Smart Maintenance:

AI-Driven Solutions for Predicting Failures and Boosting Efficiency



Missi Hansson

EC Utbildning

Data Science Project

202410

# Abstract

This project developed an intelligent maintenance system using LSTM networks to predict machine failures, offering a proactive approach to maintenance. Using data from 100 machines collected over 2015, the model aims to improve upon traditional maintenance schedules by providing more accurate and timely predictions. Results demonstrate strong practical value: the model forecasts a machine's Remaining Useful Life (RUL) with a Validation Loss of 0.27, supporting proactive maintenance planning to minimize unexpected downtime and reduce operational costs. The project’s results and code are available on Jakob Rask's GitHub for further application and development.

Abbreviations and Terms  
  
**PmD (Predictive Maintenance)** - Maintenance methods enhanced by predictive data analytics to forecast equipment failures before they occur.

**RUL (Remaining Useful Life)** - The predicted duration for which a machine or component will function before needing repair or replacement.

**LSTM (Long Short-Term Memory)** - A type of recurrent neural network (RNN) capable of learning long-term dependencies in time-series data, often used for predictive maintenance.

**GRU (Gated Recurrent Unit)** - A simpler variant of LSTM designed to handle sequential data efficiently by combining certain elements.

**TCN (Temporal Convolutional Network)** - A neural network designed for sequence modeling that uses convolutions to capture dependencies over time in parallel.

**MSE (Mean Squared Error)** - A measure of prediction accuracy, calculating the average of squared differences between predicted and actual values.

**MAE (Mean Absolute Error)** - A metric that calculates the average of absolute differences between predicted and actual values, providing a straightforward error interpretation.

**RMSE (Root Mean Squared Error)** - The square root of MSE, used to measure the model's prediction error in the same units as the target variable.

**EDA (Exploratory Data Analysis)** - A process of analyzing datasets to summarize their main characteristics, often using visualizations.

**Huber Loss** - A loss function that combines Mean Squared Error (MSE) and Mean Absolute Error (MAE) to manage outliers effectively.

**ReLU (Rectified Linear Unit)** - An activation function commonly used in neural networks, particularly effective for non-sequential data.

**tanh (Hyperbolic Tangent)** - An activation function that compresses input values to a range between -1 and 1, commonly used in LSTM and GRU models.

**SQL (Structured Query Language)** - A programming language used for managing and manipulating relational databases.

**Streamlit** - An open-source app framework used for deploying machine learning models and data science projects in a web interface.

Table of Contents

[Abstract 2](#_Toc181566946)

[Abbreviations and Terms 2](#_Toc181566947)

[1 Introduction 1](#_Toc181566948)

[2 Theory 1](#_Toc181566949)

[2.1 Time Series 1](#_Toc181566950)

[2.2 RNNs for Predictive Maintenance 1](#_Toc181566951)

[2.2.1 LSTM 2](#_Toc181566952)

[**2.2.2** GRU 2](#_Toc181566953)

[2.3 TCN 3](#_Toc181566954)

[2.4 Evaluation Metrics 3](#_Toc181566955)

[2.4.1 Mean Squared Error (MSE) 3](#_Toc181566956)

[2.4.2 Mean Absolute Error (MAE) 3](#_Toc181566957)

[2.4.3 Huber Loss 4](#_Toc181566958)

[3 Methodology 4](#_Toc181566959)

[3.1 Data Collection 4](#_Toc181566960)

[3.1.1 Dataset Overview 4](#_Toc181566961)

[3.2 Exploratory Data Analysis (EDA) 4](#_Toc181566962)

[3.3 Data Preprocessing and Cleaning 5](#_Toc181566963)

[3.3.1 Data Cleaning 5](#_Toc181566964)

[3.3.2 One-Hot Encoding 5](#_Toc181566965)

[3.3.3 Feature Engineering 5](#_Toc181566966)

[3.3.4 Scaling 7](#_Toc181566967)

[3.3.5 Model Development and Optimization 8](#_Toc181566968)

[3.3.6 Training, Validation and Testing 9](#_Toc181566969)

[3.3.7 Model Evaluation 9](#_Toc181566970)

[3.4 Agile Workflow 9](#_Toc181566971)

[3.5 Deployment and Automation 10](#_Toc181566972)

[3.5.1 Automation 10](#_Toc181566973)

[3.5.2 3.6.2 Streamlit App 10](#_Toc181566974)

[3.5.3 GitHub Repository and Documentation 10](#_Toc181566975)

[**4** **Results and Discussion** 11](#_Toc181566976)

[4.1 Feature Importance and Selection 11](#_Toc181566977)

[4.2 Model Performance Comparison 11](#_Toc181566978)

[4.2.1 Sensor Prediction Model Performance 12](#_Toc181566979)

[4.3 Challenges and Limitations 12](#_Toc181566980)

[4.3.1 Limitations of the Current Model: 13](#_Toc181566981)

[5 Conclusions 13](#_Toc181566982)

[5.1 Future Work 13](#_Toc181566983)

[5.2 Self-Evaluation and Critical Reflection 14](#_Toc181566984)

[6 Referenser 15](#_Toc181566985)

# Introduction

In industrial operations, maximizing efficiency and minimizing downtime are essential. Predictive maintenance has transformed traditional maintenance practices by using AI and machine learning techniques, such as time-series forecasting, to predict equipment failures and optimize maintenance schedules. This project focuses on developing a predictive model hosted on Jakob Rask's GitHub, utilizing Long Short-Term Memory (LSTM) networks to analyze sensor data, error logs, and maintenance records from 100 machines recorded throughout 2015.

The goal is to build a model capable of identifying early signs of machine failure, providing actionable insights for proactive maintenance planning. Traditional approaches, like scheduled preventive checks or reactive repairs, lack the precision to assess real-time machine health, often leading to high maintenance costs and unexpected downtimes. By predicting Remaining Useful Life (RUL) and potential failures, this model will help companies make informed maintenance decisions based on real-time data patterns, reducing unplanned breakdowns.

The project addresses the following key questions:

* Can an LSTM model accurately predict the Remaining Useful Life (RUL) of machines?
* How effective is the model in predicting future telemetry data (volt, rotate, pressure, vibration) using past telemetry sequences?

To achieve these goals, the project will explore several sequential models (LSTM, GRU, and TCN) and evaluate their performance using metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Deliverables include a functional predictive maintenance tool that can be periodically updated, enabling reliable, data-driven maintenance planning and optimal scheduling.

This project even includes an automated pipeline that regularly updates data, retrains the model, and deploys improvements to an accessible app for real-time monitoring. This approach ensures continuous model refinement and simplifies predictive maintenance for end users.

# Theory

## Time Series

Time-series data consists of observations collected sequentially over time, with each observation dependent on the preceding ones. This structure allows models to capture patterns, trends, and periodic fluctuations, making time-series analysis essential in fields like predictive maintenance. By identifying changes in sensor readings over time, a model can detect early warning signs of potential failures, enabling proactive intervention. Time-series forecasting techniques are thus critical for predicting Remaining Useful Life (RUL) and anticipating future machine states based on historical data patterns.

## RNNs for Predictive Maintenance

Recurrent Neural Networks (RNNs) are a type of neural network designed to work with sequential data, such as time-series data or text. Unlike traditional neural networks, RNNs have connections that form cycles, which allows them to retain information from previous steps in a sequence. This "memory" enables RNNs to capture dependencies and patterns over time, making them suitable for tasks where the order of data points matters.

RNNs process input one step at a time, passing information forward to influence predictions at the next step. However, standard RNNs struggle with long-term dependencies due to issues like the "vanishing gradient" problem, which makes it hard for them to remember information over long sequences. To address this, advanced RNN variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) were developed, which include mechanisms to better capture and retain long-term dependencies in sequential data.

### LSTM

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) that excel at capturing long-term dependencies in sequential data. Unlike standard RNNs, LSTMs are designed to retain relevant information over longer sequences, making them ideal for tasks like time-series forecasting. LSTM networks achieve this by using a memory cell that can maintain information across time steps. The key innovation in LSTM is its use of three "gates" that control the flow of information in and out of this cell:

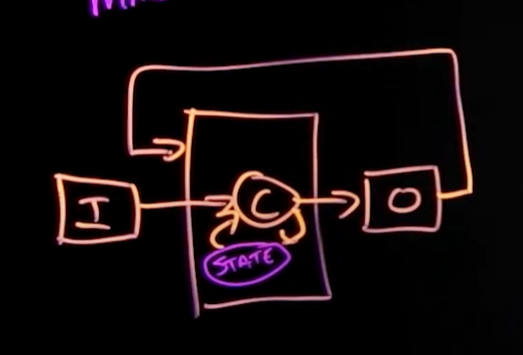


Figure 1 (Keen, 2021)

* Forget Gate: Determines which information from the previous cell state should be discarded. It uses a sigmoid function that outputs values between 0 and 1, where 0 means “forget everything” and 1 means “keep everything.” This gate helps the LSTM decide what information is no longer relevant.

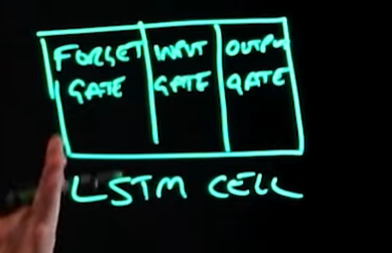


Figure 2 (Keen, 2021)

* Input Gate: Controls how much new information should be added to the cell state. The input gate evaluates new data and decides which values to update, filtering out irrelevant information and ensuring that only the important information is stored in the cell state.
* Output Gate: Decides which information from the cell state should be used as output and passed to the next layer. This gate determines the current hidden state, which is then carried forward and potentially influences predictions at later time steps. (Keen, 2021)

### GRU

Gated Recurrent Units (GRU) are also a variant of RNN)s that, like LSTMs, are designed to handle sequential data and capture dependencies over time. GRUs simplify the architecture of LSTMs by combining certain elements, leading to faster computation and less complexity while often achieving similar performance. GRUs achieve their functionality using two main gates:

* Update Gate: This gate determines the amount of information from the previous hidden state that should be carried forward to the next step. It controls what information should be retained and what should be updated with new data.
* Reset Gate: The reset gate controls how much of the previous information to forget, making the model focus on new input if necessary. When the reset gate value is close to 0, the model ignores the past information almost entirely for the current computation. This feature allows GRUs to effectively handle sequences where some parts of the sequence may need to be disregarded to focus on more recent information. (Pao, 2020)

## TCN

Temporal Convolutional Networks (TCNs) are a type of neural network designed specifically for sequence modeling tasks, like time-series forecasting. Unlike traditional RNNs, which process data step-by-step, TCNs use convolutional layers that allow them to process entire sequences in parallel. This parallelism provides greater efficiency and faster computation, especially beneficial for long sequences. Key Components of TCNs:

* 1D Convolutions: TCNs apply 1D convolutions along the time axis, where each convolutional filter captures patterns over fixed time intervals. This structure helps the model recognize local patterns in time.
* Dilated Convolutions: To capture long-term dependencies, TCNs use dilated convolutions, where filters skip over certain time steps. By increasing the dilation rate at each layer, TCNs expand their "receptive field" exponentially without adding more layers. This enables the model to cover a wider temporal range.
* Causal Convolutions: In TCNs, causal convolutions ensure that predictions at time step *t* are made only using data from time steps *t* and earlier. This maintains the relationship of time data, meaning that future values do not influence the predictions for past or present time steps. (Yadav, 2024)

## Evaluation Metrics

### Mean Squared Error (MSE)

Mean Squared Error (MSE) calculates the average of the squares of errors, where errors are the differences between predicted and actual values. It’s particularly useful because it penalizes larger errors more than smaller ones, making it sensitive to outliers. MSE is defined as:

*where is the actual value, and is the predicted value. is the number of data points.* (Wahlgren, 2015)

### Mean Absolute Error (MAE)

Mean Absolute Error (MAE) calculates the average of the absolute differences between predicted and actual values, treating all errors equally. Unlike MSE, it doesn’t penalize large errors. MAE is defined as:

*where is the actual value, and is the predicted value. is the number of data points.*

It’s more robust to outliers and provides a straightforward interpretation of prediction accuracy in units of the target variable. (Wahlgren, 2015)

### Huber Loss

Huber Loss combines Mean Squared Error (MSE) and Mean Absolute Error (MAE) to handle outliers more effectively. Unlike MSE, which heavily penalizes large errors, Huber Loss is less sensitive to outliers, making it more stable. Its smooth gradient helps with optimization, making it easier for the model to adjust weights during training, which can lead to better performance. Huber Loss strikes a balance between MSE and MAE, reducing the impact of extreme errors while still supporting efficient training. Huber Loss is defined as:

*Where L represents the Huber Loss function. δ is the delta parameter, which determines the threshold for switching between the quadratic and linear components of the loss function, y is the true value or target value, f(x) is the predicted value.* (DataCamp, 2023)

# Methodology

## Data Collection

The dataset used for training our predictive maintenance model was sourced from Azure Predictive Maintenance, available on Kaggle. (Arnab, 2020)

### Dataset Overview

The dataset used for training our predictive maintenance model is sourced from Microsoft Azure and includes detailed data related to the operating conditions, error logs, maintenance history, telemetry readings, and machine metadata for manufacturing equipment.

* Telemetry Data (PdM\_telemetry.csv): Contains hourly averaged sensor readings, including voltage, rotation, pressure, and vibration metrics, collected from 100 machines over 2015.
* Error Logs (PdM\_errors.csv): This file logs non-critical errors recorded while machines were still operational.
* Maintenance Records (PdM\_maint.csv): Logs component replacements for both proactive and reactive maintenance activities. Maintenance data includes records from 2014 and 2015, with timestamps rounded to the nearest hour to align with telemetry data.
* Failure Events (PdM\_failures.csv): This file records instances of component failures, where each failure led to the replacement of the failed component.
* Machine Metadata (PdM\_machines.csv): Contains static information about each machine, such as model type and age.

This dataset includes both continuous (sensor readings) and categorical (error and maintenance logs) data, which enables a comprehensive approach to predicting Remaining Useful Life (RUL) and identifying faulty components.

## Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed on all datasets to understand the data structure, identify key patterns, and detect any potential issues such as outliers or anomalies. Each dataset, including telemetry, error logs, and maintenance records, was analyzed to prepare for feature engineering and model development.

## Data Preprocessing and Cleaning

### Data Cleaning

Since the dataset is synthetic, there are no missing values. A check for duplicate rows was preformed to ensure data quality and removed any duplicates found. Datetime columns were changed to the proper datatype format to facilitate accurate time-based analysis.

### One-Hot Encoding

To enable the model to interpret categorical variables, one-hot encoding was applied to several key columns.

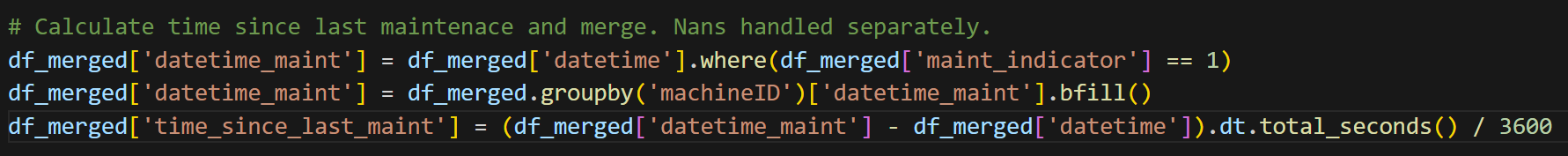
* The failure column was encoded to create binary columns representing different components that failed, and a failure\_indicator column was created to signify any failure occurrence.
* The errorID column was similarly one-hot encoded, resulting in columns representing different error types, with an additional error\_indicator column to indicate the presence of any error.
* The comp column from the maintenance data was encoded to create columns for each component’s maintenance, alongside a maint\_indicator column marking any maintenance event.

### Feature Engineering

For this model, three key features were engineered.

#### Time Since Last Maintenance

This feature calculates the time (in hours) since the machine last underwent maintenance. Maintenance events are essential as they impact the machine’s operating condition and lifespan. Machines tend to perform better immediately after maintenance, with wear and potential failures gradually increasing as time passes. This feature helps the model understand the machine's current state by providing a measure of how “fresh” or “worn” it is since its last maintenance. For each data entry (or row), the elapsed time was calculated from the last maintenance timestamp up to the timestamp of that specific row, resetting to zero every time maintenance occurs. This calculation was done for the entire machine, though it could also be applied on a component level within each machine.



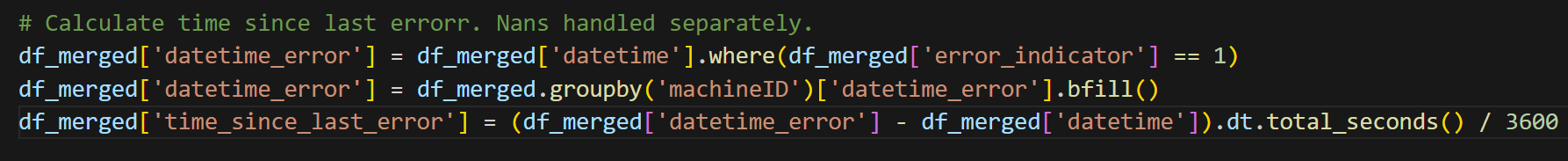
NaN values, which may arise in cases where no prior maintenance event exists, were handled by calculating the mean time since last maintenance for each machine and using this value to fill NaNs. If a machine had no maintenance records at all, an average time since last maintenance was applied across all rows, decrementing gradually until reaching zero, then resetting to the initial mean. This approach ensures that the model has continuous data for each machine, even in the absence of recorded maintenance events.

En bild som visar text, skärmbild, programvara

Automatiskt genererad beskrivning

#### Time Since Last Error

This feature tracks the time since the last error or fault was recorded for the machine. Errors often signal increased stress or potential issues within the system, indicating areas that might require attention. By including this feature, the model can account for recent instabilities or anomalies in the machine’s operation. For each data entry (or row), the elapsed time was calculated from the last error event up to the timestamp of that specific row, resetting to zero with each new error occurrence. Like maintenance, this calculation was done for the entire machine, but it could be adapted to track specific errors within each machine.



In cases where a machine has no recorded errors, NaN values arise in the time since last error feature. These NaNs were handled by calculating the average time since the last error for each machine and filling in NaN values based on this mean. If a machine had no error records at all, the average time since the last error was applied across all rows, decrementing gradually until reaching zero, at which point it reset to the initial mean.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

#### 3. Remaining Useful Life (RUL)

Remaining Useful Life (RUL) is the target feature used in predicting the estimated time remaining before a machine component fails or requires replacement. RUL is crucial as it provides insight into the machine's remaining operational life, enabling proactive maintenance scheduling and helping to prevent unexpected breakdowns. Machines with higher RUL are expected to operate longer without failure, while a lower RUL signals the need for imminent maintenance or replacement.

RUL was calculated similarly to “Time Since Last Maintenance” and “Time Since Last Error,” using the time since the last recorded failure. For rows where no previous failure event exists, the average RUL for that specific machine was used, decreasing with each timestep until it reached zero, signaling a failure. Once zero is reached, the RUL value resets to the machine’s average RUL. For machines without any recorded failures, these instances are treated as anomalies, and the average RUL is assigned based on similar machines in the dataset matching model types and age ranges.

### Scaling

For most features, Min-Max Scaling was used to normalize values to a range between 0 and 1. This method was applied to features like time\_since\_last\_maintenance and time\_since\_last\_error, preserving relative differences while constraining values within a consistent range.

For the Remaining Useful Life (RUL) feature, a logarithmic transformation was applied before scaling to address the skewness in its distribution. As shown in the histograms below, the original RUL distribution was heavily skewed towards lower values, which could hinder the model’s learning process. By applying a log transformation, the RUL distribution becomes more normalized, allowing the model to learn patterns more effectively and reduce the impact of extreme values.

En bild som visar diagram, Graf, linje, text

Automatiskt genererad beskrivningEn bild som visar Graf, diagram, linje, text

Automatiskt genererad beskrivning

After transforming, the scaled RUL values maintain their relative order but have a distribution closer to normal, making the data better suited for modeling. Each feature was scaled based on parameters derived from the training dataset, and these parameters were consistently applied to the validation and test datasets to avoid data leakage.

### Model Development and Optimization

#### Model Architecture and Parameter Tuning

The Bidirectional LSTM network was designed to predict Remaining Useful Life (RUL) by analyzing sequences of time-series data. Each data input has a shape of (time\_steps, features), where time\_steps represents the number of observations included in each sequence fed to the model. For this model, a sequence length of 24 time steps was selected to capture critical temporal trend. Initial exploration of the data suggested that anomalies frequently occurred within the 24 hours leading up to a failure, providing an early indication that a 24-hour sequence could be effective.

The model was trained on a dataset with 15 features, including key telemetry metrics (voltage, rotation, pressure, and vibration), as well as engineered features like time since last maintenance and time since last error.

Subsequent model experiments further validated this choice, as the 24-hour sequence length struck a balance between capturing necessary context and maintaining computational efficiency. Shorter sequences missed key long-term dependencies, while longer sequences introduced additional noise. Selecting an optimal time step length was key to balancing prediction accuracy and efficiency.

Key Components:

* Bidirectional LSTM Layers: The model starts with two bidirectional LSTM layers—128 units in the first and 64 units in the second. The bidirectional setup allows the model to learn from both past and future information within each sequence, improving predictive power.
* Standard LSTM Layers: Two additional LSTM layers with 50 units each follow. This setup captures sequential dependencies, with the final output representing the predicted RUL for each sequence.
* Dropout Regularization: Dropout layers with rates of 0.3, 0.2, and 0.2 are placed between LSTM layers to prevent overfitting. This regularization technique randomly deactivates a percentage of units during training, ensuring the model generalizes well to new data.
* Activation Function: The tanh activation function was chosen over ReLU because it often performs better in LSTM layers, as it mitigates issues related to vanishing gradients when dealing with sequences. For the final dense layer, ReLU was used to produce non-negative values, which aligns better with predicting RUL.

#### Rationale Behind Method Selection

The choice to use sequential models like LSTM, GRU, and TCN was driven by the nature of the data and the goals of predictive maintenance. These models are designed to handle time-series data, where understanding temporal patterns is crucial. LSTM and GRU were selected due to their ability to retain information over long sequences and capture dependencies in sequential data. Specifically, LSTM networks’ memory cell mechanism, including forget, input, and output gates, allows the model to focus on relevant information and discard noise, making it highly suitable for tracking changes in machine condition over time. The simpler GRU was also tested for potential performance improvements due to its fewer parameters and faster training time, which is often beneficial when computational resources are limited.

Temporal Convolutional Networks (TCNs) were included because of their potential for parallel processing and long-range dependency capture through dilated convolutions. TCNs handle sequence data differently than RNNs by capturing temporal patterns across entire sequences, which can increase computational efficiency.

Dropout Regularization was applied to control overfitting, which became apparent during model training. Dropout layers with rates of 0.3 and 0.2 were introduced between LSTM layers to avoid high variance and enhance generalization. By disabling random units during training, dropout regularization improved the model’s robustness on unseen data.

Min-Max Scaling was chosen as a preprocessing technique to normalize data values to a range of 0 to 1, which helps the models converge faster and perform more accurately by ensuring that input features contribute equally to learning. For the RUL feature, a log transformation was added before scaling to reduce skewness, making the model less sensitive to extreme RUL values. This method choice aimed to support the primary performance goal of minimizing error (MSE) for RUL predictions.

### Training, Validation and Testing

The model is trained using the following settings:

* Callbacks:
  + ModelCheckpoint: Saves only the best model based on the lowest validation loss, with the model saved to a unique, time-stamped path.
  + EarlyStopping: Stops training if validation loss does not improve for 5 epochs, restoring the best weights.
  + ReduceLROnPlateau: Reduces the learning rate by half if validation loss does not improve for 3 consecutive epochs, with a minimum learning rate of .
  + Training Parameters: The model is trained for up to 50 epochs with a batch size of 256, using a 3D training dataset X\_train\_seq for inputs and y\_train\_rul\_seq for targets. Validation is performed on X\_val\_seq and y\_val\_rul\_seq.

### Model Evaluation

Performance Metrics: The model's performance was evaluated on a separate test dataset (X\_test\_seq and y\_test\_rul\_seq) using Huber loss and Mean Absolute Error (MAE) as primary metrics. The evaluation process yields both Test Loss (Huber loss) and MAE for RUL predictions, providing insights into the model’s ability to generalize to unseen data.

## Agile Workflow

Agile principles shaped how we managed this predictive maintenance project. We worked in small steps where needed, gathering feedback regularly and adjusting our approach along the way. At the same time, we divided tasks to work on different parts of the project in parallel. This flexible way of working helped us stay on track and adapt quickly as things evolved. Key concepts:

* Kanban Board for Task Management: We used a Kanban board to organize and track tasks. This gave us a clear overview of the project’s progress, showing what was in progress, completed, or still pending.
* Daily Meetings: We kept in touch daily to talk about any obstacles we were facing and set goals for the next few days.
* Working in Parallel: Our team worked on different parts of the project at the same time to cover more ground. One person focused on the LSTM model, others tested TCN and GRU models to see which performed best. We also divided other tasks: one person handled automation scripts, another worked on the Streamlit app, and someone else took care of organizing the GitHub directory and documentation.

## Deployment and Automation

### Automation

The project is organized with modular scripts in the src folder to handle various stages of data processing and model training in order to automate the process to run every 6 months.

Key scripts include:

* CSV\_module.py
* PreProcess\_module.py
* FinalProcessing\_module.py
* LSTMModel\_Trainer\_module.py
* main\_LSTM.py
* SQL\_module.py
* Logging\_module.py

The main\_script.py combines these modules into a pipeline that runs every six months, automatically updating the model and data.

### 3.6.2 Streamlit App

The Streamlit app serves as a user-friendly interface for visualizing predictions and interacting with the predictive maintenance model. Key features include:

* RUL Prediction: Users can select a machine ID to view its predicted RUL along with a comparison of predicted vs. actual values.
* Sensor Data Prediction: Users can check predictions for key sensor metrics (voltage, rotation, pressure, and vibration) for the chosen machine.
* Time Range Selection: Users can specify a time range to view sensor predictions over a selected period, enhancing data exploration.

The app loads pre-trained models and test sequences, displaying results interactively. It is deployed locally, with potential for hosting on cloud platforms for wider access if needed.

### GitHub Repository and Documentation

All code and documentation are maintained in a GitHub repository, organized for easy navigation and version control.

# **Results and Discussion**

## Feature Importance and Selection

The predictive maintenance model relied heavily on a set of key features that were instrumental in forecasting RUL of components accurately. The analysis revealed that maintenance records and sensor readings (volt, rotate, pressure, and vibration) were critical in driving the predictions. Here is an insight into the significance of these features:

* En bild som visar text, skärmbild, diagram, linje

  Automatiskt genererad beskrivningMaintenance Records: These were identified as the most significant predictors. The timing and frequency of maintenance activities provided crucial context for the model, clearly indicating that the health and operational status of components are most effectively predicted through these records. Which is no surprise.
* Sensor Readings: Telemetry data such as voltage, rotation speed, pressure, and vibration levels were essential for real-time monitoring and anomaly detection. These features helped the model detect patterns that preceded failures, allowing for timely predictive maintenance.
* Time Since Last Maintenance: Incorporating the elapsed time since the last maintenance activity for each component proved to be a highly effective strategy. This feature helped the model contextualize the current data point in terms of how long a component had been operating since its last servicing, which is critical for predicting its likelihood of failure. Improvements in the model were particularly noticeable when handling missing data for this feature. By replacing NaN values with the average time since last maintenance per machine, the model's predictions became more robust and accurate, reflecting a more realistic scenario of component wear and tear.
* Determining the optimal time window for aggregating sensor data was key to enhancing model input quality. The analysis showed that a rolling window of 24 to 48 hours struck the best balance between capturing relevant operational trends and maintaining computational efficiency. This window allowed the model to smooth out the noise in the data while retaining critical short-term variations in sensor readings.

## Model Performance Comparison

The performance of two predictive maintenance models was evaluated to determine the most effective approach for RUL prediction. Each model was assessed on key metrics, including validation loss, test loss, Mean Absolute Error (MAE), and prediction consistency. Here’s a summary of the results:

First Model:

Validation Loss: 0.3, Test Loss: 0.5443, Test MAE: 0.8827

This shows reasonable performance, achieving a validation loss of 0.3 MAE was a bit higher, indicating that this model struggled with maintaining consistency, especially in predicting exact RUL values.

Second Model:

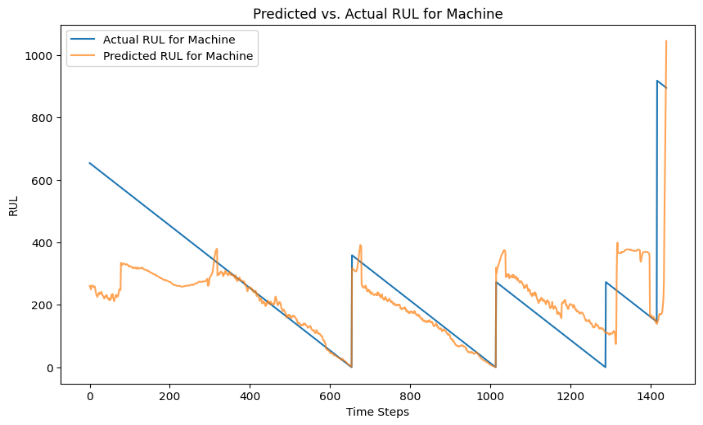
Validation Loss: 0.2712, Test Loss: 0.2654, Test MAE: 0.4793

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivningThe second model demonstrated significantly improved performance, achieving a lower validation loss of 0.2712 and a test loss of 0.2655. The MAE of 0.4793 indicates higher accuracy and stability in RUL predictions. This model’s architecture, with bidirectional layers, allowed it to capture temporal dependencies better, resulting in more consistent and reliable predictions, especially in lower RUL ranges.

En bild som visar skärmbild, text, Graf, linje

Automatiskt genererad beskrivning



Figur 4‑1 Predictions for machine 17

Comparing both models, the second Model outperformed the first Model across critical metrics. The LSTM generalizes well, evidenced by lower test loss and MAE, makes it the preferred model for deployment in predictive maintenance tasks. This model is more robust and provides accurate RUL predictions, with better alignment to actual values and reduced error variability.

### Sensor Prediction Model Performance

The sensor prediction model, focused on forecasting voltage, rotation speed, pressure, and vibration, utilized both GRU and standard LSTM architectures. Initial training showed that the LSTM model performed well, achieving an MSE of 802.7 and a final validation MSE of 795.9 with GRU, indicating comparable results between these models for sensor data.

## Challenges and Limitations

Challenges faced during model training include:

* Data Quality and Imputation: One of the primary challenges was dealing with inconsistencies and gaps in the data, necessitating the imputation of approximately 13% of the RUL values. While necessary, this imputation could introduce bias or inaccuracies in the model's predictions.
* Data Imbalance: The dataset exhibited imbalance, particularly in the frequency and occurrence of component failures, which can lead the model to develop a bias towards more frequently failing components or those with more data points.
* Computational Constraints: Given how complex and large our dataset was, we ran into some computational limits. This meant we couldn’t tweak the parameters as much as we wanted or try out bigger models that might have been more effective. Having access to a more powerful GPU would definitely have helped.

### Limitations of the Current Model:

The model showed signs of overfitting, where it performed well on training data but less so on unseen validation data. Regularization techniques and architecture adjustments attempted to mitigate this issue were not fully successful, suggesting a possible need for a different approach.

* Feature Completeness: There is a potential that not all influential factors are being captured. Missing key features that significantly affect component life could be limiting the model’s ability to make accurate predictions.

# Conclusions

Can an LSTM model accurately predict the RUL of machines?

Yes, an LSTM model can predict the Remaining Useful Life (RUL) of machines with a reasonable level of accuracy. In this project, the Bidirectional LSTM model achieved a test loss of 0.2655 and a mean absolute error (MAE) of 0.4793, indicating that it effectively learned both short- and long-term dependencies in the sensor data. These results demonstrate that the LSTM model can provide reliable RUL predictions, especially for lower RUL ranges. While the model’s accuracy is promising, there is still room for improvement, such as fine-tuning hyperparameters, enhancing feature engineering, or experimenting with alternative architectures like Transformers for potentially higher precision.

How effective is the model in predicting future telemetry data (volt, rotate, pressure, vibration) using past telemetry sequences?

The model is moderately effective in predicting future telemetry data, such as voltage, rotation, pressure, and vibration, by leveraging past telemetry sequences. It successfully captures general trends and patterns over time, which can be useful for high-level insights and identifying anomalies. However, the model lacks the precision needed to predict exact future values for individual sensor readings.

## Future Work

Looking forward, there’s potential to improve the model by enhancing data quality, exploring additional features, and fine-tuning hyperparameters. Expanding the model to predict not only RUL but also which components are likely to fail next would elevate predictive maintenance, allowing for better resource management and operational planning.

To address the inherent complexity of predictive maintenance, future work could focus on:

* Enhanced Data Collection: Gathering more comprehensive data, particularly on underrepresented components or failure events, would create a more balanced dataset and could improve model accuracy.
* Exploring Additional Features and Interactions: Considering more complex features, such as interaction effects or lagged indicators, might reveal hidden patterns critical to improving prediction accuracy.
* Sequence length: Picking the right sequence length and figuring out how much history to include was tricky. If the sequence is too short, important patterns might be missed, but making it too long adds noise and slows things down. With data coming from different sources, like telemetry, error logs, and maintenance records, keeping everything in sync was key for accuracy. Moving forward, testing more precise ways to align these features and fine-tuning the sequence length could help improve the model’s predictions.
* Data Leakage: Avoiding data leakage when building time-based features and splitting data by time was challenging but essential for keeping the model honest. Small tweaks to back-filling and forward-filling techniques could help ensure the model only uses past data during training, preventing any accidental leakage from the future. Strengthening this process would make the model’s predictions more reliable and easier to apply in real-world situations.
* Increased Training and Validation Time: Allocating more time and computational resources to fine-tune parameters could yield better insights and optimizations.

## Self-Evaluation and Critical Reflection

Reflecting on the project, several choices proved impactful, with some areas identified for future refinement:

* Model Architecture: Selecting LSTM as the primary model worked well for tracking RUL over time. However, experimenting with additional layers or alternate configurations, like more complex LSTM or GRU structures, might capture subtler, long-range trends. Trying these options in the future could potentially improve the model’s performance.
* Feature Engineering: Initially, we investigated workload spikes and other sudden changes as potential predictors but didn’t find clear patterns. In future iterations, we could explore interaction effects or create lagged features that capture gradual shifts, as well as add domain-specific indicators of machine stress. These features might help uncover subtle trends that influence RUL.
* Resources and Workflow: With more computational resources, I’d like to further explore hyperparameter tuning and broader model testing. Our Agile approach kept us focused and flexible, though a more structured sprint plan could free up time for additional model evaluations and fine-tuning.

# Referenser

Arnab. (2020). *Kaggle*. Hämtat från Microsoft Azure Predictive Maintenance: https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance

B, P. (den 25 May 2023). *Medium.* Hämtat från Predictive Maintenance of the Process Equipment using Python.: https://medium.com/@pranavbanabakode37/predictive-maintenance-of-the-process-equipment-using-python-d15f95f279ce

DataCamp. (den 4 November 2023). *DataCamp*. Hämtat från Loss Functions in Machine Learning Explained: https://www.datacamp.com/tutorial/loss-function-in-machine-learning

din, N. u. (den 17 January 2023). *Predicting Failure Before it Happens: The Power of Machine Learning in Predictive Maintenance.* Hämtat från Medium: https://medium.com/@naseer1015922/predicting-failure-before-it-happens-the-power-of-machine-learning-in-predictive-maintenance-e271fd8e7396

Gonfalonieri, A. (den 7 November 2019). *Towards Data Science.* Hämtat från How to Implement Machine Learning For Predictive Maintenance: https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860

Hogg, G. (den 5 October 2021). *LSTM Time Series Forecasting Tutorial in Python*. Hämtat från YouTube: https://www.youtube.com/watch?v=c0k-YLQGKjY&t=746s

Keen, M. (den 22 November 2021). *What is LSTM (Long Short Term Memory)?* Hämtat från YouTube: https://www.youtube.com/watch?v=b61DPVFX03I

Pao, S. (den 27 December 2020). *YouTube*. Hämtat från [GRU] Applying and Understanding Gated Recurrent Unit in Python: https://www.youtube.com/watch?v=rdz0UqQz5Sw&t=87s

Peters, M. (den 27 February 2024). *Data & Analytics Blog on Medium.* Hämtat från Unlocking the Power of Predictive Maintenance: How PdM Works and Why It’s Essential: https://medium.com/data-analytics-magazine/unlocking-the-power-of-predictive-maintenance-how-pdm-works-and-why-its-essential-8f4a7f0fc279

Wahlgren, S. K. (2015). *Statistisk Dataanalys.* Lund: Studentlitteratur AB.

Yadav, A. (den 9 Oktober 2024). *Medium*. Hämtat från Temporal Convolutional Network — An Overview: https://medium.com/@amit25173/temporal-convolutional-network-an-overview-4d2b6f03d6f8