BAB 2
LANDASAN TEORI

Bab ini menguraikan landasan teori yang mendasari penelitian ini. Pembahasan mencakup konsep prediksi curah hujan, *machine learning*, *deep learning*, *Long Short Term Memory* (LSTM), *hyperparameter*, metode optimasi *hyperparameter* yang meliputi *Bayesian Optimization* dan *Genetic Algorithm*, serta metrik evaluasi yang digunakan untuk menilai performa model prediksi.

2.1 Prediksi Curah Hujan

Prediksi curah hujan merupakan proses memperkirakan curah hujan pada periode mendatang dengan menggunakan data historis dan kondisi cuaca. Hasil prediksi dapat digunakan untuk membantu kegiatan pertanian, pengelolaan sumber daya air, serta persiapan menghadapi banjir dan kekeringan. Besarnya curah hujan dipengaruhi oleh beberapa kondisi cuaca seperti suhu udara, kelembapan, kecepatan angin, tekanan udara, radiasi, dan penyinaran matahari. Hubungan antarvariabel tersebut tidak selalu linier dan dapat berubah dari waktu ke waktu [4, 1]. Prediksi curah hujan perlu memperhatikan hubungan antarvariabel cuaca agar pola yang terdapat pada data dapat dipelajari dengan baik.

Data curah hujan termasuk data *time series* karena setiap data tersusun berdasarkan waktu. Nilai curah hujan pada suatu periode dapat berkaitan dengan kondisi pada beberapa periode sebelumnya. Pola dari data historis tersebut digunakan untuk memperkirakan curah hujan pada periode berikutnya [12, 13]. Prediksi curah hujan dapat dilakukan menggunakan model statistik, *machine learning*, dan *deep learning*. Model *machine learning* dan *deep learning* mempelajari hubungan antarvariabel dari data yang digunakan selama pelatihan. Pendekatan tersebut banyak digunakan pada data cuaca karena dapat menangani pola waktu dan hubungan antarvariabel yang kompleks [13].

2.2 Machine Learning

Machine learning merupakan bagian dari kecerdasan buatan yang digunakan untuk mempelajari pola dari data. Pola tersebut dapat digunakan untuk melakukan klasifikasi atau prediksi. Dalam prediksi curah hujan, *machine learning* digunakan untuk mempelajari hubungan antara variabel meteorologi dan curah

hujan berdasarkan data historis [1]. *Machine learning* terdiri dari beberapa pendekatan, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Penelitian ini menggunakan *supervised learning* karena data yang digunakan memiliki pasangan antara masukan dan target. Variabel meteorologi digunakan sebagai masukan, sedangkan curah hujan pada periode berikutnya menjadi target prediksi.

Beberapa metode *machine learning* telah digunakan untuk prediksi curah hujan, seperti regresi linier, *Support Vector Regression* (SVR), *Decision Tree*, dan *Artificial Neural Network* (ANN). Metode-metode tersebut mempelajari hubungan antara variabel cuaca dan curah hujan dengan cara yang berbeda [1]. Metode yang tidak dirancang untuk data berurutan memerlukan fitur waktu agar informasi dari periode sebelumnya dapat digunakan. Hal ini penting karena data curah hujan memiliki urutan waktu dan dapat berkaitan dengan kondisi sebelumnya. Metode *deep learning* dapat digunakan untuk mempelajari pola pada data yang berbentuk urutan [12, 13].

2.3 Deep Learning

Deep learning merupakan bagian dari *machine learning* yang menggunakan jaringan saraf dengan beberapa lapisan. Setiap lapisan memproses informasi dari lapisan sebelumnya sehingga model dapat mempelajari pola yang lebih kompleks [13]. Model *deep learning* dapat membentuk representasi data selama proses pelatihan. Kebutuhan untuk menyusun fitur secara manual dapat dikurangi karena pola pada data dipelajari melalui lapisan-lapisan jaringan. Pendekatan ini digunakan pada pengolahan citra, pengenalan suara, pemrosesan bahasa, dan prediksi *time series*.

Pada prediksi curah hujan, *deep learning* digunakan untuk mempelajari hubungan antarvariabel meteorologi dan perubahan data berdasarkan waktu [4]. Salah satu arsitektur untuk data berurutan adalah *Recurrent Neural Network* (RNN). RNN menggunakan informasi dari periode sebelumnya dalam proses perhitungan pada periode berikutnya. RNN memiliki kesulitan dalam mempertahankan informasi dari urutan yang panjang. Gradien dapat semakin kecil selama proses pelatihan sehingga informasi dari periode yang jauh tidak lagi memberikan pengaruh yang cukup. Kondisi tersebut dikenal sebagai *vanishing gradient* [12]. LSTM dikembangkan untuk mengurangi masalah tersebut dan akan dijelaskan pada bagian berikutnya.

2.4 Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan pengembangan dari *Recurrent Neural Network* (RNN) yang dirancang untuk mempertahankan informasi pada urutan data yang panjang. LSTM menggunakan *cell state* dan beberapa *gate* untuk mengatur informasi yang disimpan, diperbarui, atau dikeluarkan pada setiap langkah waktu [13]. Metode ini dapat digunakan pada prediksi curah hujan karena kondisi pada suatu periode dapat berkaitan dengan kondisi pada beberapa periode sebelumnya. Informasi tersebut dapat dipertahankan oleh LSTM untuk mempelajari pola perubahan curah hujan dari waktu ke waktu [4].

LSTM memiliki dua komponen utama, yaitu *cell state* (C_t) dan *hidden state* (h_t). *Cell state* membawa informasi dari langkah waktu sebelumnya, sedangkan *hidden state* menjadi keluaran pada langkah waktu saat ini. Aliran informasi tersebut diatur oleh *forget gate*, *input gate*, dan *output gate*.

Forget gate menentukan informasi dari *cell state* sebelumnya yang masih dipertahankan. Nilainya dihitung menggunakan Rumus 2.1.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{2.1}$$

Input gate menentukan informasi baru yang akan dimasukkan ke dalam *cell state*. Perhitungannya ditunjukkan pada Rumus 2.2.

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right) \tag{2.2}$$

Kandidat *cell state* berisi informasi baru yang dapat ditambahkan ke dalam memori. Nilainya dihitung menggunakan Rumus 2.3.

$$\widetilde{C}_t = \tanh\left(W_c[h_{t-1}, x_t] + b_c\right) \tag{2.3}$$

Nilai *cell state* diperbarui dengan menggabungkan informasi lama yang dipertahankan dan informasi baru dari *input gate*. Proses tersebut ditunjukkan pada Rumus 2.4.

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{2.4}$$

Output gate menentukan bagian dari *cell state* yang digunakan sebagai keluaran. Nilainya dihitung menggunakan Rumus 2.5.

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right) \tag{2.5}$$

Nilai *hidden state* pada waktu ke-t dihitung berdasarkan *output gate* dan *cell state* seperti pada Rumus 2.6.

$$h_t = o_t \odot \tanh(C_t) \tag{2.6}$$

Pada Rumus tersebut, x_t merupakan data masukan pada waktu ke-t. Simbol h_{t-1} dan C_{t-1} menyatakan *hidden state* dan *cell state* dari waktu sebelumnya. Simbol f_t, i_t, dan o_t merupakan nilai yang dihasilkan oleh *forget gate*, *input gate*, dan *output gate*. Kandidat *cell state* dinyatakan dengan $\widetilde{C}_t$.

Simbol W dan b menunjukkan bobot dan bias yang dipelajari selama proses pelatihan. Fungsi σ merupakan fungsi *sigmoid* yang menghasilkan nilai antara 0 dan 1, sedangkan fungsi tanh menghasilkan nilai antara -1 dan 1. Simbol $\odot$ menunjukkan operasi perkalian pada setiap elemen.

2.5 Hyperparameter

Hyperparameter merupakan pengaturan yang ditentukan sebelum proses pelatihan dimulai dan tidak dipelajari secara langsung dari data. Berbeda dengan bobot dan bias yang diperbarui selama pelatihan, nilai *hyperparameter* digunakan untuk mengatur arsitektur model dan proses pembaruan bobot. Pemilihan nilai *hyperparameter* dapat memengaruhi waktu pelatihan, kemampuan model dalam mempelajari pola, serta hasil prediksi yang dihasilkan [7]. Pada model LSTM, jumlah *layer* dan jumlah unit menentukan kapasitas model dalam mempelajari pola dari data. Kapasitas yang terlalu kecil dapat menyebabkan model kesulitan menangkap hubungan yang terdapat pada data, sedangkan kapasitas yang terlalu besar dapat meningkatkan waktu komputasi dan membuat model lebih mudah menyesuaikan diri secara berlebihan terhadap data pelatihan.

Nilai *dropout rate* digunakan untuk menonaktifkan sebagian keluaran secara acak selama proses pelatihan sehingga ketergantungan model terhadap unit tertentu

dapat dikurangi. Oleh karena itu, pengaturan arsitektur dan nilai *dropout* perlu disesuaikan dengan karakteristik data yang digunakan [6]. *Learning rate* mengatur besar perubahan bobot pada setiap proses optimasi. Nilai yang terlalu besar dapat menyebabkan proses pelatihan menjadi tidak stabil, sedangkan nilai yang terlalu kecil dapat membuat proses pembaruan bobot berlangsung lebih lambat. Sementara itu, *batch size* menentukan jumlah data yang diproses sebelum bobot model diperbarui. Nilai *batch size* dapat memengaruhi penggunaan memori, waktu pelatihan, dan perubahan nilai *loss* selama proses pelatihan.

Penelitian ini mengoptimasi jumlah *LSTM layer*, jumlah unit pada *layer* pertama, jumlah unit pada *layer* kedua, *dropout rate*, *learning rate*, dan *batch size*. Jumlah unit pada *layer* kedua hanya digunakan apabila model terdiri atas dua *LSTM layer*. Seluruh *hyperparameter* tersebut dicari menggunakan *Bayesian Optimization* dan *Genetic Algorithm* dengan ruang pencarian yang sama. Beberapa pengaturan model, yaitu *optimizer*, *loss function*, fungsi aktivasi, jumlah maksimum *epoch*, dan pengaturan *early stopping*, ditetapkan sama pada seluruh percobaan agar perbandingan lebih berfokus pada cara kedua metode menentukan kombinasi *hyperparameter*. Penentuan *hyperparameter* secara manual memerlukan banyak percobaan karena setiap pengaturan dapat memiliki beberapa pilihan nilai. Oleh karena itu, penelitian ini menggunakan *Bayesian Optimization* dan *Genetic Algorithm* untuk mencari kombinasi *hyperparameter* model LSTM. Cara kerja kedua metode tersebut dijelaskan pada bagian berikutnya.

2.6 Bayesian Optimization

Bayesian Optimization merupakan metode optimasi yang menentukan konfigurasi *hyperparameter* berdasarkan hasil evaluasi yang telah diperoleh sebelumnya. Metode ini tidak perlu mengevaluasi seluruh kombinasi yang terdapat dalam ruang pencarian. Setiap hasil evaluasi digunakan untuk memperkirakan konfigurasi berikutnya yang berpeluang menghasilkan nilai fungsi objektif yang lebih baik [8]. Dalam penelitian ini, fungsi objektif yang diminimalkan adalah nilai RMSE pada data validasi. Penggunaan RMSE validasi memungkinkan proses pencarian menilai kualitas setiap konfigurasi tanpa melibatkan data uji yang digunakan pada evaluasi akhir.

Proses *Bayesian Optimization* diawali dengan memilih beberapa konfigurasi secara acak dari ruang pencarian *hyperparameter*. Setiap konfigurasi digunakan untuk membangun dan melatih model LSTM, kemudian menghasilkan nilai RMSE

pada data validasi. Hasil evaluasi tersebut menjadi data awal yang menggambarkan hubungan antara konfigurasi *hyperparameter* dan nilai fungsi objektif. Data awal selanjutnya digunakan untuk membentuk *surrogate model*. Penelitian ini menggunakan *Gaussian Process* (GP) sebagai *surrogate model*. Setiap konfigurasi *hyperparameter* dinyatakan sebagai vektor x, sedangkan nilai fungsi objektifnya dinyatakan dengan $f(x)$. Hubungan tersebut dimodelkan menggunakan fungsi rata-rata dan fungsi *kernel* seperti ditunjukkan pada Rumus 2.7.

$$f(x) \sim \mathscr{GP}\left(m(x), k(x,x')\right) \tag{2.7}$$

Pada Rumus 2.7, fungsi $m(x)$ menunjukkan rata-rata nilai fungsi objektif yang diperkirakan, sedangkan $k(x,x')$ menunjukkan hubungan antara dua konfigurasi. Konfigurasi yang memiliki nilai *hyperparameter* berdekatan dapat menghasilkan nilai fungsi objektif dengan pola yang serupa. Fungsi *kernel* digunakan oleh GP untuk mengukur kedekatan tersebut dan menentukan pengaruh hasil evaluasi yang telah tersedia terhadap perkiraan pada konfigurasi yang belum dievaluasi.

Kernel yang digunakan terdiri atas *Constant Kernel*, *Matérn Kernel*, dan *White Kernel*. *Constant Kernel* mengatur skala variasi fungsi yang diperkirakan, sedangkan *Matérn Kernel* dengan nilai $\nu = 2{,}5$ digunakan untuk mengukur kemiripan antarkonfigurasi. *White Kernel* digunakan untuk menambahkan komponen *noise* pada model. Bentuk kombinasi *kernel* tersebut ditunjukkan pada Rumus 2.8.

$$k(x,x') = \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right) + \sigma_n^2 \delta(x,x') \tag{2.8}$$

Jarak antara dua konfigurasi dihitung menggunakan Rumus 2.9.

$$r = \left\|x - x'\right\| \tag{2.9}$$

Pada Rumus 2.8, simbol ℓ menunjukkan *length scale* yang mengatur pengaruh jarak antarkonfigurasi. Nilai σ_f^2 menunjukkan skala variasi fungsi yang diperkirakan oleh GP, sedangkan σ_n^2 menunjukkan besarnya komponen *noise*.

Fungsi $\delta(x,x')$ bernilai satu ketika $x = x'$ dan bernilai nol ketika kedua konfigurasi berbeda. Nilai r pada Rumus 2.9 menunjukkan jarak antara konfigurasi x dan x'.

Setelah GP dibentuk, model tersebut digunakan untuk memperkirakan nilai fungsi objektif pada konfigurasi yang belum dievaluasi. Hasil perkiraan terdiri atas nilai rata-rata $\mu(x_*)$ dan varians $\sigma^2(x_*)$. Nilai rata-rata menunjukkan RMSE validasi yang diperkirakan pada konfigurasi tersebut, sedangkan varians menunjukkan tingkat ketidakpastian GP terhadap perkiraannya. Rata-rata prediksi GP dihitung menggunakan Rumus 2.10.

$$\mu(x_*) = m(x_*) + k_*^T \left(K + \sigma_n^2 I\right)^{-1} (y - m(X)) \tag{2.10}$$

Varians prediksi GP dihitung menggunakan Rumus 2.11.

$$\sigma^2(x_*) = k(x_*, x_*) - k_*^T \left(K + \sigma_n^2 I\right)^{-1} k_* \tag{2.11}$$

Pada Rumus 2.10 dan Rumus 2.11, simbol x_* menunjukkan konfigurasi yang belum dievaluasi. Matriks K memuat nilai *kernel* dari pasangan konfigurasi yang telah dievaluasi, sedangkan vektor k_* menunjukkan hubungan antara konfigurasi baru dan konfigurasi yang telah tersedia. Nilai fungsi objektif dari seluruh konfigurasi yang telah dievaluasi dinyatakan dengan y, sedangkan I merupakan matriks identitas. Hasil perhitungan tersebut memungkinkan GP memberikan perkiraan nilai RMSE sekaligus tingkat ketidakpastian pada setiap konfigurasi baru.

Nilai rata-rata dan ketidakpastian yang dihasilkan oleh GP digunakan oleh *acquisition function* untuk menentukan konfigurasi berikutnya. Penelitian ini menggunakan *Expected Improvement* (EI), yaitu fungsi yang memperkirakan besarnya perbaikan yang mungkin diperoleh dibandingkan nilai fungsi objektif terbaik yang telah ditemukan. Karena fungsi objektif berupa RMSE yang perlu diminimalkan, EI dihitung menggunakan Rumus 2.12 [8].

$$EI(x) = (f_{\text{best}} - \mu(x) - \xi)\,\Phi(Z) + \sigma(x)\phi(Z) \tag{2.12}$$

Nilai Z pada perhitungan EI diperoleh melalui Rumus 2.13.

$$Z = \frac{f_{\text{best}} - \mu(x) - \xi}{\sigma(x)} \tag{2.13}$$

Pada Rumus 2.12 dan Rumus 2.13, f_{best} merupakan RMSE validasi terendah yang telah diperoleh. Simbol $\mu(x)$ menunjukkan nilai RMSE yang diperkirakan oleh GP, sedangkan $\sigma(x)$ menunjukkan tingkat ketidakpastian terhadap perkiraan tersebut. Parameter ξ mengatur keseimbangan antara mencoba konfigurasi pada bagian ruang pencarian yang belum banyak diperiksa dan memilih konfigurasi di sekitar hasil terbaik. Simbol $\Phi(Z)$ menunjukkan fungsi distribusi kumulatif, sedangkan $\phi(Z)$ menunjukkan fungsi kepadatan probabilitas dari distribusi normal standar. Nilai EI ditetapkan menjadi nol ketika $\sigma(x)$ sama dengan nol.

Nilai EI dihitung pada sekumpulan konfigurasi yang belum dievaluasi. Konfigurasi dengan nilai EI tertinggi dipilih sebagai konfigurasi berikutnya, seperti ditunjukkan pada Rumus 2.14.

$$x_{\text{next}} = \arg\max_{x \in \mathscr{C}} EI(x) \tag{2.14}$$

Pada Rumus 2.14, simbol $\mathscr{C}$ menunjukkan kumpulan konfigurasi yang belum dievaluasi. Konfigurasi x_{next} digunakan untuk membangun dan melatih model LSTM, kemudian nilai RMSE validasinya dihitung. Hasil evaluasi tersebut ditambahkan ke data yang telah tersedia sehingga GP dapat diperbarui untuk menentukan konfigurasi berikutnya. Proses tersebut dilakukan secara berulang sampai jumlah evaluasi yang telah ditetapkan tercapai. Konfigurasi dengan nilai RMSE validasi terendah kemudian dipilih sebagai hasil akhir *Bayesian Optimization*.

2.7 Genetic Algorithm

Genetic Algorithm (GA) merupakan metode optimasi berbasis populasi yang meniru proses evolusi melalui tahapan seleksi, *crossover*, mutasi, dan *elitism* [10]. Dalam optimasi *hyperparameter*, setiap individu merepresentasikan satu konfigurasi model, sedangkan setiap gen menyimpan nilai *hyperparameter* tertentu. Pada model LSTM, gen dapat berupa jumlah *layer*, jumlah unit pada setiap *layer*, *dropout rate*, *learning rate*, dan *batch size*. Sekumpulan individu membentuk satu populasi yang dievaluasi untuk menentukan kualitas setiap konfigurasi. Dalam penelitian ini, kualitas individu diukur menggunakan RMSE pada data validasi sehingga konfigurasi dengan nilai RMSE yang lebih rendah dianggap memiliki *fitness* yang lebih baik.

Proses GA dimulai dengan membentuk populasi awal secara acak dari ruang pencarian *hyperparameter*. Setiap individu dalam populasi digunakan untuk membangun dan melatih model LSTM, kemudian nilai RMSE pada data validasi dihitung. Setelah seluruh populasi dievaluasi, pemilihan induk dilakukan menggunakan *tournament selection*. Beberapa individu dipilih secara acak dari populasi untuk membentuk satu kelompok turnamen, kemudian individu dengan nilai *fitness* terbaik dipilih sebagai induk. Karena nilai RMSE digunakan sebagai fungsi objektif yang diminimalkan, individu dengan RMSE terendah menjadi pemenang turnamen. Proses pemilihan tersebut ditunjukkan pada Rumus 2.15.

$$p^* = \arg\min_{p_i \in T} F(p_i) \tag{2.15}$$

Pada Rumus 2.15, T merupakan kelompok individu yang dipilih dalam satu turnamen, sedangkan p_i menyatakan individu ke i di dalam kelompok tersebut. Fungsi $F(p_i)$ menunjukkan nilai *fitness* dari individu p_i, sedangkan p^* menunjukkan individu dengan nilai *fitness* terendah yang dipilih sebagai induk. Proses turnamen dilakukan kembali untuk memperoleh induk kedua. Mekanisme tersebut memberikan peluang lebih besar kepada individu dengan hasil yang baik untuk menghasilkan individu baru, tetapi tetap memungkinkan individu lain dalam populasi untuk terpilih.

Kedua induk yang telah dipilih selanjutnya digunakan untuk membentuk individu baru melalui *uniform crossover*. Pada proses ini, setiap gen pada individu baru dapat diambil dari induk pertama atau induk kedua dengan peluang yang sama. Dengan demikian, individu baru dapat memiliki kombinasi *hyperparameter* yang berasal dari kedua induk. Pembentukan gen ke j melalui *uniform crossover* ditunjukkan pada Rumus 2.16.

$$c_j = \begin{cases} p_j^{(1)}, & r_j < 0{,}5, \\ p_j^{(2)}, & r_j \geq 0{,}5. \end{cases} \tag{2.16}$$

Pada Rumus 2.16, c_j merupakan gen ke j pada individu baru. Simbol $p_j^{(1)}$ dan $p_j^{(2)}$ menunjukkan gen ke j dari induk pertama dan induk kedua, sedangkan r_j merupakan bilangan acak pada rentang 0 sampai 1. Gen diambil dari induk pertama ketika $r_j < 0{,}5$, sedangkan gen dari induk kedua digunakan ketika $r_j \geq 0{,}5$. Pelaksanaan *uniform crossover* tetap dipengaruhi oleh nilai *crossover rate*. Apabila

proses *crossover* tidak dilakukan, individu baru dapat menggunakan konfigurasi salah satu induk sesuai dengan aturan yang telah ditetapkan.

Setelah proses *crossover*, beberapa gen pada individu baru dapat diubah melalui mutasi. Proses ini dilakukan dengan mengganti nilai gen menggunakan nilai lain yang masih berada dalam ruang pencarian *hyperparameter*. Mutasi membantu menjaga keberagaman populasi dan mengurangi kemungkinan proses pencarian berhenti terlalu cepat pada satu bagian ruang pencarian. Laju mutasi dapat dibuat lebih tinggi pada generasi awal agar lebih banyak konfigurasi dapat diperiksa, kemudian diturunkan secara bertahap agar pencarian menjadi lebih berfokus pada konfigurasi yang telah menunjukkan hasil baik. Perubahan laju mutasi tersebut ditunjukkan pada Rumus 2.17.

$$p_m(g) = p_{m,\text{awal}} + \left(p_{m,\text{akhir}} - p_{m,\text{awal}}\right) \frac{g-1}{G-1} \tag{2.17}$$

Pada Rumus 2.17, $p_m(g)$ merupakan laju mutasi pada generasi ke g. Simbol $p_{m,\text{awal}}$ dan $p_{m,\text{akhir}}$ menunjukkan laju mutasi pada generasi awal dan generasi terakhir, sedangkan G merupakan jumlah generasi. Setiap gen diubah apabila bilangan acak yang dihasilkan lebih kecil daripada laju mutasi pada generasi tersebut. Nilai penggantinya diambil kembali dari ruang pencarian sehingga individu yang dihasilkan tetap memenuhi batas *hyperparameter* yang telah ditentukan.

Sebagian individu dengan nilai *fitness* terbaik dipertahankan melalui *elitism*. Individu tersebut diteruskan secara langsung ke generasi berikutnya agar konfigurasi terbaik yang telah ditemukan tidak hilang selama proses pembentukan populasi baru. Individu hasil *crossover* dan mutasi kemudian digabungkan dengan individu hasil *elitism* untuk membentuk generasi berikutnya. Seluruh individu baru dievaluasi menggunakan proses yang sama, kemudian tahapan seleksi, *crossover*, mutasi, dan *elitism* diulang sampai jumlah evaluasi yang ditetapkan tercapai. Dari seluruh konfigurasi yang telah dievaluasi, konfigurasi dengan nilai RMSE validasi terendah dipilih sebagai hasil akhir *Genetic Algorithm*.

2.8 Evaluasi Model

Pengukuran kesalahan model dilakukan pada proses pelatihan dan evaluasi akhir. Selama pelatihan, penelitian ini menggunakan *Mean Squared Error* (MSE) sebagai *loss function*. Sementara itu, kinerja akhir model diukur menggunakan

tiga metrik regresi, yaitu *Mean Absolute Error* (MAE), *Root Mean Square Error* (RMSE), dan koefisien determinasi (R^2). MAE dan RMSE digunakan untuk mengukur kesalahan prediksi, sedangkan R^2 digunakan untuk menunjukkan kemampuan model dalam menjelaskan variasi nilai curah hujan aktual [14].

MSE menghitung rata-rata kuadrat selisih antara nilai aktual dan nilai prediksi. Pengkuadratan menyebabkan kesalahan dengan nilai yang lebih besar memberikan pengaruh lebih tinggi terhadap nilai MSE. Dalam proses pelatihan, nilai MSE digunakan oleh *optimizer* untuk memperbarui bobot dan bias model. Perhitungan MSE ditunjukkan pada Rumus 2.18.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.18}$$

MAE menghitung rata-rata selisih absolut antara nilai aktual dan nilai prediksi. Nilainya memiliki satuan yang sama dengan curah hujan sehingga dapat menunjukkan rata-rata besar kesalahan model secara langsung. Perhitungan MAE ditunjukkan pada Rumus 2.19.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.19}$$

RMSE menghitung akar dari rata-rata kuadrat selisih antara nilai aktual dan nilai prediksi. Proses pengkuadratan membuat kesalahan dengan nilai besar memberikan pengaruh lebih tinggi terhadap hasil RMSE. Berbeda dengan MSE, nilai RMSE kembali memiliki satuan yang sama dengan curah hujan. Perhitungan RMSE ditunjukkan pada Rumus 2.20.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.20}$$

Pada Rumus 2.18, Rumus 2.19, dan Rumus 2.20, y_i merupakan nilai aktual, $\hat{y}_i$ merupakan nilai hasil prediksi, dan n menunjukkan jumlah data yang dihitung. Nilai MSE, MAE, dan RMSE yang lebih rendah menunjukkan kesalahan prediksi yang lebih kecil.

Koefisien determinasi atau R^2 digunakan untuk mengukur kemampuan model dalam menjelaskan variasi nilai aktual. Perhitungannya membandingkan jumlah kuadrat kesalahan model dengan jumlah kuadrat selisih nilai aktual terhadap

rata-ratanya. Rumus R^2 ditunjukkan pada Rumus 2.21.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{2.21}$$

Pada Rumus 2.21, $\bar{y}$ merupakan rata-rata nilai aktual. Nilai R^2 yang mendekati 1 menunjukkan bahwa hasil prediksi mampu menjelaskan variasi data aktual dengan lebih baik. Nilai R^2 dapat bernilai negatif ketika hasil prediksi model lebih buruk dibandingkan penggunaan rata-rata nilai aktual sebagai prediksi.

MSE digunakan untuk mengarahkan proses pembaruan bobot selama pelatihan, sedangkan MAE, RMSE, dan R^2 digunakan untuk mengevaluasi hasil akhir model. MAE menunjukkan rata-rata kesalahan prediksi, RMSE memberikan pengaruh lebih besar terhadap kesalahan bernilai tinggi, dan R^2 menunjukkan kemampuan model dalam menjelaskan variasi nilai curah hujan.

2.9 Penelitian Terdahulu

Penelitian terdahulu dipilih berdasarkan keterkaitannya dengan prediksi curah hujan menggunakan LSTM dan optimasi *hyperparameter*. Penelitian yang dibahas mencakup penggunaan LSTM pada data meteorologi, penerapan *Bayesian Optimization*, penggunaan *Genetic Algorithm*, dan metode optimasi lain pada prediksi curah hujan. Ringkasan penelitian tersebut dijelaskan pada Tabel 2.1.

Tabel 2.1. Penelitian terdahulu

Referensi	Data atau Objek	Metode	Hasil dan Keterkaitan
[1]	Data meteorologi dari enam stasiun di wilayah Himalaya Barat Laut pada periode 1980–2021.	ARIMA, TBATS, RF, SVR, ANN, KNN, RNN, GRU, LSTM, *deep* LSTM, dan BiLSTM.	BiLSTM menghasilkan kesalahan prediksi paling rendah dan diikuti oleh LSTM. Hasil tersebut menunjukkan bahwa LSTM dapat digunakan untuk mempelajari pola curah hujan dari data meteorologi. Penelitian ini belum membahas optimasi *hyperparameter* maupun perbandingan BO dan GA.
Lanjut pada halaman berikutnya			

Tabel 2.1 Penelitian terdahulu (lanjutan)

Referensi	Data atau Objek	Metode	Hasil dan Keterkaitan
[5]	Data meteorologi harian Kota Palembang periode 2011–2021 yang terdiri atas 13 variabel.	GRU dan LSTM dengan pengujian beberapa nilai *window size*, jumlah unit, dan konfigurasi pelatihan.	LSTM memperoleh R^2 sebesar 0,70, sedangkan GRU memperoleh R^2 sebesar 0,54. Penelitian ini memiliki kesamaan pada penggunaan data meteorologi dan LSTM untuk prediksi curah hujan di Indonesia, tetapi konfigurasi model belum ditentukan menggunakan BO atau GA.
[9]	Data curah hujan bulanan dari sembilan stasiun cuaca di Bangladesh.	LSTM dengan *Bayesian Optimization* untuk menentukan jumlah *lag* dan *hyperparameter* model.	LSTM dengan jumlah *lag* dan *hyperparameter* hasil optimasi mengungguli sebagian besar model pembanding berdasarkan RMSE, MAE, dan SMAPE. Penelitian ini sama-sama menerapkan BO pada LSTM untuk prediksi curah hujan, tetapi tidak membandingkannya dengan GA dalam kondisi pencarian yang sama.
[11]	Model LSTM untuk prediksi kata berikutnya pada data teks.	*Genetic Algorithm* untuk menentukan jumlah *hidden layer*, jumlah unit, *batch size*, jumlah langkah waktu, dan jumlah *epoch*.	Konfigurasi hasil GA memberikan performa yang lebih baik daripada konfigurasi awal. Penelitian tersebut menunjukkan bahwa GA dapat digunakan untuk menentukan *hyperparameter* LSTM, tetapi objek yang digunakan bukan curah hujan dan GA tidak dibandingkan dengan BO.
Lanjut pada halaman berikutnya			

Tabel 2.1 Penelitian terdahulu (lanjutan)

Referensi	Data atau Objek	Metode	Hasil dan Keterkaitan
[15]	Data curah hujan yang dilengkapi dengan beberapa variabel meteorologi.	Pemilihan fitur dan optimasi *hyperparameter* menggunakan AD-PSO-Guided WOA pada DT, RF, MLP, LSTM, dan KNN.	LSTM memberikan hasil terbaik di antara model yang diuji. Model AD-PSO-Guided WOA-LSTM memperoleh R^2 sebesar 0,9636. Penelitian tersebut menunjukkan penggunaan metode metaheuristik untuk mengoptimasi LSTM pada prediksi curah hujan, tetapi menggunakan metode hibrida dan tidak membandingkan BO dengan GA.

Penelitian terdahulu menunjukkan bahwa LSTM telah digunakan untuk mempelajari pola curah hujan berdasarkan data meteorologi. Hasil model juga dipengaruhi oleh pemilihan *hyperparameter*. Beberapa penelitian telah menggunakan *Bayesian Optimization*, *Genetic Algorithm*, dan metode metaheuristik untuk menentukan konfigurasi model, tetapi penerapannya dilakukan pada objek dan kondisi pengujian yang berbeda.

Penelitian yang dirangkum pada Tabel 2.1 belum membandingkan *Bayesian Optimization* dan *Genetic Algorithm* pada model LSTM menggunakan data, ruang *hyperparameter*, fungsi objektif, dan jumlah evaluasi yang sama. Penelitian ini membandingkan kedua metode pada prediksi curah hujan di Kota Tangerang. Model hasil optimasi kemudian dibandingkan dengan LSTM *baseline* menggunakan MAE, RMSE, dan R^2.