

**PREDIKSI PASANG SURUT AIR LAUT MENGGUNAKAN  
RECURRENT NEURAL NETWORK – LONG SHORT TERM  
MEMORY: STUDI KASUS STASIUN METEOROLOGI  
MARITIM TANJUNG PERAK SURABAYA**

**SKRIPSI**

oleh:  
**LAILA LILIK PUSPITASARI**  
**185090307111002**

**PROGRAM STUDI: S1 FISIKA**



**DEPARTEMEN FISIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS BRAWIJAYA  
MALANG  
2024**

*Halaman ini sengaja dikosongkan*

**PREDIKSI PASANG SURUT AIR LAUT MENGGUNAKAN  
RECURRENT NEURAL NETWORK – LONG SHORT TERM  
MEMORY: STUDI KASUS STASIUN METEOROLOGI  
MARITIM TANJUNG PERAK SURABAYA**

**SKRIPSI**

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains  
dalam bidang fisika

oleh:  
**LAILA LILIK PUSPITASARI**  
**185090307111002**

**PROGRAM STUDI: S1 FISIKA**



**DEPARTEMEN FISIKA**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**UNIVERSITAS BRAWIJAYA**  
**MALANG**  
**2024**

*Halaman ini sengaja dikosongkan*

## LEMBAR PENGESAHAN

### PREDIKSI PASANG SURUT AIR LAUT MENGGUNAKAN *RECURRENT NEURAL NETWORK – LONG SHORT TERM MEMORY: STUDI KASUS STASIUN METEOROLOGI MARITIM TANJUNG PERAK SURABAYA*

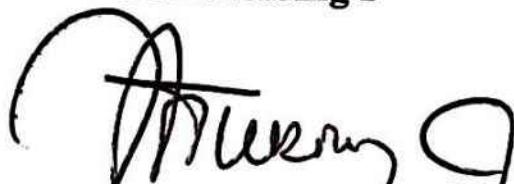
oleh:

**LAILA LILIK PUSPITASARI**  
**185090307111002**

**PROGRAM STUDI: S1 FISIKA**

Telah dipertahankan di depan Majelis Penguji pada tanggal 23  
Desember 2024 dan dinyatakan memenuhi syarat untuk  
memperoleh gelar Sarjana Sains dalam bidang fisika

**Pembimbing I**



Prof.Dr.rer.nat. Abdurrof, S.Si., M.  
NIP. 197209031994121001

**Pembimbing II**



Gancang Saroja, S.Si., M.T.  
NIP. 197711182005011002

Mengetahui,  
Ketua Departemen Fisika  
Fakultas MIPA Universitas Brawijaya



Dr. Eng. Masruroh, S.Si., M.Si.  
NIP. 197512312002122002

*Halaman ini sengaja dikosongkan*

## **LEMBAR PERNYATAAN**

Saya yang bertanda tangan di bawah ini :

Nama : Laila Lilik Puspitasari  
NIM : 185090307111002  
Departemen/Program Studi : Fisika/Fisika  
Penulis Skripsi berjudul :

### **PREDIKSI PASANG SURUT AIR LAUT MENGGUNAKAN *RECURRENT NEURAL NETWORK – LONG SHORT TERM MEMORY: STUDI KASUS STASIUN METEOROLOGI MARITIM TANJUNG PERAK SURABAYA***

Dengan ini menyatakan bahwa:

1. Tugas akhir ini adalah benar-benar karya saya sendiri dan bukan hasil plagiat dari karya orang lain. Karya-karya yang tercantum dalam Daftar Pustaka tugas akhir ini, semata-mata digunakan sebagai rujukan atau referensi.
2. Apabila di kemudian hari diketahui bahwa isi tugas akhir merupakan hasil plagiat, maka saya bersedia menanggung segala akibat dari keadaan tersebut.

Demikian pernyataan ini dibuat dengan segala kesadaran.

**Malang, 04 Desember 2024**

**Yang menyatakan,**



**LAILA LILIK PUSPITASARI**  
**185090307111002**

*Halaman ini sengaja dikosongkan*

**PREDIKSI PASANG SURUT AIR LAUT MENGGUNAKAN  
RECURRENT NEURAL NETWORK – LONG SHORT TERM  
MEMORY: STUDI KASUS STASIUN METEOROLOGI  
MARITIM TANJUNG PERAK SURABAYA**

**ABSTRAK**

Prediksi pasang surut air laut memiliki peranan penting dalam berbagai aplikasi maritim dan pesisir. Penelitian ini menggunakan model *Recurrent Neural Network-Long Short-Term Memory* (RNN-LSTM) untuk memprediksi ketinggian pasang surut air laut menggunakan data dari Stasiun Meteorologi Maritim Tanjung Perak Surabaya. Model yang diterapkan terdiri dari LSTM *Bidirectional* dengan 128 *neuron* dan 256 *neuron*, yang memproses data secara sekuensial dari dua arah, serta *layer flatten* untuk meratakan *output* dari LSTM menjadi vektor satu dimensi. Ditambahkan juga *layer dense* dengan 256 *neuron* dan fungsi aktivasi *Swish*, serta *layer dropout* untuk menghindari *overfitting*. *Hyperparameter* yang digunakan meliputi *scheduler* dan *Adam optimizer* untuk memaksimalkan efisiensi pembelajaran. *Huber Loss* digunakan sebagai *loss function* untuk mengukur perbedaan antara prediksi dan nilai aktual. Sementara itu, *Root Mean Square Error* (RMSE) dan *Mean Absolute Error* (MAE) digunakan sebagai metrik evaluasi kinerja model. Hasil penelitian menunjukkan bahwa model RNN-LSTM mampu memprediksi pasang surut air laut dengan akurasi yang baik. Penggunaan rentang data yang lebih panjang dalam pelatihan terbukti meningkatkan akurasi prediksi, meskipun terdapat fluktiasi pada prediksi jangka pendek. Model ini juga mampu memprediksi pasang surut air laut untuk beberapa bulan ke depan, meskipun terdapat penurunan presisi pada prediksi jangka panjang. Secara keseluruhan, penelitian ini menunjukkan bahwa model RNN-LSTM dapat diandalkan untuk aplikasi prediksi pasang surut air laut dalam jangka panjang.

**Kata kunci:** RNN-LSTM, pasang surut air laut, prediksi, RMSE, MAE.

*Halaman ini sengaja dikosongkan*

**TIDAL WAVES PREDICTION USING RECURRENT  
NEURAL NETWORK - LONG SHORT TERM MEMORY:  
CASE STUDY TANJUNG PERAK METEOROLOGY  
MARITIME STATION SURABAYA**

**ABSTRACT**

Tidal prediction plays an important role in various maritime and coastal applications. This study utilizes the Recurrent Neural Network-Long Short-Term Memory (RNN-LSTM) model to predict sea level heights using data from the Maritime Meteorology Station in Tanjung Perak, Surabaya. The implemented model consists of a Bidirectional LSTM with 128 and 256 neurons, which sequentially processes data in two directions, as well as a flatten layer to convert the LSTM output into a one-dimensional vector. A dense layer with 256 neurons and the Swish activation function is also added, along with a dropout layer to prevent overfitting. The hyperparameters include a scheduler and Adam optimizer to maximize learning efficiency. Huber Loss is the loss function to measure the difference between predictions and actual values. At the same time, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are employed as metrics to evaluate the model's performance. The results show that the RNN-LSTM model can predict tidal heights accurately. Using a longer data range during training has been proven to enhance prediction accuracy, although fluctuations occur in short-term predictions. The model is also able to predict tidal heights several months into the future, despite a slight decline in precision for long-term predictions. Overall, this study demonstrates that the RNN-LSTM model is reliable for long-term tidal prediction applications.

**Keywords:** RNN-LSTM, tidal waves, prediction, RMSE, MAE

*Halaman ini sengaja dikosongkan*

## KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadiran Allah SWT, karena atas segala rahmat serta karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul “ **Prediksi Pasang Surut Air Laut Menggunakan Recurrent Neural Network – Long Short Term Memory: Studi Kasus Stasiun Meteorologi Maritim Tanjung Perak Surabaya**” dengan baik dan lancar.

Skripsi ini dapat selesai dan dalam pembuatannya tidak lepas dari bantuan dan bimbingan dari berbagai pihak. Oleh karena itu penulis mengucapkan terima kasih kepada:

1. Orang tua yang telah sabar memberi dukungan dan doa mulai dari nol.
2. Bapak Prof.Dr.rer.nat. Abdurrouf, S.Si., M.Si., selaku dosen pembimbing pertama dan Bapak Gancang Saroja, S.Si, M.T., selaku dosen pembimbing kedua yang telah memberikan waktu, bimbingan, dan arahan yang sangat berharga dalam penyusunan skripsi ini.
3. Bapak Fajar Setiawan, S.Si, sebagai pembimbing dari BMKG Stasiun Meteorologi Maritim Tanjung Perak yang telah membantu dalam mengarahakan pengambilan data dan menganalisa data pasang surut air laut.
4. BMKG Stasiun Meteorologi Maritim Tanjung Perak yang menyediakan data dan akses untuk keperluan tugas akhir penulis.
5. Segenap Dosen Jurusan Fisika Universitas Brawijaya yang telah memberikan banyak ilmu selama penulis menempuh Pendidikan sarjana.
6. Google, Github, Stackoverflow, Coursera karena telah menunjang penulisan skripsi ini.
7. Teman-teman Fisika UB 2018, teman-teman dari Alumni Bangkit 2021 dan semua sahabat saya serta semua pihak, yang tidak dapat penulis sebutkan satu-persatu disini.
8. Pembaca yang telah meluangkan waktu untuk membaca isi dari skripsi ini.

Semoga Allah SWT memberikan balasan berlipat atas bantuan dan dukungan yang telah diberikan. Dan semoga skripsi ini bermanfaat khususnya bagi penulis, dan pembaca.

Penulis menyadari, skripsi ini masih banyak terdapat kekurangan, sehingga kritik dan saran sangat penulis harapkan.

Malang, 23 Desember 2024



Penulis

## DAFTAR ISI

HALAMAN JUDUL .....	i
LEMBAR PENGESAHAN .....	v
LEMBAR PERNYATAAN .....	vii
ABSTRAK.....	ix
ABSTRACT.....	xi
KATA PENGANTAR .....	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR .....	xvii
DAFTAR TABEL.....	xxi
DAFTAR LAMPIRAN.....	xxiii
BAB I PENDAHULUAN .....	1
1.1.    Latar Belakang.....	1
1.2.    Rumusan Masalah .....	2
1.3.    Batasan Masalah .....	3
1.4.    Tujuan Penelitian.....	3
1.5.    Manfaat Penelitian.....	3
BAB II TINJAUAN PUSTAKA .....	5
2.1.    Gelombang Permukaan Air Laut.....	5
2.2.    Gelombang Pasang Surut .....	6
2.2. <i>Artificial Intelligence</i> .....	7
2.3. <i>Machine Learning</i> .....	7
2.4.    Supervised Learning.....	8
2.5. <i>Artificial Neural Network (ANN)</i> .....	8
2.6. <i>Recurrent Neural Networks (RNN)</i> .....	10
2.7. <i>Long Short Term Memory</i> .....	11
2.8. <i>Root Mean Square Error (RMSE)</i> .....	13

2.9. <i>Mean Absolute Error</i> (MAE) .....	14
2.10. Python .....	14
2.11. TensorFlow .....	15
2.12. Keras .....	15
<b>BAB III METODOLOGI .....</b>	<b>17</b>
3.1. Tempat dan Waktu Penelitian .....	17
3.2. Alat dan Bahan.....	17
3.3. Tahapan Penelitian.....	17
3.3.1 Pengambilan dataset.....	18
3.3.2 Penyiapan data .....	19
3.3.3 Pembuatan model <i>neural network</i> dan <i>training</i> .....	20
3.3.4 Pengujian <i>neural network</i> .....	22
3.3.5 Evaluasi model.....	23
<b>BAB IV HASIL DAN PEMBAHASAN .....</b>	<b>25</b>
4.1. Perbandingan Kinerja RNN-LSTM dan CNN .....	25
4.2. Evaluasi Kinerja RNN-LSTM dengan Variasi Rentang <i>Data Input</i> .....	28
4.3. Evaluasi Kinerja RNN-LSTM untuk Prediksi Beberapa Bulan Kedepan .....	30
4.4. Evaluasi Perbandingan RMSE dan MAE.....	32
4.5. Evaluasi Pengaruh Kesalahan terhadap Sektor Kelautan.	33
<b>BAB V PENUTUP .....</b>	<b>35</b>
5.1. Kesimpulan .....	35
5.2. Saran.....	35
<b>DAFTAR PUSTAKA.....</b>	<b>37</b>
<b>LAMPIRAN .....</b>	<b>39</b>

## DAFTAR GAMBAR

Gambar 2.1	Skema Pembentukan Tonjolan Pasang Surut.....	6
Gambar 2.2	Model ANN secara umum .....	9
Gambar 2.3	Arsitektur RNN .....	10
Gambar 2.4	Arsitektur LSTM.....	11
Gambar 3.1	Diagram Tahapan Penelitian.....	18
Gambar 3.2	Data Pasang Surut Air Laut.....	19
Gambar 3.3	Data Pasang Surut Air Laut per-30 Menit.....	20
Gambar 4.1	Perbandingan antara Data Real, CNN, dan RNN-LSTM untuk Sensor prs .....	26
Gambar 4.2	Perbandingan antara Data Real, CNN, dan RNN-LSTM untuk Sensor pr2.....	26
Gambar 4.3	Grafik Tren Nilai Metrik Variasi <i>Data Input</i> .....	29
Gambar 4.4	Grafik Tren Nilai Metrik Variasi <i>Data Output</i> .....	31
Gambar A.1.1	Hasil Prediksi Model CNN Sensor pr2 .....	39
Gambar A.1.2	Hasil Prediksi Model CNN Sensor prs .....	39
Gambar A.1.3	Hasil Pengukuran Metrik CNN .....	40
Gambar A.1.4	Prediksi CNN Sensor pr2 Bulan ke 21 .....	40
Gambar A.1.5	Prediksi CNN Sensor prs Bulan ke 21 .....	40
Gambar A.2.1	Prediksi RNN-LSTM Sensor pr2 dengan <i>Input</i> 23 Bulan Sebelumnya.....	41
Gambar A.2.2	Prediksi RNN-LSTM Sensor pr2 dengan <i>Input</i> 23 Bulan Sebelumnya.....	41
Gambar A.2.3	Hasil Pengukuran Metrik untuk RNN-LSTM <i>Input</i> 23 Bulan Sebelumnya.....	42
Gambar A.2.4	Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan <i>Input</i> 23 Bulan Sebelumnya.....	42
Gambar A.2.5	Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan <i>Input</i> 23 Bulan Sebelumnya.....	42
Gambar A.2.6	Prediksi RNN-LSTM Sensor pr2 dengan <i>Input</i> 18 Bulan Sebelumnya.....	43
Gambar A.2.7	Prediksi RNN-LSTM Sensor prs dengan <i>Input</i> 18 Bulan Sebelumnya.....	43
Gambar A.2.8	Hasil Pengukuran Metrik untuk RNN-LSTM <i>Input</i> 18 Bulan Sebelumnya.....	44

Gambar A.2.9	Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan <i>Input</i> 18 Bulan Sebelumnya .....	44
Gambar A.2.10	Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan <i>Input</i> 18 Bulan Sebelumnya .....	44
Gambar A.2.11	Prediksi RNN-LSTM Sensor pr2 dengan <i>Input</i> 13 Bulan Sebelumnya .....	45
Gambar A.2.12	Prediksi RNN-LSTM Sensor prs dengan <i>Input</i> 13 Bulan Sebelumnya .....	45
Gambar A.2.13	Hasil Pengukuran Metrik untuk RNN-LSTM <i>Input</i> 13 Bulan Sebelumnya .....	46
Gambar A.2.14	Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan <i>Input</i> 13 Bulan Sebelumnya .....	46
Gambar A.2.15	Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan <i>Input</i> 13 Bulan Sebelumnya .....	46
Gambar A.2.16	Prediksi RNN-LSTM Sensor pr2 dengan <i>Input</i> 8 Bulan Sebelumnya .....	47
Gambar A.2.17	Prediksi RNN-LSTM Sensor prs dengan <i>Input</i> 8 Bulan Sebelumnya .....	47
Gambar A.2.18	Hasil Pengukuran Metrik untuk RNN-LSTM <i>Input</i> 8 Bulan Sebelumnya .....	48
Gambar A.2.19	Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan <i>Input</i> 8 Bulan Sebelumnya .....	48
Gambar A.2.20	Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan <i>Input</i> 8 Bulan Sebelumnya .....	48
Gambar A.3.1	Prediksi RNN-LSTM 5 Bulan Pertama pada Sensor pr2 .....	49
Gambar A.3.2	Prediksi RNN-LSTM 5 Bulan Pertama pada Sensor prs.....	49
Gambar A.3.3	Hasil Pengukuran Metrik Prediksi 5 Bulan Pertama .....	49
Gambar A.3.4	Prediksi RNN-LSTM 5 Bulan Kedua pada Sensor pr2 .....	50
Gambar A.3.5	Prediksi RNN-LSTM 5 Bulan Kedua pada Sensor prs.....	50
Gambar A.3.6	Hasil Pengukuran Metrik Prediksi 5 Bulan Kedua ..	50
Gambar A.3.7	Prediksi RNN-LSTM 5 Bulan Ketiga pada Sensor pr2 .....	51

Gambar A.3.8 Prediksi RNN-LSTM 5 Bulan Ketiga pada Sensor prs .....	51
Gambar A.3.9 Hasil Pengukuran Metrik Prediksi 5 Bulan Ketiga	51
Gambar A.3.10 Prediksi RNN-LSTM 5 Bulan Keempat pada Sensor pr2 .....	52
Gambar A.3.11 Prediksi RNN-LSTM 5 Bulan Keempat pada Sensor prs .....	52
Gambar A.3.12 Hasil Pengukuran Metrik Prediksi 5 Bulan Keempat .....	52

*Halaman ini sengaja dikosongkan*

## **DAFTAR TABEL**

Tabel 3.1 Variasi <i>Data Training</i> .....	23
Tabel 3.2 Variasi <i>Data Testing</i> .....	23
Tabel 4.1 Perbandingan antara RNN-LSTM dan CNN .....	25
Tabel 4.2 Performa RNN-LSTM untuk Prediksi 5 Bulan Kedepan	28
Tabel 4.3 Performa RNN-LSTM Prediksi Beberapa Bulan Kedepan .....	30

*Halaman ini sengaja dikosongkan*

## **DAFTAR LAMPIRAN**

LAMPIRAN A.....	39
LAMPIRAN B .....	53

*Halaman ini sengaja dikosongkan*

## **BAB I**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Pasang surut adalah salah satu elemen fisik terpenting dari lautan. Gaya gravitasi yang diberikan pada permukaan laut oleh benda-benda langit menyebabkan naik turunnya air laut secara berkala. Pasang surut sendiri dalam sektor kelautan sangat penting guna lancarnya aktivitas maritim seperti navigasi, perikanan, dan pembangunan infrastruktur. Dengan menganalisis dan memahami pasang surut air laut, mengidentifikasi, meneliti, dan memanfaatkan laut secara efektif membutuhkan prediksi pasang surut sebagai sarana teknis yang penting. Maka dari itu prediksi pasang surut air laut memiliki peranan penting dalam berbagai aplikasi maritim dan pesisir (Ban dkk., 2023).

Perkembangan teknologi kelautan modern, pemantauan dan prediksi pasang surut menjadi pekerjaan dasar dalam pengembangan sumber daya laut atau konstruksi rekayasa pesisir. Pasang surut dapat dibagi menjadi dua kategori: pasang surut astronomi dan pasang surut meteorologi. Pasang surut astronomi adalah fluktuasi periodik air laut yang disebabkan oleh gravitasi benda langit, sedangkan pasang surut meteorologi lebih tidak teratur dan umumnya disebabkan oleh faktor cuaca seperti angin dan tekanan udara. Dalam kondisi ekstrem, seperti angin kencang dan tekanan udara rendah, dapat terjadi gelombang badai (Xu dkk., 2022).

*Machine Learning* (ML) adalah salah satu disiplin ilmu komputer yang mengembangkan algoritma dinamis yang dapat membuat keputusan berdasarkan data. Kemampuan ML telah terbukti menjadi solusi untuk banyak masalah dunia nyata. ML memiliki keunggulan dibandingkan metode tradisional karena dapat membangun model dimensi variabel yang besar dan nonlinier, dengan hubungan yang kompleks dan nilai yang hilang. ML telah terbukti berguna untuk berbagai aplikasi di berbagai bagian sistem Bumi (daratan, lautan, dan atmosfer) dan seterusnya. Contohnya mulai dari algoritma pengambilan, deteksi penyakit tanaman, pembuatan produk baru, koreksi bias, dan akselerasi kode (Ahmad, 2019). *Convolutional Neural Network* (CNN) yang dikembangkan secara khusus untuk *image processing* dapat menangani permasalahan untuk memprediksi data *time series*. Kemampuan CNN dalam memprediksi data *time*

*series* dapat disandingkan dengan LSTM (Luaran Nosius & Alfred, 2021).

Pada penelitian yang dilakukan oleh (Ramadhan *et al.*, 2021) mengenai prediksi pasang surut air laut menggunakan Long Short-Term Memory (LSTM) menunjukkan bahwa LSTM memiliki performa yang lebih unggul dibandingkan dengan metode konvensional, yaitu *Tidal Harmonic Analysis* (THA). Dalam penelitian tersebut, LSTM terbukti lebih efektif dalam menangkap pola jangka panjang dan kompleks pada data pasang surut, yang seringkali sulit diprediksi menggunakan metode tradisional. Dibandingkan dengan THA, yang mengandalkan model harmonik untuk menganalisis komponen periodik pasang surut, LSTM menunjukkan akurasi yang lebih tinggi dalam memprediksi perubahan pasang surut yang tidak terduga. Hasil ini menunjukkan potensi besar dari teknik pembelajaran mendalam, khususnya LSTM, dalam meningkatkan akurasi prediksi pasang surut air laut, yang sangat penting untuk perencanaan dan manajemen sumber daya pesisir.

Berdasarkan penelitian prediksi pasang surut air laut yang sudah dilakukan menggunakan CNN, penulis akan melakukan prediksi pasang surut air laut menggunakan RNN-LSTM yaitu tipe *neural network* yang cocok digunakan untuk data *time series*. Setelah itu, hasil prediksi RNN-LSTM akan dibandingkan dengan data *real* dan hasil prediksi CNN. Untuk area studi yang digunakan dalam penelitian ini adalah di Stasiun BMKG Maritim Tanjung Perak Surabaya.

## 1.2. Rumusan Masalah

Rumusan masalah pada penelitian ini adalah:

1. Bagaimana hasil keakuratan Recurrent Neural Network - *Long Short-Term Memory* (RNN-LSTM) dalam memprediksi pasang surut air laut dibandingkan dengan *Convolutional Neural Network* (CNN)?
2. Bagaimana kemampuan model *Recurrent Neural Network - Long Short-Term Memory* (RNN-LSTM) ketika memprediksi dengan masukan rentang data yang berbeda?
3. Bagaimana kemampuan model *Recurrent Neural Network - Long Short-Term Memory* (RNN-LSTM) ketika memprediksi

pasang surut air laut untuk beberapa bulan mendatang yang semakin jauh?

### **1.3. Batasan Masalah**

Batasan masalah pada penelitian ini adalah:

1. Data masukan yang digunakan diambil dari data Stasiun Meteorologi Maritim Tanjung Perak Surabaya.
2. Data masukan yang diproses sistem berjangka waktu dari bulan Januari 2022 hingga April 2024.
3. Metode penelitian yang digunakan terfokus pada model RNN-LSTM. Model CNN digunakan sebagai pembanding saja.
4. Data pasang surut diambil per 30 menit.

### **1.4. Tujuan Penelitian**

Tujuan pada penelitian ini adalah:

1. Membandingkan kemampuan prediksi antara RNN-LSTM dan CNN dengan mengamati seberapa besar nilai masing-masing Root Mean Square Error (RMSE) dan Mean Absolute Error (MAE).
2. Mengetahui kemampuan RNN-LSTM dalam prediksi ketika menggunakan masukan rentang data yang berbeda.
3. Mengetahui kemampuan RNN-LSTM ketika memprediksi pasang surut air laut beberapa bulan mendatang.

### **1.5. Manfaat Penelitian**

Manfaat yang didapatkan dari penelitian ini adalah

1. Bagi Perguruan Tinggi : Penelitian ini dapat digunakan sebagai referensi dan acuan pada perpustakaan universitas brawijaya
2. Bagi Akademisi : Penelitian ini dapat digunakan sebagai referensi dan pembelajaran di Bidang Fisika Komputasi,
3. Bagi Instansi : Penelitian ini dapat membantu instansi dalam analisis prediksi pasang surut air laut.



## **BAB II**

### **TINJAUAN PUSTAKA**

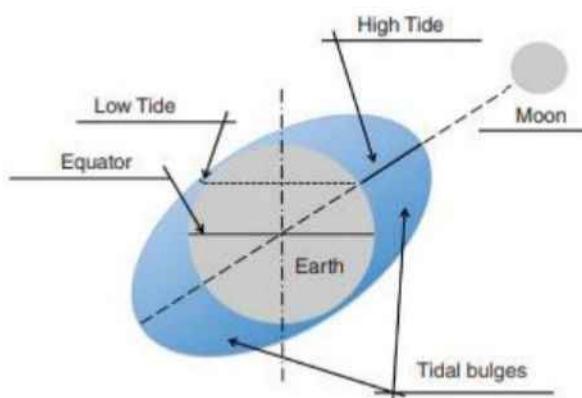
#### **2.1. Gelombang Permukaan Air Laut**

Gelombang permukaan air laut merupakan gelombang yang dihasilkan oleh angin, gempa bumi, dan variasi tekanan atmosfer, serta gaya gravitasi dari Bumi, Matahari, dan Bulan, dapat mempengaruhi struktur laut, transportasi sedimen, dan dinamika pantai. Gelombang laut dapat diklasifikasikan berdasarkan periode gelombang atau panjang gelombangnya, yang juga mencakup mekanisme pembentukan dan pemulihannya. Adapun jenis-jenis gelombang yang diklasifikasikan adalah: Gelombang Kapiler dengan periode kurang dari 0,1 detik yang dipengaruhi oleh tegangan permukaan, Gelombang Ultragravitas dengan periode 0,1-1 detik yang dipengaruhi oleh tegangan permukaan dan gravitasi, Gelombang Gravitasi dengan periode 1-20 detik yang dipengaruhi oleh gravitasi, Gelombang Infragravitas dengan periode 20 detik hingga 5 menit yang dipengaruhi oleh gravitasi dan variasi tekanan atmosfer, Gelombang Periode Panjang dengan periode 5 menit hingga 12 jam yang dipengaruhi oleh gempa bumi dan variasi tekanan atmosfer, Gelombang Pasang Surut dengan periode 12-24 jam yang dipengaruhi oleh gaya gravitasi, Gelombang Transtidal dengan periode lebih dari 24 jam yang dipengaruhi oleh badai dan gaya gravitasi (Toffoli & Bitner-Gregersen, 2017)

Selain periode atau panjang gelombang, karakteristik lain yang penting adalah tipe gelombang berdasarkan kedalaman air di mana gelombang tersebut merambat. Gelombang dapat diklasifikasikan sebagai gelombang air dalam, gelombang air menengah, dan gelombang air dangkal, tergantung pada kedalaman relatif terhadap panjang gelombang. Gelombang di permukaan laut memiliki dampak besar pada banyak aktivitas maritim dan proses biokimia yang terjadi di permukaan dan bawah permukaan laut (Toffoli & Bitner-Gregersen, 2017).

## 2.2 Gelombang Pasang Surut

Gelombang pasang surut adalah gelombang yang terbentuk akibat adanya gaya gravitasi yang tidak seimbang yang ditarik oleh Bulan dan Matahari pada samudra bumi. Dua tonjolan air yang mana satu menghadap Bulan dan satunya lagi berada di sisi yang berlawanan. Hal tersebut tercipta karena adanya kombinasi efek gravitasi dan gaya sentrifugal. Gelombang ini memiliki periode antara 12 hingga 24 jam namun di beberapa tempat ada yang mengalami variasi periode yang berbeda akibat interaksi kompleks dari berbagai faktor.



Gambar 2.1 Skema Pembentukan Tonjolan Pasang Surut (Sumber: Toffoli & Bitner-Gregersen, 2017)

Pada Gambar 2.1 merupakan representasi skematis dari pembentukan tonjolan pasang surut yang dihasilkan oleh daya tarik gravitasi Bulan. Dalam hal ini, Bulan (*Moon*) adalah benda langit yang memiliki pengaruh signifikan terhadap pasang surut air laut di Bumi karena gaya gravitasinya. Gaya gravitasi Bulan menyebabkan dua tonjolan air yang dinamakan Tonjolan Pasang Surut (*Tidal Bulges*) satu di sisi yang menghadap langsung ke Bulan (*High Tide*) dan satu lagi di sisi yang berlawanan (*High Tide*). Hal ini terjadi karena gaya gravitasi Bulan menarik air laut di sisi permukaan, sementara gaya sentrifugal akibat rotasi Bumi menyebabkan tonjolan di sisi yang berlawanan. Pasang Naik (*High Tide*) merupakan area pada permukaan Bumi yang mengalami peningkatan permukaan air laut karena tarikan gravitasi Bulan, hal tersebut ditunjukkan oleh dua titik di sisi yang menghadap ke Bulan dan sisi yang berlawanan. Pasang Surut (*Low Tide*) merupakan area pada permukaan Bumi yang

mengalami penurunan permukaan air laut karena air laut berpindah ke daerah pasang naik. Ini ditunjukkan oleh area di sekitar ekuator yang tidak sejajar dengan Bulan. Ekuator (*Equator*) adalah garis khayal yang membagi Bumi menjadi belahan utara dan selatan. Dalam konteks pasang surut, ekuator adalah acuan untuk menggambarkan distribusi pasang surut (Toffoli & Bitner-Gregersen, 2017).

## **2.2. Artificial Intelligence**

*Artificial Intelligence* (AI) pada dasarnya adalah sistem komputer yang mereproduksi sistem kognisi manusia dengan menggunakan data dari berbagai sumber untuk membuat keputusan dan belajar dari pola yang dihasilkan. AI awalnya diusulkan dalam konferensi komputasi. Pada tahun 1956, John McCarthy dari Dartmouth University mengumpulkan para ahli matematika dan ilmuwan untuk sesi *brainstorming* tentang AI. Mereka berhipotesis bahwa setiap aspek pembelajaran atau fitur lain dari kecerdasan dapat dijelaskan secara rinci sehingga mesin dapat dibuat untuk meniru kecerdasaan. Sejak pertemuan awal tersebut, AI telah berkembang dari sekadar bidang akademis menjadi komponen utama dalam teknologi sosial dan ekonomi, seperti pengenalan suara, diagnosis medis, dan kendaraan otonom. Dalam teknologi modern AI dapat digunakan untuk mengenali pola dan mengambil tindakan berdasarkan data yang tersedia dan model statistik. AI telah menunjukkan kinerja superior dalam berbagai bidang seperti algoritma pengenalan suara dan pola, pemantauan proses di industri, deteksi kesalahan, dan perkiraan, terutama di sektor kesehatan untuk meningkatkan proses perawatan. AI memiliki relevansi dengan *Internet of Things* (IoT). IoT adalah salah satu tren komputasi utama yang membentuk AI. IoT memungkinkan pengumpulan data dalam jumlah besar yang kemudian digunakan untuk melatih model *Machine Learning* (ML) dan *Deep Learning* (DL), yang meningkatkan akurasi dan presisi model-model ini (Hassani dkk., 2020),

## **2.3. Machine Learning**

*Machine Learning* (ML) adalah cabang dari *Artificial Intelligence* (AI) yang memungkinkan komputer untuk memperoleh pengetahuan dari data dan meningkatkan kinerjanya secara bertahap, tanpa memerlukan pemrograman eksplisit untuk setiap tugas tertentu. Kemampuan untuk belajar dari pengalaman inilah yang membedakan

Pembelajaran Mesin dari metode pemrograman konvensional. ML memiliki beberapa tipe, yaitu: *Supervised Learning*, *Unsupervised Learning*, *Semi-supervised Learning*, dan *Reinforcement Learning*. Terdapat beberapa algoritma ML, yaitu *Classification*, *Regression*, *Clustering*, dan *Dimensionality Reduction*. *Classification* biasanya digunakan untuk tugas-tugas yang tujuannya adalah menetapkan data *input* ke kategori yang telah ditentukan sebelumnya. Yang termasuk dalam algoritma tersebut adalah *Naive Bayes*, *SVM*, dan *Random Forest*. *Regression* digunakan untuk memprediksi hasil yang bersifat kontinu. Contoh metode tersebut adalah *Linear Regression* dan *Polynomial Regression*. *Clustering* biasanya digunakan untuk mengelompokkan titik data-data yang tidak diketahui definisnya. Algoritma yang termasuk dalam metode *clustering* adalah *K-means* dan *DBSCAN*. *Dimensionality Reduction* seperti *PCA* digunakan untuk mengurangi jumlah variabel input dalam sebuah model, membantu menyederhanakan model dan meningkatkan kinerjanya. (Sarker, 2021).

## 2.4. Supervised Learning

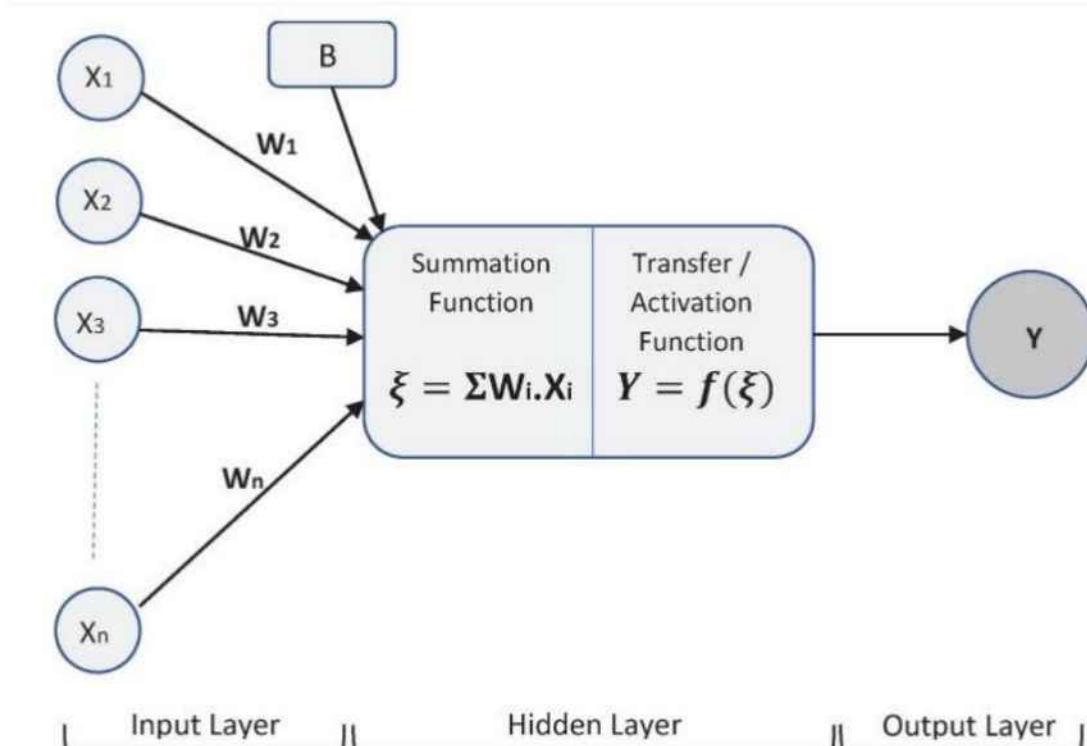
*Supervised Learning* merupakan jenis *Machine Learning*. Algoritma *Supervised Learning* belajar dari dataset berlabel. Dalam pendekatannya, algoritma dilatih menggunakan *dataset* yang berisi pasangan *input-output*, artinya untuk setiap *input*, *output* yang diinginkan sudah diketahui. Algoritma ini bertujuan mempelajari pemetaan dari *input* ke *output* yang dapat digunakan untuk memprediksi *output* untuk data yang baru.

*Supervised Learning* memiliki algoritma untuk membuat prediksi dan kemudian diperbaiki. Perbaikan tersebut yaitu membandingkan prediksinya dengan *output* yang sebenarnya, sehingga secara bertahap meningkatkan kinerjanya. *Supervised Learning* biasanya digunakan untuk klasifikasi, di mana outputnya adalah label kategori, dan regresi, di mana hasil *output* adalah nilai kontinu (Sharma, 2020).

## 2.5. Artificial Neural Network (ANN)

*Artificial Neural Network* adalah model komputasi yang terinspirasi oleh struktur dan fungsi otak manusia, yang terdiri dari simpul-simpul yang saling terhubung (*neuron*) yang bekerja bersama

untuk memproses informasi. Jaringan ini dirancang untuk mengenali pola, belajar dari data, dan membuat prediksi atau keputusan berdasarkan data input. ANN adalah bagian dari algoritma ML dan sangat kuat dalam menangani hubungan yang kompleks dan non-linear dalam data. Jaringan ini terdiri dari beberapa lapisan, termasuk *input layer*, satu atau lebih *hidden layer*, dan *output layer*.



Gambar 2.2 Model ANN secara umum (Sumber: Salah Alaloul & Hannan Qureshi, 2020).

Pada gambar 2.2 pada *input layer*  $X_n$  menandakan bahwa terdapat beberapa input masuk ke jaringan.  $W_n$  merupakan bobot (*Weights*) dari koneksi yang merupakan representasi kekuatan dari tiap-tiap *node*. Bagian pemrosesan dalam ANN terjadi di bagian *hidden layer*. Dalam *hidden layer* terdapat dua operasi fungsi yang dieksekusi. Dua operasi fungsi tersebut adalah *summation function* dan *transfer function* atau dikenal dengan *activation function*. Dalam *summation function* yang merupakan langkah pertama dalam ANN, tiap input  $X_i$  dikalikan dengan bobotnya  $W_i$  lalu hasil dari perkalian masing-masing input dan bobotnya diakumulasikan menjadi *summation function* berikut,

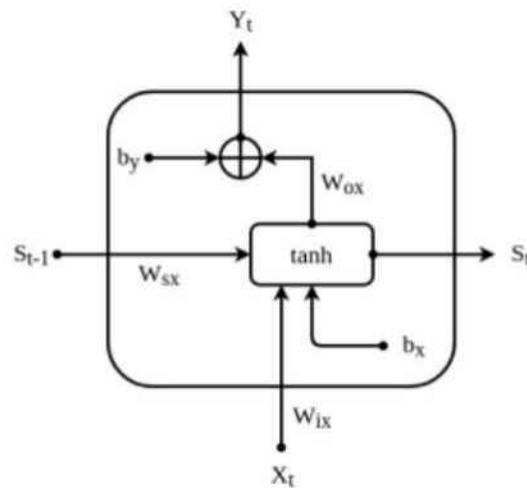
$$\xi = \sum W_i \cdot X_1 \quad (2.1)$$

*Activation function* adalah langkah kedua yang mengubah sinyal input yang diterima dari modul *summation function* dan mengubahnya menjadi *output* dari sebuah *node*. Fungsi tersebut menggunakan nilai dari persamaan 2.1 sebagai input untuk memroses dan mengontrol input untuk aktivasi neuron. *Activation function* yang umum ditemui adalah *Linear*, *Unit Step*, *Rectified Linear Unit* (ReLU), *Sigmoid*, *Gaussian*, dan *Hyperbolic Tangent* (Salah Alaloul & Hannan Qureshi, 2020).

## 2.6. Recurrent Neural Networks (RNN)

RNN merupakan bagian dari arsitektur *Artificial Neural Network* (ANN) yang khususnya cocok untuk menangani data berurutan. RNN dapat mempertahankan keadaan internal yang dapat menangkap informasi dari input sebelumnya sehingga RNN ideal untuk tugas-tugas yang mana urutan input sangat penting, seperti prediksi *time series*, *natural language processing*, dan analisis data sekuensial lainnya.

Arsitektur dari sebuah RNN digambarkan sebagai rangkaian *node* yang saling terhubung. *Node* yang terhubung ini merepresentasikan setiap langkah waktu dalam urutan *input*. Keadaan internal mengalir melalui *node-node* ini, menangkap sifat sekuensial dari data tersebut.

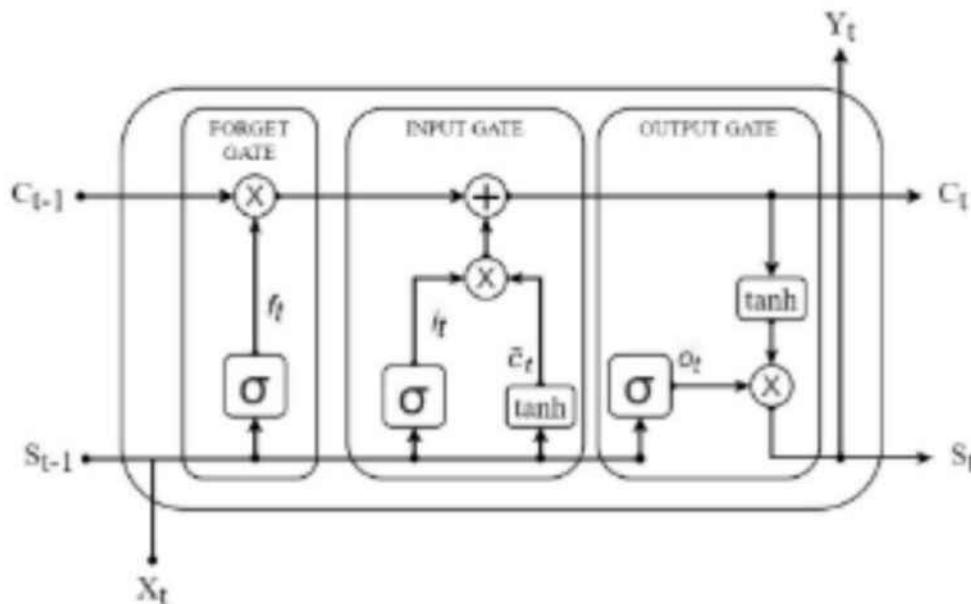


Gambar 2.3 Arsitektur RNN (Sumber: Ramadhan dkk., 2021).

Pada gambar 2.3 menunjukkan bagaimana setiap *input* pada waktu  $t$  mempengaruhi keadaan  $S_t$  dan selanjutnya *output*  $Y_t$ . Gambar tersebut juga menunjukkan sifat RNN yang mana setiap *node* pada waktu  $t$  menerima *input*  $X_t$  dan keadaan sebelumnya  $S_{t-1}$ . Keadaan saat ini  $S_t$  kemudian dihitung dan digunakan untuk menghasilkan *output*. Aliran informasi yang bersifat siklis ini memungkinkan RNN untuk mempertahankan dan memanfaatkan konteks dari *input* sebelumnya saat membuat prediksi (Ramadhan dkk., 2021).

## 2.7. Long Short Term Memory

*Long Short Term Mmoery* (LSTM) adalah penyempurnaan dari RNN tradisional, yang mengatasi masalah *vanishing* dan *exploding gradients*. Masalah-masalah ini seringkali menghambat kemampuan RNN untuk mempelajari ketergantungan jangka panjang dalam data sekuensial, yang dapat menyebabkan penurunan akurasi prediksi. Keterbatasan terkait dengan ketergantungan jangka panjang diatasi dengan menggabungkan *memory cell* dan *gate mechanism*. Arsitektur ini memungkinkan jaringan LSTM untuk secara selektif mempertahankan informasi yang relevan selama periode yang lebih lama, yang mengarah pada peningkatan kinerja dalam berbagai tugas pemodelan prediktif.



Gambar 2.4 Arsitektur LSTM (Sumber: Ramadhan dkk., 2021).

*Memory cell* dan *gates* adalah komponen-komponen yang ada pada LSTM dapat dilihat pada gambar 2.4. *Memory cell* merupakan komponen inti dari LSTM. *Memory cell* didesain untuk mempertahankan informasi dalam jangka waktu lama yang membuat jaringan mengingat detail dari urutan sebelumnya karena hal tersebut sangat penting untuk membuat prediksi yang akurat. *Gates* dalam LSTM terdapat *forget gate*, *input gate* dan *output gate*. Langkah awal dari LSTM ada dalam *forget gate* yaitu dengan membedakan data yang perlu dan yang tidak diperlukan. Dalam proses tersebut digunakanlah fungsi sigmoid berikut,

$$f_t = \sigma(W_{tx}X_t + W_{fs}S_{t-1} + b_f) \quad (2.2)$$

Jika nilai fungsi sigmoid adalah 0, maka data akan diabaikan. Jika nilainya 1, maka data akan diperbarui atau diteruskan.  $X_t$  adalah nilai *input* saat ini dan  $S_{t-1}$  adalah keadaan tersembunyi dari nilai sebelumnya.  $W$  dan  $b$  adalah koefisien yang nilainya ditentukan dari proses training. Proses selanjutnya adalah di *input gate* yaitu mengontrol sejauh mana informasi baru akan ditambahkan ke *Memory cell*. Fungsi sigmoid juga digunakan untuk memutuskan seberapa banyak *input* baru yang harus ditulis ke dalam keadaan sel pada layer pertama (persamaan 2.3), dan fungsi tangen hiperbolik untuk layer kedua (persamaan 2.4)

$$i_t = \sigma(W_{ix}X_t + W_{is}S_{t-1} + b_i) \quad (2.3)$$

$$\tilde{c}_t = \tanh(W_{cx}X_t + W_{cs}S_{t-1} + b_c) \quad (2.4)$$

$$C_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \quad (2.5)$$

Nilai dari *cell state* didefnisikan dengan persamaan 2.5. *Cell state* adalah kombinasi dari memori lama  $C_{t-1}$ , yang dimodulasi dengan *forget gate*, dan informasi kandidat baru  $\tilde{c}_t$ , yang dimodulasi oleh gerbang *input*  $i_t$ . Kombinasi ini memungkinkan LSTM untuk secara selektif mengingat atau melupakan informasi, yang sangat penting untuk memodelkan ketergantungan jangka panjang dalam data sekuensial. Dan yang terakhir adalah *output gate*. *Output gate* menentukan bagian mana dari *cell state* yang harus dikeluarkan pada langkah waktu saat ini. *Output gate* ini membantu dalam menentukan *output* akhir berdasarkan *cell state* dan *input* ke unit LSTM. Perhitungan dari *output gate* dijelaskan dalam persamaan berikut,

$$o_t = \sigma(W_{ox}X_t + W_{os}S_{t-1} + b_o) \quad (2.6)$$

Persamaan 2.6 adalah persamaan untuk aktivasi *output gate* yang mana  $\sigma$  adalah fungsi sigmoid,  $W_{ox}$  dan  $W_{os}$  adalah matriks untuk input  $X_t$  dan *hidden state* sebelumnya  $S_{t-1}$  dan  $b_o$  bias untuk *output gate*. *Output gate* menggunakan fungsi sigmoid untuk menentukan berapa banyak *cell state*  $C_t$  yang mempengaruhi *hidden state* yang juga merupakan outputnya.

$$S_t = o_t \times \tanh(c_t) \quad (2.7)$$

Persamaan 2.7 adalah *hidden state* atau *output* LSTM pada suatu waktu. Kombinasi aktivasi sigmoid dan fungsi tangen hiperbolik pada persamaan 2.7 memastikan bahwa *output* diatur, memungkinkan LSTM untuk mengontrol pengaruh *input* saat ini dan *cell state* pada *output* (Ramadhan dkk., 2021)

## 2.8. Root Mean Square Error (RMSE)

Metrik digunakan untuk mengevaluasi performa model setelah proses pelatihan selesai untuk membantu mengukur sejauh mana model mampu melakukan generalisasi atau membuat prediksi yang akurat pada data baru yang belum pernah dilihat sebelumnya, serta memberikan ukuran efektivitas yang lebih mudah diinterpretasikan. Metrik juga digunakan untuk membandingkan berbagai model guna mengidentifikasi model mana yang memberikan hasil terbaik. *Root Mean Square Error* (RMSE) merupakan salah satu metrik yang sering digunakan. RMSE merupakan *Mean Square Error* (MSE) yang diakar, RMSE mengukur akar dari rata-rata selisih kuadrat antara nilai prediksi dan nilai sebenarnya. RMSE digunakan sebagai metrik untuk evaluasi akurasi prediksi. Adapun persamaan dari MSE ada pada persamaan 2.8 berikut,

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.9)$$

RMSE selalu bernilai positif karena selisih antara nilai prediksi dan nilai aktual dikuadratkan dan diakar.  $y_i$  merupakan nilai sebenarnya dan  $\hat{y}_i$  adalah nilai prediksi. Nilai 0 menunjukkan kecocokan sempurna, sementara nilai yang lebih besar menunjukkan perbedaan yang lebih tinggi antara prediksi dan nilai aktual. RMSE sensitif terhadap *outlier* dan model memprioritaskan pengurangan kesalahan

besar karena RMSE memberikan penekanan lebih pada kesalahan besar dibandingkan kesalahan kecil (Terven dkk., 2023)

## 2.9. Mean Absolute Error (MAE)

*Mean Absolute Error* (MAE) yang juga disebut *Mean Absolute Deviation* (MAD), sebuah metrik yang digunakan untuk mengukur sejauh mana hasil prediksi dari sebuah model berbeda dengan nilai sebenarnya. MAE memberikan gambaran tentang rata-rata besar kesalahan yang dihasilkan oleh model prediksi. Adapun persamaan MAE ada pada persamaan 2.10

$$MAE = \frac{1}{n} \sum_{t=1}^n |A_t - F_t| \quad (2.10)$$

$A_t$  dalam persamaan tersebut merupakan nilai prediksi dan  $F_t$  adalah data real. Sedangkan n adalah banyaknya data. Berbeda dengan MSE, yang menghitung kuadrat dari selisih nilai prediksi dan nilai aktual sehingga memberikan penalti lebih besar untuk kesalahan yang besar. MAE memperlakukan semua kesalahan secara setara. Pendekatan linier ini membuat MAE menjadi metrik yang lebih kokoh dan mudah dipahami, terutama untuk aplikasi di mana data memiliki nilai ekstrem (*outliers*) atau nilai yang sangat jauh dari rata-rata yang dapat memengaruhi hasil analisis.

Dalam konteks prediksi data *timeseries*, MAE digunakan sebagai salah satu metrik utama bersama dengan MSE untuk mengevaluasi kinerja model. Misalnya, MAE lebih disukai ketika tujuan utama adalah meminimalkan rata-rata keseluruhan besarnya kesalahan, daripada memberikan perhatian lebih pada kesalahan besar tertentu. Sementara, MSE cenderung lebih sensitif terhadap kesalahan besar yang jarang terjadi, MAE memberikan pandangan yang lebih seimbang yang mencerminkan besarnya kesalahan rata-rata yang lebih umum terjadi. MAE memiliki keunggulan khusus dalam situasi nyata karena mudah diinterpretasikan karena melaporkan rata-rata deviasi absolut dalam satuan yang sama dengan data aslinya. MAE juga memberikan wawasan yang dapat langsung digunakan untuk menilai akurasi model (Saigal S & Mehrotra D, 2012).

## 2.10. Python

Python adalah bahasa pemrograman tingkat tinggi yang serbaguna dan terkenal karena sederhana. Hal itu membuatnya mudah

diakses bagi pemula namun Python juga mampu melakukan tugas-tugas kompleks. Selain dari keunggulan bahasanya yang mudah, *tools* dan *library* Python yang tersedia membuatnya semakin diminati, terutama untuk digunakan dalam *data science*, *machine learning*, dan komputasi ilmiah. Pada tahun 2019, KDnuggets membuat pemilihan yang melibatkan lebih dari 1800 responden, menyoroti dominasi Python yang terus berlanjut sebagai bahasa paling populer dalam analitik, *data science*, dan *machine learning* (Raschka dkk., 2020).

## 2.11. TensorFlow

Tensorflow merupakan *library* perangkat lunak yang digunakan untuk komputasi numerik menggunakan *dataflow graph*. TensorFlow memudahkan pengguna untuk melatih neural network dan model *machine learning* yang lain untuk diproduksi. Algoritma inti TensorFlow ditulis sangat optimal dengan C++ dan CUDA (*Computer Unified Device Architecture*) platform komputasi paralel dan *Application Programming Interface* (API) yang dibuat oleh NVIDIA. TensorFlow memiliki API untuk berbagai bahasa pemrograman. API dari Python adalah yang paling stabil dan paling lengkap. Tensorflow memiliki API level rendah dan juga API level tinggi yaitu (Keras dan Estimators). Program TensorFlow umumnya terdiri dari dua sesi, yaitu fase konstruksi dan fase eksekusi. TensorFlow menyediakan konstruksi dasar seperti *layer* yang terkoneksi penuh, *convolutional layer*, modul *recurrent neural network*, dan fungsi aktivasi nonlinier. TensorFlow juga menyediakan berbagai *loss function* seperti *cross-entropy* dan *Mean Square Error* (MSE) (Pang dkk., 2020).

## 2.12. Keras

Keras adalah API *neural network* tingkat tinggi yang ditulis dalam Python. Keras dapat dijalankan di TensorFlow dan *framework* tingkat rendah yang lain. Keras didesain dengan mengutamakan untuk memungkinkan eksperimen cepat dan memungkinkan pembuatan *prototype* yang sederhana dan cepat. Berbagai bentuk komponen *neural network* seperti *thick layers*, *convolutional layers*, *recurrent layers*, *dropout layers* dan variasi lainnya dapat diterima oleh Keras. kode Keras secara otomatis mengelola penggunaan alat-alat pemrosesan seperti *Central Processing Unit* (CPU) dan *Graphics Processing Unit* (GPU) untuk memastikan kinerja yang optimal yang

berarti Keras dapat menyesuaikan penggunaan CPU dan GPU agar sesuai dengan kebutuhan komputasi saat melatih model *neural network* dan menjadikan proses pelatihan menjadi lebih efisien.

Keras sudah dilengkapi dengan implementasi dari berbagai fungsi yang diperlukan dalam pelatihan model *deep learning*, seperti fungsi aktivasi, pengoptimal (*optimizers*), rumus metrik, dan prosedur lainnya yang dibutuhkan untuk menangani sesi *training* dengan mudah. *Plugin deep learning* di Keras mengintegrasikan fungsionalitas dari *library-library* Keras, dan pada akhirnya menghubungkannya dengan fungsi-fungsi dari TensorFlow ke dalam Python yang berarti Keras memudahkan pengguna untuk memanfaatkan kemampuan TensorFlow dalam mengembangkan dan melatih model *deep learning* (Chicho & Sallow, 2021).

## **BAB III**

## **METODOLOGI**

### **3.1. Tempat dan Waktu Penelitian**

Penelitian ini dilaksanakan pada dua tempat yang berbeda yaitu di BMKG Stasiun Maritim Perak Kota Surabaya, dan kediaman penulis yang berada di Kecamatan Sukomanunggal Kota Surabaya..

### **3.2. Alat dan Bahan**

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Sebuah laptop HP dengan spesifikasi Processor Intel Core i5-6200 CPU @ 2,30 GHz, 2 Core(s), 4 Logical Processor(s), RAM DDR4 8 GB, SSD 256 GB.
2. Sebuah sistem operasi Windows 10 Pro 64-bit.
3. *Browser* Google Chrome untuk mengakses IDE Google Colaboratory secara *online*.
4. Google Drive untuk menyimpan dataset, mengakses dataset dan menyimpan hasil prediksi.

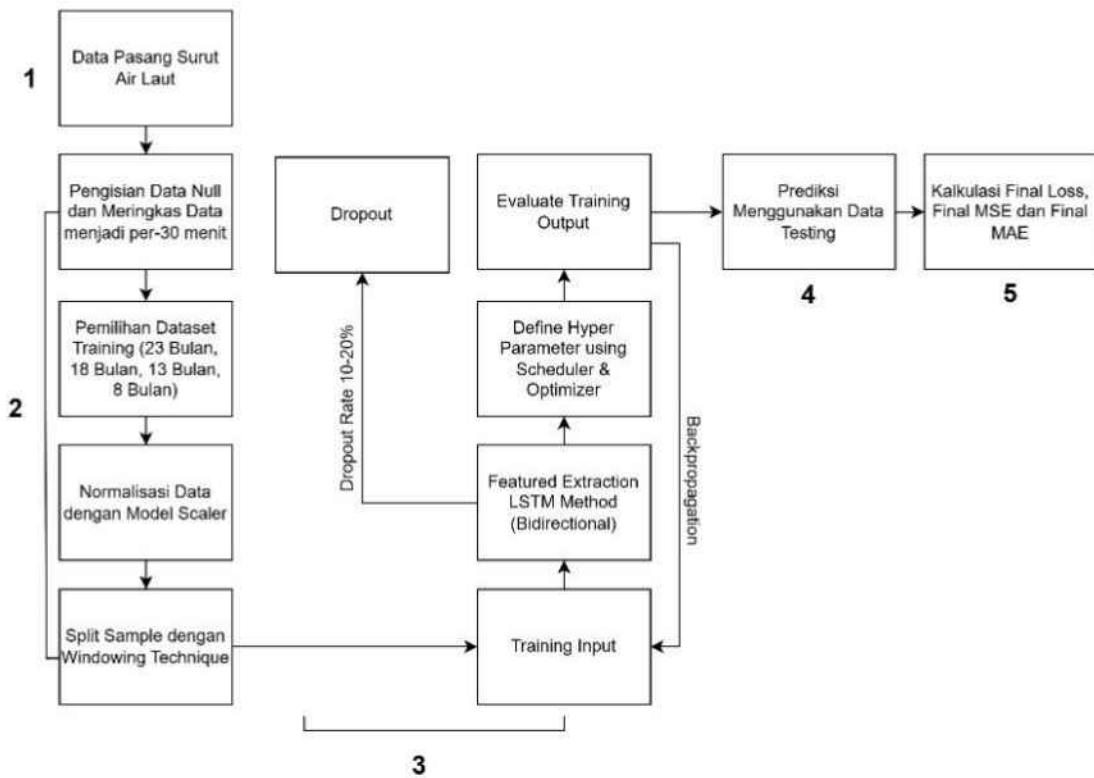
Bahan yang digunakan dalam penelitian ini adalah data pasang surut air laut yang didapatkan dari BMKG Stasiun Maritim Perak Kota Surabaya.

### **3.3. Tahapan Penelitian**

Penelitian ini memiliki tahapan sebagai berikut:

1. Pengambilan *dataset*
2. Penyiapan data
3. Pembuatan model *neural network* dan *training*
4. Pengujian *neural network*
5. Evaluasi model

Ilustrasi dari tahapan penelitian ini dapat dilihat pada Gambar 3.1 dan penjelasan masing-masing tahapan penelitian akan dijelaskan lebih lanjut dan mendetil pada setiap sub-subbab.



Gambar 3.1 Diagram Tahapan Penelitian

### 3.3.1 Pengambilan dataset

Data pasang surut air laut dari Stasiun Meteorologi Maritim Tanjung Perak Surabaya dapat diakses melalui website Sea Level Station Monitoring Facility milik Intergovernmental Oceanographic Commission (IOC) dari UNESCO. Data yang ditampilkan di *website* tersebut merupakan data pasang surut per menit dalam rentang 30 hari. Data yang diambil adalah data dengan total 28 bulan, yaitu dari Bulan Januari 2022 sampai Bulan April 2024. Data-data tersebut perlu di salin dan di tempel ke dalam Microsoft Excel. Data tersebut ditampilkan pada Gambar 3.2,

	Time (UTC)	bat(V)	pr2(m)	prs(m)	rad(m)
1	2022-01-01 00:15:00	12.7	1.1025	0.645	4.9675
2	2022-01-01 00:30:00	12.7	1.184	0.726	5.05
3	2022-01-01 00:45:00	12.7	1.264	0.806	5.123
4	2022-01-01 01:00:00	12.8	1.351	0.891	5.207
5	2022-01-01 01:15:00	13.3	1.427	0.968	5.288
...	...	...	...	...	...
61072	2024-04-30 23:00:00	12.1	1.797	1.54	5.618
61073	2024-04-30 23:15:00	12.2	1.811	1.555	5.632
61074	2024-04-30 23:30:00	12.4	1.825	1.575	5.65
61075	2024-04-30 23:45:00	12.5	1.834	1.583	5.656
61076	2024-05-01 00:00:00	12.7	1.866	1.615	5.688

61076 rows × 5 columns

Gambar 3.2 Data Pasang Surut Air Laut

Terdapat 3 sensor yaitu prs, pr2 dan rad. Dikarenakan sensor rad memiliki fluktuasi yang ekstrem dan memiliki banyak *outlier*, data sensor rad tidak digunakan dalam *training*.

### 3.3.2 Penyiapan data

Data yang didapatkan masih perlu diolah lagi karena data tersebut perlu dijadikan per-30 menit untuk mengurangi dimensi data, menghilangkan *noise*, meningkatkan generalisasi model, dan mempercepat *training*. Hal ini membuat model lebih efisien, relevan, dan stabil. Selain itu, terdapat beberapa data yang kosong yang perlu dilengkapi untuk menghindari *error*, meningkatkan akurasi model, menjaga konsistensi temporal, dan mengurangi *noise*. Gambar 3.3 adalah tabel data pasang surut air laut yang telah dilengkapi dan diringkas menjadi per-30 menit.

	Time (UTC)	bat(V)	pr2(m)	prs(m)	rad(m)
1	2022-01-01 00:30:00	12.7	1.184	0.726	5.05
2	2022-01-01 01:00:00	12.8	1.351	0.891	5.207
3	2022-01-01 01:30:00	13.4	1.501	1.041	5.358
4	2022-01-01 02:00:00	12.9	1.635	1.177	5.489
5	2022-01-01 02:30:00	13.1	1.719	1.261	5.576
...	...	...	...	...	...
30487	2024-04-30 22:00:00	12	1.702	1.461	5.536
30488	2024-04-30 22:30:00	12	1.759	1.517	5.593
30489	2024-04-30 23:00:00	12.1	1.797	1.54	5.618
30490	2024-04-30 23:30:00	12.4	1.825	1.575	5.65
30491	2024-05-01 00:00:00	12.7	1.866	1.615	5.688

30491 rows × 5 columns

Gambar 3.3 Data Pasang Surut Air Laut per-30 Menit

Setelah dataset sudah menjadi per-30 menit maka diupload dalam Google Drive

*Dataset* yang digunakan untuk *training* harus memiliki nilai sebanding dan untuk memastikan bahwa semua fitur *dataset* memiliki rentang nilai yang sebanding agar model dapat lebih mudah beradaptasi dan memberikan hasil yang lebih akurat. Maka diperlukan *model scaler* yaitu metode yang digunakan untuk mengubah skala atau rentang nilai dari fitur-fitur dalam dataset sebelum model dibangun. *Model scaler* yang digunakan adalah *StandardScaler*. Dalam tahap penyiapan data, digunakan pula Teknik *Windowing* yaitu teknik untuk memecah *dataset* menjadi beberapa *window* dengan panjang tertentu. Setiap *window* merupakan subset dari *dataset* yang akan digunakan untuk melatih model.

### 3.3.3 Pembuatan model *neural network* dan *training*

Model yang dibuat ini menggunakan layer *bidirectional LSTM* dengan 128 *neuron* dan 256 *neuron* yang memproses data sekuensial dari dua arah, baik maju maupun mundur. Lalu

digunakan juga *layer flatten* meratakan output dari LSTM menjadi vektor satu dimensi. Hal ini diperlukan untuk mengonversinya ke bentuk yang dapat diterima oleh *layer dense*. Berikutnya, *layer dense* dengan 256 *neuron* dan fungsi aktivasi *Swish*. Lalu *layer dropout* digunakan untuk mencegah *overfitting* dengan mematikan secara acak 10%-20% *neuron* selama *training*. Dan *layer output* diatur untuk memprediksi 2 target. Selain layer *bidirectional LSTM*.

*Hyperparameter* digunakan untuk mengendalikan proses pembelajaran model. *Hyperparameter* yang digunakan adalah *scheduler* dan *optimizer*. Fungsi *scheduler* digunakan untuk mengubah *learning rate* seiring bertambahnya *epoch* untuk mencegah *overshooting gradient*. Optimizer yang digunakan dalam model ini adalah *Adam optimizer*, yang merupakan optimasi berbasis gradien membantu model konvergen lebih cepat dan efisien.

Model LSTM dengan *Huber Loss*, *Swish*, RMSE, MAE dan *L1 regularizer* bekerja dalam proses berulang yang bertujuan untuk mengoptimalkan kinerja model dalam memahami pola data sekuensial. Proses ini dimulai dengan data *input* yang diproses melalui *layer LSTM*. *Layer* tersebut menggunakan mekanisme *cell state* dan *gates* untuk mengingat atau melupakan informasi penting, sehingga memungkinkan model mempelajari hubungan jangka panjang dalam data. Fungsi aktivasi *Swish* diterapkan untuk memastikan aliran gradien yang halus, meningkatkan efisiensi *training*, dan membantu model menangkap pola *non-linear*.

Setelah LSTM memproses data, model membuat prediksi, dan perbedaan antara prediksi dan nilai aktual diukur menggunakan *Huber Loss*. *Huber Loss* mengombinasikan keunggulan MSE dan MAE yang memberikan penalti lebih besar untuk *error* besar (linier seperti MAE) dan penalti yang lebih kecil untuk *error* kecil (kuadrat seperti MSE). Hal tersebut membantu menjaga stabilitas *training* terutama saat menghadapi *outlier*.

Proses *backpropagation* dimulai setelah *loss* dihitung. Selama *backpropagation*, gradien dari fungsi *loss* dihitung

terhadap setiap bobot dalam jaringan. Gradien ini menunjukkan arah dan seberapa besar perubahan yang diperlukan untuk memperbarui bobot guna meminimalkan *loss*. Dalam model ini, pembaruan bobot dilakukan dengan algoritma *Adam Optimizer* yang memanfaatkan *adaptive learning rate* dan momentum. *Adam Optimizer* menggabungkan rata-rata gradien yang mempercepat *training* dan meningkatkan stabilitas serta momentum yaitu mengurangi osilasi gradien yang memungkinkan pembaruan bobot yang lebih efisien.

Selain itu, *L1 regularizer* bekerja dengan menambahkan penalti proporsional terhadap bobot besar, yang menyebabkan beberapa bobot mendekati atau menjadi nol. Selama *backpropagation*, gradien dari penalti L1 juga dihitung, dan bobot diperbarui dengan memperhitungkan penalti ini. *L1 regularizer* membantu menyederhanakan model dengan membuat bobot lebih jarang (*sparse*). Hal tersebut dapat meningkatkan generalisasi dan mengurangi risiko *overfitting*.

Secara keseluruhan, selama proses *backpropagation*, gradien dari *Huber Loss* dan penalti L1 digunakan untuk memperbarui bobot dengan algoritma *Adam*, mengarahkan model untuk secara iteratif meminimalkan *loss* dan meningkatkan akurasi prediksi.

### 3.3.4 Pengujian *neural network*

Pelatihan menggunakan data *training* yang merupakan data masukan untuk melatih model dan pengujian menggunakan *data testing* yang merupakan data yang nantinya tidak akan dilihat oleh model dan data tersebut merupakan data yang akan diprediksi oleh model. *Data testing* digunakan untuk mengevaluasi seberapa baik model dapat memprediksi atau menggeneralisasi pada data baru karena hasil prediksi akan dibandingkan dengan hasil data asli yang ada di *data testing*.

Pada tahap pengujian model *neural network*, dilakukan dua jenis evaluasi yaitu dengan variasi *data training* untuk memprediksi 5 bulan kedepan dan variasi *data testing* dengan *data training* 8 bulan untuk memprediksi beberapa bulan kedepan. Evaluasi dengan variasi *data training* ditunjukkan di

Tabel 3.1 dan evaluasi dengan variasi *data testing* dapat dilihat dalam Tabel 3.2.

Tabel 3.1 Variasi *Data Training*

<b>Data Training (Rentang Input)</b>	<b>Data Testing (Rentang Prediksi)</b>
23 Bulan Sebelumnya	5 Bulan Kedepan
18 Bulan Sebelumnya	5 Bulan Kedepan
13 Bulan Sebelumnya	5 Bulan Kedepan
8 Bulan Sebelumnya	5 Bulan Kedepan

Tabel 3.2 Variasi *Data Testing*

<b>Data Training (Rentang Input)</b>	<b>Data Testing (Rentang Prediksi Setelah Data Training)</b>
8 Bulan Sebelumnya	5 Bulan Pertama
8 Bulan Sebelumnya	5 Bulan Kedua
8 Bulan Sebelumnya	5 Bulan Ketiga
8 Bulan Sebelumnya	5 Bulan Keempat

Hasil evaluasi dari pembagian tersebut nantinya dapat menjawab dua tujuan dari penelitian ini. Pembagian tersebut nantinya akan diketahui semakin banyak *data training* yang digunakan akankah menunjukkan bahwa model menjadi semakin baik. Selain itu, juga dapat diketahui bahwa semakin jauh data yang diprediksi akankah membuat kinerja model tersebut semakin menurun.

### 3.3.5 Evaluasi model

Model ini kemudian dievaluasi menggunakan RMSE dan MAE yang memberikan ukuran seberapa baik model memprediksi data baru setelah proses pengujian. RMSE dipilih karena sering digunakan dalam optimasi model terutama dalam *neural network*, karena sifatnya yang mendukung proses *backpropagation* dan sensitivitasnya terhadap kesalahan besar. Sementara itu, MAE dipilih juga sebagai interpretasi langsung dalam satuan data yang sama dan outlier tidak menjadi fokus

utama. Setelah nilai RMSE dan MAE diketahui, dari nilai itulah dapat diketahui model mana yang memiliki performa terbaik.

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1. Perbandingan Kinerja RNN-LSTM dan CNN**

Penelitian ini membandingkan performa model RNN-LSTM dan CNN dalam memprediksi pasang surut air laut menggunakan dua metrik utama, yaitu *Root Mean Square Error* (RMSE) dan *Mean Absolute Error* (MAE). RMSE digunakan untuk mengukurakar dari rata-rata kuadrat kesalahan antara nilai prediksi dan nilai aktual, dengan memberikan penalti yang lebih besar untuk kesalahan yang besar. RMSE digunakan sebagai metrik utama karena RMSE sangat sensitif terhadap nilai *outlier* dalam data prediksi. Sementara itu, MAE mengukur rata-rata kesalahan prediksi secara langsung dalam satuan yang sama dengan data aktual, sehingga memberikan gambaran yang lebih intuitif mengenai tingkat kesalahan prediksi. Pada Tabel 4.1 adalah hasil performa dari model dalam skenario terbaik, yaitu dengan data training 23 bulan dan data testing 5 bulan.

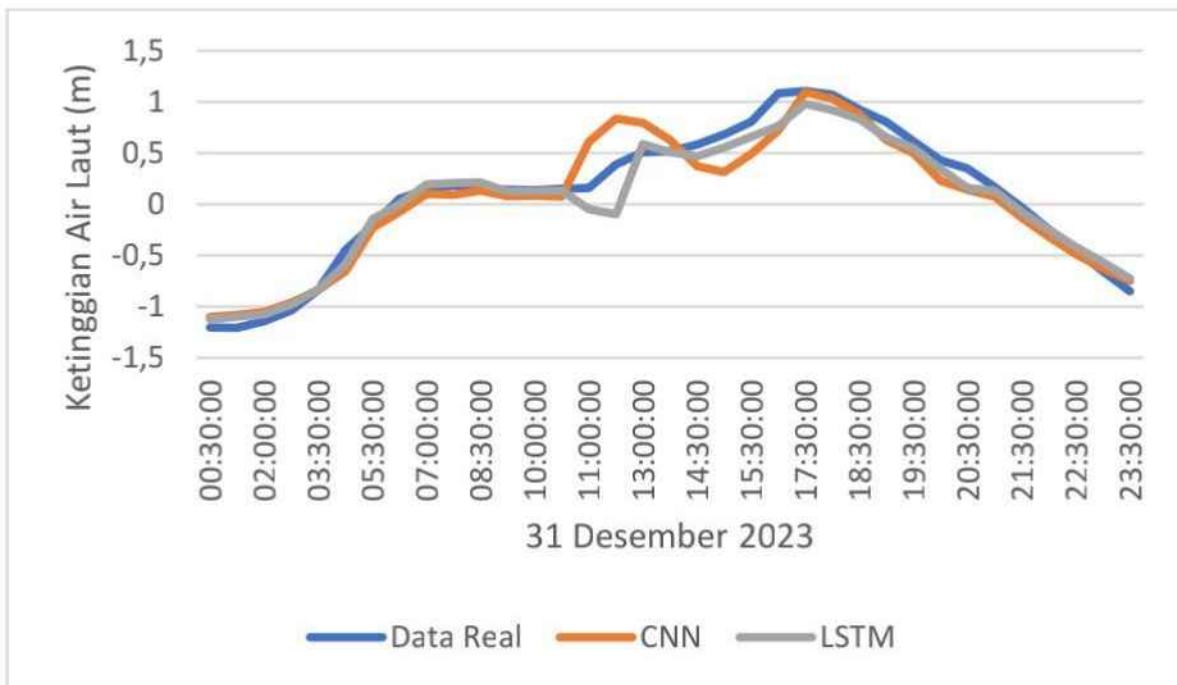
Tabel 4.1 Perbandingan Nilai Metrik antara RNN-LSTM dan CNN

Model	RMSE (m)	MAE (m)
RNN-LSTM	0,224	0,120
CNN	0,249	0,153

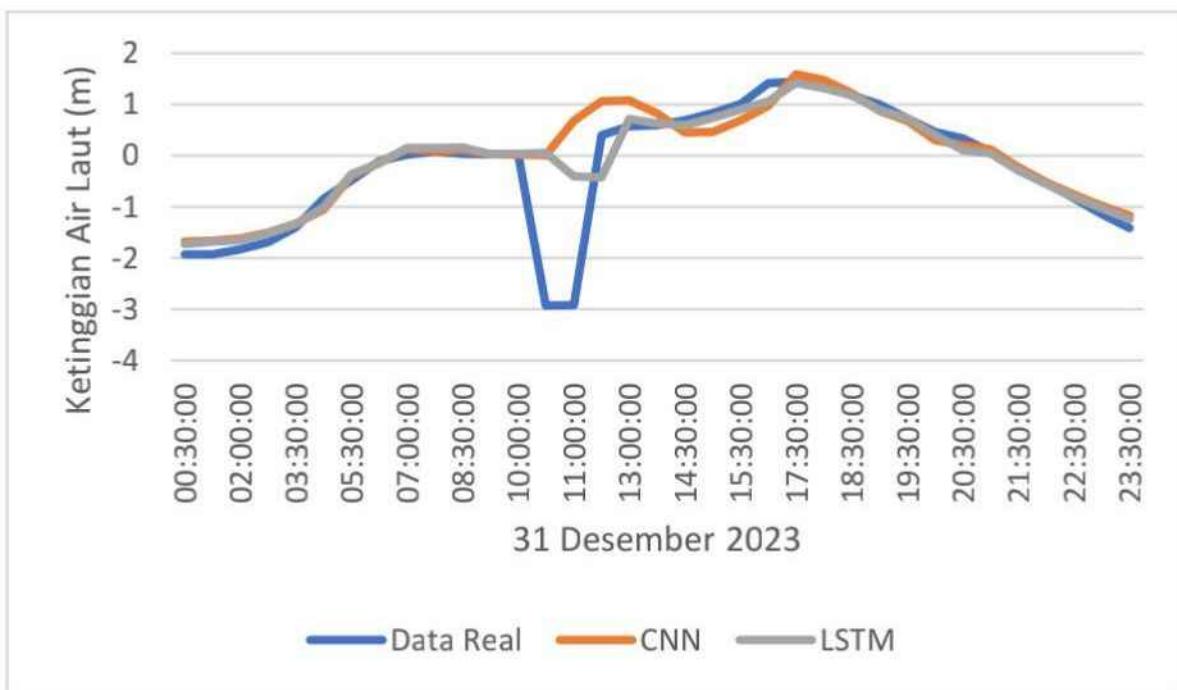
Hasil evaluasi kinerja kedua model menunjukkan bahwa RNN-LSTM memiliki nilai RMSE sebesar 0,224 m yang lebih rendah dibandingkan CNN dengan nilai MSE sebesar 0,249 m. Hal ini menunjukkan bahwa RNN-LSTM lebih mampu meminimalkan kesalahan prediksi secara keseluruhan terutama untuk kesalahan besar yang dapat memengaruhi keakuratan model dalam menangkap pola pasang surut.

Dari segi MAE, RNN-LSTM memiliki nilai sebesar 0,120 m yang berarti rata-rata kesalahan prediksi model ini adalah 12 cm. Sebaliknya, CNN memiliki nilai MAE sebesar 0,153 cm, atau rata-rata kesalahan prediksi 15,3 cm.

Selain perbandingan nilai RMSE dan MAE, lebih jelasnya dapat dilihat visualisasi perbandingan antara hasil prediksi CNN, LSTM dan data real dalam satu hari pada gambar berikut,



Gambar 4.1 Perbandingan antara Data Real, CNN, dan RNN-LSTM untuk Sensor prs



Gambar 4.2 Perbandingan antara Data Real, CNN, dan RNN-LSTM untuk Sensor pr2

Kedua grafik dari Sensor prs dan Sensor pr2 menunjukkan pola pasang surut air laut pada tanggal 31 Desember 2023. Berdasarkan grafik dalam satu hari memiliki pasang dua kali dan satu kali surut dengan amplitudo yang berbeda. Pola pasang surut di Surabaya

menunjukkan *Mixed Semi-Diurnal Tide* (Pasang Surut Campuran condong ke Harian Ganda). Pada grafik Sensor prs, puncak pasang pertama dan kedua menunjukkan perbedaan amplitudo, begitu pula dengan kedalaman surutnya. Pola serupa juga terlihat pada grafik Sensor pr2, di mana perbedaan signifikan terjadi antara puncak pasang pertama dan kedua. Grafik menunjukkan hasil model prediksi RNN-LSTM dan CNN dibandingkan dengan data asli. Pada Sensor prs, model CNN dan RNN-LSTM mampu mengikuti pola pasang surut dengan cukup baik, walau terdapat deviasi kecil di beberapa titik, khususnya pada puncak pasang pertama. Model RNN-LSTM tampak lebih stabil dalam mendekati data real dibandingkan CNN, yang memiliki fluktuasi lebih tajam di beberapa bagian. Sebaliknya, pada Sensor pr2, prediksi kedua model mengalami deviasi lebih besar pada surut ekstrem sekitar pukul 11:00. CNN menunjukkan hasil prediksi yang cenderung lebih tinggi dari data asli, sedangkan LSTM menunjukkan stabilitas yang lebih baik. Pada grafik Sensor prs, pola pasang surut ditangkap dengan baik oleh kedua model. Model LSTM menunjukkan hasil prediksi yang lebih mendekati data real dibandingkan CNN terutama pada fase transisi dari pasang ke surut. Prediksi CNN cenderung memberikan hasil yang sedikit overestimasi pada puncak pasang antara pukul 14:30 hingga 18:00. Di sisi lain, LSTM mampu mencerminkan tren data real dengan lebih akurat. Hal ini menunjukkan keunggulannya dalam menangkap pola temporal yang kompleks dalam data pasang surut.

Secara keseluruhan, model LSTM lebih unggul dibandingkan CNN dalam memprediksi ketinggian air laut karena memiliki kemampuan yang lebih akurat dalam mencerminkan pola temporal dalam data. Faktor yang menjadikan RNN-LSTM lebih unggul dibandingkan CNN dalam memprediksi pasang surut air laut adalah karena arsitekturnya dirancang khusus untuk menangani data sekuensial. RNN-LSTM memiliki mekanisme *gate* (*input*, *forget*, dan *output*) yang memungkinkan model untuk menyimpan informasi penting dari langkah waktu sebelumnya dan menangkap pola temporal jangka panjang. Hal ini sangat cocok untuk data pasang surut yang memiliki pola waktu yang berulang, seperti siklus harian atau bulanan. Sebaliknya, CNN lebih fokus pada analisis pola spasial dan kurang efektif dalam memahami hubungan antar waktu. Sehingga, CNN cenderung memberikan prediksi yang kurang akurat

dibandingkan RNN-LSTM pada data dengan pola temporal kompleks. Fleksibilitas RNN-LSTM dalam mempelajari pola jangka panjang menjadikannya lebih baik untuk aplikasi prediksi pasang surut air laut CNN.

#### **4.2. Evaluasi Kinerja RNN-LSTM dengan Variasi Rentang Data Input**

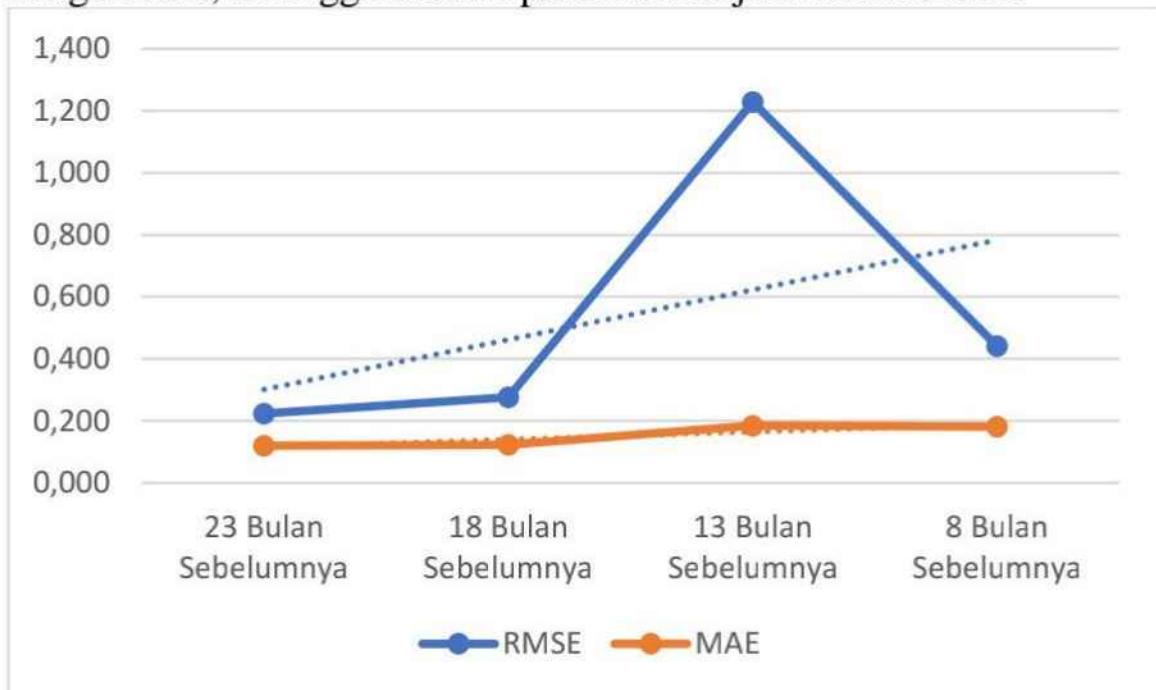
Selain membandingkan performa RNN-LSTM dan CNN pada skenario terbaik, penelitian ini juga mengevaluasi pengaruh variasi rentang data pelatihan terhadap kinerja model RNN-LSTM. Rentang data pelatihan yang lebih panjang memungkinkan model untuk mempelajari pola temporal yang lebih lengkap, sehingga meningkatkan akurasi prediksi. Variasi data pelatihan yang digunakan dirangkum pada Tabel 4.2.

Tabel 4.2 Performa RNN-LSTM untuk Prediksi 5 Bulan Kedepan

Rentang <i>Data Training</i>	RMSE (m)	MAE (m)
23 Bulan Sebelumnya	0,224	0,120
18 Bulan Sebelumnya	0,276	0,123
13 Bulan Sebelumnya	1,228	0,185
8 Bulan Sebelumnya	0,442	0,181

Hasil analisis variasi rentang *data training* pada model prediksi pasang surut air laut menunjukkan bahwa performa terbaik dicapai ketika menggunakan data *training* selama 23 bulan sebelumnya, dengan nilai RMSE sebesar 0,224 m dan MAE 0,120 m. Hal ini mengindikasikan bahwa semakin banyak data input yang digunakan untuk pelatihan, semakin baik kemampuan model dalam menangkap pola pasang surut air laut yang kompleks dan berulang, sehingga menghasilkan prediksi yang lebih akurat. Ketika rentang data dikurangi menjadi 18 bulan, performa model sedikit menurun dengan RMSE naik menjadi sebesar 0,276 m dan MAE 0,123 m. Tetapi, saat rentang *data training* hanya 13 bulan, kesalahan meningkat signifikan dengan RMSE mencapai 1,228 m dan MAE 0,185 m. Hal ini dapat dikaitkan dengan anomali pada data pasang surut sensor prs pada tanggal 17 Mei 2023. Pada tanggal tersebut terjadi lonjakan nilai yang tidak wajar. Lonjakan ini kemungkinan disebabkan oleh fluktuasi ekstrem pada kondisi lingkungan, seperti perubahan tekanan atmosfer mendadak, aktivitas manusia di sekitar lokasi sensor, atau kerusakan

sensor akibat kelembapan atau kotoran. Untuk rentang data 8 bulan sebelumnya, RMSE menurun menjadi 0,442 m dengan MAE 0,181 m. Tetapi model tetap tidak mampu menangkap pola jangka panjang dengan baik, sehingga akurasi prediksi menjadi lebih rendah.



Gambar 4.3 Grafik Tren Nilai Metrik Variasi *Data Input*

Secara keseluruhan, semakin banyak *data input* yang digunakan, semakin baik performa model dalam memprediksi pasang surut air laut. Sebaliknya, semakin sedikit *data input* yang digunakan semakin buruk performanya. Hal tersebut dapat dilihat pada Gambar 4.3. Garis titik-titik pada grafik menunjukkan tren garis linear atau regresi linier yang menggambarkan pola perubahan nilai RMSE dan MAE secara keseluruhan. Kedua tren dari nilai RMSE dan MAE mengalami kenaikan dengan semakin sedikitnya *data input*.

Rentang *data training* 23 bulan memberikan hasil terbaik karena mencakup informasi yang cukup untuk menangkap pola pasang surut jangka panjang. Pengurangan durasi *data training*, meningkatkan risiko kehilangan informasi penting karena model menjadi kekurangan konteks. Selain itu, kualitas data sangat penting, dan anomali seperti lonjakan nilai pada sensor perlu diidentifikasi dan diatasi untuk memastikan akurasi model tetap optimal.

### **4.3. Evaluasi Kinerja RNN-LSTM untuk Prediksi Beberapa Bulan Kedepan**

Penelitian ini juga mengevaluasi kemampuan model RNN-LSTM untuk memprediksi pasang surut air laut pada beberapa interval waktu yang lebih panjang. Prediksi dilakukan untuk empat interval 5 bulan pertama hingga keempat dengan *data training* tetap yaitu selama 8 bulan sebelumnya. Hasil performa prediksi beberapa bulan kedepan dirangkum pada Tabel 4.3.

Tabel 4.3 Performa RNN-LSTM Prediksi Beberapa Bulan Kedepan

Rentang Prediksi	RMSE (m)	MAE (m)
5 Bulan Pertama	0,442	0,181
5 Bulan Kedua	1,222	0,170
5 Bulan Ketiga	0,811	0,132
5 Bulan Keempat	0,302	0,196

Hasil analisis kemampuan model RNN-LSTM dalam memprediksi pasang surut air laut untuk beberapa interval waktu menunjukkan variasi performa yang signifikan, tergantung pada kualitas data dan panjang interval prediksi. Pada 5 bulan pertama, model menunjukkan nilai RMSE 0,442 m dan MAE 0,181 m, menunjukkan kemampuan model untuk menangkap pola awal dengan *data training* selama 8 bulan. Namun, pada 5 bulan kedua, terjadi peningkatan kesalahan yang signifikan dengan RMSE 1,222 m meskipun MAE tetap stabil pada 0,170 m. Lonjakan kesalahan ini dapat dikaitkan dengan anomali pada data pasang surut dari sensor prs pada tanggal 17 Mei 2023, yang menunjukkan nilai fluktuatif dan tidak wajar. Anomali ini kemungkinan besar disebabkan oleh faktor lingkungan seperti perubahan ekstrem tekanan atmosfer, aktivitas manusia, atau gangguan teknis pada sensor.

Pada 5 bulan ketiga, nilai RMSE menurun menjadi 0,811 m dan MAE 0,132 m, menunjukkan peningkatan performa ketika data lebih konsisten dan stabil. Menariknya, pada 5 bulan keempat, nilai RMSE turun drastis menjadi 0,302 m, meskipun MAE sedikit meningkat menjadi 0,196 m. Fenomena ini menunjukkan bahwa model mampu mengenali pola jangka panjang dengan lebih baik, asalkan data tidak terganggu oleh anomali.



Gambar 4.4 Grafik Tren Nilai Metrik Variasi *Data Output*

Garis titik-titik pada grafik menunjukkan tren garis linear atau regresi linier yang menggambarkan pola perubahan nilai RMSE dan MAE secara keseluruhan. Pada grafik tersebut, terlihat bahwa tren RMSE mengalami penurunan bertahap seiring waktu. Meskipun terjadi lonjakan tajam pada 5 Bulan Kedua, dengan nilai RMSE mencapai 1,222, tren keseluruhan mengarah turun hingga periode 5 Bulan Keempat dengan nilai 0,302. Hal ini menunjukkan adanya peningkatan akurasi model secara bertahap, karena kesalahan prediksi semakin berkurang di periode-periode selanjutnya.

Sementara itu, MAE menunjukkan tren yang lebih stabil dengan sedikit kenaikan. Nilai MAE cenderung mendatar dan tidak mengalami fluktuasi signifikan seperti RMSE. Dengan nilai yang relatif kecil dan konsisten, perbedaan rata-rata absolut antara prediksi dan nilai aktual tetap terjaga rendah sepanjang periode analisis. Secara keseluruhan, garis tren ini mengindikasikan bahwa performa model semakin baik dari waktu ke waktu meskipun di akhir terdapat kenaikan.

Secara teoritis, semakin panjang waktu yang diprediksi, nilai error seharusnya semakin tinggi karena akumulasi ketidakpastian dalam pola data. Tetapi, dalam analisis ini hasil pada 5 bulan ketiga dan keempat justru menunjukkan penurunan *error*. Hal ini

kemungkinan besar disebabkan oleh stabilitas data pada interval tersebut yang membantu model mempelajari pola secara lebih konsisten. Sebaliknya, kesalahan besar pada 5 bulan kedua mengindikasikan bahwa kualitas data, terutama anomali pada sensor, memiliki dampak yang lebih signifikan dibandingkan panjang interval prediksi itu sendiri. Tetapi hal ini juga bisa terjadi dikarenakan model belum memprediksi waktu yang lebih jauh lagi untuk terjadinya penurunan performa.

Hasil dari penelitian ini menunjukkan bahwa model RNN-LSTM memiliki kemampuan prediksi yang baik, tetapi performanya sangat bergantung pada kualitas data input. Meskipun prediksi untuk interval waktu yang lebih panjang biasanya menghasilkan error yang lebih besar, stabilitas data dapat membantu menurunkan error. Oleh karena itu, deteksi dan mitigasi anomali pada data pasang surut menjadi langkah penting untuk meningkatkan akurasi prediksi model, terutama untuk interval prediksi yang lebih panjang.

#### **4.4. Evaluasi Perbandingan RMSE dan MAE**

Dalam evaluasi model prediksi, penggunaan metrik MAE dan RMSE sering kali menjadi standar untuk mengukur akurasi prediksi. Kedua metrik ini memiliki keunggulan masing-masing. Metrik MAE menggambarkan rata-rata kesalahan absolut, sedangkan RMSE lebih sensitif terhadap kesalahan besar atau *outlier*. Oleh karena itu, membandingkan kedua metrik ini memberikan wawasan yang lebih komprehensif mengenai kemampuan model dalam menangkap pola prediksi.

Nilai dari RMSE selalu lebih besar dari nilai MAE. Hal tersebut dapat dilihat pada Tabel 4.1. Selain dari tabel tersebut, Gambar 4.3 dan Gambar 4.4 memberikan informasi dengan jelas bahwa RMSE nilainya lebih besar daripada nilai MAE. Terlihat pada gambar grafik tersebut nilai RMSE yang ditandai dengan garis warna biru selalu diatas nilai MAE yang ditandai dengan warna jingga. MAE mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual. Metrik ini lebih intuitif dan tidak memberikan bobot tambahan pada kesalahan besar. Sebaliknya, RMSE memberikan bobot yang lebih besar pada kesalahan besar karena menghitung akar dari rata-rata kuadrat kesalahan. Dalam konteks penelitian ini, MAE mencerminkan seberapa jauh prediksi secara umum dari nilai aktual, sedangkan

RMSE mengindikasikan sensitivitas model terhadap kesalahan besar. Hal yang menjadikan nilai RMSE lebih besar dapat dilihat pada rumus matematisnya. Secara matematis, persamaan MAE dapat dilihat pada persamaan 2.10. dan RMSE dapat dilihat di persamaan 2.9. Perbedaan utama terletak pada kuadrat kesalahan dalam RMSE, yang memberikan bobot lebih besar pada kesalahan besar. Selain itu, nilai RMSE juga lebih besar karena proses perhitungan melibatkan akar kuadrat rata-rata dari jumlah kesalahan kuadrat, yang memperbesar dampak kesalahan individu jika dibandingkan dengan rata-rata absolut dalam MAE. Selain itu, n yang merupakan pembagi dalam RMSE ikut diakar sedangkan pada MAE tidak sehingga jelas pembaginya berbeda dan menyebabkan nilai RMSE memiliki nilai yang lebih besar daripada MAE.

#### **4.5. Evaluasi Pengaruh Kesalahan terhadap Sektor Kelautan**

Prediksi pasang surut merupakan aspek penting dalam berbagai kegiatan sektor kelautan, seperti deteksi potensi banjir rob dan pengelolaan aktivitas pelabuhan. Akurasi prediksi ketinggian pasang surut sangat memengaruhi keputusan operasional dan mitigasi risiko sehingga hasil prediksi juga perlu dievaluasi. Nilai MAE dalam prediksi menggunakan Model RNN-LSTM memiliki nilai dalam rentang 12-19 cm. Berdasarkan wawancara dengan seorang prakirawan BMKG Maritim Tanjung Perak, Surabaya, (Setiawan, wawancara pribadi, 2 Desember 2024) mengatakan bahwa kesalahan prediksi dalam rentang 12-19 cm dapat berdampak signifikan terhadap sektor kelautan. Kesalahan pun juga dilihat apakah nilai prediksi lebih besar dengan nilai yang asli atau sebaliknya. Hal tersebut disebut dengan kesalahan *underestimate* dan *overestimate*.

Kesalahan *underestimate* merupakan prediksi ketinggian lebih rendah dari kenyataan. Dalam kasus banjir rob hal tersebut menyebabkan banjir rob tidak terdeteksi. Misalnya, jika elevasi pesisir adalah 1,5 meter dari Mean Sea Level (MSL) dan prediksi ketinggian pasang hanya 1,4 meter, maka risiko banjir rob yang sebenarnya terjadi tidak akan terdeteksi oleh model prediksi. Akibatnya, infrastruktur pesisir dan fasilitas logistik menjadi rentan terhadap kerusakan tanpa adanya persiapan yang memadai. Sebaliknya, kesalahan *overestimate* yang merupakan prediksi ketinggian lebih tinggi dari kenyataan dapat menyebabkan alarm palsu. Kondisi ini

mengarah pada keputusan mitigasi yang tidak perlu, seperti penghentian aktivitas bongkar muat kapal yang berpotensi mengganggu operasional pelabuhan dan menyebabkan kerugian ekonomi.

Akurasi dari prediksi dikategorikan ke dalam tiga istilah utama: *Hit*, yaitu prediksi yang sesuai dengan kenyataan. *Miss*, yaitu prediksi yang lebih rendah dari kenyataan sehingga risiko tidak terdeteksi. dan *False Alarm*, yaitu prediksi yang lebih tinggi dari kenyataan yang mengindikasikan risiko yang sebenarnya tidak ada.

Sektor kelautan tidak semuanya memerlukan tingkat akurasi yang sama dalam prediksi ketinggian. Sebagai contoh, dalam operasi pelabuhan seperti bongkar muat kapal, yang lebih penting adalah akurasi waktu pasang surut daripada ketinggian spesifik. Namun, untuk mitigasi risiko banjir rob, prediksi ketinggian yang detail menjadi hal yang sangat krusial.

Dengan begitu, kesalahan prediksi ketinggian pasang surut sebesar 12-19 cm, baik *overestimate* maupun *underestimate* dapat memberikan dampak serius pada sektor kelautan, terutama dalam konteks mitigasi risiko banjir rob dan efisiensi operasional. Hal ini menekankan pentingnya pengembangan model prediksi yang lebih akurat dan perlu disesuaikan dengan kebutuhan spesifik pengguna data kelautan.

## **BAB V**

### **PENUTUP**

#### **5.1. Kesimpulan**

Penelitian ini telah berhasil membandingkan kemampuan prediksi antara model *Recurrent Neural Network-Long Short-Term Memory* (RNN-LSTM) dan *Convolutional Neural Network* (CNN) dalam memprediksi pasang surut air laut. Berdasarkan analisis yang dilakukan, kesimpulan yang dapat ditarik adalah:

1. Hasil evaluasi menggunakan RMSE dan MAE, model RNN-LSTM menunjukkan performa yang lebih baik dalam memprediksi pasang surut air laut dengan nilai RMSE 0,224 m dan nilai MAE 0,120 m dibandingkan dengan CNN yang memiliki nilai RMSE 0,249 m dan nilai MAE 0,153.
2. Penggunaan *input* data *training* dengan rentang data yang lebih panjang memberikan hasil yang lebih akurat dalam prediksi RNN-LSTM.
3. RNN-LSTM memiliki kemampuan yang cukup baik dalam memprediksi pasang surut air laut untuk beberapa bulan mendatang, dengan akurasi yang cenderung meningkat pada periode prediksi yang lebih panjang, asalkan data bebas dari anomali.

#### **5.2. Saran**

Saran Berdasarkan hasil penelitian ini, beberapa saran dapat diberikan untuk pengembangan lebih lanjut, baik dalam konteks penelitian maupun aplikasi praktis:

1. Memperluas rentang data input dengan memasukkan variabel-variabel eksternal yang dapat mempengaruhi pasang surut, seperti faktor cuaca, suhu permukaan laut, dan tekanan atmosfer. Dengan memasukkan lebih banyak variabel, diharapkan dapat meningkatkan ketepatan model dalam memprediksi kondisi pasang surut yang lebih kompleks.
2. Eksplorasi penggunaan arsitektur RNN-LSTM yang lebih kompleks atau mengkombinasikan RNN-LSTM dengan teknik lain seperti Attention Mechanism untuk meningkatkan kemampuan model dalam menangani data deret waktu yang panjang dan berfluktuasi.

3. Memasukkan informasi terkait posisi benda langit (seperti fase bulan, jarak bulan, dan posisi matahari) ke dalam model prediksi. Penambahan variabel ini dapat meningkatkan akurasi model RNN-LSTM dalam memprediksi pasang surut, khususnya untuk prediksi jangka panjang yang membutuhkan pemahaman lebih mendalam tentang pola gravitasi bumi-bulan-matahari.

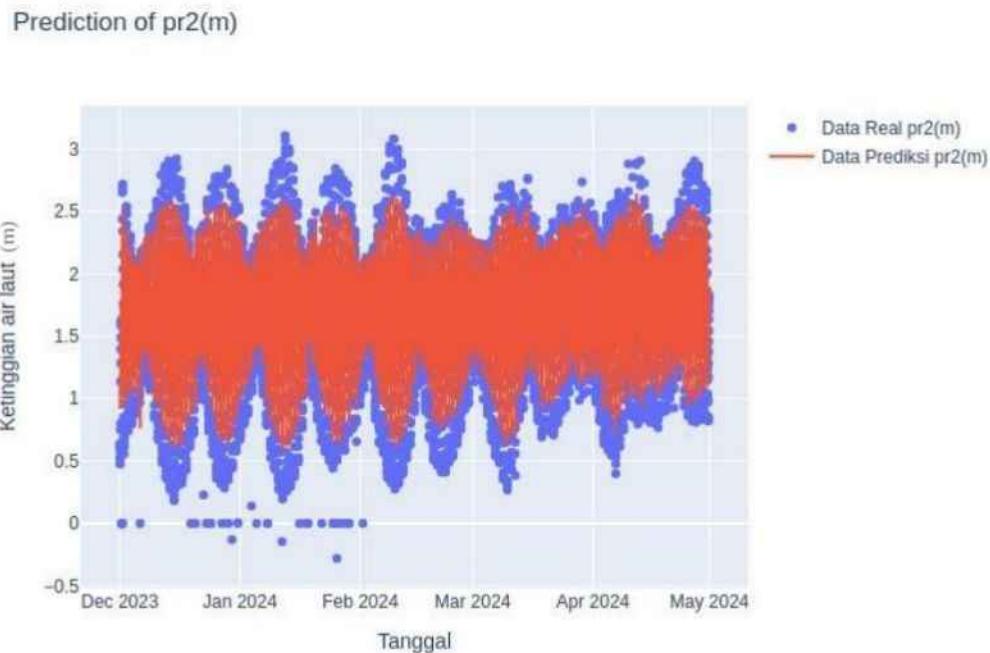
## DAFTAR PUSTAKA

- Ahmad, H. (2019). Machine Learning Application in Oceanography. *Aquatic Research*, 161–169. <https://doi.org/10.3153/ar19014>
- Ban, W., Shen, L., Lu, F., Liu, X., & Pan, Y. (2023). Research on Long-Term Tidal-Height-Prediction-Based Decomposition Algorithms and Machine Learning Models. *Remote Sensing*, 15(12), 3045. <https://doi.org/10.3390/rs15123045>
- Chicho, B. T., & Sallow, A. B. (2021). A Comprehensive Survey of Deep Learning Models Based on Keras Framework. *Journal of Soft Computing and Data Mining*, 2(2), 49–62. <https://doi.org/10.30880/jscdm.2021.02.02.005>
- Hassani, H., Silva, E. S., Unger, S., TajMazinani, M., & Mac Feely, S. (2020). Artificial Intelligence (AI) or Intelligence Augmentation (IA): What Is the Future? *AI (Switzerland)*, 1(2). <https://doi.org/10.3390/ai1020008>
- Luaran Nosius and Alfred, R. and O. J. H. and O. C. K. (2021). A Review on Deep Learning Approaches to Forecasting the Changes of Sea Level. In H. and H. H. and A. P. Alfred Rayner and Iida (Ed.), *Computational Science and Technology* (pp. 563–573). Springer Singapore.
- Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. In *Journal of Educational and Behavioral Statistics* (Vol. 45, Issue 2, pp. 227–248). SAGE Publications Inc. <https://doi.org/10.3102/1076998619872761>
- Ramadhan, A. W., Adytia, D., Saepudin, D., Husrin, S., & Adiwijaya, A. (2021). Forecasting of Sea Level Time Series using RNN and LSTM Case Study in Sunda Strait. *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, 12(3), 130. <https://doi.org/10.24843/lkjiti.2021.v12.i03.p01>
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information (Switzerland)* (Vol. 11, Issue 4). MDPI AG. <https://doi.org/10.3390/info11040193>
- Saigal S, & Mehrotra D. (2012). Performance comparison of time series data using predictive data mining techniques. *Advances in*

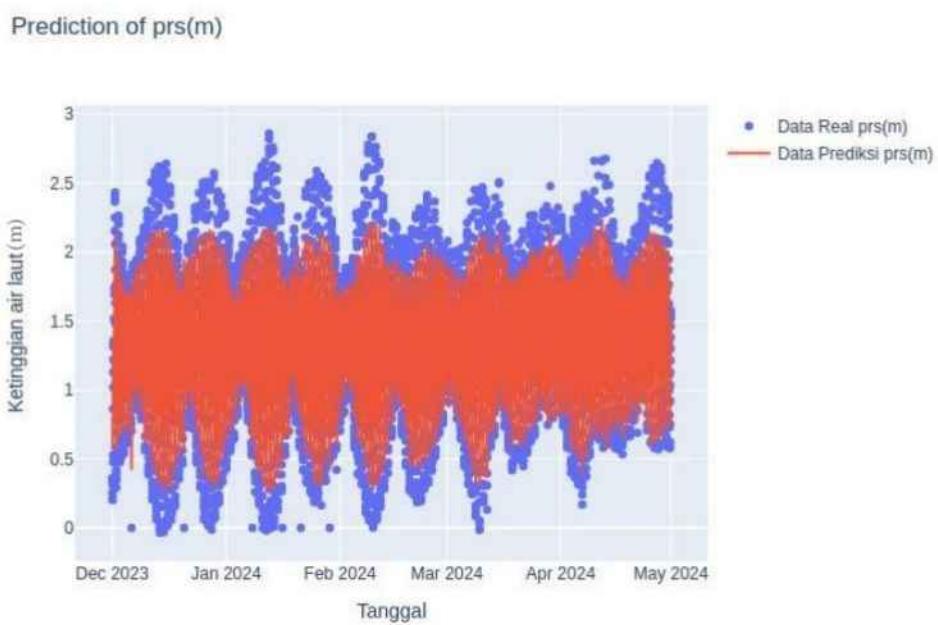
- Information Mining.* 4(1), 57–66.  
<http://www.bioinfo.in/contents.php?id=32>
- Salah Alaloul, W., & Hannan Qureshi, A. (2020). Data Processing Using Artificial Neural Networks. In *Dynamic Data Assimilation - Beating the Uncertainties*. IntechOpen.  
<https://doi.org/10.5772/intechopen.91935>
- Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. In *SN Computer Science* (Vol. 2, Issue 3). Springer. <https://doi.org/10.1007/s42979-021-00592-x>
- Setiawan, Fajar. (2024, 5 Oktober). Wawancara tentang kesalahan prediksi terhadap sector kelautan. Wawancara tidak diterbitkan.
- Sharma, R. (2020). Study of Supervised Learning and Unsupervised Learning. *International Journal for Research in Applied Science and Engineering Technology*, 8(6), 588–593.  
<https://doi.org/10.22214/ijraset.2020.6095>
- Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2023). *Loss Functions and Metrics in Deep Learning*.  
<http://arxiv.org/abs/2307.02694>
- Toffoli, A., & Bitner-Gregersen, E. M. (2017). Types of Ocean Surface Waves, Wave Classification. In *Encyclopedia of Maritime and Offshore Engineering* (pp. 1–8). Wiley.  
<https://doi.org/10.1002/9781118476406.emoe077>
- Xu, H., Shi, H., & Ni, S. (2022). Application of BP Neural Networks in Tide Forecasting. *Atmosphere*, 13(12).  
<https://doi.org/10.3390/atmos13121999>

## LAMPIRAN A DATA HASIL PENELITIAN

### A.1 Prediksi Model CNN



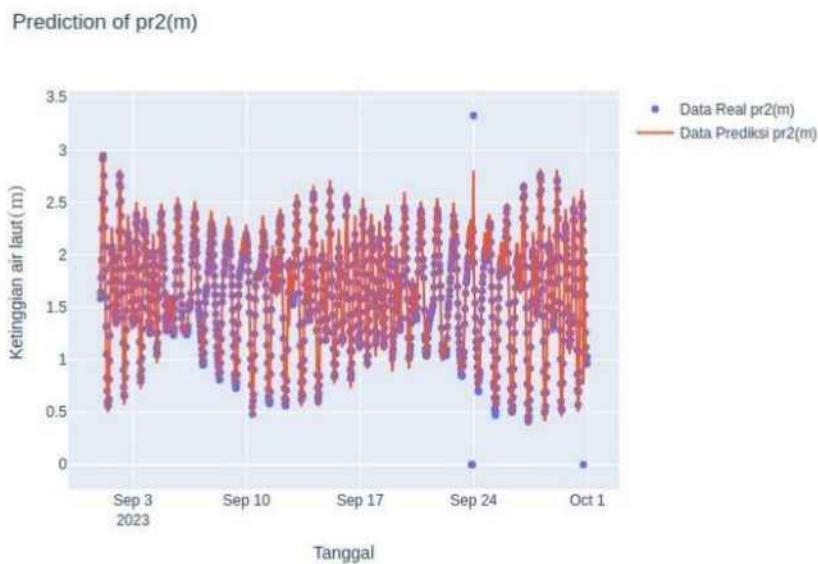
Gambar A.1.1 Hasil Prediksi Model CNN Sensor pr2



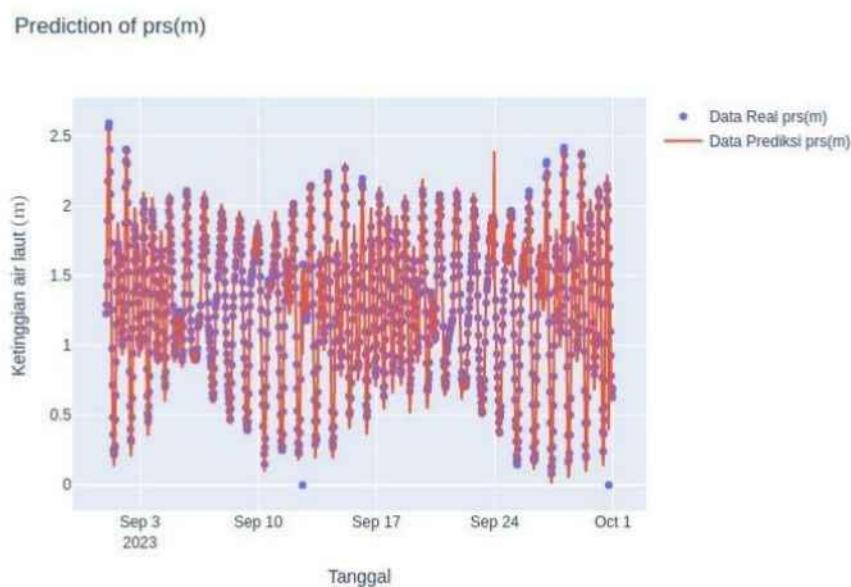
Gambar A.1.2 Hasil Prediksi Model CNN Sensor prs

```
test_loss = 0.026503007858991623
test_mse = 0.061580806970596313
test_mae = 0.15286152064800262
```

Gambar A.1.3 Hasil Pengukuran Metrik CNN

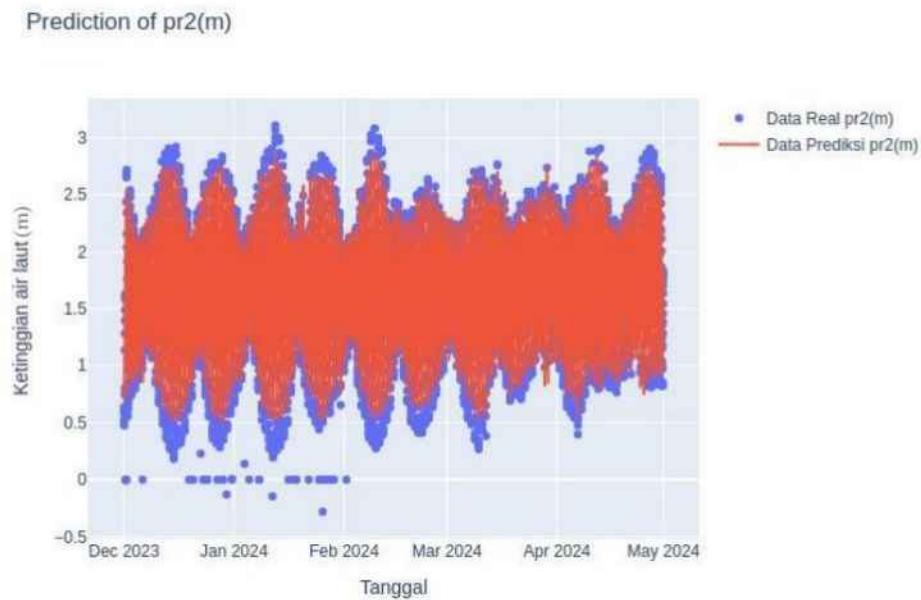


Gambar A.1.4 Prediksi CNN Sensor pr2 Bulan ke 21

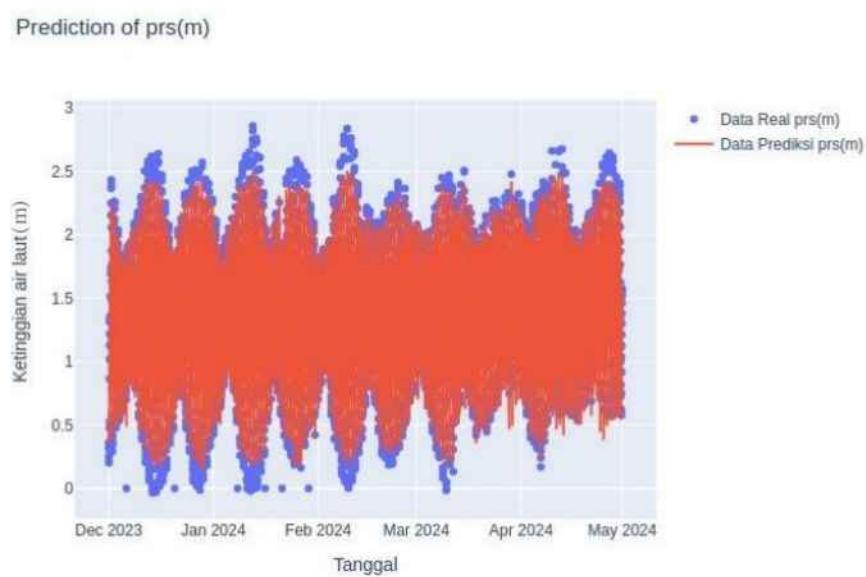


Gambar A.1.5 Prediksi CNN Sensor prs Bulan ke 21

## A.2 Prediksi Model RNN-LSTM *Output 5 Bulan dengan Variasi Input*



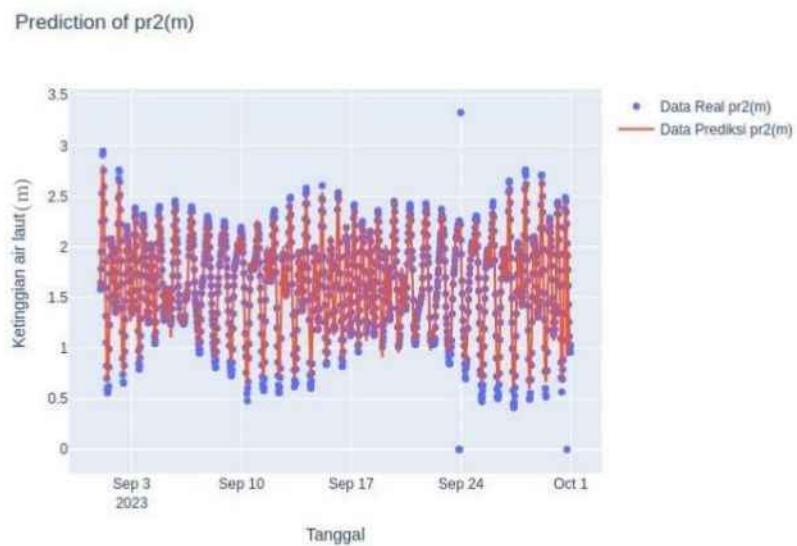
Gambar A.2.1 Prediksi RNN-LSTM Sensor pr2 dengan *Input 23 Bulan Sebelumnya*



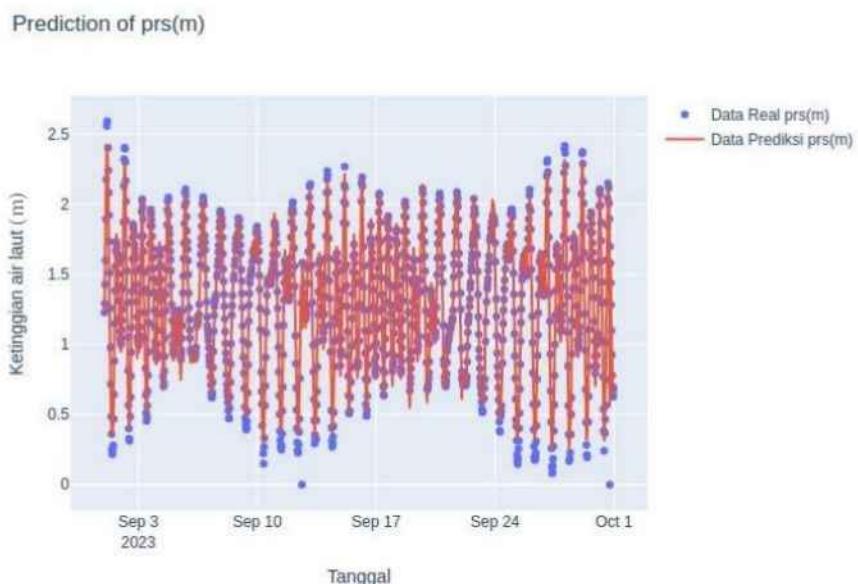
Gambar A.2.2 Prediksi RNN-LSTM Sensor prs dengan *Input 23 Bulan Sebelumnya*

```
test_loss = 0.0325421504676342
test_mse = 0.049598101526498795
test_mae = 0.12007687240839005
```

Gambar A.2.3 Hasil Pengukuran Metrik untuk RNN-LSTM *Input 23 Bulan Sebelumnya*

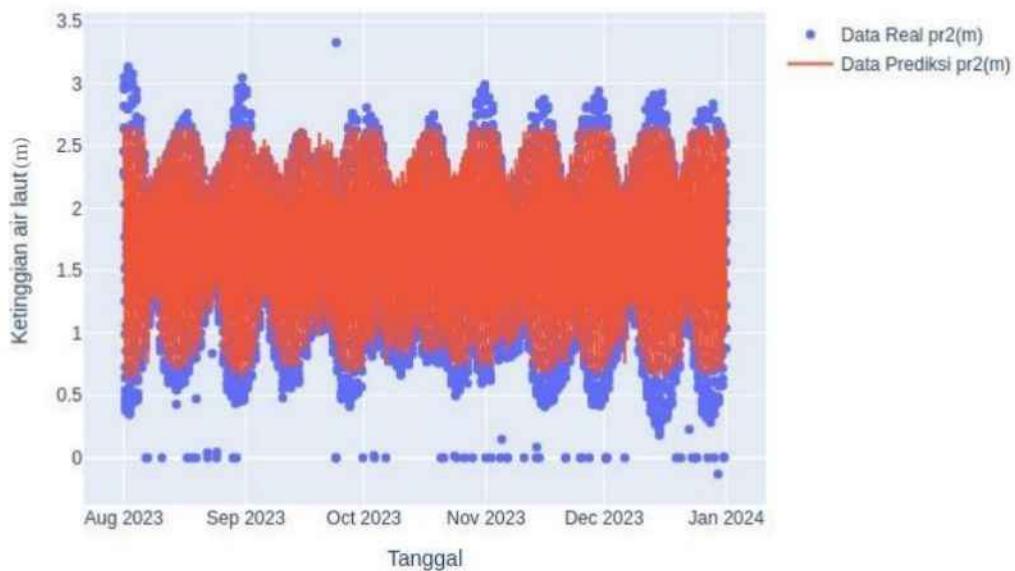


Gambar A.2.4 Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan *Input 23 Bulan Sebelumnya*



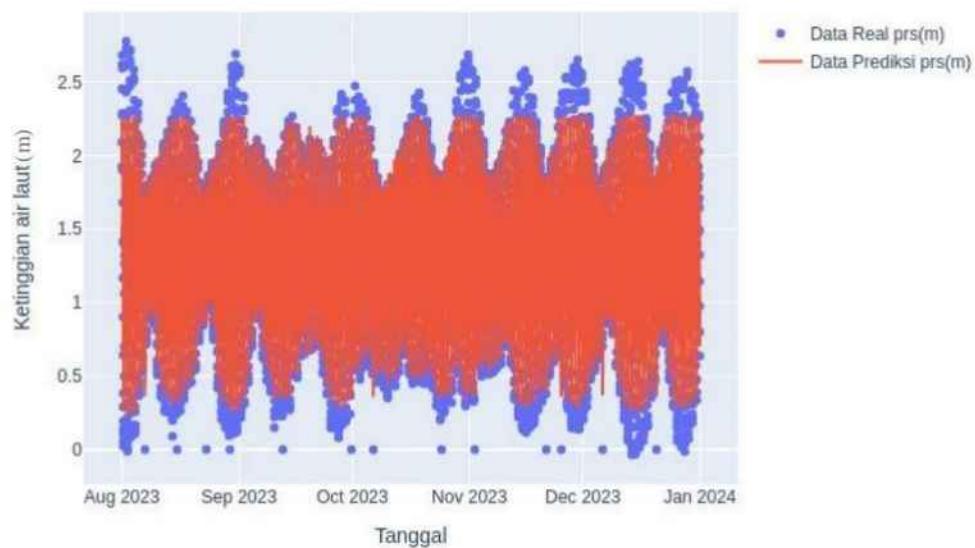
Gambar A.2.5 Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan *Input 23 Bulan Sebelumnya*

Prediction of pr2(m)



Gambar A.2.6 Prediksi RNN-LSTM Sensor pr2 dengan *Input* 18 Bulan Sebelumnya

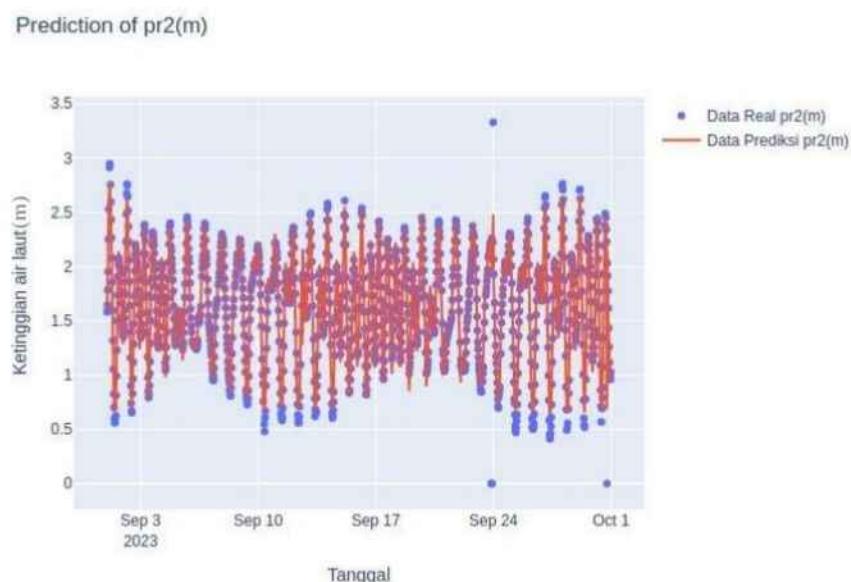
Prediction of prs(m)



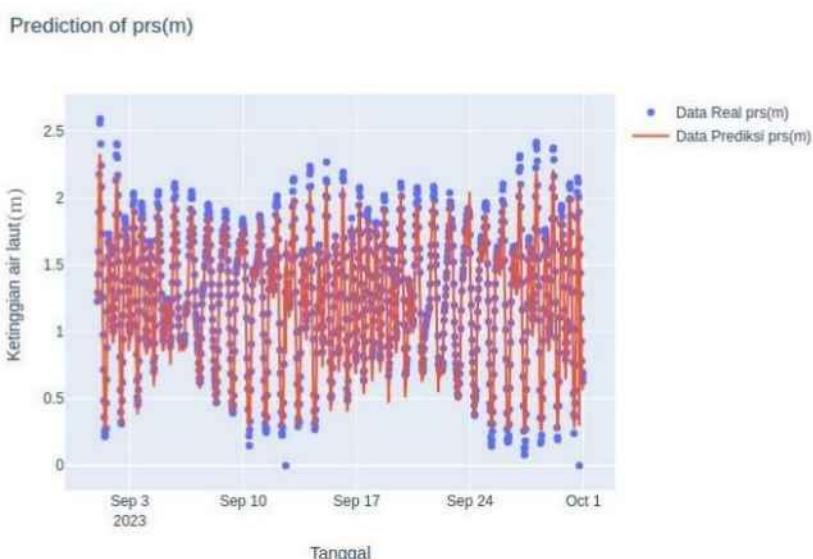
Gambar A.2.7 Prediksi RNN-LSTM Sensor prs dengan *Input* 18 Bulan Sebelumnya

```
test_loss = 0.03467368707060814
test_mse = 0.07613926380872726
test_mae = 0.12299912422895432
```

Gambar A.2.8 Hasil Pengukuran Metrik untuk RNN-LSTM *Input 18 Bulan Sebelumnya*

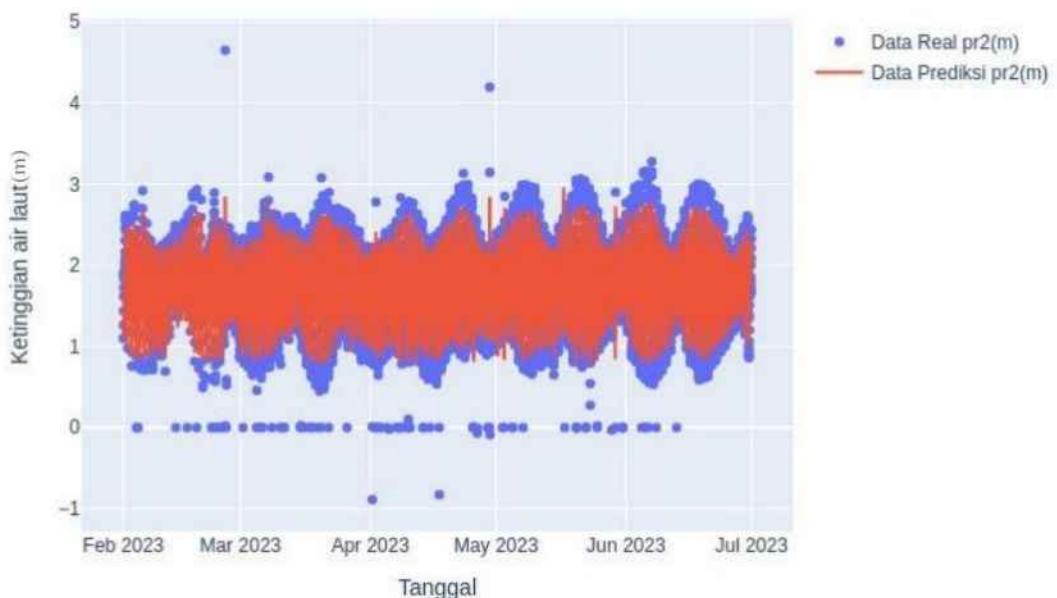


Gambar A.2.9 Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan *Input 18 Bulan Sebelumnya*



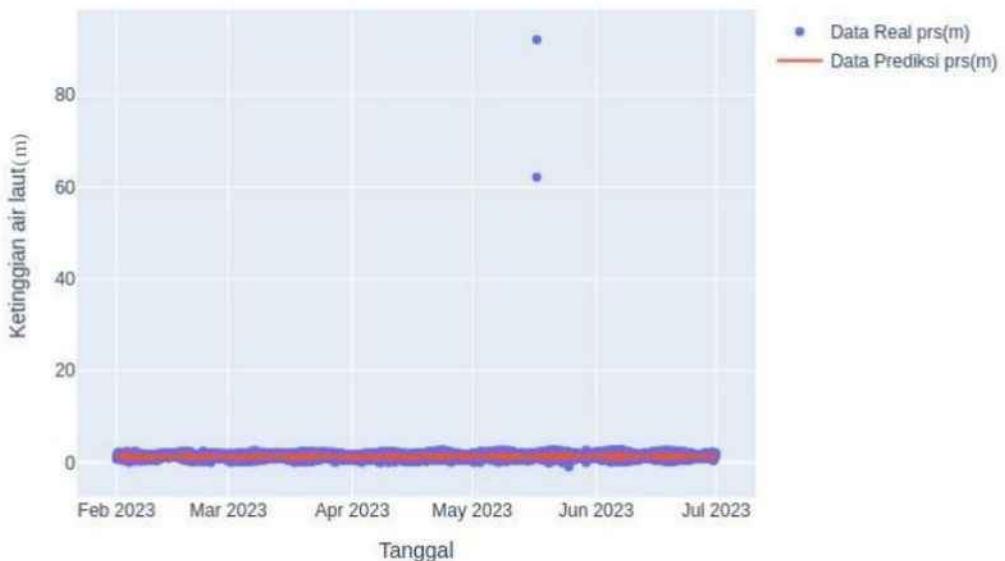
Gambar A.2.10 Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan *Input 18 Bulan Sebelumnya*

Prediction of pr2(m)



Gambar A.2.11 Prediksi RNN-LSTM Sensor pr2 dengan *Input* 13  
Bulan Sebelumnya

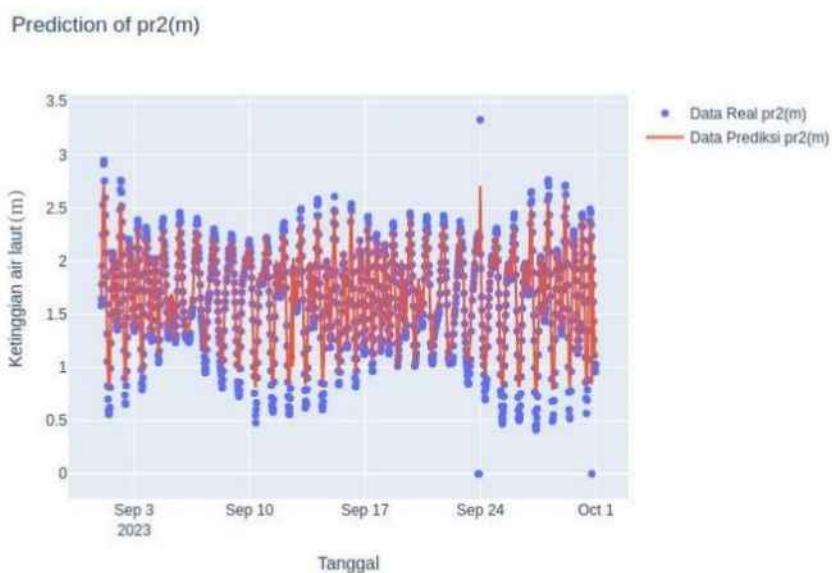
Prediction of prs(m)



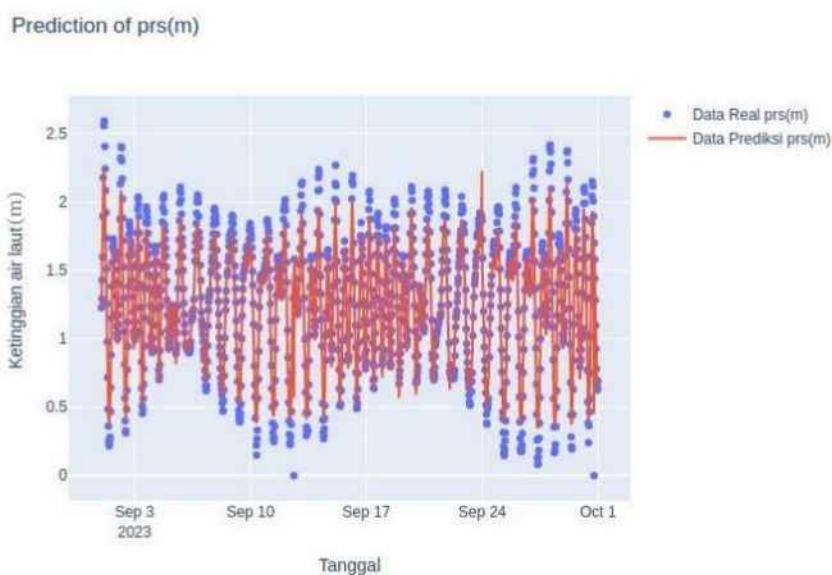
Gambar A.2.12 Prediksi RNN-LSTM Sensor prs dengan *Input* 13  
Bulan Sebelumnya

```
test_loss = 0.06920469552278519
test_mse = 1.508800983428955
test_mae = 0.18486548960208893
```

Gambar A.2.13 Hasil Pengukuran Metrik untuk RNN-LSTM *Input 13 Bulan Sebelumnya*

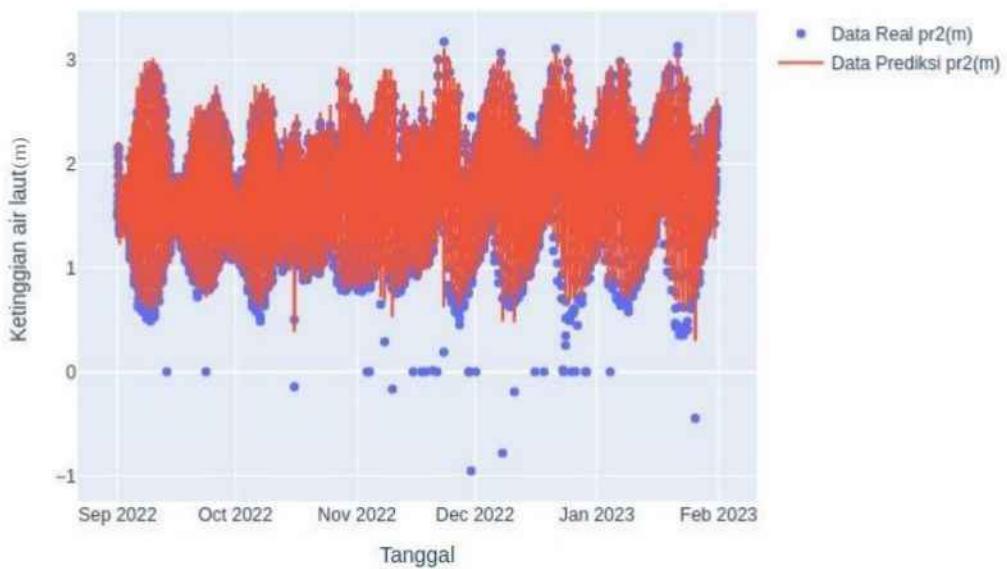


Gambar A.2.14 Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan *Input 13 Bulan Sebelumnya*



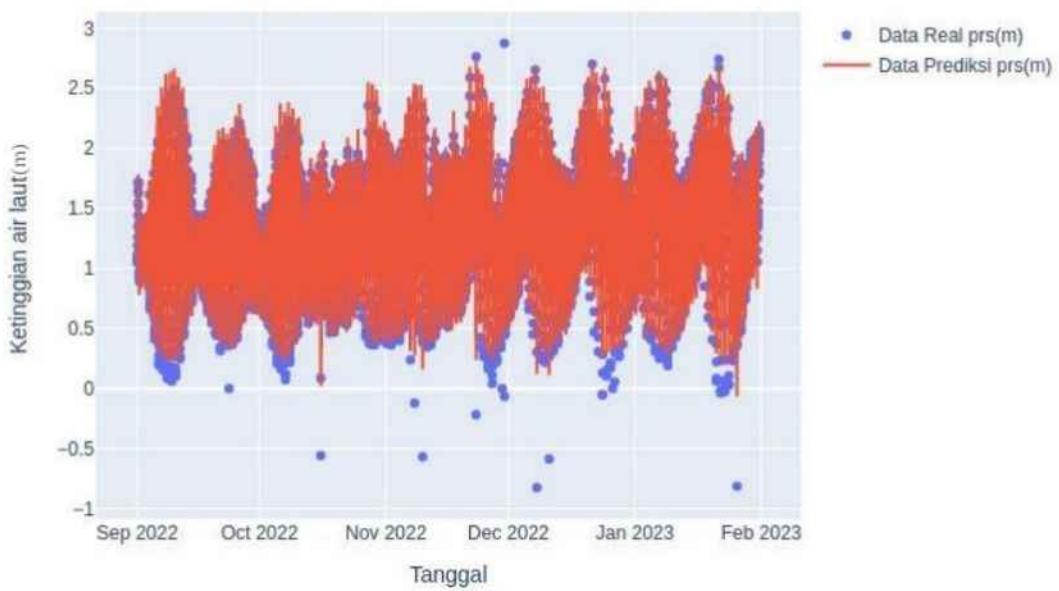
Gambar A.2.15 Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan *Input 13 Bulan Sebelumnya*

Prediction of pr2(m)



Gambar A.2.16 Prediksi RNN-LSTM Sensor pr2 dengan *Input 8 Bulan Sebelumnya*

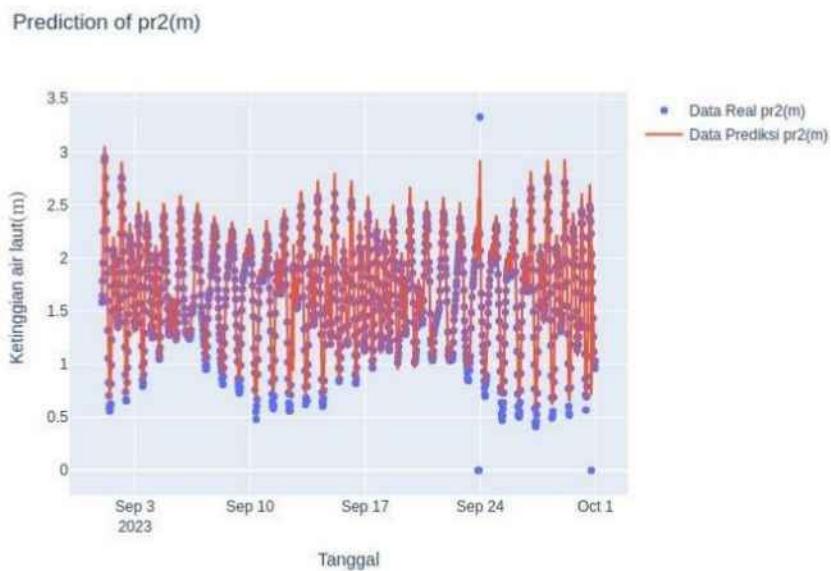
Prediction of prs(m)



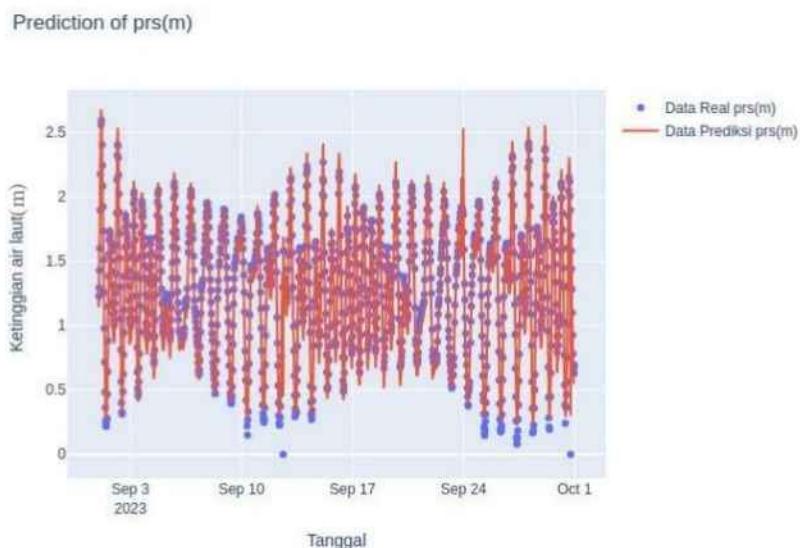
Gambar A.2.17 Prediksi RNN-LSTM Sensor prs dengan *Input 8 Bulan Sebelumnya*

```
test_loss = 0.07258886098861694
test_mse = 0.1950789988040924
test_mae = 0.18198682367801666
```

Gambar A.2.18 Hasil Pengukuran Metrik untuk RNN-LSTM *Input 8 Bulan Sebelumnya*



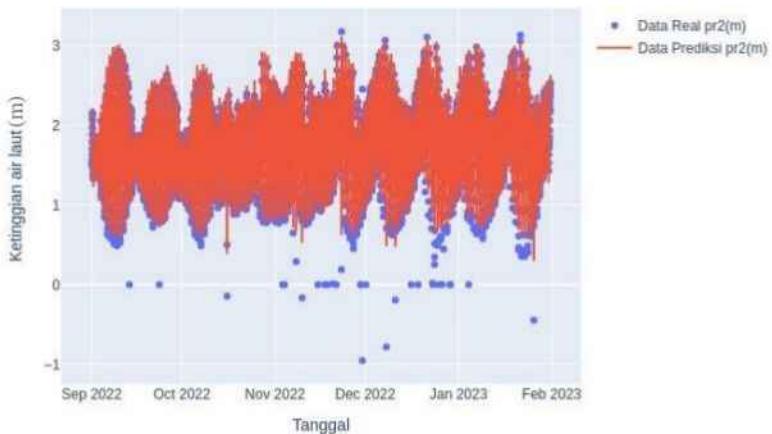
Gambar A.2.19 Prediksi RNN-LSTM Sensor pr2 Bulan ke 21 dengan *Input 8 Bulan Sebelumnya*



Gambar A.2.20 Prediksi RNN-LSTM Sensor prs Bulan ke 21 dengan *Input 8 Bulan Sebelumnya*

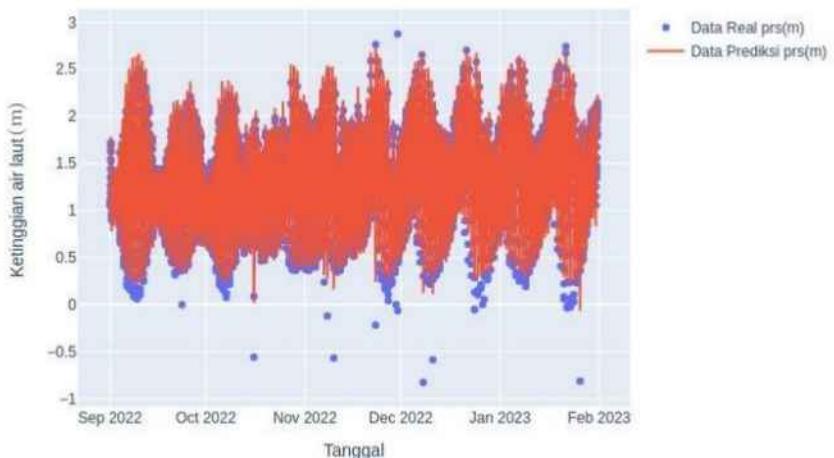
### A.3 Prediksi Model RNN-LSTM *Input* 8 Bulan untuk Prediksi Beberapa Bulan Kedepan

Prediction of pr2(m)



Gambar A.3.1 Prediksi RNN-LSTM 5 Bulan Pertama pada Sensor pr2

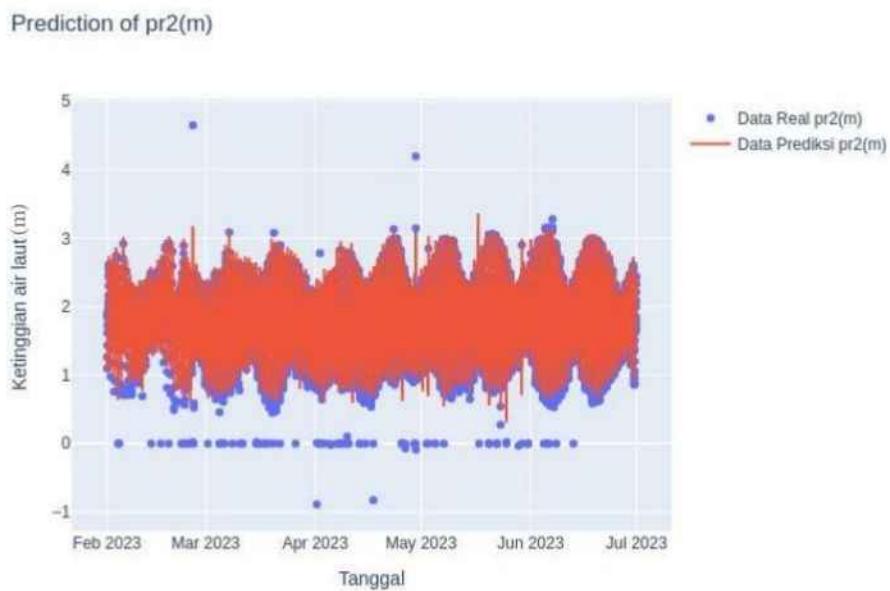
Prediction of prs(m)



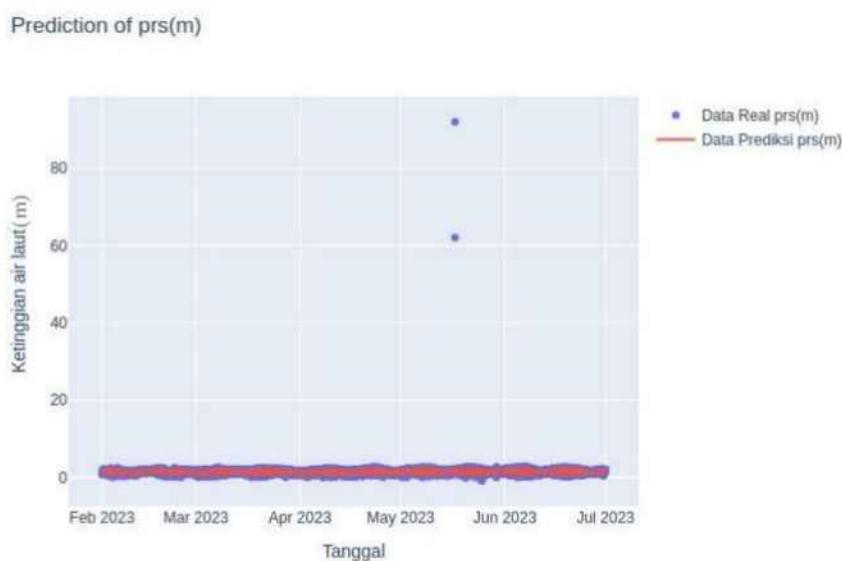
Gambar A.3.2 Prediksi RNN-LSTM 5 Bulan Pertama pada Sensor prs

```
test_loss = 0.07258886098861694
test_mse = 0.1950789988040924
test_mae = 0.18198682367801666
```

Gambar A.3.3 Hasil Pengukuran Metrik Prediksi 5 Bulan Pertama



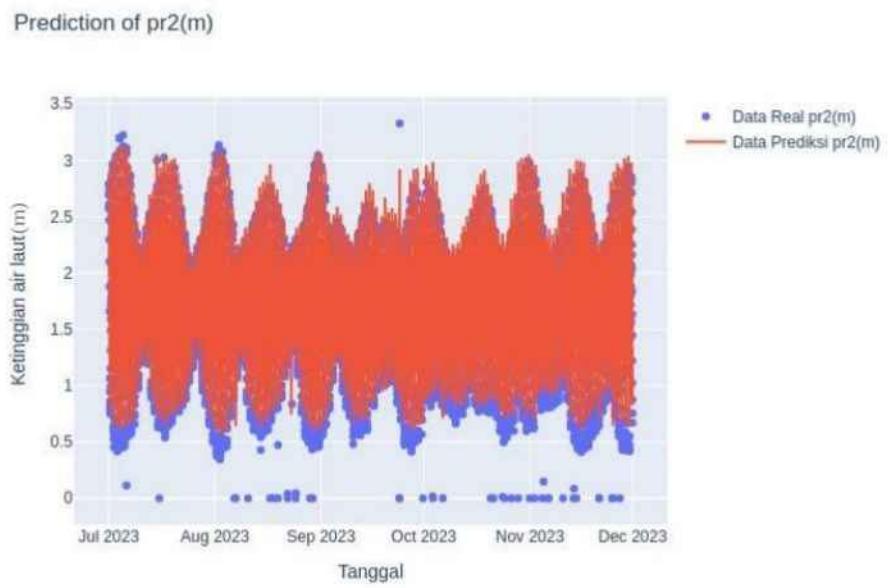
Gambar A.3.4 Prediksi RNN-LSTM 5 Bulan Kedua pada Sensor pr2



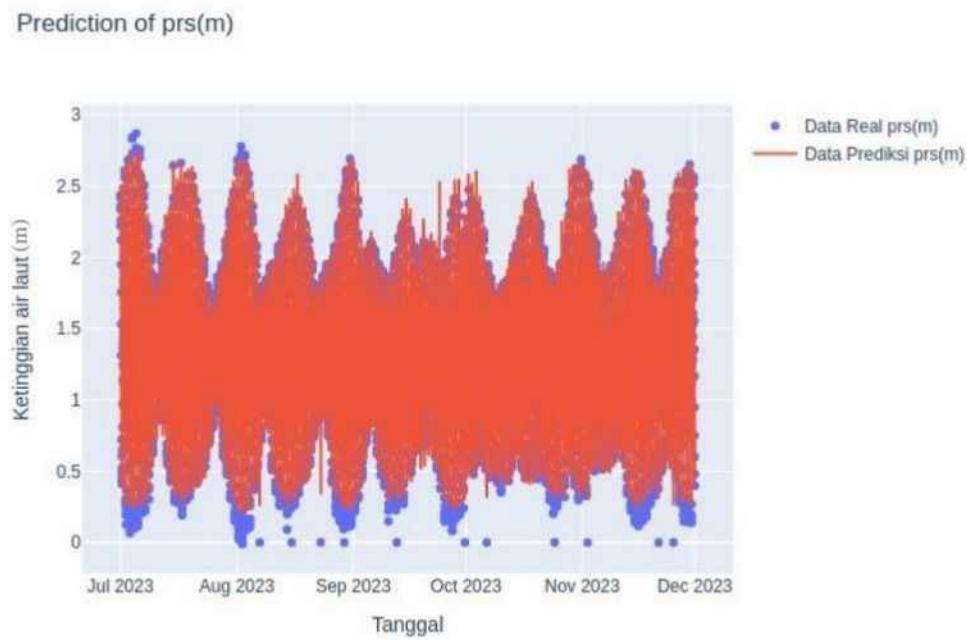
Gambar A.3.5 Prediksi RNN-LSTM 5 Bulan Kedua pada Sensor prs

```
test_loss = 0.06825382262468338
test_mse = 1.4940462112426758
test_mae = 0.17064177989959717
```

Gambar A.3.6 Hasil Pengukuran Metrik Prediksi 5 Bulan Kedua



Gambar A.3.7 Prediksi RNN-LSTM 5 Bulan Ketiga pada Sensor pr2

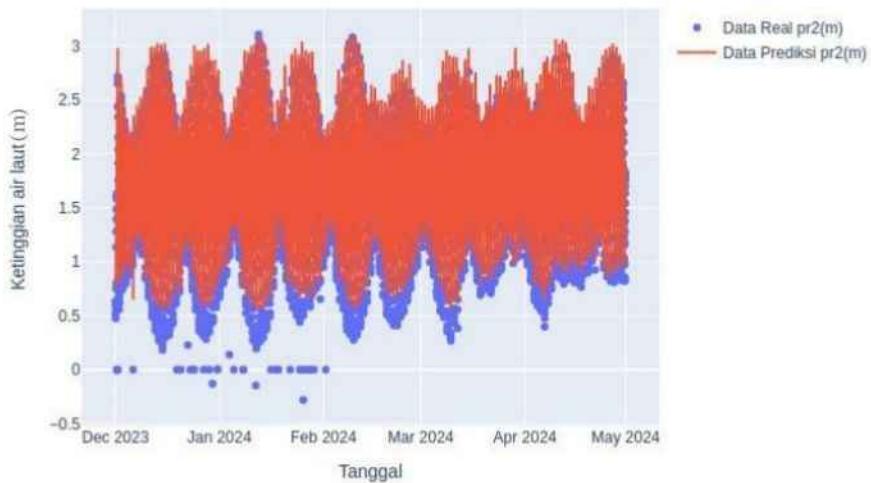


Gambar A.3.8 Prediksi RNN-LSTM 5 Bulan Ketiga pada Sensor prs

```
test_loss = 0.03200923278927803
test_mse = 0.06577444821596146
test_mae = 0.13248105347156525
```

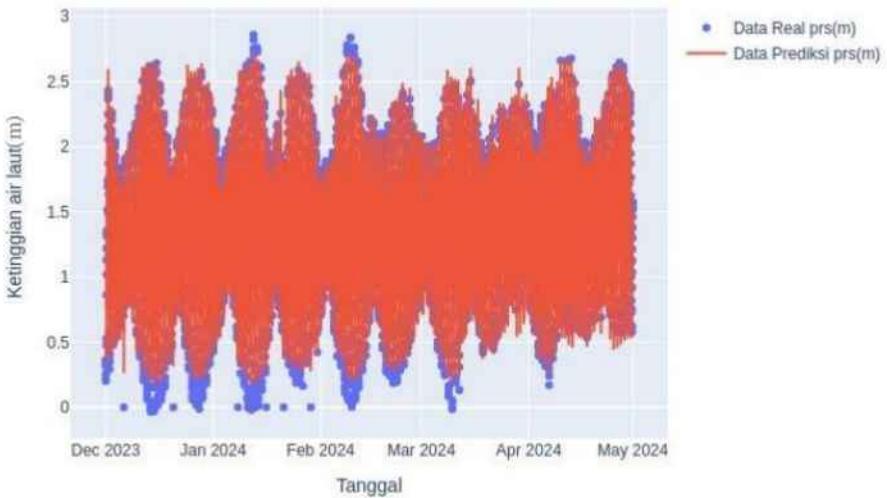
Gambar A.3.9 Hasil Pengukuran Metrik Prediksi 5 Bulan Ketiga

Prediction of pr2(m)



Gambar A.3.10 Prediksi RNN-LSTM 5 Bulan Keempat pada Sensor pr2

Prediction of prs(m)



Gambar A.3.11 Prediksi RNN-LSTM 5 Bulan Keempat pada Sensor prs

```
test_loss = 0.04687538370490074
test_mse = 0.09137874841690063
test_mae = 0.19696144759655
```

Gambar A.3.12 Hasil Pengukuran Metrik Prediksi 5 Bulan Keempat

## LAMPIRAN B KODE PROGRAM

```
↳ pip install -U kaleido

[1]: from google.colab import drive
drive.mount('/content/drive')

[2]: import pandas as pd

df_final = pd.read_excel('drive/MyDrive/Tugas Akhir/Dataset/Per_30_Mins_Data Tidal Wave Surabaya for Training.xlsx')

[3]: new_header = df_final.iloc[0] #grab the first row for the header
df_final = df_final[1:] #take the data less the header row
df_final.columns = new_header #set the header row as the df header

[4]: df_final

[5]: # Set Index with Time
df_final = df_final.set_index("Time (UTC)")

#main_data = df_final[['pr2(m)', 'prs(m)', 'rad(m)']].values.tolist()
main_data = df_final[['pr2(m)', 'prs(m)']].values.tolist()
main_data_index = df_final.index.tolist()

[6]: monthly = '5'

# Obtain the test set first
test_set = df_final.last('({}M'.format(monthly))
#test_data = test_set[['pr2(m)', 'prs(m)', 'rad(m)']].values.tolist()
test_data = test_set[['pr2(m)', 'prs(m)']].values.tolist()
test_data_index = test_set.index.tolist()

[7]: # Get every data previously from the final test set
max_date = test_set.index[0]
train_set = df_final[:max_date]

#train_data = train_set[['pr2(m)', 'prs(m)', 'rad(m)']].values.tolist()
train_data = train_set[['pr2(m)', 'prs(m)']].values.tolist()
train_data_index = train_set.index.tolist()
```

### Normalization and Windowing Technique

```
[1]: from sklearn.preprocessing import StandardScaler

# Normalize with standard scaler
scaler = StandardScaler()
scaler.fit(main_data)

[2]: # Apply Normalization
train_data = scaler.transform(train_data)
test_data = scaler.transform(test_data)

[3]: import numpy as np

# split a univariate sequence into samples
def split_sequence(sequence, n_steps, date):
    X, y, date_list = list(), list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        if date is not None:
            seq_x, seq_y, seq_date = sequence[i:end_ix], sequence[end_ix], date[end_ix]
            X.append(seq_x)
            y.append(seq_y)
            date_list.append(seq_date)

    date_list.append(seq_date)
else:
    seq_x, seq_y= sequence[i:end_ix], sequence[end_ix]
    X.append(seq_x)
    y.append(seq_y)
    #date_list.append(seq_date)

return np.array(X), np.array(y), np.array(date_list)

n_steps=16 #intuitional, semakin panjang ambil x step nanti modelnya akan terlalu berpatokan pada data yang terlalu ke belakang. kal
x_sequence_train, y_sequence_train, train_seq_date = split_sequence(train_data, n_steps=n_steps, date=train_data_index)

[4]: x_sequence_train.shape
```

```

11 # Save normalized train and test data to Excel
import pandas as pd

# Convert train and test data back to DataFrame for saving
train_data_normalized_df = pd.DataFrame(train_data, columns=['pr2(m)_normalized', 'prs(m)_normalized'], index=train_data_index)
test_data_normalized_df = pd.DataFrame(test_data, columns=['pr2(m)_normalized', 'prs(m)_normalized'], index=test_data_index)

# Save the train and test normalized data as Excel files
train_data_normalized_df.to_excel('normalized_train_data.xlsx', sheet_name='Train Data')
test_data_normalized_df.to_excel('normalized_test_data.xlsx', sheet_name='Test Data')

# For Google Colab: Add functionality to download the Excel files
from google.colab import files
files.download('normalized_train_data.xlsx')
files.download('normalized_test_data.xlsx')

print("Train and test normalized data have been saved to Excel and are ready for download.")

```

## Define and Train the Model

```

11 import tensorflow as tf

from tensorflow.keras.layers import Bidirectional

def create_model():
    # Input layers
    return tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_steps, 2)),
        Bidirectional(tf.keras.layers.LSTM(128, recurrent_regularizer=tf.keras.regularizers.l1(1e-5), return_sequences=True)),
        tf.keras.layers.Dropout(0.1),
        Bidirectional(tf.keras.layers.LSTM(256, recurrent_regularizer=tf.keras.regularizers.l1(1e-5), return_sequences=False)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(128, activation=tf.keras.activations.swish),
        tf.keras.layers.Dense(64, activation=tf.keras.activations.swish),
        tf.keras.layers.Dense(2)
    ])

11 model = create_model()

11 def scheduler(epoch:int) -> float:
    if epoch < 4:
        return 0.01
    elif epoch < 30:
        return 0.001
    else:
        return 0.0001

scheduler_callback = tf.keras.callbacks.LearningRateScheduler(scheduler)

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=scheduler(0)),
    loss=tf.keras.losses.Huber(),
    metrics=['mae', 'mse']
)

11 model_history = model.fit(x_sequence_train, y_sequence_train, epochs=5, verbose=1)

```

## Evaluate

```

11 final_loss = model_history.history['loss'][-1]
final_mae = model_history.history['mae'][-1]
final_mse = model_history.history['mse'][-1]

print("Model Loss : {}".format(final_loss))
print("Final MAE : {}".format(final_mae))
print("Final MSE : {}".format(final_mse))

11 # Prediction
# Append the final 4
evaluation_list = [x for x in train_data[-n_steps:]]
evaluation_list_data_index = [x for x in train_data_index[-n_steps:]]

for idx, x in enumerate(test_data):
    evaluation_list.append(x)
    evaluation_list_data_index.append(test_data_index[idx])

11 x_sequence_eval, y_sequence_eval, date_seq_eval = split_sequence(evaluation_list, n_steps=n_steps, date=evaluation_list_data_index)

11 final_loss_pred, final_mae_pred, final_mse_pred = model.evaluate(x_sequence_eval, y_sequence_eval)
predicted_result = model.predict(x_sequence_eval)

11 print("Model Loss : {}".format(final_loss_pred))
print("Final MAE : {}".format(final_mae_pred))
print("Final MSE : {}".format(final_mse_pred))

```

```

[1] # Reverse Transform
# Retransform the predicted result
predict_result = scaler.inverse_transform(predicted_result)
print(len(predicted_result))

# Get the predicted result
first_sensors, second_sensors, third_sensors = [], [], []
for result in predict_result:
    first_sensors.append(result[0])
    second_sensors.append(result[1])
    #third_sensors.append(result[2])

[1] # Separate by index
index_1 = df_final[['pr2(m)', 'prs(m)', 'rad(m)']].values.tolist()
index_1 = df_final[['pr2(m)', 'prs(m)']].values.tolist()
value_2 = np.array(df_final[['pr2(m)']].values.tolist()).reshape(-1, )
value_3 = np.array(df_final[['prs(m)']].values.tolist()).reshape(-1, )
value_4 = np.array(df_final[['rad(m)']].values.tolist()).reshape(-1, )
y = [integer for integer in range(len(value_3))]

[1] import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from plotly.subplots import make_subplots

[1] def make_plot(column_name, datetime_main, datetime_pred, value_main, value_pred, save_path:str):
    fig = make_subplots(rows=1, cols=1)

    trace1 = go.Scatter(x=datetime_main, y=value_main, mode='markers', name='Data Real {}'.format(column_name))
    trace2 = go.Scatter(x=datetime_pred, y=value_pred, mode='lines', name='Data Prediksi {}'.format(column_name))

    fig.add_trace(trace1)
    fig.add_trace(trace2)

    fig.update_layout(title='Prediction of {}'.format(column_name),
                      xaxis_title='Tanggal',
                      yaxis_title='Ketinggian air laut')

    fig.show()

    if save_path != '':
        fig.write_image(save_path)

```

#### Plot and document result

```

[1] import os

# Saving Name Folder
saving_dir_name = 'drive/MyDrive/Tugas Akhir/Result'

# Initialize Directory
if os.path.isdir(saving_dir_name) is False:
    os.mkdir(saving_dir_name)

# Document the data
final_txt_eval = "train_loss = {}\ntrain_mse = {}\ntrain_mae = {}\ntest_loss = {}\ntest_mse = {}\ntest_mae = {}".format(
    final_loss,
    final_mse,
    final_mae,
    final_loss_pred,
    final_mse_pred,
    final_mae_pred
)

[1] # Save evaluation result
final_saving_txt_path = os.path.join(saving_dir_name, "plain_evaluation.txt")
with open(final_saving_txt_path, 'w') as f:
    f.write(final_txt_eval)

[1] import matplotlib.pyplot as plt

fig_model_loss = plt.gcf()
# plt.plot(model_history.history['mae'][10:])
# plt.plot(model_history.history['mse'][10:])
# plt.plot(model_history.history['loss'][10:])
plt.plot(model_history.history['mae'])
plt.plot(model_history.history['mse'])
plt.plot(model_history.history['loss'])
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['mae', 'mse', 'loss'], loc='upper left')
plt.savefig(os.path.join(saving_dir_name, 'loss.jpg'))
plt.show()

```

```

[1] # First Sensor
make_plot(
    column_name = 'pr2(m)',
    datetime_main = main_data_index,
    datetime_pred = test_data_index,
    value_main=value_2,
    value_pred=first_sensors,
    save_path=os.path.join(saving_dir_name, 'pr2.jpg')
)

[2] # Second Sensor
make_plot(
    column_name = 'prs(m)',
    datetime_main = main_data_index,
    datetime_pred = test_data_index,
    value_main=value_3,
    value_pred=second_sensors,
    save_path=os.path.join(saving_dir_name, 'prs.jpg')
)

[3] # # Third Sensor
# make_plot(
#     column_name = 'rad(m)',
#     datetime_main = main_data_index,
#     datetime_pred = test_data_index,
#     value_main=value_4,
#     value_pred=third_sensors,
#     save_path=os.path.join(saving_dir_name, 'rad.jpg')
# )

```

## Second Round Evaluation

```

[1] def separate_eval(month: int, year: int, save_name: str, evaluate_5_months: bool = False):
    if evaluate_5_months:
        #Time frame for evaluation: 5 months.
        start_date = pd.Timestamp(year=year, month=month, day=1) - pd.DateOffset(months=4)
        end_date = pd.Timestamp(year=year, month=month, day=1) + pd.DateOffset(months=1)

        #Filter data for a 5-month time range.
        selected_row_data = df_final.loc[
            (df_final.index >= start_date) & (df_final.index < end_date)
        ][['pr2(m)', 'prs(m)']]

    else:
        #Specific month evaluation
        selected_row_data = df_final.loc[
            (df_final.index.month == month) & (df_final.index.year == year)
        ][['pr2(m)', 'prs(m)']]

    #Add 15 previous data points to provide context.
    minimal_row_data = selected_row_data.index[0]
    append_data = df_final.loc[:minimal_row_data][-15:][['pr2(m)', 'prs(m)']]

    #Combine the main data and the additional data
    merged_dataframe = pd.concat([selected_row_data, append_data]).sort_index()

    #data conversion for evaluation
    evaluation_list_data_index = selected_row_data.index.tolist()
    selected_row_data_final = merged_dataframe.values.tolist()
    selected_row_data_final = scaler.transform(selected_row_data_final)

    #Convert the data into a sequence for prediction.
    x_sequence_eval, y_sequence_eval, date_seq_eval = split_sequence(
        selected_row_data_final, n_steps=n_steps, date=None
    )

    #Prediction and Evaluation
    predicted_value = model.predict(x_sequence_eval)
    predicted_value = scaler.inverse_transform(predicted_value)
    final_loss_pred, final_mae_pred, final_mse_pred = model.evaluate(x_sequence_eval, y_sequence_eval)

[1] print("Model Loss : {}".format(final_loss_pred))
print("Final MAE : {}".format(final_mae_pred))
print("Final MSE : {}".format(final_mse_pred))

#Save the evaluation results to a file
final_txt_eval = "test_loss = {}\ntest_mse = {}\ntest_mae = {}".format(
    final_loss_pred, final_mse_pred, final_mae_pred
)
final_saving_txt_path = os.path.join(saving_dir_name, "{}_evaluation.txt".format(save_name))
with open(final_saving_txt_path, 'w') as f:
    f.write(final_txt_eval)

#make plot for sensor
first_sensors, second_sensors = [], []
for result in predicted_value:
    first_sensors.append(result[0])
    second_sensors.append(result[1])

```

```

# Plot sensor pr2
make_plot(
    column_name='pr2(m)',
    datetime_main=evaluation_list_data_index,
    datetime_pred=evaluation_list_data_index,
    value_main=selected_row_data['pr2(m)'].values.tolist(),
    value_pred=first_sensors,
    save_path=os.path.join(saving_dir_name, '{}_pr2.jpg'.format(save_name))
)

# Plot sensor prs
make_plot(
    column_name='prs(m)',
    datetime_main=evaluation_list_data_index,
    datetime_pred=evaluation_list_data_index,
    value_main=selected_row_data['prs(m)'].values.tolist(),
    value_pred=second_sensors,
    save_path=os.path.join(saving_dir_name, '{}_prs.jpg'.format(save_name))
)

[1] separate_eval(month=4, year=2024, save_name="evaluasi_5_bulan_pertama", evaluate_5_months=True)

[1] separate_eval(month=1, year=2023, save_name="evaluasi_5_bulan_pertama", evaluate_5_months=True)

[1] separate_eval(month=6, year=2023, save_name="evaluasi_5_bulan_kedua", evaluate_5_months=True)

[1] separate_eval(month=11, year=2023, save_name="evaluasi_5_bulan_ketiga", evaluate_5_months=True)

[1] separate_eval(month=4, year=2024, save_name="evaluasi_5_bulan_keempat", evaluate_5_months=True)

[1] # bulan 21-25 9/23 hingga 01/24
# Bulan 21
separate_eval(month=9, year=2023, save_name="bulan_21")

[1] # Bulan 22
separate_eval(month=10, year=2023, save_name="bulan_22")

[1] # Bulan 23
separate_eval(month=11, year=2023, save_name="bulan_23")

[1] # Bulan 24
separate_eval(month=12, year=2023, save_name="bulan_24")

[1] # Bulan 25
separate_eval(month=1, year=2024, save_name="bulan_25")

[1] # Ensure df_final is defined (for example, from a CSV file)
try:
    # Replace 'your_data_file.csv' with the actual data file name
    df_final = pd.read_csv('your_data_file.csv')
    print("df_final loaded successfully. Columns available:", df_final.columns)
except FileNotFoundError:
    print("File 'your_data_file.csv' not found. Please upload the file or check the path.")
    # Example fallback definition for demonstration purposes
    data = {'pr2(m)': [0.1, 0.2, 0.3], 'prs(m)': [0.4, 0.5, 0.6]}
    df_final = pd.DataFrame(data)
    print("Sample df_final created:", df_final)

[1] # Assuming 'predicted_result' contains the predictions and has 2 columns
# Modify column names if necessary
predictions_df = pd.DataFrame(predicted_result, columns=['Predicted Value 1', 'Predicted Value 2'])

# Save to Excel
predictions_df.to_excel('predicted_results.xlsx', index=False)
print("Predicted results have been saved to 'predicted_results.xlsx'")

[1]
from google.colab import files

# Download the saved Excel file
files.download('predicted_results.xlsx')

```