilai di masa depan dengan mengandalkan data historis dari suatu variabel yang dikumpulkan secara berkala. Analisis time series mencakup berbagai teknik untuk mengevaluasi data runtun waktu dengan memanfaatkan parameter statistik serta karakteristik lain dari data guna memprediksi nilai mendatang berdasarkan pengamatan nilai sebelumnya (Phumchusri & Udom, 2014). Sementara itu, metode regresi berfokus pada pemodelan hubungan sebab-akibat antara variabel dependen (target) dengan variabel independen (prediktor)**,** seperti dalam analisis regresi linier dimana perubahan pada variabel prediktor digunakan untuk memprediksi variabel target dengan asumsi bahwa pola hubungan ini akan tetap konsisten di masa depan (Gujarati & Porter, 2009)

1.6.3 Decision Tree

Decision Tree adalah algotitma pembelajaran mesin berbentuk pohon yang digunakan dalam pengambilan keputusan seperti klasifikasi dan regresi berdasarkan data *input* (fitur). Penilitian (Eksa, 2016) membagi struktur dari Decision Tree yang terdiri dari simpul akar (*root node*), simpul keputusan (*decision nodes*), dan simpul daun (*leaf nodes*). Simpul akar mewakili pertanyaan pertama untuk memisahkan data berdasarkan fitur paling penting, selanjutnya hasil keputusan disepanjang pohon antara simpul akar dan simpul keputusan yang bertugas dalam menguji fitur tertentu disebut simpul keputusan, sedangkan simpul daun adalah hasil akhir dari keputusan.

Decision Tree bekerja dengan memilih fitur paling informatif untuk membagi dataset menjadi subset yang lebih kecil dan homogen. Proses dilakukan secara berulang hingga mencapai hasil keputusan atau tidak ada fitur tersisa untuk diproses. Pemilihan fitur paling informatif didasarkan pada pengukuran seberapa baik suatu fitur mampu mengurangi ketidakpastian (*impurity*) dalam data, yang dihitung menggunakan berbagai metode seperti berikut.

1. Entropi

   Konsep pengukuran entropi berasal dari teori informasi yang diperkenalkan oleh **Claude Shannon** pada tahun 1948, di mana informasi yang diperoleh dari suatu peristiwa i dengan probabilitas p_i adalah:

   $$I(p_i) = -\log_2 p_i \tag{1}$$

   Namun, karena setiap informasi pada peristiwa i menggunakan data acak dan setiap data bisa masuk ke berbagai peristiwa lain dengan probabilitas yang berbeda. Maka digunakan rumus harapan atau ekspektasi dari semua kemungkinan peristiwa/kelas c dan dijabarkan sebagai berikut:

   $$\mathbb{E}\,[I(p_i)] = \mathbb{E}\,[-\log_2 p_i] \tag{2}$$

   Sehingga,

$$\mathbb{E}\,[I(p_i)] \;=\; \sum_{i=1}^{c} p_i \,.\, -\log_2 p_i \;\;=\; -\sum_{i=1}^{c} p_i \,.\, \log_2 p_i \tag{3}$$

Kesimpulan yang diperoleh, ekpektasi pada entropi berfungsi menghitung nilai rata-rata informasi dari setiap kemungkinan peristiwa/kelas. Semakin besar nilai entropi artinya semakin banyak kemungkinan besar pada tiap peristiwa/kelas dan mengakibatkan ketidakpastian bertambah, sehingga semakin sulit menebak peristiwa/kelas suatu sampel.

2. *Information Gain* (IG)

Konsep pengukuran information gain menggunakan entropi, yang mengukur selisih antara entropi sebelum dan sesudah pemisahan dataset berdasarkan suatu fitur tertentu. Misalkan H menyatakan entropi dan S menyatakan himpunan data sebelum pemisahan (*parent node*):

$$H(S) \;=\; -\sum_{i=1}^{c} p_i \,.\, \log_2 p_i \tag{4}$$

Jika terdapat S_v yang menyatakan subset hasil himpunan data setelah pemisahan (*child node*). Maka terdapat $H(S_v)$ yang merupakan entropi dari setiap *child node* dan terdapat A sebagai fitur yang digunakan untuk pemisahan, maka:

$$IG\;(S,A) = H(S) \;\; - \sum_{v\;\in\; Values(A)} \frac{|S_v|}{|S|} . H(S_v) \tag{5}$$

Tanda *absolut* pada himpunan menyatakan jumlah elemen, sehingga $|S_v|$ adalah jumlah elemen dalam subset S_v dan $|S|$ adalah jumlah elemen pada dataset awal. Jika nilai Information Gain tinggi, itu berarti nilai entropi setelah pemisahan rendah atau ketidakpastian mengecil. Kesimpulan yang diperoleh, atribut yang digunakan sangat efektif.

3. Gain Ratio

Fitur dengan banyak kategori peristiwa/kelas seperti Nomor Identitas memiliki nilai berbeda pada setiap data. Hal ini akan dinilai sangat informatif bagi algoritma *Information Gain*, padahal secara logis tidak relavan dengan hasil prediksi klasifikasi maupun regresi.

Mengatasi masalah ini, Quinlan memperkenalkan *split information* yang mengukur seberapa besar fitur dalam membagi data. *Split information* juga mengambil ide dari teori informasi **Claude Shannon** dan mirip dengan entropi**,** tetapi bukan mengukur ketidakpastian peristiwa/kelas, melainkan mengukur ketidakpastian distribusi data setelah dibagi oleh fitur. Misalkan distribusi data $\frac{|S_v|}{|S|}, v \in Values(A)$, dan fitur A, maka:

$$Split\ Information\ (A)\ =\ -\sum_{v}\frac{|S_v|}{|S|}\ .\log_2\frac{|S_v|}{|S|} \tag{6}$$

Dengan adanya *Split Information*, Quinlan mendefinisikan *Gain Ratio* (GR) sebagai:

$$GR(A) = \frac{IG\ (A)}{Split\ Information\ (A)} \tag{7}$$

Split Information pada fitur yang membagi data dengan jumlah besar, mengakibatkan distribusi data $\frac{|S_v|}{|S|}$ menjadi lebih kecil untuk tiap *subset*, sehingga $\log_2\frac{|S_v|}{|S|}$ akan memperoleh nilai negatif yang besar. Dengan menjumlahkan banyak nilai negatif dari tiap distribusi data, maka *Split Information* menjadi besar. Akibat dari *Split Information* yang besar, *Gain Ratio* akan kecil dan menandakan fitur tersebut tidak terlalu berguna.

4. Gini Index

   Konsep pengukuran Gini Index menggunakan probabilitas dengan mengukur kemungkinan bahwa dua sampel yang dipilih secara acak dari dataset akan diklasifikasikan ke dalam kelas yang berbeda.

   Misalkan memilih satu sampel dari peristiwa/kelas i dengan probabilitas p_i dan memilih satu sampel lainnya dari kelas yang sama p_i. Maka, probabilitas memilih dua sampel dengan kelas yang sama adalah $\sum_i p_i^2$. Sehingga probabilitas memilih dua sampel dengan kelas yang berbeda (Gini Index) adalah:

$$Gini = 1 - \sum_i p_i^2 \tag{8}$$

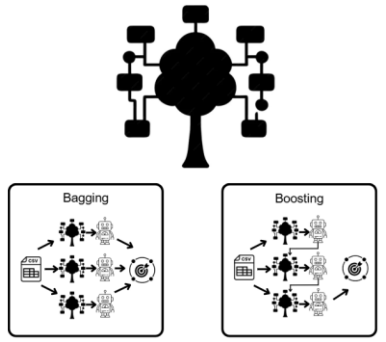
   Semakin rendah Gini Index, semakin homogen dataset dalam suatu *node*, karena ketidakpastian tentang kelas berkurang. Dalam hal ini, keputusan atau prediksi yang dibuat oleh pohon keputusan menjadi lebih pasti dan jelas.

$$Gain = Gini_{parent\ node} - Gini_{child\ node} \tag{9}$$

   Cara yang sama dari *Information Gain* digunakan untuk menentukan fitur terbaik, dengan menghitung selisih antara Gini Index sebelum dan sesudah pemisahan dataset berdasarkan fitur tertentu. *Gain* tertinggi dipilih sebagai fitur terbaik untuk membagi data di setiap *node* pohon keputusan. Gini Index lebih cepat dihitung dibandingkan entropi karena tidak menggunakan logaritma di dalam rumus.

Pemilihan fitur di atas berasal dari tiga algoritma pembentukan pohon keputusan yang paling *popular* yaitu *Iterative Dichotomiser* 3 (ID3), C4.5 yang merupakan adopsi dari ID3, dan *Classification and Regression Trees* (CART). ID3 menggunakan

Information Gain untuk menentukan atribut terbaik berdasarkan pengurangan entropi. C4.5 adalah pengembangan ID3 yang mengatasi keterbatasan ID3 dengan menggunakan *Gain Ratio* untuk mencegah kecenderungan terhadap atribut dengan banyak nilai dan mampu menangani data kontinu. Sementara itu, CART menggunakan Gini Index sebagai kriteria pemilihan fitur dan menghasilkan pohon biner, yang membuatnya lebih fleksibel dalam menangani data dengan ketidakpastian untuk prediksi masalah klasifikasi maupun regresi



Gambar 2.1 Ensamble Learning dari Decision Tree

Sumber: Huan et al, 2022

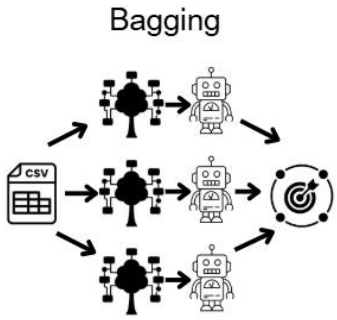
Decision Tree adalah metode yang kuat dalam prediksi, namun banyak penelitian yang menggabungkan *decision tree* dengan pendekatan lain untuk meningkatkan akurasi dan stabilitas. Dua pendekatan utama adalah *bagging* dan *boosting*. Salah satu contoh *bagging* adalah *Random Forest*, membuat beragam *Decision Trees* melalui pengambilan sampel acak (*bookstrap sampling)* dari dataset. Setiap pohon memberikan suara untuk hasil akhir, yang membantu mengurangi *varians* model. Sedangkan *Boosting* berfokus pada pembangunan model secara bertahap, dengan menyesuaikan pohon baru untuk memperbaiki kesalahan dari model sebelumnya. Metode ini banyak diterapkan dalam algoritma seperti *Gradient Boosting*

1.6.4 Random Forest

Random Forest adalah salah satu algoritma yang menggunakan teknik *bootstrap aggregating* (*bagging*). *Bagging* adalah metode *ensemble* yang menggabungkan beberapa model prediktif, dalam hal ini *random forest* menggunakan *decision trees*, yang dibuat dengan *bootstrap sampling*. *Bootstrap sampling* adalah teknik pengambilan sampel secara acak dengan pengembalian (*replacement*) dari dataset asli. Penelitian oleh (Kulesa et al, 2015) menyimpulkan bahwa pengembalian data berarti data yang telah dipilih akan dimasukkan ke dalam sampel dan data tersebut dapat dipilih kembali. Dengan kata lain, setiap elemen dalam dataset asli memiliki peluang untuk melatih lebih dari satu pohon, sementara beberapa data mungkin tidak digunakan untuk melatih pohon sama sekali.

Data yang tidak dipilih untuk membangun pohon keputusan disebut *out-of-bag* (OOB) *samples*. Data ini tidak digunakan untuk melatih pohon tersebut, namun bisa digunakan sebagai data penguji performa pohon setelah pelatihan selesai. Hal ini menambah keunikan *Random Forest* sehingga data uji tidak bergantung pada pelatihan, mengurangi risiko *overfitting*, dan lebih mampu menangani data yang bervariasi. (Janitza & Hornung, 2018)

Random Forest juga menggunakan sampel acak dalam pemilihan fitur. Setiap kali pohon membagi suatu *node*, bukan menggunakan semua fitur secara berurut, melainkan hanya sejumlah fitur yang dipilih secara acak untuk dipertimbangkan dalam proses pembagian data dalam suatu node. Hal ini mengurangi korelasi antar pohon dan meningkatkan kemampuan generalisasi *Random Forest*. Oleh karena itu, meskipun setiap pohon dapat *overfitting* pada data yang dilatih, gabungan banyak pohon yang dilatih pada data dan fitur yang berbeda memberikan model yang lebih stabil dan akurat.



Gambar 2.2 Cara kerja *Random Forest*

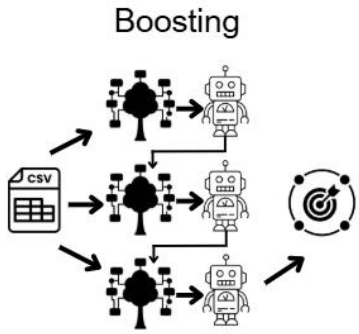
Setelah proses pelatihan selesai, prediksi akhir dilakukan dengan cara *voting* mayoritas untuk klasifikasi atau rata-rata untuk regresi dari prediksi yang diberikan oleh setiap pohon keputusan. Teknik ini meningkatkan akurasi model secara signifikan karena keputusan akhir bergantung pada prediksi kolektif dari banyak pohon yang dilatih, mengurangi efek kesalahan dari satu pohon yang mungkin terjadi *overfitting*.

Random Forest juga menyediakan *feature importance* yang mengukur kontribusi setiap fitur dalam membuat prediksi. Mengukur kontribusi setiap fitur terhadap penurunan ketidakpastian Gini Index atau *Information Gain* pada setiap pohon, *Random Forest* mampu menunjukkan fitur-fitur mana yang paling berpengaruh dalam membuat keputusan, yang dapat sangat berguna dalam analisis data dan pemilihan fitur.

1.6.5 Gradient Boosting

Gradient Boosting adalah salah satu algoritma *machine learning* yang bekerja dengan membuat model yang kuat secara berulang dari model-model sebelumnya yang dianggap lemah. Algoritma ini mengandalkan pohon keputusan (*decision tree)*, setiap model dibangun berulang dengan memperbaiki gradiennya agar model yang baru menghasilkan nilai yang lebih akurat, sehingga algoritma ini sering disebut

Gradient Boosting Decision Tree (GDBT). *Gradient Boosting* muncul dari ide observasi Leo Breiman 1997 yang menyadari bahwa *boosting* dapat diinterpretasikan sebagai algoritma optimasi fungsi biaya. Penelitian dilakukan oleh (Jerome, 2001) merumuskan algoritma regresi gradien sebagai *Gradient Boosting Machine* yang menjadi landasan matematis modern. Penelitian baru juga bermunculan dengan mengimplementasikan gradient boosting dalam beberapa algoritma seperti LightGBM (*Light Gradient Boosting Machine*), XGBoost (*Extreeme Gradient Boosting*), dan CatBoost.



Gambar 2.3 Cara kerja Gradient Boosting

Gradient Boosting bekerja dengan membangun model secara bertahap berdasarkan kinerja dan kesalahan dari model sebelumnya. Setiap iterasi dalam pembuatan model menggunakan fungsi loss seperti *Mean Squared Error* (MSE) untuk regresi atau *log-loss* untuk klasifikasi, *loss function* bekerja dengan menghitung residu antara prediksi model saat ini dan nilai sebenarnya. Proses yang dilakukan dalam mengurangi nilai residu untuk meminimalkan *loss function* adalah *gradient descent*. *Gradient Descent* bekerja dengan cara memperbarui parameter model secara iteratif ke arah negatif dari gradien fungsi tersebut.

Misalkan $D = (x_i,\ y_i)_{i=1}^{n} = (x_1, y_1), \dots, (x_n, y_n)$ sebagai data latih, dengan x_i menyatakan fitur dan y_i adalah target. Secara matematika *gradient boosting* dapat dialgoritmakan sebagai berikut:

1. Inisialisasi Model Awal
   Model diawali dengan prediksi konstan yang meminimalkan *loss function*

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{9}$$

   Keterangan
   $f_0\ (x)$: Model iterasi pertama
   $L(y_i, \gamma)$: *loss function* antara nilai aktual y_i dengan nilai prediksi γ
   n : Jumlah data sampel

2. Iterasi Pembelajaran

Pada setiap iterasi *m, di mana* $m = 1, 2, 3, \dots, M$ algoritma berjalan sebagai berikut:

a. Hitung Pseudo-Residual

Pseudo-residual menyatakan gradien negatif fungsi kerugian terhadap prediksi model sebelumnya sehingga:

$$r_i^{(m)} = -\left[\frac{\delta L\big(y_i, f_{m-1}(x_i)\big)}{\delta f_{m-1}(x_i)}\right] \tag{10}$$

Keterangan

$r_i^{(m)}$: Residual pada sampel ke-i dan iterasi ke-m

$f_{m-1}(x_i)$: Prediksi model sebelum iterasi ke-i

L : *Loss function*

Secara umum *loss function* menggunakan Mean Squared Error (MSE) atau Mean Absolute Error (MAE).

b. Perbaikan Residual dengan Model Baru

Gunakan h sebagai fungsi untuk memperbaiki nilai residual pada model $f_{m-1}(x_i)$.

$$D_{modified} = (x_i,\ r_i^m)_{i=1}^n \tag{11}$$

Tahap ini mengambil dataset baru dari sampel dataset sebelumnya untuk dimodifikasi dengan melibatkan berbagai cara seperti menghapus outlier, mengisi nilai hilang, melakukan *resampling*, dan sebagainya sehingga model pada iterasi ini dibuatkan pohon keputusan sederhana yang lebih memperhatikan kesalahan model sebelumnya dibanding data target. Sehingga dapat diinterpretasikan sebagai berikut.

$$h_m(x) = \arg min \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma h(x_i)) \tag{12}$$

Model baru $h_m(x)$ adalah hasil dari fungsi tambahan h yang dilatih dan dicocokkan pada dataset yang telah dimodifikasi.

c. Penentuan Bobot Optimasi

Diketahui fungsi objektif

$$\mathcal{L} = \sum_{i=1}^{n} L(y_i, f_m(x_i)) \tag{13}$$

Mencari $\gamma_{optimum}$ dapat diselesaikan dengan mencari argumen dari minimal antara model sebelumnya dengan model baru.

$$\gamma_{opt} = \arg\min \sum_{i=1}^{N} L\left(y_i, f_m(x_i)\right) \tag{14}$$

$$\gamma_{opt} = \arg\min \sum_{i=1}^{N} L(L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i)\,)(x_i)\,) \tag{15}$$

Bobot optimasi γ_m yang telah diperoleh selanjutnya dimasukkan ke model baru.

d. Perbarui Model
Model diperbarui dengan menambahkan basis learner yang diskalakan *learning rate* (v):

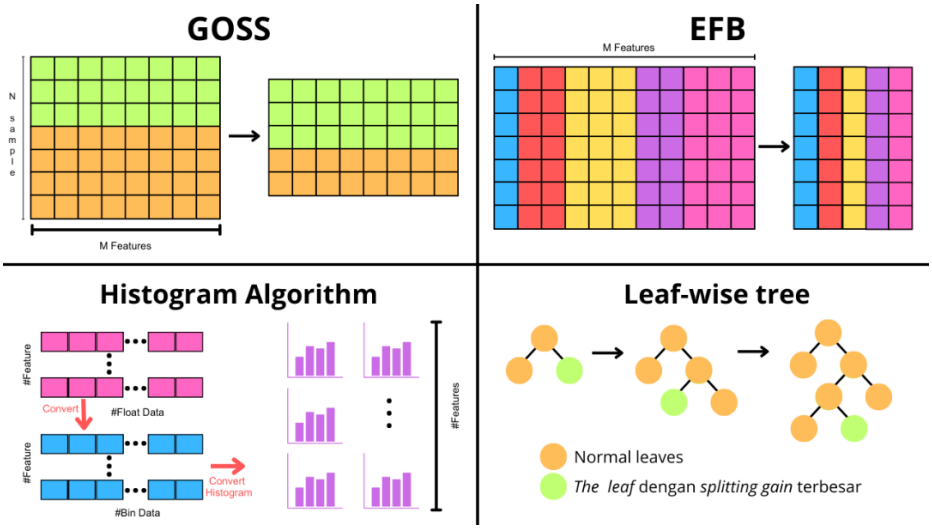
$$f_{m(x)} = F_{m-1}(x) + v.\gamma_m h_m(x) \tag{16}$$

3. Output Model Tetakhir
Setelah M iterasi, model akhir merupakan penjumlahan tertimbang dari

$$F_M = F_0(x) + v.\sum_{m=1}^{M} \gamma_m h_m(x) \tag{17}$$

1.6.6 Light Gradient Boosting Machine (LightGBM)

LightGBM (*Light Gradient Boosting Machine*) adalah sebuah *framework* gradient boosting yang dikembangkan oleh *Microsoft* pada tahun 2017. *Framework* ini dirancang untuk mengatasi keterbatasan metode gradient boosting tradisional dengan teknik optimasi khusus untuk menangani data besar dan high-dimensional. Teknik yang dimaksud untuk meningkatkan performa dalam LightGBM adalah *Gradient-based One-Side Sampling* (GOSS), *Exclusive Feature Bundling* (EFB), *Histogram Algorithm*, dan *Leaf-Wise Tree Growth*.



Gambar 2.4 Elemen LightGBM

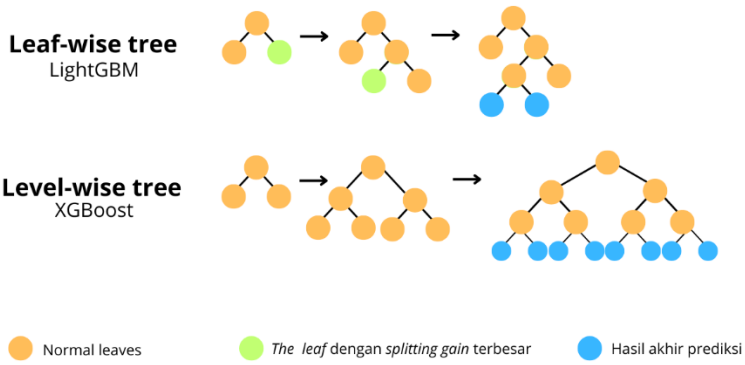
Sumber: Zhao et al, 2023

GOSS bekerja dengan mempertahankan *instance* data yang memiliki gradien besar dan secara acak mengabaikan sebagian *instance* dengan gradien kecil, sehingga mengurangi jumlah data yang perlu diproses tanpa kehilangan akurasi signifikan. EFB mengidentifikasi fitur-fitur yang tidak pernah aktif bersamaan (*exclusive*) dan menggabungkannya menjadi satu fitur, yang mengurangi dimensi data dan mempercepat komputasi. *Histogram Algorithm* membagi nilai-nilai kontinu dari setiap fitur menjadi beberapa interval atau bin. Misalnya, jika sebuah fitur memiliki nilai antara 0 dan 100, algoritma dapat membaginya menjadi 10 bin, di mana setiap bin mewakili rentang nilai tertentu (misalnya, 0-10, 10-20, ..., 90-100). *Histogram* ini menyimpan informasi statistik penting, seperti jumlah sampel (*count*) dan jumlah gradien (*gradient sum*) dalam setiap bin. Informasi ini digunakan untuk menghitung *gain* (peningkatan kualitas *split*) saat membangun pohon keputusan. *Leaf-Wise Tree Growth* memilih daun yang memberikan pengurangan loss terbesar untuk dipecah, membuat pohon tumbuh lebih efisien dan akurat.

1.6.7 Extreme Gradient Boosting (XGBoost)

XGBoost (*Extreme Gradient Boosting*) adalah sebuah *framework gradient boosting* yang dikembangkan oleh Chen dan Guestrin pada tahun 2014. Mirip dengan LightGBM, XGBoost juga menggunakan pendekatan berbasis gradien untuk meminimalkan *loss function*. Namun, XGBoost menggunakan pendekatan level-wise tree growth, di mana pohon tumbuh secara seimbang di setiap level. XGBoost juga menerapkan teknik *column block* untuk memproses fitur-fitur secara paralel, sehingga mempercepat proses *training*. XGBoost mengambil ekspansi Taylor dari *loss function* hingga orde kedua dan menambahkan istilah regularisasi untuk

menemukan solusi optimal, yang digunakan untuk menyeimbangkan penurunan *loss function* dan kompleksitas model untuk menghindari *overfitting* (Chen & Guestrin., 2018).



Gambar 2.5 Perbedaan pohon pada LightGBM dan XGBoost

Pada *leaf-wise tree*, algoritma selalu memilih *leaf* (daun) dengan *loss* terbesar untuk dipisah terlebih dahulu, sehingga menghasilkan pohon yang lebih dalam di area tertentu. Ini membuat *leaf-wise tree* lebih efisien dalam mengurangi *loss*, tetapi berisiko *overfitting* jika data tidak diatur dengan baik. Sebaliknya, *level-wise tree* menambahkan cabang secara seimbang di semua *level* pohon, sehingga lebih stabil dan mudah diatur, tetapi kurang efisien karena memproses semua *node* di tingkat yang sama, termasuk yang kurang informatif. *Leaf-wise* tree lebih fokus pada performa, sedangkan *level-wise tree* lebih mengutamakan keseimbangan dan interpretabilitas.

Fungsi objektif pada *gradient boosting* adalah fungsi yang meminimalkan *loss function* untuk suatu dataset seperti yang tertulis pada rumus/poin 13. Sedangkan pada XGBoost, fungsi objektif didefinisikan sebagai jumlahan dari *loss function* dan regularisasi.

$$\mathcal{L} = \sum_{i=1}^{n} L(y_i, \gamma) + \sum_{k=1}^{K} \Omega\,(f_m) \tag{18}$$

Regularisasi untuk fungsi prediktor f_m dirumuskan sebagai:

$$\Omega\,(f_m) = \eta T + \frac{1}{2}\,\lambda \|w\|^2 \tag{19}$$

Keterangan

T : Jumlah daun dalam tree
ω : Berat (*weihtgs*) yang terkait dengan daun
η : Parameter yang mengontrol penalti untuk jumlah daun
λ : Parameter regularisasi untuk menghindari *overfitting*

Dengan menggunakan konsep *gradient boosting* iterasi berulang dengan mengasumsikan $F_0 = 0$, diperoleh:

$$F_m = \sum_{m=1}^{M} \gamma_m h_m(x_i) = \sum_{m=1}^{M} f_m(x_i) \; = F_{m-1} + f_m(x_i) \tag{20}$$

Dengan mengkombinasikan formula (9) dan (10), fungsi objektifnya dapat dituliskan sebagai berikut:

$$\mathcal{L}^m = \sum_{i=1}^{n} [L\big(y_i, \gamma^{m-1} + f_m(x_i)\big) + \; \Omega\,(f_m) \tag{21}$$

Untuk menemukan fungsi objektif yang dapat diminimalkan, ambil perluasan Taylor dari *loss function* hingga orde kedua, maka fungsi objektifnya menjadi:

$$\mathcal{L}^m = \sum_{i=1}^{n} [L\big(y_i, \gamma^{m-1} + f_m(x_i)\big) + \frac{1}{2}\, h_i f_m^2(x_i)] + \; \Omega\,(f_m) \tag{22}$$

Tambahkan *loss function* untuk setiap data dan prosesnya akan menjadi sebagai berikut:

$$\mathcal{L} = \sum_{i=1}^{n} [g_i f_m(x_i) + \frac{1}{2}\, h_i f_m^2(x_i)] + \; \Omega\,(f_m) \tag{23}$$

$$\mathcal{L} = \sum_{i=1}^{n} [g_i w_q(x_i) + \frac{1}{2}\, h_i w_q^2(x_i)] + \; \Omega\,(f_m) + \lambda T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{24}$$

$$\mathcal{L} = \sum_{j=1}^{n} [\left(\sum\nolimits_{i \,\epsilon\, I_j} g_i\right) w_j + \frac{1}{2}\left(\sum\nolimits_{i \,\epsilon\, I_j} h_i + \lambda\right) w_j^2] \; + \; \lambda T \tag{25}$$

Keterangan

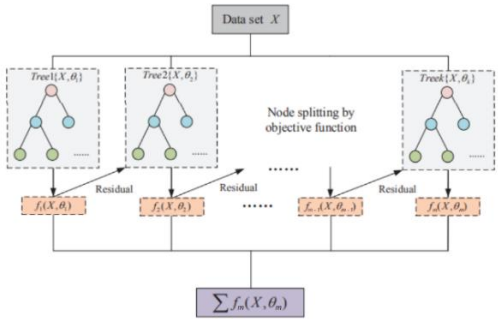
$g_i = \; \partial \gamma^{m-1}\, L\,(y_i, \gamma^{m-1})$: Turunan pertama dari *loss function*
$h_i = \; \partial^2 \gamma^{m-1}\, L\,(y_i, \gamma^{m-1})$: Turunan kedua dari *loss function*

Nilai w optimal dan fungsi objektif yang diperoleh dari solusi adalah sebagai berikut:

$$w_j = -\frac{G_j}{H_j + \lambda}$$

$$\mathcal{L} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j}{H_j + \lambda} + \lambda T, \qquad G_j = \sum_{i \in I_i} g_i \;\; dan \; H_j = \sum_{i \in I_j} h_i \tag{26}$$

Selama proses pelatihan, model terus menghitung kehilangan simpul (*loss node*) untuk memilih simpul daun dengan kehilangan *gain* terbesar. XGBoost menambahkan pohon baru dengan terus membagi fitur. Menambahkan pohon setiap kali mempelajari fungsi baru $f_m(x, \theta_m)$ untuk menyesuaikan residu prediksi terakhir.



Gambar 2.6 Cara kerja XGBoost

Ketika pohon M diperoleh setelah pelatihan yang Panjang, fitur dari prediksi sampel akan bersesuaian pada setiap pohon, begitu juga untuk setiap simpul daun (leaf node). Skor yang saling bersesuaian pada setiap pohon dijumlahkan untuk memperoleh nilai prediksi dari sampel.

1.6.8 H*yperparameter Tuning*

Hyperparameter merupakan sekumpulan parameter utama yang berfungsi sebagai pengendali dalam proses pembelajaran mesin. Berbeda dengan parameter model yang diperoleh melalui proses pelatihan, hyperparameter harus ditetapkan sebelum pelatihan dimulai dan bersifat tetap selama proses pembelajaran berlangsung. Dalam konteks pemodelan prediktif, hyperparameter dapat dianggap sebagai parameter kritis yang memiliki pengaruh dominan terhadap performa model, terutama dalam hal akurasi prediksi dan kemampuan generalisasi. Contoh hyperparameter termasuk learning rate, momentum, weight decay, batch size, dan parameter lainnya yang mempengaruhi perilaku algoritma pembelajaran (Smith, 2019).

Proses hyperparameter tuning adalah upaya optimasi hyperparameter dengan mencari kombinasi terbaik dari beberapa parameter, yang bertujuan untuk meningkatkan akurasi prediksi serta efisiensi model (Hossain and Timmer, 2021). Hyperparameter tuning sangat penting dilakukan pada model untuk mencegah overfitting atau underfitting terutama bagi model yang membutuhkan banyak parameter kompleks dan struktur yang lebih rumit, seperti neural networks atau gradient boosting.

1.6.9 *Optuna Hyperparamete Optimization*

Optuna adalah alah satu framework open-source yang dikembangkan pada tahun 2019 oleh Preferred Network, sebuah Perusahaan riset AI asal jepang. Optuna dirancang untuk menjalankan program dengan kriteria *Define-by-run,* artinya perlu didefinisikan terlebih dahulu oleh pengguna sebelum menjalankan algoritma, sehingga optuna mengoptimalkan model secara dinamis. Hal ini membuat optuna fleksibel mengeksplorasi solusi dalam ruang pencarian pada ekspeimen skala besar dan memungkinkan pemangkasan (*pruning*) dan pengambilan sampel yang efisien.

Fungsi objektif (objective function) adalah inti dari proses optimisasi dalam Optuna. Fungsi ini mendefinisikan metrik atau tujuan yang ingin dioptimalkan, seperti akurasi model, loss, atau metrik evaluasi lainnya. Misalkan fungsi objektif dipresentasikan sebagai $f(x)$ dengan x sebagai vector hyperparameter yang ingin dioptimalkan dan $x = (x_1, x_2, \dots, x_n)$, di mana setiap x_i mewakili satu parameter seperti *learning rate*, *max depth*, dan lainnya.

Tujuan dari Optuna adalah menemukan nilai x^* yang meminimalkan atau memaksimalkan $f(x)$, tergantung pada masalahnya sehingga hyperparameter terbaik adalah:

$$x^* = \arg\min f(x) \text{ atau } x^* = \arg\max f(x) \tag{27}$$

Proses optuna pada iterasi awal menggunakan sampling acak untuk memilih hyperparameter awal. Setelah beberapa iterasi awal, Optuna beralih dari sampling acak ke pendekatan yang lebih cerdas dengan memanfaatkan informasi dari evaluasi fungsi objektif sebelumnya. Ini dilakukan menggunakan teknik optimisasi Bayesian, khususnya Tree-structured Parzen Estimator (TPE).

1.6.10 Time Series Cross Validation (TSCV)

Time Series Cross-Validation (TSCV) adalah metode validasi yang dirancang khusus untuk data deret waktu, metode ini mengasumsikan bahwa urutan temporal data merupakan bagian yang sangat penting karena kemampuannya dalam mempelajari pola musiman. erbeda dengan *k-fold cross-validation* konvensional yang melakukan *random shuffling*, TSCV membagi data secara berurutan sehingga model selalu dilatih pada data historis dan divalidasi pada data masa depan (Bergmeir & Benítez, 2012). Hal ini secara eksplisit mencegah *data leakage*, yaitu situasi ketika informasi dari masa depan secara tidak sengaja memengaruhi proses pelatihan model.

Penggunaan tuning hyperparameter menggunakan Optuna, TSCV membantu menemukan kombinasi hyperparameter yang benar-benar generalizable karena model diuji dalam skenario yang menyerupai prediksi data baru di dunia nyata. Misalnya, jika data mencakup periode Januari-Desember, TSCV akan melatih model pada Januari-Juni dan mengujinya pada Juli-Agustus, lalu memperluas training set ke September dan menguji Oktober, dan seterusnya. Pendekatan ini memaksa model untuk beradaptasi dengan pola baru secara bertahap, sehingga ketika data test benar-benar baru (misalnya, tahun berikutnya), model sudah terbiasa dengan dinamika perubahan data.

1.6.11 Metode Evaluasi Kinerja Model

Evaluasi kinerja model sangat penting dilakukan pada jenis peramalan kuantitatif seperti metode *machine learning* untuk menilai tingkat akurasi sebuah model ketika dijalankan. Evaluasi kinerja model dalam penelitian ini menggunakan empat metrik utama, yaitu *Mean Absolute Error* (MAE) yang menghitung rata-rata selisih absolut antara nilai aktual dan prediksi, *Mean Absolute Percentage Error* (MAPE) yang mengukur persentase kesalahan peramalan, *Root Mean Square Error* (RMSE) yang mengkaji selisih antara nilai aktual dan nilai prediksi, R-*squared score* yang mengukur seberapa baik model menjelaskan variabilitas data, serta *Directional Accuracy (*DA) yang mengukur seberapa sering model berhasil memprediksi arah pergerakan harga saham dengan benar.

1. *Mean Absolute Error* (MAE)
   MAE merupakan metrik yang digunakan untuk mengukur rata-rata selisih absolut antara nilai aktual dan nilai prediksi. Metrik ini menunjukkan seberapa besar kesalahan rata-rata yang terjadi dalam unit yang sama dengan data asli.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{28}$$

   Keterangan:
   - y_i menyatakan nilai aktual
   - $\hat{y}_i$ menyatakan nilai prediksi
   - n menyatakan jumlah data

   MAE menghitung kesalahan tanpa memperhitungkan arah kesalahan seperti nilai positif maupun negatif.

2. *Mean Absolute Percentage Error* (MAPE)
   Mean Absolute Percentage Error (MAPE) merupakan metrik yang digunakan untuk mengukur prediksi dengan cara menghitung nilai rata-rata selisih antara nilai aktual dan nilai prediksi pada sepanjang garis regresi, hasilnya dinyatakan dalam bentuk persentase.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| . 100\% \quad (29)$$

MAPE hanya mengubah perhitungan MAE ke dalam bentuk persentase sehingga memberikan interpretasi yang mudah tentang seberapa jauh prediksi menyimpang dari nilai sebenarnya. Nilai evaluasi MAPE memiliki kriteria sebagai berikut (Nurvianti et al, 2019):

MAPE < 10%	: Kemampuan prediksi sangat baik
10% < MAPE < 20%	: Kemampuan prediksi baik
20% < MAPE < 50%	: Kemampuan prediksi cukup baik
MAPE >50%	: Kemampuan prediksi buruk

3. *Mean Square Error* (MSE)
   Mean Squared Error (MSE) adalah metrik evaluasi yang mengukur rata-rata kuadrat selisih antara nilai aktual (y_i) dan nilai prediksi ($\hat{y}_i$).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (30)$$

   MSE memberikan penalti lebih besar pada kesalahan prediksi yang besar karena menggunakan kuadrat residual. Hal Ini berguna untuk mendeteksi anomali atau prediksi yang sangat jauh dari nilai sebenarnya. Nilai MSE yang rendah menunjukkan akurasi prediksi yang tinggi, sedangkan nilai MSE yang tinggi mengindikasikan ketidakpresisian model dalam memprediksi data.

4. *Root Mean Square Error* (RMSE)
   Root Mean Square Error (RMSE) merupakan metrik pengembangan dari *Mean Square Error* (MSE). RMSE adalah nilai akar kuadrat dari MSE.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (31)$$

   RMSE mengukur seluruh nilai rata-rata kesalahan prediksi model dalam satuan yang sama dengan data asli.

5. R-*squared* (R^2) *score*
   R-*squared* (R^2) *score* merupakan metrik yang mengukur seberapa baik model regresi menjelaskan variabilitas dengan data.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{32}$$

Keterangan tambahan $\bar{y}$ menyatakan rata-rata dari nilai aktual. Nilai R^2 berkisar antara 0 hingga 1. Nilai 1 menandakan model cerdas dalam menjelaskan seluruh variabilitas dalam data, begitu juga sebaliknya dengan nilai 0.

Meskipun terdapat berbagai metrik evaluasi lainnya, keempat metrik ini dianggap cukup untuk memberikan gambaran akurat mengenai tingkat kesalahan prediksi serta kualitas hasil peramalan pada penelitiaan ini

BAB II
METODOLOGI PENELITIAN

2.1 Sumber dan Jenis Data

Penelitian ini menggunakan data sekunder, yaitu data yang dikumpulkan oleh pihak kedua. Data pada penelitian ini merupakan data historis harga saham harian *McDonald's Corporation* periode November 2019 hingga November 2024 yang diperoleh secara *online* melalui website https://finance.yahoo.com/ dengan kode MCD. Penelitiaan ini menggunakan lima dari enam fitur yang tersedia di dalam *website*, yaitu data *open, low, high, volume*, dan *close*.

2.2 Metode Penelitian

Metode penelitian ini menggunakan dua model *machine learning*, yaitu LightGBM dan XGBoost, untuk memprediksi harga saham. Kedua model ini akan dibandingkan berdasarkan kinerjanya dalam melakukan prediksi, yang kemudian dievaluasi menggunakan empat metrik evaluasi, yaitu MAPE untuk mengukur persentase kesalahan absolut dari metrik MAE, MSE yang mengukur rata-rata kuadrat selisih antara nilai aktual dan prediksi, RMSE yang mengukur rata-rata kesalahan prediksi model dalam datuan yang sama dengan data asli, R^2 *score* yang mengukur seberapa baik model menjelaskan variabilitas data.

2.3 Prosedur Penelitian

2.3.1 Identifikasi Masalah

Identifikasi masalah diatur untuk menganalisis seluruh faktor hambatan dalam menyelesaikan hasil penelitian. Tahap ini penting agar penelitian memiliki fokus yang terarah dalam menyelesaikan penelitian.

2.3.2 Studi Literatur

Studi literatur adalah tahap awal yang dilakukan untuk memahami perkembangan penelitian yang telah ada dan menemukan celah yang dapat dijadikan dasar dalam penelitian baru. Melalui tinjauan terhadap berbagai sumber ilmiah, peneliti dapat memperoleh wawasan tentang teori, metode, dan pendekatan yang telah digunakan sebelumnya.

2.3.3 Pengumpulan Data

Data historis harga saham McDonald's diperoleh dari Yahoo Finance, yang mencakup informasi seperti harga pembukaan (open), penutupan (close), harga tertinggi (high), terendah (low), serta volume perdagangan. Data ini dikumpulkan selama lima tahun, yaitu dari November 2019 hingga November 2024. Rentang waktu tersebut dipilih karena dianggap cukup representatif dalam menangkap dinamika pergerakan harga saham, baik dalam kondisi pasar yang stabil maupun saat terjadi volatilitas ekstrem. Periode ini mencakup masa awal pandemi COVID-19, pemulihan ekonomi pasca-pandemi, hingga fluktuasi yang muncul akibat

dinamika geopolitik dan inflasi global, sehingga menyediakan keragaman pola pasar yang diperlukan untuk membangun model prediksi yang andal.

2.3.4 *Preprocessing* Data

Data yang telah dikumpulkan kemudian diolah untuk memastikan kualitasnya baik sebelum digunakan dalam pelatihan model. Tahapan dalam *preprocessing* data diantaranya *reduce memory*, *data transformation, dan input missing value/ cleansing data*. Proses *reduce memory* dilakukan untuk mengurangi penggunaan memori sehingga data yang digunakan tidak mengambil waktu yang lama dalam pengolahan model . Data *transformation* digunakan untuk mengubah tipe data atau skala dataset sehingga data bisa fleksibel digunakan dalam model, hal ini dilakukan karena terdapat beberapa model yang sensitif terhadap data dengan tipe yang berbeda. Tahap terakhir dalam proses pra-pemrosesan data adalah mengidentifikasi dan menangani nilai yang hilang (*missing value*) pada setiap variabel. Penanganan dilakukan agar data yang tidak lengkap tidak mengganggu proses pelatihan model atau menurunkan akurasi prediksi. Penanganan yang biasa dillakukan adalah dengan menginput data dengan nilai *mean* atau modus.

2.3.5 *Split* Data

Tahap ini digunakan untuk membagi data menjadi tiga bagian, yaitu data latih (*training* set), data validasi (*validation set*), dan data uji (*test set*). Data latith digunakan untuk melatih model agar mengenali pola dalam data, sementara data validasi digunakan untuk menyesuaikan hyperparameter dan mengevaluasi performa model dengan bantuan Optuna dan *Time Series Cross-Validation* untuk menyesuaikan proses training menggunakan data latih dalam memprediksi data validasi, hal ini dilakukan agar parameter yang ditemukan mampu menghasilkan model yang baik dalam menangkap pola musiman dan tren yang muncul. Setelah model selesai dilatih dan disesuaikan, data latih digunakan sebagai evaluasi akhir untuk mengukur seberapa baik model dapat memprediksi data baru yang belum pernah dilihat sebelumnya. Sekitar satu tahun data harian disisihkan terlebih dahulu dari total lima tahun data historis yang diperoleh. Data tersebut dialokasikan sebagai test set yang setara dengan 20% dari keseluruhan data, dan sengaja tidak dilibatkan dalam proses pelatihan maupun tuning parameter. Sisa 80% data, digunakan dalam proses pelatihan dan validasi model, yang mana penentuan rasio antara data train dan data valid akan diuji menggunakan *Time Series Cross-Validation* (TSCV).

2.3.6 Implementasi Model

Penelitian ini menggunakan bahasa pemrograman *python* dengan berbagai *library* dalam untuk implementasi model LightGBM dan XGBoost. Model dibangun menggunakan parameter default dari penelitian sebelumnya, kemudian dilakukan optimasi hyperparameter untuk meningkatkan performa prediksi dengan tahap-tahapan sebagai berikut:

1. Penentuan Parameter Awal LightGBM dan XGBoost
   Tahap ini model LightGBM dan XGBoost akan dibangun menggunakan parameter awal yang mengacu pada penelitian sebelumnya atau nilai default dari masing-masing algoritma. Penentuan parameter awal ini bertujuan untuk membangun baseline performa model sebelum dilakukan optimasi lebih lanjut.

2. Training Model
   Setelah parameter awal ditentukan, model akan dilatih menggunakan data latih untuk mempelajari pola dari hubungan antara fitur dan target. Proses pelatihan ini bertujuan untuk memungkinkan model mengenali karakteristik data historis, sehingga dapat melakukan prediksi dengan lebih akurat. Evaluasi awal akan dilakukan menggunakan data validasi, yang digunakan untuk menilai apakah model sudah mampu melakukan generalisasi dengan baik atau masih mengalami overfitting/underfitting.

3. Optimasi Hyperparameter Tuning dengan Optuna
   Hyperparameter tuning dilakukan untuk meningkatkan performa model dengan mencari kombinasi parameter terbaik yang dapat mengoptimalkan hasil prediksi. Pada penelitian ini, digunakan Optuna sebagai alat untuk melakukan pencarian kombinasi hyperparameter terbaik pada kedua model algoritma yang digunakan. Proses tuning dilakukan dengan skema Time Series Cross-Validation (TSCV) untuk menjaga urutan kronologis data dan menangkap pola musiman atau tren jangka panjang yang mungkin ada dalam data historis. Dengan cara ini, model dapat secara dinamis menyesuaikan parameter dan risiko *overfitting* berkurang.

4. Hasil Prediksi Model Terbaik Menggunakan Data Uji
   Tahap ini dilakukan untuk memilih model terbaik setelah proses *training* dan optimasi *hyperparameter* terbaik diperoleh. Model terbaik yang terpilih akan digunakan untuk melakukan prediksi dengan data uji, data yang tidak pernah digunakan dalam proses pelatihan dan *hyperparameter tuning*, sehingga dapat memberikan gambaran objektif mengenai performa model dalam menghadapi data yang benar-benar baru.

2.3.7 Evaluasi Kinerja Model

EValuasi kinerja model mencakup perhitungan nilai error terhadap model terbaik yang telah dipilih. Tahap ini dilakukan dengan membandingkan nilai actual data uji dan hasil prediksi. Perbandingan evaluasi dilakukan menggunakan tiga metrik evaluasi utama, yaitu *Mean Absolute Percentage Error* (MAPE), *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), and *R-squared Score.*

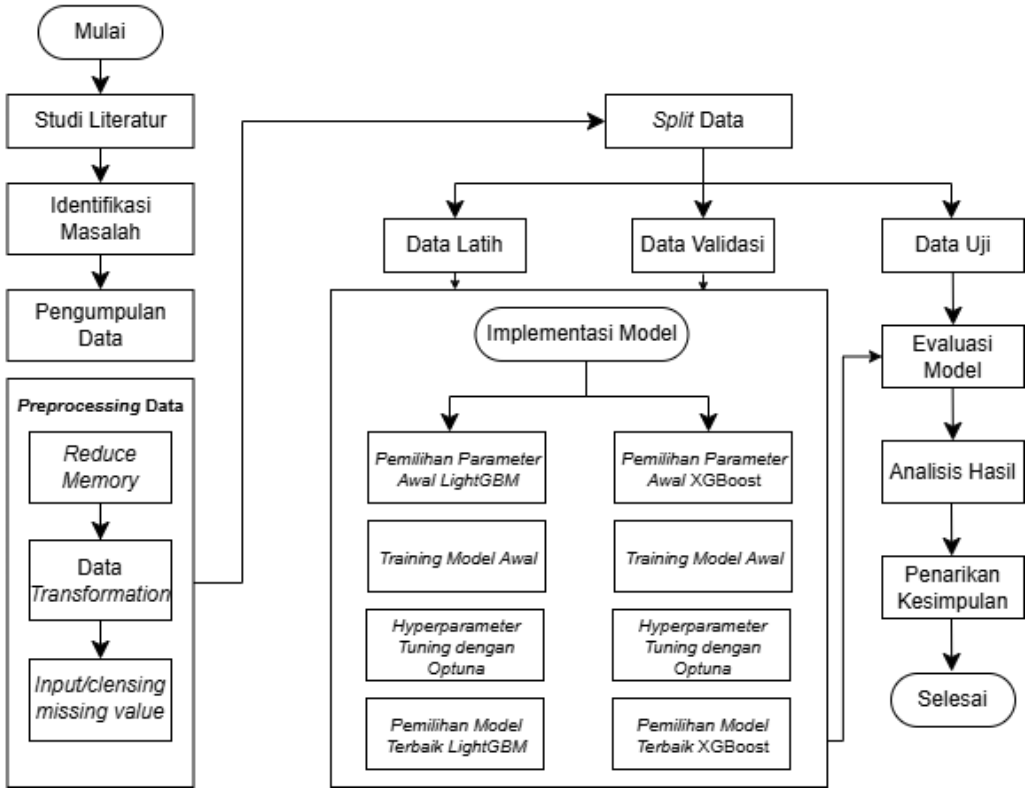
2.3.6 Analisis Hasil

Berdasarkan hasil evaluasi, dilakukan analisis terhadap kelebihan dan kekurangan pada masing-masing model. Hasil prediksi yang lebih baik dilihat dari nilai MAPE, MAE, dan RMSE yang lebih rendah, serta nilai *R-Squared Score* tinggi yang menjelaskan variasi data.

2.3.7 Menarik Kesimpulan

Akhir dari penelitian ini adalah dengan menarik Kesimpulan sebagai jawaban dari permasalahan penelitian. Kesimpulan yang diperoleh adalah algortima model prediksi mana yang menhasilkan tingkat akurasi terbaik.

2.4 Alur Penelitian



Gambar 2.7 Alur Penelitian