Pengembangan Sistem Informasi Prediksi Kualitas Udara Dengan Menggunakan Metode *Long Short Term Memory* Berbasis Website

Ananta Alief Rochmanditya[1], Salamun Rohman Nudin[2]

D4 Manajemen Informatika, Fakultas Vokasi, Universitas Negeri Surabaya Kampus Unesa 1, Jalan ketintang, Surabaya

[1]ananta.20074@mhs.unesa.ac.id
[2]salamunrohman@unesa.ac.id

Abstrak - **Keberlanjutan hidup makhluk hidup dan kesehatan manusia bergantung pada kualitas udara yang baik. Polusi udara yang semakin parah menjadi masalah besar yang berdampak pada kesehatan dan lingkungan, terutama di kota besar seperti Surabaya. Sistem prediksi kualitas udara yang tepat sangat penting. Dalam penelitian ini, metode Long Short-Term Memory (LSTM) dan website digunakan untuk membuat sistem untuk memprediksi kualitas udara. LSTM adalah salah satu algoritma pembelajaran mendalam yang terbaik untuk menangani data berupa deret waktu. Sistem, yang dibangun menggunakan framework Streamlit, memiliki kemampuan untuk memprediksi polusi udara berdasarkan lima parameter setiap minggu. MAPE rata-rata adalah 11,61% (akurasi rata-rata = 88,39%), dengan hasil evaluasi sebagai berikut: PM10 (8,99%), SO2 (12,27%), CO (13,23%), O3 (12,23%), dan NO2 (9,34%). Hasil ini menunjukkan bahwa sistem memiliki tingkat akurasi yang tinggi dan layak digunakan sebagai alat bantu prediksi kualitas udara. Dengan adanya sistem ini, diharapkan masyarakat dan pihak terkait dapat lebih mudah memantau dan mengantisipasi penurunan kualitas udara di masa mendatang.**
Kata kunci : Kualitas udara, Sistem prediksi, Streamlit, Deep Learning, LSTM.

Abstarck **- The sustainability of living things and human health depend on good air quality. Increasingly severe air pollution is a major problem that impacts health and the environment, especially in large cities like Surabaya. An accurate air quality prediction system is crucial. In this study, the Long Short-Term Memory (LSTM) method and a website were used to create a system for predicting air quality. LSTM is one of the best deep learning algorithms for handling time series data. The system, built using the Streamlit framework, has the ability to predict air pollution based on five parameters every week. The average MAPE was 11.61% (average accuracy = 88.39%), with the following evaluation results: PM10 (8.99%), SO2 (12.27%), CO (13.23%), O3 (12.23%), and NO2 (9.34%). These results indicate that the system has a high level of accuracy and is suitable for use as an air quality prediction tool. With this system, it is hoped that the public and related parties can more easily monitor and anticipate future air quality declines.**
Keywords: Air quality, Prediction system, Streamlit, Deep Learning, LSTM.

I. PENDAHULUAN

Keberlanjutan hidup makhluk hidup dan Kesehatan manusia bergantung pada kualitas udara yang baik. Polusi udara, perubahan iklim, dan kegiatan manusia dapat sangat mempengaruhi kualitas udara. Dunia sedang memperhatikan masalah kualitas udara yang buruk karena dapat membahayakan Kesehatan manusia dan makhluk hidup lainnya. Sangat penting untuk melakukan evaluasi dan perkiraan kualitas udara karena polusiudara merupakan salah satu tantangan terbesar yang dihadapi dunia saat ini. Jika Masyarakat mengetahui kualitas udara, mereka dapat mengambil tindakan pencegahan untuk mencegah penyakit [10]. Tujuan dari prediksi kualitas udara ini adalah untuk menghindari dampak negatif dari gas-gas polutan yang terhirup yang berbahaya bagi Kesehatan manusia. Prediksi data kualitas udara sangat penting untuk pengambilan Keputusan dalam pengendalian pencemaran udara [6]. Salah satu jenis arsitektur jaringan saraf tiruan berulang (RNN) yang paling umum digunakan adalah LSTM [18]. Penelitian ini akan berkonsentrasi pada pengembangan metode LSTM dengan *framework streamlit* dan Bahasa pemrograman python untuk membangun sistem informasi prediksi berbasi website. *Streamlit* adalah sebuah *framework* berbasis bahasa pemrograman Python dan bersifat *open-source* yang dibuat untuk memudahkan dalam membangun apikasi web di bidang sains data yang interaktif [20]. Dalam implementasi antarmuka [2], streamlit digunakan untuk membuat sistem rekomendasi tanaman berbasis pembelajaran kelompok. Ini menunjukan keunggulan streamlit dalam menyajikan sistem prediktif yang ringan dan mudah digunakan. Selanjutnya, dalam upaya untuk memprediksi polusi udara dengan mempertimbangkan berbagai parameter sekaligus, model

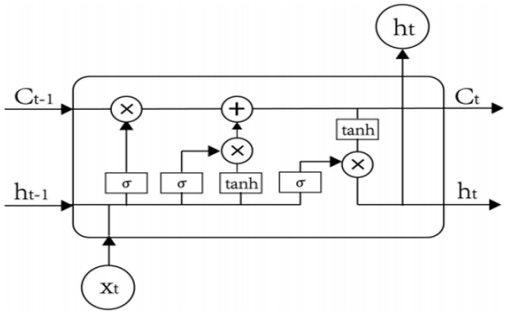
multivariat LSTM berhasil dikembangkan [12]. Ini menunjukkan keunggulan arsitektur LSTM dalam memodelkan data lingkungan yang komplek dan terus berubah.

II. TINJAUAN PUSTAKA

A. Pencemaran Udara

Pencemaran udara merupakan kondisi ketika zat atau partikel asing masuk ke atmosfer dan mengubah komposisi alami udara sehingga menurunkan kualitasnya. Udara secara alami tersusun dari berbagai gas, terutama oksigen dan nitrogen, yang penting bagi kehidupan. Ketika zat pencemar terakumulasi dalam jumlah besar dan dalam waktu lama, kualitas udara menurun dan berdampak negatif pada ekosistem. Pencemaran udara terutama disebabkan oleh aktivitas manusia dan dapat berupa gas atau partikel yang termasuk dalam beberapa kelompok seperti gas berbahaya, gas belerang, gas nitrogen, dan senyawa karbon [11].

B. *Metode Long Short Term Memory*



Gambar 1. Struktur LSTM [15]

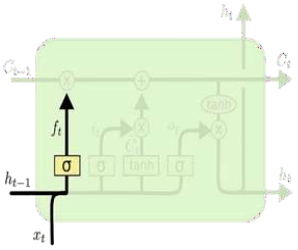
LSTM merupakan pengembangan dari Recurrent Neural Network (RNN) yang dirancang untuk menyimpan informasi jangka panjang melalui struktur gerbang seperti input gate, forget gate, dan output gate. Metode ini, yang diperkenalkan oleh Hochreiter dan Schmidhuber, mampu mempelajari pola hingga ribuan langkah sebelumnya sehingga efektif digunakan dalam berbagai bidang, termasuk peramalan. LSTM mengatasi kelemahan RNN yang cenderung kehilangan informasi masa lalu akibat masalah hilangnya memori, sehingga RNN kurang akurat dalam prediksi jangka panjang [20].

Model LSTM mengelola informasi dengan menggunakan struktur gerbang yang berfungsi untuk mempertahankan atau memperbarui memori dalam sel. Setiap sel memori terdiri dari tiga lapisan sigmoid dan satu lapisan tanh. Salah satu bagian pentingnya adalah gerbang lupa, yaitu pintu yang menentukan mana informasi yang perlu diingat dan mana yang harus dilupakan.

Gerbang ini memproses hidden state sebelumnya dan input saat ini menggunakan fungsi sigmoid untuk menghasilkan nilai antara 0 hingga 1, di mana nilai mendekati 0 berarti informasi dibuang, sedangkan nilai mendekati 1 berarti informasi dipertahankan.

1. Forget gate

Berfungsi menentukan apakah suatu informasi perlu disimpan atau dibuang. Gerbang ini menerima hidden state dari sel sebelumnya serta input saat ini, kemudian menggabungkannya dan memprosesnya menggunakan fungsi sigmoid. Hasil keluaran berupa nilai antara 0 hingga 1, di mana nilai mendekati 0 berarti informasi akan dibuang, sedangkan nilai mendekati 1 berarti informasi



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

tersebut disimpan.

Gambar 2. *Forget Gate* [14]

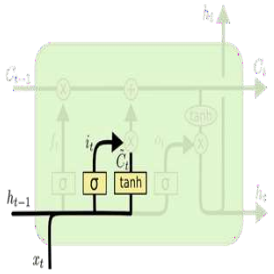
Sel memori LSTM menerima hidden state sebelumnya (ht–1) dan input saat ini (xt), lalu menggabungkannya melalui operasi konkatenasi sebelum diproses menggunakan fungsi sigmoid. Forget gate berperan menentukan seberapa banyak informasi dari status sel sebelumnya (Ct–1) akan dipertahankan pada status sel saat ini (Ct). Nilai keluaran berada pada rentang 0 hingga 1, dengan nilai 1 menunjukkan bahwa informasi sepenuhnya dipertahankan, sedangkan nilai 0 berarti informasi dibuang.

$$f_t = \sigma\left(W_t \cdot \left[h_{t-1}, x_t\right] + b_f\right) \qquad (1)$$

Keterangan:

f_t	: *Forget gate*
σ	: Fungsi sigmoid
W_t	: 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*

2. Input gate



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

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] \; + \; b_C\right)$$

Gambar 3. *Input Gate* [14]

Memproses hidden state sebelumnya dan input saat ini dengan menggabungkannya, lalu mengolahnya menggunakan fungsi sigmoid dan tanh. Fungsi sigmoid menghasilkan nilai 0 hingga 1 untuk menentukan bagian informasi yang akan diperbarui—nilai mendekati 0 berarti tidak penting, sedangkan nilai mendekati 1 menunjukkan informasi penting. Sementara itu, fungsi tanh menghasilkan nilai antara –1 hingga 1 yang membantu sel mempelajari dan membentuk informasi baru secara lebih efektif. Gerbang input menentukan seberapa banyak informasi dari input saat ini (xt) yang akan disimpan ke dalam status sel (Ct), sehingga mencegah informasi yang tidak signifikan masuk ke memori. Gerbang ini memiliki dua fungsi utama, salah satunya adalah memilih bagian dari status sel yang perlu diperbarui, yang ditentukan oleh lapisan sigmoid.

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \quad (2)$$

Keterangan:

i_t : *Input gate*

σ : Fungsi sigmoid

W_i : Nilai *weight* untuk *input gate*

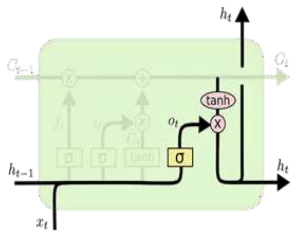
h_{t-1} : Nilai *output* sebelum orde ke-t

x_t : Nilai *input* pada orde ke-t

b_i : Nilai bias pada *input gate*

3. Output gate

mengontrol hidden state yang diteruskan ke sel berikutnya dengan memproses hidden state sebelumnya dan input saat ini melalui sigmoid, kemudian mengalikan hasilnya dengan cell state yang diproses tanh untuk menentukan informasi akhir yang disimpan dan diteruskan.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Gambar 4. *Output Gate* [14]

Gerbang output mengontrol seberapa banyak keadaan sel yang akan dibuang.

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

x_t : Nilai *input* pada orde ke-t

b_o : Nilai bias pada *output gate*

Nilai output akhir sel didefinisikan sebagai:

$$h_t = O_t * tanh(C_t)$$

Keterangan:

h_t : Nilai *output* orde ke t

O_t : *Output gate*

$Tanh$: Fungsi tanh

C_t : *Cell state*

$$O_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \quad (3)$$

Keterangan:

O_t : *Output gate*

σ : Fungsi sigmoid

W_o : Nilai *weight* untuk *output gate*

3. Optimasi

Pemilihan optimizer yang tepat sangat penting dalam pelatihan model deep learning seperti LSTM, karena memengaruhi kecepatan konvergensi dan akurasi akhir model.

I. Adam (Adaptive Moment Estimation):

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \; v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \; \hat{m}_t = \frac{m_t}{1-\beta_1^t} \; \hat{v}_t = \frac{v_t}{1-\beta_2^t} \; \theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (4)$$

- α: learning rate
- β1,β2: parameter decay (default β1=0.9,β2=0.009)
- ϵ: nilai kecil untuk mencegah pembagian nol (default 10-8)

Adam merupakan optimizer yang menggabungkan momentum dan RMSProp, menyesuaikan learning rate tiap parameter secara adaptif, efisien, dan tahan terhadap data tidak stasioner serta noise [9].

II. Nadam (Nesterov-accelerated Adam):

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \hat{m}_t = \frac{m_t}{1-\beta_1^t} \hat{v}_t = \frac{v_t}{1-\beta_2^t} \theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \hat{m}_t = \frac{m_t}{1-\beta_1^t} \hat{v}_t = \frac{v_t}{1-\beta_2^t} \theta_{t+1} = \theta_t - \alpha \cdot \left(\frac{\beta_1 \cdot \hat{m}_t + \frac{(1-\beta_1)}{1-\beta_1^t} \cdot g_t}{\sqrt{\hat{v}_t}+\epsilon} \right) \quad (5)$$

Nadam Adalah varian Adam yang menggunakan momentum Nesterov untuk update parameter, meningkatkan stabilitas kecepatan konvergensi dan performa terutama pada LSTM [1].

4. Google Colab

Platform berbasis *cloud* seperti *Jupyter Notebook* memungkinkan pengguna untuk menulis, menyimpan, dan menjalankan kode Python tanpa harus menginstal perangkat lunak, terutama untuk keperluan *machine learning* dan *deep learning* [19].

5. Visual Studio Code

Editor kode gratis dari Microsoft yang berjalan di Linux, Windows, dan Mac, mendukung berbagai bahasa pemrograman, debugging, Git, serta pengembangan aplikasi web, seluler, dan cloud [16].

6. Python

Bahasa pemrograman populer yang fleksibel dan sederhana, banyak digunakan dalam data science, analisis data, visualisasi, machine learning, serta pengembangan web dan perangkat lunak [5]. Python mendukung perhitungan statistik kompleks, pembuatan visualisasi interaktif, dan pembangunan algoritma machine learning, dengan bantuan pustaka populer seperti Pandas, NumPy, Matplotlib, dan TensorFlow. Keunggulannya meliputi kemampuan analisis data yang lengkap dan pengembangan website, termasuk pengelolaan server, perutean URL, keamanan, serta framework seperti Streamlit dan Flask.

7. Streamlit

Framework open-source berbasis Python ini memudahkan pembuatan aplikasi web yang bisa diakses dan dioperasikan untuk menjalankan, menampilkan hasil, serta menganalisis model *machine learning* dan *deep learning*, serta menjelajahi data tanpa perlu menulis banyak kode [7].

8. Dataset DLH Kota Surabaya

kumpulan data yang digunakan untuk analisis atau pelatihan model, bisa berbentuk terstruktur atau tidak terstruktur. Penelitian ini menggunakan dataset harian ISPU dari dua Stasiun Pemantau Kualitas Udara Ambien (SPKUA) Kota Surabaya tahun 2022–2024, mencakup lima parameter utama: PM10, SO_2, CO, O_3, dan NO_2. Data telah dibersihkan, dinormalisasi, dan disusun sebagai time series untuk melatih model LSTM, dengan output berupa prediksi mingguan dan visualisasi grafik melalui aplikasi.

Tabel 1. Fitur Dataset ISPU Kota Surabaya

No	Fitur	Type Data
1	***WAKTU***	Time / DateTime
2	***PM10***	Floating Point (float)
3	***SO2***	Floating Point (float)
4	***CO***	Floating Point (float)
5	***O3***	Floating Point (float)
6	***NO2***	Floating Point (float)

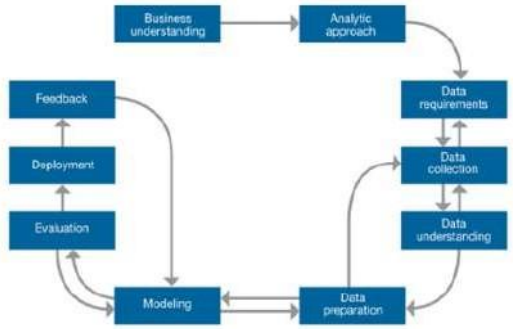
- WAKTU: Atribut waktu pencatatan data, digunakan sebagai indeks time series.
- PM10: Partikel udara ≤10 µm dari debu, kendaraan, atau industri; dapat menimbulkan gangguan pernapasan.
- SO_2: Gas beracun dari pembakaran bahan bakar fosil; menyebabkan iritasi saluran pernapasan dan hujan asam.
- CO: Gas tidak berwarna/bau dari pembakaran tidak sempurna; dapat mengganggu distribusi oksigen dalam darah.
- O_3: Ozon troposferik dari reaksi sinar matahari dengan emisi; polutan yang mengiritasi mata dan pernapasan.
- NO_2: Gas dari pembakaran suhu tinggi; penyebab peradangan saluran pernapasan dan pembentukan ozon troposferik serta PM.

9. IBM Data Science

Pendekatan untuk menganalisis data besar dan semi-terstruktur dengan tujuan mengekstraksi informasi berguna. IBM mengembangkan metodologi 10 tahap yang membentuk proses berulang dalam penggunaan data [4].

a. Business Understanding: Menentukan masalah, tujuan, dan solusi untuk memastikan keberhasilan proyek.
b. Analytic Approach: Memilih teknik analitik; penelitian ini menggunakan Deep Learning (LSTM) untuk prediksi polutan udara.

c. Data Requirements: Menentukan data yang relevan dan representatif, seperti waktu dan konsentrasi polutan.
d. Data Collection: Mengumpulkan data ISPU Kota Surabaya (SO_2, CO, O_3, NO_2).
e. Data Understanding: Memahami kualitas dan karakteristik data menggunakan statistik dan visualisasi.
f. Data Preparation: Membersihkan dan mentransformasi data agar siap untuk pelatihan model LSTM.
g. Modelling: Membangun model prediktif dengan Deep Learning untuk menangkap pola sekuensial dan non-linear.
h. Evaluation: Menilai kinerja model terhadap data uji, memastikan akurasi dan menghindari overfitting/underfitting.
i. Deployment: Menerapkan model ke lingkungan produksi atau pengujian untuk digunakan operasional.
j. Feedback: Mengumpulkan umpan balik untuk menyempurnakan model dan meningkatkan akurasi serta kegunaannya.



Gambar 5. Diagram Alur IBM Data Science [4]

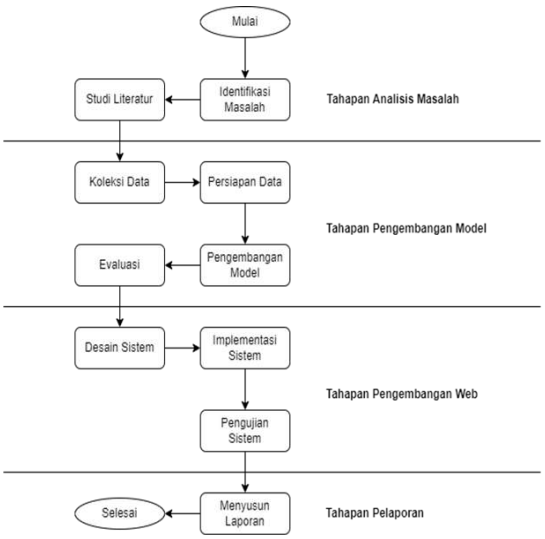
10. Waterfall

Model pengembangan perangkat lunak klasik yang bersifat linear dan terstruktur [14] di mana setiap tahap harus diselesaikan sebelum melanjutkan ke tahap berikutnya. Model ini digunakan untuk sistem prediksi kualitas udara dan terdiri dari beberapa tahap:

1. Requirement Analysis: Mengidentifikasi dan mendokumentasikan kebutuhan sistem melalui diskusi, observasi, survei, atau wawancara dengan stakeholder [17].
2. System & Software Design: Merancang solusi teknis termasuk antarmuka pengguna, arsitektur sistem, diagram alir data, dan database sebagai dasar implementasi.
3. Implementation & Unit Testing: Mengembangkan kode program dan menguji setiap modul secara individual untuk memastikan fungsionalitasnya.
4. Integration & System Testing: Menggabungkan seluruh modul dan menguji sistem secara menyeluruh untuk mendeteksi kesalahan atau bug.
5. Operation & Maintenance: Menerapkan sistem ke lingkungan nyata dan melakukan pemeliharaan berkala, termasuk perbaikan kesalahan, penyesuaian, dan peningkatan performa berdasarkan umpan balik pengguna.

Gambar 6. *Hierarki Waterfall*

III. METODOLOGI PENELITIAN



Gambar 7. Diagram Alur Penelitian

Metodologi penelitian menggabungkan IBM Data Science Methodology dan Waterfall dalam empat tahapan: analisis masalah, pengembangan model LSTM, pembuatan

aplikasi web untuk visualisasi prediksi, dan penyusunan laporan hasil penelitian [4].

A. Tahap Identifikasi Masalah

1. Tahap analisis masalah dalam penelitian ini meliputi penentuan topik, perumusan masalah, penetapan tujuan, serta pencarian referensi terkait. Langkah pertama adalah identifikasi masalah, dilakukan melalui observasi isu kualitas udara untuk memahami fokus penelitian dan membatasi ruang lingkup. Selanjutnya dilakukan studi literatur dengan mengumpulkan jurnal dan artikel relevan mengenai kualitas udara, metode LSTM, dan penerapan deep learning sebagai dasar teori dan acuan bagi langkah penelitian berikutnya.

2. Identifikasi Masalah

Langkah awal yang dilakukan adalah mengidentifikasi masalah. Proses ini dilakukan lewat observasi terhadap isu-isu kualitas udara yang pernah dibahas dalam penelitian atau artikel sebelumnya. Tujuannya adalah untuk memahami masalah yang akan diteliti, sekaligus membatasi ruang lingkup penelitian agar tidak melebar ke mana-mana dan tetap sesuai kebutuhan.

3. Studi Literatur

Pada Setelah masalah berhasil diidentifikasi, penulis lanjut ke studi literatur. Pada tahap ini, berbagai sumber seperti jurnal dan artikel yang relevan mulai dikumpulkan. Fokusnya adalah pada topik kualitas udara, metode Long Short Term Memory (LSTM), dan penerapan deep learning. Studi ini berguna sebagai dasar teori dan acuan untuk menyusun langkah-langkah penelitian selanjutnya.

B. Tahap II Pengembangan Model

Tahap ini diadaptasi dari IBM Data Science Methodology dan terdapat empat proses dalam pengembangannya. Pada tahap ini data akan diolah hingga menjadi data model yang digunakan untuk analisis data menggunakan metode Long Short Term Memory.

A. Pengumpulan Data

Pada tahap ini, penulis menggunakan dataset yang diperoleh dari Dinas Lingkungan Hidup Kota Surabaya. Dataset tersebut berjudul "Indeks Standar Pencemaran Udara di Kota Surabaya" dan memuat data pengukuran kualitas udara dari tahun 2022 hingga 2024. Data dikumpulkan pada tanggal 10 Maret 2025 melalui permintaan resmi kepada instansi terkait. Dataset ini terdiri dari 1.096 entri data harian yang mencakup lima parameter polutan utama, yaitu PM10, SO_2, CO, O_3, dan NO_2.

B. Persiapan Data

Setelah data diperoleh, penulis melakukan pemeriksaan awal untuk mendeteksi nilai kosong, duplikasi, atau inkonsistensi, kemudian melakukan pembersihan dan transformasi agar siap digunakan untuk pelatihan model. Data harian dari Dinas Lingkungan Hidup Surabaya, yang mencakup lima parameter utama pencemar udara, dibaca ke dalam dataframe Pandas. Dataframe ini memungkinkan pengolahan data secara tabular dua dimensi, mempermudah analisis, normalisasi, dan pembentukan struktur time series yang diperlukan untuk model LSTM.

C. Pengembangan Model

Penulis menggunakan metode Long Short Term Memory (LSTM) sebagai teknik pemodelan, diimplementasikan menggunakan Python dan diintegrasikan ke sistem informasi prediksi kualitas udara berbasis website. LSTM dipilih karena efektif untuk data sekuensial dan masalah deep learning, sebagaimana dibahas dalam literatur [3]. Proses pengembangan mencakup normalisasi data, pembagian dataset latih dan uji, pelatihan model, serta evaluasi menggunakan metrik kuantitatif sebelum diintegrasikan ke antarmuka web untuk penggunaan interaktif.

D Evaluasi

Setelah tahap pemodelan, dilakukan evaluasi untuk menilai kinerja model. Penelitian ini menggunakan Mean Absolute Percentage Error (MAPE), yang mengukur rata-rata selisih absolut antara nilai prediksi dan aktual dalam bentuk persentase. MAPE membantu menentukan tingkat kesalahan prediksi model. Selain MAPE, ukuran akurasi lain yang umum digunakan adalah Root Mean Square Error (RMSE).

Mean Absolute Percentage Error (MAPE)

MAPE= √(∑_(t=1)^n▒(|Yi-Ýi|)/Yi)/n (8)

Keterangan :

Yi : Nilai hasil peramalan

$Y'I$: Nilai aktual / Nilai sebenarnya

n : Jumlah data

C. Tahap III Pengembangan Website

Pengembangan sistem informasi prediksi kualitas udara menggunakan model Waterfall, yang memiliki tahapan berurutan dan sistematis mulai dari analisis kebutuhan hingga implementasi dan pemeliharaan. Tahap pertama adalah Requirement Analysis, bertujuan mengidentifikasi masalah, menentukan kebutuhan pengguna, dan merumuskan ruang lingkup sistem. Hasil dari tahap ini menjadi dasar untuk merancang sistem secara terstruktur, termasuk desain antarmuka, arsitektur website, dan alur interaksi pengguna, sehingga sistem yang dikembangkan sesuai dengan kebutuhan dan mudah digunakan.

1. Identifikasi Masalah :

- Kurangnya pengetahuan Masyarakat tentang indikator ramalan kualitas udara.
- Banyak dampak dari pencemaran udara yang dapat menyebabkan pemanasan global dan perubahan iklim.
- Kualitas udara yang buruk dapat mengakibatkan penyakit pada Masyarakat dan Makhluk Hidup.

2. Analisa Kebutuhan :

Sebelum melakukan penelitian penulis mengidentifikasi permasalahan yang akan diangkat ke dalam topik. Setelah

mendapatkan identifikasi masalah diatas, penulis akan membuat sistem informasi prediksi kualitas udara berbasis website. Sistem informasi tersebut untuk memprediksi kualitas udara pada pengguna sehingga pengguna dapat lebih muda mengetahui ramalan kualitas udara pada masa yang akan mendatang dan Sebelum melakukan penelitian penulis mengidentifikasi permasalahan yang akan diangkat ke dalam topik. Setelah mendapatkan identifikasi masalah diatas, penulis akan membuat sistem informasi prediksi kualitas udara berbasis website. Sistem informasi tersebut untuk memprediksi kualitas udara pada pengguna sehingga pengguna dapat lebih muda mengetahui ramalan kualitas udara pada masa yang akan mendatang

3. Desain Sistem :

Alur cara kerja sistem informasi prediksi kualitas udara ini merupakan:

A. Tahap 1 Input Parameter oleh Pengguna

Pengguna membuka aplikasi dan mengatur jumlah minggu yang ingin diprediksi ke depan melalui fitur slider yang tersedia pada antarmuka. Sistem akan memberikan pilihan waktu (misalnya hingga 52 minggu), yang digunakan sebagai panjang horizon prediksi.

B. Tahap 2 permintaan prediksi akan dikirimk ke back-end

Setelah pengguna menentukan jumlah minggu prediksi, sistem akan mengirimkan permintaan tersebut ke backend Streamlit. Permintaan ini memicu proses prediksi berdasarkan data terkini yang telah diproses sebelumnya.

C. Tahap 3 Proses Prediksi oleh Model

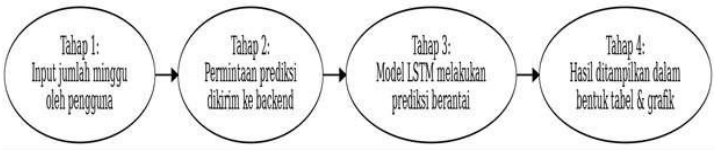
Server Streamlit akan memuat model LSTM terlatih (model_lstm.h5) dan normalizer scaler yang telah disimpan menggunakan joblib. Model kemudian melakukan proses prediksi secara recursive (berantai) berdasarkan input data 21 minggu terakhir. Untuk setiap langkah prediksi, sistem:

- Menyusun input sekuens waktu
- Melakukan normalisasi data
- Menghasilkan output prediksi
- Menggunakan output sebagai input baru untuk prediksi minggu berikutnya

D. Tahap 4: Visualisasi dan Interpretasi Hasil

Hasil prediksi dikembalikan ke antarmuka dan ditampilkan dalam bentuk:

- Tabel hasil prediksi lima parameter (PM10, SO_2, CO, O_3, NO_2)
- Grafik tren waktu
- Narasi interpretatif otomatis berdasarkan kondisi parameter
- Fitur unduhan Excel untuk menyimpan hasil prediksi.



Gambar 8. Flowchart alur kerja sistem informasi prediksi

5. Implementasi Sistem

Tahapan ini merupakan tahap penulisan kode program sesuai dengan metode analisis yaitu long short term memory dan dilakukan pengujian akurasi pada metode tersebut. Serta penulis memasukkan kode program tersebut sesuai desain sistem yang telah dibuat.

6. Pengujian Sistem

Setelah proses pengembangan sistem informasi prediksi kualitas udara selesai dilakukan, tahap selanjutnya adalah pengujian sistem untuk memastikan bahwa sistem berjalan sesuai dengan fungsionalitas yang telah dirancang. Pengujian ini bertujuan untuk mengevaluasi apakah sistem dapat memberikan output yang sesuai berdasarkan input yang diberikan, serta memastikan bahwa sistem dapat digunakan oleh masyarakat secara efektif. Penulis melakukan pengujian menggunakan metode Blackbox Testing, yaitu metode pengujian yang berfokus pada fungsionalitas sistem tanpa mengetahui struktur internal atau kode program. Pengujian dilakukan dengan memberikan berbagai macam input dan mengamati apakah keluaran yang dihasilkan sesuai dengan yang diharapkan.

D. Tahap IV Pelaporan

Tahap pelaporan merupakan tahap akhir dari penelitian tahap ini berupa proses penyusunan laporan tugas akhir.

Menyusun Laporan Tahapan ini merupakan tahap akhir dari penelitian yaitu Menyusun laporan secara tertulis dan memaparkan hasil penelitian yang telah dilakukan dalam bentuk laporan tugas akhir. Pada tahap ini peneliti memaparkan keberhasilan dan keggalan dalam penelitian beserta alasannya. Pada penyusunan laporan terdapat dua bab tambahan yaitu hasil penelitian dan pembahasan serta bab terakhir yaitu kesimpulan dari serangkaian penelitian yang telah dilakukan.

IV. HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini akan membahas hasil dari penelitian yang telah dilakukan, berdasarkan analisis data yang dikumpulkan dengan metode yang sudah dijelaskan pada Bab III. Pembahasan juga disertakan untuk menjelaskan hasil yang diperoleh.

A. Tahap Pengembangan Model

Tahap ini mencakup keseluruhan proses yang dilakukan untuk membangun dan menyempurnakan model prediksi kualitas udara berbasis Long Short-Term Memory (LSTM). Proses ini tidak hanya sebatas membangun arsitektur model, tetapi juga melibatkan akuisisi data, pemrosesan awal data mentah, eksperimen terhadap berbagai konfigurasi model, hingga evaluasi performa akhir untuk memastikan keandalan sistem yang dikembangkan.

- Pengumpulan Data: Menghimpun data ISPU dari sumber resmi untuk menjamin keabsahan input model.
- Persiapan Data: Menyiapkan data mentah agar dapat digunakan dalam pelatihan model melalui pembersihan, transformasi, dan normalisasi.
- Pengembangan Model: Membangun struktur arsitektur LSTM dan mengatur parameter pelatihan.
- Evaluasi Model: Menguji performa model dengan berbagai metrik dan membandingkan hasil eksperimen dari beberapa konfigurasi.

B. Pengumpulan Data

Langkah pertama dalam membangun sistem prediksi adalah memperoleh data yang akurat dan relevan. Data yang berkualitas tinggi merupakan fondasi utama bagi performa model machine learning, termasuk LSTM. Dalam penelitian ini, data diperoleh dari Dinas Lingkungan Hidup Kota Surabaya yang menyediakan informasi ISPU (Indeks Standar Pencemar Udara) secara harian. Data tersebut memuat lima parameter polusi udara utama, yaitu PM10, SO2, CO, O3, dan NO2. Dataset mencakup rentang waktu dari tahun 2022 hingga 2024 dan dikumpulkan dalam bentuk tabel harian. Karena model yang digunakan bersifat time-series mingguan, maka data harian diubah menjadi data mingguan menggunakan metode resampling. Resampling ini dilakukan untuk menyederhanakan pola data dan memudahkan model dalam menangkap tren musiman yang muncul secara periodik.

Data yang digunakan dalam penelitian ini bersumber dari Dinas Lingkungan Hidup Kota Surabaya. Dataset terdiri dari data harian Indeks Standar Pencemar Udara (ISPU) dengan lima parameter utama: PM10, SO2, CO, O3, dan NO2, dalam rentang waktu tahun 2022 hingga 2024.

```
df_weekly = df.resample('W').mean()
```

C. Persiapan Data

Sebelum memulai proses persiapan data, terlebih dahulu dilakukan impor berbagai library yang dibutuhkan dalam proses preprocessing dan pemodelan:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScal
er
from scipy import stats
```

Langkah pertama adalah pembersihan dan normalisasi dataset harian agar siap digunakan LSTM

```
Hapus baris kosong pada seluruh kolom

clean_df = raw_df.dropna(how='all')
imputasi rata-rata per parameter
clean_df = clean_df.fillna(clean_df.mean())
deteksi & hapus outlier (Z-score)

z = np.abs(stats.zscore(clean_df))
clean_df = clean_df[(z < 3).all(axis=1)]
normalisasi ke rentang 0-1

scaler = MinMaxScaler()
scaled = scaler.fit_transform(clean_df)
```

Setelah data bersih, bentuk struktur *time-series* 21 minggu agar cocok sebagai input LSTM:

```
def create_sequences(dataset, look_back=21):
    X, y = [], []
    for i in range(len(dataset) - look_back):
        X.append(dataset[i:(i + look_back), :])
        y.append(dataset[i + look_back, :])  # multioutput
    return np.array(X), np.array(y)

look_back = 21
X_raw, y_raw = create_sequences(data, look_back)
```

Tahapan ini mencakup beberapa proses penting untuk memastikan data siap digunakan dalam pelatihan model. Berikut isi dari proses persiapan data :

- Data Cleaning dengan menghapus baris yang memiliki nilai kosong dan (-) di semua kolom parameter dengan mengganti dengan NaN.
- Missing Values (Imputasi nilai hilang) menggunakan nilai rata-rata per parameter untuk menjaga kontinuitas data.
- Deteksi dan penghapusan outlier menggunakan metode Z-score agar model tidak bias terhadap nilai ekstrem.
- Normalisasi data menggunakan MinMaxScaler agar seluruh fitur berada dalam skala yang seragam.
- Pembentukan struktur time series dengan menggunakan 21 minggu sebagai window input model.

Setelah dilakukan proses cleaning dan filtering, jumlah data berubah sebagai berikut :

```
Jumlah data awal: 1096
Jumlah data setelah bersih (Z-score): 1078
```

Gambar 9. Jumlah data awal dan data bersih

Dataset dibagi menjadi :

- Train dataset 845 data sequence
- Validation dataset : 106 data sequence
- Test dataset : 106 data sequence

D. Pembangunan Model

Setelah data siap digunakan, proses selanjutnya adalah membangun dan mengimplementasikan arsitektur model Long Short-Term Memory (LSTM). Model ini dipilih karena kemampuannya dalam mempelajari urutan data time series dan mempertahankan informasi jangka panjang. Potongan kode berikut menunjukkan arsitektur final model LSTM multi-output yang digunakan:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
```

```
def build_model(optimizer=None, input_shape=(21, 5), output_dim=5):
    model = Sequential()
    model.add(LSTM(64, return_sequences=True, input_shape=input_shape))
    model.add(LSTM(32))
    model.add(Dense(output_dim))
    if optimizer:
        model.compile(optimizer=optimizer, loss='mse')
    else:
        model.compile(loss='mse')
    return model
```

Struktur arsitektur ini dirancang untuk memproses data sel ama 21 minggu sebelumnya dan menghasilkan prediksi li ma parameter sekaligus. Model dikembangkan dengan pe ndekatan deep learning menggunakan arsitektur *Long Sho rt-Term Memory* (LSTM) multi-output untuk memprediks i lima parameter polusi udara secara simultan. Struktur ars itektur model adalah sebagai berikut:

- **Input Layer**: 21 minggu data time series dengan 5 fitur
- **LSTM (64 unit)**, return_sequences=True
- **LSTM (32 unit)**
- **Dense Layer (output_dim)**

Model dilatih menggunakan *callback EarlyStopping* dan *ReduceLROnPlateau* untuk menghindari *overfitting*.

Setelah seluruh tahapan pelatihan selesai dan model menunjukkan performa yang optimal berdasarkan hasil evaluasi pada data validasi, model disimpan dalam format (.h5) agar dapat dimuat kembali saat implementasi sistem. Proses penyimpanan ini memungkinkan integrasi model ke dalam aplikasi berbasis Streamlit tanpa perlu melakukan pelatihan ulang.

Berikut adalah potongan kode untuk menyimpan model akhir:

```
model.save("model_lstm.h5")

joblib.dump(scaler_y, "scaler_y.pkl")

np.save("last_input.npy", X_test[-1:])
```

E. Evaluasi Model

Tahap terakhir dalam pengembangan model adalah melakukan evaluasi untuk mengukur kinerja model dalam melakukan prediksi. Evaluasi dilakukan dalam dua bentuk utama, yaitu eksperimen terhadap beberapa jenis optimizer dan pengukuran performa akhir model dengan metrik MAPE. Melalui evaluasi ini, diperoleh pemahaman mendalam tentang seberapa akurat model dalam memprediksi parameter polusi udara dan konfigurasi mana yang paling optimal.

1. Eksperimen Optimizer

Pengujian optimizer—Tanpa Optimizer, Adam, dan Nadam—dilakukan dengan loop otomatis.

berikut agar setiap konfigurasi dilatih dan dievaluasi secara konsisten:

```
from tensorflow.keras.optimizers import Adam, Nadam,
from tensorflow.keras.callbacks import EarlyStopping, Reduce LROnPlateau
import time, pandas as pd
```

```
optimizers = {
    "Tanpa Optimizer": None,
    "Adam": Adam(learning_rate=0.001),
```

```
    "Nadam": Nadam(learning_rate=0.001)

}

results = {}

loss_histories = {}

for name, opt in optimizers.items():

    print(f"\n🚀 Training with {name}...")

    if opt:

        model = build_model(opt,
input_shape=(X_train.shape[1],
X_train.shape[2]), output_dim=y_train.shape[1])

    else:

        model =
build_model(input_shape=(X_train.shape[1],
X_train.shape[2]), output_dim=y_train.shape[1])

    start = time.time() # Mulai timer

    history = model.fit(

        X_train, y_train,

        validation_data=(X_val, y_val),

        epochs=150,

        batch_size=64,

        verbose=0,

        callbacks=[

            EarlyStopping(patience=15,
restore_best_weights=True),

            ReduceLROnPlateau(patience=7,
factor=0.5)

        ]

    )

    duration = round((time.time() - start) /
60,2) # Durasi menit
```

Pada tahap ini, dilakukan pengujian tanpa optimizer dan 2 jenis optimizer yaitu Adam dan Nadam. Tujuannya adalah untuk mengetahui sejauh mana masing-masing *optimizer* mampu meningkatkan performa model dalam hal konvergensi, kestabilan pelatihan, serta akurasi hasil prediksi. Setiap model dilatih menggunakan struktur arsitektur LSTM yang sama dan diuji pada dataset yang telah dibagi menjadi data latih, validasi, dan uji.

Tabel 2. Ringkasan Hasil Evaluasi

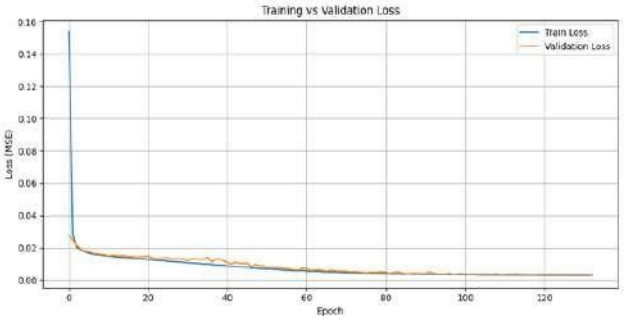
Optimizer	Train Loss	Val Loss	Test Loss	Time (min)	PM10	SO2	CO	O3	NO2
Tanpa Optimizer	0.0123	0.0129	0.0144	0.61	17.46	21.74	22.78	27.26	21.50
Adam	0.0039	0.0045	0.0051	1.05	10.44	14.69	17.34	15.62	11.25
Nadam	**0.0028**	**0.0034**	0.0044	1.19	**8.77**	**13.21**	**19.84**	**14.49**	**11.84**

Tabel berikut menunjukkan hasil evaluasi dari ketiga model berdasarkan nilai loss pada data pelatihan, validasi, dan pengujian, waktu pelatihan, serta nilai MAPE untuk masing-masing parameter kualitas udara. Pengujian dilakukan untuk membandingkan performa model dengan tiga jenis optimizer: tanpa optimizer, Adam, dan Nadam. Hasil menunjukkan bahwa Nadam memberikan performa terbaik dan paling seimbang.

Optimizer Nadam menghasilkan nilai *loss* paling rendah di data latih, validasi, dan uji, yaitu masing-masing 0.0028, 0.0034, dan 0.0044. Waktu pelatihan juga cukup singkat, yaitu 1,19 menit. Dari sisi akurasi, model dengan Nadam menghasilkan nilai MAPE terendah untuk PM10 (8,77%), dan secara umum memiliki hasil prediksi yang lebih stabil dibandingkan optimizer lain. Optimizer Adam juga cukup baik, namun memiliki nilai MAPE CO dan O_3 yang lebih tinggi dibanding Nadam. Sementara itu, model tanpa optimizer khusus menunjukkan performa terburuk, dengan *loss* dan MAPE yang paling tinggi. Berdasarkan hasil tersebut, dapat disimpulkan bahwa Nadam merupakan pilihan terbaik untuk pengembangan model prediksi kualitas udara dalam penelitian ini. Oleh karena itu, model akhir menggunakan optimizer Nadam sebagai konfigurasi utama.

2. **Evaluasi Akhir Model**

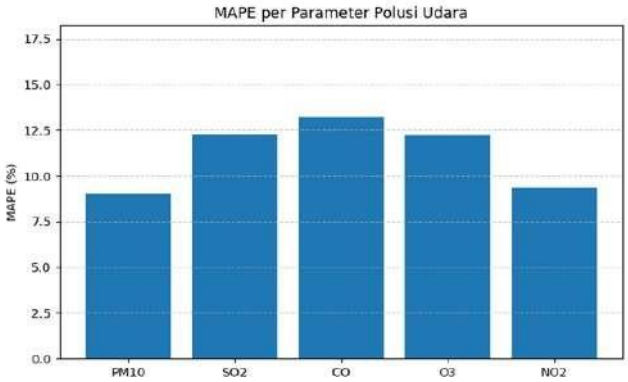
Evaluasi akhir dilakukan untuk menilai sejauh mana model LSTM yang telah dikembangkan mampu memberikan prediksi yang akurat terhadap lima parameter kualitas udara. Pengujian dilakukan menggunakan data uji yang tidak dilibatkan dalam proses pelatihan model, sehingga hasil evaluasi ini mencerminkan kemampuan generalisasi model secara objektif. Model terbaik diperoleh dengan menggunakan optimizer Nadam, yang menunjukkan performa konsisten pada data validasi dan uji. Proses pelatihan model ini juga diawasi dengan metode EarlyStopping dan ReduceLROnPlateau untuk menghindari overfitting dan memastikan konvergensi yang stabil.



Gambar 10. Kurva Training vs Validation Loss

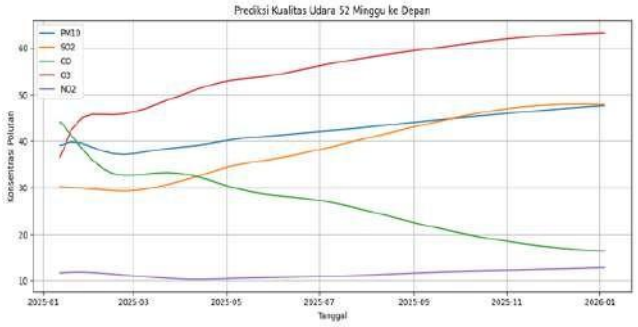
Berdasarkan Berdasarkan hasil evaluasi model, terlihat bahwa performa prediksi berada pada kategori baik, dengan seluruh nilai MAPE berada di bawah 15%. PM10 dan NO2 memiliki MAPE terendah, masing-masing sekitar 9%, menunjukkan tingkat akurasi yang sangat baik dalam memprediksi dua parameter tersebut. Sementara itu, CO mencatat MAPE tertinggi, namun tetap berada di bawah ambang batas 15%, yang masih dianggap cukup akurat dalam konteks prediksi data lingkungan yang bersifat fluktuatif.

Berikut adalah grafik nilai MAPE tiap parameter dari model akhir:



Gambar 11. Bar Chart MAPE per Parameter

Hasil ini menunjukkan bahwa proses pra-pemrosesan data yang mencakup resampling mingguan, penghapusan outlier, dan normalisasi terpisah berhasil meningkatkan kualitas input ke dalam model. Selain itu, arsitektur LSTM yang digunakan, beserta konfigurasi hyperparameter yang optimal seperti jumlah unit, batch size, dan penggunaan optimizer Nadam, turut berkontribusi terhadap peningkatan akurasi model. Secara keseluruhan, sistem prediksi berbasis LSTM ini terbukti mampu memberikan estimasi mingguan yang cukup akurat untuk seluruh parameter polusi udara, dan layak digunakan sebagai alat bantu dalam pengawasan kualitas udara ke depannya.



Gambar 12. Grafik Prediksi 52 Minggu ke Depan

F. Tahap Pengembangan Sistem

Tahap ini merupakan proses implementasi arsitektur model LSTM yang telah dirancang pada bagian sebelumnya. Fokus utama pada bagian ini adalah membangun dan melatih model LSTM dengan data yang telah diproses, serta mengevaluasi performa hasil prediksinya. Model ini dibangun dengan pendekatan multi-output, sehingga mampu memprediksi lima parameter polusi udara secara simultan dalam satu proses inferensi.

G. Pengujian Sistem

Pengujian sistem dilakukan menggunakan black-box testing, yaitu metode yang fokus pada keluaran sistem berdasarkan berbagai input tanpa melihat kode internal. Seluruh fitur diuji satu per satu dengan input valid dan tidak valid untuk memastikan sistem bekerja sesuai spesifikasi. Sistem diharapkan menghasilkan prediksi dan visualisasi yang tepat untuk input benar, serta menampilkan peringatan untuk input tidak sesuai.

Pengujian melibatkan pegawai Dinas Lingkungan Hidup Kota Surabaya untuk memperoleh masukan dari pengguna berpengalaman, terutama pada fitur pemilihan parameter polusi, prediksi mingguan, visualisasi, dan ekspor data. Metode ini menilai fungsi teknis sekaligus kenyamanan penggunaan bagi pengguna non-teknis. Hasil pengujian menunjukkan sistem berjalan stabil, fitur utama berfungsi dengan baik, dan informasi yang dihasilkan mudah dipahami. Setiap hasil dicatat dan dianalisis sebagai dasar validasi sebelum sistem siap digunakan secara luas.

H. Hasil Prediksi

Hasil sistem prediksi kualitas udara menggunakan model **LSTM** dievaluasi secara visual dan kuantitatif menggunakan **MAPE**. Secara keseluruhan, model menunjukkan performa baik dengan MAPE antara 9%–13%. Parameter **PM10** memiliki akurasi tertinggi (MAPE 9,00%), diikuti **NO2** (9,30%), yang keduanya mampu menangkap tren dan pola musiman secara konsisten. Parameter **SO2** dan **O3** memiliki MAPE 12,30%, cukup akurat meski kurang sensitif terhadap perubahan tajam. Parameter **CO** paling menantang dengan MAPE 13,20%, karena model kesulitan memprediksi lonjakan mendadak akibat faktor eksternal seperti lalu lintas atau cuaca.

Secara keseluruhan, model LSTM efektif mempelajari pola historis dan musiman data mingguan, andal untuk parameter yang cenderung teratur, dan cukup baik untuk digunakan dalam sistem prediksi ISPU. Hasil visualisasi

grafik prediksi memperlihatkan bahwa model mampu mengikuti tren data nyata, menangkap arah perubahan, serta memberikan estimasi mingguan yang dapat diandalkan untuk pengambilan keputusan. Meskipun ada beberapa perbedaan pada titik ekstrem dan lonjakan mendadak, sistem tetap mampu memberikan informasi yang berguna bagi masyarakat dan pihak berwenang, khususnya dalam merencanakan langkah mitigasi kualitas udara. Peningkatan lebih lanjut dapat dilakukan dengan memasukkan variabel eksternal seperti kondisi cuaca atau intensitas lalu lintas untuk meningkatkan akurasi pada parameter yang fluktuatif.

V. KESIMPULAN DAN SARAN

Berdasarkan penelitian mengenai sistem prediksi kualitas udara menggunakan metode Long Short-Term Memory (LSTM) dan implementasinya dalam aplikasi berbasis Streamlit, dapat disimpulkan bahwa model LSTM mampu memprediksi lima parameter utama kualitas udara (PM10, SO_2, CO, O_3, dan NO_2) secara mingguan dengan akurasi yang cukup baik berdasarkan data ISPU Kota Surabaya selama 21 minggu. Optimasi menggunakan Nadam terbukti meningkatkan kinerja model, dengan MAPE terendah pada PM10 sebesar 15,25% dan tertinggi pada CO sebesar 21,97%, menunjukkan performa yang masih dalam batas yang dapat diterima. Selain itu, aplikasi Streamlit berhasil mengintegrasikan model LSTM, menampilkan antarmuka interaktif, visualisasi prediksi, narasi otomatis, serta fitur unduhan data dalam format CSV dan PNG, sehingga siap digunakan sebagai media informasi kualitas udara mingguan.

Beberapa saran dari penulis untuk pengembangan selanjutnya antara lain:

1. Menambah cakupan data dari wilayah lain untuk memperluas generalisasi model.
2. Mengintegrasikan faktor eksternal seperti cuaca, suhu, kelembaban, dan lalu lintas.
3. Mengeksplorasi arsitektur model yang lebih maju seperti LSTM-attention, encoder-decoder, atau model hybrid.
4. Meningkatkan fitur aplikasi, termasuk sistem peringatan dini, rekomendasi mitigasi polusi, dan integrasi data sensor real-time.
5. Menyediakan dokumentasi teknis dan API terbuka untuk memudahkan pengembangan lanjutan.

REFERENSI

[1] Agarwal, A. (2025). *Overview of All Optimizers: Making Bad Choices Faster Since 1986*.

[2] Akkem, Y., Kumar, B. S., & Varanasi, A. (2023). Streamlit Application for Advanced Ensemble Learning Methods in Crop Recommendation Systems – A Review and Implementation. *Indian Journal Of Science And Technology*, *16*(48), 4688–4702. https://doi.org/10.17485/ijst/v16i48.2850

[3] Arkadia, A., Hananto, B., & Prasvita, D. S. (2022). Optimasi Long Short Term Memory Dengan Adam Menggunakan Data Udara Kota DKI Jakarta. *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (SENAMIKA)*, 92–101.

[4] B. Rollins Jhon. (2015). Foundational Methodology for Data Science. *IBM Analytics*, 1–4.

[5] Butwall, M., Ranka, P., & Shah, S. (2019). Python in Field of Data Science: A Review. *International Journal of Computer Applications*. https://doi.org/https://doi.org/10.5120/ijca2019919404

[6] Karyadi, Y. (2022). Prediksi Kualitas Udara Dengan Metoda LSTM, Bidirectional LSTM, dan GRU. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *9*(1), 671–684. https://doi.org/10.35957/jatisi.v9i1.1588

[7] Keerthi, M., Reddy, G., Raghava, V., & Reddy, K. (2023). Streamlit Interface for Multiple Disease Diagnosis. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/https://doi.org/10.22214/ijraset.2023.49166

[8] Khumaidi, A., Raafi, R., Permana Solihin, I., & Rs Fatmawati, J. (2020). Pengujian Algoritma Long Short Term Memory untuk Prediksi Kualitas Udara dan Suhu Kota Bandung. *Jurnal Telematika*, *15*(1), 13–18.

[9] Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. *ArXiv Preprint ArXiv:1412.6980*. https://arxiv.org/abs/1412.6980

[10] Mengara Mengara, A. G., Park, E., Jang, J., & Yoo, Y. (2022). Attention-Based Distributed Deep Learning Model for Air Quality Forecasting. *Sustainability (Switzerland)*, *14*(6). https://doi.org/10.3390/su14063269

[11] Muslim, B., & Prabowo, K. (2018). *Penyehatan Udara*. Pusat Pendidikan Sumber Daya Kesehatan, Kementerian Kesehatan Republik Indonesia.

[12] Naresh, G., & Indira, B. (2023). Air pollution prediction using multivariate LSTM deep learning model. *International Journal of Innovative Science and Advanced Engineering (IJISAE)*, *5*(4), 19–24. https://www.ijisae.org/index.php/IJISAE/article/view/4111

[13] Olah, C. (2015). *Understanding LSTM Networks*.

[14] Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach (7th ed.)*. McGraw-Hill.

[15] 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). https://doi.org/10.1371/journal.pone.0227222

[16] Sole, A. (2019). *Introducing Visual Studio Code*. Visual Studio Code. https://doi.org/https://doi.org/10.1007/978-1-4842-4224-7_1

[17] Sommerville, I. (2011). *Software Engineering (9th ed.)*. Addison-Wesley.
[18] Song, X., Liu, Y., Xue, L., Wang, J., Zhang, J., Wang, J., Jiang, L., & Cheng, Z. (2020). Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model. *Journal of Petroleum Science and Engineering*, *186*(November 2019), 106682. https://doi.org/10.1016/j.petrol.2019.106682
[19] Tock, K. (2019). *Google CoLaboratory as a Platform for Python Coding with Students*. https://doi.org/https://doi.org/10.32374/rtsre.2019.013
[20] Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). LSTM network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), 68–75.