BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini mengacu pada beberapa topik penelitian serupa yang telah berhasil dilaksanakan. Tabel 2.1 berikut menyajikan beberapa penelitian terdahulu sebagai acuan dalam penelitian ini.

Tabel 2. 1 Penelitian Terdahulu

No	Nama Jurnal, Vol, No., Tahun	Judul Artikel, Penulis	Hasil
1	Jurnal Sistem Informasi (*Ultima Infosys*), 13, 01, 2022 [32]	"*Finding Features of Multiple Linear Regression on Currency Exchange Pairs",* Oetama R., Gaol F., Soewito B., Warnars H.	Mencari fitur yang paling berpengaruh pada hasil prediksi *forex* menggunakan MLR dan pendekatan CRISP-DM dengan hasil ketiga grup fitur yang dipilih memiliki hubungan kuat yang menghasilkan nilai *error* yang sangat rendah dengan nilai RMSE terendah adalah 0.00408 dan nilai R^2 adalah 0.99477.
2	Jurnal *Entropy,* 22, 08, 2020 [17]	"*Deep Learning for Stock Market Prediction",* Nabipour M., Nayyeri P., Jabani H., Mosavi A., Salwana, E., S. S.	Menguji performa model *berbasis tree* (Bagging, Decision Tree, Random Forest, Gradient Boosting, AdaBoost dan XGBoost) dan *neural networks* (ANN, RNN, dan LSTM) dengan hasil bahwa model LSTM memberikan performa paling baik dengan *error* paling kecil dan *fitting* paling baik (*Average* MAPE: 0.60, 1.18, 1.52 dan 0.54).
3	Jurnal *Applied Sciences*, 10, 02, 2020 [20]	"*Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading*", Ta V., Liu C., Tadesse D.	Menguji performa model LSTM dibandingkan dengan GRU, SVR, dan LR serta melakukan optimasi model dalam memprediksi harga saham dengan hasil bahwa model LSTM memberikan performa paling baik dengan *error* paling kecil berdasarkan MAE dan MSE.
4	Jurnal *Gênero & Direito,* 09, 02, 2020 [21]	"*Comparing the Prediction Accuracy of LSTM and ARIMA Models for Time-Series with Permanent Fluctuation*", Abdoli G., MehrAra M., Ardalani M.	Menguji performa model LSTM dan ARIMA dengan hasil bahwa model LSTM lebih unggul daripada model ARIMA dengan MAPE 10.04% dan MSE 6.334.744, sedangkan ARIMA memiliki MAPE 19.11% dan MSE 21.215.311.

No	Nama Jurnal, Vol, No., Tahun	Judul Artikel, Penulis	Hasil
6	Jurnal *Journal of Big Data,* 08, 104, 2021 [23]	"*Predicting LQ45 Financial Sector Indices Using RNN-LSTM",* Hansun S., Young J.	Menguji performa LSTM 3-*layer* pada data saham sektor finansial LQ45 dengan hasil bahwa model sederhana LSTM dapat memberikan performa yang baik dengan RMSE dan MAPE paling baik pada kode saham BMRI dengan nilai 739.1828 dan 18.6135.
7	Jurnal *Statistics, Optimization, and Information Computing,* 12, 01, 2024 [24]	"*Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, and Backtesting XGBoost Models*", Oukhouya H., Kadiri H., Himdi K., Guerbaz R.	Menguji performa model XGboost, LSTM, dan LSTM-XGBoost dalam memprediksi representasi harga saham dari 6 negara dengan hasil bahwa LSTM-XGBoost memiliki performa terbaik berdasarkan nilai MAE, MAPE, RMSE, dan R^2 paling baik adalah 25.513, 0.393, 34.349, dan 0.994.
8	Jurnal *Multimedia Systems,* 29, 03, 2023 [28]	"*A Graph-Based CNN-LSTM Stock Price Prediction Algorithm with Leading Indicators*", Wu J., Li Z., Herenscar N., Vo B., Lin J.	Menguji performa model LSTM tunggal, CNN tunggal, dan algoritma hibrida CNN-LSTM (SACLSTM) dalam memprediksi harga saham Taiwan dan Amerika dengan 3 *window* waktu dengan hasil bahwa SACLSTM memiliki performa terbaik berdasarkan nilai akurasi.
9	Jurnal *Financial Engineering and Risk Management,* 05, 07, 2022 [29]	"*Prediction Stock Price Based on CNN and LSTM Models*", Kexin H., Zhijing Z.	Menguji performa model LSTM dan CNN dalam memprediksi harga saham dengan hasil bahwa CNN memiliki performa yang lebih baik berdasarkan nilai RMSE, MAE, dan MAPE dengan nilai 0.010, 104.6630, dan 3.5933.
10	Jurnal *Complexity*, Vol. 2020, 2020 [30]	"*A CNN-LSTM Based Model to Forecast Stock Prices*", Lu W., Li J., Li Y., Sun A., Wang J.	Menguji performa model MLP, CNN, RNN, LSTM, CNN-RNN, dan CNN-LSTM dalam memprediksi harga saham dengan hasil bahwa CNN-LSTM dapat memberikan model prediksi dengan akurasi paling tinggi dengan nilai MAE, RMSE, dan R^2 adalah 27.564, 39.688, dan 0.946.
11	*Proceedings of The 2022 2nd International Conference on Economic Development and Business Culture (ICEDBC 2022)*, 2022 [31]	"*Stock Price Prediction based on CNN-LSTM Model in the PyTorch Environment*", Xu W.	Menguji performa model LSTM, CNN, dan CNN-LSTM dalam memprediksi harga saham dengan hasil bahwa CNN-LSTM merefleksikan hasil prediksi yang paling baik, hasil yang konsisten dengan nilai tren aslinya, dan tingkat kesalahan yang kecil dengan MSE < 0.002.

No	Nama Jurnal, Vol, No., Tahun	Judul Artikel, Penulis	Hasil
12	Jurnal *Latex Class File,* 61, 14, 08, 2021 [25]	"*Attention-Based CNN-LSTM and XGBoost Hybrid Model for Stock Prediction*", Shi Z., Hu Y., Mo G., Wu J.	Menguji performa model ARIMA, LSTM, CNN-LSTM, dan penerapan XGBoost dalam memprediksi harga saham dengan hasil bahwa CNN-LSTM + XGBoost merupakan model yang paling baik dengan nilai MSE, RMSE, MAE, dan R^2 adalah 0.00020, 0.01424, 0.01126, dan 0.88342.

Berdasarkan Tabel 2.1 Penelitian Terdahulu, penelitian-penelitian tersebut menggunakan algoritma *deep learning* dalam melakukan prediksi data yang bersifat *time-series*. Penelitian-penelitian ini menggunakan perbandingan atau penilaian berdasarkan matriks evaluasi seperti MSE, RMSE, MAE, dan MAPE. Berdasarkan enam penelitian pada Tabel 2.1, ditemukan bahwa algoritma *Long Short-Term Memory* (LSTM) dapat memberikan performa model prediksi yang baik untuk data yang bersifat *time-series*, yang juga merupakan sifat data yang digunakan pada penelitian ini [17], [20], [21], [22], [23], [24]. Temuan-temuan ini dijadikan acuan penggunaan model LSTM sebagai model prediksi harga saham pada sektor infrastruktur.

Selain itu, satu empat penelitian lainnya pada Tabel 2.1 telah meningkatkan hasil prediksi menggunakan model hibrida CNN-LSTM [25], [28], [30], [31]. Berdasarkan temuan ini, penelitian ini juga menguji performa CNN-LSTM sebagai upaya untuk meningkatkan hasil prediksi model *Long Short-Term Memory* (LSTM) dan *Convolutional Neural Network* (CNN). Model CNN memiliki kemampuan dalam mengekstrak fitur efektif dari data yang kemudian akan digabungkan dengan kemampuan LSTM yang dapat menemukan ketergantungan data dalam data *time-series*, metode ini efektif meningkatkan akurasi prediksi harga saham. Penelitian [29] pun menemukan bahwa CNN memiliki performa yang lebih baik daripada LSTM dalam memprediksi harga saham, hal ini menguatkan alasan digunakannya CNN untuk dilakukan hibrida dengan LSTM. Model CNN-LSTM menggunakan CNN untuk mengekstrak fitur dari data waktu masukan dan menggunakan LSTM untuk memprediksi

harga penutupan saham pada hari berikutnya [30]. Penelitian ini juga menerapkan algoritma *ensemble* XGBoost karena kemampuannya dalam meningkatkan kemampuan prediksi dan generalisasi sebagaimana diterapkan pada penelitian [24], [25].

Pada penelitian ini terdapat pembeda dari penelitian-penelitian terdahulu yang dijadikan pedoman pada penelitian ini. Penelitian ini mengoptimasi performa model tunggal LSTM dan CNN. Model tunggal dilakukan *hyperparameter tuning* menggunakan *Grid Search* untuk mendapatkan kombinasi *hyperparameter* terbaik. Selain itu, dilakukan penerapan algoritma *ensemble* XGBoost dan hibrida kedua model tunggal sebagai upaya untuk mengoptimasi hasil model. Hasil performa model kemudian dievaluasi menggunakan matriks evaluasi RMSE, MSE, MAE, dan MAPE.

2.2 Tinjauan Teori

2.2.1 Saham

Saham, sebagaimana diartikan dalam Kamus Besar Bahasa Indonesia (KBBI), merupakan bukti kepemilikan bagian modal perseroan terbatas yang memberikan hak kepada pemegang saham atas dividen dan sebagainya dalam suatu perusahaan [33]. Apabila memiliki saham suatu organisasi, seseorang dikatakan sebagai pemegang saham dan berhak dalam mendapatkan sebagian keuntungan perusahaan (dividen) dan memberikan hak suaranya [34]. Berinvestasi saham dapat dilakukan di mana saja dengan modal yang relatif kecil, fleksibilitas waktu, dan keuntungan yang terbatas [35].

2.2.2 *Machine Learning*

Machine learning didefinisikan sebagai kemampuan atau kapasitas suatu sistem dalam mempelajari data guna mengotomatisasi proses pembuatan model analitis dan memecahkan masalah [36]. *Machine learning* melibatkan peningkatan kinerja program komputer dari dari berbagai disiplin ilmu seperti statistika, teori probabilitas, teori aproksimasi, dan disiplin lainnya [37]. *Machine learning*

menyesuaikan model menggunakan data latih atau keadaan historis untuk membuat prediksi di masa depan atau memperoleh pengetahuan dari data, hal ini membantu dalam menghasilkan keputusan yang dapat diandalkan dan konsisten [38].

Algoritma *machine learning* digunakan untuk melatih model dalam tiga jenis pembelajaran utama, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. *Supervised Learning* menggunakan dataset yang sudah diberi label untuk membentuk model perilaku, sementara *Unsupervised Learning* bekerja tanpa label, mengidentifikasi struktur tersembunyi dalam dataset. *Reinforcement Learning*, di sisi lain, adalah metode pembelajaran yang tidak memerlukan pengetahuan sebelumnya, mengandalkan interaksi mandiri dengan lingkungan untuk memperoleh pengetahuan melalui percobaan-percobaan [39], [40].

2.2.3 *Deep Learning*

Deep learning merupakan algoritma yang menggunakan jaringan syaraf tiruan sebagai arsitektur untuk memahami dan mempelajari data, yang berasal dari penelitian jaringan syaraf tiruan dengan *multilayer perceptron* sebagai strukturnya [41]. Lapisan-lapisan tersebut terdiri dari sejumlah simpul, di mana setiap simpul berfungsi sebagai tempat untuk melakukan perhitungan. Dalam jaringan saraf, terdapat tiga jenis lapisan utama. *Input layer* berfungsi untuk berkomunikasi dengan lingkungan luar dan menentukan aturan dasar untuk melatih jaringan. *Output layer* menyajikan hasil dari informasi yang dimasukkan ke dalam jaringan ke lingkungan luar. Terakhir, *hidden layer* yang berada di antara *input* dan *output layer*. Lapisan ini berisi neuron-neuron kecil yang membantu mentransfer data dan pelatihan ke lapisan-lapisan lain di dalam jaringan [42].

2.2.4 *Missing Values*

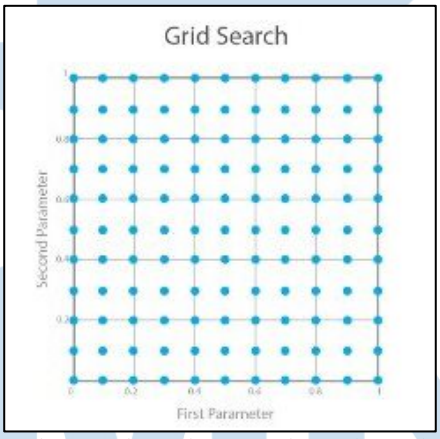
Missing values merupakan suatu isu pada pengolahan data yang sangat umum terjadi karena *human-error*, kurangnya observasi data, kendala teknis, kendala privasi, dan sebagainya [43]. Data yang hilang atau tidak ada dapat berada pada tabular atau data *time* series [43]. Semakin banyak *missing values* atau data *null* yang ada, maka *machine learning* akan mengalami kesulitan saat melakukan pelatihan untuk mendapatkan hasil performa yang baik [44], [45]. *Missing values* pada *data frame* atau *series* dapat diidentifikasi dengan fungsi *isna()* atau untuk data *null* menggunakan fungsi *isnull()* [46]. Penghapusan baris atau kolom yang memiliki *missing values* atau data *null* dapat dilakukan menggunakan fungsi *dropna()* [47].

2.2.5 *Hyperparameter Tuning*

Hyperparameter pada algoritma *machine learning* adalah parameter yang nilai-nilainya mengatur proses pembelajaran dalam model *machine learning* yang juga menentukan parameter-parameter final untuk model tersebut [48]. *Hyperparameter* merupakan parameter eksternal yang tidak dapat diprediksi dari data juga bukan merupakan bagian dari model, namun dapat disesuaikan sehingga akurasi model yang diinginkan tercapai. Adapun beberapa *hyperparameter* yang ada antara lain adalah *hidden layers, batch size, learning rate,* dan *optimizer* [49]. Agar mendapatkan hasil yang baik dari model, *hyperparameter* harus diatur dengan pengaturan yang paling sesuai dengan melakukan optimasi *hyperparameter* atau *hyperparameter tuning* [50]. Terdapat dua pendekatan dalam upaya optimasi *hyperparameter,* yaitu secara *manual search* dan *automatic search*. Pendekatan *manual search* dilakukan dengan menyesuaikan dan menguji pengaturan *hyperparameter* secara manual yang bergantung pada intuisi dan keahlian pengguna atau *developer,* sedangkan pendekatan *automatic search* seperti *Grid Search* dan *Random Search,* dilakukan secara otomatis [51].

2.2.5.1 *Grid Search*

Grid Search melatih model dengan seluruh kemungkinan *hyperparameter* dan nilai-nilainya, kemudian mengevaluasi kinerjanya, lalu menghasilkan *hyperparameter* dengan hasil terbaik [51]. *Grid Search* bekerja dengan melakukan pencarian secara keseluruhan pada ruang *hyperparameter* yang ditentukan. Ruang *hyperparameter* perlu ditentukan dengan alasan nilai parameter pada *machine learning* merupakan nilai nyata dan sangatlah tidak terbatas pada beberapa parameter. *Grid Search* terbatas dalam melakukan pencarian dengan dimensi ruang yang besar, namun dapat diparalelkan dengan mudah karena algoritmanya umumnya bekerja secara independen terhadap *hyperparameter*. Cara kerja *Grid Search* diilustrasikan pada Gambar 2.1 [52].

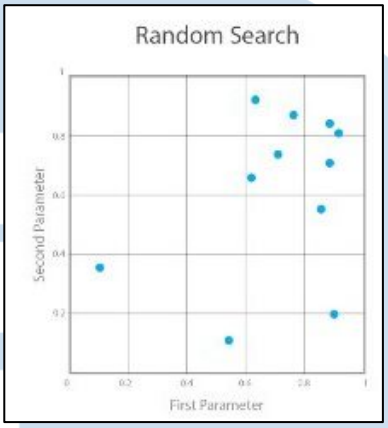


Gambar 2. 1 Ilustrasi Cara Kerja *Grid Search*
Sumber: [52]

2.2.5.2 *Random Search*

Random Search mencoba berbagai kemungkinan nilai secara acak untuk membatasi pemeriksaan pada *hyperparameter* yang tidak penting, sehingga dapat meningkatkan efisiensi keseluruhan dan mencapai optimasi model [51]. Daripada melakukan seluruh posibilitas kombinasi, *Random Search* memilih kombinasinya secara acak, kemudian *hyperparameter*-

nya dievaluasi dan hasil terbaiknya diambil [52], [53]. Cara kerja *Random Search* diilustrasikan pada Gambar 2.2 [52].



Gambar 2. 2 Ilustrasi Cara Kerja *Random Search*
Sumber: [52]

2.2.6 *MinMax Scaling*

Pada tahap preparasi data, normalisasi data penting untuk dilakukan guna mencegah model *machine learning* memiliki dependensi yang salah pada data, yang menyebabkan kinerjanya berkurang [54]. Tanpa merubah perbedaan dalam rentang nilai, teknik *data scaling* atau normalisasi mengubah nilai numerik dalam dataset ke dalam skala yang seragam [55]. Pada kasus-kasus tertentu, penting untuk melakukan *scaling* atau normalisasi data agar kebutuhan algoritma terpenuhi atau memastikan data memiliki kisaran yang merata [47]. *MinMax Scaling* merupakan salah satu teknik dalam *scaling* data pada tahap preparasi data. *MinMax Scaling* berfungsi untuk memastikan seluruh fitur pada nilai terletak pada rentang 0 hingga satu. Fungsi ini cocok digunakan apabila distribusi fiturnya memiliki nilai minimum dan maksimum yang jelas [56]. Persamaan fungsi *MinMax Scaling* didefinisikan sebagai berikut: [56]

$$X_{Scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (2.1)$$

Rumus 2. 1 Rumus Fungsi *MinMax Scaling*

Keterangan notasi persamaan: [56]

- $X = nilai\ asli\ fitur$

- $X_{Scaled} = nilai\ fitur\ skala$
- $X_{min} = nilai\ fitur\ minimum\ pada\ data$
- $X_{max} = nilai\ fitur\ maksimum\ pada\ data$

2.2.7 *Activation*

Activation merupakan suatu fungsi yang diterapkan untuk membuat keputusan suatu akan diaktifkannya suatu *node* pada model *neural network.* Penerapan *activation* memiliki peran penting dalam *neural network* sebab kemampuannya mempengaruhi data output satu lapisan *neuron* sebelum ditransfer sebagai input ke lapisan selanjutnya. Adapun beberapa jenis *activation* yang umum digunakan pada model *neural network* sebagai berikut: [57], [58]

1. *Linear*

   Linear activation merupakan fungsi aktivasi yang nilai gradiennya berbanding lurus dengan masukan yang diberikan pada model jaringan saraf. Fungsi ini menghitung bobot total semua input dan ditentukan oleh persamaan berikut: [59]

$$f(x) = x \qquad (2.2)$$

Rumus 2. 2 Rumus *Linear Activation*

2. *Sigmoid*

   Sigmoid activation merupakan jenis fungsi aktivasi dengan rentang fungsi dari 0 hingga 1. Fungsi ini utamanya mengambil nilai *real* yang digunakan untuk memprediksi probabilitas sebagai output. Kekurangan utama dari fungsi sigmoid adalah terbatas dalam rentang antara 0 dan 1. Oleh karena itu, fungsi ini selalu menghasilkan nilai non-negatif sebagai output. Persamaan fungsi *sigmoid activation* didefinisikan sebagai berikut: [57]

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2.3)$$

Rumus 2. 3 Rumus *Sigmoid Activation*

3. *Tanh*

Tanh activation, atau *hyperbolic tangent* merupakan solusi terhadap kelemahan *sigmoid function* dengan memberikan performa *training* yang lebih baik untuk *multi-layer neural network.* Fungsi *tanh* bersifat kontinu, dapat dibedakan, dan terbatas, dengan rentang antara -1 dan 1. Ini memperluas keluaran yang mungkin, termasuk nilai negatif, positif, dan nol. Keunikan lainnya adalah fungsi tanh berpusat pada nol, mengurangi jumlah epoch yang diperlukan untuk melatih jaringan dibandingkan dengan fungsi sigmoid. Kekurangannya adalah nilai gradien yang mendekati nol sehingga mengalami masalah *vanishing gradient*. Persamaan fungsi *tanh activation* didefinisikan sebagai berikut: [57]

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \qquad (2.4)$$

Rumus 2. 4 Rumus *Tanh Activation*

4. *ReLU*

ReLU (Rectified Linear Unit) activation merupakan jenis aktivasi yang berperan sebagai identitas untuk nilai input positif dan menghasilkan 0 untuk nilai input yang negatif. *ReLU* mengungguli *sigmoid* dan *tanh* karena tidak berpusat pada nol dan memberikan kinerja dan generalisasi yang lebih baik. Neuron dengan bobot negatif dievaluasi menjadi output nol, yang mengakibatkan neuron dengan bobot negatif tidak memberikan kontribusi signifikan pada kinerja keseluruhan jaringan, menghindari masalah neuron mati yang terlihat sebelumnya. Persamaan fungsi *ReLU activation* didefinisikan sebagai berikut: [57]

$$f(x) = \max(0, x) \tag{2.5}$$

Rumus 2. 5 Rumus *ReLU Activation*

2.2.8 *Optimizer*

Optimizer merupakan suatu algoritma yang dapat meningkatkan efisiensi proses pembelajaran model algoritma dengan meningkatkan akurasinya [60]. *Optimizer* membentuk model yang paling akurat dengan mencari parameter optimal dan merubah parameter sedemikian mungkin untuk meminimalisir *loss* atau *cost function* [61]. Adapun beberapa *optimizer* yang umum digunakan antara lain: [61], [62]

1. *Stochastic Gradient Descent* (SGD)

   Stochastic Gradient Descent (SGD) merupakan salah satu jenis dari metode *optimizer gradient descent* yang bekerja dengan meminimalisir error melalui pertimbangan error dari setiap poin datanya pada setiap iterasinya [63]. Singkatnya, SGD bekerja dengan mengikuti gradien *minibatch* yang secara acak dipilih ke bawah [62]. Persamaan SGD dituliskan sebagai berikut: [62]

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \tag{2.6}$$

Rumus 2. 6 Rumus SGD

2. *Adagrad*

   Adagrad adalah algoritma modifikasi dari *Stochastic Gradient Descent* (SGD) yang mengoptimalkan berdasarkan gradien, menyesuaikan learning rate untuk setiap parameter [62]. *Optimizer* ini memanfaatkan *learning rate* yang berbeda untuk setiap parameter setiap kali, dengan menggunakan gradien yang telah dihitung sebelumnya, sehingga menghilangkan kebutuhan penyesuaian *learning rate* saat pelatihan [64]. Persamaan Adagrad dituliskan sebagai berikut: [62]

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_\theta L(\theta_t) \quad (2.7)$$

Rumus 2. 7 Rumus *Adagrad*

3. *Adadelta*

Adadelta merupakan perkembangan dari *Adagrad* yang menyesuaikan *learning rate* berdasarkan jendela pergerakan gradien daripada mengakumulasi jumlah gradien kuadrat sepanjang waktu. Berbeda dengan *Adagrad*, *Adadelta* tidak perlu menentukan *learning rate* di awal [63]. Penyesuaian nilai yang lebih rendah disediakan oleh *Adadelta* untuk parameter yang sering diubah, sedangkan penyesuaian yang lebih besar disediakan untuk parameter yang jarang diperbarui [65]. Persamaan Adadelta dituliskan sebagai berikut: [62]

$$\theta_{t+1} = \theta_t - \frac{\Delta\theta_t}{\sqrt{E[\Delta\theta_t^2] + \epsilon}} \quad (2.8)$$

Rumus 2. 8 Rumus Adadelta

4. *Root Mean Square Propagation* (*RMSprop*)

Root Mean Square Propagation (*RMSprop*) juga merupakan perkembangan dari algoritma *Adagrad* yang dibuat dengan performa lebih baik dalam pengaturan non konveks dengan mengganti akumulasi gradien menjadi *weighted moving average*. Berbeda dengan *adagrad*, *RMSProp* cepat konvergen dengan mengabaikan sejarah gradien kuadrat yang sangat lama setelah menemukan cekungan cembung [62], [66]. Persamaan RMSprop dituliskan sebagai berikut: [62]

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g_t^2] + \epsilon}} \nabla_\theta L(\theta_t) \quad (2.9)$$

Rumus 2. 9 Rumus RMSprop

5. *Adaptive Moment Estimation* (*Adam*)

Adaptive Moment Estimation (*Adam*) adalah satu algoritma optimasi yang paling banyak digunakan dalam *deep learning*. *Adam* menghitung tingkat pembelajaran adaptif untuk setiap parameter berdasarkan perkiraan momen pertama dan kedua dari gradien. *Adam* menggabungkan kelebihan dari *Adagrad* yang efektif untuk gradien yang jarang (*sparse*) dan *RMSProp* yang baik dalam pengaturan online dan non-stasioner [62]. Algoritma ini menggabungkan metode SGD dari *RMSProp* untuk menyesuaikan *learning rate* dan *weighted moving average* dengan memanfaatkan momentum [64]. Dalam implementasi modelnya, *Adam* memiliki kemampuan untuk secara otomatis memperbarui bobot dan *learning rate* [67]. Persamaan Adam dituliskan sebagai berikut: [62]

$$\theta_{t+1} = \theta_t - \eta \frac{\frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1^t}}{\sqrt{\frac{\beta_2 v_{t-1} + (1-\beta_2) g_t^2}{1-\beta_2^t}} + \epsilon} \qquad (2.10)$$

Rumus 2. 10 Rumus Adam

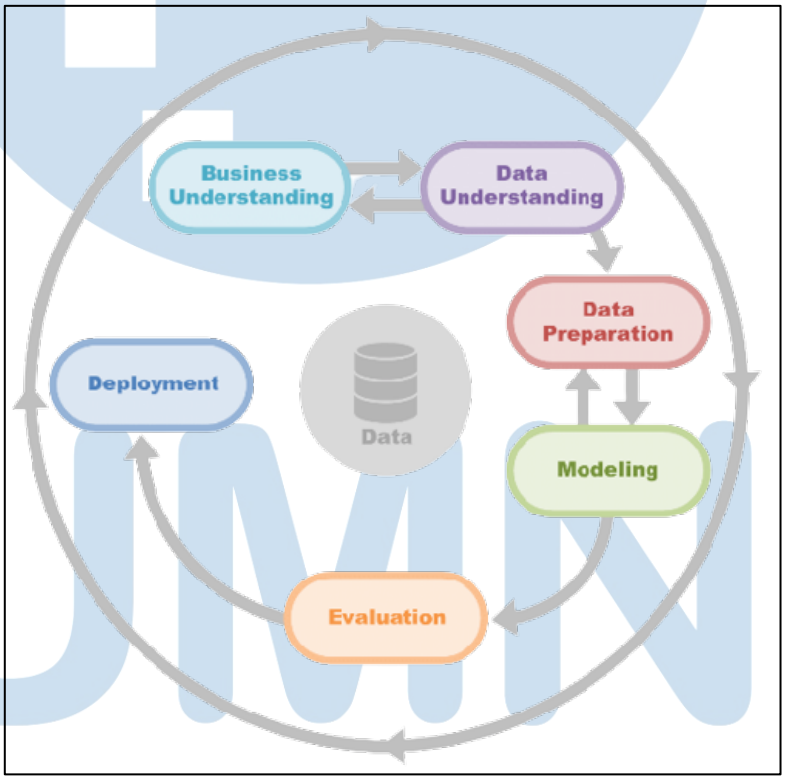
Keterangan notasi:

- θ = model
- $\eta = learning\ rate$
- $\nabla_\theta L(\theta_t) =$ $gradien\ loss\ function\ L\ terhadap\ parameter\ \theta$
- $G_t = jumlah\ kumulatif\ \ gradien\ kuadrat$
- $\epsilon = nilai\ kecil\ untuk\ stabilitas\ numerik$
- $E[g_t^2] = moving\ average\ gradien\ kuadrat$
- $\Delta\theta_t = estimasi\ perubahan\ parameter$
- $\beta_1\ dan\ \beta_2 =$ $estimasi\ pengurangan\ momen\ pertama\ dan\ kedua$

2.3 *Framework*, Algoritma, dan Metode Evaluasi

2.3.1 *Framework* CRISP-DM

Cross-Industry Standard Process model for Data Mining (CRISP-DM) merupakan sebuah model dalam proses *data mining* yang menggambarkan *lifecycle* pada proses *data mining* [68]. CRISP-DM telah dianggap sebagai standar *de facto* dalam penemuan-penemuan *data mining* [69]. Penggunaan CRISP-DM dapat memberikan banyak keuntungan dalam tahap *data preparation, data modeling,* dan *evaluation* [70], serta memberikan efisiensi biaya dan waktu [68]. CRISP-DM memiliki enam fase iteratif sebagai berikut: [71], [72], [73], [74]



Gambar 2. 3 CRISP-DM *Framework*
Sumber: [73]

a. *Business Understanding*

Business Understanding merupakan fase dalam memahami tujuan dan keperluan proyek data dari segi perspektif bisnis, yang kemudian menghasilkan rumusan masalah *data mining* serta rencana pencapaian tujuan bisnis tersebut. Pada fase ini, situasi bisnis harus ditelaah untuk

mendapatkan gambaran mengenai ketersediaan sumber daya. Selain itu, tujuan dari *data mining* ditetapkan pada fase ini [71], [72], [73], [74].

b. *Data Understanding*

Data Understanding merupakan fase setelah *business understanding.* Pada fase ini dilakukan pengumpulan data, kemudian data tersebut dieksplorasi, dideskripsikan, serta dipersiapkan kualitas dan keutuhan data. Pendeskripsian data pada fase ini dilakukan menggunakan analisis statistik dan penentuan atribut [71], [72], [73], [74].

c. *Data Preparation*

Data preparation merupakan fase lanjutan dari fase *data understanding*, di mana pengolahan data dilakukan. Fase ini meliputi seluruh kegiatan dalam pembangunan dataset sebelum diterapkan pada fase *modeling.* Data yang ditemukan kualitas yang buruk serta tidak utuh akan dilakukan pembersihan, seperti penghilangan data *null* atau data yang redundan. Metode yang diterapkan pada fase ini akan bergantung pada model penelitian yang akan dilakukan. Fase ini dapat dilakukan berulang dan tidak harus dilakukan secara runtut [71], [72], [73], [74].

d. *Modeling*

Modeling merupakan fase di mana teknik model dan pengujian model dipilih. Pemilihan model akan bergantung pada masalah bisnis atau data yang ada juga penentuan parameter yang digunakan, sehingga dapat memberikan hasil yang optimal. Setelah model yang tepat diterapkan, akan dilakukan fase selanjutnya yaitu *evaluation* untuk memilih model terbaik [71], [72], [73], [74].

e. *Evaluation*

Evaluation merupakan fase lanjutan setelah pemodelan dilakukan. Fase ini memeriksa hasil untuk memastikan tidak ada fase yang terlewat sehingga tetap berada pada jalur tujuan bisnis yang sudah ditetapkan. Selain itu, fase ini juga memastikan model yang digunakan

sudah tepat untuk dilanjutkan ke fase *deployment* [71], [72], [73], [74].

f. *Deployment*

Deployment merupakan fase terakhir pada CRISP-DM di mana pengetahuan yang dihasilkan akan disajikan sedemikian rupa agar dapat digunakan. Model akan diimplementasikan secara langsung dalam proses pengambilan keputusan organisasi atau penggunanya. Fase ini dapat diaplikasikan menjadi aplikasi website, sistem, laporan, dan sebagainya yang dapat disesuaikan dengan kebutuhan objektif bisnis [71], [72], [73], [74].

2.3.2 *Framework* KDD

Framework KDD atau *Knowledge Discovery in Database* digunakan atas adanya kebutuhan tinggi dari analisis data [75]. KDD merupakan salah satu proses *data mining* yang mencari pengetahuan (atau ekstraksi informasi dari data) menggunakan metode tingkat tinggi [76]. KDD digunakan dalam menemukan pola data yang valid, baru, berguna, sehingga pada akhirnya dapat dimengerti [75]. KDD menggabungkan alat, teknologi, dan metodologi sebagai upaya mengidentifikasi pola dan hubungan dalam data yang dapat dimanfaatkan untuk menghasilkan wawasan baru dan meningkatkan pengambilan keputusan [75]. KDD memiliki 7 tahapan, antara lain adalah *pre-kdd, selection, preprocessing, transformation, interpretation/evaluation,* dan *post-kdd* [77].

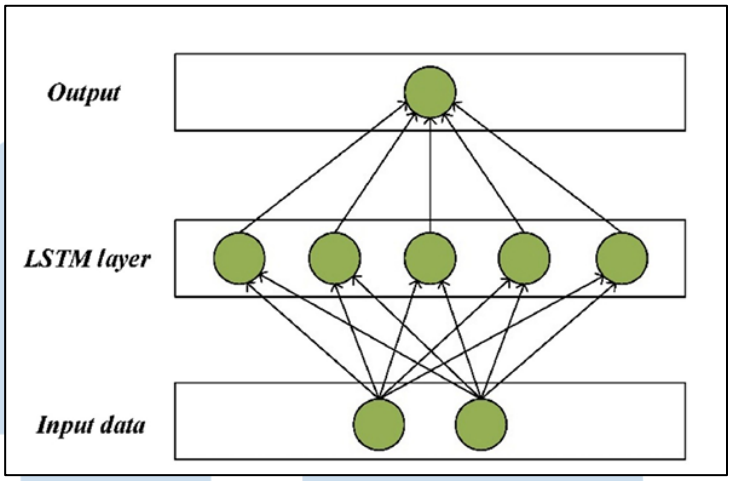
2.3.3 *Framework* SEMMA

Framework SEMMA merupakan salah satu metode *data mining* yang dikembangkan oleh SAS Institute pada 2005 yang berdasar pada KDD. Berbeda dengan KDD, *framework* ini dikhususkan untuk satu penyedia, sehingga tidak dapat digunakan di semua situasi. SEMMA tidak meliputi dua tahap dari proses KDD yang tergolong krusial, yakni *pre-kdd* dan *post-kdd*. Adapun SEMMA terdiri atas 5 tahapan, antara lain adalah *Sample, Explore, Modify, Model,* dan *Asses* [77].

2.3.4 Algoritma

2.3.4.1 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) adalah variasi dari sel memori *Recurrent Neural Network* (RNN) yang kerap digunakan dalam berbagai konteks. Pengembangannya dimulai pada akhir tahun 90-an oleh Sepp Hochreiter dan Jurgen Schmidhuber. LSTM dikenal memiliki kemampuan yang cukup baik dalam membangun model prediktif. Sebagai turunan dari RNN, LSTM dirancang khusus untuk memproses data sekuensial [78]. LSTM dapat menyelesaikan isu mengenai ledakan dan hilangnya gradien pada RNN. Oleh karena kapasitas memori bawaannya dan kemampuannya dalam menghasilkan prediksi yang relatif akurat, LSTM telah diterapkan pada cakupan yang luas seperti analisis teks, analisis emosi, dan *speech recognition* [30]. Adapun sel memori LSTM terdiri atas tiga elemen: *forget gate*, *input gate,* dan *output gate. Forget gate* memiliki fungsi untuk mengubah skala dari state internal suatu sel sebelum menambahkannya sebagai masukan ke sel tersebut melalui koneksi rekurensi. Hal ini memungkinkan *forget gate* untuk adaptif melakukan *forgetting* atau *resetting* memori sel. *Input gate*, di sisi lain, bertugas mengontrol sinyal-sinyal dari jaringan ke dalam sel memori dengan penyesuaian skala yang tepat. Saat *gate* ini ditutup, aktivasi mendekati nol. Sementara itu, *output gate* mempelajari cara mengatur akses ke konten dalam sel memori, berfungsi sebagai pengaman untuk melindungi memori sel dari gangguan yang tidak diinginkan [30], [79].



Gambar 2. 4 Gambaran Model LSTM

Sumber: [79]

Adapun persamaan kalkulasi LSTM dijelaskan sebagai berikut: [79]

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2.11)$$

Rumus 2. 11 Rumus *Forget Gate*

$$i_t = \sigma(W_f x_t + U_i h_{t-1} + b_i) \quad (2.12)$$

Rumus 2. 12 Rumus *Memory Gate*

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (2.13)$$

Rumus 2. 13 Rumus *Temporary Cell State*

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t) \quad (2.14)$$

Rumus 2. 14 Rumus *Current Cell State*

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (2.15)$$

Rumus 2. 15 Rumus *Output Gate*

$$h_t = o_t * \tanh(C_t) \quad (2.16)$$

Rumus 2. 16 Rumus *Hidden State*

Keterangan notasi persamaan: [79]

- $W = input\ weights$
- $U = recurrent\ weights$
- $b = bias$
- $f = forget\ gate$
- $i = input\ gate$
- $o = output\ gate$

- $\sigma = fungsi\ sigmoid$

Rumus 2.11 menjelaskan bagaimana *forget gate*, yang disimbolkan sebagai f_t, menentukan informasi apa yang perlu dihapus dari *cell state* sebelumnya. Selanjutnya, persamaan Rumus 2.12 dan Rumus 2.13 mengilustrasikan langkah-langkah identifikasi pada *input gate* (i_t), di mana informasi dari input x_t dinilai apakah seharusnya disimpan pada *cell state* C_t dan kandidat *cell state* $\tilde{C}_t$ harus diperbarui. Rumus 2.14 menjelaskan proses pembaruan *cell state* pada langkah waktu saat ini (C_t), yang melibatkan penggabungan antara kandidat memori $\tilde{C}_t$ dan *long-term memory* C_{t-1}. Sementara itu, persamaan Rumus 2.15 dan Rumus 2.16 menegaskan hasilnya pada *output gate* (o_t), di mana informasi output h_t dan *cell state* C_t digunakan dalam mempertimbangkan hasil dari *forget gates* [30], [79].

2.3.4.2 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan model yang diusung oleh Lecun dkk. pada tahun 1998. CNN merupakan suatu bentuk jaringan saraf bersifat *feedforward* yang telah dikenal efektif dalam penerapan prediksi data deret waktu. CNN unggul pada kemampuan persepsi lokal dan pembagian bobot, di mana dapat mengurangi jumlah parameter dan meningkatkan efisiensi pembelajaran model dengan signifikan. Selain itu, CNN dibagi menjadi tiga lapisan (*layer*) utama, yaitu *convolutional layer, pooling layer,* dan *fully connected layer* [80]. Pertama, *convolutional layer* digunakan untuk mengekstrak fitur-fitur penting dari data input, seperti pola atau karakteristik khusus, yang akan digunakan dalam proses pelatihan model. Kemudian, *pooling layer* berperan dalam menyederhanakan representasi data dengan menciptakan filter baru berdasarkan aturan tertentu,

membantu mengurangi dimensi dan kompleksitas data. Terakhir, *fully connected layer* (*dense layer*), yang pada dasarnya mirip dengan *Multilayer Perceptron* (MLP), menggabungkan dan memproses fitur-fitur yang telah diekstrak sebelumnya untuk membuat prediksi atau output akhir [41], [81], [82].

Penelitian ini melakukan *tuning* terdahap beberapa parameter CNN. Pertama, *filters* yang berfungsi untuk mengidentifikasi pola pada data. Jumlah *filter* menentukan kedalaman fitur yang ditangkap dari data input. Kedua, *kernel size* yang berfungsi sebagai penangkap ketergantungan jangka pendek. Ukuran *kernel* menentukan Panjang *filter* dalam menangkap pola. Ketiga, *pool size* yang berfungsi untuk mengurangi dimensi spasial untuk membantu mengurangi kompleksitas komputasi. Ukuran *pool* menentukan ukuran pengurangan dimensi. Keempat, *dense units* digunakan untuk mengintegrasi fitur yang terekstraksi dan menciptakan prediksi final. Jumlah *dense units* menentukan kemampuan jaringan dalam menangkap pola [41]. Persamaan CNN berdasarkan penelitian ini dijelaskan sebagai berikut: [28], [41]

1. *Input layer*: tensor tiga dimensi dengan n sebagai ukuran sampel (*batch size*), t jumlah *time step*, dan f jumlah fitur.
2. *Convolution layer* (*Conv1D*) dengan aktivasi ReLU: terdiri atas $W^{(l)}$ sebagai *filter* konvolusi, $b^{(l)}$ sebagai bias, x_i adalah input pada *time step I,* dan $*$ sebagai simbol operasi konvolusi.

$$h_i^{(l)} = ReLU\left(W^{(l)} * x_i + b^{(l)}\right) \quad (2.17)$$

Rumus 2. 17 Rumus Lapisan *Conv1D*

3. *Maxpooling layer* (*Maxpooling1D*): mengurangi dimensi dengan mengambil nilai maksimum pada jendela *pooling* dengan *k* adalah *pool size.*

$$p_i^{(l)} = max\left(h_{i:i+k}^{(l)}\right) \quad (2.18)$$

Rumus 2. 18 Rumus *Maxpooling1D*

4. *Flatten layer*: mengkonversi fitur menjadi bentuk satu dimensi dengan mengatur elemen *h* menjadi vektor tunggal.

$$flatten(h) = vector(h) \quad (2.19)$$

Rumus 2. 19 Rumus Lapisan *Flatten*

5. *Dense layer* dengan aktivasi ReLU: mengintegrasi fitur yang telah diekstrak dan membuat keluaran prediksi dengan $W^{(d)}$ sebagai *weight* dan $b^{(d)}$ sebagai bias pada *dense layer.*

$$o^{(d)} = ReLU\left(W^{(d)} \cdot flatten(h) + b^{(d)}\right) \quad (2.20)$$

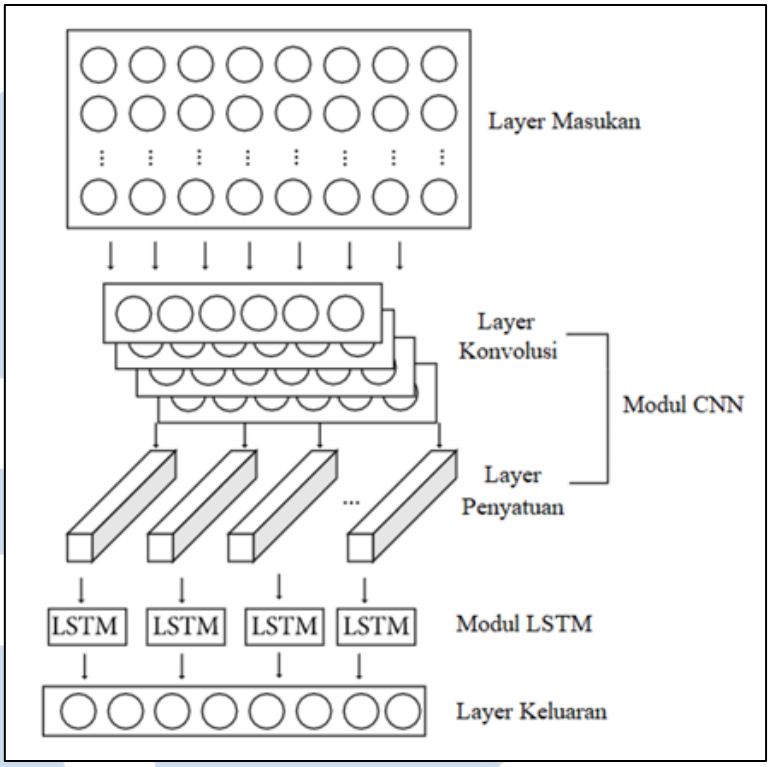
Rumus 2. 20 Rumus Lapisan *Dense*

6. *Output layer:* membuat keluaran prediksi dengan $W^{(o)}$ sebagai *weight* dan $b^{(o)}$ sebagai bias pada *output layer.*

$$\hat{y} = W^{(o)} \cdot o^{(d)} + b^{(o)} \quad (2.21)$$

Rumus 2. 21 Rumus Lapisan *Output*

2.3.4.3 *Algoritma Hibrida CNN-LSTM*



Gambar 2. 5 Struktur CNN-LSTM untuk Prediksi Deret Waktu
Sumber: [83]

Algoritma hibrida CNN-LSTM merupakan algoritma gabungan antara *Convolutional Neural Networks* (CNN) dan *Long-Short Term Memory* (LSTM). CNN berperan sebagai pengekstrak informasi spasial dari data input, sedangkan LSTM berperan dalam menangkap pola temporal atau urutan data. Lapisan konvolusi dan lapisan LSTM merupakan bagian utama pada arsitektur model ini. Lapisan konvolusi akan menganalisis bagian-bagian dari data input dengan filter konvolusi, kemudian filter tersebut akan mempelajari pola dan korelasi yang penting pada data. Lapisan ini akan menghasilkan kumpulan peta fitur yang merepresentasikan informasi spasial yang telah diekstrak. Pada sisi LSTM, pola temporal atau urutan data akan ditangkap. LSTM memiliki sel memori yang dapat mengambil informasi dari input sebelumnya [84].

Berdasarkan diagram pada Gambar 2.5, alur proses CNN-LSTM dimulai dari proses dimasukkannya data *training* yang kemudian akan diinisialisasikan parameter jaringannya untuk menetapkan bobotnya pada awal setiap layer model. Setelah itu, data input akan melewati layer konvolusi dan layer *pooling* (penyatuan) pada lapisan CNN yang kemudian akan mengekstrak fitur data input yang menghasilkan data input untuk lapisan LSTM. Data input yang dihasilkan CNN ini akan dilakukan proses prediksi nilai deret waktu yang kemudian akan menghasilkan nilai sebagai data input pada lapisan *full connection* (koneksi penuh). Lapisan *full connection* ini akan menghasilkan nilai akhir prediksi [83]. Sebagaimana telah dijelaskan pada bagian 2.3.4.1 untuk persamaan LSTM dan bagian 2.3.4.2 untuk persamaan CNN, berikut adalah persamaan hibrida CNN-LSTM pada penelitian ini: [28], [41], [79]

$$h_i^{(1)} = ReLU\left(W^{(1)} * x_i + b^{(1)}\right) \quad (2.22)$$

Rumus 2. 22 Rumus *Conv1D* pada CNN-LSTM

$$p_i^{(1)} = max\left(h_{i:i+k}^{(1)}\right) \quad (2.23)$$

Rumus 2. 23 Rumus *Maxpooling1D* pada CNN-LSTM

$$d_i^{(1)} = ReLU\left(W^{(2)} \cdot p_i^{(1)} + b^{(2)}\right) \quad (2.24)$$

Rumus 2. 24 Rumus CNN *Dense* pada CNN-LSTM

$$h_t^{(2)} = LSTM\left(d_t^{(1)}, h_{t-1}^{(2)}, c_{t-1}^{(2)}\right) \quad (2.25)$$

Rumus 2. 25 Rumus LSTM pada CNN-LSTM

$$d_i^{(2)} = ReLU\left(W^{(3)} \cdot h^{(2)} + b^{(3)}\right) \quad (2.26)$$

Rumus 2. 26 Rumus LSTM *Dense* pada CNN-LSTM

$$\hat{y} = W^{(o)} \cdot h^{(2)} + b^{(o)} \quad (2.27)$$

Rumus 2. 27 Rumus *Output* pada CNN-LSTM

2.3.4.4 Algoritma XGBoost

XGBoost merupakan *gradient boosting framework* yang diciptakan pada tahun 2016 oleh Chen et al. XGBoost menggabungkan *classifier* yang bersifat lemah ke dalam *classifier* yang kuat secara linear agar menjadi lebih kuat [85]. XGBoost bertujuan untuk *fit* atau melatih data pada residualnya, yaitu selisih antara nilai asli dan nilai prediksi [86]. XGBoost memiliki kemampuan untuk mengurangi *overfitting* dengan optimalisasi pada fitur granularitas [87]. XGBoost juga dapat digunakan untuk *fine-tuning* yang membantu pengambilan data secara keseluruhan pada data [25].

Pada penelitian in, terdapat beberapa parameter XGBoost yang diterapkan *tuning.* Pertama, *n_estimators* yang berfungsi untuk menentukan jumlah pohon. Kedua, *learning rate* yang berfungsi untuk mengatur skala kontribusi pohon pada model. Ketiga, *max_depth* yang menentukan kedalaman maksimum pohon. Keempat, *subsample* yang menentukan fraksi data latih dalam pembangunan setiap pohon dan mengurangi *overfitting.* Kelima, *colsample_bytree* yang menentukan fraksi fitur secara acak juga mengurangi *overfitting* [88]. Persamaan XGBoost dituliskan pada persamaan sebagai berikut: [24]

$$\hat{y}_{t+j} = \sum_{i=1}^{I} f_i(x_j), \quad f_i \in F, (j = 1,2, \dots, N) \tag{2.28}$$

Rumus 2. 28 Rumus Prediksi Target XGBoost

$$L_k = \sum_{j=1}^{N} (y_{t+j} - \hat{y}_{t+j})^2 + \sum_{k=1}^{K} \left(\gamma T + \frac{1}{2} \propto \sum_{m=1}^{T} \omega_m^2 \right) \tag{2.29}$$

Rumus 2. 29 Rumus Minimalisir *Loss* XGBoost

Rangkaian target sampel i diprediksi menggunakan persamaan pada Rumus 2.28, dimana F adalah singkatan dari

ruang pohon regresi. Tujuan XGBoost adalah meminimalkan fungsi *loss*, seperti yang dinyatakan dalam Rumus 2.29. Pada rumus ini, *T* adalah jumlah simpul daun, dan *N* adalah himpunan semua sampel di daun *m*. Pengukuran skor *m* daun diukur dengan $\omega_m \cdot \propto$ dan parameter pohonnya adalah γ.

2.3.5 Metode Evaluasi

Evaluation metrics merupakan matriks-matriks yang digunakan dalam mengevaluasi kinerja model *machine learning* maupun *deep learning* yang dibuat. Proses evaluasi ini umumnya digunakan dalam pemeriksaan tingkatan akurasi dari hasil eksperimen, yang kemudian dibagi menjadi beberapa skenario pengujian [89]. Adapun beberapa matriks evaluasi dijelaskan sebagai berikut: [90], [91]

1. *Mean Squared Error* (MSE)

   Mean Squared Error atau MSE merupakan suatu ukuran yang mengevaluasi rata-rata dari nilai kuadrat setiap selisih antara nilai yang diprediksi oleh model dengan nilai sebenarnya. Semakin kecil nilai MSE yang diperoleh, semakin baik performa model, menunjukkan tingkat kesesuaian yang lebih tinggi antara prediksi dan nilai yang seharusnya. Adapun persamaan *Mean Squared Error* didefinisikan sebagai berikut: [90], [91]

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2 \quad (2.30)$$

Rumus 2. 30 Rumus *Mean Squared Error*

2. *Root Mean Squared Error* (RMSE)

   Root Mean Squared Error (RMSE) adalah nilai akar dari *Mean Squared Error* (MSE). RMSE memberikan ukuran yang lebih mudah dimengerti karena nilainya memiliki satuan yang sama dengan variabel yang diukur. Semakin kecil nilai RMSE, maka semakin baik kinerja

model dalam memprediksi nilai sebenarnya. Adapun persamaan *Root Mean Squared Error* didefinisikan sebagai berikut: [90], [91]

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \quad (2.31)$$

Rumus 2. 31 Rumus *Root Mean Squared Error*

3. *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) merupakan metrik yang mengevaluasi rata-rata kesalahan absolut antara nilai prediksi dengan nilai sebenarnya. MAE menunjukkan selisih kesalahan prediksi model. Semakin rendah nilai MAE yang diperoleh, maka semakin baik kinerja prediksi dari model dalam memprediksi nilai sebenarnya. Adapun persamaan *Mean Absolute Error* didefinisikan sebagai berikut: [90], [91]

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|X_i - Y_i| \quad (2.32)$$

Rumus 2. 32 Rumus *Mean Absolute Error*

4. *Mean Absolute Percentage Error* (MAPE)

Mean Absolute Percentage Error (MAPE) merupakan metrik yang mengevaluasi rata-rata persentase kesalahan prediksi terhadap nilai sebenarnya. MAPE mengukur persentase rata-rata dari selisih absolut antara nilai prediksi dan nilai yang sebenarnya. Semakin rendah nilai MAPE yang diperoleh, maka semakin baik kinerja prediksi dari model dalam memperkirakan nilai sebenarnya. Adapun persamaan *Mean Absolute Percentage Error* didefinisikan sebagai berikut: [90], [91]

$$MAPE = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{X_i - Y_i}{Y_i}\right| * 100 \quad (2.33)$$

Rumus 2. 33 Rumus *Mean Absolute Percentage Error*

2.4 Alat Penelitian

2.4.1 TensorFlow

TensorFlow merupakan library *open-source* yang dapat digunakan untuk membuat dan melatih model *machine learning* [92]. TensorFlow dikembangkan oleh Google sebagai bagian dari proyek GoogleBrain dan dirilis pertama kalinya pada November 2015 [93]. Penamaan TensorFlow mengacu pada sebuah Tensor, yaitu struktur data *multidimensional array of numbers* pada algoritma *deep learning* [94]. TensorFlow telah umum digunakan untuk memproses dan menganalisis data spasial, melakukan komputasi numerik dan *machine learning* berskala besar, dan telah digunakan oleh banyak komunitas *developer* dan *researcher* [93], [95].

Penelitian ini menggunakan salah satu *high-level* API TensorFlow, yaitu Keras. Keras dibuat agar ramah pengguna dan memberikan antarmuka yang konsisten dalam penyelesaian masalah *machine learning*, khususnya *deep learning* [96]. Keras dapat meliputi seluruh tahapan dalam *machine learning*, mulai dari *data preprocessing, hyperparameter tuning,* hingga *deployment* [96]. Pada *deep learning*, Keras menyediakan komponen modular yang dapat terintegrasi dengan berbagai framework seperti PyTorch dan TensorFlow yang memungkinkan pengguna untuk melatih model dengan beragam struktur data [97].

2.4.2 Python

Python merupakan bahasa pemrograman tingkat tinggi yang ditafsirkan. Python diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991 [98]. Python merupakan bahasa pemrograman yang bersifat *open-source* dan *cross-platform.* CPython

merupakan referensi atas bahasa pemrograman Python sebab penulisannya dilakukan dalam bahasa C. Python merupakan bahasa pemrograman *multi-purpose* sebab ekstensinya yang banyak [99].

2.4.3 R

Bahasa pemrograman R merupakan bahasa komputasi statistic dan grafis yang diciptakan dalam proyek GNU oleh John Chambers et al. R menyediakan berbagai macam teknik grafis dan statistical seperti pemodelan linear dan nonlinear, klasifikasi, klastering, dan sebagainya. R bersifat *open-source* dan menawarkan kemudahan dalam membuat plot dengan baik yang mencakup simbol matematika dan formula. R memberikan fasilitas untuk kalkulasi data, manipulasi data, dan penampilan hasil secara grafis [100].

2.4.4 Jupyter Notebook

Jupyter Notebook merupakan perangkat lunak *open-standard* berbasis layanan website untuk komputasi interaktif di seluruh bahasa pemrograman. Jupyter Notebook merupakan aplikasi web sumber terbuka yang memungkinkan pembuatan dan berbagi dokumen yang berisi *live code*, persamaan, visualisasi, dan teks. Selain itu, Jupyter Notebook memungkinkan penggunanya untuk menjalankan ulang *code* setelah adanya perubahan algoritma atau data yang diterapkan sehingga akan berpengaruh pada hasil akhirnya [99].