PROSIDING DISKUSI PANEL NASIONAL PENDIDIKAN MATEMATIKA

Original Research

Penggunaan Metode Long Short Term Memory untuk Peramalan Kecepatan Angin di PLTB Jeneponto

Fitri Ademia Rachma[1], Mutia Nur Estri[2]*), & Ari Wardayani[3]
Universitas Jenderal Soedirman

INFO ARTICLES

Key Words:

kecepatan angin; *Long Short-Term Memory*; peramalan

Abstract**:** *The Jeneponto Wind Power Plant (PLTB) often faces obstacles because the turbine is turned on when the wind speed is insufficient. This can result in energy waste and operational losses. This study aims to predict wind speed using the Long Short-Term Memory (LSTM) method with wind speed data for the period 2000–2024. The LSTM model was built with four divisions of training data and testing data (60:40, 70:30, 80:20, and 90:10). The evaluation was carried out with the RMSE value, where the lowest RMSE for u100 was 0.0541 at 90:10 division and v100 was 0.0567 at 70:30. The best models were each used to predict wind speed for the next 120 days. The forecasting results showed that wind conditions during the prediction period were within safe operational limits for the turbine.*

Abstrak: Pembangkit Listrik Tenaga Bayu (PLTB) Jeneponto kerap menghadapi kendala karena turbin dinyalakan saat kecepatan angin tidak mencukupi. Hal ini dapat mengakibatkan pemborosan energi dan kerugian operasional. Penelitian ini bertujuan meramalkan kecepatan angin menggunakan metode Long Short-Term Memory (LSTM) dengan data kecepatan angin periode 2000–2024. Model LSTM dibangun dengan empat pembagian data training dan data testing (60:40, 70:30, 80:20 dan 90:10). Evaluasi dilakukan dengan nilai RMSE, di mana RMSE terendah untuk u100 sebesar 0,0541 pada pembagian 90:10 dan v100 sebesar 0,0567 pada 70:30. Model terbaik masing-masing digunakan untuk memprediksi kecepatan angin 120 hari ke depan. Hasil peramalan menunjukkan bahwa kondisi angin selama rentang waktu prediksi berada dalam batas operasional yang aman bagi turbin.

Correspondence Address: Jln. Dr. Soeparno Utara Kampus Unsoed Grendeng, Purwokerto, Kode Pos: 53123; e-mail: mutia.estri@unsoed.ac.id

How to Cite (APA 6th Style): Rachma, F.A., Estri, M.N., & Wardayani, A. (2025). Penggunaan Metode *Long Short Term Memory* untuk Peramalan Kecepatan Angin di PLTB Jeneponto. *Prosiding Diskusi Panel Nasional Pendidikan Matematika*, 333-346.

PENDAHULUAN

Penggunaan energi di masa sekarang banyak bergantung pada energi fosil seperti minyak bumi, batu bara, gas alam, dan lain-lain. Penggunaan energi secara masal dan dalam jumlah yang banyak menyebabkan mulai terjadi kelangkaan pada energi fosil. Pada beberapa sektor, penggunaan energi fosil mulai digantikan dengan energi terbarukan. Salah satu contohnya adalah penggunaan energi angin pada Pembangkit Listrik Tenaga Bayu (PLTB) sebagai bentuk upaya pengalihan dari ketergantungan pada energi fosil terutama Batubara.

Angin adalah besaran vektor yang memiliki dua komponen utama, yaitu arah dan kecepatan. Kecepatan angin biasanya dinyatakan dalam satuan knot atau m/s, sedangkan arah angin menunjukkan dari mana angin tersebut bertiup, bukan ke mana arahnya. Pengamatan arah angin dapat dilakukan menggunakan empat mata angin utama, yaitu utara, selatan, barat, dan timur. Selain arah dan kecepatan, lokasi pengamatan juga menjadi bagian penting dalam pencatatan data angin. Titik pengamatan dicatat menggunakan koordinat geografis berupa lintang (latitude) dan bujur (longitude), di mana latitude menunjukkan posisi suatu tempat terhadap garis khatulistiwa dalam arah utara–selatan, sedangkan longitude menunjukkan posisinya terhadap garis bujur nol dalam arah timur–barat. Pencatatan lokasi ini berperan penting dalam pemetaan data cuaca dan iklim, serta dalam integrasi data ke dalam sistem model numerik dan analisis spasial lainnya (Miftahuddin dkk., 2020)

Pembangkit Listrik Tenaga Bayu menghasilkan Listrik dari turbin yang digerakkan oleh angin. Menurut(HARYANTI dkk., 2023) , diperlukan angin dengan kecepatan minimal 2 meter per sekon (2 m/s) untuk dapat menggerakkan turbin PLTB. Kecepatan angin yang tidak konstan dapat menimbulkan masalah bagi PLTB. Hal ini dikarenakan terdapat biaya yang dikeluarkan untuk menyalakan turbin, akan ada kerugian yang timbul apabila turbin dinyalakan saat kecepatan angin minimum tidak terpenuhi. Oleh karena itu, diperlukan peramalan kecepatan angin untuk meminimalisir kerugian yang timbul.

Indonesia memiliki beberapa PLTB dengan kapasitas yang tinggi, salah satunya PLTB Jeneponto yang berada di Kabupaten Jeneponto Provinsi Sulawesi Selatan, dengan kapasitas 72 Mega Watt (72 MW). Dengan kapasitas tersebut, PLTB Jeneponto dapat mengaliri Listrik pada minimalnya 200.000 rumah. Besarnya kapasitas PLTB ini, berbanding lurus dengan potensi kerugiannya apabila turbin dinyalakan saat kecepatan minimum tidak terpenuhi. Oleh karena itu, perlu dilakukan peramalan kecepatan angin pada PLTB Jeneponto.

Beberapa penelitian telah dilakukan untuk meramalkan kecepatan angin, diantaranya adalah penelitian yang dilakukan oleh Hakimi dkk (2017). Penelitian ini menggunakan data kecepatan angin pada bulan Januari 2015 yang diperoleh dari system *Automatic Weather Season* (AWS) pada Lentera Angin Nusantara di Tasikmalaya. Hasil peramalan diperoleh nilai *Mean Squared Error* (MSE) sebesar 1,0909. Nilai tersebut menunjukkan Tingkat kesalahan model dalam peramalan. Semakin kecil nilai MSE, mencerminkan hasil yang lebih akurat terhadap data aktual. Penelitian terkait peramalan kecepatan angin juga dilakukan oleh Yusuf (2020). Penelitian ini menggunakan *Reccurent Neural Network* (RNN) untuk menghitung peramalan kecepatan angin guna mengetahui besar data yang dibangkitkan pada PLTB. Penelitian ini menggunakan beberapa model perhitungan, yang masing-masing menghasilkan nilai *Mean Percentage Error* (MAPE) 20,02%, 23,31%, 18,15%, dan 12,58%. Model yang paling aktual adalah model dengan nilai MAPE paling rendah, yakni model keempat.

Meramalkan kecepatan angin dapat dilakukan dengan berbagai metode, salah satunya adalah LSTM. Metode LSTM memproses data secara berurutan dari waktu awal hingga akhir, dengan menyimpan informasi dalam lapisan tersembunyi. Setiap dimasukkan data baru, informasi yang telah disimpan akan diperbarui. Informasi yang masih relevan akan tetap dipertahankan, sedangkan informasi yang tidak lagi relevan akan dihapus melalui proses filterisasi. Mekanisme ini

memungkinkan pengurangan beban penyimpanan dalam lapisan tersembunyi, sehingga model dapat mengakomodasi lebih banyak data penting dan mengenali pola secara lebih efektif (Puteri, 2023).

Penelitian menggunakan metode LSTM dilakukan oleh Wiranda & Sadikin (2019).Metode LSTM digunakan untuk meramalkan penjualan produk "X" di PT. Metiska Farma. Lima eskperimen dilakukan untuk menentukan kombinasi terbaik antara Nilai Range Interval dan Komposisi Dataset. Parameter MAPE dan RMSE digunakan untuk evaluasi. Hasil terbaik diperoleh pada eksperimen kelima dengan interval [-1,1], komposisi 90% *data training* dan 10% *data testing*, serta MAPE 12%, menghasilkan prediksi penjualan sebesar Rp. 13.762.154.

Berdasarkan uraian di atas, pada penelitian ini akan dikaji mengenai Peramalan Kecepatan Angin di PLTB Jeneponto menggunakan Metode LSTM. Harapannya penelitian ini dapat digunakan PLTB Jeneponto dalam pengoperasian turbin angin sehingga dapat meminimalisir potensi kerugian.

METODE

Perhitungan pada penelitian ini dilakukan dengan bahasa pemrograman Phyton menggunakan Kaggle. Penelitian ini dilakukan dengan bahasa pemprograman Python menggunakan Google Colaboratory. Prosedur penelitian yang dilakukan yaitu pengumpulan data, *data preprocessing*, pembagian *data training* dan *data testing*, perhitungan peramalan dengan model LSTM, perhitungan evaluasi model, dan perhitungan peramalan data.

Pengumpulan data

Data yang digunakan dalam penelitian ini adalah data kecepatan angin setiap 6 jam, dari 1 Januari 2000 sampai 31 Desember 2024 di PLTB Jeneponto, secara keseluruhan terdapat 36.259 data. Data diperoleh dari data set ***ERA5 hourly data on single levels from 1940 to present*** pada laman https://cds.climate.copernicus.eu/. Data diunduh dalam format *GRIdded Binary* (GRIB), yang dikonversi ke dalam format excel.

Data Preprocesing

Data preprocessing adalah tahap awal dalam mempersiapkan data sebelum perhitungan dengan model LSTM. Prosedur ini memiliki beberapa tahapan yang dapat berbeda pada setiap penelitian. Pada penelitian ini, digunakan pembersihan data, normalisasi, dan *sliding window* pada data.

a. Pembersihan data

   Data yang digunakan pada penelitian tidak selalu berada dalam kondisi ideal. Pembersihandata merupakan proses yang dilakukan untuk menyeleksi, memperbaiki, atau menghapus data yang berpotensi menurunkan akurasi prediksi. Proses ini dilakukan dengan mengidentifikasi keberadaan nilai yang tidak tercatat (*missing values*) menggunakan fungsi *isnull*, serta nilai yang tercatat secara ganda (*duplicated values*) menggunakan fungsi *duplicated* dalam bahasa pemrograman Phyton. Jika terdapat kedua nilai tersebut, dapat dilakukan penanganan dengan penghapusan atau penggantian nilai (Muhammad & Nurhaida, 2025).

b. Normalisasi

   Data dengan rentang nilai yang luas dapat memengaruhi kinerja model. Untuk mengatasi hal tersebut, normalisasi diterapkan guna menyederhanakan proses komputasi dan meningkatkan akurasi prediksi. Menurut Putra dkk (2024)normalisasi data adalah proses mengubah skala data ke dalam rentang tertentu agar lebih sesuai dengan kebutuhan pemodelan. Untuk melakukan normalisasi, diperlukan informasi mengenai nilai maksimum dan minimum dari data. Nilai-nilai ini digunakan agar skala hasil normalisasi berada dalam rentang yang diharapkan. Apabila nilai data melebihi batas maksimum atau lebih kecil dari nilai minimum, hasil normalisasi dapat berada di luar rentang yang ditentukan. Rentang nilai hasil normalisasi dapat disesuaikan dengan kebutuhan serta karakteristik model yang digunakan. Normalisasi data dengan rentang 0,1-0,9 dapat dihitung dengan persamaan berikut:

$$a = \frac{0{,}8(b - \min(b))}{\max(b) - \min(b)} + 0{,}1 \qquad 2.1$$

c. *Sliding window*

Menurut Rahmawati dkk (2023)*sliding window* adalah proses pembentukan struktur dari data runtun waktu untuk mengurangi *error* dari aproksimasi. *Sliding window* memotong data yang Panjang menjadi beberapa kelompok yang lebih kecil seperti yang ada pada Gambar 1.

(a) (b)

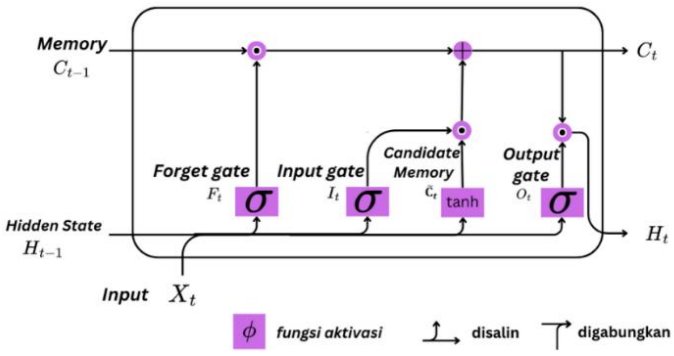
Gambar 1 Proses *sliding window*

Pembagian *data training* dan *data testing*

Data training adalah data yang digunakan untuk melatih model. *Data testing* adalah data yang digunakan untuk mengetahui akurasi model. Pada penelitian ini digunakan empat ukuran pembagian data, meliputi 60% data training dengan 40% data testing, 70% data training dengan 30% data testing, 80 % data training dengan 20% data testing, dan 90 % data training dengan 10% data testing .

Long Short-Term Memory

Penjelasan pada bagian ini, diambil dari Zhang dkk (2021).*Long Short-Term Memory* merupakan bentuk pembaruan dari metode RNN. Pada metode RNN diproses secara berurutan dari waktu awal hingga akhir. Setiap data baru akan memperbarui informasi sebelumnya, sehingga model dapat mengenali pola dalam urutan data. Namun, jika pembaruan ini berlangsung terus-menerus tanpa pengendalian, informasi dari data awal dapat terlupakan karena tergantikan oleh data terbaru. LSTM mengatasi masalah ini dengan hanya menyimpan data yang relevan dan menghapus data yang tidak lagi relevan, sehingga informasi penting pada data awal tetap tersimpan. LSTM tersusun atas beberapa struktur seperti yang ada pada Gambar 2, yang meliputi *Gate* LSTM, *candidate memory cell*, *memory cell*, dan *hidden state*.



Gambar 2. Struktur *cell* LSTM

a. *Gate* LSTM

Lapisan LSTM memiliki dua nilai yang dimasukkan ke dalam setiap lapisannya, yaitu *Input* (X_t) dan *hidden state* (H_{t-1}) dari lapisan sebelumnya. Kedua nilai tersebut digabungkan kemudian disalin ke tiga gate LSTM, meliputi *forget gate, input gate*, dan *output gate*. Setiap *gate* memiliki fungsi dan perhitungan yang berbeda. *Forget gate* merupakan *gate* yang digunakan untuk menentukan informasi mana yang akan diingat dan dilupakan oleh jaringan. Perhitungannya dilakukan menggunakan persamaan berikut:

$$\boldsymbol{F_t} = \sigma(\boldsymbol{X_t W_{xf}} + \boldsymbol{H_{t-1} W_{hf}} + \boldsymbol{b_f}) \quad (2.12)$$

dengan

$\boldsymbol{F_t} \in \mathbb{R}^{n \times p}$: matriks *forget gate* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* adalah jumlah sampel dan *p* adalah jumlah neuron pada *forget gate*;

σ : fungsi aktivasi sigmoid;

$\boldsymbol{X_t} \in \mathbb{R}^{n \times d}$: matriks *input* berukuran $n \times d$ pada waktu ke-*t*, dengan *n* adalah jumlah sampel dan *d* adalah jumlah fitur pada data;

$\boldsymbol{W_{xf}} \in \mathbb{R}^{d \times p}$: matriks bobot berukuran $d \times p$ yang menghubungkan *neuron input* (x) ke *forget gate* (f), dengan *d* adalah jumlah fitur data dan *p* adalah jumlah neuron;

$\boldsymbol{H_{t-1}} \in \mathbb{R}^{n \times p}$: matriks *hidden state* berukuran $n \times p$ pada waktu ke-*t*-1, dengan n adalah jumlah sampel dan *p* adalah jumlah neuron;

$\boldsymbol{W_{hf}} \in \mathbb{R}^{p \times p}$: matriks bobot berukuran $p \times p$ yang menghubungkan *hidden state* (*h*) ke *forget gate* (*f*), *p* adalah jumlah neuron;

$\boldsymbol{b_f} \in \mathbb{R}^{n \times p}$: matriks bias berukuran $n \times p$ pada *forget gate* (*f*), dengan *n* adalah jumlah sampel dan *p* adalah jumlah neuron.

Input gate merupakan *gate* yang digunakan untuk menentukan bagian mana dari *input* dan *hidden state* yang akan dimasukkan ke dalam *memory cell*. Perhitungannya dilakukan dengan persamaan berikut:

$$\boldsymbol{I_t} = \sigma(\boldsymbol{X_t W_{xi}} + \boldsymbol{H_{t-1} W_{hi}} + \boldsymbol{b_i}) \quad (2.13)$$

dengan

$\boldsymbol{I_t} \in \mathbb{R}^{n \times p}$: matriks *input gate* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* adalah jumlah neuron*;*

σ : fungsi aktivasi sigmoid;

$\boldsymbol{X_t} \in \mathbb{R}^{n \times d}$: matriks *input* berukuran $n \times d$ pada waktu ke-*t*, dengan *n* adalah jumlah sampel dan *d* adalah jumlah fitur pada data sampel;

$\boldsymbol{W_{xi}} \in \mathbb{R}^{d \times p}$: matriks bobot berukuran $d \times p$ yang menghubungkan *neuron input (x)* ke *input gate* (*i*), dengan *d* adalah jumlah fitur data dan *p* adalah jumlah neuron;

$\boldsymbol{H_{t-1}} \in \mathbb{R}^{n \times p}$: matriks *hidden state* berukuran $n \times p$ pada waktu ke-*t*-1, dengan *n* jumlah sampel dan *p* adalah jumlah neuron;

$\boldsymbol{W_{hi}} \in \mathbb{R}^{p \times p}$: matriks bobot berukuran $p \times p$ yang menghubungkan *neuron hidden state* (*h*) ke *input gate* (*i*), dengan *p* adalah jumlah neuron;

$\boldsymbol{b_i} \in \mathbb{R}^{1 \times p}$: vektor bias berukuran $1 \times p$ pada *input gate* (*i*), dengan *p* adalah jumlah neuron.

Output gate, berfungsi mengatur seberapa banyak informasi yang akan dikeluarkan dari *cell* sebagai *output.* Menggunakan persamaan berikut:

$$\boldsymbol{O_t} = \sigma(\boldsymbol{X_t W_{xo}} + \boldsymbol{H_{t-1} W_{ho}} + \boldsymbol{b_o}) \quad (2.14)$$

dengan

$\boldsymbol{O_t} \in \mathbb{R}^{n \times p}$: matriks *output gate* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* adalah jumlah neuron;

σ : fungsi aktivasi sigmoid;

$\boldsymbol{X_t} \in \mathbb{R}^{n \times d}$: matriks *input* berukuran $n \times d$ pada waktu ke-*t*, dengan *n* adalah jumlah sampel dan *d* adalah jumlah fitur pada data sampel ;

$\boldsymbol{W_{xo}} \in \mathbb{R}^{d \times p}$: matriks bobot berukuran $d \times p$ yang menghubungkan *neuron input* (*x*) ke *neuron output gate* (*o*), dengan *d* adalah jumlah fitur data dan *p* adalah jumlah neuron;

$\boldsymbol{H_{t-1}} \in \mathbb{R}^{n \times p}$: matriks *hidden state* berukuran $n \times p$ pada waktu ke- *t*-1, dengan *n* jumlah sampel dan *p* adalah jumlah neuron;

$\boldsymbol{W_{ho}} \in \mathbb{R}^{p \times p}$: matriks bobot berukuran $p \times p$ yang menghubungkan *neuron hidden state* (*h*) ke *output gate* (*o*), dengan *p* adalah jumlah neuron;

$\boldsymbol{b_o} \in \mathbb{R}^{1 \times p}$: vektor bias berukuran $1 \times p$ pada *output gate*, dengan *p* adalah jumlah

neuron.

b. *Candidate memory cell*
Candidate memory cell merupakan bagian yang berfungsi untuk mempertimbangkan informasi mana saja yang akan disimpan di *memory cell*. Perhitungannya dilakukan dengan persamaan berikut:

$$\tilde{\mathbf{C}}_t = \tanh(\boldsymbol{X_t W_{xc}} + \boldsymbol{H_{t-1} W_{hc}} + \boldsymbol{b_C}) \quad (2.15)$$

dengan

$\tilde{\mathbf{C}}_t \in \mathbb{R}^{n\times p}$: matriks *candidate memory* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* adalah jumlah neuron;
tanh : fungsi aktivasi tangen hiperbolik;
$\boldsymbol{X}_t \in \mathbb{R}^{n\times d}$: matriks *input* berukuran $n \times d$ pada *t*, dengan *n* adalah jumlah sampel dan *d* adalah jumlah fitur pada data;
$\boldsymbol{W}_{xc} \in \mathbb{R}^{d\times p}$: matriks bobot berukuran $d \times p$ yang menghubungkan *neuron input* (*x*) ke *candidate memory* (*c*), dengan *d* adalah jumlah fitur data dan *p* adalah jumlah neuron;
$\boldsymbol{H}_{t-1} \in \mathbb{R}^{n\times p}$: matriks *hidden state* berukuran $n \times p$ pada waktu ke-*t*-1, dengan *n* jumlah sampel dan *p* adalah jumlah neuron;
$\boldsymbol{W}_{hc} \in \mathbb{R}^{p\times p}$: matriks bobot berukuran $p \times p$ yang menghubungkan *neuron hidden state* (*h*) ke *neuron candidate memory* (*c*), dengan *p* adalah jumlah neuron;
$\boldsymbol{b}_c \in \mathbb{R}^{1\times p}$: vektor bias berukuran $1\times p$ pada *candidate memory*, dengan *p* adalah jumlah neuron.

c. *Memory cell*
Memory cell merupakan tempat penyimpanan pada jaringan LSTM. Informasi lama yang tidak dilupakan oleh *forget gate* dan informasi baru yang ditambahkan oleh *input gate* dimasukkan ke dalam *memory cell*. Perhitungan pada *memory cell* dilakukan dengan operasi *Hadamard product* yang definisinya tertuang pada definisi berikut :

Definisi 2.1 *Hadamard product*

Operasi Hadamard product dinotasikan dengan $\odot$ *dari matriks* $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times p}$ *dengan* $\boldsymbol{A} = [a_{\{ij)}]$ *dan* $\boldsymbol{B} = [b_{\{ij)}]$ *adalah* $(\mathbf{A} \odot \mathbf{B})_{ij} = [a_{ij}\, b_{ij}]$ *untuk setiap* $i = 1, 2, \dots, n$ *dan* $j = 1, 2, \dots, p$.

Perhitungan pada *memory cell* dilakukan dengan persamaan berikut :

$$\boldsymbol{C_t} = \boldsymbol{F_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{I_t} \odot \tilde{\boldsymbol{C}}_t \quad (2.16)$$

dengan

$\boldsymbol{C}_t \in \mathbb{R}^{n\times p}$: matriks *memory cell* berukuran $n \times p$ pada waktu ke-*t*, dengan n jumlah sampel dan p jumlah neuron;
$\odot$: operasi *Hadamard product*
$\boldsymbol{F}_t \in \mathbb{R}^{n\times p}$: matriks *forget gate* berukuran $n \times p$ pada waktu ke-*t*, dengan n jumlah sampel dan p jumlah neuron;
$\boldsymbol{C}_{t-1} \in \mathbb{R}^{n\times p}$: matriks *memory cell* berukuran $n \times p$ pada waktu ke-*t*-1, dengan n jumlah sampel dan p jumlah neuron;
$\boldsymbol{I}_t \in \mathbb{R}^{n\times p}$: matriks *input gate* berukuran $n \times p$ pada waktu ke-*t*, dengan dengan n jumlah sampel dan *p* jumlah neuron;
$\tilde{\boldsymbol{C}}_t \in \mathbb{R}^{n\times p}$: matriks *candidate memory* berukuran $n \times p$ pada waktu ke-*t*, dengan n jumlah sampel dan *p* adalah jumlah neuron.

d. *Hidden state*

Hidden state merupakan tempat informasi yang akan digunakan sebagai *input* pada lapisan selanjutnya. Informasi yang tersimpan di memory cell disalin menjadi dua bagian, satu bagian diteruskan ke tahap selanjutnya dan bagian yang lain dimasukkan ke dalam output gate untuk dipertimbangkan akan disimpan atau diabaikan. Apabila nilai output gate mendekati 1, semua informasi dalam memory cell akan dikeluarkan sebagai output pada lapisan tersebut dalam bentuk hidden state. Sebaliknya, apabila nilai output gate mendekati 0 maka informasi tersebut hanya disimpan dalam memory cell untuk perhitungan berikutnya dan tidak dikeluarkan sebagai output. Perhitungan hidden state dilakukan menggunakan persamaan berikut:

$$\boldsymbol{H_t} = \boldsymbol{O_t} \odot \tanh(\boldsymbol{C_t}) \tag{2.17}$$

dengan

$\boldsymbol{H_t} \in \mathbb{R}^{n \times p}$: matriks *hidden state* berukuran $n \times h$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* jumlah neuron;

$\boldsymbol{O_t} \in \mathbb{R}^{n \times p}$: matriks *output gate* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* jumlah neuron;

$\boldsymbol{C_t} \in \mathbb{R}^{n \times p}$: matriks *memory cell* berukuran $n \times p$ pada waktu ke-*t*, dengan *n* jumlah sampel dan *p* jumlah neuron.

2.1. Perhitungan evaluasi model

Evaluasi peramalan dengan LSTM pada penelitian ini dilakukan dengan *Root Mean Square Error (*RMSE). Menurut Indra Sanjaya & Heksaputra (2020), RMSE adalah metode evaluasi yang digunakan untuk mengukur tingkat akurasi suatu model prediksi dengan menghitung rata-rata kuadrat dari selisih antara nilai aktual dan nilai prediksi. Pada data dengan selisih prediksi yang besar terhadap nilai aktual, RMSE akan menghasilkan nilai error yang tinggi. Semakin kecil nilai RMSE, maka semakin baik akurasi model tersebut.

Nilai RMSE dinyatakan dalam skala dan satuan yang sama dengan data yang digunakan. Nilai RMSE dihitung menggunakan persamaan berikut:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{2.1}$$

dengan

n : banyaknya data;

y_i : nilai aktual data ke-*i*;

$\tilde{y}_i$: nilai prediksi data ke-*i*.

2.2. Peramalan dan Denormalisasi Data

Peramalan data dilakukan menggunakan model dengan tingkat akurasi tertinggi, ditandai oleh nilai RMSE yang paling rendah. Hasil peramalan yang dihasilkan masih dalam bentuk data yang telah dinormalisasi, sehingga perlu dikonversi kembali ke skala aslinya melalui proses denormalisasi dengan menggunakan persamaan berikut:

$$b = \frac{(a - 0{,}1)(\max(b) - \min(b))}{0{,}8} + \min(b) \tag{2.4}$$

HASIL

Pengumpulan data

Contoh hasil pengumpulan data terdapat pada Tabel 1.

Tabel 1 Contoh hasil pengumpulan data

No	*Date*	*U*100	*V*100
1	2000-01-01 00:00:00	3,0848	3,0325
2	2000-01-01 06:00:00	2,5884	3,4509
3	2000-01-01 12:00:00	2,6693	3,4249
4	2000-01-01 18:00:00	3,0587	4,3738
5	2000-01-02 00:00:00	3,2594	3,7474
6	2000-01-02 06:00:00	2,3952	3,3906
7	2000-01-02 12:00:00	1,1871	3,8010
8	2000-01-02 18:00:00	1,2199	2,8360
9	2000-01-03 00:00:00	2,1600	1,3102
10	2000-01-03 06:00:00	1,0760	0,3000

Data Preprocessing

Data preprocessing terbagi atas tiga tahapan, meliputi pembersihan data, normalisasi data, dan *sliding window*. Gambar 2 adalah hasil pembersihan data.

```
#Pembersihan Data
#Memeriksa data kosong
Df.isnull().sum()
#Memeriksa data duplikat
Df.duplicated().sum()
```

```
np.int64(0)
```

Gambar 3. Hasil pembersihan data

Hasil normalisasi data dalam rentang 0,1-0,9 terdapat pada Tabel 2

Tabel 2 Contoh hasil normalisasi data

No	Sebelum Normalisasi		Setelah Normalisasi	
	*U*100	*V* 100	*U*100	*V* 100
0	3,085	3,032	0,316	0,331
1	2,588	3,451	0,280	0,363
2	2,669	3,425	0,286	0,361
3	3,059	4,374	0,314	0,434
4	3,259	3,747	0,328	0,386
5	2,395	3,391	0,267	0,358
6	1,187	3,801	0,181	0,390
7	1,220	2,836	0,183	0,316
8	2,160	1,310	0,250	0,198
9	1,076	0,300	0,173	0,120
10	1,142	0,311	0,177	0,121

Contoh hasil *sliding window* untuk *u*100 terdapat pada Tabel 3. Data dibagi ke dalam beberapa kelompok array, setiap array terdiri atas 120 input dan 1 target.

Tabel 3. Contoh hasil *sliding window* untuk *u*100

Array ke-	Variabel *Input*						Variabel *Output*
	X_1	X_2	X_3	X_4	…	X_{120}	Y
1	0,316	0,280	0,286	0,314	…	0,546	0,686
2	0,280	0,286	0,314	0,328	…	0,686	0,674
3	0,286	0,314	0,328	0,267	…	0,674	0,539
4	0,314	0,328	0,267	0,181	…	0,539	0,482
5	0,328	0,267	0,181	0,183	…	0,482	0,439
6	0,267	0,181	0,183	0,250	…	0,439	0,438
7	0,181	0,183	0,250	0,173	…	0,438	0,244
8	0,183	0,250	0,173	0,177	…	0,244	0,357
9	0,250	0,173	0,177	0,213	…	0,357	0,410
10	0,173	0,177	0,213	0,250	…	0,410	0,381

Contoh hasil sliding window untuk data *v*100 terdapat pada Tabel 4.

Tabel 4 Contoh hasil *sliding window* untuk *v*100

Array ke-	Variabel *Input*						Variabel *Output*
	X_1	X_2	X_3	X_4	…	X_{120}	Y
1	0,331	0,363	0,361	0,434	…	0,548	0,568
2	0,363	0,361	0,434	0,386	…	0,568	0,536
3	0,361	0,434	0,386	0,358	…	0,536	0,494
4	0,434	0,386	0,358	0,390	…	0,494	0,571
5	0,386	0,358	0,390	0,316	…	0,571	0,461
6	0,358	0,390	0,316	0,198	…	0,461	0,500
7	0,390	0,316	0,198	0,120	…	0,500	0,465
8	0,316	0,198	0,120	0,121	…	0,465	0,421
9	0,198	0,120	0,121	0,156	…	0,421	0,350
10	0,120	0,121	0,156	0,234	…	0,350	0,363

Pembagian *Data Training* dan *Data Testing*

Hasil ukuran pembagian data training dan data testing terdapat pada Tabel 5.

Tabel 5 Hasil ukuran pembagian *data training* dan *data testing*

Pembagian Data	Jumlah data
60:40	14564
70:30	10923
80:20	7282
90:10	3641

Perhitungan Model LSTM dengan *Data Training*

Contoh hasil perhitungan dengan *data training* berupa nilai *loss* dan *vall loss* terdapat pada Tabel 6.

Tabel 6 Contoh hasil *loss* dan *vall loss*

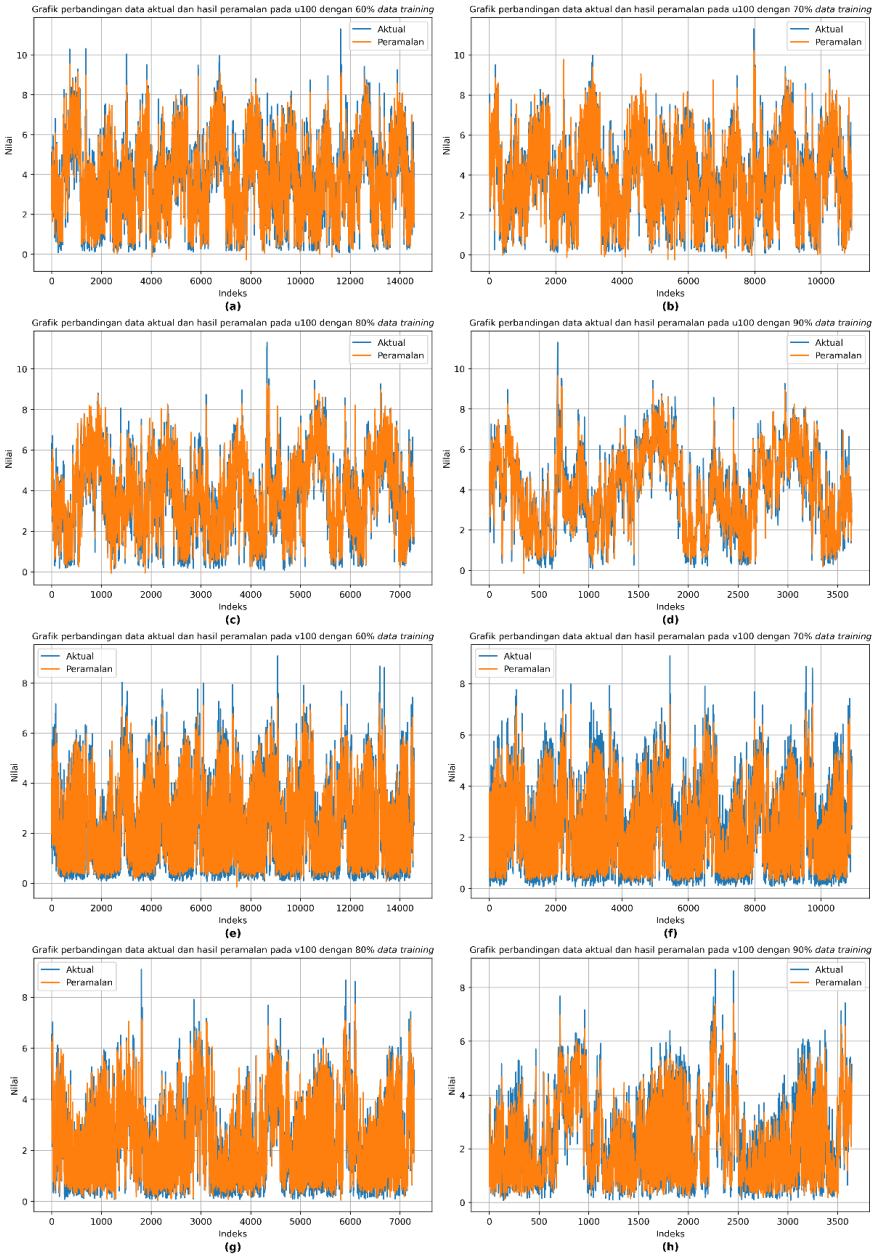
Iterasi	*u*100								*v*100							
	60:40		70:30		80:20		90:10		60:40		70:30		80:20		90:10	
	loss	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*	*loss*	*Vall loss*
1.	0,073	0,065	0,068	0,060	0,068	0,063	0,068	0,061	0,074	0,069	0,081	0,069	0,073	0,068	0,072	0,069
2.	0,059	0,059	0,057	0,057	0,054	0,052	0,054	0,046	0,067	0,074	0,067	0,069	0,068	0,067	0,066	0,065
3.	0,051	0,047	0,047	0,044	0,047	0,049	0,045	0,050	0,067	0,066	0,066	0,068	0,061	0,053	0,056	0,053
4.	0,045	0,049	0,044	0,045	0,044	0,044	0,043	0,048	0,062	0,057	0,058	0,054	0,048	0,046	0,047	0,045
5.	0,043	0,044	0,044	0,043	0,043	0,046	0,043	0,043	0,053	0,055	0,051	0,046	0,044	0,046	0,044	0,046
6.	0,043	0,045	0,043	0,043	0,043	0,044	0,043	0,043	0,046	0,046	0,044	0,045	0,044	0,045	0,044	0,046
7.	0,043	0,043	0,043	0,044	0,043	0,045	0,043	0,044	0,044	0,047	0,043	0,044	0,043	0,044	0,043	0,043
8.	0,043	0,045	0,043	0,043	0,042	0,045	0,042	0,043	0,043	0,044	0,043	0,043	0,043	0,043	0,043	0,043

Perhitungan Model LSTM dengan *Data Testing*

Pada bagian ini, model disimulasikan untuk meramalkan nilai target pada *data testing*. Contoh perbandingan hasil peramalan dan nilai aktualnya terdapat pada Tabel 7.

Tabel 7 Contoh perbandingan data aktual dan hasil peramalan

No	***u*100**								***v*100**							
	60:40		**70:30**		**80:20**		**90:10**		**60:40**		**70:30**		**80:20**		**90:10**	
	Aktual	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**	**Aktual**	**Peramalan**
1	3,280	4,122	4,707	5,074	3,639	4,681	5,213	4,033	2,299	3,027	2,287	2,622	6,422	6,255	1,316	0,594
2	4,099	3,757	4,540	4,836	4,833	4,068	4,771	5,374	1,422	2,034	1,746	1,214	6,545	6,136	2,076	1,453
3	3,565	2,228	5,315	4,065	5,009	5,432	4,747	4,800	2,836	1,981	1,625	1,020	5,546	5,788	3,285	2,132
4	3,316	3,329	5,716	5,218	6,332	5,199	4,910	4,689	2,517	3,904	1,444	1,751	3,965	5,919	3,039	1,686
5	3,701	4,268	5,953	5,891	3,182	4,234	4,287	4,764	2,309	2,335	2,437	1,827	4,469	5,342	2,310	2,238
6	4,688	3,851	6,120	5,639	3,212	3,563	3,231	4,410	3,923	2,553	1,554	1,398	4,355	5,047	2,585	2,461
7	3,742	3,073	5,738	5,707	4,322	5,020	3,141	3,671	4,055	4,069	1,821	0,869	2,157	4,019	3,894	2,945
8	2,684	3,488	6,894	5,750	4,211	4,624	4,579	3,799	3,839	3,982	3,702	1,820	3,008	2,380	2,374	2,323
9	4,352	3,773	6,758	7,154	2,773	2,791	3,291	4,765	3,676	3,495	2,272	3,758	3,905	4,285	2,506	1,759



Gambar 4. Grafik perbandingan data aktual dan hasil peramalan masing masing pembagian data

Evaluasi Model

Evaluasi model dilakukan dengan menghitung RMSE pada masing-masing pembagian data. Hasilnya terdapat pada Tabel 8.

Tabel 8 Hasil akurasi untuk setiap pembagian data

Variabel Data	Pembagian Data			
	60:40	**70:30**	**80:20**	**90:10**
*u*100	0,0552	0,0555	0,0573	0,0541
*v*100	0,0573	0,0567	0,0633	0,0572

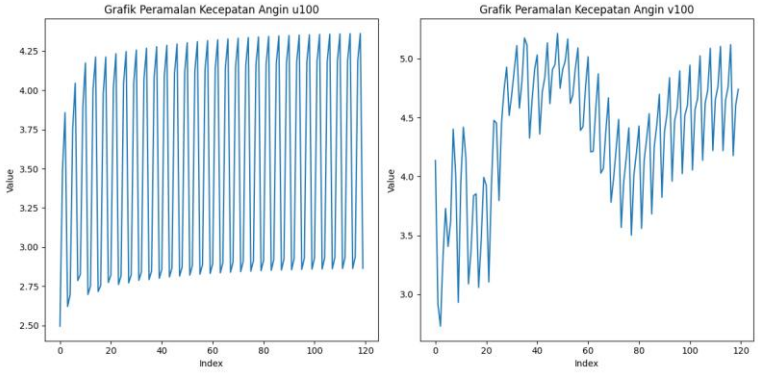
Peramalan dan Denormalisasi

Contoh hasil peramalan kecepatan angin di PLTB Jeneponto pada bulan Januari 2025 terdapat pada Tabel 9.

Tabel 9 Contoh hasil peramalan kecepatan angin sebelum dan sesudah didenormalisasi

No	Sebelum Denormalisasi		Setelah Denormalisasi	
	***U*100**	***V* 100**	***U*100**	***V* 100**
0	0,299	0,284	2,495	4,136
1	0,355	0,341	3,488	2,915
2	0,377	0,367	3,858	2,729
3	0,304	0,294	2,621	3,323
4	0,314	0,300	2,696	3,729
5	0,379	0,366	3,754	3,406
6	0,397	0,382	4,045	3,619
7	0,313	0,288	2,786	4,402
8	0,326	0,293	2,825	4,010
9	0,394	0,363	3,900	2,931
10	0,407	0,381	2,495	4,136

Gambar 3 memuat grafik hasil peramalan kecepatan angin setelah dinormalisasi.



Gambar 5. Grafik hasil peramalan kecepatan angin

PEMBAHASAN

Data preprocessing

Data preprocessing dilakukan dengan pembersihan data, normalisasi, dan *sliding window*. Pembersihan data, dilakukan dengan fungsi *isnull* untuk memeriksa apakah terdapat data yang tidak memiliki nilai dan fungsi *duplicated* untuk memeriksa apakah terdapat data yang memiliki duplikat. Pada Gambar 3.1 dapat dilihat bahwa nilai *missing value* dan *duplicated value* adalah 0, yang artinya tidak terdapat *missing value* dan duplikasi pada data. Oleh karena itu, tahapan dilanjutkan ke normalisasi data. Normalisasi data dilakukan dengan batas 0,1 sampai 0,9 untuk menghindari nilai 0

dan 1, ini perlu dilakukan karena fungsi aktivasi yang akan digunakan adalah fungsi aktivasi sigmoid yang tidak akan menghasilkan nilai 0 dan 1. Setelah dinormalisasi, semua data akan berada dalam rentang 0,1 sampai 0,9. Contoh hasil normalisasi tercantum pada Tabel 3.2. Tahap selanjutnya adalah *sliding window*. Pada *sliding window*, data dibagi dalam pasangan *input* dan *target*. Setiap *n* data berurutan akan dimasukkan ke dalam *input*, dan data setelahnya atau data ke-*n+1* dijadikan *target*. Proses ini dilakukan hingga seluruh data terbentuk menjadi pasangan *input-target*. Array *input* nantinya akan diperlakukan sebagai variabel *x* pada saat perhitungan model LSTM dan array *target* akan menjadi variabel *y* sebagai acuan atau nilai aktual yang akan dituju saat peramalannya. Hasilnya, dari normalisasi terdapat 36.373 array. Pada variabel *input*, setiap array nya berisi yang berisi 120 data, sedangkan pada variabel *target* berisi 1 data sebagai acuan peramalan pada setiap array. Contoh hasil proses *sliding window* terdapat pada Tabel 3.3. dan Tabel 3.4

Perhitungan Model LSTM dengan Data Training

Perhitungan LSTM dilakukan dengan *data training* dan *data testing* yang telah dibagi sebelumnya. Pada tahap ini, model dilatih dengan 100 iterasi perhitungan. Hasil dari pelatihan model adalah nilai *loss* dan *vall_loss*. *Loss* adalah nilai *error* dari perhitungan pada *data training* dan *vall_loss* adalah nilai *error* dari perhitungan *data testing*. Contoh nilai *loss* dan *vall loss* yang diperoleh terdapat pada Tabel 3.6

Perhitungan Model LSTM dengan Data Testing

Pada perhitungan model LSTM dengan data testing, dilakukan dengan melakukan simulasi perhitungan variabel target pada data testing. Contoh hasil perbandingan data aktual dengan data peramalan pada variabel target data testing terdapat pada Tabel 3.7. Grafik perbandingan data aktual dan hasil peramalan terdapat pada Gambar 3.1. Pada Gambar 3.1, terdapat 8 grafik yang masing-masing mewakili pembagian data pada satu variabel. Grafik (a) dan (e) menggunakan 60% *data training* dan 40% *data testing* dengan 14.567 data aktual dan hasil peramalan. Grafik (b) dan (f) menggunakan pembagian 70% *data training* dan 30% *data testing* dengan 10.926 data. Grafik (c) dan (g) memakai 80% *data training* dan 20% *data testing* dengan 7.285 data. Sedangkan grafik (d) dan (h) menggunakan 90% *data training* dan 10% *data testing*, menampilkan 3.644 data. Pada semua grafik, nilai aktual diwakili oleh garis berwarna biru, sedangkan nilai peramalan ditunjukkan dengan garis oranye. Terlihat bahwa hasil peramalan dapat memprediksi data dengan baik, kecuali pada beberapa nilai ekstrim. Dari hasil peramalan ini pada bagian ini, dihitung akurasi model pada tahapan selanjutnya.

Evaluasi Model

Evaluasi model dilakukan dengan menghitung nilai RMSE dari peramalan yang dilakukan pada *data testing*. Nilai RMSE menghitung tingkat akurasi model dalam memprediksi variabel *target* pada *data training*, Berdasarkan Tabel 8, diperoleh bahwa pada variabel *u*100, nilai RMSE terendahnya adalah 0,0541 dengan pembagian 90% data training dan 10% data testing. Pada v100, nilai RMSE terendahnya adalah 0,0567 dengan 70% data training dengan 30% data testing. Oleh karena itu, peramalan selanjutnya dilakukan menggunakan kedua pembagian data tersebut.

Hasil Peramalan

Peramalan dilakukan pada pembagian data dengan nilai *error* paling rendah berdasarkan Tabel 8. Hasil peramalan, masih berada dalam rentang 0,1-0,9. Setelah dilakukan denormalisasi, hasilnya dalam rentang sebenarnya seperti yang terdapat pada Tabel 9. Grafik hasil peramalan terlihat pada Gambar 3. yang memperlihatkan dua grafik hasil peramalan kecepatan angin selama 120 langkah ke depan, atau setara dengan satu bulan setelah titik data terakhir. Grafik sebelah kiri menunjukkan kecepatan angin u100 yang mengalami tren kenaikan secara bertahap dengan pola periodik yang konsisten. Seluruh nilai berada di atas 2 m/s. Sementara itu, grafik sebelah kanan menggambarkan kecepatan angin v100 yang menunjukkan fluktuasi lebih besar dan pola yang

kurang stabil dibandingkan u100, namun nilainya tetap berada di atas 2 m/s sepanjang periode peramalan.

SIMPULAN

Berdasarkan uraian pada bagian Hasil dan Pembahasan, dapat disimpulkan bahwa hasil peramalan keempat pembagian data, memiliki nilai RMSE yang berbeda. Pada u100, nilai RMSE masing-masing pembagian data secara berurutan adalah 0,0552, 0,0555, 0,0573, dan 0,0541. Sementara itu, untuk v100 adalah 0,0573, 0,0567, 0,0633, dan 0,0572. Sehingga nilai RMSE terendah pada pembagian 90% data training dengan 10% data testing untuk u100 dan 70% data training dengan 30% data testing untuk v100. Peramalan dilakukan untuk 120 langkah atau satu bulan setelah dataset terakhir, menggunakan model yang telah dilatih dengan pembagian dataset terbaik pada masing-masing variabel. Hasil peramalan kecepatan angin di PLTB Jeneponto, kedua variabel memiliki nilai yang fluktuatif dan trend naik. Hasil rata-rata kecepatan angin pada u100 adalah 3,31 m/s dalam rentang 2,49 m/s - 4,36 m/s. Hal tersebut berarti angin dari timur ke barat atau sebaliknya diramalkan berada dalam kisaran tersebut. Pada variabel v100, rata-rata kecepatan anginnya adalah 4,36 m/s dan semuanya berada dalam rentang 2,72 m/s -5,21 m/s. Berlaku hal yang sama, angin dari timur ke barat atau sebaliknya diramalkan pada kisaran tersebut. Turbin angin PLTB dapat beroperasi .

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih kepada Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Universitas Jenderal Soedirman atas dukungan fasilitas dan bimbingan akademik selama proses penelitian ini berlangsung. Penghargaan juga disampaikan kepada tim laboratorium dan pihak-pihak yang telah membantu dalam proses pengolahan data kecepatan angin, serta kepada rekan-rekan sejawat yang telah memberikan kritik dan saran yang membangun dalam penyusunan naskah ini. Seluruh kontribusi tersebut sangat berarti dalam mendukung kelancaran dan penyelesaian penelitian mengenai peramalan kecepatan angin yang dilakukan.

DAFTAR RUJUKAN

Hakimi, F. D. D., Anshori, Moch. N. A., & Asyhar, A. H. (2017). Peramalan Kecepatan Angin yang Direkam oleh Sistem AWS dengan Analisis Fuzzy Time Series. *Kubik: Jurnal Publikasi Ilmiah Matematika*, *2*(2), 24–32. https://doi.org/10.15575/kubik.v2i2.1857

Haryanti, M., Yulianti, B., & Ningrum, N. K. (2023). Pembangkit Listrik Tenaga Angin untuk Aplikasi Mikropower menggunakan Mikroturbin Generator. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *11*(1), 143. https://doi.org/10.26760/elkomika.v11i1.143

Indra Sanjaya, F., & Heksaputra, D. (2020). Prediksi Rerata Harga Beras Tingkat Grosir Indonesia dengan Long Short Term Memory. *7*(2), 163–174. http://jurnal.mdp.ac.id

Miftahuddin, Y., Umaroh, S., & Karim, F. R. (2020). Perbandingan Metode Perhitungan Jarak Euclidean, Haversine, dan Manhattan Dalam Penentuan Posisi Karyawan. *Jurnal Tekno Insentif*, *14*(2), 69–77. https://doi.org/10.36787/jti.v14i2.270

Muhammad, R., & Nurhaida, I. (2025). Penerapan LSTM dalam Deep Learning untuk Prediksi Harga Kopi Jangka Pendek dan Jangka Panjang. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, *10*(1), 554–564. https://doi.org/10.29100/jipi.v10i1.5904

Puteri, D. I. (2023). Implementasi Long Short Term Memory (LSTM) dan Bidirectional Long Short Term Memory (BiLSTM) dalam Prediksi Harga Saham Syariah. *Euler : Jurnal Ilmiah Matematika, Sains dan Teknologi*, *11*(1), 35–43. https://doi.org/10.34312/euler.v11i1.19791

Putra, B. T., Hunusalela, Z. F., & Satya, R. R. D. (2024). Usulan Perencanaan Persediaan Produk FMCG Menggunakan Metode Algoritma Apriori dan Jaringan Syaraf Tiruan (JST) pada PT Borwita Indah. *Jurnal Rekayasa Sistem Industri*, *13*(2), 29–44. https://doi.org/10.26593/jrsi.v13i2.7015.29-44

Rahmawati, A., Sulandari, W., Subanti, S., & Yudhanto, Y. (2023). Penerapan Metode Reccurent Neural Network dengan Pendekatan Long Short-Term Memory (LSTM) untuk Meramalkan Harga Saham Hybe Corporation The Application of Recurrent Neural Network Method with the Long Short-Term Memory (LSTM) Approach to Forecast Hybe Corporation's Stock Price. *Jurnal Bumigora Information Technology (BITe)*, *5*(1), 65–76. https://doi.org/10.30812/bite/v5i1.2973

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*, *8*(3), 184–196.

Yusuf, M. (2020). *Peramalan Kecepatan Angin Menggunakan Elman Reccurent Neural Network untuk Mengetahui Besar Daya yang dibangkitkan pada Turbin Angin (Studi Kasus Kota Sibolga)*. Universitas Sumatera Utara.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into Deep Learning Release 0.16.1*.