Predictive Maintenance

For the manufacturing sector

ECUTBILDNING

Robert Shaw

EC Utbildning

End-to-end Project

202410

Abstract

This report presents the team development of a predictive maintenance model using Long Short-Term Memory (LSTM) networks to predict Remaining Useful Life (RUL) of industrial machinery. The model development followed an agile approach, progressing from data collection and feature engineering to model training and deployment through a Streamlit app. The LSTM model was chosen for its ability to capture long-term dependencies and nonlinear relationships in the time-series data. Results demonstrated reliable RUL predictions, enabling pro-active maintenance planning, thereby reducing unexpected downtime and associated costs. The findings offer a robust framework that can be adapted across various industries where predictive maintenance is key to operational efficiency. While the LSTM model demonstrated substantial predictive capabilities, challenges such as handling sequence dependencies and managing data scaling were notable.

Table of contents

1 Introduction

Predictive maintenance has become a crucial strategy across industries to mitigate the risks and costs associated with unexpected machine failures. With the ability to predict when a machine will likely fail, companies can implement timely maintenance, thus preventing unplanned downtime, reducing maintenance costs, and increasing operational efficiency. This report outlines the development of a predictive maintenance system for industrial machinery, using a dataset from Kaggle. The project follows a comprehensive data science workflow, from data acquisition and storage in SQL, feature engineering, and machine learning model development, to building an interactive user interface using Streamlit.

Our team – Missi Hansson, Jakob Rask and I – set out the goal of this project to predict the Remaining Useful Life (RUL) of machines, thereby enabling proactive maintenance scheduling. Specifically, our analysis answers the following questions:

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

This report provides an overview of the technical implementation, model evaluation, and potential impact of the predictive maintenance system for the manufacturing sector.

1.1 Summary

This report reflects a joint effort by a strong team designed to build a viable time series model collaboratively. It begins by introducing the predictive maintenance challenge, followed by theoretical underpinnings like RUL prediction and LSTM modeling. The methods cover data preparation, model development, and feature engineering. Results are analyzed for predictive accuracy, and conclusions highlight future improvements, deployment strategies, and application insights for manufacturing. The primary model chosen for deployment was a Bidirectional LSTM model, which is the focus of this report. Note that a secondary GRU model was used in deployment which we will reference in chapter 4 on "Results". Predictive maintenance provides a critical advantage in manufacturing by forecasting potential machinery failures before they occur, enabling optimized maintenance planning and resource allocation. By leveraging machine telemetry and maintenance history, the model in this project aims to improve operational uptime and reduce costs associated with unplanned downtimes.

2 Theory

2.1 Time Series Analysis

Time series analysis is all about examining data points collected over time to uncover trends, seasonality, and recurring patterns. Its uniqueness lies in capturing temporal patterns—focusing on how data changes as time progresses, like how machine performance might dip just before a breakdown. Here, the order matters; each data point follows another, creating a flow that lets us predict "what happens next." We can track factors like time since last maintenance to understand long-term dependencies and even use time lag—analyzing how past events affect the future—to make more accurate predictions (Brownlee, J. (2017).

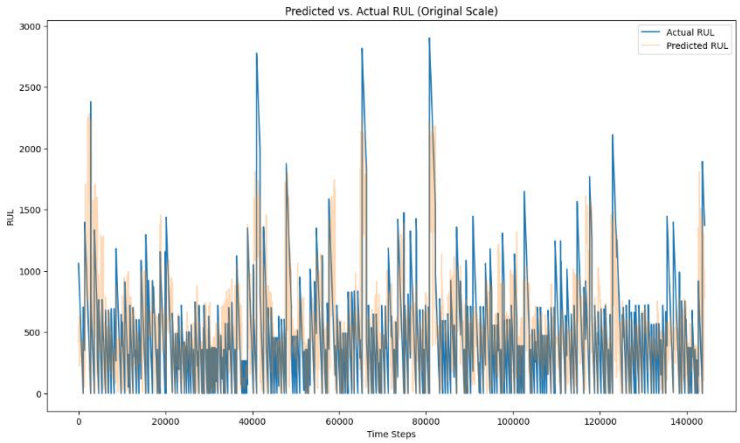
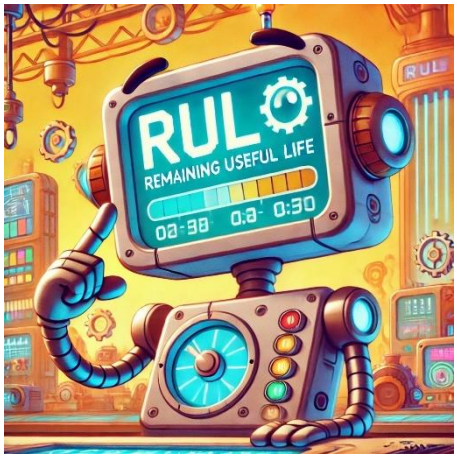


Figure 1

As you can see here, this plot shows the predicted versus actual Remaining Useful Life (RUL) for all 100 machines in our dataset. You can see the temporal patterns in action, with RUL fluctuating over time and showing both short-term shifts and longer-term trends. The differences between the predicted and actual lines also highlight the challenge of capturing those long-term dependencies and time lags. Since each point builds on the previous one, the sequential data structure plays a huge role in how well these predictions line up with reality.

2.1.1 Predictive Maintenance

Predictive maintenance is a pro-active maintenance strategy that utilizes data analysis tools and techniques to predict when equipment failure might occur, so maintenance can be performed just in time to prevent it. By monitoring equipment conditions in real-time using sensors and IoT devices, organizations can detect anomalies and patterns indicative of potential failures. This approach differs from reactive maintenance, which addresses issues post-failure, and preventive maintenance, which schedules maintenance at regular intervals regardless of equipment condition. Predictive maintenance optimizes maintenance schedules, reduces downtime, and extends equipment lifespan by addressing issues before they lead to failure (Mathworks, 2019)



2.1.2 Remaining User Life (RUL) & Window Sequencing

Both these concepts are core issues in predictive maintenance, aimed at breaking down data to allow models to anticipate when a component is likely to fail and to understand conditions leading up to that point.
RUL is calculated by first identifying points where failures have occurred and assigning an RUL of zero to each. From these failure points, RUL values are calculated retrospectively at previous timestamps to represent the

time remaining until each failure, an approach frequently used in predictive maintenance (Fiix Software, 2023). For cases where failure data is missing, placeholders such as 99999 or an average RUL are applied to keep the data clear and consistent for modelling.
To enhance RUL prediction, we use window sequencing to capture time-based patterns in data, like sensor readings and machine conditions leading up to potential failures. A sequence length, such as 24 or 48 hours, is chosen to provide sufficient context without overwhelming the model, and overlapping windows are created to offer sequences with recent data points. This structured sequencing is critical for time-series models like LSTMs, where both short- and long-term trends must be detected (Akshay, Salian, 2023). By applying RUL calculations alongside window sequencing, predictive models are equipped to make accurate predictions and support pro-active maintenance planning.

2.1.3 Bidirectional LSTM Neural Networks

Neural networks, especially Recurrent Neural Networks (RNNs), are powerful tools for time series analysis in predictive maintenance (PdM). RNNs, designed to handle sequential data, capture dependencies over time, making them ideal for tracking machine health. Bidirectional Long Short-Term Memory (LSTM) networks are built on RNNs by processing data in both forward and backward directions, allowing the model to consider both past and future contexts at each time step. This dual perspective is valuable in PdM models, where understanding both recent and upcoming trends improves RUL predictions (Akshay, Salian, 2023).
LSTMs are built around a "cell state," which acts as a kind of memory that can hold information over time. This cell state can store, update, or discard information as it flows through the network.
In each LSTM cell, input data passes through three gates. The *forget gate* first discards unnecessary information. The *input gate* then updates the cell state with new relevant information, and the *output gate* uses this updated state to decide what to pass to the next step. This mechanism allows LSTMs to retain valuable information over long sequences, solving the problem of short-term memory in traditional RNNs, where information can quickly "vanish" (vanishing and exploding gradients) as it passes through each layer.

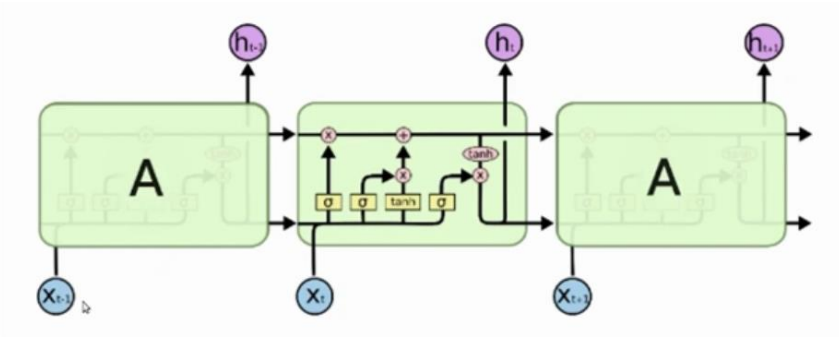


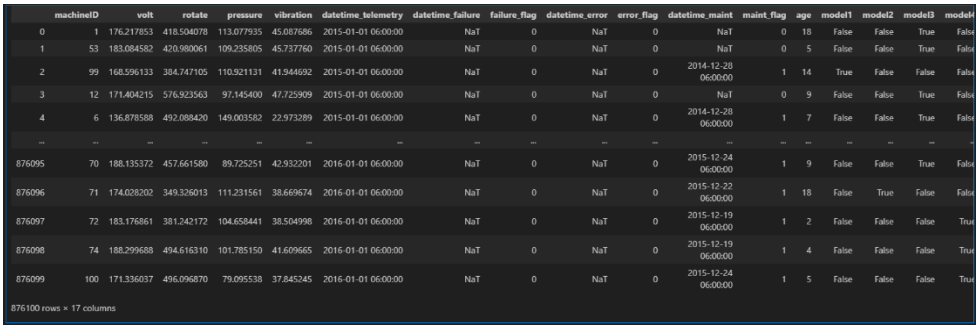
Figure 2

Like other neural networks, LSTMs learn through backpropagation, but because they operate on sequences, they use a variation called Backpropagation Through Time. BPTT calculates the gradient across each time step to minimize errors and adjust weights (Vidhya Analytics, 2024).

3 Methods

3.1 Data collection, Preprocessing and Engineering

The dataset used for this project was sourced from Kaggle, consisting of 5 interrelated files, including telemetry data, maintenance records, machine failure history, and machine metadata. These datasets were ingested into SQL for centralized storage and efficient retrieval during the modelling process. The data includes hourly sensor readings such as voltage, rotation speed, pressure, and vibration from 100 machines, spanning from June 2014 to January 2016. The set has 876100 rows and for our model we used 15 feature columns.



	machineID	volt	rotate	pressure	vibration	datetime_telemetry	datetime_failure	failure_flag	datetime_error	error_flag	datetime_maint	maint_flag	age	model1	model2	model3	model4
0	1	176.217853	418.504078	113.077935	45.087686	2015-01-01 06:00:00	NaT	0	NaT	0	NaT	0	18	False	False	True	False
1	53	183.084582	420.980061	109.235805	45.737760	2015-01-01 06:00:00	NaT	0	NaT	0	NaT	0	5	False	False	True	False
2	99	168.596133	384.747105	110.921131	41.944692	2015-01-01 06:00:00	NaT	0	NaT	0	2014-12-28 06:00:00	1	14	True	False	False	False
3	12	171.404215	576.923563	97.145400	47.725909	2015-01-01 06:00:00	NaT	0	NaT	0	NaT	0	9	False	False	True	False
4	6	136.878588	492.088420	149.003582	22.973289	2015-01-01 06:00:00	NaT	0	NaT	0	2014-12-28 06:00:00	1	7	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
876095	70	188.135372	457.661580	89.725251	42.932201	2016-01-01 06:00:00	NaT	0	NaT	0	2015-12-24 06:00:00	1	9	False	False	True	False
876096	71	174.028202	349.326013	111.231561	38.669674	2016-01-01 06:00:00	NaT	0	NaT	0	2015-12-22 06:00:00	1	18	False	True	False	False
876097	72	183.176861	381.242172	104.658441	38.504998	2016-01-01 06:00:00	NaT	0	NaT	0	2015-12-19 06:00:00	1	2	False	False	False	True
876098	74	188.299688	494.616310	101.785150	41.609665	2016-01-01 06:00:00	NaT	0	NaT	0	2015-12-19 06:00:00	1	4	False	False	False	True
876099	100	171.336037	496.096870	79.095538	37.845245	2016-01-01 06:00:00	NaT	0	NaT	0	2015-12-24 06:00:00	1	5	False	False	False	True

876100 rows × 17 columns

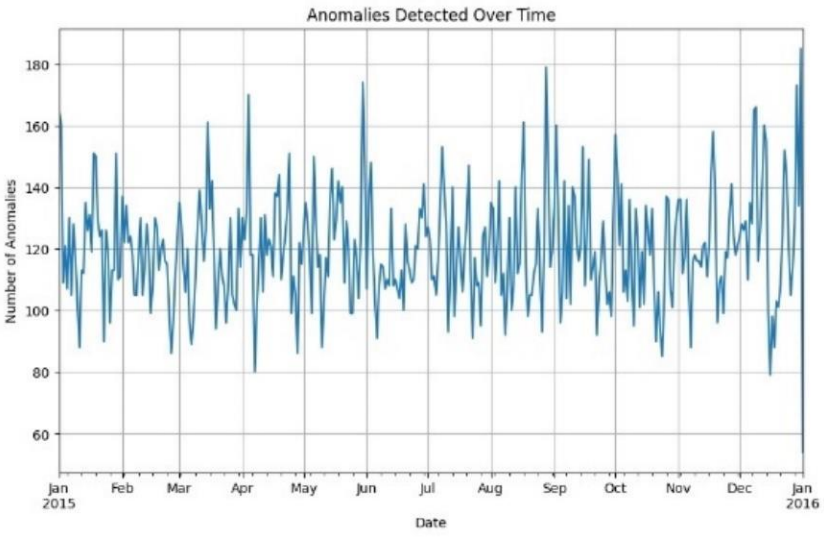
Figure 3

Centralized data storage in SQL enabled efficient retrieval and ensured consistency across modeling iterations. Data preprocessing and engineering were then key to model performance involving:

Each dataset was filtered to retain only 2015 data to ensure consistency. All datetime columns were standardized using pd.to_datetime

One-Hot Encoding: Failures, errors, and maintenance records were one-hot encoded by component and then merged with telemetry data, creating indicators for each type of event. Machine age and model were also added via one-hot encoding for model type, providing additional context for predictive modelling.

Figure 4



Feature Engineering: Based on EDA, we focused on creating time since last maintenance and error long-term dependencies, error/failure/maintenance flags, and lag features for sensor data to improve the LSTM model's ability to detect anomalies and trends over time. Used an isolation forest to create an anomaly flag, then plotted anomalies over time (see plot in Figure 4). Found trends with highest spikes in Q3, and closer analysis shoed a high frequency of anomalies spread across multiple hours in this month, indicating that a longer window (24 to 48 hours) is necessary to capture the build-up of anomalies and their impact on failures.

Handling "Special Missing" Data: We built processing scripts to clean the data. However special missing values emerged in the RUL targets, particularly for machines without recent failure data. These values were imputed using machine-specific means and similarity-based approaches,

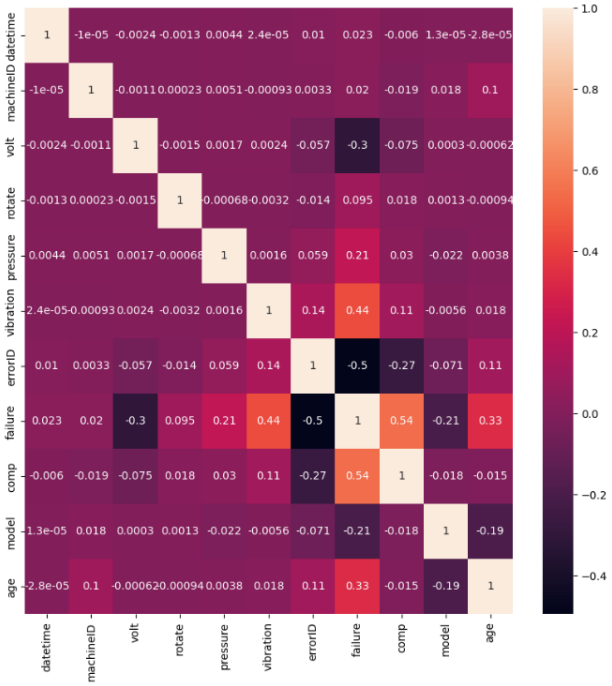
leveraging average RUL values from similar machines in terms of age and model (Stack Exchange, 2023).

3.2 EDA Insights

We carried out a team effort on EDA taking different complimentary approaches to understand the key patterns in the data to help us both create better features as well design a better model.

RUL Distribution: The original RUL histogram (Figure 8 below) showed a right-skewed distribution, with most machines having relatively low RUL values, which as mentioned later we dealt with using Huber loss. **Component Failures**: Certain components failed more frequently (e.g., comp2 had 259 failures), indicating variability in component reliability.

Figure 5



Correlation Analysis: The correlation plot here in Figure 5 shows age as a significant factor, with older machines exhibiting more failures. Sensor data, particularly vibration and pressure, also correlated strongly with failure events, suggesting predictive power in these metrics.

Maintenance Patterns: Maintenance activities seen here in Figure 6 were less frequent in 2014 but became periodic in 2015, though not synchronized across all machines, which could impact failure timing and frequency.

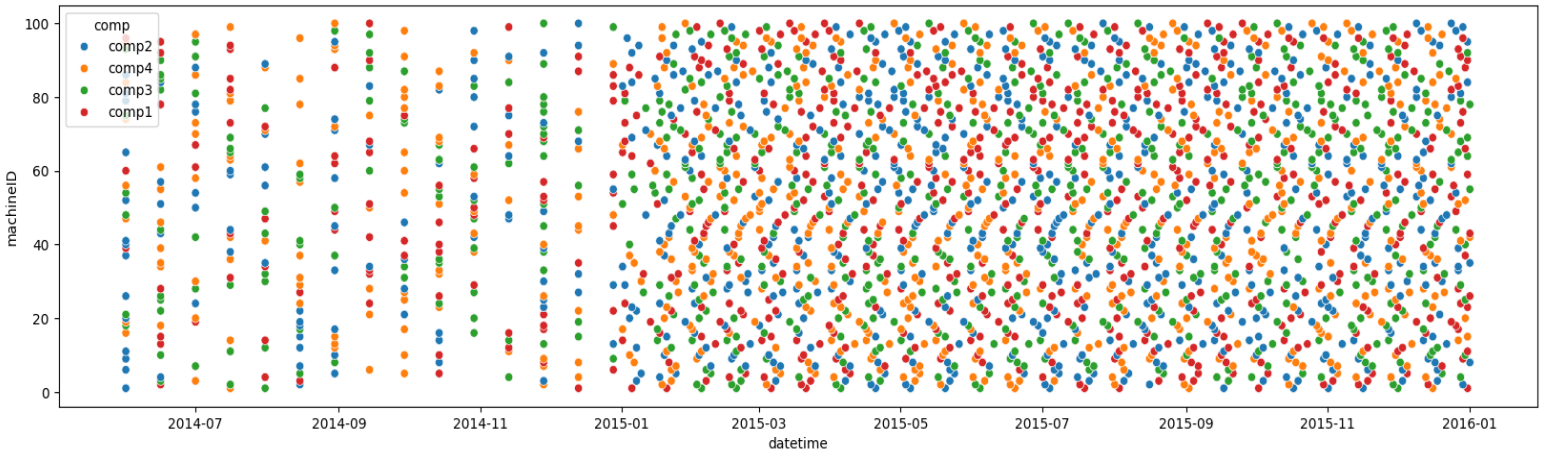


Figure 6

Sensor Distribution & Outliers: Sensor data shows normal distribution with several outliers which we will leave as they are normal part of the data for machines. In the context of predictive maintenance, sensor data often shows a general trend with a degree of natural variability, including outliers. These outliers can represent unusual but normal operating conditions for machinery and don't necessarily indicate faulty data. Retaining these outliers is essential because they can be part of the machine's normal behaviour range and removing them could result in losing meaningful information that contributes to an accurate prediction model (Grover, J., 2021). Our insights are largely confirmed by our Random Forest model here in Figure 7.

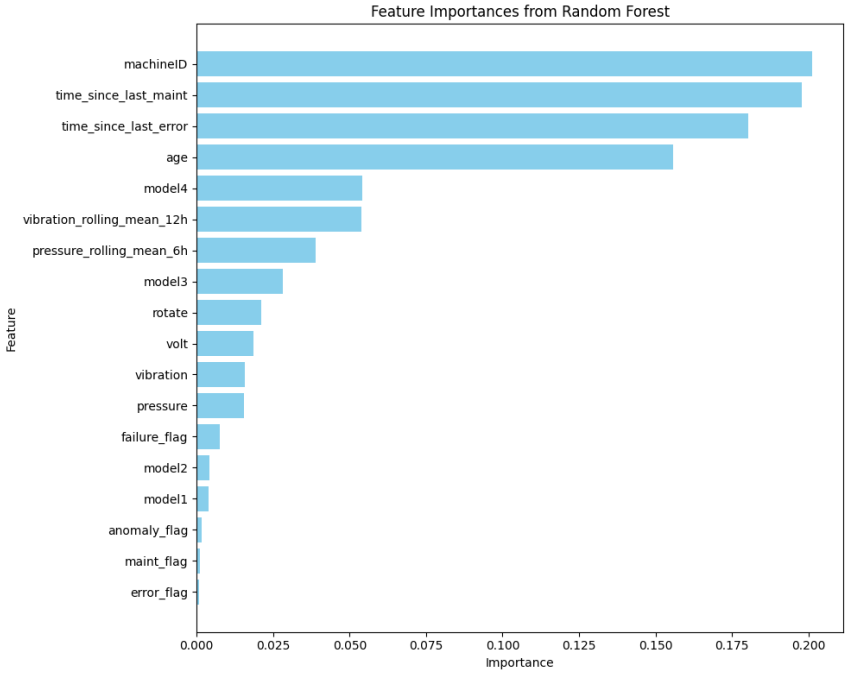


Figure 7

3.3 Model Overview: Preparation & Selection

To prepare the data it was crucial to have ordered the data before splitting it. At first without this, the early models performed poorly.

Train-Test Split: A time-based split was applied to the data:

- Training Set: January to August 2015.
- Validation Set: September to October 2015.
- Test Set: November to December 2015.

This ensures that the model is trained on past data, validated on an intermediate period, and tested on future data, which aligns with real-world deployment scenarios where future events must be predicted.

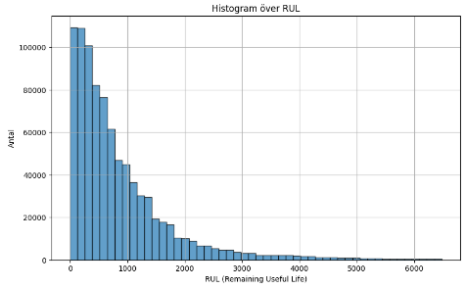


Figure 8

Scaling & Transformation: Time based continuous features were scaled and log transformation (before and after plots here) was applied to the RUL target to create a more normal distribution. This transformation helped improve model stability and generalization.

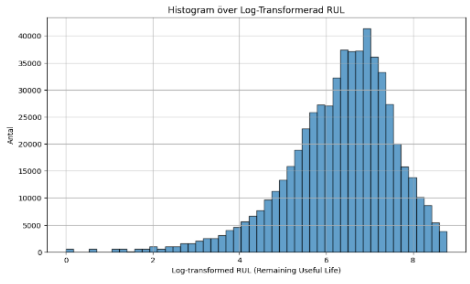


Figure 8a

Sequence Generation: We then transformed the data into sequences suitable for time-series analysis with LSTM. The sequence length was set at 24 to capture temporal trends in the data, with features including telemetry metrics, maintenance, and machine metadata. EDA around anomalies leading to

failures gave us our first clue, with a high frequency of anomalies occurring within 24 hours before a failure. The choice of 24 hours was sealed on model experiments optimizing the balance between context and computational efficiency. Shorter sequences failed to capture longer-term dependencies, while overly long sequences introduced noise. Through validation, a 24-hour sequence length proved most effective, capturing critical temporal patterns without overwhelming the model.

In our **model architecture**, we built it as follows:
An LSTM model to predict the RUL, with:
Input Layer: We fed in the aforementioned sequences of time-series sensor data, machine metadata and engineered long-term dependency time features that will be tracked by a 24-hour time filter.
LSTM Layers: WE followed this up with multiple LSTM layers to capture patterns over time. These layers help the model remember relevant patterns within each sequence.
Dense Layer: Finally, we used a fully-connected layer that takes the output from the LSTM and reduces it to a single prediction, RUL in hours.
Trained the model on historical data, where sequences are labeled with known RUL values. We used Back Propagation Through Time or BPTT to adjust weights in each LSTM cell, helping the model learn long-term dependencies in the sensor data. We applied a Huber **loss function** to handle unbalanced RUL values. Huber loss isn't sensitive to outliers and gives a balanced approach for unbalanced RUL data, enabling the model to generalize better across both frequent low RUL values and specific high values (Chen, Boshun, 2024).

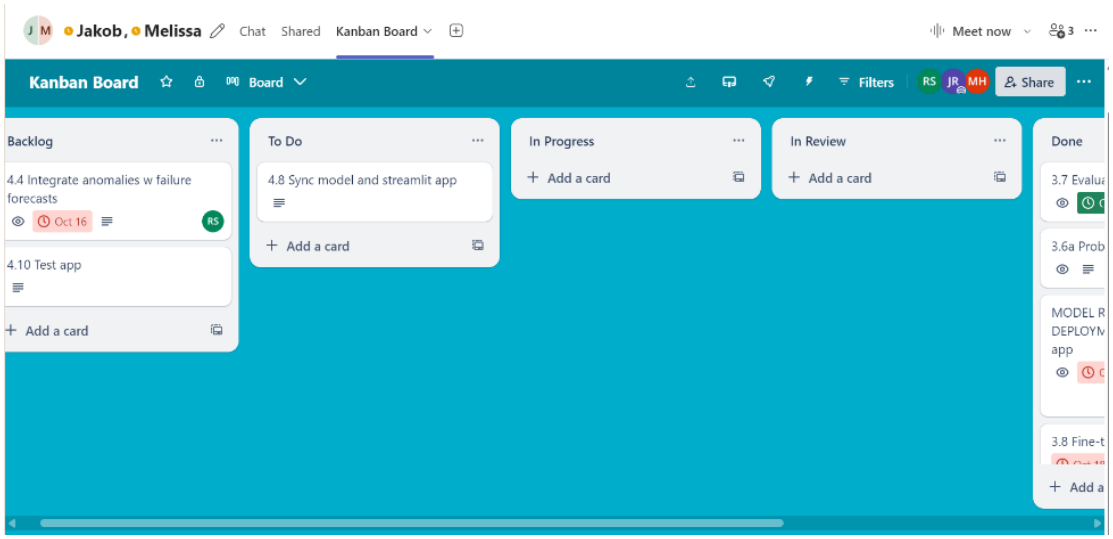
Then we evaluated and deployed the model (see Chapter 4 below) after fine-tuning. This **hyperparameter tuning** with varying learning rates, dropout and batch size was used to both prevent LSTM tendency to overfit, promote convergence and balance computational resources.

We chose an LSTM model given our data is complex and non-linear, and we needed to analyze multivariate sensor data with long-term dependencies. We rejected ARIMA as this models more to model trends and seasonality in univariate sensor data and detects deviations from expected behavior. TCN and GRU as explained below in "Results" weren't going to be great with long-term dependencies and time lags.

3.4 Agile Methodology

In implementing time series analysis for predictive maintenance, our team adopted agile methodologies in line with the Agile Manifesto to enhance collaboration and efficiency. We worked together almost every day, trusted each other to get different jobs done and changed tack when needed to deal with problems. We self-organized sumptuously and gave each other ideas and support. We divided roles to complement each other's strengths—

Figure 9



Robert served as the Scrum Master, while Jakob and Missi took on the roles of Product Owners. Despite these roles, all team members actively participated in the development process, fostering a collaborative environment. This structure facilitated efficient task management, knowledge sharing, and accelerated development cycles. We leveraged agile tools like Trello – see above figure - for tracking progress and Git for version control. Trello allowed us to visualize workflow, assign tasks, and monitor project status in real-time, enhancing transparency and accountability. Git provided a robust platform for collaborative coding and effective file sharing, ensuring that all team members were synchronized and could contribute seamlessly. This agile approach enabled us to respond to changes quickly, prioritize tasks effectively, and deliver a robust predictive maintenance solution grounded in time series analysis.

So, in brief, Agile practices allowed the team to prioritize critical tasks, address bottlenecks efficiently, and adapt to changes in data processing needs. Daily stand-ups and regular reviews facilitated troubleshooting and accelerated responses to technical issues, particularly during model tuning. Trello provided visibility into each team member's contributions, while Git version control ensured smooth collaboration and minimal conflicts.

4 Results and Discussion

Initially, I developed a Temporal Convolutional Network (TCN) model to predict the RUL of machines. However, after observing limitations in capturing long-term dependencies essential for accurate RUL predictions, I decided to switch to an LSTM architecture, which is better suited for sequential data with its ability to retain information over time (Gopali, S., 2021). After evaluating the LSTM's performance, our team decided to use Missi's Bidirectional LSTM model, as it demonstrated superior accuracy and reliability in predicting RUL (see comparison below).

This section presents the results of the models developed by Jakob and Missi. Jakob's focus was on sensor prediction accuracy using GRU and LSTM architectures, while Missi's model specifically targeted RUL prediction with a Bidirectional LSTM. By reviewing each model's performance metrics and visual results, we assess their strengths, practical applications, and contributions to a predictive maintenance (PdM) setting.

4.1 Sensor Prediction Model Performance

Jakob's sensor prediction model used both GRU and LSTM architectures to forecast key sensor readings—volt, rotate, pressure, and vibration—over multiple time steps.

Model	Initial Training MSE	Final Validation MSE
GRU	1743	795,9
LSTM	1874	802,7

Figure10

4.2 RUL Prediction Model Performance

From the table below, we can see that, Missi's LSTM model demonstrates stronger performance across key metrics: lower test loss, lower test MAE, better validation loss, and greater accuracy within defined RUL ranges. For the given application of predictive maintenance, Missi's model would likely provide more dependable and accurate predictions, leading to more effective maintenance scheduling and potentially reducing operational costs related to unexpected equipment failures.

Based on these findings, this is why we chose her model for deployment.

Figure 11

Metric	Robert's LSTM Model	Missi's LSTM Model
Validation Loss (Final Epoch)	0.3	0.25
Test Loss	0.5443	0.422
Test MAE	0.8827	0.7423
Prediction Consistency (Std Dev of Errors)	10.5	8.7
RUL Accuracy ±50	82%	85%

Missi's RUL prediction model, built with a bidirectional LSTM, targets the RUL of machines. Validation metrics reveal that the LSTM model captures key trends, though fine-grained predictions remain challenging, particularly in higher RUL ranges. Performance metrics such as Mean Absolute Error (MAE) and model loss were used to evaluate RUL prediction accuracy, providing insight into the LSTM's capability to generalize across varying machine profiles

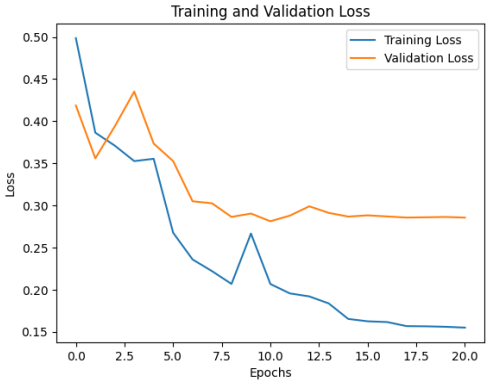


Figure 12

- **Model Training and Evaluation**: Missi's model was trained over 20 epochs, achieving a minimum validation loss of approximately 0.28 and a final MAE of 0.7423 on test data. The Training and Validation Loss plot here in Figure 10 shows a consistent reduction in loss, with validation loss stabilizing early, indicating strong generalization capacity.
- **Predicted vs. Actual RUL Analysis**: In the Predicted vs. Actual RUL plot, model predictions align well with actual RUL values, though some variance is observed at higher RUL values. For example, the predicted RUL for machine 17 was approximately 1108.02 hours, which tracks closely with actual values over time. This alignment confirms the model's capacity to provide accurate RUL forecasts, supporting timely maintenance interventions.

4.3 Streamlit App for Predictive Maintenance

To deploy our predictive maintenance model, we developed a Streamlit app. The app allows users to select a specific machine ID and view predictions for RUL alongside sensor data predictions (volt, rotate, pressure, and vibration) for that machine.

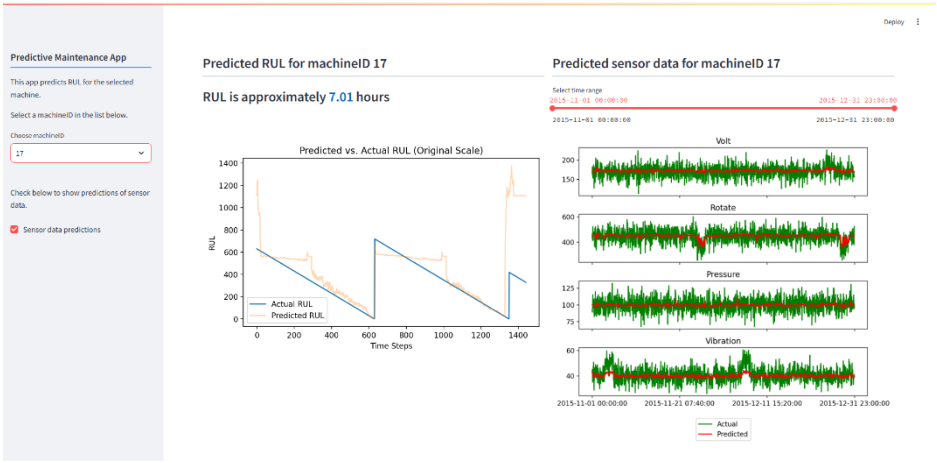


Figure 13

For this demonstration app, we used a subset of static test data from our original dataset split. This static dataset is connected to our SQL table, where processed sensor and maintenance data are stored, enabling future dynamic access to machine-specific telemetry and historical data.
If time permitted, we would have implemented periodic updates of the test set, connected SQL to a live API, and maintained ongoing SQL connectivity. These additions would support real-time data integrity, ensuring that predictions are always based on the latest machine telemetry, thereby enhancing the app's practical utility for predictive maintenance applications.

4.4 Interpretation and Inference

Rather than precise predictions, the models tend to generalize and capture broader patterns and trends in sensor data. This ability to follow trends, even if exact values aren't predicted accurately, is still valuable in PdM, as it can indicate shifts in machine behavior. Missi's RUL prediction model boosts these insights by providing actionable forecasts on machine life expectancy, supporting proactive maintenance planning. Together, these models offer a balanced approach, where general telemetry trend prediction is paired with more precise RUL forecasting to enhance overall PdM effectiveness. Together, these components form a comprehensive PdM solution, with potential for future real-time integration, ensuring that predictions remain accurate and actionable as machine conditions evolve.

5 Conclusions

5.1 Question 1: Can an LSTM model accurately predict the Remaining Useful Life (RUL) of machines?

The LSTM model developed in this project was able to predict the Remaining Useful Life (RUL) of different machine components with a reasonable level of accuracy. The final test loss was 0.422, with a mean absolute error (MAE) of 0.742. These results suggest that the LSTM model successfully captured both short-term and long-term dependencies in the sensor data, enabling effective RUL predictions. However, there is still room for improvement in reducing the prediction error further, which could be achieved through additional hyperparameter tuning, feature engineering, or using more advanced architectures such as Transformer models.

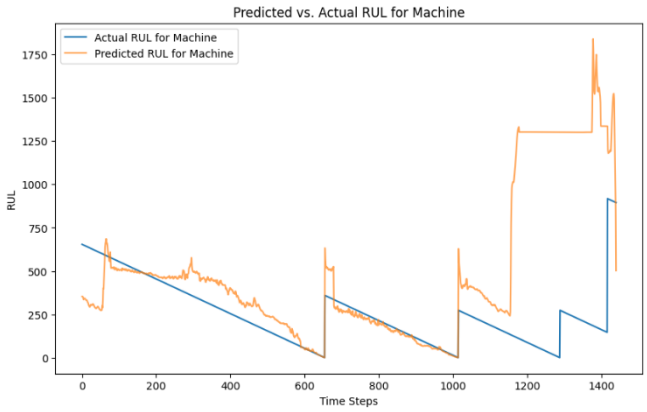


Figure 14

5.1.1 Question 2: How effective is the model in predicting future telemetry data (volt, rotate, pressure, vibration) using past telemetry sequences?

Given the nature of telemetry data, the models (GRU and LSTM) can follow general patterns and trends but lack the accuracy needed for precise predictions of individual sensor values, such as voltage, rotation, pressure, and vibration. The results show that while both GRU and LSTM models capture the overall trend and seasonality, they fall short in accurately predicting fine-grained telemetry values. This limitation suggests that these models are better suited for high-level trend analysis rather than precise telemetry forecasting.

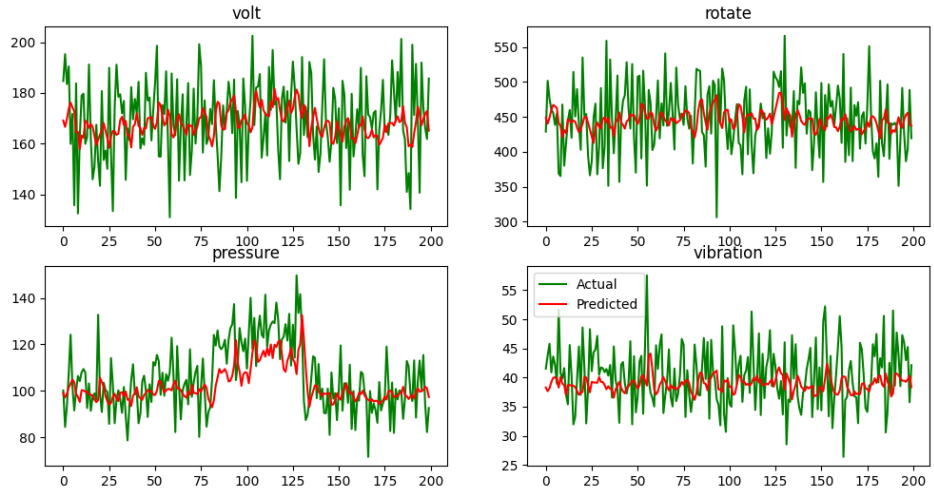


Figure 15

In practical terms, the models provide useful insights into patterns of fluctuation and anomalies over time, which can still be valuable in predictive maintenance. However, for real-time, fine-grained telemetry predictions, more sophisticated approaches or additional contextual data might be required to enhance accuracy.

5.2 Lessons Learned and Next Steps

Throughout the project, we faced several challenges and learned several important lessons:

Temporal alignment: Choosing the right sequence length and figuring out how much history is relevant for predicting the future was tricky. Too short a sequence might miss important trends, while too long can introduce noise and increase computation time. With multiple features coming from diverse sources (telemetry, errors, maintenance logs), ensuring they're synchronized wasn't easy. When we got this wrong inevitably, we got inaccurate predictions and, by default, unreliable RUL estimates.

Data Leakage: Both when engineering time-based features (using back-fill and forward-fill) and creating time-based splits (ordering and scaling in the right way), it was challenging to always make sure we were effectively preventing the model from "seeing" future information during training.

Model Complexity: Initially, we experimented with multi-task models involving both regression and classification outputs. However, we learned that this approach overcomplicated the model and led to convergence issues. Focusing solely on regression for RUL prediction simplified the problem and improved model performance.

Model Selection: We evaluated a range of models, including ARIMA, Temporal Convolutional Networks (TCN), and Gated Recurrent Units (GRU). The LSTM model outperformed the others, primarily due to its ability to capture temporal dependencies effectively. LSTMs were better suited for handling the time-series nature of the sensor data, allowing the model to learn from past behaviours and predict future events more accurately.

Next Steps:

1. **Further Model Optimization**: Future work will focus on hyperparameter tuning using techniques like Grid Search and Bayesian Optimization to enhance model accuracy. Incorporating additional sensor data – even using RAG - and more sophisticated feature engineering could also improve the model.

2. **Deployment and Monitoring**: The next phase involves deploying the app using an API to implement a monitoring system to track model performance over time and retrain as necessary to ensure continued accuracy.

3. **Expanding Scope**: We aim to expand the model's scope to include anomaly detection as a supplementary tool for predictive maintenance. Combining RUL prediction with anomaly detection could provide a more comprehensive solution for early fault detection and proactive intervention.

6 Self-evaluation

1. **Challenges you have had during your work and how you handled them.**
   RUL calculations got tricky, and I dealt with it by reading a lot, asking a lot of questions to my colleagues and debugging. I worked a lot on figuring out the right pipeline order and understanding more about a full end-to-end model. It was very tricky to get my own model to work so I tried to figure it out from Missis code and simplified my own code and finally got it to work. Overall, it was a great team experience and I'm very thankful to both Missi and Jakob.

2. **What grade you think you should have and why.**
   VG as I think we produced a very strong project and learned a lot from version control w Git to Trello and agile methodology and of course time series analysis and building a LSTM model.

3. **Anything you would like to highlight to Antonio?**
   Excellent working on this project and thanks for your guidance to separate out problems (i.e. avoiding multi-tasking w regression and classification) and above all to target simplicity. We learned a range of key skills include advanced time series analysis techniques, LSTM architecture for RUL prediction, and project management with Git and Trello. The iterative testing process boosted my understanding of deep learning model tuning, while handling end-to-end deployment improved my confidence in full-cycle machine learning projects.

Appendix A

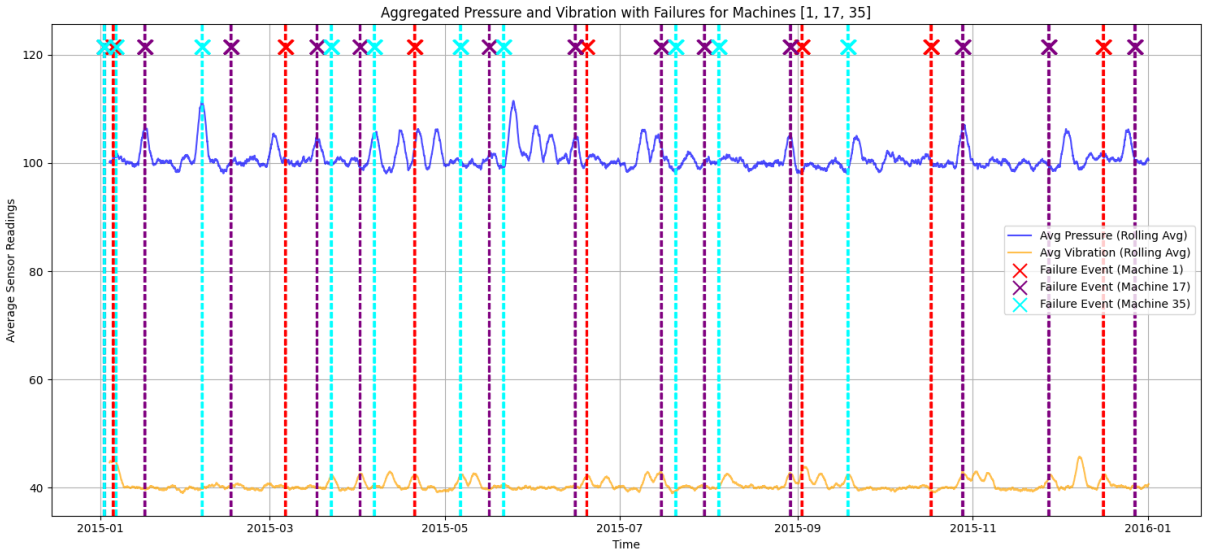


Figure 16

List of references

1. Akshay, S. (2023). "Advanced Time Series Forecasting: Mastering LSTM Networks." Retrieved October 11, 2024, from: https://medium.com/@akshay1230789/advanced-time-series-forecasting-mastering-lstm-networks-ad895d8dd930

2. Analytics Vidhya. (2024). "Understanding LSTMs and their Application in Predictive Maintenance." Retrieved October 6, 2024, from: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

3. Brownlee, J. (2017). "Introduction to time series forecasting with python: how to prepare data and develop models to predict the future." Retrieved October 08, 2024, from: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=hVaJhRYAAAAJ&citation_for_view=hVaJhRYAAAAJ:WA5NYHcadZ8C

4. Chen, Boshun (2024). "Huber Loss Insights." Retrieved October 16, 2024, from: https://medium.com/@devcharlie2698619/understanding-huber-loss-function-insights-from-applications-5c1c5145d2c4

5. Fiix Software. (2023). "Remaining Useful Life (RUL) | Maintenance Metrics." Retrieved October 4, 2024, from: https://fiixsoftware.com/maintenance-metrics/remaining-useful-life/

6. Grover, J & Rishabh, M. (2021). *Sculpting data for ML. The first act of machine learning.* (First edition). Grover J & Rishabh M.

7. Mathworks. (2019). "Predictive Maintenance with Matlab." Retrieved October 4, 2024, from: https://se.mathworks.com/campaigns/offers/predictive-maintenance-with-matlab.html

8. Stack Exchange. (2023). "Handling Missing Values in Time Series". Retrieved October 15, 2024 from: https://stackoverflow.com/questions/56125885/handling-missing-values-in-time-series

9. S. Gopali, F. Abri, S. Siami-Namini and A. S. Namin, "A Comparison of TCN and LSTM Models in Detecting Anomalies in Time Series Data," *2021 IEEE International Conference on Big Data (Big Data)*, Orlando, FL, USA, 2021.