SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya sendiri. Sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang ditulis atau diterbitkan orang lain kecuali sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang telah lazim.

Tanda tangan yang tertera dalam halaman pengesahan adalah asli. Jika tidak asli, saya siap menerima sanksi sesuai dengan peraturan yang berlaku.

Jambi, 04 April 2025

Yang menyatakan,

Ayu Indryani

F1E121095

PREDIKSI CURAH HUJAN MENGGUNAKAN METODE *LONG SHORT TERM MEMORY* (LSTM) (STUDI KASUS : KOTA JAMBI)

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh
Gelar Sarjana pada Program Studi Sistem Informasi

AYU INDRYANI

F1E121095

PROGRAM STUDI SISTEM INFORMASI
JURUSAN TEKNIK ELEKTRO DAN INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS JAMBI
2025

HALAMAN PENGESAHAN

Skripsi dengan judul **PREDIKSI CURAH HUJAN MENGGUNAKAN METODE *LONG SHORT TERM MEMORY* (LSTM) (STUDI KASUS : KOTA JAMBI)** yang disusun oleh **Ayu Indryani**, **NIM: F1E121095** telah dipertahankan di depan tim penguji pada tanggal 15 Mei 2025 dan dinyatakan lulus.

Susunan Tim Penguji :

Ketua	: Ulfa Khaira, S.Komp., M.Kom
Sekretaris	: Mutia Fadhila Putri, M.Kom
Anggota	:1. Benedika Ferdian Hutabarat, S.Komp., M.Kom
	2. Zainil Abidin, S.T., M.Eng

Disetujui:

Pembimbing Utama	Pembimbing Pendamping
Ulfa Khaira, S.Komp., M.Kom.	Mutia Fadhila Putri, M.Kom.
NIP. 198912292019032018	NIP. 199612162022032016

Diketahui:

Dekan Fakultas Sains dan Teknologi	Ketua Jurusan Teknik Elektro Dan Informatika
Drs. Jefri Marzal, M.Sc., D.I.T.	Edi Saputra, S.T., M.Sc.
NIP. 196806021993031004	NIP. 198501082015041003

RINGKASAN

Indonesia adalah negara beriklim tropis yang terletak di garis khatulistiwa, sehingga memiliki pola curah hujan yang bervariasi di setiap wilayahnya. Kota Jambi, sebagai salah satu daerah di Indonesia, sering mengalami fluktuasi curah hujan yang signifikan. Oleh karena itu, prediksi curah hujan yang akurat sangat diperlukan untuk mendukung berbagai sektor, seperti pertanian, mitigasi bencana, dan perencanaan infrastruktur. Namun, salah satu tantangan dalam prediksi curah hujan adalah keberadaan *missing value* dalam data yang disebabkan oleh sensor yang tidak berfungsi atau data yang tidak terukur. Data curah hujan dari BMKG sering kali mengandung nilai 8888, yang menunjukkan curah hujan tidak terukur, dan 9999, yang menunjukkan data kosong. Keberadaan nilai-nilai ini dapat mempengaruhi akurasi model prediksi.

Penelitian ini bertujuan untuk mengimplementasikan model *Long Short-Term Memory* (LSTM) dalam memprediksi curah hujan di Kota Jambi dengan berfokus pada penanganan *missing value* menggunakan tiga metode interpolasi, yaitu interpolasi linear, interpolasi kuadrat, dan interpolasi spline kuadratik. Dataset yang digunakan dalam penelitian ini adalah data curah hujan harian Kota Jambi dari tahun 2016-2024 yang diperoleh dari website DataOnline BMKG. Setelah dilakukan *preprocessing* data dan interpolasi, data yang telah diperbaiki digunakan untuk melatih model LSTM. Model yang dikembangkan dievaluasi menggunakan *Root Mean Square Error* (RMSE) untuk mengukur tingkat akurasinya. Hasil penelitian menunjukkan bahwa metode interpolasi berpengaruh terhadap performa model, di mana interpolasi linear menghasilkan tingkat akurasi yang lebih baik dibandingkan metode interpolasi lainnya dengan *Train Score* RMSE sebesar 13.0661 dan *Test Score* RMSE sebesar 13.1388 dengan komposisi 50 neuron, 75 *epoch* dan 32 *batch size* menggunakan optimasi RMSprop.

Sebagai implementasi akhir, hasil penelitian ini dituangkan dalam bentuk *Graphical User Interface* (GUI) berbasis Streamlit, yang memungkinkan pengguna untuk memasukkan data curah hujan, menampilkan grafik data aktual, melihat pembagian *data training* dan *testing*, serta melakukan prediksi hingga dua minggu ke depan. Dengan hasil yang diperoleh, penelitian ini diharapkan dapat memberikan kontribusi dalam meningkatkan akurasi prediksi curah hujan di Kota Jambi, serta menjadi referensi dalam penggunaan metode interpolasi untuk menangani *missing value* dalam analisis data *time series*.

RIWAYAT HIDUP

Ayu Indryani lahir di Solok Selatan, pada tanggal 06 Juli 2003. Penulis merupakan anak kedua dari tiga bersaudara dari pasangan Efendi dan Yeni Eflinda. Jalur pendidikan formal yang ditempuh penulis sebagai berikut :

1. SD N 02 Batang Limpaung (2009-2015)
2. SMP N 04 Solok Selatan (2015-2018)
3. SMA N 01 Solok Selatan (2018-2021)

Pada tahun 2021, penulis melanjutkan pendidikan jenjang Strata 1 dan tercatat sebagai mahasiswa di Program Studi Sistem Informasi, Jurusan Teknik Elektro dan Informatika, Fakultas Sains dan Teknologi, Universitas Jambi melalui jalur SBMPTN. Selama menempuh pendidikan S1, penulis aktif dalam bidang akademik maupun non-akademik. Penulis mengikuti organisasi Unit Kegiatan Mahasiswa Entrepreneur Mahasiswa dan Himpunan Mahasiswa Sistem Informasi sebagai anggota divisi pengembangan sumber daya anggota (PSDA). Pada tahun 2024, penulis mengikuti kegiatan Studi Independen Kampus Merdeka oleh Kemendikbudristek pada bidang *Data Analyst* yang diselenggarakan oleh GreatEdu. Dalam menambah wawasan dalam dunia kerja serta menerapkan ilmu, penulis telah mengikuti kegiatan magang di Dinas Komunikasi dan Informatika Provinsi Jambi, tepatnya di Layanan E-Government pada bagian Jambi Data & Analytic Center (JDAC).

PRAKATA

Puji dan syukur penulis ucapkan atas kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul **"PREDIKSI CURAH HUJAN MENGGUNAKAN METODE *LONG SHORT TERM MEMORY* (LSTM) (STUDI KASUS: KOTA JAMBI)"** sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Jambi.

Dalam proses penulisan skripsi ini, penulis banyak mendapatkan bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, dengan segala kerendahan hati, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, sebab berkat rahmat, karunia serta hidayah-Nya penulis dapat menyelesaikan tugas akhir dengan baik.
2. Kedua orang tua tercinta, Bapak Efendi dan Ibu Yeni Eflinda yang telah memberikan dukungan baik secara mental, fisik dan finansial, serta bekerja keras mendidik agar penulis menjadi manusia pantang menyerah, bertanggung jawab dan bermanfaat bagi orang banyak.
3. Bapak Prof. Dr. Helmi, S.H., M.H. selaku Rektor Universitas Jambi.
4. Bapak Drs. Jefri Marzal, M.Sc., D.I.T. selaku Dekan Fakultas Sains dan Teknologi Universitas Jambi.
5. Bapak Edi Saputra, S.T., M.Sc selaku Ketua Jurusan Teknik Elektro dan Informatika Fakultas Sains dan Teknologi.
6. Ibu Ulfa Khaira, S.komp., M.Kom. dan Ibu Mutia Fadhila Putri, M.Kom. selaku Dosen pembimbing yang telah banyak memberikan arahan, dukungan, dan motivasi dalam penyusunan skripsi ini.
7. Tim Penguji Skripsi, Bapak Benedika Ferdian Hutabarat, S.Komp., M.Kom. dan Bapak Zainil Abidin, S.T., M.Eng. yang telah memberikan masukan dan pengarahan untuk kesempurnaan skripsi ini.
8. Bapak Daniel Arsa, S.Kom., M.S.I. selaku Dosen Pembimbing Akademik yang senantiasa memberikan bantuan, motivasi dan pengarahan selama masa studi.
9. Seluruh Dosen di Program Studi Sistem Informasi Universitas Jambi atas segala ilmu dan bimbingannya selama masa studi.

10. dr. Arisca Indryani sebagai Kakak penulis dan Amel Indryani sebagai Adik penulis yang telah memberikan semangat dan dukungan agar penulis dapat menyelesaikan skripsi ini tepat waktu.
11. Seluruh sahabat mahasiswa/i Sistem Informasi Angkatan 2021 terutama Putri Anggellina Khairunisa, Ghaitsa Althafah Wandira, dan Jesica Pitos Dwi Putri yang telah membersamai penulis dan memberikan dukungan serta semangat dalam proses penyusunan skripsi ini.
12. Terakhir, terima kasih untuk diri sendiri, karena telah mampu berusaha keras dan berjuang sejauh ini untuk bertanggung jawab menyelesaikan apa yang telah dimulai, terima kasih sudah kuat dalam menghadapi situasi apapun, tetap semangat dan jangan putus asa. Semoga langkah ini menjadi awal dari perjalanan yang lebih baik ke depannya.

Semoga segala bantuan, dukungan dan kerjasama yang telah diberikan semua pihak diatas menjadi amal baik di sisi Allah SWT dan menjadi langkah untuk sukses bersama. Semoga skripsi ini dapat bermanfaat bagi banyak orang dan memberikan sumbangsih pada dunia pendidikan khususnya di bidang Sistem Informasi. Penulis menyadari bahwa skripsi ini masih terdapat kekurangan, untuk itu, segala bentuk kritik dan saran dengan senang hati diterima untuk perbaikan dimasa mendatang.

Jambi, 04 April 2025

Ayu Indryani
F1E121095

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

I. PENDAHULUAN

1.1 Latar Belakang

Indonesia adalah negara beriklim tropis karena terletak di garis khatulistiwa (Susilo, 2021). Iklim tropis ini membuat Indonesia rentan terhadap perubahan pola cuaca terutama dalam hal curah hujan (Toyib et al., 2024). Cuaca adalah kualitas udara yang diamati dalam waktu yang relatif singkat atau di daerah yang kecil (BMKG, 2021). Cuaca menurut WCC *(World Climate Conference)* adalah keadaan atmosfer yang diukur secara menyeluruh dengan memperhitungkan perubahan, perkembangan, dan datang atau lenyapnya suatu fenomena udara (Luthfiarta et al., 2020). Unsur-unsur yang memengaruhi cuaca yaitu suhu, udara, tekanan udara, kelembaban udara, laju uap air, awan, curah hujan dan angin (Farikhul Firdaus & Paputungan, 2022). Curah hujan sebagai salah satu unsur yang memengaruhi cuaca didefinisikan sebagai ketinggian air hujan yang terkumpul di alat penakar hujan pada permukaan datar yang tidak menyerap, meresap, mengalir, ataupun menguap. Semakin tinggi ketinggian air hujan yang terukur menunjukkan semakin banyaknya frekuensi hujan atau intensitas hujan yang terjadi di wilayah tersebut (Prayoga Dhenanta & Kholifah, 2022).

Menurut Putratama (2020), pemanasan global telah menyebabkan pola cuaca menjadi semakin tidak terduga. Peningkatan suhu udara global memengaruhi kelembaban, pola angin, dan intensitas curah hujan di seluruh dunia, termasuk Indonesia. Curah hujan adalah elemen penting dalam cuaca yang sangat memengaruhi kehidupan sehari-hari, mulai dari sektor pertanian hingga perencanaan infrastruktur (Agungnoe, 2024). Namun, sifat curah hujan yang fluktuatif membuatnya sulit diprediksi secara akurat (Wicaksono, 2022). Badan Meteorologi, Klimatologi, dan Geofisika (BMKG), sebagai lembaga resmi di Indonesia yang bertugas memberikan informasi prakiraan cuaca, telah menyediakan data historis dan model prediksi untuk berbagai wilayah, termasuk Kota Jambi. Namun, data curah hujan sering kali mengandung nilai yang tidak terukur atau rusak *(missing value)*. Nilai 8888 menunjukkan curah hujan yang tidak terukur dengan intensitas <0,5 mm, yang dapat digantikan dengan simbol TTU atau nilai 0. Sementara itu, nilai 9999 menunjukkan data kosong akibat alat pengukur yang rusak atau data yang belum tercatat (BMKG, 2024). Keberadaan nilai-nilai ini dapat mengganggu analisis data, karena dalam suatu penelitian, kualitas data sangat mempengaruhi hasil dari suatu penelitian (Ericko et al., 2023). *Missing value* dapat menyebabkan tingkat keakuratan suatu data menjadi

berkurang dan menurunnya kualitas data pada saat akan dilakukan pengolahan data lanjut, seperti prediksi (Lutfi & Hasyim, 2019).

Penanganan *missing value* menjadi tahap yang sangat penting dalam *preprocessing* data, terutama untuk data *time series* seperti curah hujan (Riko Anshori Prasetya & Mudi Priyatno, 2023). Salah satu teknik yang dapat digunakan untuk menangani *missing value* adalah interpolasi (Widianti & Pratama, 2024). Teknik interpolasi digunakan untuk memperkirakan nilai yang hilang dengan cara mengisi data berdasarkan pola di antara titik data yang diketahui (Febrianti, 2019). Dalam penelitian ini, tiga metode interpolasi akan digunakan untuk menangani *missing value* pada data curah hujan harian Kota Jambi, yaitu interpolasi linier, kuadrat, dan spline kuadratik. Teknik interpolasi linear adalah polynomial tingkat pertama yang digunakan untuk mengetahui nilai suatu titik dalam garis lurus (Rizky Ismail et al., 2023). Teknik interpolasi kuadrat adalah polynomial berderajat dua yang digunakan untuk mencari titik-titik diantara 3 buah titik. Sedangkan teknik interpolasi spline kuadratik adalah metode aproksimasi dengan melakukan interpolasi dari titik-titik *x* yang terletak diantara dua titik *xn* dan *xn*+1 dengan mengasumsikan fungsi berbentuk polinomial berpangkat dua (Sofiyani & Permanasari, 2023). Penanganan *missing value* melalui teknik interpolasi ini bertujuan untuk menghasilkan dataset yang bersih dan berkualitas, yang akan digunakan dalam proses prediksi curah hujan menggunakan metode *Long Short Term Memory* (LSTM).

Penelitian mengenai prediksi yang telah dilakukan akhir-akhir ini menunjukkan bahwa prediksi menggunakan *deep learning* telah mencapai akurasi lebih dari 85% (Sulistyo Budi et al., 2021). CNN, RNN, dan *Long Short Term Memory* (LSTM) adalah model *deep learning* yang sering digunakan untuk prediksi *time series* (Tian et al., 2018). Menurut Arrofiqoh & Harintaka (2018) CNN merupakan salah satu metode *deep learning* yang mampu melakukan proses pembelajaran mandiri untuk pengenalan objek, ekstrasi objek dan klasifikasi. Dalam penelitian Nurhikmat (2018) yang berjudul Implementasi *Convolutional Neural Network* untuk *Image Classification* pada citra wayang golek memperoleh hasil Tingkat keakurasian *training* dan *testing* dalam pengklasifikasian gambar wayang golek sebesar 95% *training* dan 90% *testing*. Sementara itu RNN cocok untuk menangani data sekuensial namun RNN tradisional menghadapi masalah *vanishing gradient*, yang menyebabkan kesulitan dalam mempelajari ketergantungan jangka panjang dalam data *time series* (Arwansyah et al., 2022). Masalah *vanishing gradient* terjadi ketika nilai gradien menjadi sangat kecil seiring waktu, sehingga mengurangi kontribusinya dalam proses pembelajaran (Carnegie & Chairani, 2023). Dalam penelitian yang dilakukan oleh Irawan (2024)

menunjukkan bahwa LSTM memiliki rata-rata nilai RMSE sebesar 4.21%, lebih rendah dibandingkan RNN dengan RMSE sebesar 4.26%, menunjukkan akurasi prediksi yang lebih tinggi. Oleh karena itu, LSTM dipilih dalam penelitian ini karena kemampuannya yang unggul dalam menangkap pola data *time series* yang kompleks dan ketergantungan jangka panjang. Dibandingkan dengan metode lain, seperti *Recurrent Neural Network* (RNN), LSTM memiliki mekanisme *forget gate*, *input gate*, dan *output gate* yang dapat memitigasi masalah *vanishing gradient*, sehingga lebih efektif dalam memprediksi data dengan pola musiman atau fluktuasi tinggi seperti curah hujan (Tarkus et al., 2020).

Meskipun LSTM telah terbukti efektif dalam berbagai aplikasi prediksi, banyak penelitian sebelumnya yang tidak memberikan perhatian yang cukup terhadap penanganan *missing value* dalam dataset. Penelitian Terdahulu, seperti yang dilakukan oleh Rizki et al. (2020) dan Farikhul Firdaus & Paputungan (2022), lebih fokus pada pengembangan arsitektur model tanpa mempertimbangkan teknik yang yang digunakan untuk menangani *missing value*. Kekurangan ini menunjukkan adanya kebutuhan untuk mengintegrasikan metode penanganan *missing value* yang efektif sebelum model prediksi seperti LSTM diterapkan, guna meningkatkan kualitas dan akurasi prediksi.

Berdasarkan uraian di atas, penelitian ini akan berfokus pada tahap *preprocessing data*, terutama dalam menangani *missing value* pada data curah hujan. Teknik interpolasi linear, kuadrat, dan spline kuadratik dipilih untuk mengevaluasi metode yang paling efektif dalam mengisi data yang hilang. Pendekatan ini penting karena kualitas data masukan sangat memengaruhi akurasi model prediksi, sehingga diperlukan metode pengisian *missing value* yang tidak hanya sederhana tetapi juga mampu menangkap pola data secara akurat. Dataset hasil interpolasi kemudian akan digunakan untuk melatih model prediksi *Long Short Term Memory* (LSTM) dan di evaluasi menggunakan metrik *Root Mean Square Error* (RMSE) untuk menentukan metode interpolasi yang paling optimal.

Penelitian ini bertujuan untuk memberikan solusi yang lebih efektif dalam menangani *missing value* pada data curah hujan, sekaligus mengevaluasi teknik interpolasi mana dari ketiga teknik interpolasi yang paling optimal dalam mendukung prediksi curah hujan. Dengan hasil yang diperoleh, diharapkan penelitian ini dapat memberikan kontribusi signifikan dalam pengolahan data *time series*, khususnya dalam meningkatkan akurasi prediksi curah hujan di Kota Jambi.

1.2 Rumusan Masalah

Berdasarkan latar belakang penelitian maka dapat dirumuskan permasalahannya sebagai berikut:

1. Bagaimana hasil interpolasi untuk menangani *missing value* pada data curah hujan dalam penelitian ini?
2. Bagaimana mengimplementasikan metode *Long Short Term Memory* (LSTM) dalam memprediksi curah hujan di Kota Jambi?
3. Bagaimana evaluasi performa dan akurasi metode *Long Short Term Memory* dalam memprediksi curah hujan di Kota Jambi?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui hasil interpolasi untuk menangani *missing value* pada data curah hujan dalam penelitian ini.
2. Mengimplementasikan metode *Long Short Term Memory* (LSTM) dalam memprediksi curah hujan di Kota Jambi.
3. Mengevaluasi performa dan akurasi metode *Long Short Term Memory* (LSTM) dalam memprediksi curah hujan di Kota Jambi.

1.4 Batasan Penelitian

Batasan masalah dari penelitian yang dibahas yaitu sebagai berikut:

1. Data yang digunakan didapatkan dari website Data Online BMKG (Badan Meteorologi, Klimatologi dan Geofisika) dan ukuran curah hujan menggunakan *milimeter*.
2. Penelitian difokuskan pada prediksi curah hujan di Kota Jambi.
3. Data yang digunakan adalah data curah hujan harian Kota Jambi dari tahun 2016-2024.
4. Penelitian ini menggunakan metode interpolasi linear, kuadrat, dan spline kuadratik.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Manfaat Teoritis
   a. Penelitian ini memberikan pemahaman lebih terhadap proses interpolasi dalam menangani *missing value* pada data *time series* dengan menggunakan metode *Long Short Term Memory* (LSTM).
   b. Sebagai bahan referensi untuk penelitian selanjutnya yang berkaitan dengan prediksi curah hujan.

2. Manfaat Praktis
   a. Hasil penelitian ini dapat digunakan sebagai alternatif metode prediksi curah hujan yang melengkapi model yang telah digunakan BMKG, khususnya Kota Jambi.
   b. Penelitian ini dapat menjadi bahan rujukan untuk pengembangan algoritma serupa dan studi terkait prediksi cuaca yang membutuhkan pendekatan yang lebih adaptif terhadap pola data yang kompleks.

II. TINJAUAN PUSTAKA

2.1 Curah Hujan

Hujan adalah proses kondensasi uap air di atmosfer yang berubah menjadi tetesan air cukup berat untuk jatuh ke permukaan bumi. Hujan umumnya terjadi akibat penurunan suhu udara atau peningkatan uap air di atmosfer, dan kedua faktor ini bisa terjadi bersamaan. Proses turunnya hujan juga erat kaitannya dengan kelembapan udara yang memicu peningkatan jumlah titik-titik air di udara. Indonesia, sebagai negara yang dilalui garis khatulistiwa dan sebagian besar wilayahnya beriklim tropis, memiliki beberapa daerah dengan intensitas hujan yang cukup tinggi (Prihandono, 2022).

Curah hujan adalah air hujan yang turun ke permukaan bumi dengan asumsi bahwa permukaan tersebut tidak mengalami penguapan, penyerapan, atau aliran. Satuan untuk mengukur curah hujan adalah *millimeter* (mm), yang diukur menggunakan alat penakar hujan *observatorium* dan penakar hujan jenis *Hellman*. Intensitas curah hujan merujuk pada jumlah curah hujan yang terjadi dalam periode waktu tertentu. BMKG mengklasifikasikan intensitas curah hujan menjadi lima kategori (Carnegie & Chairani, 2023).

Tabel 1. Intensitas Curah Hujan

NO	Kriteria Hujan	Intensitas Curah Hujan Per Jam	Intensitas Curah Hujan Per Hari
1	Sangat Ringan	< 1 mm	< 5 mm
2	Ringan	1 – 5 mm	5 – 20 mm
3	Normal	5 – 10 mm	20 – 50 mm
4	Lebat	10 – 20 mm	50 – 100 mm
5	Sangat Lebat	> 20 mm	> 100 mm

Menurut Nawangwulan (2022), beberapa faktor yang memengaruhi curah hujan adalah sebagai berikut:

1. Jarak dari sumber air: Sumber air, seperti laut, danau, atau sungai, berfungsi sebagai pusat penguapan yang memengaruhi tingkat curah hujan di suatu daerah.
2. Perbedaan suhu antara daratan dan perairan: Ketika suhu di daratan lebih tinggi dibandingkan perairan, hujan cenderung lebih sering terjadi di wilayah perairan, dan sebaliknya.
3. Topografi: Ketinggian wilayah berbanding terbalik dengan curah hujan yang terjadi, karena temperatur udara di wilayah tinggi cenderung lebih rendah.
4. Lintang geografis: Daerah dengan curah hujan rendah biasanya terletak di lintang rendah atau dekat dengan garis khatulistiwa.

5. Arah angin: Angin berfungsi memindahkan awan dari satu wilayah ke wilayah lain. Daerah dengan intensitas angin yang rendah biasanya jarang mengalami hujan.

2.2 Analisis Deret Waktu *(Time Series)*

Analisis deret waktu *(time series)* adalah analisis yang mempertimbangkan pengaruh waktu secara berurutan. Sedangkan data deret waktu itu sendiri adalah data yang dikumpulkan berdasarkan urutan dan interval waktu tertentu, seperti dalam jam, hari, minggu, bulan, kuartal, semester dan tahun (Ruhiat & Suwanda, 2019). Data deret waktu yang merupakan data harian dapat berupa data harga saham, dan laporan cuaca. Data mingguan dapat berupa informasi uang yang beredar. Data kuartalan dapat berupa data PDRB dan data tahunan dapat berupa data anggaran pemerintah (Putri et al., 2017). Tujuan dilakukannya analisis deret waktu ialah untuk dapat memahami dan menjelaskan mekanisme tertentu, meramalkan suatu nilai ataupun kejadian dimasa depan, dan mengoptimalkan sistem kendali (Rofi, 2019).

2.2.1 Pola Data Deret Waktu *(Time Series)*

Data yang terkandung dalam tipe deret waktu dapat dipetakan berdasarkan waktu. Pola data harus diamati untuk selanjutnya menentukan langkah-langkah analisis yang dilakukan. Menurut polanya, data deret waktu dibagi menjadi empat bagian. Data terutama terdiri dari satu atau lebih komponen utama yaitu (Hansun, 2013):

a. Siklus (*Cycles*/C)

Siklus adalah rangkaian fluktuasi atau siklus gelombang yang berlangsung lama dan tidak merata. Durasi siklus ini lebih panjang dibandingkan dengan yang lain yaitu 2 tahun atau lebih. Contoh dari siklus ini yaitu dapat kita lihat data curah hujan dari database internet BMKG.

b. *Irregular* (I)

Irregular merupakan gerakan fluktuasi yang diakibatkan kejadian yang tak terprediksi atau kejadian non-periodik contohnya seperti kejadian bencana alam yang tidak bisa diprediksi kejadiannya.

c. Musiman *(Seasonality/S)*

Data runtun waktu ini menggambarkan pola fluktuasi permintaan yang bergerak naik atau turun di sekitar garis tren setiap tahunnya. Fluktuasi musiman merujuk pada perubahan yang dapat dibagi berdasarkan triwulan, bulanan, mingguan, atau harian, dengan menunjukkan pola yang berubah secara teratur dari waktu ke waktu. Sebagai contoh, jumlah penumpang yang

menggunakan transportasi umum cenderung meningkat pada waktu-waktu tertentu, seperti saat jam sibuk atau musim liburan.

d. *Trend (T)*

Dalam data deret waktu, *trend* merupakan komponen jangka panjang yang menunjukkan kenaikan atau penurunan selama periode tertentu. Secara sederhana, *trend* adalah kurva yang menggambarkan arah umum dari suatu kumpulan data waktu. Contoh *trend* ini meliputi perubahan populasi dunia dan tingkat inflasi.

2.2.2 Tipe Data Analisis Deret Waktu *(Time Series)*

Tipe data *time series* adalah istilah yang digunakan untuk semua jenis data yang memiliki komponen waktu secara berurutan. Data ini merupakan rangkaian pengamatan atau kumpulan data yang tersusun rapi dan berurutan berdasarkan waktu dengan interval yang kontinu. Pemrosesan dan analisis data *time series* membutuhkan keterampilan khusus dan metode yang mendukung guna memahami kerumitan dari deret data tersebut serta menyelidiki proses yang mendasarinya. Tipe data *time series* ini dapat dibagi menjadi dua kategori, yaitu *univariate* dan *multivariate* (Alqahtani et al., 2021).

a. Tipe Data *Univariate*

Tipe data *Univariate* adalah tipe data yang memiliki sebuah deretan data yang hanya mengandung satu nilai data per deret waktu yang primitif (Alqahtani et al., 2018). Tipe data ini hanya melibatkan satu variabel yang direkam atau diamati dari waktu ke waktu, sehingga dianggap sebagai tipe data yang sederhana. Misalnya, dalam pencatatan suhu kota, hanya ada satu variabel yang diamati, yaitu suhu. Penggunaan tipe data ini menekankan variabel dependen yang bergantung pada waktu secara konsisten. Dengan memantau data suhu dari waktu ke waktu, kebijakan dapat disusun untuk mengatasi peningkatan suhu yang drastis, yang menunjukkan bahwa meskipun tipe data ini terbilang primitif, dalam beberapa kasus, kesederhanaannya justru membuatnya lebih fokus dan langsung pada inti masalah.

b. Tipe Data *Multivariate*

Tipe data *multivariate* adalah kumpulan *time series* yang memiliki *timestamp* yang sama dan memungkinkan adanya lebih dari satu variabel (Alqahtani et al., 2018). Tipe data *time series* ini terdiri dari deretan variabel angka pada setiap titik *time series*, dan dapat berupa kumpulan beberapa tipe data *univariate* yang dicatat dari waktu ke waktu. Contoh penerapannya adalah pengembangan dari tipe data *univariate*, seperti pengukuran suhu dan tekanan, di mana beberapa variabel, seperti data suhu, tekanan, dan *timestamp*, dapat

dikumpulkan. Data *time series* merupakan rangkaian pengamatan yang teratur, terdiri dari titik-titik data yang dicatat secara berurutan dari waktu ke waktu. Tipe data *multivariate* khusus ini sangat berguna di bidang-bidang seperti biologi, kedokteran, keuangan, animasi, dan juga relevan di berbagai bidang lainnya (Alqahtani et al., 2021).

2.3 Interpolasi

Interpolasi adalah sebuah teknik untuk mengisi celah dalam sebuah data dengan memperkirakan nilai diantara titik-titik data yang diketahui (Widianti & Pratama, 2024). Menurut Kosasih (2006), Interpolasi adalah cara untuk mendapatkan fungsi yang meliputi semua titik dalam set data diskrit. Dengan kata lain, itu adalah cara untuk memperkirakan atau memperkirakan suatu titik atau nilai di antara titik-titik diskrit atau set data yang sudah diketahui. Ada beberapa jenis metode interpolasi yang dapat digunakan untuk memperkirakan suatu titik yaitu, interpolasi linear, interpolasi kuadrat, dan interpolasi spline kuadratik yang dibahas dalam penelitian ini.

2.3.1 Interpolasi Linear

Interpolasi linear merupakan metode paling sederhana yang digunakan untuk memperkirakan nilai suatu fungsi atau nilai tengah F(X) berdasarkan dua nilai yang telah diketahui, yaitu nilai sebelumnya F(X0) dan nilai sesudahnya F(X1) (Ignasius & Lamabelawa, 2018). Interpolasi linear dapat dirumuskan sebagai berikut:

$$f_1(X) = f(X_0) + \frac{f(X_1) - f(X_0)}{(X_1 - X_0)}(X - X_0) \quad (1)$$

Dimana:

X_0 = Nomor titik awal

X_1 = Nomor titik akhir

X = Nomor yang dicari

$f_1(X)$ = Data yang dicari

$f(X_0)$ = Data titik awal

$f(X_1)$ = Data titik akhir

2.3.2 Interpolasi Kuadrat

Interpolasi kuadrat merupakan pengembangan dari interpolasi linear. Berbeda dengan interpolasi linear yang hanya memerlukan dua nilai, interpolasi kuadrat memanfaatkan tiga nilai atau tiga set data untuk memperkirakan satu nilai. Dimisalkan yakni X_0, X_1, dan X_2 dengan nilai fungsi yakni $f(X_0)$, $f(X_1)$, $f(X_2)$ untuk memperkirakan nilai suatu fungsi atau nilai tengah yang didefinisikan dengan $f(X)$ dari nilai titik X yang telah diketahui (Ignasius & Lamabelawa, 2018). Interpolasi kuadrat dapat dirumuskan sebagai berikut:

$$f(X) = b_0 + b_1 (X - X_0) - b_2 (X - X_0)(X - X_0) \quad (2)$$

Dimana*:*

$$b_0 = f(X_0) \quad (3)$$

$$b_1 = \frac{f(X_1) - f(X_0)}{(X_1 - X_0)} \quad (4)$$

$$b_2 = \frac{\frac{f(X_2) - f(X_1)}{(X_2 - X_1)} - \frac{f(X_1) - f(X_0)}{(X_1 - X_0)}}{X_2 - X_0} \quad (5)$$

2.3.3 Interpolasi Spline Kuadratik

Interpolasi spline kuadratik adalah metode aproksimasi atau metode untuk memperkirakan suatu nilai atau data dengan melakukan interpolasi dari titik-titik x yang terletak diantar dua titik xn dan xn+1dengan mengasumsikan fungsi berbentuk polinomial berpangkat dua. Spline kuadratik merupakan salah satu bentuk spline yang banyak digunakan dalam bidang rekayasa. Dalam fungsi spline kuadratik, setiap pasang titik di interpolasi (Sofiyani & Permanasari, 2023). Formulasi umum untuk interpolasi spline kuadratik adalah sebagai berikut:

$$S_i(x) = a_i x^2 + b_i x + c_i \quad (6)$$

Dimana:

S_i = Spline kuadratik untuk interval ke -1

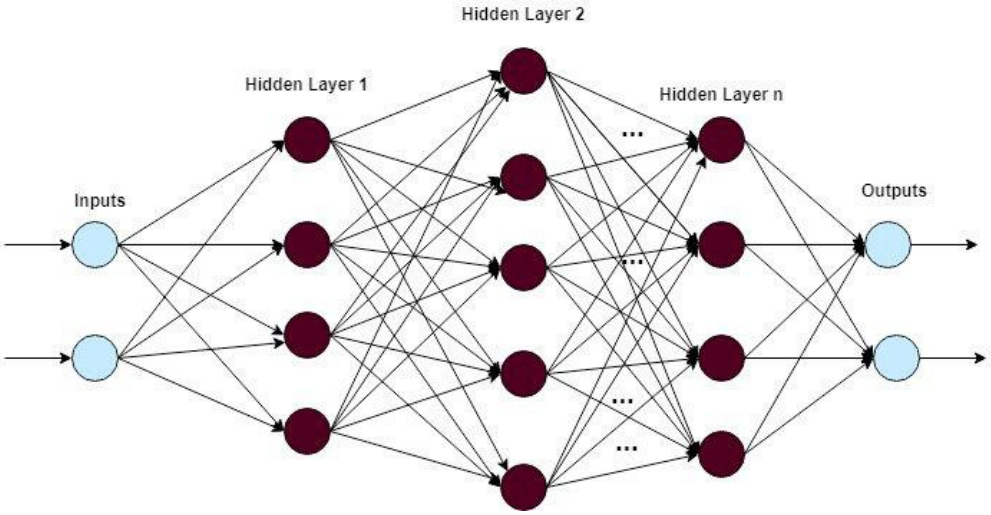
a_i, b_i, c_i= Koefisien yang ditentukan untuk setiap interval

$x_i \; dan \; x_{i+1}$ = Titik data yang berdekatan

2.4 *Deep Learning*

Deep learning merupakan cabang dari pembelajaran mesin yang memanfaatkan jaringan saraf tiruan untuk menyelesaikan masalah dengan menggunakan kumpulan data yang besar (Sutriawan et al., 2023). Salah satu keunggulan *deep learning* adalah kemampuannya dalam menyelesaikan masalah yang kompleks, seperti pengenalan citra, pengenalan suara, serta meniru cara kerja otak manusia melalui jaringan saraf tiruan yang digunakan dalam algoritmanya (Purnama, 2021).

Deep learning adalah algoritma pembelajaran mesin yang berlandaskan pada pembelajaran tanpa pengawasan *(unsupervised learning)* dan terdiri dari beberapa lapisan fungsional serta ekstraksi data. Untuk membentuk representasi hierarkis, fitur-fitur yang lebih tinggi diperoleh dari fitur-fitur yang lebih rendah (Farikhul Firdaus & Paputungan, 2022). Gambar 1 memperlihatkan representasi visual struktur *deep learning* yang terdiri dari 4 lapisan.



Gambar 1. Ilustrasi Struktur *Deep Learning*

Dalam *Deep Learning*, jaringan terdiri dari beberapa layer yang merupakan kumpulan dari *node-node*. *Deep Learning* memiliki lebih banyak lapisan tersembunyi daripada *neural network*. Lapisan tersembunyi dapat mencapai lebih dari tiga lapisan, termasuk lapisan *input* dan *output* atau bahkan mencapai ratusan (Rizki et al., 2020).

Gambar 1 menunjukkan ilustrasi beberapa properti *Deep Learning* yang mengukuhkan statusnya sebagai revolusi AI. Meskipun kita mungkin tidak menggunakan *Neural Network* dalam dua dekade, apa pun yang kita gunakan akan secara langsung mewarisi pembelajaran mendalam kontemporer dan konsep inti. Properti penting ini secara luas dapat diurutkan menjadi tiga kategori (Ningrum et al., 2021):

1. Kesederhanaan

Deep Learning menghilangkan kebutuhan akan rekayasa fitur, menggantikan *pipeline* yang rumit, rapuh, dan banyak rekayasa dengan model sederhana yang dapat dilatih secara menyeluruh yang biasanya dibuat hanya menggunakan lima atau enam operasi tensor yang berbeda.

2. Skalabilitas

Deep learning sangat cocok untuk paralelisasi pada GPU atau TPU, sehingga dapat memanfaatkan sepenuhnya hukum *Moore*.

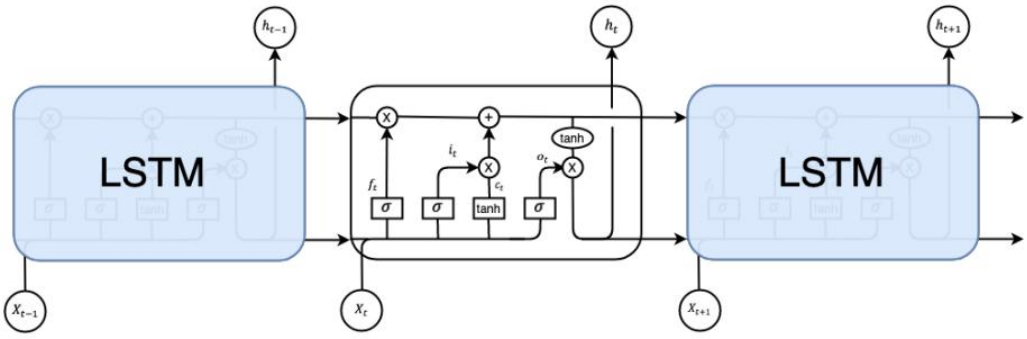
3. Keserbagunaan dan kegunaan kembali

Tidak seperti banyak pendekatan pembelajaran mesin sebelumnya, model *Deep Learning* dapat dilatih pada data tambahan tanpa memulai ulang dari awal, membuatnya layak untuk pembelajaran online berkelanjutan (Ningrum et al., 2021).

2.5 *Long Short Term Memory* (LSTM)

Long Short Term Memory (LSTM) merupakan sebuah evolusi dari arsitektur RNN yang diperkenalkan oleh Horchreiter & Schmidhuber (1997). *Recurrent Neural Network* (RNN) adalah jenis arsitektur jaringan syaraf berulang yang didesain khusus untuk dapat mengatasi permasalahan dengan data berurutan *(sequence data)* (Wiranda & Sadikin, 2019). Seperti RNN, jaringan LSTM *(LSTM network)* juga terdiri dari modul-modul dengan pemrosesan berulang. Singkatnya, LSTM menambahkan sebuah proses seleksi di dalam *cell* (kotak kontrol) nya sehingga bisa menyeleksi informasi mana yang layak untuk diteruskan, sekaligus menjadi solusi bagi permasalahan *vanishing gradient* (Tarkus et al., 2020).

Long Short Term Memory (LSTM) memiliki kelebihan karena mampu mempertahankan sekuens *long term* (ukuran data) yang sulit dilakukan dengan menggunakan pendekatan fitur konvensional (Wiranda & Sadikin, 2019). *Reccurent Neural Network* (RNN) memiliki kekurangan yaitu kemampuan untuk mengelola informasi dalam periode yang lama. Untuk mengatasi permasalahan tersebut, *Reccurent Neural Network* (RNN) mengembangkan *Long Short Term Memory* (LSTM) yang mampu mengelola informasi dalam periode yang lama. *Long Short Term Memory* (LSTM) sangat dikenal dan banyak dipilih untuk prediksi berbasis waktu atau *time series* karena kemampuannya untuk memprediksi dalam waktu yang lama dibanding algoritma lain (Zahara & Ilmiddafiq, 2019).



Gambar 2. Arsitektur Jaringan LSTM

Sumber : (Wiranda & Sadikin, 2019)

Lapisan tersembunyi terdiri dari sel memori, satu sel memori memiliki tiga *gate* yaitu *input gate, forget gate, output gate* (Vinayakumar et al., 2017). *Input gate* berfungsi mengatur berapa jumlah informasi yang harus disimpan dalam keadaan sel, yang akan mencegah sel dari menyimpan data yang tidak perlu. *Forget gate* berfungsi mengatur sejauh mana nilai tetap di dalam sel memori. *Output Gate* berfungsi untuk memutuskan berapa banyak konten atau nilai

dalam sel memori, digunakan untuk menghitung *output* (Oyen, 2018). Tiga lapisan *sigmoid* dan satu lapisan *tanh* hadir di setiap sel memori (Qiu et al., 2020). Fungsi *sigmoid* dapat dilihat pada persamaan berikut (Ma et al., 2015):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

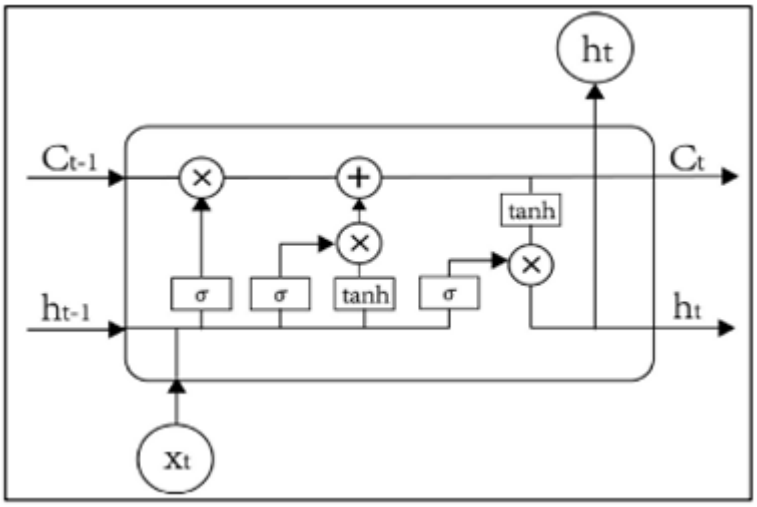
Fungsi *tanh* dapat dilihat sebagai berikut:

$$\tanh(x) = 2\sigma\,(2x) - 1 \tag{8}$$

Keterangan:

σ = Fungsi aktivasi *sigmoid*

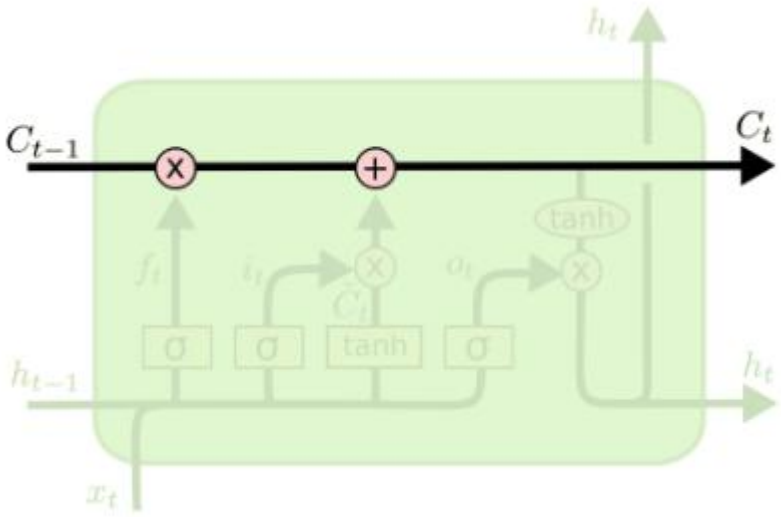
x = Data *input*



Gambar 3. Struktur Dalam Satu Sel LSTM

Sumber : (Qiu et al., 2020)

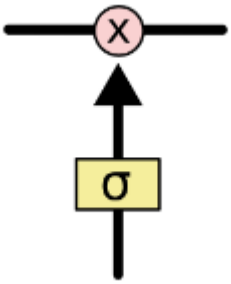
Gambar 2.11 menggambarkan struktur dalam satu sel *Long Short-Term Memory* (LSTM). Kunci dari arsitektur LSTM adalah jalur yang menghubungkan sel memori lama ($Ct-1$) dengan sel memori baru (Ct). Garis *horizontal* yang menghubungkan semua lapisan *output* di LSTM merupakan definisi dari sel memori tersebut. Melalui jalur ini, nilai sel memori lama dapat ditransfer ke sel memori baru dengan upaya yang minimal (Farikhul Firdaus & Paputungan, 2022).



Gambar 4. *Memory Cell*

Sumber : (Colah, 2015)

Salah satu fungsi penting dari *Long Short-Term Memory* (LSTM) adalah memberikan kontrol untuk menambah atau menghapus data dari masa lalu yang akan dimasukkan ke dalam sel saat ini. Seberapa besar setiap komponen yang akan dibiarkan masuk ke sel ditunjukkan dengan nilai antara 0 dan 1 pada lapisan *sigmoid*. Nilai 0 berarti "tidak lolos", sedangkan nilai 1 berarti "lolos" (Colah, 2015).

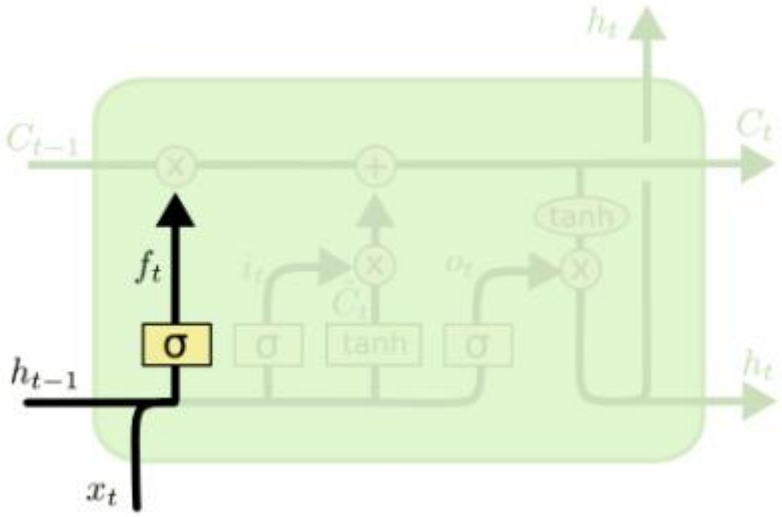


Gambar 5. Lapisan Sigmoid

Sumber : (Colah, 2015)

Proses dalam metode LSTM terdiri dari beberapa langkah, dengan langkah awal berfokus pada *forget gate*. *Forget gate* ini bertanggung jawab untuk menentukan informasi yang akan dihapus dari *cell state*. Proses ini terjadi melalui lapisan *sigmoid*, yang menghasilkan *output* berupa angka antara 0 dan 1. Perhitungan *forget gate* dilakukan dengan memanfaatkan *output* data

sebelumnya ($ht-1$) dan data *input* saat ini (xt), seperti yang terlihat pada Gambar 6 (Colah, 2015).



Gambar 6. Alur *Forget Gate* Pada LSTM

Sumber : (Colah, 2015)

Persamaan *forget gate* diuraikan pada persamaan berikut:

$$f_t = \sigma\left(w_f . [h_{t-1}, x_t] + b_f\right) \quad (9)$$

Keterangan :

f_t = *Forget gate*

σ = Fungsi *Sigmoid*

w_f = Nilai *weight* untuk *forget gate*

h_{t-1} = Nilai output sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_f = Nilai bias pada *forget gate*

Nilai *weight* dirumuskan dengan persamaan berikut:

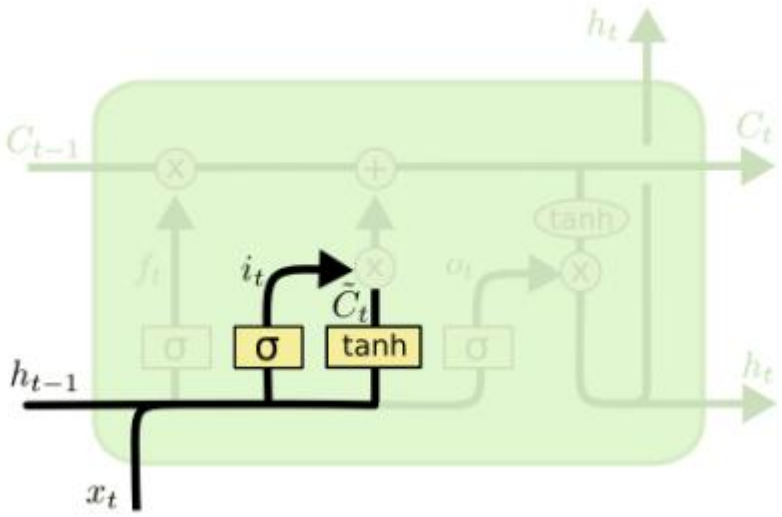
$$W = \left(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right) \quad (10)$$

Keterangan:

W = *Weight*

d = Jumlah variabel

Tahap ini diilustrasikan pada Gambar 7:



Gambar 7. Alur *Input Gate* Pada LSTM

Sumber: (Colah, 2015)

Tahapan kedua adalah *input gate*, yang merupakan proses mengolah informasi untuk menentukan informasi yang akan diperbarui dan kemudian dicadangkan ke dalam *cell state*.

Persamaan pada *input gate* ditulis seperti persamaan berikut:

$$i_t = \sigma\,(w_i\,.\,[h_{t-1}, x_t] + b_i\,) \tag{11}$$

Keterangan:

i_t = *Input gate*

σ = Fungsi *Sigmoid*

w_i = Nilai *weight* untuk *Input gate*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_i = Nilai bias pada *Input gate*

Persamaan Kandidat baru dituliskan sebagai berikut:

$$c_t^- = tanh\,(w_c\,.\,[h_{t-1}, x_t] + b_c\,) \tag{12}$$

Keterangan:

c_t^- = Nilai baru yang dapat ditambahkan ke *cell state*

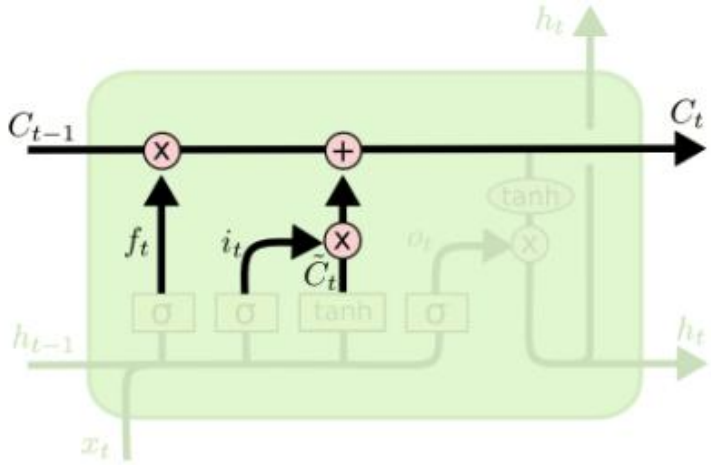
$tanh$ = Fungsi *Tangen Hiperbolik*

w_c = Nilai *weight* untuk *cell state*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_c = Nilai bias pada *cell state*



Gambar 8. Alur Memperbaharui *Cell State* Pada LSTM

Sumber: (Colah, 2015)

Langkah ketiga adalah pembaruan *cell state*, yang dijelaskan pada Gambar 8. Dalam ilustrasi tersebut, pembaruan *cell state* terjadi dengan mengalikan *output* dari *forget gate* dengan *cell state* sebelumnya, lalu hasilnya ditambahkan dengan hasil dari langkah kedua, yaitu $i_t \ * C_t^-$. Persamaan untuk pembaruan *cell state* tersebut ditampilkan pada persamaan berikut:

$$C_t = f_{t-1} . C_{t-1} + i_t * C_t^- \quad (13)$$

Keterangan:

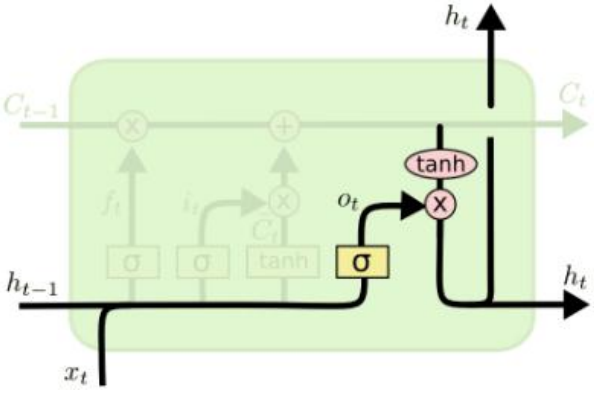
C_t = *Cell state*

f_t= *Forget Gate*

C_{t-1} = *Cell state* sebelum orde ke t

i_t = *Input Gate*

C_t^- = Nilai baru yang dapat ditambahkan ke *Cell state*



Gambar 9. Alur *Output* Pada LSTM

Sumber: (Colah, 2015)

Langkah terakhir dalam metode LSTM adalah menentukan hasil *output*, sebagaimana dapat dilihat pada Gambar 9. Lapisan *sigmoid* menentukan bagian dari *cell state* yang akan dijadikan *output*. Operasi *Output Gate* digambarkan dalam persamaan berikut.

$$o_t = \sigma\,(w_o\,.[h_{t-1}, x_t] + b_o\,) \tag{14}$$

Keterangan:

o_t = *Output gate*

σ = Fungsi *Sigmoid*

w_o = Nilai *weight* untuk *Output gate*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_o = Nilai bias pada *Output gate*

Setelah diperoleh nilai dari *Output gate*, kemudian masukkan *cell state* melalui fungsi *tanh* untuk memperoleh nilai dalam rentang -1 hingga 1. Lalu, nilai tersebut dikalikan dengan *output gate* dari lapisan *sigmoid*. Persamaan untuk nilai *output* pada orde t dapat disajikan dalam persamaan berikut.

$$h_t = o_t * \tanh(\,C_t\,) \tag{15}$$

Keterangan:

h_t = Nilai *output* orde t

o_t = *Output gate*

tanh = Fungsi *tangen hiperbolik*

C_t = *Cell state*

2.6 *Root Mean Square Error* (RMSE)

Root Mean Square Error (RMSE) adalah rumus yang digunakan untuk memperkirakan suatu pengamatan dan menilai penyimpangan dari nilai yang diharapkan dari suatu model. RMSE dihitung dengan mengambil akar kuadrat dari kesalahan kuadrat rata-rata. Semakin rendah nilai RMSE, menunjukkan semakin akurat metode estimasi kesalahan pengukuran yang digunakan (Prasetyo et al., 2021).

Persamaan untuk RMSE dapat dijelaskan melalui persamaan berikut:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \tag{16}$$

Keterangan:

Y_i = Nilai aktual

$\hat{Y}_i$ = Nilai hasil peramalan / prediksi

n = Jumlah data

Tabel 2. Pengelompokkan nilai RMSE

RMSE	Kategori
<5	Performa model prediksi akurat
5-15	Performa model prediksi baik
15-50	Performa model prediksi layak
>50	Performa model prediksi tidak akurat

2.7 Penelitian Terdahulu

Penelitian terdahulu merupakan penelitian yang berfungsi sebagai bentuk perbandingan penelitian yang akan dilakukan dengan penelitian yang sebelumnya sudah pernah ada (Irawan, 2024). Pada bagian penelitian terdahulu ini berisi penelitian-penelitian yang telah dilakukan sebelumnya dan menjadi sumber rujukan pada penelitian ini.

Tabel 3. Penelitian Terdahulu

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
1	(Freecenta et al., 2022)	Prediksi Curah Hujan Di Kab. Malang Menggunakan LSTM (*Long Short Term Memory*)	*Long Short Term Memory* (LSTM)	Hasil dari penelitian ini menunjukkan bahwa skenario uji coba pertama menggunakan empat layer LSTM dengan masing-masing 100 *neuron*. Skenario uji coba kedua memiliki dua layer LSTM, dengan masing masing 50 *neuron*. Dalam penelitian ini, nilai akurasi model terbesar adalah MAE sebesar 7,90, RMSE sebesar 10,16, dan MSE sebesar 103,37.
2	(Irawan, 2024)	Prediksi Curah Hujan Menggunakan	*Recurrent Neural Network*	Hasil dari penelitian ini menunjukkan bahwa metode LSTM dan RNN

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
		Recurrent Neural Network (RNN) Dan *Long Short-Term Memory* (LSTM)	(RNN) Dan *Long Short-Term Memory* (LSTM)	memiliki akurasi prediksi yang telah dievaluasi menggunakan MAPE dan RMSE dengan pembagian data 90%, 80%, 70%, 60% data *Training* dan 10%, 20%, 30%, 40% data *Testing* metode LSTM lebih baik dari pada RNN dengan rata-rata nilai MAPE LSTM 0.29%, 4.02%, 4.05%, 2.21% RMSE LSTM 2.44%, 7.53%, 3.96%, 4.21% dan MAPE RNN 0.31%, 2.33%, 4.22%, 2.23%, nilai RMSE RNN 2.52%, 7.69%, 3.97%, 4.26% pengujian paling baik metode LSTM ada pada pembagian data *Training* 60% dan *testing* 40% dengan rata-rata nilai MAPE LSTM 2.21% dan RMSE LSTM 4.21%.
3	(Wiranda & Sadikin, 2019)	Penerapan *Long Short Term Memory* pada Data *Time Series* untuk Memprediksi Penjualan	*Long Short Term Memory* (LSTM)	Penggunaan LSTM untuk prediksi penjualan produk dengan parameter evaluasi MAPE dan RMSE. LSTM menunjukkan akurasi yang tinggi dengan

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
		Produk di PT. Metiska Farma		RMSE sebesar 13.762.154,00 dan MAPE sebesar 12% untuk skenario terbaik.
4	(Zahara & Ilmiddafiq, 2019)	Prediksi Indeks Harga Konsumen Menggunakan Metode *Long Short Term Memory* (LSTM) Berbasis *Cloud Computing*	*Long Short Term Memory* (LSTM)	Hasil dari penelitian ini didapatkan bahwa Implementasi metode LSTM dalam prediksi yang bersifat *time-series* sangat tepat digunakan karena LSTM mampu mengingat dan menyimpan *history* data baik *short term* atau jangka pendek, maupun *long term* atau jangka panjang. Dan dari 7 jenis algoritma optimasi, *Nesterov Adam* (Adam) mempunyai tingkat akurasi paling tinggi dibanding dengan algoritma lainnya dengan nilai RMSE terkecil yaitu 4.088.
5	(Rizki et al., 2020)	Implementasi *Deep Learning* Menggunakan Arsitektur *Long Short Term Memory* Untuk Prediksi Curah Hujan Kota Malang	*Long Short Term Memory* (LSTM)	Setelah melalui tahap analisis, perancangan, implementasi, dan pengujian, hasil prediksi menunjukkan tingkat akurasi yang baik. Hal ini menandakan bahwa arsitektur *Deep*

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
				Learning berbasis LSTM dapat berfungsi secara optimal pada tahap akhir penelitian.
6	(Farikhul Firdaus & Paputungan, 2022)	Prediksi Curah Hujan di Kota Bandung Menggunakan *Metode Long Short Term Memory*	*Long Short Term Memory* (LSTM)	Model LSTM menunjukkan hasil yang cukup akurat dalam memprediksi curah hujan di Bandung dengan akurasi tertinggi diperoleh dengan *epoch* 50 serta *batch size* 1 dan nilai RMSE terbaik yaitu *Train Score* 12.24 dan *Test Score* 8.86.
7	(Ningrum et al., 2021)	Algoritma *Deep Learning*-Lstm Untuk Memprediksi Umur *Transformator*	*Long Short Term Memory* (LSTM)	Hasil dari penelitian ini menunjukkan bahwa metode *Deep Learning*-LSTM mempunyai kinerja yang lebih baik daripada 3 algoritma lain yaitu nilai RMSE= 0,0004 dan nilai *Squared Correlation*= 0,9690.
8	(Rizky Ismail et al., 2023)	Perhitungan Data Curah Hujan yang hilang dengan Menggunakan Metode Interpolasi Linier	Interpolasi Linear	Interpolasi linier menghasilkan data curah hujan yang valid dengan rata-rata korelasi 0,85%, meskipun ada selisih cukup besar pada beberapa bulan tertentu.

No	Penulis, Tahun	Judul Penelitian	Metode	Hasil
9	(Ignasius & Lamabelawa, 2018)	Analisis Perhitungan Metode Interpolasi Pada Data *Time Series* Kemiskinan di NTT.	Interpolasi	Hasil dari penelitian ini menunjukkan bahwa Perhitungan yang dilakukan terhadap 21 data deret waktu menunjukkan bahwa nilai *Mean Square Error* (MSE) dan *Mean Average Percentage Error* (MAPE) untuk interpolasi linear lebih baik daripada interpolasi kuadratik dan interpolasi Newton.
10	(Febrianti, 2019)	Pemodelan *Autoregressive Integrated Moving Average* (ARIMA) dengan Data Hilang Melalui Metode Interpolasi	Interpolasi	Pada penelitian ini untuk contoh tiga data hilang dengan ordo 2, model ARIMA yang paling layak digunakan yaitu model ARIMA (0,1,1) dengan nilai MSE = 267026.

III. METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian ini dilaksanakan di Program Studi Sistem Informasi, Jurusan Teknik Elektro dan Informatika, Fakultas Sains dan Teknologi, Universitas Jambi. Jl. Jambi-Muara Bulian No. KM. 15, Mendalo Darat, Kec. Jambi Luar Kota, Kabupaten Muaro Jambi. Penelitian ini dimulai sejak proposal penelitian disetujui dan disahkan melalui seminar proposal. Adapun rentang jadwal penelitian yang telah dilakukan sebagai berikut:

Tabel 4. Jadwal Penelitian

NO	Kegiatan	Waktu yang digunakan						
		Nov 2024	Des 2024	Jan 2025	Feb 2025	Mar 2025	Apr 2025	Mei 2025
1	Membuat proposal dan bimbingan							
2	Seminar proposal							
3	Penelitian							
	a. Pengumpulan data							
	b. Pengolahan data							
4	Membuat analisis data penelitian							
5	Ujian Skripsi							

3.2 Perangkat Penelitian

Perangkat keras *(Hardware)* dan perangkat lunak *(Software)* yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Perangkat keras *(Hardware)*
   a. Laptop Acer dengan *processor* 11th Gen Intel(R) *Core*(TM) i5-1135G7 @ 2.40GHz 2.42 GHz, RAM 16 GB, dan system type 64-bit *operating system*, x64-*based processor*.
2. Perangkat lunak *(Software)*
   a. Sistem Operasi Windows 11.
   b. Anaconda Navigator sebagai platform distribusi Python yang menyediakan *environment* management system dan *package* management system. Perangkat lunak ini memudahkan pengelolaan

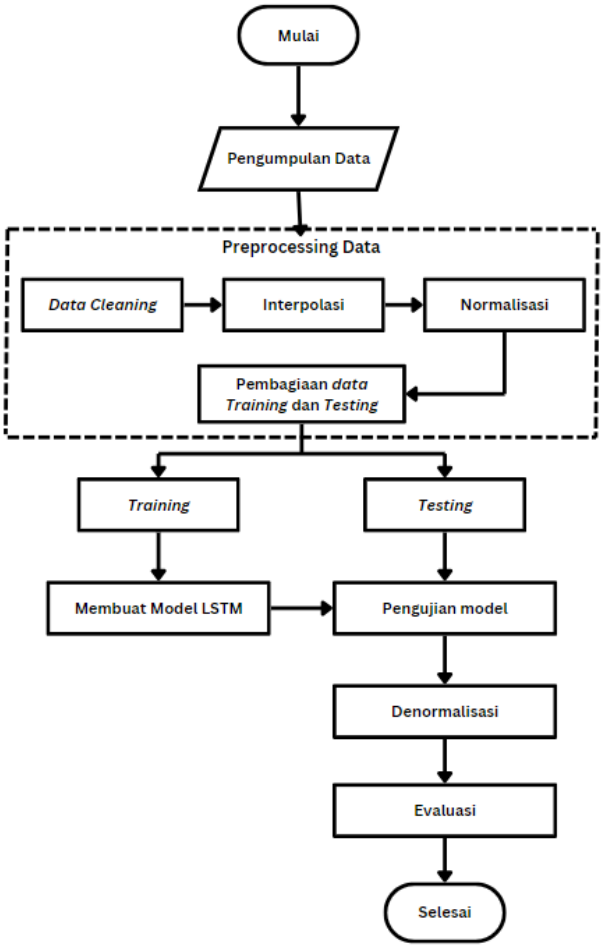
library dan *dependencies* yang diperlukan dalam pengembangan model LSTM, serta menjamin konsistensi versi *packages* yang digunakan.

c. Jupyter Lab sebagai *Integrated Development Environment* (IDE) dalam penelitian ini. Platform ini dipilih karena menyediakan antarmuka interaktif yang memungkinkan pengembangan dan pengujian kode secara *real-time*, visualisasi data dan hasil prediksi secara langsung, dokumentasi kode yang terstruktur dengan format *markdown*, dan eksekusi kode secara modular melalui sistem *cell-based execution.*

d. Microsoft Excel 2019 sebagai *tool* untuk penyimpanan dan pengorganisasian data.

3.3 Tahapan Penelitian

Penelitian ini memerlukan kerangka penelitian yang jelas untuk memudahkan proses pelaksanaannya. Kerangka penelitian berfungsi untuk menjelaskan langkah-langkah yang akan diambil dalam menyelesaikan masalah penelitian ini. Langkah-langkah tersebut dapat dilihat pada gambar di bawah ini.

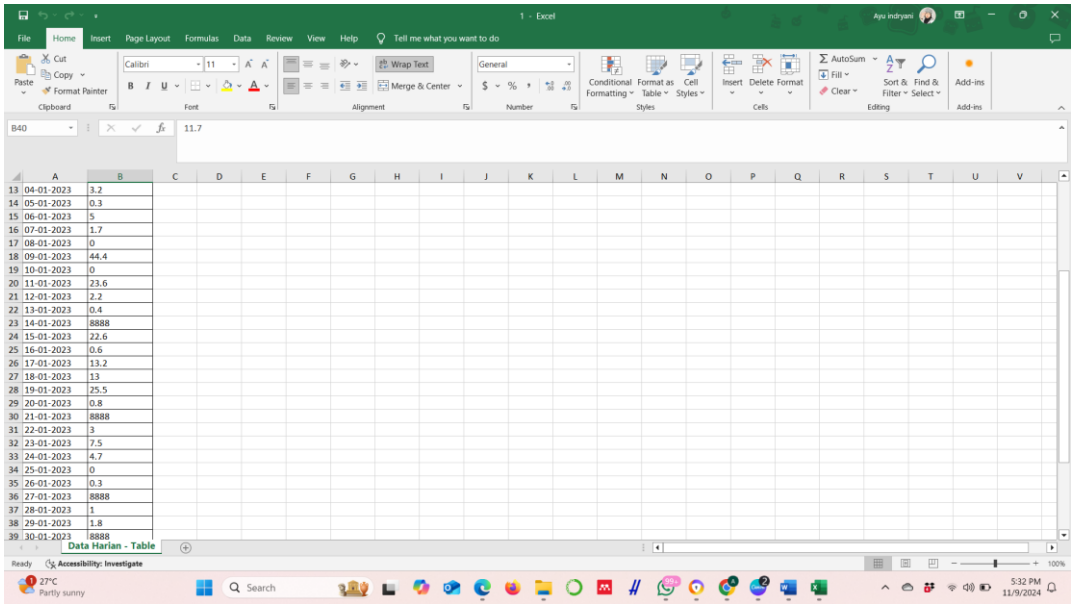


Gambar 10. Tahapan Penelitian

3.3.1 Pengumpulan Data

Tahap pengumpulan data merupakan langkah awal yang krusial dalam penelitian ini. Data yang dibutuhkan didapat dari Website Data Online BMKG (https://dataonline.bmkg.go.id/home), data yang dikumpulkan yaitu data curah hujan dari tahun 2016 - 2024 di Kota Jambi sebanyak 3.288 data yang contohnya dapat dilihat pada gambar 12. Data ini mencakup informasi curah hujan harian yang diukur dalam satuan *millimeter* (mm). Pengumpulan data yang akurat dan komprehensif sangat penting karena akan mempengaruhi kualitas prediksi yang dihasilkan oleh model.

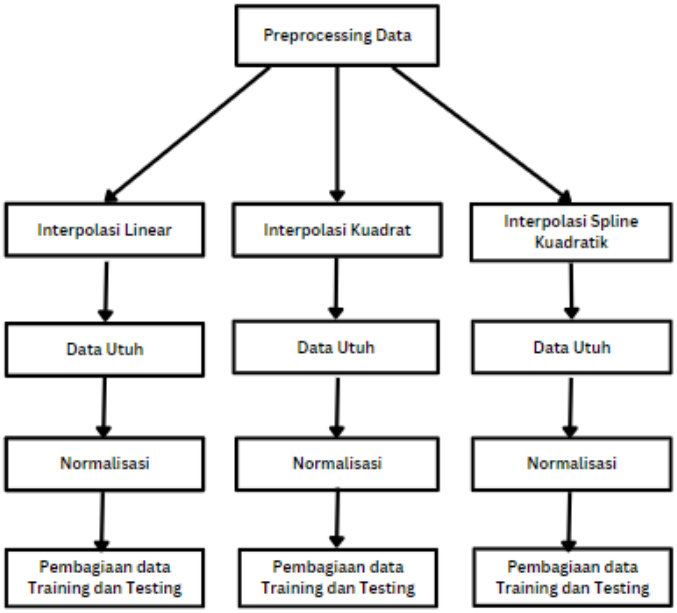
Gambar 11. Tampilan Home Web Data Online BMKG



Gambar 12. Tampilan Dataset di Excel

3.3.2 *Preprocessing* Data

Tahap berikutnya adalah proses *prepocessing* data, pada proses ini perlu dilakukannya pembersihan data *(data cleaning)*. *Preprocessing* data merupakan tahap pengolahan data mentah menjadi format yang siap digunakan untuk pemodelan. Tahap ini meliputi beberapa proses penting:



Gambar 13. *Preprocessing Data*

1. *Data Cleaning*

Dalam proses *data cleaning*, Langkah yang dilakukan adalah identifikasi *missing value* dalam dataset curah hujan yang digunakan. Dalam hal ini, nilai 8888 yang berarti nilai curah hujan yang tidak terukur atau <0,5 dan nilai 9999 yang berarti data belum masuk atau alat rusak, perlu ditangani. Untuk mempermudah proses analisis lebih lanjut, nilai 8888 dan 9999 diubah menjadi nilai kosong (NaN). Konversi ini akan memudahkan penerapan teknik Interpolasi untuk mengatasi nilai hilang tersebut. Dengan mengubah nilai tidak terukur menjadi NaN, dataset dapat diproses dengan lebih efektif dan penerapan teknik interpolasi linear, kuadrat, dan spline kuadratik yang direncanakan untuk menghasilkan prediksi curah hujan yang lebih akurat di Kota Jambi. Berikut adalah contoh data sebelum dilakukan proses *data cleaning* pada bulan Desember 2023:

Tabel 5. Data Mentah Periode Desember 2023

Tanggal	RR
01-12-2023	26.9

02-12-2023	22.2
03-12-2023	11.8
04-12-2023	4.1
05-12-2023	0
06-12-2023	6.6
07-12-2023	3
08-12-2023	
09-12-2023	0.7
10-12-2023	18.8
11-12-2023	5.6
12-12-2023	3.5
13-12-2023	0.9
14-12-2023	0
15-12-2023	0
16-12-2023	0
17-12-2023	0.3
18-12-2023	4.6
19-12-2023	0
20-12-2023	11.1
21-12-2023	1.4
22-12-2023	23.8
23-12-2023	15.1
24-12-2023	0
25-12-2023	43.4
26-12-2023	0
27-12-2023	
28-12-2023	
29-12-2023	8888
30-12-2023	2.5
31-12-2023	3.6

Setelah dilakukan proses identifikasi *missing value* diperoleh bahwa pada tanggal 29-12-2023 terdapat nilai 8888 yang harus diubah menjadi nilai kosong (NaN) sebelum melakukan proses selanjutnya yaitu interpolasi.

Tabel 6. Data Setelah Proses *Data Cleaning*

Tanggal	**RR**
01-12-2023	26.9
02-12-2023	22.2
03-12-2023	11.8
04-12-2023	4.1
05-12-2023	0
06-12-2023	6.6
07-12-2023	3

08-12-2023	
09-12-2023	0.7
10-12-2023	18.8
11-12-2023	5.6
12-12-2023	3.5
13-12-2023	0.9
14-12-2023	0
15-12-2023	0
16-12-2023	0
17-12-2023	0.3
18-12-2023	4.6
19-12-2023	0
20-12-2023	11.1
21-12-2023	1.4
22-12-2023	23.8
23-12-2023	15.1
24-12-2023	0
25-12-2023	43.4
26-12-2023	0
27-12-2023	
28-12-2023	
29-12-2023	
30-12-2023	2.5
31-12-2023	3.6

2. Interpolasi

Dalam penelitian ini, tahapan interpolasi dilakukan untuk menangani *missing values* pada data curah hujan yang teridentifikasi. Pertama, data yang mengandung nilai 8888 dan 9999 akan diubah menjadi NaN menggunakan teknik *data cleaning*. Selanjutnya, tiga metode interpolasi akan diterapkan: interpolasi linear, interpolasi kuadrat, dan interpolasi spline kuadratik. Pada tahap interpolasi linear, nilai yang hilang diperkirakan berdasarkan dua titik data yang berdekatan, memberikan pendekatan sederhana namun efektif. Interpolasi kuadrat, di sisi lain, menggunakan tiga titik data untuk memberikan estimasi yang lebih halus. Terakhir, interpolasi spline kuadratik akan digunakan sebagai pendekatan yang lebih kompleks dengan memanfaatkan fungsi polinomial pangkat dua untuk mengisi nilai yang hilang. Setelah proses interpolasi selesai, dataset yang telah diperbaiki akan dinormalisasi.

Contoh Perhitungan Interpolasi:

$$f_1(X) = f(X_0) + \frac{f(X_1) - f(X_0)}{(X_1 - X_0)}(X - X_0) \tag{17}$$

Dimana:

X_0 = Nomor titik awal

X_1 = Nomor titik akhir

X = Nomor yang dicari

$f_1(X)$ = Data yang dicari

$f(X_0)$ = Data titik awal

$f(X_1)$ = Data titik akhir

Contoh Perhitungan:

$$f_1(X) = 3 + \frac{(0{,}7 - 3)}{(3 - 1)}(2 - 1) = 1{,}85$$

Tabel 7. Hasil Interpolasi Linear

Tanggal	RR
01-12-2023	26.9
02-12-2023	22.2
03-12-2023	11.8
04-12-2023	4.1
05-12-2023	0
06-12-2023	6.6
07-12-2023	3
08-12-2023	1.85
09-12-2023	0.7
10-12-2023	18.8
11-12-2023	5.6
12-12-2023	3.5
13-12-2023	0.9
14-12-2023	0
15-12-2023	0
16-12-2023	0
17-12-2023	0.3
18-12-2023	4.6
19-12-2023	0
20-12-2023	11.1
21-12-2023	1.4
22-12-2023	23.8
23-12-2023	15.1
24-12-2023	0
25-12-2023	43.4
26-12-2023	0
27-12-2023	0
28-12-2023	0
29-12-2023	0
30-12-2023	2.5
31-12-2023	3.6

3. Normalisasi data

Data dinormalisasi menggunakan metode *Min-Max Scaling* untuk mengubah rentang nilai menjadi skala seragam antara 0 dan 1. Tujuannya adalah untuk menghasilkan skala nilai yang konsisten antar data, baik sebelum maupun sesudah proses. Metode ini menggunakan persamaan berikut:

$$X' = \frac{X - Xmin}{Xmax - Xmin} \quad (18)$$

X' = Hasil Normalisasi Data

$Xmin$ = Data minimum dari data X

$Xmax$ = Data maksimum dari data X

Berikut adalah pengaplikasian *Min Max Scaling* data curah hujan bulan Desember 2023 pada penelitian ini:

$$X' = \frac{26{,}9 - 0}{43{,}4 - 0} = 0{,}6195$$

Tabel 8. Data Hasil Normalisasi

Tanggal	RR
01-12-2023	0,61981
02-12-2023	0,51152
03-12-2023	0,27251
04-12-2023	0,09447
05-12-2023	0
06-12-2023	0,15207
07-12-2023	0,06912
08-12-2023	0,04262
09-12-2023	0,01612
10-12-2023	0,43317
11-12-2023	0,12903
12-12-2023	0,08064
13-12-2023	0,02073
14-12-2023	0
15-12-2023	0
16-12-2023	0
17-12-2023	0,00691
18-12-2023	0,10599
19-12-2023	0
20-12-2023	0,25576
21-12-2023	0,03225
22-12-2023	0,54838
23-12-2023	0,34792
24-12-2023	0

25-12-2023	1
26-12-2023	0
27-12-2023	0
28-12-2023	0
29-12-2023	0
30-12-2023	0,05760
31-12-2023	0,08294

4. Data *Training* dan *Testing*

Pembagian data atau *data split* adalah proses membagi dataset menjadi subset yang saling eksklusif untuk keperluan pelatihan *(training)* dan pengujian *(testing)* model. Tujuan utama pembagian data adalah untuk memberikan gambaran yang lebih akurat tentang seberapa baik model akan bekerja saat diterapkan pada data baru. Dalam pembagian data dengan perbandingan 80:20, sebanyak 80% dari data digunakan untuk melatih model *(training set)*, sedangkan 20% sisanya digunakan untuk menguji model *(testing set)*. Perbandingan ini merupakan standar umum yang sering digunakan dalam praktik *machine learning* (Maulana, 2024).

Untuk menghindari *overfitting*, yaitu kondisi di mana model terlalu mempelajari pola dari data latih dan gagal melakukan generalisasi yang efektif pada data yang belum pernah dilihat sebelumnya, pembagian data menjadi data uji dan data latih sangat penting. Dengan melakukan ini, penulis dapat mengevaluasi kinerja model secara objektif dan menentukan apakah model dapat mengeneralisasi pada data baru.

3.3.3 Membuat Model LSTM

Setelah dilakukannya *preprocessing* dataset, data tersebut di proses dengan model yang diinginkan. Model yang digunakan untuk melakukan prediksi curah hujan pada penelitian ini yaitu *Long Short Term Memory* (LSTM) dengan masing-masing pengisian *missing value.* Proses ini mencakup pengaturan awal bobot untuk *forget gate, input gate, cell state, dan output gate,* serta pengaturan awal nilai bias untuk masing-masing gerbang (Wf, Wi, Wct, Wo dan bias bf, bi, bct, bo). Penulis menggunakan Persamaan (19) untuk menentukan nilai bobot W. Bobot yang diperoleh melalui Persamaan (19) kemudian digunakan sebagai bobot awal untuk Wf, Wi, Wct, dan Wo.

$$W = \left(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right) \tag{19}$$

Keterangan:

W = *Weight*

d = Jumlah variabel

Contoh perhitungan:

$$W = \left(-\frac{1}{\sqrt{31}}, \frac{1}{\sqrt{31}}\right) = (-0{,}17960, 0{,}17960)$$

Nilai d=31 didapatkan dari banyaknya hari dibulan Desember 2023.

LSTM memiliki sejumlah *gates* yang terdiri dari lapisan *sigmoid* dan operasi *pointwise*. *Gates* ini berfungsi untuk menambahkan atau menghapus informasi yang akan diteruskan atau dihentikan. *Output* dari lapisan *sigmoid* dibatasi dalam rentang [0,1], dimana nilai 0 menandakan bahwa informasi tidak akan diteruskan, sedangkan nilai 1 menunjukkan bahwa informasi akan dilanjutkan. Persamaan *sigmoid* dapat ditemukan pada Persamaan berikut:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (20)$$

a. Perhitungan *Forget Gate*

Forget gate adalah gerbang pertama tempat informasi masuk untuk diproses menggunakan algoritma LSTM. Gerbang ini berfungsi menentukan informasi mana yang akan dihapus dari *cell state*. Keputusan ini dibuat oleh lapisan *sigmoid* yang menghasilkan *output* berupa nilai antara 0 dan 1. Persamaan untuk *forget gate* adalah sebagai berikut:

$$f_t = \sigma\left(w_f . [h_{t-1}, x_t] + b_f\right) \quad (21)$$

Keterangan:

f_t = *Forget gate*

σ = Fungsi *Sigmoid*

w_f = Nilai *weight* untuk *forget gate*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_f = Nilai bias pada *forget gate*

Contoh Perhitungan:

$$f_t = \sigma\,(0{,}17960 . [0{,}0.61981] + 1\,)$$
$$f_t = \sigma\,(1{,}11131)$$
$$f_t = \frac{1}{1 + e^{-1{,}11131}} = 0{,}75237$$

Nilai 0,17960 adalah nilai W (bobot) yang sudah dihitung sebelumnya. Xt adalah nilai variabel yang didapatkan dari dataset dan bf adalah nilai bias *forget gate.*

Forget Gate	1,11131
	0,75237

Nilai 1,11131 adalah nilai hasil dari perhitungan *forget gate* sebelum memasuki fungsi *sigmoid*, sedangkan 0,75237 adalah nilai *forget gate* setelah diproses dengan fungsi *sigmoid*.

b. Perhitungan *Input Gate*

Input gate berperan mengambil *output* sebelumnya dan *input* baru serta melewatkan mereka melalui lapisan *sigmoid*. *Gate* ini mengembalikan nilai 0 atau 1.

$$i_t = \sigma\,(w_i\,.\,[h_{t-1}, x_t] + b_i\,) \quad (22)$$

Keterangan:

i_t = *Input gate*

σ = Fungsi *Sigmoid*

w_i = Nilai *weight* untuk *Input gate*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_i = Nilai bias pada *Input gate*

Contoh Perhitungan:

$$i_t = \sigma\,(0{,}17960\,.\,[0{,}0.61981] + 0{,}5\,)$$
$$i_t = \sigma(0{,}61131)$$
$$i_t = \frac{1}{1 + e^{-0{,}61131}} = 0{,}64824$$

Nilai 0,17960 adalah nilai W (bobot) yang sudah dihitung menggunakan persamaan sebelumnya. Xt adalah nilai dari variabel yang didapatkan dari dataset dan *bi* adalah nilai bias *input gate*.

Input Gate	0,61131
	0,64824

Nilai 0,61131 adalah hasil dari *input gate* sebelum memasuki fungsi *sigmoid*, sedangkan 0,64824 adalah hasil *input gate* setelah diproses dengan fungsi *sigmoid*.

c. Perhitungan *Cell State*

Cell state berperan mengangkut informasi melalui seluruh rantai sel, memungkinkan jaringan untuk menyimpan informasi untuk periode waktu yang panjang. Berikut adalah persamaan *cell state*.

$$\bar{c_t} = tanh\,(w_c\,.\,[h_{t-1}, x_t] + b_c\,) \quad (23)$$

Keterangan:

$\bar{c_t}$ = Nilai baru yang dapat ditambahkan ke *cell state*

$tanh$ = Fungsi Tangen Hiperbolik

w_c = Nilai *weight* untuk *cell state*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_c = Nilai bias pada *cell state*

Contoh Perhitungan:

$$c_t^- = \tanh(0{,}17960\,.\,[0, 0.61981] + 0\,)$$

$$c_t^- = \tanh(0{,}11131\,) = 0{,}11085$$

Selanjutnya yaitu pembaharuan *cell state* dengan persamaan berikut:

$$C_t = f_{t-1}\,.\,C_{t-1} + i_t * C_t^- \quad (24)$$

Contoh Perhitungan:

$$C_t = 0{,}75237\,.\,0 + 0{,}64824 * 0{,}11085$$
$$C_t = 0{,}07185$$

Nilai 0,11085 adalah hasil dari perhitungan *tanh*, dan nilai 0,07185 adalah hasil dari perhitungan *cell state*.

d. Perhitungan *Output gate*

Output gate mengontrol seberapa banyak *state* yang lewat ke *output* dan bekerja dengan cara yang sama dengan *gate* lainnya. Dan terakhir menghasilkan *cell state* yang baru (*ht*).

$$o_t = \sigma\,(w_o\,.\,[h_{t-1}, x_t] + b_o\,) \quad (25)$$

Keterangan:

o_t = *Output gate*

σ = Fungsi *Sigmoid*

w_o = Nilai *weight* untuk *Output gate*

h_{t-1} = Nilai *output* sebelum orde ke t

x_t = Nilai *input* pada orde ke t

b_o = Nilai bias pada *Output gate*

Contoh Perhitungan:

$$o_t = \sigma\,(0{,}17960\,.\,[0{,}0.61981] + 0{,}1\,)$$

$$o_t = \sigma\,(0{,}21131)$$

$$o_t = \frac{1}{1 + e^{-0{,}21131}} = 0{,}55263$$

Nilai 0,17960 adalah nilai W (bobot) yang telah dihitung dengan persamaan sebelumnya. Xt adalah nilai dari variabel yang didapatkan dari dataset dan *bo* adalah nilai bias *output gate.*

Output Gate	0,21131

	0,55263

Nilai 0,21131 adalah nilai hasil dari perhitungan *output gate* sebelum memasuki fungsi *sigmoid*, sedangkan 0,55263 adalah nilai *output gate* setelah diproses dengan fungsi *sigmoid*.

e. Perhitungan *Hidden State*

Hidden state merupakan hasil akhir ataupun prediksi yang telah diproses di setiap gerbang LSTM. Hasil *hidden state* ini merupakan nilai prediksi. Berikut adalah persamaan untuk *hidden state*:

$$h_t = o_t * \tanh(C_t) \quad (26)$$

Keterangan:

h_t = Nilai *output* orde t

o_t = *Output gate*

tanh = Fungsi *tangen hiperbolik*

C_t = *Cell state*

Contoh Perhitungan:

$h_t = 0{,}55263 * \tanh(0{,}7185)$

$h_t = 0{,}03963$

Tabel 9. Nilai Perhitungan *Gate* LSTM pada 1 Desember 2023

Forget Gate	1,11131
	0,75237
Input Gate	0,61131
	0,64824
c_t^-	0,11131
	0,11085
Output Gate	0,21131
	0,55263
Cell State	0,07185
Hidden State	0,03963

Dengan menggunakan sampel data yang diambil pada tanggal 1 Desember 2023, proses perhitungan LSTM menggunakan rumus-rumus telah diuraikan satu per satu di sini. Hasil dari setiap *gate* ditunjukkan dalam Tabel 5. Dalam kasus ini, keadaan sel merupakan komponen penting dari metode LSTM. Semua output lapisan LSTM terhubung melalui garis horizontal yang ditemukan di bagian atas diagram.

3.3.4 Pengujian Model LSTM

Pengujian model dilakukan untuk menilai kemampuan prediksi model LSTM yang telah dilatih. Pada tahap ini, model digunakan untuk melakukan prediksi curah hujan dengan memanfaatkan data *testing* dan *training* yang telah disiapkan. Nilai prediksi yang dihasilkan dibandingkan dengan nilai aktual untuk mengukur performa model.

3.3.5 Denormalisasi

Denormalisasi merupakan proses mengembalikan data hasil prediksi ke dalam skala aslinya. Tahap ini penting dilakukan karena data yang digunakan untuk *training* model telah melalui proses normalisasi sebelumnya. Proses denormalisasi menggunakan formula yang berkebalikan dengan metode normalisasi yang diterapkan pada tahap *preprocessing*. Berikut adalah persamaan untuk denormalisasi.

$$d = d'(\max - \min) + min \quad (27)$$

Keterangan :
d = Nilai hasil denormalisasi
d' = Nilai data normalisasi
max = Nilai maksimum dari data *actual*
min = Nilai minimum dari data *actual*
Contoh perhitungan:

$$d = 0{,}03963(43{,}4 - 0) + 0$$

$$d = 1{,}72$$

Nilai 1,72 adalah hasil denormalisasi hasil prediksi data pada tanggal 1 Desember 2023.

Denormalisasi diperlukan agar hasil prediksi dapat diinterpretasikan dalam satuan curah hujan yang sebenarnya *(millimeter)* dan dapat dibandingkan secara langsung dengan data aktual untuk keperluan evaluasi model.

3.3.6 Evaluasi

Evaluasi model dilakukan untuk mengukur tingkat akurasi dan performa dari model LSTM dalam memprediksi curah hujan di Kota Jambi. Pada penelitian ini, metrik evaluasi yang digunakan adalah *Root Mean Square Error* (RMSE) . RMSE merupakan metrik evaluasi yang umum digunakan dalam kasus prediksi data *time series*, khususnya untuk prediksi curah hujan.

RMSE mengukur rata-rata dari akar kuadrat selisih antara nilai prediksi dengan nilai aktual. Semakin kecil nilai RMSE, semakin akurat model dalam melakukan prediksi. Formula perhitungan RMSE adalah sebagai berikut:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \quad (29)$$

Keterangan:

Y_i = Nilai aktual

$\hat{Y}_i$ = Nilai hasil peramalan / prediksi

n = Jumlah data

Pemilihan RMSE sebagai metrik evaluasi didasarkan pada beberapa kelebihan:

1. RMSE memberikan bobot yang lebih besar pada *error* yang besar, sehingga cocok untuk kasus di mana *error* besar sangat tidak diinginkan
2. RMSE menghasilkan nilai dalam satuan yang sama dengan data asli (*millimeter*), sehingga mudah diinterpretasikan
3. RMSE merupakan metrik yang *robust* dan telah banyak digunakan dalam penelitian prediksi curah hujan sebelumnya

Hasil evaluasi RMSE akan menunjukkan rata-rata kesalahan prediksi model dalam satuan *millimeter*, yang dapat digunakan untuk menilai kehandalan model LSTM dalam memprediksi curah hujan di Kota Jambi.

Contoh Perhitungan:

$$RMSE = \sqrt{\frac{(26{,}9 - 1{,}72)^2}{31}}$$

$$RMSE = \sqrt{\frac{634{,}03}{31}}$$

$$RMSE = 4{,}52$$

Berdasarkan hasil perhitungan dalam memprediksi curah hujan selama 31 hari dengan menggunakan algoritma LSTM memberikan hasil yang cukup akurat dengan nilai RMSE yang rendah sebesar 4,52 dalam kategori “akurat”, hal sesuai dengan penelitian yang dilakukan oleh Ashari & Sadiki (2020) yang menyatakan bahwa nilai RMSE yang rendah menggambarkan bahwa nilai tersebut merupakan prediksi terbaik.

3.3.7 Implementasi GUI *(Graphical User Interface)*

Graphical User Interface (GUI) merupakan antarmuka program yang berfungsi sebagai perantara komunikasi antara pengguna dan perangkat lunak (Muhtadi et al., 2019). Setelah menyelesaikan tahap *preprocessing* data hingga evaluasi manual untuk mengukur prediksi curah hujan, peneliti mengembangkan GUI yang dirancang agar mudah digunakan tanpa memerlukan

pengetahuan pemrograman. *Mockup* sistem peramalan disusun berdasarkan hasil pengujian yang telah dilakukan pada tahap sebelumnya. Proses pengembangan GUI menghasilkan sebuah sistem berbasis web, yang memungkinkan pengguna melakukan peramalan data dengan memasukkan data deret waktu *(time series)*, yang kemudian akan diproses oleh sistem.

IV. HASIL DAN PEMBAHASAN

4.1 Pengumpulan data

Data yang digunakan dalam penelitian ini diperoleh dari BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) melalui situs resmi Data Online BMKG (https://dataonline.bmkg.go.id/home). Dataset berisi data curah hujan harian di Kota Jambi dari tahun 2016 hingga 2024 sebanyak 3.288 baris data. Seluruh data curah hujan dalam dataset ini diukur dalam satuan milimeter (mm), yang merupakan standar dalam pengukuran curah hujan.

Proses pengumpulan data dilakukan secara manual karena sistem di website BMKG membatasi pengguna hanya dapat mengunduh data berdasarkan periode per bulan. Oleh karena itu, penulis melakukan pengunduhan data secara bertahap, yaitu satu bulan per file, untuk setiap tahun dari 2016 hingga 2024. Setiap file hasil unduhan disimpan dalam format .csv, kemudian digabungkan secara manual menggunakan Microsoft Excel 2019. Penggabungan dilakukan dengan cara menyusun ulang seluruh file data menjadi satu file tunggal, dengan memastikan keseragaman format tanggal dan nilai curah hujan. Setelah semua data tergabung, dilakukan pengecekan ulang untuk menghindari duplikasi atau kesalahan urutan waktu.

Data ini akan digunakan sebagai input dalam model prediksi curah hujan menggunakan metode *Long Short-Term Memory* (LSTM). Gambar 14 menunjukkan *source code* yang digunakan untuk mengimpor *library* yang diperlukan dalam proses analisis dan pemodelan data.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
```

Gambar 14. *Source Code Import Library*

Dalam penelitian ini, beberapa *library* Python digunakan untuk membantu dalam proses analisis data, *preprocessing*, pembangunan model, serta evaluasi hasil prediksi curah hujan menggunakan metode *Long Short-Term Memory* (LSTM). *Library* pandas digunakan untuk membaca, mengelola, dan

memanipulasi data dalam bentuk *dataframe*, memungkinkan pengguna untuk melakukan operasi seperti *cleaning, filtering, dan* agregasi data. Selanjutnya, numpy digunakan untuk operasi komputasi numerik, terutama dalam menangani array multidimensi yang efisien dalam pemrosesan data berbasis matriks.

Dalam tahap *preprocessing* data, digunakan MinMaxScaler dari sklearn.preprocessing, yang berfungsi untuk melakukan normalisasi data ke dalam rentang tertentu, biasanya antara 0 dan 1, guna meningkatkan stabilitas dan kinerja model prediksi. Selain itu, untuk evaluasi model, digunakan mean_squared_error dari sklearn.metrics, yang berfungsi untuk mengukur tingkat kesalahan antara nilai aktual dan nilai prediksi, di mana semakin kecil nilai *error*, maka semakin baik model yang dibangun.

Model prediksi dalam penelitian ini dikembangkan menggunakan TensorFlow, sebuah *framework deep learning* yang sangat populer. Dari tensorflow.keras, digunakan berbagai komponen penting, seperti *Input* untuk mendefinisikan *layer input*, serta *Sequential* untuk menyusun model jaringan saraf tiruan secara berurutan. Selain itu, dalam pembangunan arsitektur LSTM, digunakan beberapa layer utama dari tensorflow.keras.layers, yaitu LSTM, *Dense*, dan *Dropout*. *Layer* LSTM merupakan inti dari model yang digunakan dalam penelitian ini, karena mampu menangani *sequential* data dan mempertahankan informasi penting dalam jangka waktu yang lebih panjang dibandingkan model konvensional. *Layer Dense* digunakan sebagai *fully connected layer* yang berfungsi dalam pemetaan hubungan kompleks antar-neuron, sementara *Dropout* berfungsi untuk mencegah *overfitting* dengan mengabaikan sejumlah neuron secara acak selama proses *training*.

Terakhir, untuk keperluan visualisasi, digunakan *library* matplotlib.pyplot, yang memungkinkan pembuatan grafik dan visualisasi data seperti plot tren curah hujan, hasil normalisasi data, serta performa model prediksi. Dengan kombinasi *library* ini, penelitian dapat mengelola data dengan baik, membangun model *deep learning* yang optimal, serta mengevaluasi dan memvisualisasikan hasil prediksi dengan lebih efektif.

4.1.1 Membaca Dataset Curah Hujan

Tahapan ini merupakan tahapan dimana akan membaca data curah hujan harian Kota Jambi yang digunakan dengan menggunakan *library pandas*. Dataset ini disimpan dalam format CSV *(Comma-Separated Values)*, yang merupakan format umum untuk penyimpanan data dalam bentuk tabel, dapat dilihat pada gambar 15.

```
# Membaca dataset
df = pd.read_csv ('dataset_ch_fix.csv')
df
```

Gambar 15. *Source Code* Proses Membaca Dataset

Hasil dari *source code* pada gambar 15 dapat dilihat pada gambar 16 yang menampilkan dataset rekapan curah hujan harian Kota Jambi tahun 2016-2024 yang digunakan.

	Tanggal	RR
0	01-01-2016	1.0
1	02-01-2016	NaN
2	03-01-2016	3.7
3	04-01-2016	8888.0
4	05-01-2016	0.5
...	...	...
3283	27-12-2024	0.0
3284	28-12-2024	6.1
3285	29-12-2024	0.0
3286	30-12-2024	34.2
3287	31-12-2024	7.8

3288 rows × 2 columns

Gambar 16. Dataset Curah Hujan Tahun 2016-2024

Gambar 16 menunjukkan tampilan awal dari dataset curah hujan di Kota Jambi pada periode 2016-2024. Dataset ini terdiri dari 2 kolom, yaitu kolom 'Tanggal' yang merepresentasikan waktu pencatatan curah hujan dan kolom 'RR' yang menunjukkan jumlah curah hujan dalam milimeter (mm) pada hari tersebut. Secara keseluruhan, dataset ini memiliki 3.288 baris data. Namun, terdapat beberapa nilai yang kosong (NaN) serta nilai khusus seperti 8888.0, yang menandakan data tidak terukur. Oleh karena itu, diperlukan tahap pra-pemrosesan untuk menangani data yang hilang sebelum dilakukan analisis lebih lanjut.

4.2 *Preprocessing Data*

Pada penelitian ini, tahap *preprocessing data* dilakukan untuk menyiapkan data sebelum digunakan dalam pemodelan prediksi curah hujan. *Preprocessing* bertujuan untuk meningkatkan kualitas data dengan mengatasi masalah yang dapat mempengaruhi akurasi model, seperti data yang hilang, data

tidak valid, serta skala nilai yang berbeda. Beberapa langkah utama dalam *preprocessing data* yang diterapkan dalam penelitian ini meliputi:

4.2.1 *Data Cleaning*

Dalam proses *data cleaning*, Langkah yang dilakukan adalah identifikasi *missing value* dalam dataset curah hujan yang digunakan. Dalam hal ini, nilai 8888 yang berarti nilai curah hujan yang tidak terukur atau <0,5 dan nilai 9999 yang berarti data belum masuk atau alat rusak, perlu ditangani. Nilai-nilai tersebut tidak dapat digunakan secara langsung dalam analisis karena dapat menyebabkan bias atau kesalahan dalam perhitungan statistik. Oleh karena itu, perlu dilakukan transformasi dengan mengonversinya menjadi NaN *(Not a Number)* agar dapat diproses lebih lanjut menggunakan metode interpolasi. Transformasi ini dilakukan dengan menggunakan fungsi replace() pada Pandas yang ditampilkan pada Gambar 17, yang secara otomatis menggantikan semua nilai 8888 dan 9999 dengan NaN dalam dataset.

```
# Ganti nilai 8888 dan 9999 dengan NaN
df['RR'] = df['RR'].replace([8888, 9999], np.nan)
df
```

Gambar 17. *Source Code* Proses *Data Cleaning*

Fungsi replace() dalam Pandas memungkinkan transformasi nilai dalam suatu *Dataframe* tanpa mengubah struktur data lainnya. Dengan mengonversi kedua nilai ini menjadi NaN, dataset menjadi lebih bersih dan siap untuk langkah selanjutnya, seperti interpolasi data untuk menangani *missing value*. Hasil dari *source code* pada gambar 17 dapat dilihat pada gambar 18 yang menampilkan dataset dalam kondisi lebih bersih dan siap untuk diproses lebih lanjut. Dataset yang telah dibersihkan ini nantinya akan menjadi dasar dalam penerapan metode interpolasi untuk mengisi data yang hilang.

	Tanggal	RR
0	2016-01-01	1.0
1	2016-01-02	NaN
2	2016-01-03	3.7
3	2016-01-04	NaN
4	2016-01-05	0.5
...	...	...
3283	2024-12-27	0.0
3284	2024-12-28	6.1
3285	2024-12-29	0.0
3286	2024-12-30	34.2
3287	2024-12-31	7.8

3288 rows × 2 columns

Gambar 18. Hasil *Data Cleaning*

Gambar 18 menampilkan dataset curah hujan yang telah melewati tahap pembersihan data *(data cleaning)*. Dataset ini tetap terdiri dari 2 kolom, yaitu 'Tanggal' dan 'RR', dengan total 3.288 baris data. Pada tahap ini, nilai-nilai yang tidak valid seperti 8888.0 telah diidentifikasi dan dihapus, sementara data yang kosong (NaN) tetap dipertahankan untuk proses interpolasi lebih lanjut. Dengan pembersihan ini, dataset menjadi lebih siap untuk dianalisis dan diterapkan dalam metode interpolasi guna mengatasi nilai yang hilang.

4.2.2 Interpolasi

Interpolasi adalah metode yang digunakan untuk memperkirakan nilai yang hilang dalam suatu dataset berdasarkan pola dari data yang tersedia. Dalam penelitian ini, beberapa metode interpolasi digunakan untuk menangani *missing values* pada data curah hujan, yaitu Interpolasi Linear, Kuadrat, dan Spline Kuadratik.

a. Interpolasi Linear

Interpolasi linear adalah metode untuk memperkirakan nilai yang hilang dalam suatu dataset dengan mengasumsikan bahwa perubahan nilai terjadi secara linier antara dua titik data terdekat. Artinya, jika terdapat nilai yang hilang (NaN), metode ini akan menghubungkan dua titik data yang diketahui di sekitarnya menggunakan garis lurus dan menghitung nilai interpolasi di antaranya. Pada implementasi di Python, interpolasi linear dapat dilakukan dengan menggunakan fungsi .interpolate(method='linear') dari *library* pandas.

```
# Isi nilai NaN dengan interpolasi linear
df['RR'] = df['RR'].interpolate(method='linear')
df
```

Gambar 19. *Source Code* Interpolasi Linear

Kode pada gambar 19 akan mengisi nilai yang hilang dalam kolom 'RR' berdasarkan interpolasi linear, yaitu dengan mencari dua titik data terdekat dan menghitung nilai di antaranya berdasarkan hubungan linier. Dengan metode ini, data yang memiliki *missing value* dapat diperbaiki dengan cara yang sederhana namun cukup akurat, terutama jika perubahan data cenderung linier.

	Tanggal	RR
0	2016-01-01	1.00
1	2016-01-02	2.35
2	2016-01-03	3.70
3	2016-01-04	2.10
4	2016-01-05	0.50
...	...	...
3283	2024-12-27	0.00
3284	2024-12-28	6.10
3285	2024-12-29	0.00
3286	2024-12-30	34.20
3287	2024-12-31	7.80

3288 rows × 2 columns

Gambar 20. Hasil Interpolasi Linear

Gambar 20 menampilkan hasil penerapan metode interpolasi linear pada dataset curah hujan. Dataset ini masih memiliki 2 kolom, yaitu 'Tanggal' dan 'RR', dengan total 3.288 baris data. Pada tahap ini, nilai-nilai yang sebelumnya hilang atau tidak terukur (NaN) telah diisi menggunakan metode interpolasi linear, sehingga data menjadi lebih lengkap dan kontinu. Dengan adanya interpolasi ini, dataset lebih siap untuk digunakan dalam proses analisis lebih lanjut, termasuk dalam pemodelan prediksi curah hujan.

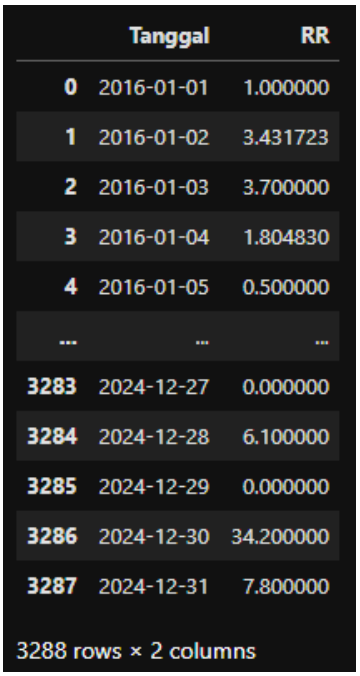
b. Interpolasi Kuadrat

Interpolasi kuadrat adalah metode interpolasi yang menggunakan polinomial derajat dua untuk memperkirakan nilai yang hilang dalam suatu dataset. Metode ini lebih fleksibel dibandingkan interpolasi linear karena mampu menangkap perubahan pola data yang lebih kompleks. Berbeda dengan interpolasi linear yang hanya mempertimbangkan dua titik terdekat, interpolasi kuadrat mempertimbangkan tiga titik terdekat untuk menghasilkan nilai estimasi. Dengan cara ini, interpolasi kuadrat dapat lebih baik dalam menangkap tren data yang tidak berubah secara linier tetapi memiliki pola melengkung. Interpolasi kuadrat dapat dilakukan menggunakan fungsi .interpolate(method='quadratic') dari *library* pandas yang dapat dilihat pada gambar 21.

```
# Isi nilai NaN dengan interpolasi kuadrat
df['RR'] = df['RR'].interpolate(method='quadratic')
df
```

Gambar 21. *Source Code* Interpolasi Kuadrat

Pada kode di atas, metode quadratic digunakan untuk memperkirakan nilai yang hilang dalam kolom 'RR' dengan mempertimbangkan tiga titik data terdekat dan menyusun polinomial kuadrat untuk menghitung nilai interpolasi.



	Tanggal	RR
0	2016-01-01	1.000000
1	2016-01-02	3.431723
2	2016-01-03	3.700000
3	2016-01-04	1.804830
4	2016-01-05	0.500000
...	...	...
3283	2024-12-27	0.000000
3284	2024-12-28	6.100000
3285	2024-12-29	0.000000
3286	2024-12-30	34.200000
3287	2024-12-31	7.800000

3288 rows × 2 columns

Gambar 22. Hasil Interpolasi kuadrat

Gambar 22 menampilkan hasil penerapan metode interpolasi kuadrat pada dataset curah hujan. Nilai yang sebelumnya kosong telah diisi menggunakan metode interpolasi kuadrat, yang menghasilkan estimasi curah hujan berdasarkan pola data yang ada.

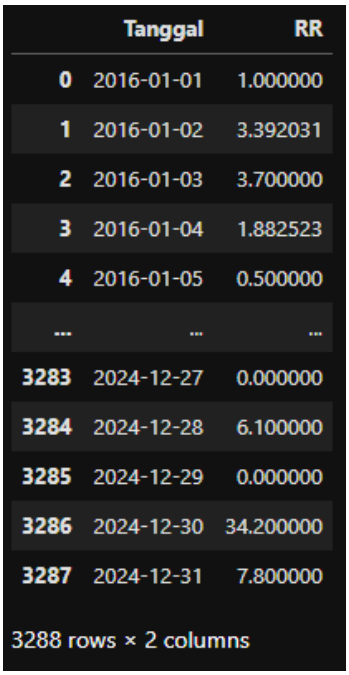
c. Interpolasi Spline Kuadratik

Interpolasi spline kuadratik adalah metode interpolasi yang menggunakan potongan-potongan fungsi polinomial kuadrat untuk memperkirakan nilai yang hilang di antara titik-titik data. Dibandingkan dengan interpolasi linear atau interpolasi kuadrat biasa, metode ini lebih fleksibel dalam menangani perubahan pola data karena mempertimbangkan lebih dari dua atau tiga titik dalam estimasi. Interpolasi spline kuadratik dapat dilakukan menggunakan fungsi .interpolate(method='spline', order=2) dari *library* pandas.

```
# Isi nilai Nan dengan Interpolasi Spline Kuadratik
df['RR'] = df['RR'].interpolate(method='spline', order = 2)
df
```

Gambar 23. *Source Code* Interpolasi Spline Kuadratik

Pada kode di atas method='spline' menunjukkan bahwa metode interpolasi yang digunakan adalah spline interpolation. order=2 menandakan bahwa polinomial yang digunakan dalam interpolasi adalah polinomial derajat dua (kuadratik).



	Tanggal	RR
0	2016-01-01	1.000000
1	2016-01-02	3.392031
2	2016-01-03	3.700000
3	2016-01-04	1.882523
4	2016-01-05	0.500000
...	...	...
3283	2024-12-27	0.000000
3284	2024-12-28	6.100000
3285	2024-12-29	0.000000
3286	2024-12-30	34.200000
3287	2024-12-31	7.800000

3288 rows × 2 columns

Gambar 24. Hasil Interpolasi Spline Kuadratik

Gambar 24 menampilkan hasil penerapan metode interpolasi spline kuadratik pada dataset curah hujan. Data yang sebelumnya memiliki nilai kosong kini telah diisi menggunakan pendekatan spline kuadratik, yang memperhitungkan kelandaian perubahan nilai agar interpolasi lebih halus. Hasil dari metode ini nantinya akan dibandingkan dengan metode interpolasi lainnya berdasarkan perhitungan error seperti RMSE untuk menentukan pendekatan terbaik dalam menangani data yang hilang.

4.2.3 Normalisasi Data

Normalisasi data merupakan tahap penting dalam *preprocessing* untuk menyamakan skala nilai pada dataset. Dalam penelitian ini, normalisasi dilakukan menggunakan metode Min-Max Scaling, yang mengubah rentang nilai curah hujan menjadi skala 0 hingga 1. Proses normalisasi ini bertujuan untuk memastikan skala nilai yang seragam sehingga dapat meningkatkan akurasi dan stabilitas model dalam analisis lebih lanjut. Proses normalisasi dalam penelitian ini diterapkan menggunakan fungsi MinMaxScaler dari *library* scikit-learn. Gambar 25 merupakan *source code* yang digunakan:

```
# Normalisasi data
scaler = MinMaxScaler(feature_range=(0, 1))
df['RR_scaled'] = scaler.fit_transform(df[['RR']])
df
```

Gambar 25. *Source Code* Normalisasi

Kode ini berfungsi untuk mengganti nilai pada kolom RR menjadi skala yang seragam. Objek MinMaxScaler diinisialisasi dengan parameter feature_range=(0, 1), kemudian diterapkan pada kolom RR menggunakan fit_transform() yang akan menyesuaikan data berdasarkan nilai minimum dan maksimum dalam dataset. Hasil normalisasi disimpan dalam kolom RR_scaled, sehingga data asli tetap tersedia untuk keperluan analisis lebih lanjut. Dataset yang telah dinormalisasi ini digunakan dalam tahap pemodelan untuk memastikan hasil prediksi yang lebih optimal.

4.2.4 Pembagian Data *Training* dan *Testing*

Setelah dilakukan normalisasi, tahap selanjutnya adalah membagi dataset menjadi data pelatihan *(training data)* dan data pengujian *(testing data)*. Pembagian dataset ini bertujuan untuk memastikan model dapat melakukan generalisasi dengan baik terhadap data baru yang belum pernah dilihat sebelumnya. Dalam penelitian ini, dataset dibagi dengan rasio 80:20, sebanyak 80% dari data digunakan untuk melatih model *(training set)*, sedangkan 20% sisanya digunakan untuk menguji model *(testing set)*. Perbandingan 80:20 ini merupakan standar umum yang sering digunakan dalam praktik *machine learning*. Gambar 26 merupakan *source code* yang digunakan:

```
# Buat Train-Test Split (Pastikan Tanggal tetap ada)
split = int(len(df) * 0.8)
train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
```

Gambar 26. *Source Code* Data *Testing* dan *Training*

Pembagian dataset dilakukan dengan menentukan titik pemisah sebesar 80% dari total data menggunakan perhitungan split = int(0.8 * len(df)). Data pelatihan diperoleh dengan mengambil 80% data pertama dari kolom tanggal dan hasil normalisasi (RR_scaled), sedangkan data pengujian diambil dari 20% data terakhir dalam kolom yang sama. Teknik ini mempertahankan urutan waktu dalam dataset, yang penting dalam analisis deret waktu *(time series)* seperti prediksi curah hujan. Dengan pembagian ini, model dapat dilatih menggunakan

data historis dan diuji menggunakan data yang lebih baru, sehingga dapat memberikan gambaran akurasi prediksi yang lebih realistis.

Selain itu pembagian data pelatihan dan pengujian juga berperan penting dalam mencegah *overfitting*, yaitu kondisi di mana model terlalu menyesuaikan diri dengan pola pada data pelatihan sehingga tidak mampu melakukan generalisasi dengan baik terhadap data yang belum pernah ditemui. Dengan adanya data uji yang terpisah, penulis dapat menilai performa model secara objektif serta menentukan apakah model memiliki kemampuan untuk menggeneralisasi pola pada data baru.

4.3 Membuat Model LSTM

Pada penelitian ini, digunakan model *Long Short-Term Memory* (LSTM), yang merupakan jenis *Recurrent Neural Network* (RNN). LSTM dirancang untuk menangani data deret waktu *(time series)* dengan mempertahankan informasi jangka panjang dan mengatasi masalah *vanishing gradient* yang sering terjadi pada RNN konvensional. Dengan adanya mekanisme *gates* di dalamnya, LSTM mampu menyimpan, memperbarui, atau menghapus informasi secara selektif, sehingga cocok untuk analisis dan prediksi data yang bersifat sekuensial, seperti curah hujan.

4.3.1 Membuat Urutan Data LSTM

Dalam model LSTM, data perlu diproses dalam bentuk *sekuens* waktu *(time series sequences)* sebelum digunakan untuk pelatihan model. Hal ini bertujuan agar model dapat mengenali pola historis dalam data dan menghasilkan prediksi yang lebih akurat. Oleh karena itu, data harus dibagi menjadi *input* (X) dan *target* (y) berdasarkan langkah waktu tertentu *(n_steps)*, yang menentukan berapa banyak data historis yang digunakan untuk memprediksi nilai di masa depan.

Pada penelitian ini, jumlah langkah waktu *(n_steps)* yang digunakan adalah 30, yang berarti model akan menggunakan data 30 hari terakhir (30 *timestep*) untuk memperkirakan curah hujan pada hari berikutnya. Semakin besar nilai *n_steps*, semakin banyak informasi historis yang digunakan, namun hal ini juga dapat meningkatkan kompleksitas model. Oleh karena itu, pemilihan jumlah *timestep* harus mempertimbangkan keseimbangan antara akurasi dan efisiensi model. Gambar 27 merupakan *source code* yang digunakan:

```python
#  Fungsi Membuat Sequence Data
def create_sequences(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps):
        X.append(data.iloc[i:i + n_steps, 1].values)  # RR_scaled ada di index 1
        y.append(data.iloc[i + n_steps, 1])
    return np.array(X), np.array(y)

n_steps = 30
X_train, y_train = create_sequences(train_data, n_steps)
X_test, y_test = create_sequences(test_data, n_steps)

# Simpan Tanggal yang sesuai untuk prediksi
train_dates = train_data["Tanggal"].iloc[n_steps:].values
test_dates = test_data["Tanggal"].iloc[n_steps:].values
```

Gambar 27. *Source Code* Urutan Data LSTM

Fungsi create_sequences digunakan untuk membentuk urutan data yang diperlukan dalam pelatihan model LSTM. Fungsi ini membagi dataset menjadi pasangan *input* (X) dan *target* (y) berdasarkan jumlah langkah waktu (n_steps) yang telah ditentukan. Proses ini diawali dengan inisialisasi dua list kosong, yaitu X untuk menyimpan data historis sebagai *input* model dan y untuk menyimpan nilai *target* yang akan diprediksi. Selanjutnya, dilakukan iterasi sepanjang dataset dengan rentang len(data) - n_steps, sehingga setiap iterasi akan mengambil n_steps data berturut-turut sebagai X, sementara nilai pada indeks ke-(i + n_steps) akan menjadi y. Setelah seluruh pasangan data terbentuk, kedua list tersebut dikonversi menjadi array NumPy agar kompatibel dengan model LSTM.

Proses berikutnya adalah *data reshape* pada *train dataset* dan *test dataset*, proses *reshape* ini dapat dilakukan menggunakan kode yang terdapat pada Gambar 28.

```python
# Reshape data agar sesuai dengan input LSTM (samples, timesteps, features)
X_train = X_train.reshape(-1, n_steps, 1)
X_test = X_test.reshape(-1, n_steps, 1)
```

Gambar 28. *Source Code Data Reshape*

Langkah ini perlu dilakukan karena LSTM bekerja dengan model 3D, sedangkan data yang tersedia saat ini masih dalam bentuk 2D. Oleh karena itu, data perlu diubah bentuknya *(reshape)* menjadi 3D. Data 2D dengan format (x, y) akan dikonversi menjadi 3D (x, y, z) dengan menambahkan satu dimensi tambahan (z).

4.3.2 *Modelling* LSTM

Tahap ini memerlukan penentuan model yang akan dibentuk berdasarkan arsitektur dan parameter yang dipilih. Arsitektur dan parameter yang dibutuhkan, antara lain:

Tabel 10. Parameter Yang Digunakan Pada Model

Karakteristik	Spesifikasi
Arsitektur	1 *Input Layer*, 1 LSTM *Layer*, 1 *Dropout Layer*, dan 1 *Dense Layer*
Jumlah Neuron	10, 20, 30, 40, 50, 100, dan 1 *(Dense)*
Optimizer	Adam, RMSprop, SGD
Epoch	25, 50, 75, 100, 200
Batch Size	32, 64
Loss Function	*Mean Squared Error* (MSE)
Activation Function	ReLU

Proses pembelajaran jaringan LSTM dapat dilakukan dengan menyesuaikan berbagai parameter untuk memperoleh model yang optimal dalam memprediksi curah hujan. Salah satu aspek penting dalam proses ini adalah pemilihan jumlah neuron pada lapisan tersembunyi *(hidden layer)*, yang tidak memiliki aturan baku dalam penentuannya. Oleh karena itu, peneliti perlu melakukan serangkaian eksperimen dengan variasi jumlah neuron untuk menemukan konfigurasi terbaik yang menghasilkan akurasi prediksi tertinggi.

Selain itu, jumlah *epoch* juga berperan penting dalam proses pelatihan jaringan saraf tiruan. *Epoch* merujuk pada jumlah iterasi atau siklus pelatihan yang dilakukan terhadap seluruh dataset. Semakin banyak *epoch* yang digunakan, semakin lama model belajar dari data, tetapi jika jumlahnya terlalu besar, dapat menyebabkan *overfitting*, di mana model terlalu menyesuaikan diri dengan data pelatihan sehingga kurang mampu memprediksi data baru dengan baik. Oleh karena itu, jumlah *epoch* harus dipilih secara optimal agar model dapat belajar dengan baik tanpa mengalami *overparameterization*.

Jumlah neuron dalam *hidden layer* yang terlalu sedikit dapat membuat model kurang mampu menangkap hubungan kompleks antara variabel *input* dan target. Sebaliknya, terlalu banyak neuron dapat meningkatkan risiko *overfitting* karena model menjadi terlalu kompleks dan kurang mampu melakukan generalisasi terhadap data baru. Oleh sebab itu, dalam penelitian ini dilakukan pengujian dengan berbagai konfigurasi jumlah neuron dan *epoch* untuk mengevaluasi kinerja model berdasarkan nilai *error* yang dihasilkan.

Menurut Tanjung et al. (2024), *tuning hyperparameter* seperti jumlah neuron, fungsi aktivasi, jumlah *epoch*, dan *learning rate* sangat penting dalam memperoleh model LSTM yang optimal. Studi tersebut menunjukkan bahwa model optimal diperoleh dengan menggunakan 100 neuron, fungsi aktivasi *tangen hiperbolik* (tanh), 100 *epoch*, dan *learning rate* sebesar 0,005, yang menghasilkan RMSE sebesar 0,0224. Berdasarkan pendekatan tersebut, dalam penelitian ini dilakukan pengujian dengan variasi jumlah neuron sebanyak 10, 20, 30, 40, 50 dan 100 serta rentang ini dipilih berdasarkan praktik umum serta pendekatan *hyperparameter search* sebagaimana dilakukan dalam penelitian oleh Karim et al. (2017), yang mengeksplorasi jumlah neuron optimal dalam rentang 8 hingga 128 untuk *time series classification* menggunakan LSTM-FCN.

Jumlah *epoch* sebanyak 25, 50, 75, 100, dan 200 didasarkan pada pendekatan empiris sebagaimana disarankan oleh Brownlee (2018) dalam bukunya *Deep Learning for Time Series Forecasting*, bahwa tidak ada aturan baku mengenai penggunaan jumlah epoch, pemilihan epoch yang optimal perlu dilakukan dengan mempertimbangkan *overfitting* dan waktu pelatihan. Pemilihan rentang ini bertujuan untuk mengevaluasi pengaruh jumlah neuron dan *epoch* terhadap performa model dalam menangani prediksi curah hujan. Dengan menguji berbagai kombinasi, diharapkan dapat ditemukan konfigurasi yang menghasilkan nilai *error* terkecil.

```
# Membangun model LSTM
model = Sequential()
# Gunakan Input layer sebagai lapisan pertama
model.add(Input(shape=(n_steps, 1)))
model.add(LSTM(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
# Kompilasi model
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(), loss='mse')
```

Gambar 29. *Source Code* Untuk Membuat Model LSTM

Model *Long Short-Term Memory* (LSTM) yang digunakan dalam penelitian ini dirancang untuk memprediksi curah hujan berdasarkan data historis. Model ini dibangun menggunakan *Sequential* dari Keras, yang memungkinkan penambahan lapisan secara berurutan. Arsitektur model terdiri dari satu lapisan *Input*, satu lapisan LSTM, satu lapisan *Dropout*, dan satu lapisan Dense. Lapisan *Input* menerima data dalam bentuk (n_steps, 1), di mana *n_steps* adalah jumlah langkah waktu yang digunakan, yaitu 30 hari terakhir. Lapisan LSTM dengan 50 neuron menggunakan fungsi aktivasi ReLU, yang dipilih karena lebih stabil

dibanding tanh dalam menangani masalah *vanishing gradient* serta dapat mempercepat konvergensi model.

Selanjutnya, lapisan *Dropout* dengan tingkat 0.2 diterapkan untuk mencegah *overfitting* dengan menonaktifkan 20% neuron secara acak selama pelatihan. Pemilihan nilai *dropout* 0.2 dalam model ini bertujuan untuk mencegah *overfitting* tanpa terlalu banyak kehilangan informasi dari data pelatihan. *Dropout* bekerja dengan menonaktifkan sejumlah neuron secara acak selama pelatihan, sehingga model tidak terlalu bergantung pada pola tertentu dalam data latih. Lapisan *Dense* dengan satu neuron digunakan sebagai lapisan keluaran untuk menghasilkan satu nilai prediksi curah hujan. Model kemudian dikompilasi menggunakan 3 optimizer (Adam, RMSprop, dan SGD) yang dapat menyesuaikan laju pembelajaran secara adaptif, serta fungsi kerugian *Mean Squared Error* (MSE), yang sesuai untuk model regresi dalam mengukur perbedaan antara nilai prediksi dan nilai aktual. Dengan arsitektur ini, diharapkan model dapat menangkap pola dari data historis dengan baik, sehingga menghasilkan prediksi curah hujan yang lebih akurat.

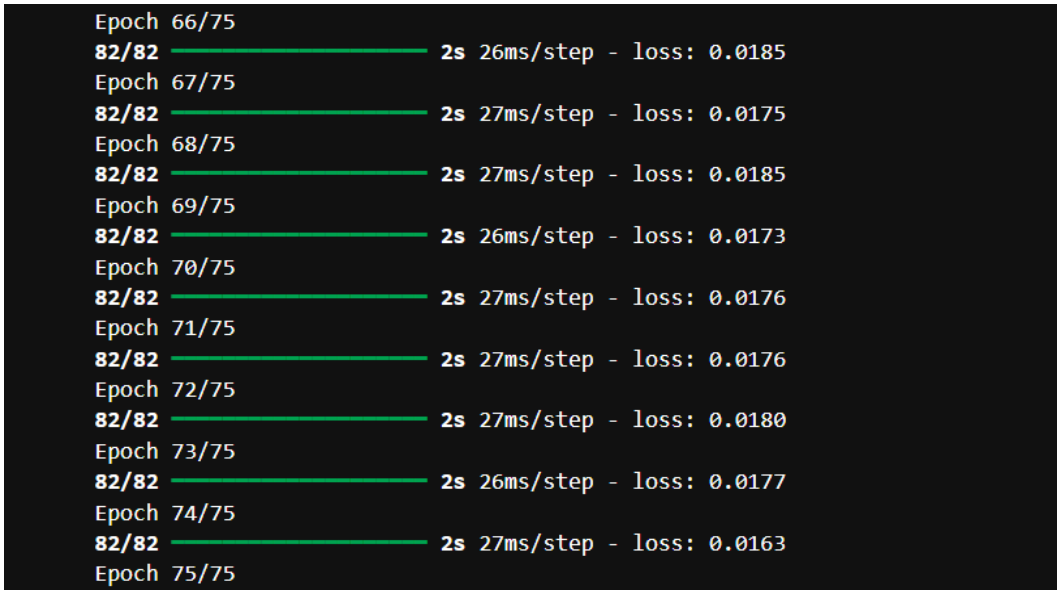
Proses selanjutnya yaitu melatih model yang dilakukan menggunakan metode *fit* dengan parameter tertentu, seperti jumlah *epoch* dan ukuran *batch*. Model dilatih dengan dataset pelatihan (X_train, y_train) selama 75 epoch, yang berarti seluruh data akan diproses sebanyak 75 kali untuk memperbarui bobot model agar dapat mengenali pola dalam data curah hujan. Gambar 30 merupakan *source code* yang digunakan:

```
# Melatih model
history = model.fit(X_train, y_train, epochs=75, batch_size=32)
```

Gambar 30. *Source Code* Melatih Model LSTM

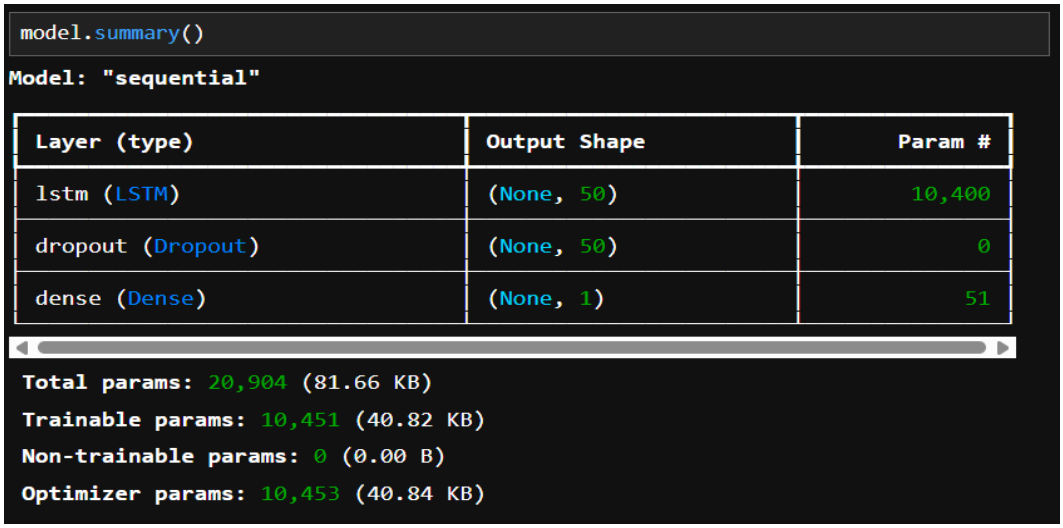
Penggunaan 75 *epoch* memberikan model waktu yang cukup dalam mempelajari pola data tanpa menyebabkan *overfitting*. Selain itu, *batch size* sebesar 32 digunakan untuk membagi data pelatihan menjadi kelompok kecil yang masing-masing terdiri dari 32 sampel per iterasi. Penggunaan mini-batch ini bertujuan untuk menyeimbangkan efisiensi komputasi dan kestabilan pembelajaran dibandingkan dengan pendekatan *batch gradient descent* (menggunakan semua data sekaligus) atau *stochastic gradient descent* (memproses satu sampel per iterasi). Hasil dari setiap proses pelatihan disimpan dalam variabel *history*, yang kemudian digunakan untuk mengevaluasi kinerja model dengan memantau perubahan nilai *loss function* selama proses pelatihan.

Hasil dari *source code* pada gambar 30 dapat dilihat pada Gambar 31 yang menampilkan hasil pengujian *epoch* pada model LSTM.



Gambar 31. Pengujian *Epoch*

Gambar 31 menunjukkan hasil pelatihan model LSTM selama 75 *epoch*, di mana setiap *epoch* ditandai dengan nilai *loss function* yang mencerminkan kesalahan prediksi model. Pada setiap *epoch*, model melakukan pembaruan bobot berdasarkan data pelatihan untuk mengurangi nilai *loss*. Dari hasil yang ditampilkan, terlihat bahwa nilai *loss* cenderung mengalami penurunan seiring bertambahnya jumlah *epoch*, yang menunjukkan bahwa model semakin mampu mengenali pola dalam data curah hujan. Selama pelatihan, *output* menunjukkan 82/82, yang berarti dalam setiap *epoch*, model memproses 82 *batch* data latih hingga seluruh data selesai dipelajari. Jumlah *batch* ini dihitung berdasarkan total jumlah data latih dibagi dengan *batch size* (32).



Gambar 32. Ringkasan Dari Model

Gambar 32 merupakan hasil dari model.summary(), yang memberikan ringkasan arsitektur model LSTM yang telah dibangun. Ringkasan menunjukkan jumlah total parameter dalam model, yaitu 20,904, dengan 10,451 di antaranya adalah *trainable parameters* (parameter yang diperbarui selama pelatihan), sedangkan 10,453 merupakan *optimizer parameters*, yang digunakan dalam proses pembaruan bobot dengan algoritma RMSprop. Tidak ada *non-trainable parameters*, yang menunjukkan bahwa seluruh parameter dalam model dapat diperbarui selama pelatihan.

4.4 Pengujian Model

Pengujian model dilakukan dengan melakukan prediksi pada data *train* dan data *test* untuk menilai performa model dalam memprediksi curah hujan berdasarkan pola historis yang telah dipelajari. Prediksi dilakukan pada data latih (X_train) dan data uji (X_test) menggunakan fungsi model.predict(). Sebelum dimasukkan ke dalam model, data diubah terlebih dahulu menggunakan reshape(-1, n_steps, 1), yang bertujuan untuk menyesuaikan dimensi input agar sesuai dengan kebutuhan model berbasis *Long Short-Term Memory* (LSTM). Transformasi ini memastikan bahwa data memiliki bentuk (jumlah sampel, *time steps*, jumlah fitur), sehingga dapat diproses dengan baik oleh model. *Output* dari prediksi ini berupa y_train_pred untuk data latih dan y_test_pred untuk data uji, yang kemudian digunakan untuk mengevaluasi kinerja model dalam memprediksi curah hujan. Berikut *source code* pengujian yang digunakan:

```
# Prediksi untuk data train dan data test
y_train_pred = model.predict(X_train.reshape(-1, n_steps, 1))
y_test_pred = model.predict(X_test.reshape(-1, n_steps, 1))
```

Gambar 33. *Source Code* Pengujian Model

Pada *source code* program gambar 33 menjelaskan bahwa tampilan prediksi data *training* dan *testing*. Tampilannya bisa dilihat pada gambar 34.

```
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

Gambar 34. Prediksi Data *Training* dan *Testing*

Output 82/82 dan 20/20 menunjukkan jumlah *batch* yang diproses selama satu *epoch* dalam proses pelatihan dan pengujian model. Angka 82/82 menandakan bahwa dataset latih telah dibagi menjadi 82 *batch*, dan seluruh *batch* telah diproses dalam satu *epoch*. Sementara itu, 20/20 menunjukkan bahwa dataset uji dibagi menjadi 20 *batch*, yang juga telah sepenuhnya diproses. Waktu eksekusi yang tertera, seperti 1s 7ms/step untuk pelatihan dan 0s 4ms/step untuk pengujian, menggambarkan rata-rata waktu pemrosesan setiap

batch serta total waktu yang dibutuhkan dalam satu *epoch*. Perbedaan jumlah *batch* antara dataset latih dan uji disebabkan oleh ukuran dataset yang berbeda, di mana dataset latih umumnya lebih besar, sehingga memerlukan lebih banyak *batch* dalam proses pelatihannya.

4.5 Denormalisasi

Denormalisasi merupakan tahap penting dalam proses pemodelan data, terutama ketika hasil prediksi harus dikembalikan ke skala aslinya agar dapat diinterpretasikan dengan lebih baik. Setelah model melakukan prediksi dalam skala yang telah dinormalisasi sebelumnya, nilai-nilai tersebut perlu dikonversi kembali ke satuan aslinya agar tetap sesuai dengan konteks data sebenarnya. Proses ini dilakukan dengan menerapkan kembali metode transformasi yang digunakan pada tahap normalisasi, tetapi dalam mode inversi. Dalam konteks ini, metode inverse_transform dari objek scaler digunakan untuk mengembalikan hasil prediksi serta nilai aktual ke skala awal sebelum normalisasi dilakukan.

Kode berikut menunjukkan proses denormalisasi yang diterapkan pada hasil prediksi model dan nilai aktual. Hasil prediksi untuk data latih (y_train_pred) dan data uji (y_test_pred) dikembalikan ke skala aslinya menggunakan scaler.inverse_transform. Hal ini bertujuan agar skala data yang digunakan dalam evaluasi tetap konsisten dengan skala asli sebelum proses normalisasi. Berikut adalah kode yang digunakan untuk melakukan denormalisasi hasil prediksi dan nilai aktual:

```
# Denormalisasi hasil prediksi
y_train_pred_original = scaler.inverse_transform(y_train_pred)
y_test_pred_original = scaler.inverse_transform(y_test_pred)
```

Gambar 35. *Source Code* Denormalisasi Data

Dengan proses ini, hasil prediksi yang diperoleh dapat dianalisis lebih lanjut dengan membandingkannya langsung dengan nilai aktual dalam skala yang sama. Gambar 35 menunjukkan implementasi kode denormalisasi yang bertujuan untuk mengembalikan hasil prediksi ke dalam satuan aslinya, sehingga lebih mudah dipahami dan digunakan dalam interpretasi lebih lanjut.

4.6 Evaluasi

Tahap selanjutnya yaitu evaluasi yang bertujuan untuk mengukur kinerja dari model yang telah dibuat. Metode Evaluasi yang digunakan adalah *Root Mean Square Error* (RMSE). RMSE digunakan untuk mengukur tingkat kesalahan prediksi dengan menghitung akar dari rata-rata kuadrat selisih antara nilai aktual dan nilai prediksi. Dalam penelitian ini, RMSE dihitung pada data yang telah dikembalikan ke skala aslinya *(denormalized RMSE)*. Perhitungan RMSE

pada data asli digunakan untuk menginterpretasikan kesalahan prediksi dalam satuan yang sebenarnya. Berikut adalah *source code* yang digunakan:

```
# Menghitung RMSE
rmse_train = np.sqrt(mean_squared_error(scaler.inverse_transform(y_train.reshape(-1, 1)), y_train_pred_original))
rmse_test = np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)), y_test_pred_original))

print(f"RMSE Train: {rmse_train:.4f}")
print(f"RMSE Test: {rmse_test:.4f}")
```

Gambar 36. *Source Code* Evaluasi

Hasil dari *source code* pada gambar 36 berupa persentase kesalahan yang dihitung menggunakan *Root Mean Square Error* (RMSE) setelah proses denormalisasi.

```
RMSE Train: 13.0661
RMSE Test: 13.1388
```

Gambar 37. Hasil RMSE

Dari hasil di atas menunjukan skor RMSE yang paling optimal yaitu 13.0661 untuk data *test* dan 13.1388 untuk data *train*. Nilai RMSE ini menunjukkan seberapa jauh hasil prediksi model berbeda dari nilai aktual dalam skala aslinya. Semakin kecil nilai RMSE, maka semakin baik kinerja model dalam melakukan prediksi. Evaluasi ini bertujuan untuk memastikan bahwa model dapat menghasilkan prediksi yang akurat dan dapat diandalkan dalam analisis data curah hujan di Kota Jambi.

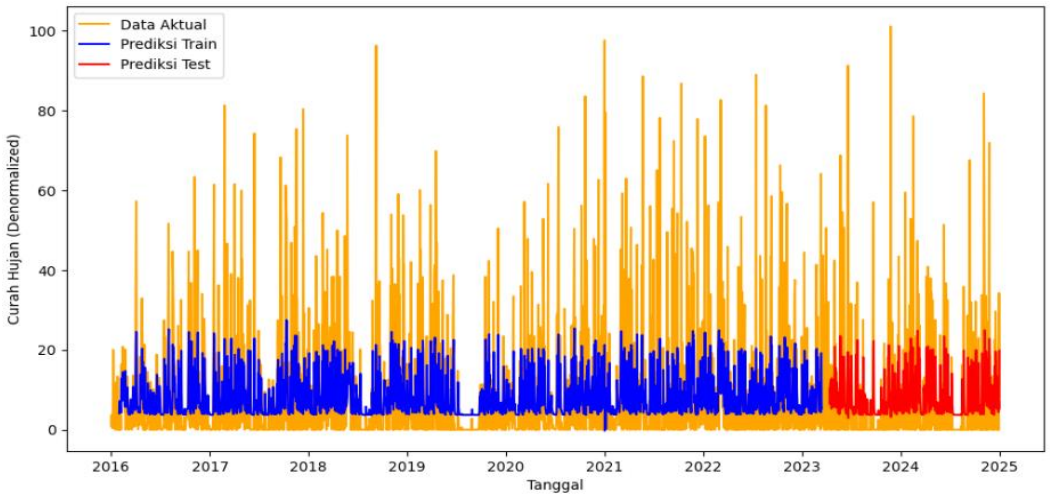
Untuk mengevaluasi hasil prediksi model, dilakukan proses visualisasi menggunakan matplotlib dengan membandingkan data aktual dan hasil prediksi model pada data *training* serta *testing*. Sebelum divisualisasikan, hasil prediksi yang telah melalui proses denormalisasi dikonversi ke dalam bentuk DataFrame dengan kolom Tanggal dan nilai prediksi. Selanjutnya, grafik dibuat dengan menampilkan tiga kurva utama, yaitu data aktual yang ditampilkan dalam warna oranye, hasil prediksi pada data *training* dalam warna biru, serta hasil prediksi pada data *testing* dalam warna merah. Sumbu X merepresentasikan tanggal, sedangkan sumbu Y menunjukkan curah hujan dalam satuan aslinya setelah denormalisasi. Berikut *source code* yang digunakan.

```
#  Buat DataFrame Hasil Prediksi
df_pred_train = pd.DataFrame({'Tanggal': train_dates, 'Prediksi_Train': y_train_pred_original.flatten()})
df_pred_test = pd.DataFrame({'Tanggal': test_dates, 'Prediksi_Test': y_test_pred_original.flatten()})

#  Plot Hasil Prediksi dengan Tanggal
plt.figure(figsize=(12,6))
plt.plot(df['Tanggal'], df['RR'], label="Data Aktual", color= 'orange')
plt.plot(df_pred_train['Tanggal'], df_pred_train['Prediksi_Train'], label="Prediksi Train", color='blue')
plt.plot(df_pred_test['Tanggal'], df_pred_test['Prediksi_Test'], label="Prediksi Test", color='red')
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Denormalized)")
plt.legend()
plt.show()
```

Gambar 38. *Source Code* Visualisasi Prediksi dan aktual

Berikut merupakan visualisasi hasil prediksi dan data aktual yang dapat dilihat pada gambar 39.



Gambar 39. Visualisasi Aktual Vs Prediksi

Visualisasi diatas menunjukkan perbandingan antara data curah hujan aktual dan hasil prediksi model setelah proses denormalisasi. Grafik ini bertujuan untuk menggambarkan sejauh mana model mampu mengikuti pola data curah hujan yang sebenarnya. Garis oranye mewakili data aktual, garis biru menunjukkan hasil prediksi model menggunakan data *train* dan garis merah menunjukkan hasil prediksi model menggunakan data *test*. Dari grafik ini, dapat diamati bahwa model masih mengalami kesulitan dalam menangkap lonjakan curah hujan yang tinggi, namun tetap mampu mengikuti pola tren secara umum. Visualisasi ini menjadi salah satu cara untuk mengevaluasi performa model dalam memprediksi curah hujan berdasarkan data historis.

Setelah model dilatih, dilakukan proses prediksi curah hujan untuk 14 hari ke depan. Proses ini dimulai dengan mengambil sejumlah langkah terakhir dari data uji yang telah dinormalisasi. Langkah-langkah tersebut digunakan sebagai *input* awal untuk model guna menghasilkan prediksi pertama. Selanjutnya, pendekatan *autoregressive* diterapkan, di mana hasil prediksi yang diperoleh digunakan sebagai *input* untuk prediksi berikutnya secara iteratif hingga 14 hari ke depan. Hasil prediksi yang masih dalam skala normalisasi kemudian dikonversi kembali ke satuan aslinya menggunakan fungsi *inverse transform* dari scaler yang digunakan sebelumnya. Untuk mendukung interpretasi hasil, dibuat daftar tanggal yang sesuai dengan rentang waktu prediksi berdasarkan tanggal terakhir dalam dataset. Data hasil prediksi ini kemudian disusun dalam bentuk DataFrame yang memuat informasi tanggal prediksi serta nilai curah hujan yang diprediksi. Berikut *source code* yang digunakan:

```python
# Prediksi 2 Minggu ke Depan
n_future = 14  # 14 hari ke depan
last_n_steps = test_data['RR_scaled'].values[-n_steps:].reshape(1, -1, 1)

future_predictions = []
for _ in range(n_future):
    next_pred = model.predict(last_n_steps, verbose=0)
    future_predictions.append(next_pred[0, 0])
    last_n_steps = np.roll(last_n_steps, -1)
    last_n_steps[0, -1, 0] = next_pred[0, 0]
```

```python
# Konversi ke satuan asli
future_predictions_original = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
```

```python
# Buat tanggal prediksi
last_date = pd.to_datetime(test_data['Tanggal'].iloc[-1])
future_dates = [last_date + pd.Timedelta(days=i) for i in range(1, n_future+1)]
```

```python
# DataFrame hasil prediksi
future_df = pd.DataFrame({'Tanggal': future_dates, 'Prediksi Curah Hujan': future_predictions_original.flatten()})
print(future_df)
```

Gambar 40. *Source Code* Prediksi 14 Hari Kedepan

Hasil dari gambar 40 dapat dilihat pada gambar 41 yang menampilkan hasil prediksi curah hujan untuk 14 hari kedepan.

	Tanggal	Prediksi Curah Hujan
0	2025-01-01	10.788293
1	2025-01-02	10.523721
2	2025-01-03	10.842383
3	2025-01-04	11.294472
4	2025-01-05	11.671933
5	2025-01-06	11.920694
6	2025-01-07	12.058013
7	2025-01-08	12.121278
8	2025-01-09	12.144642
9	2025-01-10	12.151215
10	2025-01-11	12.153596
11	2025-01-12	12.156913
12	2025-01-13	12.162073
13	2025-01-14	12.169326

Gambar 41. Hasil Prediksi 14 Hari Ke Depan

Hasil prediksi curah hujan untuk 14 hari ke depan ditampilkan dalam bentuk tabel yang memuat informasi tanggal prediksi dan nilai curah hujan yang diprediksi. Berdasarkan hasil tersebut, terlihat adanya pola peningkatan curah hujan secara bertahap dari tanggal 1 Januari 2025 hingga 14 Januari 2025. Nilai prediksi awal berada pada angka 10.78 mm dan meningkat hingga 12.16 mm pada hari ke-14. Informasi ini dapat digunakan untuk analisis lebih lanjut terkait pola curah hujan yang diprediksi oleh model.

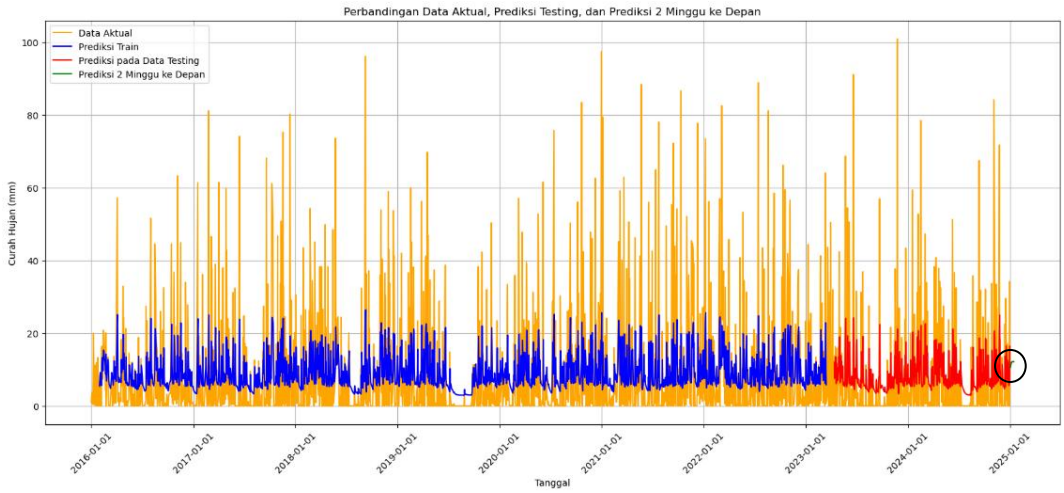
Untuk memvisualisasikan hasil prediksi model, dibuat grafik yang membandingkan data curah hujan aktual, hasil prediksi pada data latih dan uji,

serta prediksi curah hujan untuk 14 hari ke depan. Berikut *source code* yang digunakan:

```
#  Plot Perbandingan Data Aktual, Prediksi Testing, dan Prediksi 2 Minggu ke Depan
plt.figure(figsize=(20, 8))
plt.plot(df['Tanggal'], df['RR'], label="Data Aktual", color= 'orange')
plt.plot(df_pred_train['Tanggal'], df_pred_train['Prediksi_Train'], label="Prediksi Train", color='blue')
plt.plot(df_pred_test['Tanggal'], df_pred_test['Prediksi_Test'], label="Prediksi pada Data Testing", color='red')
plt.plot(future_df['Tanggal'], future_df['Prediksi Curah Hujan'], label="Prediksi 2 Minggu ke Depan", color='green')
# Mengatur sumbu X agar hanya menampilkan tanggal 1 Januari tiap tahun
ax = plt.gca()  # Mendapatkan sumbu saat ini
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (mm)")
plt.title("Perbandingan Data Aktual, Prediksi Testing, dan Prediksi 2 Minggu ke Depan")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```

Gambar 42. *Source Code* Visualisasi Perbandingan Aktual vs prediksi

Hasil dari source code gambar 42 dapat dilihat pada gambar 43 yang menampilkan perbandingan data aktual dengan prediksi yang telah dilakukan.



Gambar 43. Visualisasi Data Aktual vs Prediksi

Gambar di atas menampilkan perbandingan antara data curah hujan aktual, hasil prediksi pada data latih, hasil prediksi pada data uji, serta prediksi curah hujan untuk 14 hari ke depan. Data aktual yang ditampilkan dengan warna oranye menunjukkan fluktuasi curah hujan dari tahun 2016 hingga awal 2025, dengan pola musiman dan lonjakan signifikan pada beberapa periode. Prediksi pada data latih, yang ditampilkan dalam warna biru, menunjukkan bahwa model mampu menangkap pola dasar curah hujan dengan cukup baik. Sementara itu, prediksi pada data uji yang ditampilkan dalam warna merah memberikan gambaran sejauh mana model dapat memprediksi data yang belum pernah dilihat sebelumnya, meskipun terdapat beberapa penyimpangan dibandingkan dengan data aktual. Selain itu, model juga digunakan untuk memprediksi curah hujan selama 14 hari ke depan, yang ditampilkan dengan warna hijau, sebagai upaya untuk memberikan estimasi terhadap kondisi cuaca mendatang. Secara

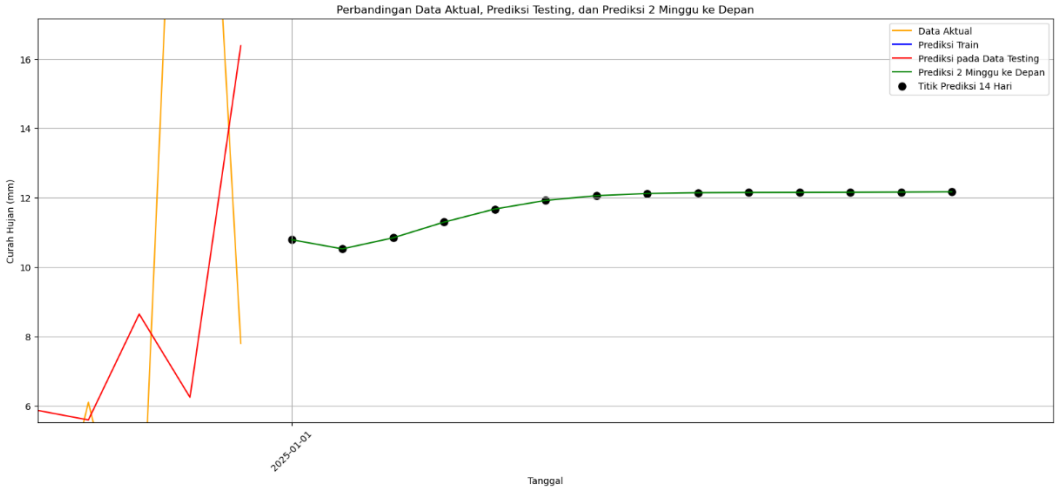
keseluruhan, hasil visualisasi ini menunjukkan bahwa model mampu mengikuti pola data historis dengan baik, meskipun terdapat beberapa deviasi terutama pada periode dengan lonjakan curah hujan yang ekstrem.

Untuk memperjelas bagian visualisasi untuk prediksi 14 hari ke depan dapat ditambahkan source code berikut:

```
# Menambahkan lingkaran pada titik prediksi 14 hari ke depan
plt.scatter(future_df['Tanggal'], future_df['Prediksi Curah Hujan'], color='black', edgecolors='white', s=100, label="Titik Prediksi 14 Hari")
# Fokuskan tampilan ke area prediksi 14 hari
plt.xlim(future_df['Tanggal'].min() - pd.Timedelta(days=5), future_df['Tanggal'].max() + pd.Timedelta(days=2))
plt.ylim(future_df['Prediksi Curah Hujan'].min() - 5, future_df['Prediksi Curah Hujan'].max() + 5)
```

Gambar 44. *Source Code* Visualisasi 14 Hari Ke Depan

Kode di atas menambahkan elemen visual tambahan pada grafik prediksi curah hujan, khususnya untuk titik-titik prediksi 14 hari ke depan. Dengan menggunakan fungsi plt.scatter(), titik-titik prediksi tersebut ditampilkan dalam warna hitam dengan tepi berwarna putih dan ukuran lebih besar (s=100), sehingga lebih mudah dibedakan dari garis prediksi lainnya. Selain itu, kode ini juga menyesuaikan tampilan grafik agar fokus pada area prediksi 14 hari ke depan dengan mengatur batas sumbu X (plt.xlim) agar mencakup rentang waktu yang relevan dan batas sumbu Y (plt.ylim) agar memuat variasi nilai curah hujan dalam prediksi tersebut. Dengan penyesuaian ini, hasil visualisasi menjadi lebih jelas dan terfokus pada prediksi cuaca mendatang.



Gambar 45. Grafik Prediksi 14 Hari ke Depan

Grafik di atas menunjukkan perbandingan antara data aktual, data *training*, prediksi pada data testing, serta prediksi curah hujan untuk dua minggu ke depan. Data aktual ditampilkan dalam warna oranye, sedangkan prediksi pada data *testing* ditunjukkan dengan garis merah. Prediksi dua minggu ke depan divisualisasikan menggunakan garis hijau, dengan titik-titik prediksi 14 hari ke depan ditandai menggunakan lingkaran hitam agar lebih mudah dikenali. Dari grafik ini, terlihat bahwa prediksi rata-rata curah hujan dua minggu ke depan

yaitu 12 mm dengan sedikit peningkatan dalam beberapa hari pertama sebelum mendatar. Sementara itu, data aktual dan prediksi pada data testing menunjukkan fluktuasi yang lebih tajam. Dengan visualisasi ini, tren prediksi curah hujan dapat lebih mudah dipahami dan dianalisis.

Tabel di bawah ini menyajikan hasil uji performa model berdasarkan metrik *Root Mean Square Error* (RMSE). Pengujian ini bertujuan untuk mengevaluasi sejauh mana akurasi model dalam melakukan prediksi setelah melalui proses interpolasi data yang digunakan untuk mengisi nilai yang hilang. Dalam uji performa ini, jumlah *epoch* menjadi salah satu parameter utama yang diperhatikan, karena berpengaruh terhadap konvergensi model dalam menghasilkan nilai RMSE *Test* yang paling rendah. Nilai RMSE yang lebih kecil menunjukkan bahwa model memiliki tingkat kesalahan prediksi yang lebih rendah, sehingga dapat dikatakan memiliki performa yang lebih baik. Selain itu, pengujian ini juga berfungsi untuk menentukan metode interpolasi yang paling optimal dalam menangani *missing value* pada data curah hujan, sehingga dapat digunakan sebagai pendekatan terbaik dalam *preprocessing* data sebelum masuk ke tahap pemodelan prediksi.

Tabel 11. Hasil Uji Performa

Interpolasi	Epoch	RMSE Train	RMSE Test
Linear	25	13.2667	13.2502
	50	13.1583	13.1775
	75	13.0661	13.1388
	100	13.0542	13.1463
	200	12.9978	13.3208
Kuadrat	25	15.6443	14.6474
	50	15.3203	14.4355
	75	14.7959	14.3105
	100	14.7000	14.4892
	200	14.3861	14.1892
Spline Kuadratik	25	15.4166	14.4863
	50	15.0268	14.3947
	75	14.7347	14.1844
	100	14.4463	14.0745
	200	14.2627	14.3252

Dari hasil uji performa yang terdapat pada tabel di atas, performa terendah didapatkan dengan nilai *epoch* 25 dengan metode interpolasi kuadrat yang berhasil memperoleh nilai RMSE *Train* sebesar 15.6443 dan RMSE *Test*

14.6474 Sedangkan performa tertinggi didapatkan dengan nilai *epoch* 75 dengan metode interpolasi linear yang berhasil memperoleh nilai RMSE *Train* sebesar 13.0661 dan RMSE *Test* 13.1388. Konfigurasi terbaik ditemukan pada 50 *neuron*, 75 *epoch*, *batch size* 32, dan *optimizer* RMSprop.

4.7 Implementasi GUI *(Graphical User Interface)*

Dalam pengembangan GUI *(Graphical User Interface)* untuk prediksi curah hujan, peneliti memanfaatkan *framework* Streamlit. Streamlit adalah *framework* berbasis Python yang bersifat *open-source* dan dirancang untuk mempermudah pembuatan aplikasi web interaktif dalam bidang sains data serta *machine learning*. Keunggulan utama Streamlit terletak pada kemudahan penggunaannya, karena tidak memerlukan pemahaman mendalam tentang pengembangan web seperti HTML *(Hypertext Markup Language)* maupun CSS *(Cascading Style Sheets)*. Dengan demikian, pengguna tidak perlu mengatur tata letak secara kompleks.

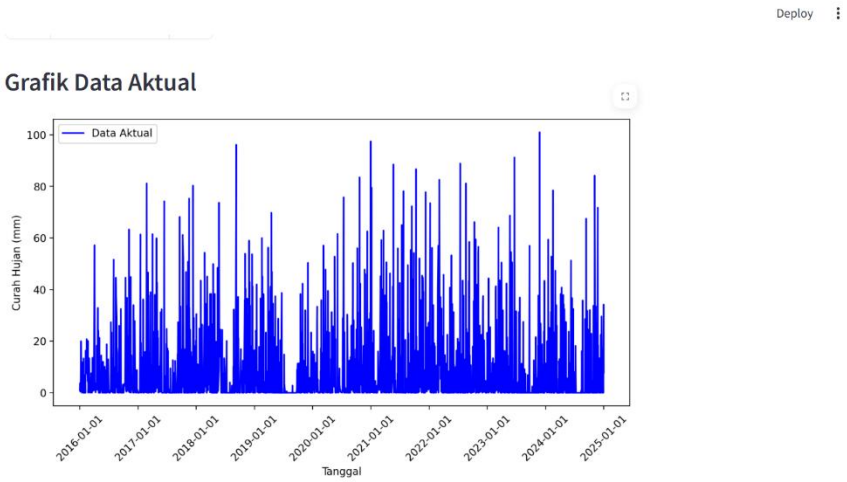
Selain itu, metode yang sederhana menjadikan Streamlit sebagai alat yang ideal untuk mengeksplorasi kumpulan data, mendemonstrasikan model *machine learning, computer vision*, pemrosesan bahasa alami, visualisasi data, serta berbagai proyek berbasis data lainnya. Setelah proses instalasi Streamlit selesai, langkah berikutnya adalah mengimpornya ke dalam proyek. Kemudian, aplikasi Streamlit dapat dijalankan dengan mengetikkan perintah streamlit run namafile.py pada terminal. Dalam penelitian ini, peneliti memanfaatkan fungsi st.file_uploader("pilih file"), yang memungkinkan pengguna untuk mengunggah atau menyematkan file ke dalam aplikasi.

Berikut adalah GUI yang dikembangkan oleh peneliti untuk prediksi curah hujan, di mana data yang *diinput* merupakan data curah hujan yang diperoleh dari Website Data Online BMKG. Proses prediksi dilakukan menggunakan metode LSTM, dan output yang dihasilkan mencakup nilai prediksi serta grafik hasil prediksi.

Gambar 46. Tampilan Utama GUI Prediksi Curah Hujan

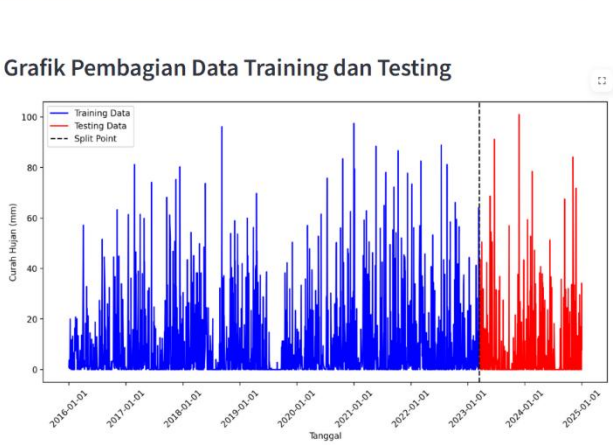
Gambar di atas menunjukkan halaman utama dari GUI yang telah dikembangkan untuk prediksi curah hujan menggunakan metode LSTM. Pada halaman ini, pengguna dapat mengunggah dataset dalam format CSV melalui fitur *drag and drop* atau dengan menekan tombol *Browse files*. Setelah file berhasil diunggah, sistem akan menampilkan data aktual dalam bentuk tabel untuk memastikan bahwa data telah dimuat dengan benar.

Tabel ini terdiri dari kolom Tanggal, yang merepresentasikan waktu pencatatan data, serta kolom RR *(Rainfall Rate)*, yang menunjukkan jumlah curah hujan dalam satuan milimeter. Tampilan ini dirancang agar pengguna dapat dengan mudah memasukkan data yang diperlukan sebelum melanjutkan ke tahap analisis dan prediksi. Dengan adanya visualisasi data awal, pengguna dapat melakukan pengecekan terhadap dataset sebelum memproses lebih lanjut menggunakan model prediksi.



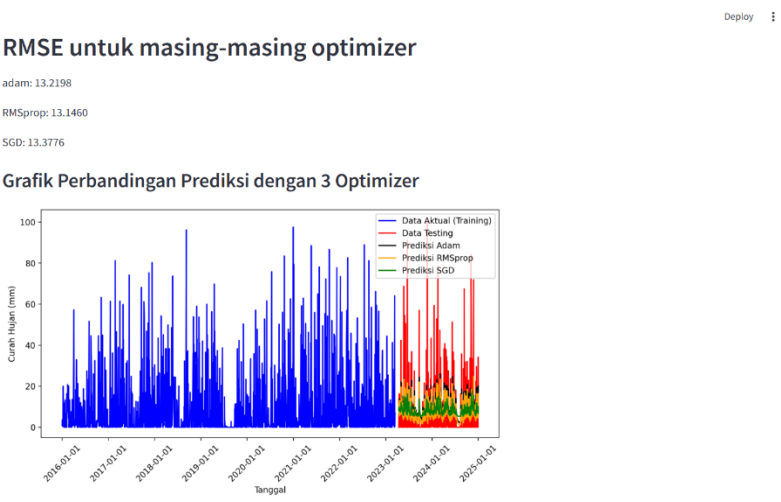
Gambar 47. Grafik Data Aktual

Setelah pengguna mengunggah dataset, sistem akan menampilkan grafik data aktual seperti yang ditunjukkan pada gambar di atas. Grafik ini berfungsi untuk memberikan gambaran visual mengenai pola curah hujan dalam rentang waktu tertentu. Sumbu horizontal (x) merepresentasikan tanggal pencatatan, sedangkan sumbu vertikal (y) menunjukkan jumlah curah hujan dalam satuan milimeter. Dalam grafik ini, data yang ditampilkan merupakan hasil interpolasi, yang telah dilakukan untuk mengatasi data yang hilang atau tidak terukur. Penggunaan interpolasi bertujuan untuk memastikan bahwa model prediksi memiliki data yang lebih lengkap dan akurat dalam proses pembelajaran. Dengan adanya visualisasi ini, pengguna dapat memahami tren curah hujan sebelum melanjutkan ke tahap pemodelan dan prediksi lebih lanjut.



Gambar 48. Grafik Pembagian Data *Training* dan *Testing*

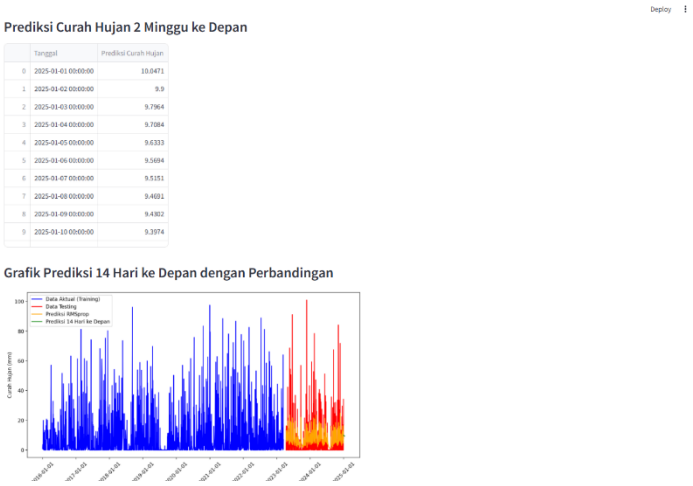
Setelah melihat grafik data aktual, langkah selanjutnya adalah membagi data menjadi dua bagian, yaitu data *training* dan data *testing*. Grafik di atas menunjukkan proses pembagian data dengan jelas, di mana data *training* ditandai dengan warna biru dan data *testing* dengan warna merah. Garis putus-putus yang terlihat pada grafik menunjukkan titik pemisahan *(split point)* antara data *training* dan *testing*. Data sebelum titik tersebut digunakan untuk melatih model *(training)*, sementara data setelahnya digunakan untuk menguji performa model *(testing)*. Proses ini penting agar model dapat belajar dari pola historis sebelum diuji dengan data yang belum pernah dilihat sebelumnya. Dengan adanya visualisasi ini, pengguna dapat memahami bagaimana data terbagi dan memastikan bahwa proporsi antara *training* dan *testing* sudah sesuai untuk menghasilkan model prediksi yang optimal.



Gambar 49. Grafik Perbandingan Evaluasi Model

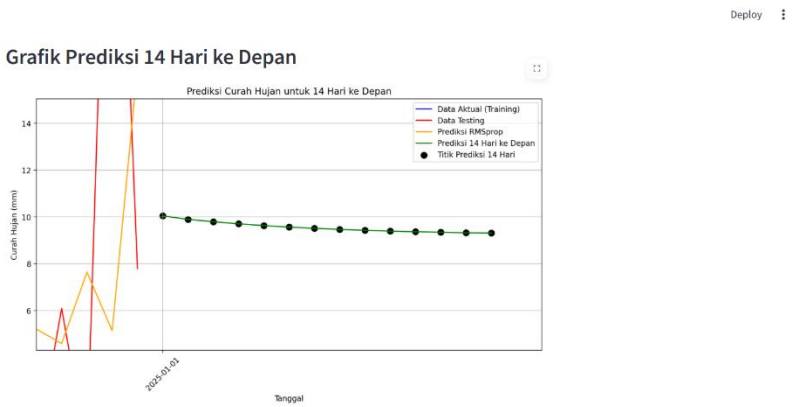
Gambar di atas menunjukkan halaman evaluasi model dengan menggunakan tiga optimizer berbeda, yaitu Adam, RMSprop, dan SGD. Nilai *Root Mean Squared Error* (RMSE) untuk masing-masing optimizer ditampilkan, di mana RMSprop memiliki nilai RMSE paling rendah (13.1460), diikuti oleh Adam (13.2198), dan SGD (13.3776). Grafik di bawahnya memperlihatkan perbandingan hasil prediksi dengan ketiga optimizer tersebut. Data aktual untuk training ditampilkan dalam warna biru, sementara data *testing* ditandai dengan warna merah. Prediksi menggunakan optimizer Adam ditampilkan dalam warna oranye, RMSprop dalam warna kuning, dan SGD dalam warna hijau.

Dari grafik ini, dapat diamati bahwa prediksi dari ketiga optimizer masih memiliki fluktuasi yang cukup tinggi, terutama pada bagian data *testing*. Namun, perbedaan performa antar-optimizer dapat dilihat dari tingkat kesesuaian hasil prediksi terhadap data aktual. RMSprop yang memiliki RMSE terendah menunjukkan bahwa model dengan optimizer ini lebih akurat dibandingkan Adam dan SGD.



Gambar 50. Grafik Prediksi 14 Hari Ke Depan dengan Perbandingan

Setelah melakukan prediksi curah hujan untuk 14 hari ke depan, hasilnya ditampilkan dalam bentuk tabel yang mencantumkan tanggal dan nilai prediksi curah hujan. Grafik di bawah tabel menunjukkan perbandingan antara data aktual, data *testing*, serta hasil prediksi menggunakan *optimizer* RMSprop. Data aktual yang digunakan untuk *training* ditampilkan dalam warna biru, sementara data testing ditandai dengan warna merah. Prediksi yang dihasilkan dengan *optimizer* RMSprop ditampilkan dalam warna kuning, dan prediksi untuk 14 hari ke depan ditampilkan dalam warna hijau. Dari grafik ini, dapat diamati bahwa data aktual memiliki fluktuasi yang cukup tinggi, sementara hasil prediksi cenderung lebih stabil dan tidak terlalu ekstrem.



Gambar 51. GUI Grafik Prediksi Prediksi 14 Hari Ke Depan

Pada grafik di atas menampilkan prediksi 14 hari ke depan dengan lebih jelas, data aktual yang digunakan untuk *training* ditampilkan dalam warna biru, sementara data *testing* ditandai dengan warna merah. Hasil prediksi menggunakan optimizer RMSprop ditampilkan dalam warna oranye, dan prediksi untuk 14 hari ke depan ditampilkan dalam warna hijau dengan titik-titik hitam yang menandai setiap nilai prediksi.

Dari grafik ini, dapat diamati bahwa data *testing* mengalami fluktuasi yang cukup tajam, sedangkan hasil prediksi untuk 14 hari ke depan cenderung lebih stabil dengan nilai yang tidak terlalu bervariasi berkisar antara 9 hingga 10 mm per hari. Hal ini menunjukkan bahwa model yang digunakan menghasilkan prediksi yang relatif halus, yang mungkin mengindikasikan adanya efek *smoothing* dari proses prediksi. Selain itu, nilai prediksi yang lebih rendah dibandingkan dengan data aktual menunjukkan bahwa model mungkin perlu penyempurnaan lebih lanjut, seperti optimasi parameter atau eksplorasi metode lain untuk meningkatkan akurasi prediksi.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan analisis, perancangan, dan implementasi serta hasil pengujian dari metode interpolasi dan *deep learning* menggunakan arsitektur *Long Short Term Memory* (LSTM) untuk prediksi curah hujan di Kota Jambi, penelitian ini menunjukkan bahwa hasil prediksi yang diperoleh memiliki tingkat akurasi yang baik. Berikut adalah kesimpulan dari penelitian yang telah dilakukan:

1. Tiga metode interpolasi diuji dalam menangani data curah hujan yang hilang, yaitu interpolasi linear, kuadrat, dan spline kuadratik. Hasil evaluasi menunjukkan bahwa interpolasi linear memberikan performa terbaik dengan nilai RMSE test sebesar 13.1388, lebih rendah dari metode lainnya. Hal ini menunjukkan bahwa interpolasi linear mampu mengisi data hilang secara sederhana namun efektif, tanpa menambahkan *noise* yang tidak perlu ke dalam data.
2. Model LSTM yang dikembangkan dalam penelitian ini mampu memprediksi curah hujan di Kota Jambi berdasarkan data historis dari BMKG. Tahapan penelitian meliputi pengumpulan data, *preprocessing*, Pembuatan Model, pengujian model, denormalisasi, dan evaluasi menggunakan RMSE.
3. Model LSTM diujicobakan dengan berbagai kombinasi jumlah *neuron*, *epoch*, *batch size*, dan *optimizer*. Konfigurasi terbaik ditemukan pada 50 *neuron*, 75 *epoch*, *batch size* 32, dan *optimizer* RMSprop. Konfigurasi ini menghasilkan nilai Train RMSE sebesar 13.0661 dan Test RMSE sebesar 13.1388, yang tergolong dalam kategori "baik" berdasarkan klasifikasi RMSE. Alasan pemilihan ini adalah karena kombinasi tersebut mampu mencapai keseimbangan antara bias dan varians. Jumlah neuron yang tidak terlalu besar menghindari *overfitting*, sedangkan 75 *epoch* memberikan cukup waktu bagi model untuk belajar pola historis tanpa *overtraining*. *Optimizer* RMSprop dipilih karena secara konsisten menghasilkan nilai *error* yang paling rendah dibanding Adam dan SGD. RMSprop secara adaptif menyesuaikan learning rate terhadap parameter, yang sangat berguna pada data time series dengan fluktuasi tinggi seperti curah hujan
4. Model prediktif yang dikembangkan berhasil diintegrasikan ke dalam GUI berbasis Streamlit, yang dapat menampilkan data aktual, memisahkan data *training* dan *testing*, menampilkan hasil evaluasi

tiga *optimizer*, serta memprediksi curah hujan 14 hari ke depan. Ini menunjukkan bahwa hasil penelitian tidak hanya berhenti pada pembangunan model, tetapi juga dikembangkan menjadi prototipe sistem informasi yang aplikatif dan mendukung pengambilan keputusan.

5.2 Saran

Penelitian ini diharapkan dapat dikembangkan lebih lanjut oleh peneliti lain di masa yang akan datang. Beberapa saran yang dapat diberikan untuk penelitian selanjutnya adalah:

1. Menerapkan metode *tuning hyperparameter* otomatis.
   Penelitian ini melakukan *tuning* secara manual. Peneliti selanjutnya dapat menerapkan metode optimasi seperti GridSearchCV atau Bayesian Optimization untuk memperoleh konfigurasi LSTM yang lebih optimal secara sistematis.
2. Mengintegrasikan lebih banyak variabel cuaca *(multivariate model).*
   Model saat ini hanya menggunakan satu fitur: curah hujan. Peneliti selanjutnya dapat menambahkan fitur eksternal seperti suhu, kelembaban, atau tekanan udara untuk melihat pengaruhnya terhadap akurasi model.
3. Mengembangkan GUI menjadi sistem prediksi berbasis web atau mobile.
   GUI pada penelitian ini masih berjalan secara lokal. Peneliti selanjutnya disarankan untuk membangun versi web atau aplikasi *mobile* agar sistem prediksi dapat digunakan oleh BPBD, petani, atau masyarakat umum secara langsung.

DAFTAR PUSTAKA

Agungnoe. (2024). *Pentingnya Memahami Cuaca dan Pengaruhnya ke Pertanian*. https://ugm.ac.id/id/berita/pentingnya-memahami-cuaca-dan-pengaruhnya-ke-pertanian/

Alqahtani, A., Ali, M., Xie, X., & Jones, M. W. (2021). Deep time-series clustering: A review. In *Electronics (Switzerland)* (Vol. 10, Issue 23). MDPI. https://doi.org/10.3390/electronics10233001

Alqahtani, A., Xie, X., Deng, J., & Jones, M. W. (2018). A Deep Convolutional Auto-Encoder with Embedded Clustering. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 4058–4062. https://doi.org/10.1109/ICIP.2018.8451506

Arrofiqoh, E. N., & Harintaka, H. (2018). IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI TANAMAN PADA CITRA RESOLUSI TINGGI. *GEOMATIKA*, *24*(2), 61. https://doi.org/10.24895/jig.2018.24-2.810

Arwansyah, A., Suryani, S., SY, H., Usman, U., Ahyuna, A., & Alam, S. (2022). Time Series Forecasting Menggunakan Deep Gated Recurrent Units. *Digital Transformation Technology*, *4*(1), 410–416. https://doi.org/10.47709/digitech.v4i1.4141

Ashari, M. L., & Sadiki, M. (2020). *PREDIKSI DATA TRANSAKSI PENJUALAN TIME SERIES MENGGUNAKAN REGRESI LSTM* (Vol. 9, Issue 1). https://doi.org/https://doi.org/10.23887/janapati.v9i1.19140

BMKG. (2021). *Buku Saku_KLIMATOLOGI_bnew* . https://iklim.bmkg.go.id/bmkgadmin/storage/brosur/Buku%20Saku_KLIMATOLOGI_bnew%20.pdf

BMKG. (2024). *Tugas dan Fungsi BMKG*. Badan Meteorologi, Klimatologi, Dan Geofisika. https://www.bmkg.go.id/profil/?p=tugas-fungsi

Brownlee, J. (2018). *Deep Learning for Time Series Forecasting Predict the Future with MLPs, CNNs and LSTMs in Python*.

Carnegie, M. D. A., & Chairani, C. (2023). Perbandingan Long Short Term Memory (LSTM) dan Gated Recurrent Unit (GRU) Untuk Memprediksi Curah Hujan. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *7*(3), 1022. https://doi.org/10.30865/mib.v7i3.6213

Colah. (2015, August 27). *Understanding LSTM Networks*. Cola's Blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Ericko, T., Dolok Lauro, M., & Handhayani, T. (2023). *Jurnal Ilmu Komputer dan Sistem Informasi PREDIKSI HARGA PANGAN DI PASAR TRADISIONAL KOTA SURABAYA DENGAN METODE LSTM*. https://www.bi.go.id/hargapangan

Farikhul Firdaus, R., & Paputungan, I. V. (2022). Prediksi Curah Hujan di Kota Bandung Menggunakan Metode Long Short Term Memory. *Jurnal Penelitian Inovatif*, *2*(3), 453–460. https://doi.org/10.54082/jupin.99

Febrianti, S. (2019). *PEMODELAN AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA) DENGAN DATA HILANG MELALUI METODE INTERPOLASI* [Universitas Lampung]. http://digilib.unila.ac.id/id/eprint/54806

Freecenta, H., Puspaningrum, E., & Maulan, H. (2022). Prediksi Curah Hujan Di Kab. Malang Menggunakan LSTM (Long Short Term Memory). *Jurnal Informatika Dan Sistem Informasi*, *3*, 51–55. https://doi.org/10.33005/jifosi.v3i1.448

Ignasius, M., & Lamabelawa, J. (2018). ANALISIS PERHITUNGAN METODE INTERPOLASI PADA DATA TIME SERIES KEMISKINAN DI NTT. *Jurnal Hoaq*, *8*(1).

Irawan, F. (2024). *PREDIKSI CURAH HUJAN MENGGUNAKAN RECURRENT NEURAL NETWORK (RNN) DAN LONG SHORT-TERM MEMORY (LSTM)*. Universitas Pakuan.

Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2017). LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, *6*, 1662–1669. https://doi.org/10.1109/ACCESS.2017.2779939

Kosasih, P. B. (2006). *Komputasi Numerik Teori dan Aplikasi*. Andi.

Lutfi, M., & Hasyim, M. (2019). *PENANGANAN DATA MISSING VALUE PADA KUALITAS PRODUKSI JAGUNG DENGAN MENGGUNAKAN METODE K-NN IMPUTATION PADA ALGORITMA C4.5* (Vol. 2, Issue 2). Online. http://jurnal.stiki-indonesia.ac.id/index.php/jurnalresistor

Luthfiarta, A., Febriyanto, A., Lestiawan, H., & Wicaksono, W. (2020). Analisa Prakiraan Cuaca dengan Parameter Suhu, Kelembaban, Tekanan Udara, dan Kecepatan Angin Menggunakan Regresi Linear Berganda. *JOINS (Journal of Information System)*, *5*(1), 10–17. https://doi.org/10.33633/joins.v5i1.2760

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197. https://doi.org/https://doi.org/10.1016/j.trc.2015.03.014

Maulana, D. (2024). *PREDIKSI JALAN BERLUBANG MENGGUNAKAN ALGORITMA LONG SHORT-TERM MEMORY (LSTM) PADA DINAS PU BINA MARGA DAN CIPTA KARYA PROVINSI JAWA TENGAH*. Universitas Semarang.

Muhtadi, M. M., Dhandy Friyadi, M., & Rahmani, A. (2019). *Analisis GUI Testing pada Aplikasi E-Commerce menggunakan Katalon*. https://doi.org/https://doi.org/10.35313/irwns.v10i1.1443

Nawangwulan, A. (2022, September 2). *Mengenal Faktor yang Mempengaruhi Curah Hujan*. HarianHaluan.Com.

Ningrum, A. A., Syarif, I., Gunawan, A. I., Satriyanto, E., Muchtar, R., Informatika, D. T., Komputer, D., Elektronika, P., & Surabaya, N. (2021). *ALGORITMA DEEP LEARNING-LSTM UNTUK MEMPREDIKSI UMUR TRANSFORMATOR*. *8*(3), 539–548. https://doi.org/10.25126/jtiik.202184587

Nurhikmat, T. (2018). *IMPLEMENTASI DEEP LEARNING UNTUK IMAGE CLASSIFICATION MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL*

NETWORK (CNN) PADA CITRA WAYANG GOLEK. https://doi.org/10.13140/RG.2.2.10880.53768

Oyen, S. (2018). *Forecasting multivariate time series data using neural networks*. NTNU.

Prasetyo, V. R., Lazuardi, H., Mulyono, A. A., & Lauw, C. (2021). Penerapan Aplikasi RapidMiner Untuk Prediksi Nilai Tukar Rupiah Terhadap US Dollar Dengan Metode Regresi Linier. *Jurnal Nasional Teknologi Dan Sistem Informasi (TEKNOSI)*, *7*(1), 8–17.

Prayoga Dhenanta, R., & Kholifah, I. B. (2022). *Prediksi Curah Hujan Bulanan Kabupaten Trenggalek Tahun 2022 dan 2023 Menggunakan Metode ARIMA*.

Prihandono, A. (2022). *MODEL PREDIKSI CURAH HUJAN PADA KABUPATEN BOGOR MENGGUNAKAN ALGORITMA C5.0*. Universitas Teknokrat Indonesia .

Purnama, A. (2021). *Implementasi Metode Deep Learning Dengan Menggunakan Algoritma Convolution Neural Network (CNN) Pada Citra Tulisan Tangan Aksara Sunda*.

Putratama, R. (2020, April 23). *Suhu Udara Terik Apakah Dipicu Pemanasan Global?* Badan Meteorologi, Klimatologi, Dan Geofisika.

Putri, G. A. M. A., Hendayanti, N. P. N., & Nurhidayati, M. (2017). Pemodelan Data Deret Waktu Dengan Autoregressive Integrated Moving Average Dan Logistic Smoothing Transition Autoregressive. In *Maulida Nurhidayati* (Vol. 1, Issue 1). Ni Putu Nanik Hendayanti.

Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS One*, *15*(1), e0227222.

Riko Anshori Prasetya, M., & Mudi Priyatno, A. (2023). *Penanganan Imputasi Missing Values pada Data Time Series dengan Menggunakan Metode Data Mining. 5*, 56–62. https://doi.org/10.37034/jidt.v5i1.324

Rizki, M., Basuki, S., & Azhar, Y. (2020). Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory Untuk Prediksi Curah Hujan Kota Malang. *REPOSITOR*, *2*(3), 331–338.

Rizky Ismail, M., Zain, A., Dewantoro, F., Pratiwi, D., & Sipil, T. (2023). *Perhitungan Data Curah Hujan yang Hilang dengan Menggunakan Metode Interpolasi Linier* (Vol. 04, Issue 02). http://jim.teknokrat.ac.id/index.php/tekniksipilJurnalTeknikSipil

Rofi, A. (2019). *PERAMALAN DATA DERET WAKTU MUSIMAN DENGAN METODE SEASONAL AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (SARIMA) DAN METODE DEKOMPOSISI PADA DATA JUMLAH PENUMPANG MELALUI BANDARA POLONIA TAHUN 2009-2018*. Universitas lampung.

Ruhiat, D., & Suwanda, C. (2019). Peramalan Data Deret Waktu Berpola Musiman Menggunakan Metode Regresi Spektral (Studi Kasus: Debit Sungai Citarum-Nanjung). In *Jurnal Teorema: Teori dan Riset Matematika* (Vol. 4, Issue 1).

Sofiyani, S., & Permanasari, Y. (2023). Penerapan Metode Cubic Spline Interpolation untuk Menentukan Peluang Kematian pada Tabel Mortalita. *Jurnal Riset Matematika*, 29–36. https://doi.org/10.29313/jrm.v3i1.1735

Sulistyo Budi, R., Patmasari, R., & Saidah, S. (2021). *KLASIFIKASI CUACA MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) WEATHER CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK (CNN) METHOD*.

Susilo, B. (2021). *Mengenal Iklim dan Cuaca di Indonesia*. DIVA PRESS. https://books.google.co.id/books?id=Cl5zEAAAQBAJ

Sutriawan, S., Fanani, A. Z., Alzami, F., & Basuki, R. S. (2023). Deep Learning Jaringan Saraf Tiruan Untuk Pemecahan Masalah Deteksi Penyakit Daun Apel. *Jurnal Teknologi Informasi Dan Komunikasi (TIKomSiN)*, *11*(1), 35. https://doi.org/10.30646/tikomsin.v11i1.729

Tanjung, R., Listiani, A., & Lestari, F. (2024). Prediksi Multivariate Time Series Parameter Cuaca Menggunakan Long Short-Term Memory (LSTM). *Seminar Nasional Sains Data*.

Tarkus, D., Sompie, S. R. U. A., & Jacobus, A. (2020). *Implementasi Metode Recurrent Neural Network pada Pengklasifikasian Kualitas Telur Puyuh*.

Tian, C., Ma, J., Zhang, C., & Zhan, P. (2018). A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies*, *11*(12). https://doi.org/10.3390/en11123493

Toyib, M., Decky, T., Pratama, K., & Aqil, I. (2024). Prediksi Kondisi Cuaca di Kabupaten Bayuwangi Menggunakan Metode LSTM. In *Jurnal Ilmiah Sains dan Teknologi* (Vol. 2, Issue 7). http://dataonline.bmkg.go.id

Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Long short-term memory based operation log anomaly detection. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 236–242.

Wicaksono, A. (2022). PENGARUH FENOMENA LA NINA TERHADAP ANOMALI CURAH HUJAN BULANAN DI SULAWESI SELATAN THE EFFECT OF THE LA NINA PHENOMENON ON MONTHLY RAINFALL ANOMALIES IN SOUTH SULAWESI. In *MARET* (Vol. 2, Issue 3).

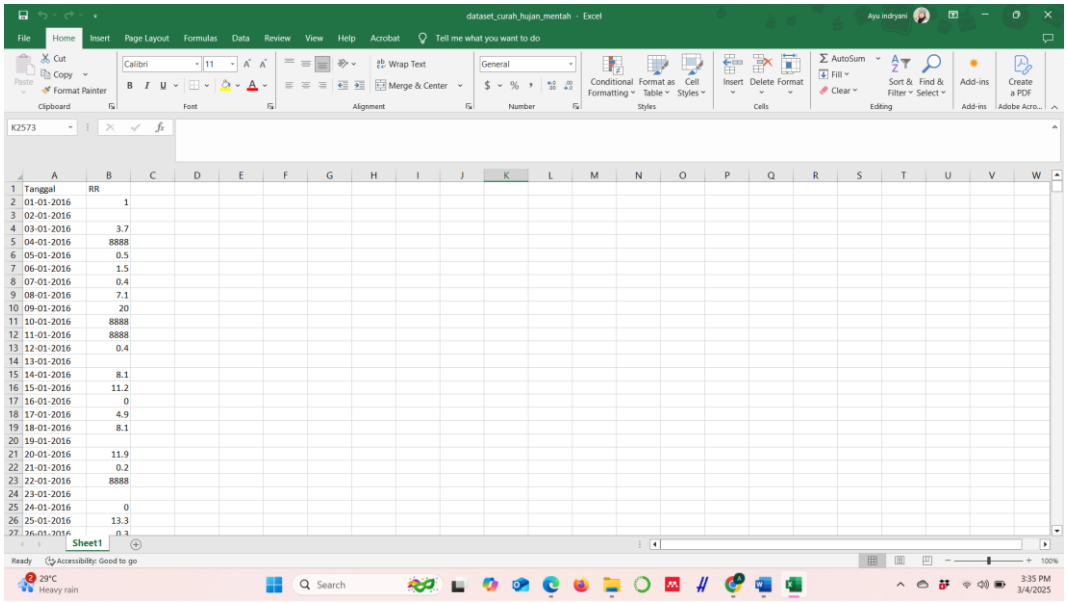
Widianti, A., & Pratama, I. (2024). PENANGANAN MISSING VALUES DAN PREDIKSI DATA TIMBUNAN SAMPAH BERBASIS MACHINE LEARNING. *Rabit : Jurnal Teknologi Dan Sistem Informasi Univrab*, *9*(2), 242–251. https://doi.org/10.36341/rabit.v9i2.4789

Wiranda, L., & Sadikin, M. (2019). Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk Pt. Metiska Farma. *Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI*, *8*(3), 184–196.

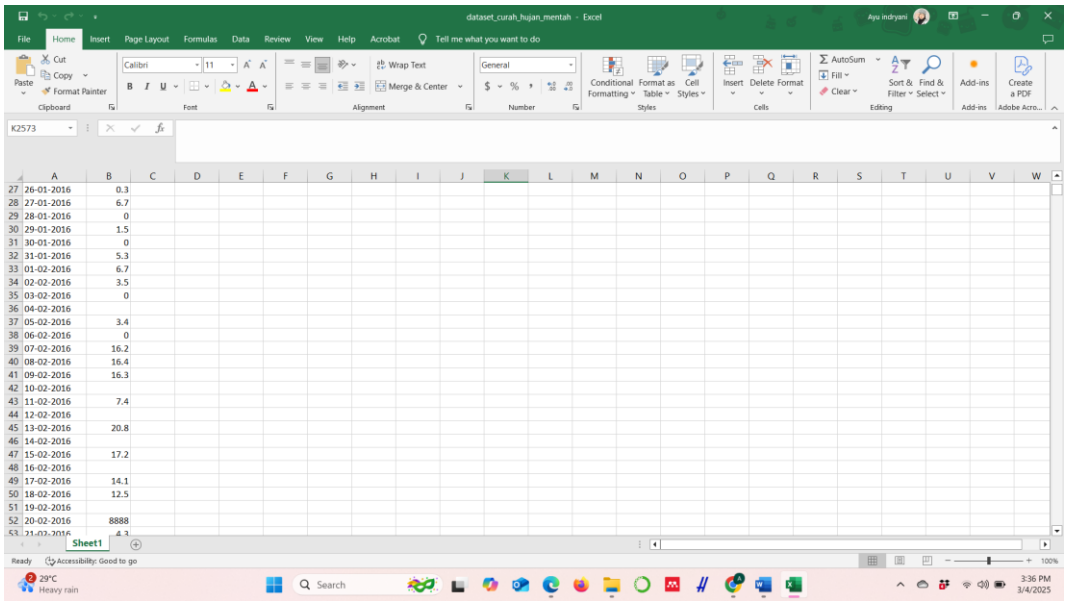
Zahara, S., & Ilmiddafiq, M. B. (2019). Prediksi Indeks Harga Konsumen Menggunakan Metode Long Short Term Memory (LSTM) Berbasis Cloud Computing. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *3*(3), 357–363.

LAMPIRAN

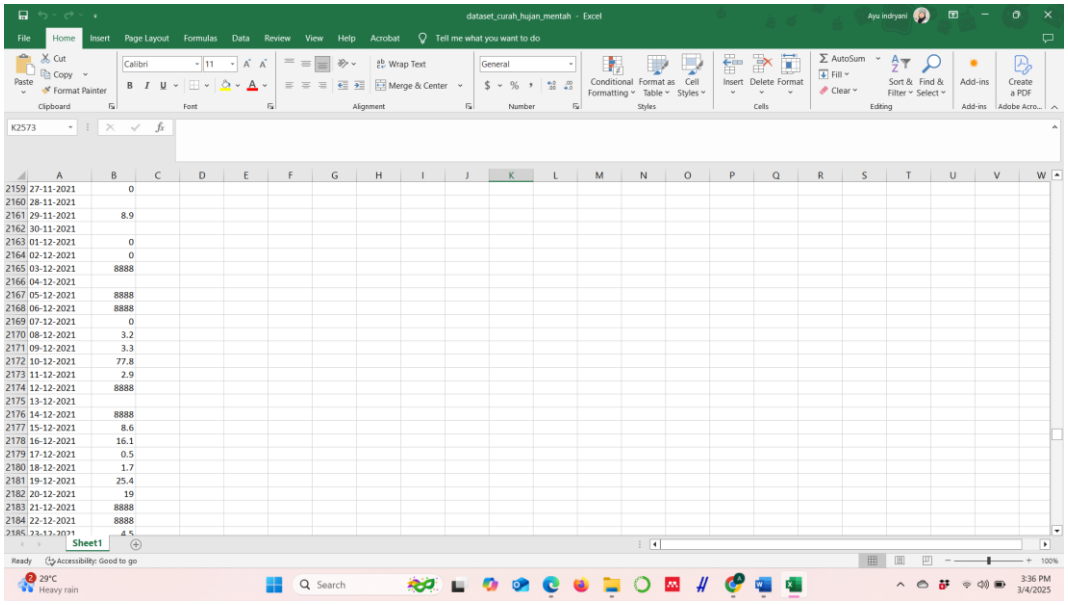
Lampiran 1. Contoh Dataset Curah Hujan Kota Jambi

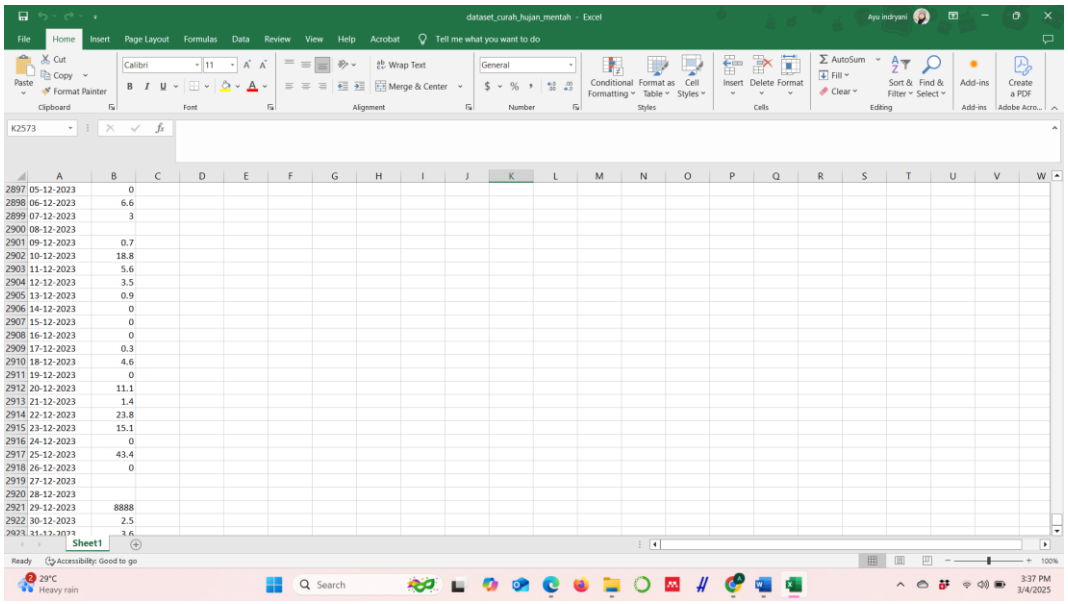


	A	B
1	Tanggal	RR
2	01-01-2016	1
3	02-01-2016	
4	03-01-2016	3.7
5	04-01-2016	8888
6	05-01-2016	0.5
7	06-01-2016	1.5
8	07-01-2016	0.4
9	08-01-2016	7.1
10	09-01-2016	20
11	10-01-2016	8888
12	11-01-2016	8888
13	12-01-2016	0.4
14	13-01-2016	
15	14-01-2016	8.1
16	15-01-2016	11.2
17	16-01-2016	0
18	17-01-2016	4.9
19	18-01-2016	8.1
20	19-01-2016	
21	20-01-2016	11.9
22	21-01-2016	0.2
23	22-01-2016	8888
24	23-01-2016	
25	24-01-2016	0
26	25-01-2016	13.3



	A	B
27	26-01-2016	0.3
28	27-01-2016	6.7
29	28-01-2016	0
30	29-01-2016	1.5
31	30-01-2016	0
32	31-01-2016	5.3
33	01-02-2016	6.7
34	02-02-2016	3.5
35	03-02-2016	0
36	04-02-2016	
37	05-02-2016	3.4
38	06-02-2016	0
39	07-02-2016	16.2
40	08-02-2016	16.4
41	09-02-2016	16.3
42	10-02-2016	
43	11-02-2016	7.4
44	12-02-2016	
45	13-02-2016	20.8
46	14-02-2016	
47	15-02-2016	17.2
48	16-02-2016	
49	17-02-2016	14.1
50	18-02-2016	12.5
51	19-02-2016	
52	20-02-2016	8888





Lampiran 2. *Source Code* LSTM dengan Interpolasi Linear

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
```

```
import matplotlib.dates as mdates
# Membaca dataset
df = pd.read_csv ('dataset_ch_fix.csv')
df
# Konversi kolom 'Tanggal' ke format datetime dengan otomatis mendeteksi
format
df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce', dayfirst=True)  #
dayfirst=True untuk format DD/MM/YYYY
# Ganti nilai 8888 dan 9999 dengan NaN
df['RR'] = df['RR'].replace([8888, 9999], np.nan)

# Isi nilai NaN dengan interpolasi linear
df['RR'] = df['RR'].interpolate(method='linear')
df
print("Grafik Data Setelah Interpolasi")  # Cetak ke console
# Membuat plot
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(df['Tanggal'], df['RR'], label="Data Setelah Interpolasi", color='blue')
# Mengatur sumbu X agar hanya menampilkan tanggal 1 Januari tiap tahun
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)
ax.set_xlabel("Tanggal")
ax.set_ylabel("Curah Hujan (mm)")
ax.legend()
plt.show()
# Normalisasi Data
scaler = MinMaxScaler()
df["RR_scaled"] = scaler.fit_transform(df[["RR"]])
df
# Buat Train-Test Split (Pastikan Tanggal tetap ada)
split = int(len(df) * 0.8)
train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
# Visualisasi Pembagian Data Training dan Testing
plt.figure(figsize=(12, 6))
# Plot data Training (80% pertama)
```

```
plt.plot(train_data['Tanggal'], train_data['RR_scaled'], label="Training Data",
color='blue')
# Plot data Testing (20% terakhir)
plt.plot(test_data['Tanggal'], test_data['RR_scaled'], label="Testing Data",
color='red')
# Garis pembatas di akhir training (awal testing)
split_date = train_data['Tanggal'].iloc[-1]  # Ambil tanggal terakhir dari data
training
plt.axvline(split_date, color='black', linestyle='--', label="Split Point")  # Garis
vertikal
# Menyesuaikan sumbu X untuk hanya menampilkan tanggal 1 Januari setiap
tahun
ax = plt.gca()  # Mendapatkan sumbu saat ini
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)  # Memutar label tanggal agar tidak tumpang tindih
# Label dan Judul
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Scaled)")
plt.title("Pembagian Data Training dan Testing")
plt.legend()
# Tampilkan plot
plt.show()
# Menentukan jumlah data training dan testing
jumlah_training = len(train_data)
jumlah_testing = len(test_data)
# Menampilkan jumlahnya
print(f"Jumlah data training: {jumlah_training}")
print(f"Jumlah data testing: {jumlah_testing}")
#  Fungsi Membuat Sequence Data
def create_sequences(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps):
        X.append(data.iloc[i:i + n_steps, 1].values)  # RR_scaled ada di index 1
        y.append(data.iloc[i + n_steps, 1])
    return np.array(X), np.array(y)
n_steps = 30
X_train, y_train = create_sequences(train_data, n_steps)
```

```
X_test, y_test = create_sequences(test_data, n_steps)
# Simpan Tanggal yang sesuai untuk prediksi nanti
train_dates = train_data["Tanggal"].iloc[n_steps:].values
test_dates = test_data["Tanggal"].iloc[n_steps:].values
# Membangun model LSTM
model = Sequential()
# Gunakan Input layer sebagai lapisan pertama
model.add(Input(shape=(n_steps, 1)))
model.add(LSTM(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
# Kompilasi model
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(), loss='mse')
# Reshape data agar sesuai dengan input LSTM (samples, timesteps, features)
X_train = X_train.reshape(-1, n_steps, 1)
X_test = X_test.reshape(-1, n_steps, 1)
# Melatih model
history = model.fit(X_train, y_train, epochs=75, batch_size=32)
y_train_pred = model.predict(X_train.reshape(-1, n_steps, 1))
y_test_pred = model.predict(X_test.reshape(-1, n_steps, 1))
# Denormalisasi hasil prediksi
y_train_pred_original = scaler.inverse_transform(y_train_pred)
y_test_pred_original = scaler.inverse_transform(y_test_pred)
# Menghitung RMSE
rmse_train =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_train.reshape(-1, 1)),
y_train_pred_original))
rmse_test =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)),
y_test_pred_original))
print(f"RMSE Train: {rmse_train:.4f}")
print(f"RMSE Test: {rmse_test:.4f}")
#  Buat DataFrame Hasil Prediksi (dengan Tanggal)
df_pred_train = pd.DataFrame({'Tanggal': train_dates, 'Prediksi_Train':
y_train_pred_original.flatten()})
df_pred_test = pd.DataFrame({'Tanggal': test_dates, 'Prediksi_Test':
y_test_pred_original.flatten()})
```

```
#  Plot Hasil Prediksi dengan Tanggal
plt.figure(figsize=(12,6))
plt.plot(df_pred_train['Tanggal'], df_pred_train['Prediksi_Train'], label="Prediksi
Train", color='blue')
plt.plot(df_pred_test['Tanggal'], df_pred_test['Prediksi_Test'], label="Prediksi Test",
color='red')
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Denormalized)")
plt.legend()
plt.show()
```

Lampiran 3. *Source Code* LSTM dengan Interpolasi Kuadrat

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
# Membaca dataset
df = pd.read_csv ('dataset_ch_fix.csv')
df
# Konversi kolom 'Tanggal' ke format datetime dengan otomatis mendeteksi
format
df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce', dayfirst=True)  #
dayfirst=True untuk format DD/MM/YYYY
print(df)
# Ganti nilai 8888 dan 9999 dengan NaN
df['RR'] = df['RR'].replace([8888, 9999], np.nan)
# Isi nilai NaN dengan interpolasi linear
df['RR'] = df['RR'].interpolate(method='quadratic')
df
print("Grafik Data Setelah Interpolasi")  # Cetak ke console
# Membuat plot
fig, ax = plt.subplots(figsize=(10, 5))
```

```
ax.plot(df['Tanggal'], df['RR'], label="Data Setelah Interpolasi", color='blue')
# Mengatur sumbu X agar hanya menampilkan tanggal 1 Januari tiap tahun
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)
ax.set_xlabel("Tanggal")
ax.set_ylabel("Curah Hujan (mm)")
ax.legend()
plt.show()
# Normalisasi Data
scaler = MinMaxScaler()
df["RR_scaled"] = scaler.fit_transform(df[["RR"]])
df
# Buat Train-Test Split (Pastikan Tanggal tetap ada)
split = int(len(df) * 0.8)
train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
# Visualisasi Pembagian Data Training dan Testing
plt.figure(figsize=(12, 6))
# Plot data Training (80% pertama)
plt.plot(train_data['Tanggal'], train_data['RR_scaled'], label="Training Data",
color='blue')
# Plot data Testing (20% terakhir)
plt.plot(test_data['Tanggal'], test_data['RR_scaled'], label="Testing Data",
color='red')
# Garis pembatas di akhir training (awal testing)
split_date = train_data['Tanggal'].iloc[-1]  # Ambil tanggal terakhir dari data
training
plt.axvline(split_date, color='black', linestyle='--', label="Split Point")  # Garis
vertikal
# Menyesuaikan sumbu X untuk hanya menampilkan tanggal 1 Januari setiap
tahun
ax = plt.gca()  # Mendapatkan sumbu saat ini
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)  # Memutar label tanggal agar tidak tumpang tindih
# Label dan Judul
plt.xlabel("Tanggal")
```

```
plt.ylabel("Curah Hujan (Scaled)")
plt.title("Pembagian Data Training dan Testing")
plt.legend()
# Tampilkan plot
plt.show()
# Menentukan jumlah data training dan testing
jumlah_training = len(train_data)
jumlah_testing = len(test_data)
# Menampilkan jumlahnya
print(f"Jumlah data training: {jumlah_training}")
print(f"Jumlah data testing: {jumlah_testing}")
#  Fungsi Membuat Sequence Data
def create_sequences(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps):
        X.append(data.iloc[i:i + n_steps, 1].values)  # RR_scaled ada di index 1
        y.append(data.iloc[i + n_steps, 1])
    return np.array(X), np.array(y)
n_steps = 30
X_train, y_train = create_sequences(train_data, n_steps)
X_test, y_test = create_sequences(test_data, n_steps)
# Simpan Tanggal yang sesuai untuk prediksi nanti
train_dates = train_data["Tanggal"].iloc[n_steps:].values
test_dates = test_data["Tanggal"].iloc[n_steps:].values
# Membangun model LSTM
model = Sequential()
# Gunakan Input layer sebagai lapisan pertama
model.add(Input(shape=(n_steps, 1)))  # n_steps adalah jumlah langkah waktu
yang sudah didefinisikan sebelumnya
model.add(LSTM(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
# Kompilasi model
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(), loss='mse')
# Reshape data agar sesuai dengan input LSTM (samples, timesteps, features)
X_train = X_train.reshape(-1, n_steps, 1)
X_test = X_test.reshape(-1, n_steps, 1)
```

```
# Melatih model
history = model.fit(X_train, y_train, epochs=75, batch_size=32)
y_train_pred = model.predict(X_train.reshape(-1, n_steps, 1))
y_test_pred = model.predict(X_test.reshape(-1, n_steps, 1))
# Denormalisasi hasil prediksi
y_train_pred_original = scaler.inverse_transform(y_train_pred)
y_test_pred_original = scaler.inverse_transform(y_test_pred)
# Menghitung RMSE
rmse_train =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_train.reshape(-1, 1)),
y_train_pred_original))
rmse_test =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)),
y_test_pred_original))
print(f"RMSE Train: {rmse_train:.4f}")
print(f"RMSE Test: {rmse_test:.4f}")
#  Buat DataFrame Hasil Prediksi (dengan Tanggal)
df_pred_train = pd.DataFrame({'Tanggal': train_dates, 'Prediksi_Train':
y_train_pred_original.flatten()})
df_pred_test = pd.DataFrame({'Tanggal': test_dates, 'Prediksi_Test':
y_test_pred_original.flatten()})
#  Plot Hasil Prediksi dengan Tanggal
plt.figure(figsize=(12,6))
plt.plot(df_pred_train['Tanggal'], df_pred_train['Prediksi_Train'], label="Prediksi
Train", color='blue')
plt.plot(df_pred_test['Tanggal'], df_pred_test['Prediksi_Test'], label="Prediksi Test",
color='red')
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Denormalized)")
plt.legend()
plt.show()
```

Lampiran 4. *Source Code* LSTM dengan Interpolasi Spline Kuadratik

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
```

```
from tensorflow.keras import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
# Membaca dataset
df = pd.read_csv ('dataset_ch_fix.csv')
df
# Konversi kolom 'Tanggal' ke format datetime dengan otomatis mendeteksi
format
df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce', dayfirst=True)  #
dayfirst=True untuk format DD/MM/YYYY
print(df)
# Ganti nilai 8888 dan 9999 dengan NaN
df['RR'] = df['RR'].replace([8888, 9999], np.nan)
# Isi nilai NaN dengan interpolasi linear
df['RR'] = df['RR'].interpolate(method='spline', order = 2)
df
print("Grafik Data Setelah Interpolasi")  # Cetak ke console
# Membuat plot
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(df['Tanggal'], df['RR'], label="Data Setelah Interpolasi", color='blue')
# Mengatur sumbu X agar hanya menampilkan tanggal 1 Januari tiap tahun
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)
ax.set_xlabel("Tanggal")
ax.set_ylabel("Curah Hujan (mm)")
ax.legend()
plt.show()
# Normalisasi Data
scaler = MinMaxScaler()
df["RR_scaled"] = scaler.fit_transform(df[["RR"]])
df
# Buat Train-Test Split (Pastikan Tanggal tetap ada)
split = int(len(df) * 0.8)
train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
```

```
# Visualisasi Pembagian Data Training dan Testing
plt.figure(figsize=(12, 6))
# Plot data Training (80% pertama)
plt.plot(train_data['Tanggal'], train_data['RR_scaled'], label="Training Data",
color='blue')
# Plot data Testing (20% terakhir)
plt.plot(test_data['Tanggal'], test_data['RR_scaled'], label="Testing Data",
color='red')
# Garis pembatas di akhir training (awal testing)
split_date = train_data['Tanggal'].iloc[-1]  # Ambil tanggal terakhir dari data
training
plt.axvline(split_date, color='black', linestyle='--', label="Split Point")  # Garis
vertikal
# Menyesuaikan sumbu X untuk hanya menampilkan tanggal 1 Januari setiap
tahun
ax = plt.gca()  # Mendapatkan sumbu saat ini
ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)  # Memutar label tanggal agar tidak tumpang tindih
# Label dan Judul
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Scaled)")
plt.title("Pembagian Data Training dan Testing")
plt.legend()
# Tampilkan plot
plt.show()
# Menentukan jumlah data training dan testing
jumlah_training = len(train_data)
jumlah_testing = len(test_data)
# Menampilkan jumlahnya
print(f"Jumlah data training: {jumlah_training}")
print(f"Jumlah data testing: {jumlah_testing}")
#  Fungsi Membuat Sequence Data
def create_sequences(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps):
        X.append(data.iloc[i:i + n_steps, 1].values)  # RR_scaled ada di index 1
        y.append(data.iloc[i + n_steps, 1])
```

```
    return np.array(X), np.array(y)
n_steps = 30
X_train, y_train = create_sequences(train_data, n_steps)
X_test, y_test = create_sequences(test_data, n_steps)

# Simpan Tanggal yang sesuai untuk prediksi nanti
train_dates = train_data["Tanggal"].iloc[n_steps:].values
test_dates = test_data["Tanggal"].iloc[n_steps:].values
# Membangun model LSTM
model = Sequential()
# Gunakan Input layer sebagai lapisan pertama
model.add(Input(shape=(n_steps, 1)))  # n_steps adalah jumlah langkah waktu
yang sudah didefinisikan sebelumnya
model.add(LSTM(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
# Kompilasi model
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(), loss='mse')
# Reshape data agar sesuai dengan input LSTM (samples, timesteps, features)
X_train = X_train.reshape(-1, n_steps, 1)
X_test = X_test.reshape(-1, n_steps, 1)
# Melatih model
history = model.fit(X_train, y_train, epochs=75, batch_size=32)
y_train_pred = model.predict(X_train.reshape(-1, n_steps, 1))
y_test_pred = model.predict(X_test.reshape(-1, n_steps, 1))
# Denormalisasi hasil prediksi
y_train_pred_original = scaler.inverse_transform(y_train_pred)
y_test_pred_original = scaler.inverse_transform(y_test_pred)
# Menghitung RMSE
rmse_train =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_train.reshape(-1, 1)),
y_train_pred_original))
rmse_test =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)),
y_test_pred_original))
print(f"RMSE Train: {rmse_train:.4f}")
print(f"RMSE Test: {rmse_test:.4f}")
```

```
#  Buat DataFrame Hasil Prediksi (dengan Tanggal)
df_pred_train = pd.DataFrame({'Tanggal': train_dates, 'Prediksi_Train':
y_train_pred_original.flatten()})
df_pred_test = pd.DataFrame({'Tanggal': test_dates, 'Prediksi_Test':
y_test_pred_original.flatten()})
#  Plot Hasil Prediksi dengan Tanggal
plt.figure(figsize=(12,6))
plt.plot(df_pred_train['Tanggal'], df_pred_train['Prediksi_Train'], label="Prediksi
Train", color='blue')
plt.plot(df_pred_test['Tanggal'], df_pred_test['Prediksi_Test'], label="Prediksi Test",
color='red')
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (Denormalized)")
plt.legend()
plt.show()
```

Lampiran 5. *Source Code* GUI Prediksi Curah Hujan dengan LSTM

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler

import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Judul Aplikasi
st.title("Prediksi Curah Hujan Dengan LSTM")
```

```
# Upload File Dataset
uploaded_file = st.file_uploader("Upload Dataset CSV", type=["csv"])

if uploaded_file:
    df = pd.read_csv(uploaded_file)
    df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce', dayfirst=True)
    df['RR'] = df['RR'].replace([8888, 9999], np.nan)
    df['RR'] = df['RR'].interpolate(method='linear')

    st.subheader("Data Aktual")
    st.write(df.head())

    # Grafik Data Setelah Interpolasi
    st.write("### Grafik Data Aktual")
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(df['Tanggal'], df['RR'], label="Data Aktual", color='blue')
    ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.xticks(rotation=45)
    ax.set_xlabel("Tanggal")
    ax.set_ylabel("Curah Hujan (mm)")
    ax.legend()
    st.pyplot(fig)

    # Grafik Pembagian Data Training dan Testing (BELUM dinormalisasi)
    st.write("### Grafik Pembagian Data Training dan Testing")
    split = int(len(df) * 0.8)
    train_data_raw = df.loc[:split, ["Tanggal", "RR"]]
    test_data_raw = df.loc[split:, ["Tanggal", "RR"]]

    fig2, ax2 = plt.subplots(figsize=(12, 6))
    ax2.plot(train_data_raw['Tanggal'], train_data_raw['RR'], label="Training Data",
color='blue')
    ax2.plot(test_data_raw['Tanggal'], test_data_raw['RR'], label="Testing Data",
color='red')
    split_date = train_data_raw['Tanggal'].iloc[-1]
    ax2.axvline(split_date, color='black', linestyle='--', label="Split Point")
    ax2.xaxis.set_major_locator(mdates.YearLocator(base=1))
```

```
    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.xticks(rotation=45)
    ax2.set_xlabel("Tanggal")
    ax2.set_ylabel("Curah Hujan (mm)")
    ax2.legend()
    st.pyplot(fig2)
    # Normalisasi Data (setelah visualisasi)
    scaler = MinMaxScaler()
    df["RR_scaled"] = scaler.fit_transform(df[["RR"]])
    # Split Data
    split = int(len(df) * 0.8)
    train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
    test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
# Membuat Sequence Data
    def create_sequences(data, n_steps):
        X, y = [], []
        for i in range(len(data) - n_steps):
            X.append(data.iloc[i:i + n_steps, 1].values)
            y.append(data.iloc[i + n_steps, 1])
        return np.array(X), np.array(y)

    n_steps = 30
    X_train, y_train = create_sequences(train_data, n_steps)
    X_test, y_test = create_sequences(test_data, n_steps)

    # Reshape Data
    X_train = X_train.reshape(-1, n_steps, 1)
    X_test = X_test.reshape(-1, n_steps, 1)

    optimizers = ['adam', 'RMSprop', 'SGD']
    rmse_results = {}
    predictions = {}

    for opt in optimizers:
        model = Sequential()
        model.add(LSTM(50, activation='relu', input_shape=(n_steps, 1)))
        model.add(Dropout(0.2))
        model.add(Dense(1))
```

```
        model.compile(optimizer=opt, loss='mse')

        model.fit(X_train, y_train, epochs=75, batch_size=32)

        y_test_pred = model.predict(X_test)
        y_test_pred_original = scaler.inverse_transform(y_test_pred)

        rmse =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)),
y_test_pred_original))
        rmse_results[opt] = rmse
        predictions[opt] = y_test_pred_original
        best_optimizer = min(rmse_results, key=rmse_results.get)
    st.write("## RMSE untuk masing-masing optimizer")
    for opt, rmse in rmse_results.items():
        st.write(f"{opt}: {rmse:.4f}")

    # Grafik Perbandingan Prediksi dengan 3 Optimizer
    st.write("### Grafik Perbandingan Prediksi dengan 3 Optimizer")
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(train_data['Tanggal'],
scaler.inverse_transform(train_data["RR_scaled"].values.reshape(-1, 1)),
label="Data Aktual (Training)", color='blue')
    ax.plot(test_data['Tanggal'].iloc[n_steps:],
scaler.inverse_transform(y_test.reshape(-1, 1)), label="Data Testing", color='red')
    ax.plot(test_data['Tanggal'].iloc[n_steps:], predictions['adam'], label="Prediksi
Adam", color='black')
    ax.plot(test_data['Tanggal'].iloc[n_steps:], predictions['RMSprop'],
label="Prediksi RMSprop", color='orange')
    ax.plot(test_data['Tanggal'].iloc[n_steps:], predictions['SGD'], label="Prediksi
SGD", color='green')
    ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.xticks(rotation=45)
    ax.set_xlabel("Tanggal")
    ax.set_ylabel("Curah Hujan (mm)")
    ax.legend()
    st.pyplot(fig)
```

```
    # Prediksi 14 Hari ke Depan
    future_predictions = []
    last_sequence = X_test[-1]
    for _ in range(14):
        next_pred = model.predict(last_sequence.reshape(1, n_steps, 1))
        future_predictions.append(next_pred[0, 0])
        last_sequence = np.roll(last_sequence, -1)
        last_sequence[-1] = next_pred[0, 0]

    future_predictions =
scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
    future_dates = pd.date_range(start=test_data['Tanggal'].iloc[-1], periods=15,
freq='D')[1:]
    future_df = pd.DataFrame({'Tanggal': future_dates, 'Prediksi Curah Hujan':
future_predictions.flatten()})
    st.subheader("Prediksi Curah Hujan 2 Minggu ke Depan")
    st.write(future_df)

    # Grafik Prediksi 14 Hari ke Depan dengan Perbandingan Data Aktual dan
Prediksi Optimizer Terbaik
    st.write("### Grafik Prediksi 14 Hari ke Depan dengan Perbandingan")
    fig, ax = plt.subplots(figsize=(12, 6))
    # 1️⃣ Data Aktual dari Training (Biru)
    ax.plot(train_data['Tanggal'],
scaler.inverse_transform(train_data["RR_scaled"].values.reshape(-1, 1)),
label="Data Aktual (Training)", color='blue')
    ax.plot(test_data['Tanggal'].iloc[n_steps:],
scaler.inverse_transform(y_test.reshape(-1, 1)), label="Data Testing", color='red')
    ax.plot(test_data['Tanggal'].iloc[n_steps:], predictions[best_optimizer],
label=f"Prediksi {best_optimizer}", color='orange')
    ax.plot(future_df['Tanggal'], future_df['Prediksi Curah Hujan'], label="Prediksi
14 Hari ke Depan", color='green')
    ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.xticks(rotation=45)
    ax.set_xlabel("Tanggal")
    ax.set_ylabel("Curah Hujan (mm)")
    ax.legend()
```

```
    st.pyplot(fig)

    st.write("### Grafik Prediksi 14 Hari ke Depan")
    # Plot Grafik
    fig, ax = plt.subplots(figsize=(12, 6))

    # Data Training (Biru)
    ax.plot(train_data['Tanggal'],
scaler.inverse_transform(train_data["RR_scaled"].values.reshape(-1, 1)),
            label="Data Aktual (Training)", color='blue')

    # Data Testing (Merah)
    ax.plot(test_data['Tanggal'].iloc[n_steps:],
scaler.inverse_transform(y_test.reshape(-1, 1)),
            label="Data Testing", color='red')

    # Prediksi Model (Orange)
    ax.plot(test_data['Tanggal'].iloc[n_steps:], predictions[best_optimizer],
            label=f"Prediksi {best_optimizer}", color='orange')
     # Prediksi 14 Hari ke Depan (Hijau)
    ax.plot(future_df['Tanggal'], future_df['Prediksi Curah Hujan'],
            label="Prediksi 14 Hari ke Depan", color='green')

    # Titik Prediksi (Hitam)
    ax.scatter(future_df['Tanggal'], future_df['Prediksi Curah Hujan'],
              color='black', edgecolors='white', s=100, label="Titik Prediksi 14 Hari")

    # Fokus pada area prediksi
    ax.set_xlim(future_df['Tanggal'].min() - pd.Timedelta(days=5),
                future_df['Tanggal'].max() + pd.Timedelta(days=2))

    ax.set_ylim(future_df['Prediksi Curah Hujan'].min() - 5,
                future_df['Prediksi Curah Hujan'].max() + 5)

    # Format tanggal
    ax.xaxis.set_major_locator(mdates.YearLocator(base=1))
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.xticks(rotation=45)
```

```
# Label dan Judul
plt.xlabel("Tanggal")
plt.ylabel("Curah Hujan (mm)")
plt.title("Prediksi Curah Hujan untuk 14 Hari ke Depan")
plt.legend()
plt.grid()

# Tampilkan di Streamlit
st.pyplot(fig)
```