BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian mengenai prediksi *Remaining Useful Life* (RUL) mesin turbofan menggunakan dataset NASA C-MAPSS telah berkembang pesat dalam beberapa tahun terakhir. Berbagai pendekatan dikembangkan untuk meningkatkan akurasi prediksi, mulai dari penggunaan arsitektur *Long Short-Term Memory* (LSTM) konvensional hingga model *deep learning* yang lebih kompleks. Subbab ini membahas penelitian-penelitian yang memiliki keterkaitan langsung dengan penelitian yang dilakukan serta mengidentifikasi *research gap* yang menjadi dasar pengembangan penelitian ini.

2.1.1 Penelitian LSTM pada Prediksi RUL NASA C-MAPSS

Penelitian oleh Asif *et al.* [9] mengusulkan model *Long Short-Term Memory* (LSTM) yang dikombinasikan dengan tahapan prapemrosesan berupa analisis korelasi untuk seleksi fitur sensor, *moving median filter* untuk mereduksi *outlier* dan *noise*, normalisasi *Z-score*, serta model *piece-wise linear degradation* yang disempurnakan untuk menentukan titik awal degradasi (*initial RUL*) secara otomatis pada setiap sub-dataset, alih-alih menggunakan nilai RUL maksimum yang ditetapkan secara tetap seperti pada penelitian sebelumnya [9]. Model diuji pada keempat sub-dataset NASA C-MAPSS (FD001, FD002, FD003, dan FD004) dan dievaluasi menggunakan metrik *Root Mean Squared Error* (RMSE) serta *Scoring Function* asimetris [9]. Meskipun demikian, penelitian tersebut hanya mengevaluasi variasi *hyperparameter* dalam satu jenis arsitektur LSTM melalui *grid search*, sehingga belum diketahui bagaimana performanya apabila dibandingkan secara eksperimental dengan algoritma *machine learning* lain seperti *Random Forest* (RF) atau *Support Vector Regression* (SVR) pada kondisi prapemrosesan yang sama [9].

Penelitian oleh Tan *et al.* [17] mengusulkan arsitektur hibrida yang menggabungkan 2D-CNN, LSTM, dan *Temporal Convolutional Network* (TCN) untuk meningkatkan kemampuan ekstraksi fitur spasial dan temporal pada dataset NASA C-MAPSS [17]. Penelitian tersebut menggunakan strategi *piece-wise linear labelling* dengan $RUL_{max} = 125$ serta mengevaluasi model menggunakan RMSE dan *Scoring Function* [17]. Pendekatan ini berhasil meningkatkan

akurasi prediksi, namun integrasi beberapa modul *deep learning* menyebabkan kompleksitas komputasi dan waktu pelatihan menjadi lebih tinggi dibandingkan penggunaan LSTM tunggal [17].

Shi dan Chehade [10] mengembangkan pendekatan Dual-LSTM yang menggabungkan proses deteksi titik perubahan degradasi (*change point detection*) dan prediksi RUL dalam satu kerangka kerja [10]. Pendekatan ini dirancang untuk mengatasi keterbatasan penggunaan nilai RUL_{max} yang bersifat tetap dengan mempertimbangkan karakteristik degradasi masing-masing mesin [10]. Evaluasi dilakukan menggunakan *Scoring Function* yang lebih sesuai untuk aplikasi *predictive maintenance* [10]. Meskipun memberikan performa yang sangat baik, arsitektur Dual-LSTM memiliki kompleksitas yang lebih tinggi sehingga implementasinya membutuhkan sumber daya komputasi yang lebih besar [10].

2.1.2 Analisis Research Gap

Berdasarkan penelitian-penelitian terdahulu, dapat diidentifikasi beberapa celah penelitian (*research gap*). Sebagian besar penelitian berfokus pada peningkatan akurasi melalui pengembangan arsitektur *deep learning* yang semakin kompleks, seperti integrasi CNN-LSTM-TCN maupun Dual-LSTM [17, 10]. Pendekatan tersebut memang menghasilkan performa yang baik, namun diikuti dengan meningkatnya kompleksitas model dan kebutuhan komputasi [17, 10].

Di sisi lain, penelitian yang menggunakan arsitektur LSTM konvensional umumnya hanya mengevaluasi satu jenis model sehingga belum memberikan gambaran mengenai keunggulan LSTM dibandingkan algoritma *machine learning* lain seperti Random Forest (RF), Support Vector Regression (SVR), dan Multi-Layer Perceptron (MLP). Selain itu, tidak seluruh penelitian menggunakan *Scoring Function* sebagai metrik evaluasi, padahal metrik tersebut lebih sesuai untuk prediksi RUL karena memberikan penalti yang lebih besar terhadap kesalahan prediksi yang terlambat.

Berdasarkan kondisi tersebut, masih terdapat kebutuhan untuk mengevaluasi kinerja arsitektur Stacked LSTM yang relatif sederhana menggunakan konfigurasi *piece-wise linear labelling* dengan $RUL_{max} = 125$ serta membandingkannya secara langsung dengan algoritma RF, SVR, dan MLP menggunakan metrik utama *Scoring Function* serta RMSE sebagai metrik pendukung.

2.1.3 Posisi Penelitian

Posisi penelitian ini terhadap penelitian terdahulu dirangkum pada Tabel 2.1.

Tabel 2.1. Perbandingan Penelitian Terdahulu

Aspek	**Penelitian [9]**	**Penelitian [17]**	**Penelitian [10]**	**Penelitian ini**
Dataset	NASA C-MAPSS FD001–FD004	NASA C-MAPSS	NASA C-MAPSS	NASA C-MAPSS FD001
Algoritma/ML	LSTM	2D-CNN, LSTM, TCN	Dual LSTM	LSTM, RF, SVR, MLP
Evaluasi	RMSE, SF	RMSE, SF	SF	SF, RMSE
Labeling	Piece-wise adaptif	Piece-wise standar	Piece-wise standar	Piece-wise standar
Limit RUL	FD001 - FD004 (78, 103, 79, 87)	125	125	125

Berdasarkan Tabel 2.1, penelitian ini menempati posisi yang berbeda dengan penelitian terdahulu karena tidak hanya menggunakan arsitektur Stacked LSTM sebagai model utama, tetapi juga melakukan evaluasi komparatif terhadap Random Forest, Support Vector Regression, dan Multi-Layer Perceptron pada konfigurasi *piece-wise linear labelling* dengan $RUL_{max} = 125$. Selain itu, penelitian ini memprioritaskan *Scoring Function* sebagai metrik utama untuk mengevaluasi performa model dari sudut pandang risiko operasional pada sistem *predictive maintenance*, sementara RMSE digunakan sebagai metrik pendukung untuk mengukur besarnya kesalahan prediksi secara numerik.

2.2 Remaining Useful Life (RUL)

Secara umum, RUL dapat dipahami sebagai indikator kuantitatif yang menjawab pertanyaan "berapa lama lagi suatu komponen dapat beroperasi sebelum mengalami kegagalan fungsional", dan menjadi parameter inti dalam setiap sistem predictive maintenance [18]. *Remaining Useful Life* (RUL) didefinisikan secara teknis sebagai interval waktu, durasi, atau jumlah siklus operasional yang tersisa sebelum suatu komponen atau sistem mencapai ambang batas kegagalan fungsional yang disebut sebagai End of Life (EOL) [18, 19]. Dalam konteks operasional,

estimasi RUL merupakan jantung dari strategi *Prognostics and Health Management* (PHM) yang bertujuan untuk mengantisipasi progres kerusakan berdasarkan kondisi saat ini guna menentukan kapan suatu aset tidak lagi mampu menjalankan fungsi yang diharapkan [14]. Pengukuran sisa umur ini secara matematis dapat dirumuskan melalui persamaan sederhana:

Rumus 2.1 menunjukkan rumus konseptual RUL

$$RUL = T_{EOL} - T_{cc} \tag{2.1}$$

di mana T_{cc} merepresentasikan jumlah siklus kapasitas saat ini dan T_{EOL} adalah titik siklus ketika kapasitas menyentuh ambang batas *End of Life* (EOL) [18].

Prognostics (prognostik) adalah cabang keilmuan dalam kerangka *Prognostics and Health Management* (PHM) yang berfokus pada estimasi kondisi kesehatan suatu sistem atau komponen pada saat ini, serta memprediks sisa umur operasionalnya sebelum mencapai kondisi kegagalan atau *End of Life* (EOL), berdasarkan analisis data historis dan kondisi operasional terkini. Prognostik berbeda dengan diagnostik, di mana diagnostik berfokus pada deteksi dan identifikasi kegagalan yang telah terjadi pada suatu sistem, sedangkan prognostik bersifat prediktif dengan tujuan mengantisipasi kegagalan sebelum terjadi sehingga tindakan pemeliharaan dapat direncanakan lebih awal.

Prognostics and Health Management (PHM) merupakan kerangka kerja yang mengintegrasikan proses *condition monitoring*, diagnostik, dan prognostik untuk mendukung pengambilan keputusan pemeliharaan berdasarkan kondisi aktual komponen. Estimasi *Remaining Useful Life* (RUL), yang menjadi fokus utama dalam penelitian ini, merupakan salah satu keluaran utama dari proses prognostik dalam kerangka PHM.

Tantangan utama dalam pemodelan RUL adalah karakteristik degradasi yang bersifat tidak linear dan tidak seragam di mana performa mesin sering kali tidak langsung menurun sejak awal siklus operasi [19]. Secara praktis, pada fase awal penggunaan, kondisi kesehatan mesin jet cenderung stabil dan perubahan sensorik yang terjadi sering kali dianggap sebagai *noise* operasional alih-alih sinyal degradasi nyata [9, 19]. Oleh karena itu, penentuan titik awal degradasi menjadi faktor penentu yang sangat krusial [9, 19]. kegagalan dalam menentukan titik ini akan mengakibatkan penurunan akurasi prediksi secara signifikan karena model dipaksa mempelajari pola degradasi pada data mesin yang sebenarnya masih sehat [9].

Untuk mengatasi ambiguitas pada fase awal tersebut, riset modern menerapkan pendekatan Piecewise Linear Degradation Model [19, 9]. Dalam model ini, nilai RUL pada tahap awal operasi ditetapkan sebagai konstanta (RUL_{max}) untuk memodelkan fase mesin sehat sempurna, dan baru akan menurun secara linear setelah sistem melewati titik atau ambang batas degradasi tertentu [19]. Pembentukan RUL_{max} dapat dilihat pada rumus 2.2

$$\text{RUL}_t = \begin{cases} \text{RUL}_{max}, & \text{jika RUL}_t > \text{RUL}_{max} \\ \text{RUL}_t, & \text{jika RUL}_t \leq \text{RUL}_{max} \end{cases} \tag{2.2}$$

Pendekatan ini sangat efektif untuk menstabilkan proses pelatihan algoritma Machine Learning, karena mencegah model dari gangguan data sensor yang belum menunjukkan tren degradasi signifikan [9]. Keakuratan estimasi RUL memiliki implikasi langsung terhadap aspek ekonomi dan keselamatan operasional; informasi sisa umur yang presisi memungkinkan tim pemeliharaan melakukan perencanaan logistik secara proaktif dan menghindari waktu henti mesin yang tidak terencana (*unplanned downtime*) yang dapat menyebabkan kerugian finansial besar hingga kegagalan katastropik [9, 14]. Dengan demikian, relevansi riset RUL tidak hanya terletak pada pencapaian akurasi statistik, tetapi pada kemampuannya memberikan peringatan dini yang andal dan objektif bagi pengambil keputusan di industri yang memiliki tingkat risiko tinggi seperti penerbangan [9, 14].

2.3 Predictive Maintenance

Terdapat tiga strategi utama pemeliharaan mesin yang dikenal dalam manajemen aset industri, yaitu Corrective Maintenance (CM), Preventive Maintenance (PvM), dan Predictive Maintenance (PdM), yang masing-masing berbeda dalam hal pemicu tindakan dan dasar pengambilan keputusan perawatan [12]. *Predictive Maintenance* (PdM) mengandalkan integrasi antara teknologi *Internet of Things* (IoT) dan algoritma *Machine Learning* (ML) untuk mengubah data sensor mentah menjadi informasi yang proaktif bagi tim pemeliharaan[12, 20, 21]. *Corrective Maintenance* (CM) yang bersifat reaktif dan hanya melakukan perbaikan setelah kerusakan fisik terjadi, atau *Preventive Maintenance* (PvM) yang sering kali menyebabkan pemborosan sumber daya karena penggantian suku cadang berbasis jadwal tetap tanpa melihat kondisi aslinya, PdM menawarkan solusi yang lebih fleksibel dan optimal secara biaya [9, 15].Sebagai evolusi cerdas dari

Condition-Based Maintenance (CBM), PdM menggunakan pemantauan parameter seperti getaran, suhu, dan tekanan secara terus-menerus untuk mendeteksi anomali pada tahap awal degradasi [20, 22]. Dengan memodelkan tren kerusakan ini, perusahaan dapat memperkirakan *Remaining Useful Life* (RUL) suatu komponen, sehingga tindakan pemeliharaan dapat dijadwalkan tepat sebelum kegagalan fungsi terjadi [15, 22].

Tabel 2.2 merangkum perbedaan karakteristik dasar dari ketiga strategi tersebut berdasarkan literatur:

Tabel 2.2. Perbandingan Strategi Maintenance Berdasarkan Karakteristik Utama

Karakteristik	**Corrective**	**Preventive**	**Predictive**
Pemicu Tindakan	Terjadi Kegagalan [23]	Jadwal Rutin [23]	Prediksi Kondisi [23]
Dasar Keputusan	Kerusakan Komponen [23]	Historis Kegagalan [23]	Kondisi *real-time* [22]
Efisiensi	Rendah [24]	Rendah [23]	Tinggi [22]

Sumber: Diolah dari berbagai sumber [22, 23, 24]

Dalam konteks Industri 4.0, PdM menempati peran strategis dalam meningkatkan efisiensi operasional dan keselamatan kerja melalui sistem manajemen berbasis pengetahuan [20, 22]. Pemanfaatan sistem *machine learning* memungkinkan pengolahan data bervolume besar secara *real-time* untuk mendukung pengambilan keputusan yang lebih akurat dan terdesentralisas [20, 21]. Secara ekonomi, penerapan strategi PdM yang efektif dilaporkan dapat mengurangi biaya pemeliharaan sebesar 25% hingga 30% dan meminimalkan kerusakan aset hingga 75% [21]. Dengan demikian, PdM bertindak sebagai instrumen penciptaan nilai yang mampu memaksimalkan siklus hidup peralatan sekaligus menjamin ketersediaan mesin yang lebih tinggi di lingkungan industri yang kompetitif [9, 20]. Berdasarkan keunggulan efisiensi dan reduksi biaya tersebut, penelitian ini secara khusus difokuskan pada pengembangan komponen prediktif dari Predictive Maintenance, yaitu estimasi RUL berbasis *machine learning* [9].

2.4 Data Sensor dan Time-Series dalam Predictive Maintenance

Data sensor dalam Predictive Maintenance (PdM) dikategorikan sebagai data multivariat yang mencakup parameter fisik esensial seperti getaran, suhu, tekanan, dan arus listrik dari berbagai perangkat IoT [21, 25]. Kumpulan data ini umumnya berbentuk *time-series* yang merepresentasikan kondisi kesehatan mesin

selama siklus hidup operasionalnya secara kontinu [21, 25]. Realitas di lapangan menunjukkan bahwa data yang dihasilkan perangkat IoT sering kali terkontaminasi oleh *noise* atau *noise* operasional yang dapat mengaburkan sinyal degradasi yang sebenarnya[21]. Kumpulan informasi yang sangat masif dan kompleks ini menuntut strategi pengolahan data yang canggih guna mengatasi tantangan terkait volume serta struktur data yang sering kali tidak terlabel [25, 21].

Proses pemantauan menghadapi tantangan berat karena karakteristik degradasi mesin bersifat non-linear dan sering kali menunjukkan pola keausan yang terjadi secara bertahap namun tidak seragam[21]. Ketidakpastian dalam tren kerusakan ini membuat identifikasi anomali menjadi sangat sulit dilakukan tanpa pemahaman mendalam mengenai ketergantungan antar variabel sensor yang kompleks[25]. Pemodelan *time-series* menjadi instrumen yang sangat penting dalam PdM untuk menangkap ketergantungan temporal di mana status mesin saat ini sangat dipengaruhi oleh riwayat operasional sebelumnya [21, 25]. Keberhasilan dalam mengekstraksi informasi dari urutan data sensor yang memiliki ketergantungan jangka panjang merupakan fondasi utama bagi model untuk memprediksi kegagalan secara presisi [21, 25].

2.5 Machine Learning dalam Predictive Maintenance

Konsep pemeliharaan berbasis *Machine Learning* (ML) adalah pendekatan berbasis data (data-driven) yang melatih perangkat lunak untuk mengenali pola dari data historis guna membuat prediksi masa depan tanpa perlu diprogram secara eksplisit untuk setiap skenario [20]. Jika dibandingkan dengan model fisik yang membutuhkan pemahaman mekanis sangat dalam dan sering kali sulit diterapkan pada sistem kompleks karena keterbatasan matematis, ML menawarkan fleksibilitas yang lebih tinggi dan generalisasi yang lebih baik [9]. Kelebihan utama ML terletak pada kemampuannya mengolah data sensor dengan dimensi tinggi serta menangkap hubungan non-linear yang rumit antar variabel [20, 26]. Contoh pendekatan ML yang umum digunakan untuk prediksi RUL meliputi Artificial Neural Networks (ANN), Support Vector Machine (SVM), Random Forest (RF), dan Multi-Layer Perceptron (MLP)[20]. Salah satu pendekatan Machine Learning yang banyak digunakan untuk memodelkan data deret waktu dalam prediksi RUL adalah LSTM, yang mampu menangkap ketergantungan temporal jangka panjang pada data sensor [20].

2.6 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu varian arsitektur dari keluarga Recurrent Neural Network (RNN) yang artinya, secara struktural setiap LSTM adalah RNN, namun tidak semua RNN adalah LSTM [14]. LSTM dikembangkan secara spesifik untuk mengatasi kelemahan mendasar RNN konvensional dalam memproses data deret waktu (time-series), yaitu permasalahan vanishing gradient, melalui penambahan mekanisme cell state dan gating yang tidak dimiliki RNN standar [14]. Model ini sangat unggul dalam memproses data sensor pada sistem pemeliharaan prediktif karena kemampuannya memetakan urutan input ke output dengan mempertimbangkan informasi kontekstual yang dinamis [12, 14]. Integrasi mekanisme gerbang (*gating mechanism*) pada LSTM memungkinkan jaringan untuk menentukan informasi mana yang perlu dipertahankan atau dibuang dari memori sistem secara selektif [12, 14]. Karakteristik tersebut membuat LSTM sangat efektif dalam memantau perilaku operasional mesin dengan memanfaatkan seluruh riwayat data sensor historis guna memprediksi perubahan status kesehatan aset secara akurat [14].

Untuk memahami posisi Long Short-Term Memory (LSTM) secara lebih menyeluruh, perlu dijelaskan terlebih dahulu perbedaannya dengan *Artificial Neural Network* (ANN) dan *Recurrent Neural Network* (RNN) [14]. ANN, yang umumnya diimplementasikan sebagai *Multi-Layer Perceptron* (MLP), memproses sinyal pemantauan secara independen untuk memetakan hubungan langsung ke nilai RUL tanpa mempertimbangkan ketergantungan temporal, sedangkan RNN mampu memproses data deret waktu dengan memanfaatkan *hidden state* untuk meneruskan informasi dari langkah waktu sebelumnya ke langkah waktu saat ini [12, 14]. *Hidden state* ini bekerja sebagai unit yang mengandalkan informasi sebelumnya untuk menghasilkan *output* pada waktu saat ini, di mana terdapat koneksi yang berfungsi mempertahankan informasi dari unit tersembunyi sebelumnya untuk digunakan kembali [12, 14].

Meskipun demikian, RNN konvensional sering kali mengalami masalah *vanishing gradient* selama proses pelatihan [14]. Fenomena ini merupakan kondisi di mana nilai gradien mengecil secara drastis saat proses pembaruan bobot, yang mengakibatkan kecepatan iterasi melambat serta efisiensi konvergensi menurun tajam [14]. Akibatnya, model kehilangan kemampuan untuk mempelajari ketergantungan jangka panjang pada sekuens data yang panjang [14]. Ringkasan perbedaan ketiga arsitektur tersebut disajikan pada tabel 2.3.

Tabel 2.3. Perbandingan Karakteristik ANN, RNN, dan LSTM

Karakteristik	**ANN/MLP**	**RNN**	**LSTM**
Kesadaran terhadap urutan waktu	Tidak ada	*hidden state*	*hidden state* dan *cell state*
Kemampuan mempelajari dependensi jangka panjang	Tidak mendukung	Terbatas (rentan *vanishing gradient*)	Lebih baik (mekanisme *gating*)
Mekanisme kontrol informasi	Tidak ada	Tidak ada mekanisme seleksi eksplisit	*Forget Gate*, *Input Gate*, dan *Output Gate*
Kesesuaian jenis data	Data statis	Data sekuensial	Data sekuensial dengan dependensi jangka panjang

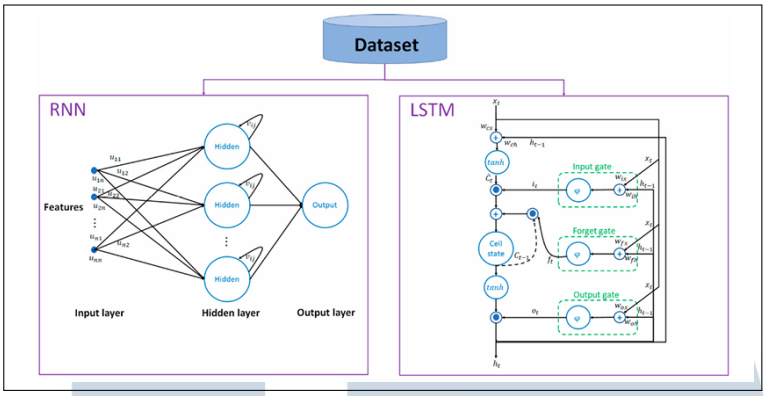
Sumber: Diolah dari berbagai sumber [12, 14].

Efektivitas *Long Short-Term Memory* (LSTM) dalam mengestimasi RUL didasarkan pada keberadaan *cell state* yang memungkinkan model mempertahankan informasi kontekstual dari seluruh data sensor historis secara stabil [12, 14]. Kemampuan memori jangka panjang ini diatur oleh mekanisme *gating* (*forget gate*, *input gate*, dan *output gate*) yang berfungsi menyimpan atau membuang informasi secara selektif untuk mengoptimalkan pembaruan status sel [14]. Arsitektur ini secara teknis dikembangkan untuk mengatasi masalah *vanishing gradient* pada RNN konvensional, di mana gradien kesalahan sering mengecil secara drastis saat proses pelatihan pada sekuens data yang panjang [12, 14]. Melalui struktur tersebut, LSTM mampu mengintegrasikan keuntungan prediksi jangka pendek sekaligus mengelola ketergantungan temporal jangka panjang yang kompleks pada mesin turbofan [12, 14]. Oleh karena itu, penggunaan LSTM menjamin aliran gradien yang lebih konsisten sehingga menghasilkan prediksi degradasi yang lebih akurat dibandingkan model ANN maupun RNN standar [12, 14].

Secara visual, mekanisme internal yang memungkinkan LSTM mengelola informasi jangka panjang tersebut direpresentasikan melalui arsitektur sel yang sangat detail pada (Gambar 2.1)

Struktur ini menunjukkan bagaimana informasi mengalir melalui unit memori dengan kontrol ketat dari tiga gerbang utama. Operasi pertama dilakukan oleh forget gate yang berfungsi menentukan tingkat retensi informasi historis dengan mengeluarkan nilai antara 0 hingga 1 melalui aktivasi sigmoid [12, 14]. Berdasarkan jurnal Chui et al [12]., Proses ini dirumuskan sebagai

Rumus 2.3 rumus *Forget Gate*



Gambar 2.1. Arsitektur Model LSTM untuk Prediksi RUL

Sumber: [12]

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{2.3}$$

Selain *forget gate*, terdapat *input gate* yang berfungsi menentukan seberapa besar informasi baru dari input saat ini akan disimpan ke dalam *cell state*. *Input gate* juga bekerja melalui fungsi aktivasi *sigmoid* sehingga menghasilkan nilai antara 0 hingga 1 sebagai bobot seleksi informasi [9]. Berdasarkan jurnal Asif et al., proses ini dirumuskan sebagai

Rumus 2.4 rumus *Input Gate*

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.4}$$

Bersamaan dengan *input gate*, jaringan juga menghitung kandidat nilai baru yang berpotensi ditambahkan ke dalam *cell state*, yang disebut *candidate cell state*. Berbeda dengan *gate* yang menggunakan *sigmoid*, *candidate cell state* menggunakan fungsi aktivasi tanh (*hyperbolic tangent*) untuk menghasilkan nilai kandidat dalam rentang -1 hingga 1 [9]. Rumus *candidate cell state* dinyatakan sebagai

Rumus 2.5 rumus *candidate cell state*

$$\tilde{C}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{2.5}$$

Nilai i_t dan $\tilde{C}_t$ inilah yang kemudian digunakan bersama nilai *forget gate* f_t pada

Rumus 2.3 untuk memperbarui *cell state* C_t, di mana *forget gate* menentukan porsi memori lama yang dipertahankan sementara *input gate* dan *candidate cell state* menentukan porsi informasi baru yang ditambahkan [9].
Tahap pembaruan status sel ini secara matematis dinyatakan oleh Chui et al [12]. sebagai

Rumus 2.9 rumus *Memory Cell Update*

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

Setelah *cell state* diperbarui, LSTM menghitung *output gate* untuk menentukan bagian dari *cell state* yang akan diteruskan sebagai output pada langkah waktu tersebut. Sama seperti *forget gate* dan *input gate*, *output gate* menggunakan fungsi aktivasi *sigmoid* [9]. Rumusnya dinyatakan sebagai

Rumus 2.7 rumus *output gate*

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

Nilai *output gate* tersebut kemudian dikalikan dengan *cell state* yang telah difilter melalui fungsi tanh untuk menghasilkan *hidden state* h_t, yaitu *output akhir* dari sel LSTM pada langkah waktu saat ini yang juga akan diteruskan sebagai bagian dari input pada langkah waktu berikutnya. Rumusnya dinyatakan sebagai Rumus 2.8 rumus *hidden state*

$$h_t = o_t \cdot \tanh(C_t) \tag{2.8}$$

Pemilihan fungsi aktivasi pada setiap gerbang LSTM tidak dilakukan secara acak, melainkan disesuaikan dengan fungsinya masing-masing. Fungsi sigmoid dipilih untuk *forget gate*, *input gate*, dan *output gate* karena menghasilkan keluaran dalam rentang 0 hingga 1, yang secara konseptual berperan sebagai "saklar" atau bobot seleksi, yaitu nilai mendekati 0 berarti informasi tersebut diabaikan, sedangkan nilai mendekati 1 berarti informasi tersebut dipertahankan sepenuhnya [9]. Sebaliknya, fungsi tanh digunakan pada *candidate cell state* dan pada tahap akhir perhitungan *hidden state* karena mampu menghasilkan nilai dalam rentang -1 hingga 1, sehingga dapat merepresentasikan baik penguatan maupun pelemahan suatu informasi secara seimbang di sekitar titik nol, sekaligus membantu menjaga

kestabilan aliran gradien selama pelatihan berlangsung [9]. Kombinasi kedua fungsi aktivasi inilah yang memungkinkan LSTM mengatur aliran informasi secara selektif sekaligus mempertahankan kestabilan numerik pada proses pembelajaran sekuens data yang panjang.

Tahap pembaruan status sel ini secara matematis dinyatakan oleh Chui et al. sebagai

Rumus 2.9 rumus *Memory Cell Update*

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.9}$$

Koordinasi matematis antara pembuangan informasi lama dan pembaruan informasi baru inilah yang menjamin model LSTM tetap memiliki memori jangka panjang yang stabil dan akurat dalam mengevaluasi tren degradasi mesinjet turbofan maupun komponen rolling bearing [12, 14].

2.7 EWM (Exponential Weighted Moving Average)

Exponential Weighted Moving Average (EWM) merupakan salah satu metode penghalusan (*smoothing*) data deret waktu yang memberikan bobot lebih besar pada data yang lebih baru dibandingkan data yang lebih lama sehingga mampu mempertahankan informasi tren sekaligus mengurangi pengaruh fluktuasi jangka pendek pada data sensor [13].

Berbeda dengan *Simple Moving Average* (SMA) yang memberikan bobot yang sama pada setiap data dalam suatu jendela pengamatan, EWM menggunakan bobot eksponensial sehingga perubahan terbaru pada data memiliki pengaruh yang lebih besar terhadap nilai rata-rata yang dihasilkan [13]. Secara matematis, nilai EWM pada waktu ke-t dapat dihitung menggunakan persamaan berikut [27].

Rumus 2.10 menunjukan rumus EWM

$$S_t = \alpha x_t + (1-\alpha) S_{t-1} \tag{2.10}$$

dengan S_t merupakan nilai hasil penghalusan pada waktu ke-t, x_t merupakan nilai observasi pada waktu ke-t, S_{t-1} merupakan nilai EWM pada waktu sebelumnya, sedangkan α merupakan faktor penghalusan (*smoothing factor*) yang bernilai $0 < \alpha \leq 1$ [13].

Nilai parameter α menentukan tingkat sensitivitas hasil penghalusan

terhadap perubahan data terbaru [13]. Nilai α yang lebih besar akan menghasilkan keluaran yang lebih responsif terhadap perubahan data, sedangkan nilai α yang lebih kecil menghasilkan proses penghalusan yang lebih kuat sehingga fluktuasi data dapat dikurangi secara lebih efektif [13].

Pada penelitian prediksi *Remaining Useful Life* (RUL) menggunakan dataset NASA C-MAPSS, penerapan EWM dilakukan sebagai tahap *preprocessing* untuk mengurangi *noise* pada pembacaan sensor mesin pesawat [13]. Data sensor pada dataset C-MAPSS memiliki fluktuasi yang cukup tinggi sehingga tren degradasi komponen sulit diamati secara langsung apabila digunakan tanpa proses penghalusan [13]. Dengan menerapkan EWM, variasi jangka pendek pada sinyal sensor dapat direduksi sehingga pola degradasi jangka panjang menjadi lebih jelas dan mudah dipelajari oleh model LSTM [13].

Peringal *et al.* juga menemukan bahwa hasil penghalusan menggunakan EWM pada beberapa titik awal data belum mampu merepresentasikan kondisi sinyal yang sebenarnya [13]. Fenomena tersebut terjadi karena metode EWM memerlukan beberapa observasi awal untuk membentuk rata-rata eksponensial yang stabil sehingga nilai hasil penghalusan pada awal deret waktu masih dipengaruhi oleh proses inisialisasi [13].

Berdasarkan hasil pengamatan tersebut, sepuluh sampel pertama dari setiap data sensor dihapus setelah proses EWM dilakukan [13]. Penghapusan data awal ini bertujuan untuk menghindari penggunaan sinyal yang belum representatif sebagai masukan model sehingga kualitas data pelatihan dapat ditingkatkan [13].

Penggunaan EWM memberikan dampak positif terhadap proses pelatihan model LSTM [13]. Data sensor yang telah dihaluskan menghasilkan pola masukan yang lebih stabil sehingga model mampu mempelajari hubungan temporal (*temporal dependencies*) secara lebih efektif tanpa terganggu oleh fluktuasi data sesaat [13] .Selain itu, proses konvergensi model menjadi lebih cepat karena nilai *loss* mengalami penurunan yang lebih konsisten pada epoch-epoch awal pelatihan [13]. Hasil penelitian Peringal *et al.* juga menunjukkan bahwa penggunaan EWM berkontribusi terhadap peningkatan performa prediksi RUL yang ditunjukkan melalui nilai *Mean Squared Error* (MSE) yang stabil pada subset FD001 [13].

Berdasarkan uraian tersebut, EWM tidak hanya berfungsi sebagai metode penghalusan data, tetapi juga berperan dalam meningkatkan kualitas representasi fitur sensor sebelum diproses oleh jaringan LSTM sehingga model dapat mempelajari pola degradasi mesin secara lebih akurat dan stabil [13].

2.8 MinMaxScaler

MinMaxScaler merupakan salah satu metode normalisasi data yang mengubah nilai setiap fitur ke dalam rentang tertentu, umumnya antara 0 hingga 1, dengan tetap mempertahankan hubungan proporsional antar data pada setiap fitur [13]. Normalisasi ini bertujuan untuk menyamakan skala seluruh fitur sehingga tidak ada fitur yang memiliki pengaruh lebih besar hanya karena memiliki rentang nilai yang lebih tinggi dibandingkan fitur lainnya [13].

Pada jaringan saraf tiruan, khususnya *Long Short-Term Memory* (LSTM), normalisasi data menjadi tahapan yang sangat penting karena fungsi aktivasi seperti *sigmoid* dan *hyperbolic* tangent (*tanh*) bekerja secara optimal ketika masukan berada pada rentang nilai yang relatif kecil [13]. Apabila data sensor memiliki rentang nilai yang sangat berbeda, proses optimisasi menjadi kurang stabil sehingga konvergensi model dapat berlangsung lebih lambat dan performa prediksi menurun [13].

Secara matematis, proses normalisasi menggunakan MinMaxScaler dinyatakan dengan Persamaan 2.11 [13].

Rumus 2.11 menunjukan rumus *MinMaxScaler*

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2.11}$$

dengan x merupakan nilai asli suatu fitur, $x_{\min}$ merupakan nilai minimum fitur pada data pelatihan, $x_{\max}$ merupakan nilai maksimum fitur pada data pelatihan, sedangkan x' merupakan hasil normalisasi yang berada pada rentang 0 hingga 1 [13].

Proses normalisasi menggunakan MinMaxScaler perlu dilakukan dengan hati-hati agar tidak menyebabkan *data leakage*. *Data leakage* merupakan kondisi ketika informasi dari data validasi atau data uji secara tidak sengaja digunakan selama proses pelatihan model sehingga menghasilkan performa evaluasi yang terlalu optimistis dan tidak merepresentasikan kemampuan generalisasi model pada data yang benar-benar baru [13].

Pada MinMaxScaler, nilai minimum ($x_{\min}$) dan maksimum ($x_{\max}$) yang digunakan pada Persamaan 2.11 harus dihitung hanya dari data pelatihan melalui proses *fitting*. Selanjutnya, parameter tersebut digunakan kembali untuk mentransformasikan data validasi maupun data uji tanpa dilakukan proses *fitting*

ulang [13]. Apabila nilai minimum dan maksimum dihitung menggunakan keseluruhan data, termasuk data validasi atau data uji, maka informasi statistik dari data tersebut akan ikut memengaruhi proses normalisasi sehingga menyebabkan *data leakage* [13]. Oleh karena itu, penggunaan parameter normalisasi yang diperoleh hanya dari data pelatihan menjadi salah satu langkah penting untuk menjaga validitas proses pelatihan dan evaluasi model [13].

Berdasarkan Persamaan 2.11, nilai minimum suatu fitur akan dipetakan menjadi 0, sedangkan nilai maksimum dipetakan menjadi 1, sementara nilai-nilai lain akan dipetakan secara linier di antara kedua batas tersebut [13]. Dengan demikian, distribusi relatif setiap fitur tetap dipertahankan meskipun skala nilainya telah diseragamkan [13].

Pada penelitian Peringal *et al.*, proses MinMaxScaler dilakukan setelah tahap seleksi fitur dan sebelum data dibentuk menjadi sekuens deret waktu (*time-series sequences*) sebagai masukan model LSTM [13]. Sebelum proses normalisasi dilakukan, beberapa sensor yang memiliki nilai konstan selama operasi mesin dieliminasi karena tidak memberikan informasi yang bermanfaat terhadap proses prediksi [13]. Sensor yang dihapus meliputi sensor 1, 5, 6, 10, 16, 18, dan 19 sehingga hanya fitur sensor yang mengandung informasi degradasi mesin yang digunakan pada tahap normalisasi [13].

Selain melakukan normalisasi, penelitian tersebut juga menyimpan parameter normalisasi berupa nilai minimum dan maksimum yang diperoleh dari data pelatihan [13]. Parameter tersebut kemudian digunakan kembali pada saat proses inferensi (*inference*) sehingga data baru dinormalisasi menggunakan rentang yang sama dengan data pelatihan [13]. Pendekatan ini menjaga konsistensi representasi data masukan yang diterima model serta mencegah terjadinya perbedaan distribusi antara data pelatihan dan data pengujian [13].

Penggunaan MinMaxScaler memberikan dampak positif terhadap proses pelatihan model LSTM [13]. Data sensor yang telah dinormalisasi menghasilkan proses optimisasi yang lebih stabil sehingga model mampu mencapai konvergensi lebih cepat selama proses pelatihan [13]. Selain itu, penyamaan rentang nilai antar fitur membantu model mempelajari pola degradasi mesin secara lebih efektif karena seluruh sensor memberikan kontribusi yang lebih seimbang terhadap proses pembelajaran [13]. Hasil penelitian Peringal *et al.* menunjukkan bahwa kombinasi tahap seleksi fitur, normalisasi menggunakan MinMaxScaler, serta pemodelan menggunakan LSTM mampu menghasilkan performa prediksi yang baik dengan nilai *Mean Squared Error* (MSE) sebesar 796,42 pada subset FD001 dataset NASA

C-MAPSS [13].

Berdasarkan uraian tersebut, MinMaxScaler tidak hanya berfungsi sebagai teknik normalisasi data, tetapi juga merupakan tahap pra-pemrosesan yang berperan dalam meningkatkan stabilitas proses pelatihan, mempercepat konvergensi model, serta menjaga konsistensi data masukan sehingga model LSTM dapat mempelajari karakteristik degradasi mesin secara lebih efektif [13].

2.9 Evaluasi Performa RMSE dan *Scoring Function* dalam Prediksi RUL

Evaluasi performa model prediksi RUL merupakan tahapan krusial untuk mengukur efektivitas algoritma dalam memetakan hubungan antara data sensor dan estimasi sisa umur mesin [9]. Proses ini melibatkan penggunaan metrik statistik objektif untuk memvalidasi kemampuan generalisasi model terhadap data uji yang belum pernah dilihat sebelumnya [15]. Pemilihan metrik yang tepat sangat menentukan keandalan keputusan operasional, terutama dalam menyeimbangkan antara efisiensi biaya dan mitigasi risiko keselamatan industri [15]. Tanpa evaluasi yang sistematis, model yang dikembangkan berisiko mengalami overfitting atau memberikan estimasi yang menyesatkan dalam kondisi kritis lapangan [9, 15].

Root Mean Squared Error (RMSE) adalah metrik standar yang paling sering digunakan dalam literatur prognostik untuk mengevaluasi kesalahan prediksi secara keseluruhan [9, 15]. Metrik ini bersifat simetris karena memberikan bobot penalti yang setara terhadap kesalahan estimasi yang terlalu awal maupun yang terlalu lambat [9]. Secara matematis, RMSE dihitung dengan rumus:

Rumus 2.12 rumus *RSME*

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{2.12}$$

di mana e_i merepresentasikan selisih antara RUL aktual dan prediksi pada sampel ke-$_i$ [9]. Penggunaan RMSE memungkinkan peneliti memantau stabilitas model selama fase pelatihan dan memastikan bahwa penyimpangan prediksi tetap berada dalam batas toleransi statistik yang dapat diterima [9].

Selain menggunakan *Root Mean Squared Error* (RMSE), penelitian pada prediksi *Remaining Useful Life* (RUL) juga banyak memanfaatkan *Scoring Function* (SF) sebagai metrik evaluasi [9]. Berbeda dengan RMSE yang memberikan penalti secara simetris terhadap seluruh kesalahan prediksi, *Scoring*

Function merupakan metrik evaluasi yang bersifat asimetris sehingga mampu membedakan tingkat penalti antara prediksi yang terlalu awal (*early prediction*) dan prediksi yang terlambat (*late prediction*) [9]. Karakteristik tersebut menjadikan *Scoring Function* lebih sesuai digunakan pada kasus prediksi RUL karena konsekuensi operasional dari kedua jenis kesalahan tersebut tidak memiliki tingkat risiko yang sama [9].

Pada sistem pemeliharaan prediktif, prediksi yang terlambat memiliki risiko yang jauh lebih besar dibandingkan prediksi yang terlalu awal [9]. Apabila model memprediksi nilai RUL lebih besar daripada kondisi sebenarnya, maka komponen berpotensi mengalami kegagalan sebelum tindakan pemeliharaan dilakukan [9]. Sebaliknya, prediksi yang terlalu awal hanya menyebabkan komponen diganti lebih cepat sehingga berdampak pada peningkatan biaya perawatan, namun tetap berada pada kondisi yang lebih aman [9]. Oleh karena itu, *Scoring Function* dirancang untuk memberikan penalti yang lebih besar terhadap *late prediction* dibandingkan *early prediction* [9].

Metrik ini pertama kali diperkenalkan pada kompetisi *PHM08 Data Challenge* dan hingga saat ini masih menjadi salah satu standar evaluasi pada penelitian prediksi RUL menggunakan dataset *NASA C-MAPSS* [9]. Angka 13 dan 10 adalah konstanta tetap (*fixed constants*) yang sudah ditentukan oleh penyelenggara kompetisi tersebut (*NASA*) untuk standarisasi penilaian Nilai *Scoring Function* dihitung menggunakan Persamaan 2.13.

Rumus 2.13 rumus *Scoring Function*

$$score = \sum_{j=1}^{N} s_j \tag{2.13}$$

dengan fungsi penalti

Rumus 2.14 penalti

$$s_j = \begin{cases} \exp\left(\frac{-h_j}{13}\right) - 1, & \text{jika } h_j < 0 \text{ (prediksi awal)} \\ \exp\left(\frac{h_j}{10}\right) - 1, & \text{jika } h_j \geq 0 \text{ (prediksi terlambat)} \end{cases} \tag{2.14}$$

Rumus 2.15 Menunjukan RUL Prediksi - RUL Aktual

$$h_i = \hat{y}_i - y_i \tag{2.15}$$

dengan $h_i = \hat{RUL}_i - RUL_i$ menyatakan selisih antara nilai RUL hasil prediksi dengan nilai RUL aktual pada sampel ke-i. Apabila $h_i < 0$, model menghasilkan prediksi yang lebih awal sehingga penalti dihitung menggunakan pembagi 13 [9]. Sebaliknya, apabila $h_i \geq 0$, model menghasilkan prediksi yang terlambat sehingga penalti dihitung menggunakan pembagi 10, yang menyebabkan nilai penalti meningkat lebih cepat [9].

Semakin kecil nilai *Scoring Function*, semakin baik performa model dalam memprediksi RUL karena menunjukkan bahwa kesalahan prediksi, terutama *late prediction*, semakin sedikit [9]. Oleh karena itu, metrik ini dianggap lebih representatif dibandingkan RMSE untuk mengevaluasi model prediksi RUL pada sistem yang memiliki risiko keselamatan tinggi, seperti mesin turbofan pesawat, karena secara eksplisit mempertimbangkan konsekuensi operasional dari arah kesalahan prediksi [9].

2.10 Dataset NASA C-MAPSS

Dataset NASA C-MAPSS (*Commercial Modular Aero-Propulsion System Simulation*) adalah dataset hasil simulasi numerik yang memodelkan proses degradasi mesin turbofan secara realistis dari kondisi sehat hingga kegagalan, dikembangkan menggunakan alat simulasi berbasis fisika dalam lingkungan *MATLAB* untuk memodelkan degradasi mesin jet turbofan [9, 5]. Dataset ini pertama kali dipublikasikan pada tahun 2008 untuk konferensi internasional PHM pertama dan tetap menjadi sumber data utama bagi komunitas riset prognostics di seluruh dunia [5]. Simulasi ini merepresentasikan proses degradasi mesin yang sangat realistis melalui pemodelan perambatan kerusakan pada komponen kritis seperti turbin dan kompresor [9]. NASA menyediakan dataset ini sebagai standar publik guna memvalidasi efektivitas berbagai algoritma dalam memprediksi RUL pada sistem siber-fisik yang kompleks [9, 10].

Karakteristik utama dataset ini terletak pada struktur deret waktu multivariat yang mencakup pembacaan dari 21 sensor berbeda dan 3 pengaturan operasional mesin [9, 12]. Setiap baris data mencatat riwayat operasional per siklus (cycle) mulai dari kondisi sehat awal hingga mencapai titik kegagalan (run-to-failure) pada unit pelatihan [9, 19]. Variabel sensor yang tersedia mencerminkan parameter fisik krusial seperti suhu total, tekanan, kecepatan kipas, dan efisiensi aliran gas yang merefleksikan status kesehatan mesin yang tersembunyi [9, 2]. Dataset ini secara

kritis dibagi menjadi empat sub-dataset (FD001 hingga FD004) dengan tingkat kesulitan yang meningkat berdasarkan jumlah kondisi operasi dan mode kegagalan yang disimulasikan [9, 10].

Tabel 2.2 dan 2.3 merangkum deskripsi dari Dataset C-MAPSS dan karakteristik Sub-dataset C-MAPSS:

Tabel 2.4. Deskripsi Atribut Dataset C-MAPSS

Nomor Kolom	Nama Atribut	Deskripsi
1	*Unit Number*	Identitas unik untuk setiap mesin pesawat.
2	*Time (Cycle)*	Jumlah siklus operasional yang telah dilalui.
3–5	*Operational Settings*	Pengaturan lingkungan (ketinggian, *Mach number*, dll).
6–26	*Sensor Measurements*	Pembacaan dari 21 sensor fisik mesin jet.

Sumber: Diolah dari berbagai sumber [9, 12, 19]

Tabel 2.5. Karakteristik Sub-dataset C-MAPSS

Sub-dataset	Kondisi Operasi	Mode Kegagalan	Jumlah Unit (Latih/Uji)
FD001	1	1 (HPC)	100 / 100
FD002	6	1 (HPC)	260 / 259
FD003	1	2 (HPC, Fan)	100 / 100
FD004	6	2 (HPC, Fan)	249 / 248

Pemilihan NASA C-MAPSS dalam penelitian ini didasarkan pada kredibilitasnya dalam mengatasi tantangan akses terhadap data sensor mesin pesawat asli yang umumnya bersifat sangat rahasia [15, 21]. Dataset ini menyediakan data kegagalan yang lengkap dan terstruktur, yang secara praktis mustahil didapatkan dari operasional industri nyata karena risiko keselamatan serta biaya pengujian yang sangat mahal [9]. Dengan tersedianya label RUL aktual pada data uji, dataset ini memungkinkan dilakukannya pengukuran akurasi model secara objektif melalui metrik evaluasi standar seperti RMSE [9, 3]. Penggunaan luas dataset ini dalam literatur akademik memfasilitasi perbandingan performa antar algoritma pembelajaran mendalam secara adil, transparan, dan terukur [9, 12].

Dataset ini sangat sesuai dengan tujuan penelitian ini karena karakteristik deret waktunya yang kompleks menuntut penggunaan arsitektur model yang mampu menangkap ketergantungan temporal jangka panjang [9, 2]. Struktur multivariat dengan dimensi tinggi pada C-MAPSS memberikan tantangan teknis yang relevan untuk menguji ketangguhan model LSTM dalam mengekstraksi fitur degradasi dari data yang bising [9, 25]. Adanya simulasi berbagai kondisi operasional pada sub-dataset FD002 dan FD004 memungkinkan validasi kemampuan generalisasi model terhadap pola degradasi yang tidak seragam antar unit [9]. Secara keseluruhan, legitimasi dataset ini menjamin bahwa hasil prediksi RUL dan evaluasi performa model dalam penelitian ini memiliki dasar ilmiah yang kuat serta dapat direproduksi oleh peneliti lain [15].

Penelitian ini menggunakan subset NASA C-MAPSS FD001 karena memiliki satu kondisi operasi dan satu mode kegagalan, sehingga karakteristik datanya lebih sederhana dibandingkan subset lainnya [9]. Kesederhanaan tersebut memungkinkan penelitian berfokus pada evaluasi kemampuan model LSTM dibandingkan Random Forest (RF), Support Vector Regression (SVR), dan Multi-Layer Perceptron (MLP) tanpa dipengaruhi variasi kondisi operasi yang kompleks [9, 25]. Selain itu, FD001 memiliki tingkat kompleksitas yang lebih rendah sehingga sesuai sebagai dasar dalam pengembangan model prediksi Remaining Useful Life (RUL) [9, 25]. Pemilihan FD001 juga mendukung evaluasi menggunakan Scoring Function (SF), yang memberikan penalti lebih besar terhadap late prediction untuk meningkatkan keandalan prediksi dalam konteks predictive maintenance [9].