Master Thesis
Machine Learning and Artificial Intelligence Enabled Failure Prediction for Maritime Propulsion Systems

Jacob Vitfell Køpke & Emil Lytje-Dorfman

AALBORG UNIVERSITY

Abstract

This thesis investigates the application of machine learning (ML) and artificial intelligence (AI) techniques for failure prediction in maritime propulsion systems, with a focus on two critical failures observed in MAN Energy Solutions' engines: accumulator membrane failure in sequential two-stroke engines and pipe damage during methanol fuel changeover in dual-fuel systems. The thesis explores the feasibility of predictive maintenance through data-driven approaches, leveraging time series data collected from operational vessels and controlled test engines. A comprehensive feature engineering pipeline is developed to address challenges related to high dimensionality, feature redundancy, and class imbalance. Feature selection methods are applied to construct interpretable and task-relevant feature subsets, resulting in insights into system behavior leading up to failure. Multiple ML and AI models, including Random Forests, 1D Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks, are implemented and evaluated for both binary failure classification and remaining useful life (RUL) regression tasks, for different reading window (RW) and prediction window (PW) sizes. Here SMOTE and DTW-SMOTE has been used to augment data, increasing the data variance in training. Results demonstrate that methanol changeover failures are highly predictable, with Random Forest and 1D CNN models achieving strong performance with an accuracy of 91% for changeover classification and an R^2 of 0.826 for changeover RUL prediction. In contrast, accumulator failure prediction proves more challenging due to limited failure data and lower signal resolution with the 1D CNN providing the greatest accuracy of 71%. The findings highlight the importance of model selection, task-specific window sizing, and feature interpretability in predictive maintenance applications. The thesis concludes that ML and AI can be used for predicting pipe damage during methanol fuel changeover, while providing valuable insight into the system and the failure through feature engineering. This task is slightly harder for accumulator membrane failure in sequential two-stroke engines, and further work is required to develop high performing and robust models for this purpose.

Titlepage

Title:
Machine Learning and Artificial Intelligence Enabled Failure Prediction for Maritime Propulsion Systems

Theme:
Predictive maintenance and machine failure prediction

Project Period:
February 1, 2025 - June 4, 2025

Project Group:
1028

Page Numbers:
118

Participants:
Jacob Vitfell Køpke
Emil Lytje-Dorfman

Supervisors:
Ramoni Ojekunle Adeogun
Carsten Hounsgaard

Date of Completion:
June 3, 2025

AALBORG UNIVERSITY

STUDENT REPORT

Electronics Systems
Institute of Electronic Systems
http://www.aau.dk

Acknowledgments

This Master's thesis was carried out in collaboration with MAN Energy Solutions. We would like to thank MAN ES for guiding the thesis in a direction that is both relevant to our academic goals and valuable to the company.

We are especially grateful to **Carsten Hounsgaard** for proposing the project and for providing guidance on data selection and methodological direction.

Special thanks go to **Christian Skaaning**, **Mikkel Sæbø Hansen**, **Usman Iqbal**, and **Morten Kjul** for generously sharing their expertise and insights into the systems, which have been essential to the development of this thesis.

We would also like to thank **Alexander Reffs** and **Tobias Bruun Brøgger** for coordinating resources and investing their time in the planning and collection of the data used in this work.

Our sincere gratitude goes to **Stefan Meyer** for establishing the right framework and welcoming us into his department, creating a supportive and enriching environment for our research.

Lastly, we would like to extend our heartfelt thanks to our supervisor, **Ramoni Ojekunle Adeogun**, from Aalborg University, for his invaluable feedback, constructive criticism, and consistent support throughout the project.

Contents

Preface

Aalborg University, June 3, 2025

[illegible]

Emil Lytje-Dorfman
elytje20@student.aau.dk

[illegible]

Jacob Vitfell Køpke
jkapke20@student.aau.dk

1 Introduction

Shipping is the main way goods are being moved around the globe, accounting for 80% of the worlds exports and imports[1]. Greener shipping is therefore a focus of several regulatory agencies, in an effort to tackle climate change and environmental harm, which has lead to an increase in regulations, requirements, and taxation on emissions in the shipping sector[2][3]. This has naturally caused an increase in demand for greener maritime propulsion systems, with reduced emissions. This demand has been identified by MAN Energy Solutions, which has several new propulsion designs aimed at tackling this demand. MAN ES is a company which designs maritime propulsion systems, with a large focus on two-stroke diesel engines which power most large container ships[4]. The effort to reduce emissions has been two fold at MAN ES. Firstly, there is a focus on reducing emissions of diesel combustion. This has resulted in the design of a sequential two-stroke engine. This design uses pre-combustion, through precise injection timing, to reduce emission of harmful NO_x gasses[5]. Secondly, new systems are developed which allow the engine to run on a secondary fuel type, such as methanol, which burns cleaner than diesel[6]. Cleaner combustion is greatly beneficial in emission control areas (ECA) which have stricter emission requirements[7]. Both new designs have experienced problems in service, which has been costly for MAN ES due to warranty claims and time and resources being required for correcting the problems. The hydraulic injection systems for the two unique and separate engine designs are described in section 1.1, and their problems in section 1.2.

1.1 System description

While several sequential two-stroke engine designs exist, the focus in this case in on the engine in service on the Porsorja Express. Similarly several engine models can run on methanol as secondary fuel. The focus here is on the larger LGIM G95 engines. These engines are described in subsection 1.1.1 and 1.1.2 respectively.

1.1.1 Two-stroke sequential engine on the Porsorja Express

As mentioned, regulations on emissions has been focused on NO_x gas emissions, as these particles are especially harmful[5]. This has naturally increased the demand for maritime propulsion systems with reduced NO_x emissions. MAN ES's sequential two-stroke engine is able to conduct a pre-combustion which reduces main combustion temperature, resulting in reduced NO_x emissions. The injection system on the Porsorja Express is described in the block diagram in figure 1.1

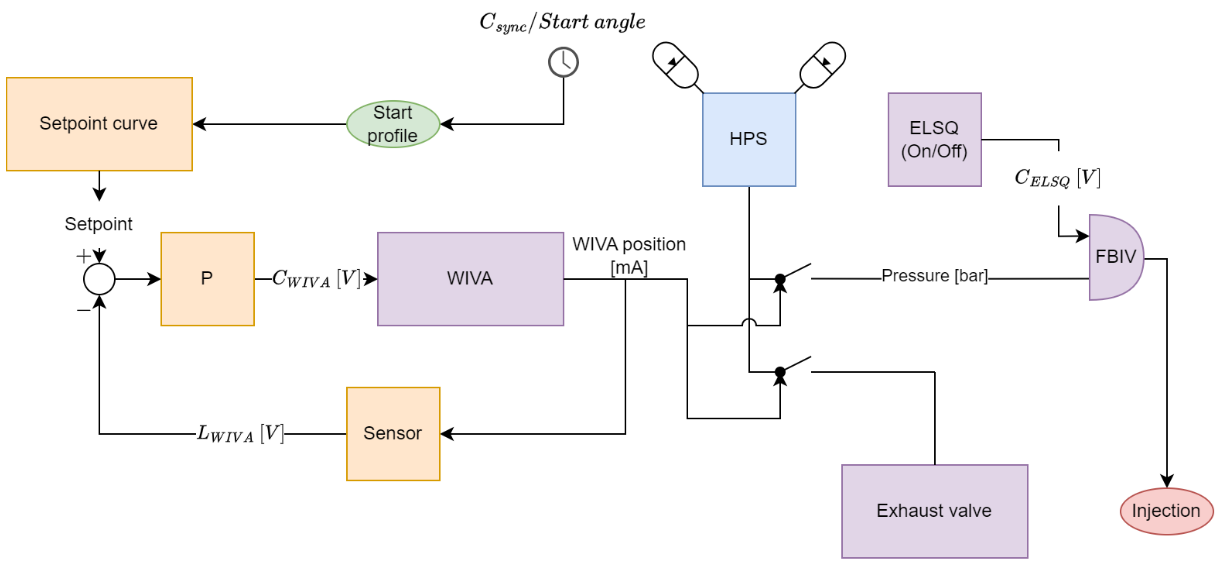


Figure 1.1: Block diagram of fuel injection system.

The injection system is responsible for injecting diesel into the cylinders at the right time and at high pressure. This is controlled through the injection system, which uses hydraulic and electrical control. The high pressure system (HPS) supplies hydraulic fluid to the system, which the WIVA controls the flow of in order to control injection and exhaust. What makes the sequential two-stroke engine unique is its use of sequential combustion (SC), where three fuel inject sequentially. To control SC precisely while maintaining high pressure, electronic ON/OFF valves (ELSQ) are used. In contrast to traditional spring designs, the use of ELSQ valves results in much more controllable but also sudden injections. An injection cycle is started when diesel is filled into the FBIV. Once diesel has been filled, the WIVA routes the HPS to the FBIV, pressurizing the diesel. Once the diesel is pressurized an injection is order by turning on the ELSQ valve. The ELSQ valve is what allows for precise and sudden injection timing. A flow chart of an injection cycle can be seen in figure 1.2.

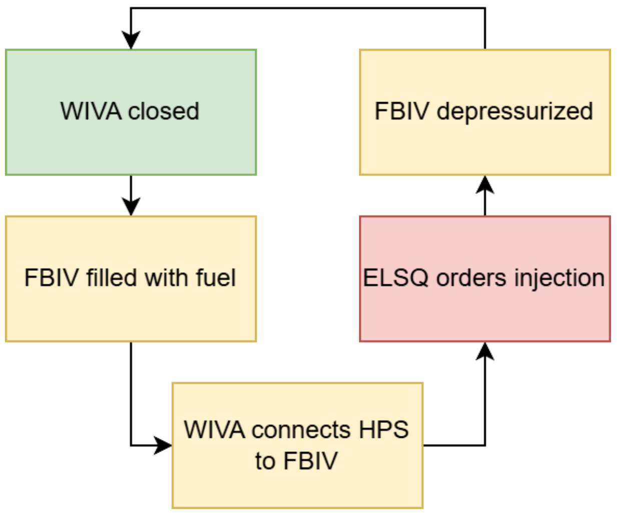


Figure 1.2: Flow chart of an injection cycle for the two-stroke sequential engine on the Porsorja Express.

1.1.2 Methanol changeover procedure

As mentioned, methanol is used for combustion due to its cleaner combustion, which is beneficial in some ECAs. However, all fuel systems have to be able to run on diesel as a safety, requiring systems that can run on both. These engine systems make use of a first fuel (FF) system (diesel), and a second fuel (SF) system (methanol). Each fuel system has its own injection system, that are similar but with slightly different components. Both systems are controlled by an ELFI valve which controls the system, through a control feedback loop which can be seen in figure 1.3.

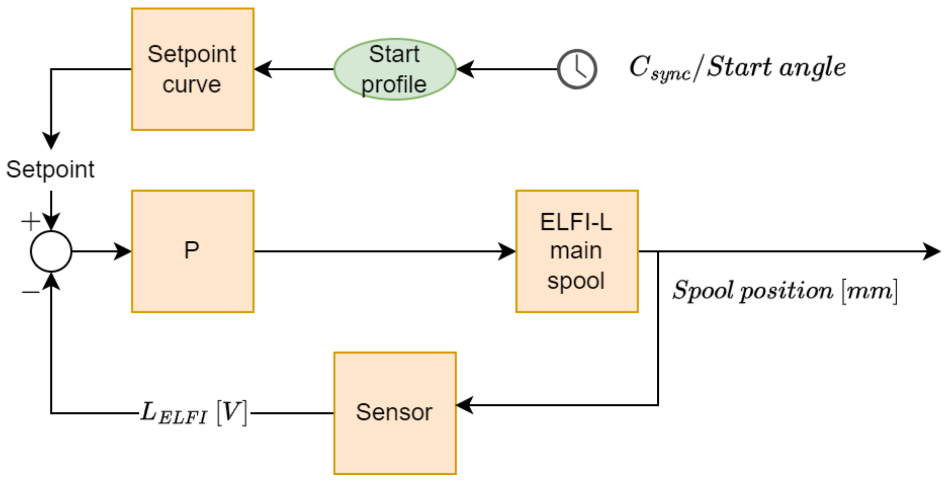


Figure 1.3: Block diagram of ELFI control system.

The ELFI valve controls hydraulic fluid flow in the system, thereby controlling injection.

The full SF injection system can be seen in figure 1.4 with ELFI-L, FBIV-M and ELBI.

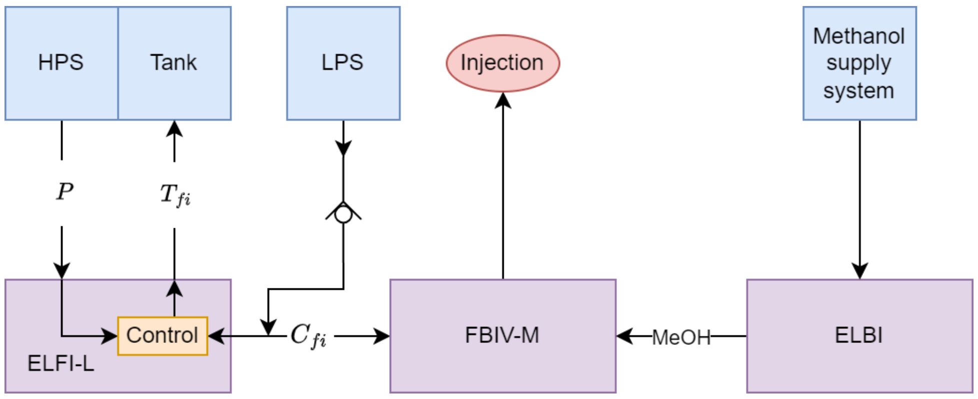


Figure 1.4: System diagram of second fuel methanol injection system.

A methanol injection cycle goes through five stages. In the first stage the ELFI-L is closed and the ELBI fills methanol into the FBIV-M. After methanol has been filled, the ELFI-L connects the HPS to the FBIV-M, pressurizing the methanol. Once the opening pressure of the fuel valve is reached, the methanol is injected. After injection, the ELFI-L routes hydraulic fluid to the tank, depressurizing the FBIV-M. Lastly, the ELFI-L is closed again, and the system is ready for the next injection. The injection cycle is summarized in the flowchart in figure 1.5.

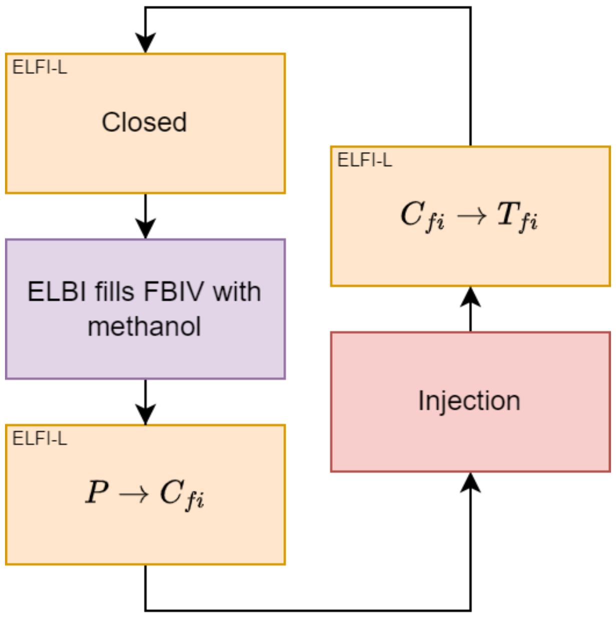
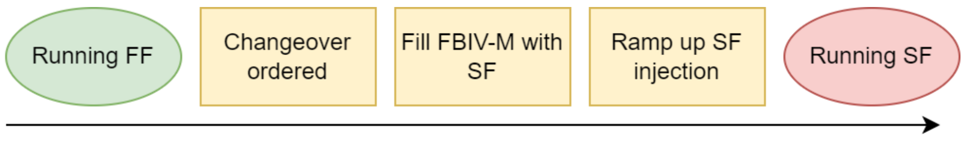


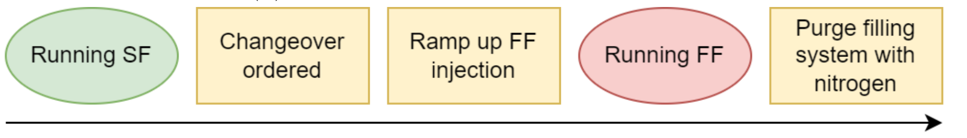
Figure 1.5: Flow chart of a methanol injection cycle. The orange box represents the ELFI-L control system.

The injection system and flow are functionally the same for FF diesel injection, which uses a FIV valve instead of an FBIV-M valve, and an ELFI instead of an ELFI-L.

When the engine wants changeover from one type of fuel to another, it has to undergo a changeover procedure. When changing from SF to FF, the methanol filling system has to be purged, and be emptied of methanol. This is done by filling the system with an inert gas, in this case nitrogen. Once the system has been purged of methanol, diesel injections can start ramping up. Vice versa, when switching from FF to SF, the nitrogen in the system is replaced with methanol. After filling, injections can start ramping up, until the desired level and stability is reached. When methanol is run, diesel is still used as a pilot flame, as methanol cannot combust without a spark or flame. These changeover procedures are summarized by the flow charts in figure. 1.6



(a) FF to SF changeover flow chart.



(b) SF to FF changeover flow chart.

Figure 1.6: Flow charts for changeover procedures.

1.2 Problem description

Both described engine designs have experienced system failures once entering service testing. These failures have not been found in the design or simulation processes of the systems, making them unforeseen. The engine on the Porsorja Express experiences accumulator membrane failure, and the LGIM G95 experiences issues with pipe knocking and damage when changing over to SF. These problems are described in subsections 1.2.1 and 1.2.2 respectively.

1.2.1 Accumulator failure on the Porsorja Express

Accumulators are a component commonly found on hydraulic injection systems where they are used for their capacitive and dampening effect. In times of peak demand for hydraulic fluid, the accumulators ensure this demand can be met by supplying additional fluid. They can be seen attached to the HPS in figure 1.1. As mentioned in section 1.1.1 the injections on this engine are very sudden. This is suspected to cause a water hammering effect, which leads to high demand from the accumulators to dampen it. On the Porsorja express the accumulators have a tendency to burst the membrane and fail, at a rate and at times not seen on other sequential engine designs. While the engine can run without accumulators, it does hurt injection, and thereby engine performance. Furthermore, these failures lead to warranty claims from the ship operator. Avoiding or reducing the frequency of accumulator failures is beneficial for both MAN ES and as well as the ship operator. Steps towards reducing the failure frequency has already been taking by revising the way the accumulators are mounted. While this has reduced the failure frequency, it has not resolved the problem entirely. So far analysis of the problem has not reached tangible reasons or solutions for the accumulator failures.

1.2.2 Changeover to methanol failure

When the engine described in section 1.1.2 changes over to SF significant knocking has been noticed in the piping on the system. In certain cases this has resulted in ripping and significant damage to the pipes. Broken pipes means the engine cannot run as intended, and is a severe issue if it happens during operation which is costly. This issue only happens when changing from diesel to methanol and not the other way around. The current hypothesis is that this is due to nitrogen gas been captured in the system when methanol injection starts. This can cause air cushions which cannot be compressed, resulting in the system overcompensating and trying to compress it even further. This hypothesis is currently being tested for some of the engines that currently have SF systems but not on all.

1.3 Predictive maintenance and machine failure prediction

Predictive maintenance is a method of maintenance, in which some method or model predicts when maintenance is necessary in order to prevent a failure[8]. In general three main maintenance categorization methods exist; physical model-based, knowledge-based, and data-driven. The physical model-based method uses mathematical system models in order to determine when failure happens. The knowledge-based methods uses expert knowledge in order to reduce the complexity of the physical model. Lastly, data-driven approaches to predictive maintenance heavily use machine failure prediction (MFP), in order to predict when maintenance is necessary. MFP is heavily enabled by machine learning (ML) and artificial intelligence (AI), in order to predict when a "machine" will fail through either anomaly detection or time-based models[8][9][10].

Applying predictive maintenance to the problems outlined in section 1.2 can reduce the frequency of system failures by predicting when they will happen. In this way, preventative action can be taken which will reduce the amount of total failures. Reducing the amount of system failures will reduce costs and warranty claims for both the ship operators and MAN ES. Given that the failures are unforeseen, and have not been found in the design or simulation process, the failure prediction tasks lend themselves to a data-driven approach. Using predictive maintenance for these problems will not solve the root cause of the failures but rather reduce the frequency of them. However, analysis of how the systems interact with the prediction models may give insight as to what happens leading up to the failures and what might be the cause of the failures.

Furthermore, in order to develop prediction models, for the outlined failure problems, the nature of the data used should be taken into account. Careful consideration of the data should be taken, in order to appropriately choose appropriate failure prediction methods and model, as this has a great impact on prediction performance[11][8]. Further data consideration are introduced in section 1.4

1.4 Data considerations

The data used to train data-driven prediction models should reflect and be related to the failure either directly or indirectly. Furthermore, depending on the data, problems with dimensionality and redundancy may arise.

When working with high-dimensional data, one must account for the curse of dimensionality. High-dimensional data can often lead to model instability, over-fitting, and poor generalization due to data sparsity. Furthermore, high dimensionality leads to higher computational complexity, and slower training[12]. High dimensionality can arise from redundant features. Feature redundancy can come in two forms, either through features that are not related to the task, or features that have high correlation to other features. Redundant information has no added model or training benefits, while contributing to the curse of dimensionality[12].

Another issue that arises with highly correlated features is multicollinearity, where more than two independent variables are correlated. Multicollinearity causes issues in regression analysis, as it causes difficulty individual features impact on the model. Furthermore, multicollinearity causes overfitting in regression models[13]. Lastly, highly correlated data can degrade interpretability of models. With correlated datasets spurious correlation occur between non-essential features, leading to poorer models in terms of generalization and robustness. This also makes identifying the importance of features, and their impact difficult to interpret correctly[14]. Furthermore, interpretable feature engineering will give an understanding of the system, and how a failure looks.

When considering the data, balancing should be accounted for. While the described system failures have a large impact on the system, the operator, as well as MAN ES, they do not occur at a high frequency, meaning data may be limited. When training on unbalanced data, a bias is learned towards to majority class[15][16][17]. In this case the majority class being non-failure, the trained model will have a difficult time learning failure patterns, if failures are heavily outweighed in the dataset.

Resolving these data considerations through data engineering will create a good foundation for training failure prediction models, as well as provide insight into what the data reveals about the failures.

1.5 Thesis scope and research question

From the previous section the following inquiries are formulate with the aim of guiding the thesis towards the goals and objectives outlined in the scope:

Can ML and AI models be used to predict marine propulsion system failures?

How can feature engineering be used to create a feature space suited for prediction models, while giving insight into the system failures?

The inquiries have lead to the following overarching research question;

"To what extent can ML and AI be used to predict and prevent total system failure in MAN Energy Solution's maritime propulsion systems?"

1.6 Thesis contributions

This thesis aims to contribute to the fields of predictive maintenance and machine failure prediction. Firstly, this thesis will expand on these fields by applying them to hydraulic injection systems within maritime propulsion. Secondly, the use of interpretable feature engineering will demonstrate that feature engineering can be used for extracting relevant information, removing redundancies, while also providing valuable insights into the system failures. Lastly, the thesis will illustrate the effect of different window sizes when applying time sequence models, not just for binary failure classification but also remaining useful life prediction. Ultimately, this thesis will also provide valuable insight into whether or not ML and AI has appropriate uses within maritime propulsion systems, and what these uses necessitate. Additionally, there is a focus on feasibilty of implementation without extra cost in these systems.

1.7 Thesis outline

In order to resolve the aims and objectives the following thesis outline is proposed.

- A literature review is conducted in chapter 2 on predictive maintenance and machine failure prediction, feature engineering, and dataset balancing. The review aims to find gaps in existing literature, and will help make informed decisions on which methods to use in the remaining chapters.

- Based on existing literature, appropriate analysis methods, ML and AI models, and techniques are described in depth in chapter 3. Underlying the theoretical background for the remainder of the thesis.
- The described data analysis methods are then be used in chapter 5 for finding relevant features, and feature engineering. The aim is to create the best possible data subsets for the prediction models.
- Having found relevant features for the two problems, prediction models and methods will be chosen and implemented in chapter 4 along with a description of the datasets and the data pipeline, including data augmentation and balancing.
- Model specifics and their results are presented in chapter 6, where they are evaluated through appropriate evaluation metrics.
- Finally the results and findings are discussed and concluded upon and chapter 7, where the research question will also be answered.

2 Literature review

In order to answer the inquiries outlined in chapter 1, existing literature on the topics of predictive maintenance, feature engineering, and dataset balancing is reviewed in this chapter. This will provide an overview over existing methods to make informed choices later in the thesis, as well as highlight the main contributions of this to the research area of predictive maintenance and machine failure prediction.

2.1 Predictive maintenance and machine failure prediction

As mentioned in the introduction, predictive maintenance is heavily enabled by machine failure prediction. Existing methods use mathematical and statistical modeling, in order to describe system behavior, and use data to learn when failures are probable[18][19]. In [19] system modeling is used used to develop a failure predictor. The predictor is then enhanced using Gaussian process regression, which is learned through Bayesian inference using a limited number of real data samples. In [18] the authors improve on Hidden Markov Model (HMM) methods by adapting Bayesian networks to work in an online setting, thereby improving temporal failure prediction.

Other literature on prediction methods leverage machine learning and deep learning techniques on industrial data with a large number of sensors enabled by IoT devices. A review shows that the most common prognostic methods are prognostics and health management (PHM), condition-based maintenance (CBM), and remaining useful life (RUL)[10]. The review in [10] also highlights that Random Forest (RF) and deep learning (DL) are the most commonly used prediction techniques for predictive maintenance. [8] and [9] compare different ML and DL techniques for failure prediction on industrial datasets.

In [9] logistic regression (LR), K-nearest neighbor (K-nn), Support Vector Machine (SVM), decision trees, bagging classifier, boosting methods, RF, and LSTMs are compared. Here data pre-processing and dataset balancing using SMOTE is used to improve the models. The paper clearly shows that deep learning can outperform classical ML models, and that ensemble methods generally perform well for failure prediction. Additionally, it is concluded that the use of SMOTE for dataset balancing has a positive impact on model performance. The study is limited to binary failure classification, and does not expand into methods such as RUL. Additionally, there is a distinct lack of dimensionality reduction or feature selection.

[8] focuses on comparing similar methods to [9], including transformer, with a focus on multivariate time series data. From [8] it is clear that the size of the RW and PW are important for model performance, in case of time variate data. [8] also shows that in cases of time variate data, deep learning methods which capture temporal information, such as Convolutional Neural Networks (CNN), LSTMs, and Transformers, perform well. This provides valuable insights into failure prediction tasks which deal with multivariate time series data, demonstrating the importance of finding appropriate window sizes, and choosing models appropriate for the task. However, this is again limited to binary failure classification, and relies heavily on expert knowledge for feature selection.

The existing literature deals with industrial data such as for production, storage and robotics. None of the existing literature deals with systems within maritime propulsion systems, which also can benefit from predictive maintenance and machine failure prediction. Additionally, while the existing literature deals with high dimensional data through either system and expert knowledge based feature selection, few use data driven feature selection methods. While literature which does deal with large scale IoT data, the methodologies used are novel and very specified for "big data"[20]. The existing literature lacks failure prediction methods which leverage data-driven feature selection methods allowing for high interpretability of models and understanding of failure causes.

2.2 Feature engineering

In feature engineering the feature space is manipulated to extract information best suited for training models, avoiding spurious correlations that are harmful for model training[14]. Some feature engineering methods such as PCA and ICA extract features and reduce dimensionality, while ensuring statistics are as preserved as possible. Manifold learning achieves something similar but is more suited for non-linear data[21][22][23].

While PCA, ICA, and manifold learning are effective for reducing dimensionality, they lack interpretability, which is desirable in some cases for model understanding. Here feature selection methods play a large role. Feature selection methods are grouped into filters and wrappers. Filters are pre-computable, whereas wrappers are built into the modelsprocessing[24][25]. Filter methods include correlation-based feature selection (CFS), where feature subsets are ranked according to a heuristic metric using MDL, Relief, and symmetrical uncertainty for estimating correlation[24]. The method has later been adapted for multi-variate time sequence data using adjusted mutual information[25]. Wrapper methods such as greedy elimination and recursive feature elimination, as well as embedded methods such as Lasso regression use algorithmic approaches to reduce the feature space by eliminating correlations [26][27][28]. More advanced methods for dealing with spurious correlations includes *causal intervention*, *invariant learning*, *feature disentangle-*

ment, and *contrastive learning*. These methods are summarized in *"Spurious Correlations in Machine Learning: A Survey"* [14]. General for these methods is that they are more complex, and often parametric or learnable[14].

2.3 Dataset balancing

Unbalanced datasets are a common problem in ML and AI. Extensive research shows that in order to train the ML and AI methods properly, with the best results, the dataset has to be balanced. This balancing can be done through either under-sampling the majority class, or over-sampling the minority class. When under-sampling the majority class, data from the majority class is removed[16][17].

Under-sampling the majority class involves the removal of samples. The removal of data can be done in numerous ways. It can be done by finding noisy pairs through TOMEK links, and removing the majority class data point. Alternatively, a nearest neighbor method can be used to find a data subset, in which majority class samples that are far from the decision boundary are removed. The two can also be combined into One Sided Selection (OSS)[15]. Nearest neighbor methods can also be used to remove majority class samples with high risk of miss-classification[16].

For oversampling Synthetic Minority Oversampling Technique (SMOTE) is the most common. SMOTE over-samples the minority class, by creating synthetic samples through interpolating minority class samples. Many different versions of SMOTE exist with differing focuses. In general SMOTE works through k-Nearest Neighbor algorithms[15][16][29]. Other methods of generating synthetic data also exist specifically for time sequence data. The open source framework TSGM combines several methods for time sequence generation, as well as augmentation, using both data-driven and simulation based methods[30].

3 | Theory and background

In this chapter the theory and background on different methods, models, and techniques are discussed in depth. This is based on the literature review conducted in chapter 2, and general knowledge regarding ML, AI, signal processing, and data analysis.

Dynamic time warping, correlation, and auto correlation analysis are discussed due to their uses for feature analysis and general data understanding. These techniques can be used to gain an understanding of the dataset, the relation to system failures, and relationships within the data itself. These methods can be combined with feature selection methods in order to engineer the feature space in a interpretable way. This will ultimately help answer the secondary inquiry *"how can feature engineering be used to create a feature space suited for prediction models, while giving insight into the system failures?"*.

Selected machine learning algorithms, as well as AI models that deal with sequential data are also described, as based on the literature review. These are will help answer the primary inquiry *"can ML and AI models be used to predict marine propulsion system failures?"* by providing the necessary background knowledge to make informed decision on the most appropriate algorithms and models to use for failure prediction.

3.1 Dynamic time warping

Dynamic time warping (DTW) is a popular and powerful time series analysis tool. In its essence, DTW finds the similarity between time sequences by finding the distance between the best alignments of them. Consider two time-dependent sequences S_1 and S_2, with n and m samples respectively. DTW aligns S_1 and S_2 such that they are as close as possible by stretching or compressing them. The alignment is evaluated by a cost matrix $C \in \mathbb{R}^{n \times m}$ with local distance scores $C(i,j) = c(s_{1_i}, s_{2_j})$ for $i \in \{0, 1, \ldots, n\}$ and $j \in \{0, 1, \ldots, m\}$, where close alignments have a low score and far alignments have a high score. The cost is computed for all paths $\pi \in \Pi$ that traverse over i and j. DTW is then the total cost of the path with the minimum cost as seen in equation 3.1[31][32].

$$DTW(S_1, S_2) = \min_{\pi \in \Pi} \sum_{i,j \in \pi} C(i,j) \tag{3.1}$$

A popular cost measure is using the two-norm for Euclidean distance $c(s_1, s_2) = ||s_1 - s_2||$.

DTW has several use cases within machine learning. This includes feature analysis by finding feature similarities in cases where features may be time shifted compared to each other or other cases where correlation cannot capture the similarity in an appropriate way[31]. Furthermore, DTW also has uses within clustering, such as using it in nearest neighbor algorithms[33][34].

3.2 Correlation

The Pearson correlation coefficient is a normalized covariance, which measures the statistical relationship between two random variables X and Y. The coefficient is found from equation 3.2[35].

$$\rho_{XY} = \frac{Cov(X,Y)}{Var(X) \cdot Var(Y)} \tag{3.2}$$

The correlation coefficient can also be computed for a signal with time-lagged versions of itself in order to find the auto-correlation coefficients(ACF). The ACF measures the time dependency of a signal with itself[36]. It is described by equation 3.2, with an algorithmic implementation seen in algorithm 1

$$\begin{aligned}\rho_{X_t X_{t-\tau}} &= \frac{Cov(X_t X_{t+\tau})}{\sigma_{X_t}\sigma_{t+\tau}} = \frac{Cov(X_t X_{t+\tau})}{\sigma_X^2} \\ Cov(X_t X_{t-\tau}) &= \mathbb{E}[(X_t - \mathbb{E}[X_t]) \cdot (X_{t-\tau} - \mathbb{E}[X_{t-\tau}])] \\ &= \mathbb{E}[(X_t - \mu_X) \cdot (X_{t+\tau} - \mu_X)] \\ &= \frac{1}{N}\sum (X_t - \mu_X) \cdot (X_{t+\tau} - \mu_X)\end{aligned} \tag{3.3}$$

Algorithm 1 Algorithm for computing the auto-correlation of signal X.

```
1: $N \leftarrow$ number of samples in $X$
2: $\mu_X \leftarrow$ sample mean of $X$
3: $\sigma_X^2 \leftarrow$ sample variance of $X$
4: for lag $\tau$ do
5:     $Cov(X_t, X_{t+\tau}) = \frac{1}{N}\sum_{i=0}^{N-\tau}(X[i] - \mu_X) \cdot (X[i+\tau] - \mu_X)$
6:     $acf(\tau) \leftarrow \frac{Cov(X_t, X_{t+\tau})}{\sigma_X^2}$
7: end for
```

3.3 Feature selection methods

For the problems described in chapter 1 it is not only important to predict the failures but also understand how and why they occur. Therefore, model interpretability is important. That is why feature selection methods are used rather than feature reduction or extraction methods. The described methods include correlation-based feature selection (CFS), greedy elimination, recursive feature elimination, and Lasso regression.

3.3.1 Correlation based feature selection

As mentioned, CFS ranks feature subsets based on a heuristic metric. The metric is based on the heuristic *"a good feature subset is one that contains features highly correlated to the class, yet uncorrelated with each other"*, which is described by equation [24].

$$M_s = \frac{k\overline{r_{cf}}}{k + k(k-1)\overline{r_{ff}}} \tag{3.4}$$

S represents the subset containing k features, $\overline{r_{cf}}$ is the mean class-feature correlation where $f \in S$, and $\overline{r_{ff}}$ is the mean feature-feature correlation. Practically, the metric has to be computed for all subsets to find the best one.

3.3.2 Greedy elimination

Greedy elimination is a wrapper, and is integrated into model training. It continuously considers smaller feature subsets, measuring the impact of removing a feature through some score like F1-score or mean square error as described in algorithm 2[27].

Algorithm 2 Greedy Feature Elimination

```
1: Input: Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, model $M$, feature set $F = \{f_1, \dots, f_n\}$, evaluation metric $\mathcal{E}$
2: Output: Reduced feature set $F^*$
3: Initialize $F^* \leftarrow F$
4: Train $M(\mathcal{D}_{\text{train}})$ with $F^*$
5: Compute performance $P_{\text{best}} \leftarrow \mathcal{E}(M, \mathcal{D}_{\text{val}})$
6: repeat
7:     $\Delta P \leftarrow 0$
8:     for each feature $f \in F^*$ do
9:         $F_{\text{temp}} \leftarrow F^* \setminus \{f\}$
10:        Retrain $M(\mathcal{D}_{\text{train}})$ with $F_{\text{temp}}$
11:        $P \leftarrow \mathcal{E}(M, \mathcal{D}_{\text{val}})$
12:        if $P \geq P_{\text{best}}$ then
13:            $F^* \leftarrow F_{\text{temp}}$
14:            $P_{\text{best}} \leftarrow P$
15:            $\Delta P \leftarrow 1$
16:            break
17:        end if
18:    end for
19: until $\Delta P = 0$
20: return $F^*$
```

3.3.3 Recursive feature elimination

Recursive feature elimination is a wrapper, and is integrated into model training. It recursively removes the least important feature based on internal model weights or importance scores, reducing the feature set until the desired number of features is reached. This functionality is described in in algorithm 3.

Algorithm 3 Recursive Feature Elimination

```
1: Input: Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, model $M$, feature set $F = \{f_1, \ldots, f_n\}$, importance metric $\mathcal{I}$, number of features to select $k$
2: Output: Selected feature subset $F^*$ of size $k$
3: Initialize $F^* \leftarrow F$
4: while $|F^*| > k$ do
5:     Train $M(\mathcal{D}_{\text{train}})$ with features $F^*$
6:     $I = \mathcal{I}(M, F^*)$
7:     $f_{\min} \leftarrow \arg\min_{f \in F^*} I(f)$
8:     Remove $f_{\min}$ from $F^*$
9: end while
10: return $F^*$
```

Importance can be measured in different ways, including weights in regression models, impurity in tree models, or drop in model performance such as F1-score[37].

3.3.4 Lasso regression

Lasso regression is a linear model that performs both feature selection and regularization by adding an L1 penalty $\lambda|\beta_j|$ to the loss function, the strength of which is controlled through λ. This penalty encourages sparsity in model coefficients, effectively driving some coefficients to zero, which eliminates less important features. The optimization balances minimizing the residual sum of squares and the absolute sum of coefficients as shown in algorithm 4[26][38].

Algorithm 4 Lasso Regression

```
1: Input: Training data $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^n$, regularization parameter $\lambda \geq 0$
2: Output: Coefficient vector $\hat{\beta}$
3: Initialize coefficients $\beta \leftarrow \mathbf{0}$
4: repeat
5:     for each feature $j = 1, \ldots, p$ do
6:         Update coefficients
7:         $\hat{\beta} \leftarrow \arg\min_\beta \frac{1}{2n} \sum_{i=1}^n \left(y_i - \sum_{k \neq j} x_{ik}\beta_k - x_{ij}\beta_j\right)^2 + \lambda|\beta_j|$
8:     end for
9: until convergence criteria met
```

3.4 Machine Learning Classification and Algorithms

In this section machine learning algorithms for classification tasks will be described. Here nearest neighbor algorithms are described due to their efficacy in MFP tasks, their ability to be adapted for time sequence data, as well as their use for synthetic data generation in SMOTE[34][29]. Random Forest is also described, as it is the most common and generally best performing algorithm within predictive maintenance[10]. Furthermore, random forest has built in feature selection, and can be adapted for time sequence data[39]. These methods lay the foundation for answering both inquiries and research question presented in the introduction 1.

3.4.1 Nearest Neighbor Algorithms

Nearest-neighbor classification are non-parametric machine learning algorithms, in which a training set $\{(x_n, y_n)\}_{n=1}^{N}$ is stored with samples from each class. Test samples x_0 are then classified by assigning the label of the closest training samples. A K-nearest neighbor (k-nn) conducts a majority vote of the k closest training points. Closeness is a distance measure, and often the Euclidean distance $c(x_0, x_i) = ||x_0 - x_i||$[40][41]. This is also described in algorithm5.

Algorithm 5 k-Nearest Neighbors Algorithm

```
1: $\{(x_i, y_i)\}_{i=1}^{N} \leftarrow$ training data
2: $x_{\text{test}} \leftarrow$ test data
3: $k \leftarrow$ number of nearest neighbors
4: **for** each $x_j$ in $x_{\text{test}}$ **do**
5:     $c \leftarrow [\,]$
6:     **for** each $x_i$ **do**
7:         $c[i] \leftarrow c(x_{\text{test}}, x_i)$
8:     **end for**
9:     $y_j \leftarrow$ majority vote of $y_i$ for $k$ closest $x_i$ measured by $c[i]$
10: **end for**
```

For time series data, DTW distance can be used as the cost $c(x_{\text{test}}, x_i)$. This is especially useful in cases where other distance measures are not appropriate due to shifting or unequal signal length. In these cases DTW can be used as a distance measure instead in a DTW-nearest neighbor (DTW-nn), which in many cases lead to improved results over traditional K-nn[34].

3.4.2 Decision Trees and Random Forest

Decision trees are traditionally a classification method but can also be used for regression. Decision trees fundamentally work by using recursive partitioning of instances in order to reach a decision[42]. They consist of a root node that branches to other internal nodes, which contain features, and to leafs where decisions are made. They are described by a maximum amount of internal nodes, given by the depth of the tree[43]. An example tree can be seen in figure 3.1, with the root node seen in green, the deepest node in red, and the leafs are marked as triangles.

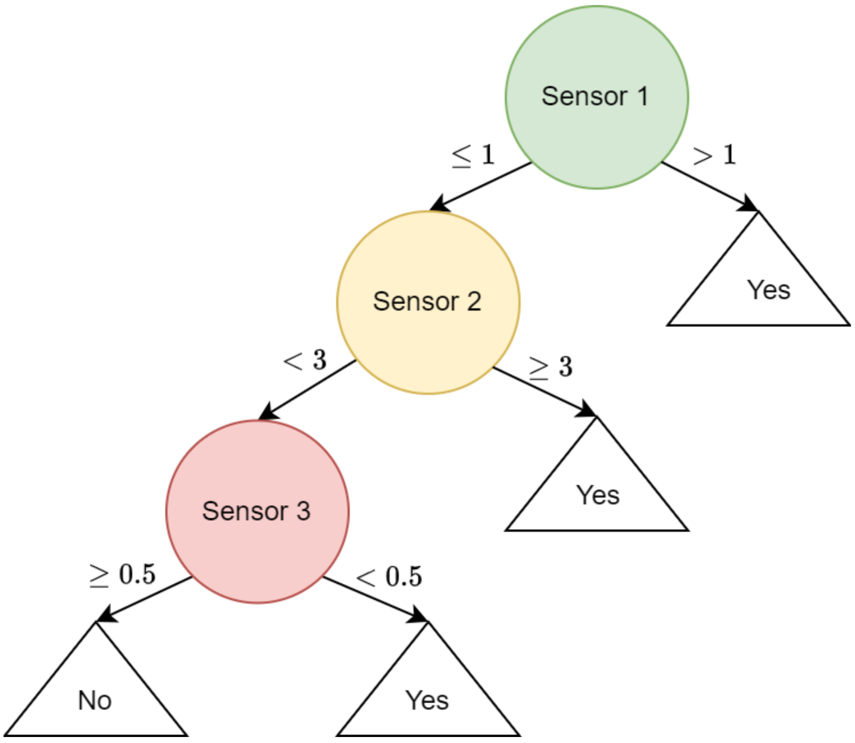


Figure 3.1: Example of a decision using sensor measurements from three sensors in order to predict a failure.

Decision trees are trained on data by minimizing the generalization error, or until a max number of nodes or depth is reached. They are learned in either a top-down or bottom-up manner[43]. Top-down trees function in a greedy manner, starting at the start node, and finding the best features and splits until the tree is full[44]. Bottom-up methods start with full trees, and go back up through the tree eliminating redundant nodes, this is also known as pruning[45]. The most common method is top-down[43]. Here the tree recursively learns the best feature and split, according to an impurity measure, this is done until some stopping criterion is met. This can be combined with bottom-up pruning, in order to remove redundant nodes and avoid overfitting[45]. This is described in algorithm 6.

Algorithm 6 Decision Tree Learning Algorithm (Classification)

```
1: $\mathcal{D} \leftarrow$ training data with features $F = \{f_1, f_2, \ldots, f_n\}$ and labels $Y$
2: Set stopping criteria (e.g. max depth, min samples)
3: repeat
4:     for $f \in F$ do
5:         for each possible split $s$ of $f$ do
6:             Split $\mathcal{D}$ according to $s$
7:             $L(f, s) \leftarrow$ Compute impurity of split (e.g. Gini index, entropy)
8:         end for
9:     end for
10:    $f^*, s^* \leftarrow min_{f,s} L(f, s)$
11:    Create node split on $f^*$ at $s^*$
12: until Stopping criteria met
13: Prune tree
```

The impurity measure depends on the task. For classification tasks the most common are entropy and Gini index. These are described in equations 3.5 and 3.6 respectively where Q_m represents the node m with (f_m, s_m), p the proportion of samples of class i in node m with a total of C classes, and n total samples[46].

$$L(Q_m) = -\sum_{i=1}^{C} p_{i,m} \log_2(p_{i,m}) \tag{3.5}$$

$$L(Q_m) = 1 - \sum_{i=1}^{C} p_{i,m}^2 \tag{3.6}$$

For regression, the mean square error described in equation 3.7 is commonly used.

$$L(Q_m) = \frac{1}{n_m} \sum_{y \in Q_m} (y_m - \bar{y}_m)^2 \tag{3.7}$$

Some the most common stopping conditions include; stopping at max depth, pure node is reached, best splitting purity is below some threshold, the number of cases in the terminal node is less than the minimum number of cases in the parent nodes, and if the child nodes of a split contain less than a minimum number of cases. These stopping criteria are also referred to as pre-pruning, as it prevents the tree from overfitting by stopping it early[43]. Post pruning runs bottom-up through the tree, removing branches by evaluating how important the branch is for the tree. This can be done by measuring the miss-classification impact of the branch through the error rate[45].

Random forest are ensembles of decision trees. Ensemble learning methods combine the outputs of multiple models to produce more accurate and robust predictions. The idea of random forests is to create multiple decision trees to reach a single result. Since this is the case, the model is generally more accurate than just using decision trees. A random forest model is also able to capture nonlinearities in the data, which can be beneficial when working with large and complex datasets [47]. They are resistant to noise and outliers, manages high-dimensional datasets effectively and yield estimates of feature relevance. Decision trees are created from the original dataset using bootstrapping, where each tree is trained on a randomly sampled subset of the data, with replacement. Some of the data, typically around one third, will not be used in the training of a particular tree. These are referred to as out-of-bag (OOB) samples and are later used to validate the model. In addition to bootstrapping the data, random forests also introduce randomness during tree construction by selecting a random subset of features at each split. This added randomness increases the diversity among the trees, helping to reduce overfitting and improve generalization [48]. When all trees have been built, the out-of-bag samples are run through the trees that did not train on them. Each tree "votes" for the class it predicts the sample belongs to. The votes from all trees are then counted, and the majority determines the final predicted class[48]. For regression tasks, the model instead outputs the average of the trees' predictions. The training structure of the random forest model can be seen in figure 3.2.

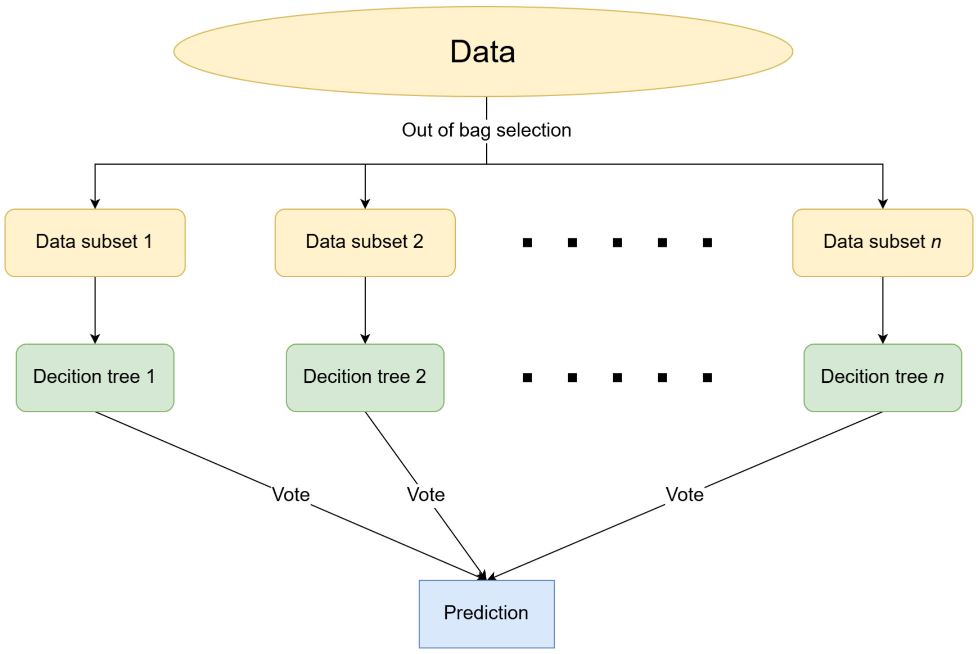


Figure 3.2: The structure of the random forest training process.

The model also uses the OOB samples to estimate the importance of each feature in the dataset. This is done using a technique called permutation importance. The OOB samples are run through the trees multiple times, but with the values of a specific feature randomly permuted. This breaks the relationship between that feature and the target variable. The model still makes a prediction, but its accuracy typically decreases. The difference in accuracy before and after permutation is used to estimate the importance of the feature. A larger drop in accuracy indicates a more important feature. These importance scores can then be normalized and are often presented as values between 0 and 1, where a higher value indicates greater importance [48].

In cases where data has a temporal or sequential structure, such as time series, random forests can still be effectively applied by incorporating lagged features. These are created by including past values of variables as additional input features, allowing the model to capture temporal dependencies even though the algorithm itself is not inherently time-aware. This approach extends the flexibility of random forests and enables them to perform well on a wide variety of prediction tasks involving historical data. Overall, random forests are a powerful and flexible modeling technique that perform well on a wide range of tasks[39].

3.5 AI for time sequential failure prediction

Given that both tasks deal with time sequential data, AI models that deal well with sequential data are reviewed. Some of these were discussed in section 2.1, where the most commonly used AI techniques for time sequence data are Convolutional neural networks (CNN), Long Short-Term Memory Networks (LSTM), and Transformer networks. These models are described in order to answer the primary inquiry and research question.

3.5.1 Convolutional Neural Network

Convolutinal neural networks use convolutional layers, and are traditionally used in image recognition. The convolutional layers extract features through convolving a kernel over an image, and pooling the results in order to extract features from the image. The extracted features can then be linearized, and used in a neural network for regression or classification[49]. An example architecture is illustrated in figure 3.3.

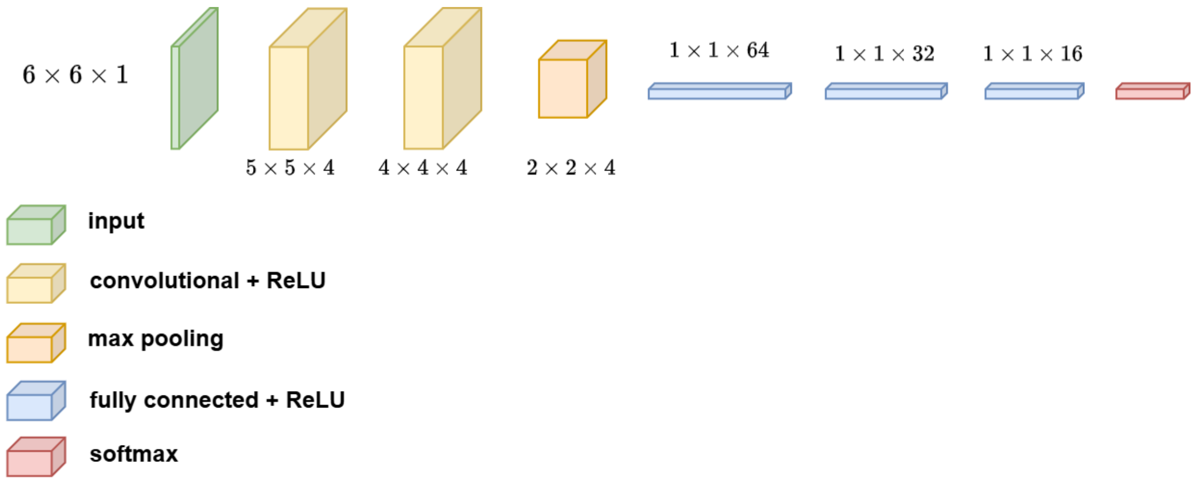


Figure 3.3: Example CNN architecture for classification with input size $6 \times 6 \times 1$, a filter bank with four filters and kernels of size 2×2, stride of 1, no padding, and using max pooling.

As mentioned the convolutional layers use matrix convolution. A filter bank, consisting of multiple learnable kernels, are convolved over the image, capturing different local features of an image. Pooling is then used, to extract the most meaningful feature from the kernels, thereby reducing dimensionality of the image. In general two types of pooling exists, max pooling and mean pooling[49]. An example of convolution and max pooling can be seen in figure 3.4.

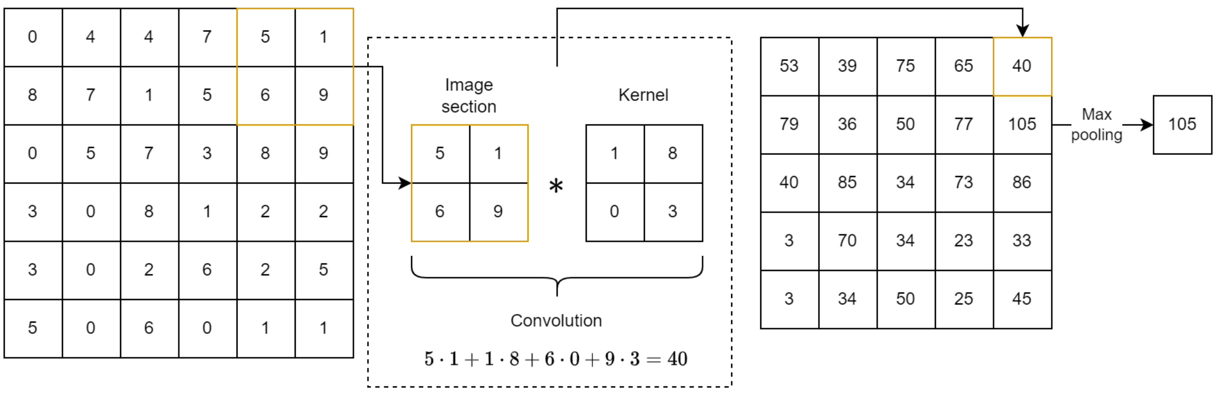


Figure 3.4: Example of a convolution and max pooling for an image of size $6 \times 6 \times 1$, kernel size of 2×2, and a stride of one.

The convolutional layers can be controlled through the hyperparameters stride, depth, zero-padding, and pooling size. The stride decides the overlap between kernel convolutions. A stride of one glides the kernel one step when convolving, as seen in figure 3.4. Depth is the number of kernels in the filter bank, and determines the number of output channels of the convolutional layer. Zero padding pads the borders of the image, such that the edges can be convolved. Lastly, pooling size decides the shape of the pooling, and thereby how much dimensionality is reduced. The convolutional layers also make use of activation functions such as ReLU. Once features have been extracted from convolutions, they can be flattened and used in a neural net[49][50].

While CNNs are developed and most commonly used for 2D images, they also have applications within 1D signal processing, where they have showed good performance for limited labeled data. The main difference between 1D and 2D convolution is the kernel and stride, since these are one dimensional in the 1D case compared to the classic 2D case[50].

3.5.2 Long Short-Term Memory Network

Long short-term memory (LSTM) networks are a form of recurrent neural networks (RNN), that capture long term dependencies while avoiding the problems with vanishing and exploding gradients. It does this by using memory cells with state c_t at sequence time t, each cells consists of three gates, an input gate i_t, an output gate o_t, and a forget gate f_t. Each cell takes in c_{t-1}, h_{t-1} which is the latent representation of the previous layer, and x_t which is the current input. The cell outputs the cell state c_t, and latent or hidden representation h_t. A single LSTM memory block can be seen in figure 3.5, and is described by equation 3.5.2[51][52].

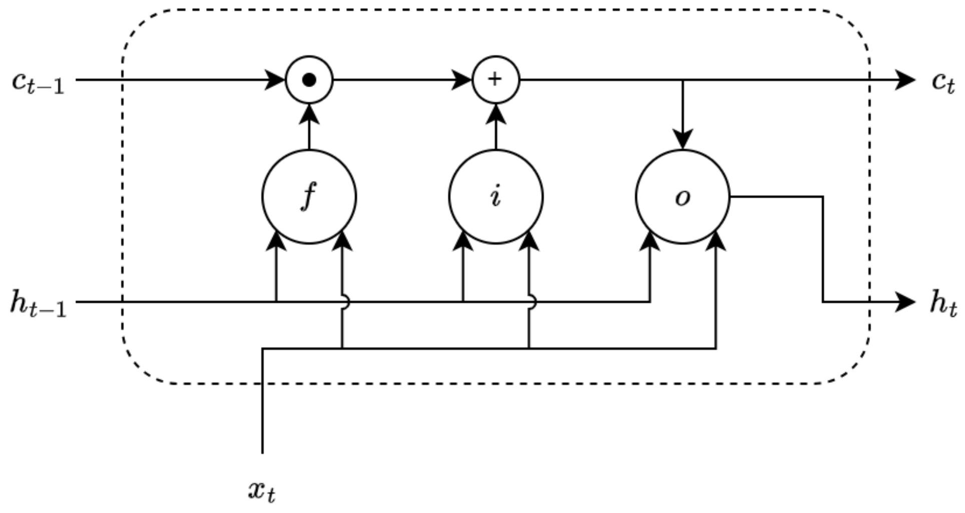


Figure 3.5: Single LSTM memory block.

$$
\begin{gathered}
f_t = \sigma_g(W_f x_t + R_f h_{t-1} + b_f) \\
i_t = \sigma_g(W_i x_t + R_i h_{t-1} + b_i) \odot \sigma_c(W_c x_t + R_c h_{t-1} + b_c) \\
o_t = \sigma_g(W_o x_t + R_o h_{t-1} + b_o)
\end{gathered}
$$

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \\ h_t &= o_t \odot \sigma_c(c_t) \end{aligned} \tag{3.8}$$

All matrices W and vectors b are weights and biases respectively, that can be learned through backpropagation. σ indicates activation functions, where σ_g are gate activations, and σ_c are cell activations, these are commonly sigmoid and tanh respectively[52]. The final hidden state can be used in neural nets for regression and classification tasks.

Bidirectional LSTM networks (Bi LSTM) are LSTM networks which evaluate a sequence in both a forwards and backward direction, combining the output from both. Depending on the task this can boost performance[53]. LSTM networks have performed well for several tasks such as language modeling, sequence denoising, sequence generation, and sequence forecasting, generally proving effective and well performing for time sequence problems[52][54].

A drawback of using a normal LSTM models is difficulty learning positional importance in sequences. To try and combat this an attention mechanism can be implemented which helps to learn positional importance in the latent representations[55]. The attention mechanism allows the model to better focus on the most relevant parts of the input sequence when making predictions. Instead of using only the final hidden state, attention gives a weighted sum of all hidden states seen in equation[56].

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T] \tag{3.9}$$

The attention mechanism computes an attention score α_t for each timestep t. This score reflects the importance of the hidden state $\mathbf{h}_t$ for the current prediction and is learned from a single perceptron layer. The perceptron weights are used to compute the score as seen in equation 3.10.

$$\alpha_t = \mathbf{w}^\top \mathbf{h}_t + b \tag{3.10}$$

$\mathbf{w}$ and b are trainable parameters of the attention layer and are found through a feedforward network. This allows the model to learn attention through training of the network. α_t is normalized using a softmax function to obtain attention weights. This is described in equation 3.11.

$$a_t = \frac{\exp(\alpha_t)}{\sum_{k=1}^{T} \exp(\alpha_k)} \tag{3.11}$$

Finally the context vector $\mathbf{c}$ is computed as the sum of the LSTM hidden states, weighted by their corresponding attention as described in equation 3.12.

$$\mathbf{c} = \sum_{t=1}^{T} a_t \mathbf{h}_t \tag{3.12}$$

The context vector serves as the input to the downstream regressor or classifier neural net. The benefit of using attention in a LSTM model is that the model focuses on the most relevant parts of the input sequence in its predictions. Furthermore, this adds better interprebility to the model since the magnitude of the weights translates to the importance of the part of the sequence it represent[55][56].

3.5.3 Transformers

Transformers are a parametric machine learning method that uses attention to model sequence transduction tasks, such as natural language processing and computer vision, by learning sequence dependencies and context[57]. A model of the transformer architecture can be seen in figure 3.6.

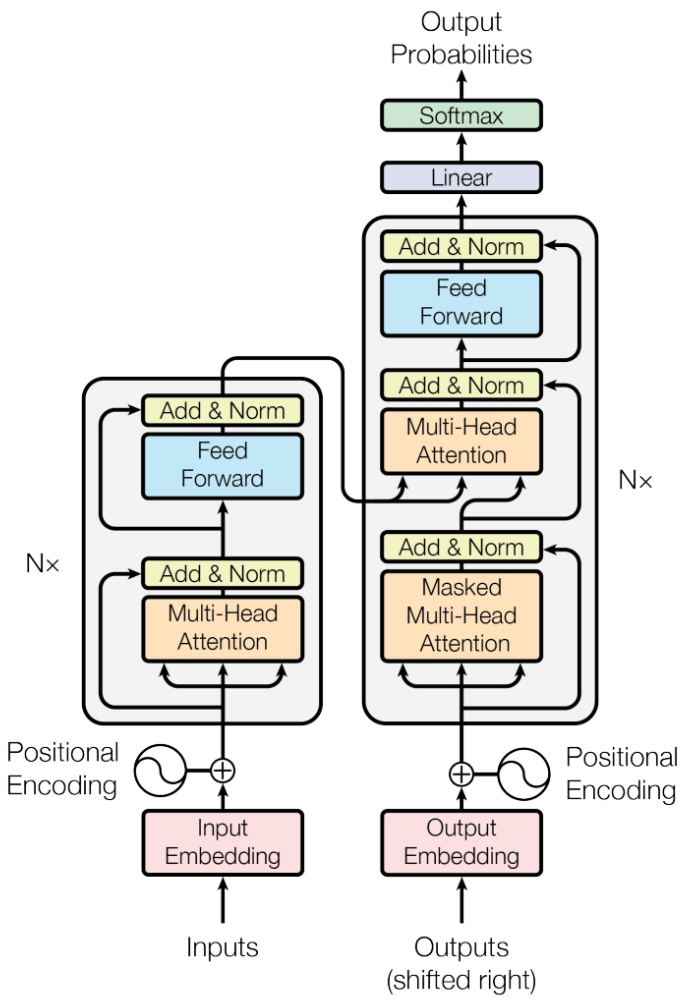


Figure 3.6: The transformer architecture as described in Ashish Vaswani et. al.[58].

The data sequence input is encoded by the input embedding. This assigns a vector to each token. The vector representation of each token is parsed through a positional encoder, which encodes information about the token's position in the sequence. After input encoding, the sequence is passed through self-attention, where the input finds context within itself. This is done through a multi-head attention block which is described in figure 3.7.

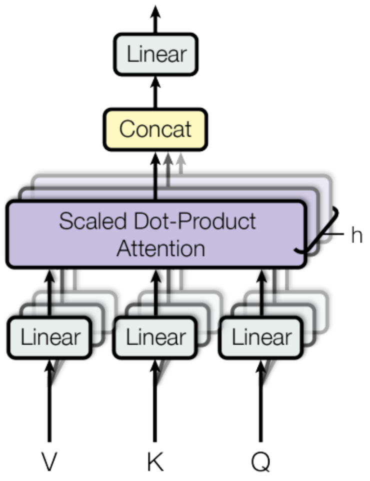


Figure 3.7: Multi-Head attention as described in Ashish Vaswani et. al.[58].

Multi-head attention is based on scaled-dot-product attention. Here a query matrix Q is associated with the encoded input sequence, each vector in the matrix represents a "question" which a token asks the other tokens. A key matrix K represents vectors with token "answers" to the queries. The dot product between the query and key indicates how much tokens are contextualized by each other. A value matrix then contains vectors that show how each token contextualizes each other. This is also described by an equation, where the attention is normalized by the query-key dimension. This is given by equation 3.13, where queries $Q \in \mathbb{R}^{N \times D_k}$, keys $K \in \mathbb{R}^{M \times D_k}$, and values $V \in \mathbb{R}^{M \times D_v}$, with N and M denoting the lengths of queries and keys (or values), and D_k and D_v denoting the dimensions of keys (or queries) and values[58].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{D_k}}\right)V \tag{3.13}$$

To then get multi-head attention the queries, keys and values are then projected H times with linear projections to d_k, d_k and d_v dimensions, respectively[58]. This allows for attention to be performed in parallel which keeps the same computational complexity as single-head attention but increases performance of the model[58]. Multi-head attention is given by equation 3.5.3.

$$\begin{aligned} \text{MultiHeadAttn}(Q, K, V) &= \text{Concat}(\text{head}_1, \cdots, \text{head}_H)W^O \\ \textbf{where } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.14}$$

One motivating factor for using transformers is that they excel at handling both long and short sequences of data due to their self-attention mechanism. This is because of how it uses the query-key feature[59]. Another benefit of using transformers is that they process the entire sequence in parallel. This is also the reason for why a positional encoder is necessary since the transformer is not able to understand sequence structure without it.

3.6 Data augmentation

As mentioned in section 2.3 having an equally sampled dataset is important. Furthermore, when training parametric deep learning networks, greater data variance significantly improves learning especially for larger networks[60][61]. Given limited data availability of the failures discussed in this thesis, it is necessary to generate augmented data for training, based on the existing data.

As also mentioned in section 2.3 SMOTE is a popular technique for generating synthetic minority class samples. This is done through K-nn, and the technique can in general be used to generate any number of augmented samples from any class. The augmetation method is summarized in algorithm 7[29][62].

Algorithm 7 SMOTE algorithm which generates synthetic minority class samples.

```
1: inputs:
       Minority class samples $X$, oversampling amount $N$, number of
       nearest neighbors $k$
2: $X_s \leftarrow$ empty array of length $N$ to store synthetic samples in
3: while $j = 1$ to $N$ do
4:     for sample $x_i \in X$ do
5:         Find $k$ nearest neighbors of $x_i$
6:         Select random nearest neighbor $x_k$ of $x_i$
7:         $\delta \leftarrow \mathcal{U}(0, 1)$
8:         $X_s[j] \leftarrow x_i + \delta(x_k - x_i)$
9:     end for
10: end while
```

In essence synthetic samples are generated along the line segment between a samples and one of its nearest neighbors.

DTW-nn also has applications within SMOTE. Here it replaces the traditional K-nn method with DTW-nn to find nearest neighbors. Instead of generating a synthetic samples as in algorithm 7, the sample is generated along the best aligned path found from DTW. DTW-SMOTE is described in algorithm 8[63].

Algorithm 8 DTW-SMOTE algorithm for generating synthetic minority class time series samples.

```
1: inputs:
       Minority class samples $X$, oversampling amount $N$, number of
       nearest neighbors $k$
2: $X_s \leftarrow$ empty array of length $N$ to store synthetic samples
3: while $j = 1$ to $N$ do
4:     for each sample $x_i \in X$ do
5:         $d \leftarrow$ empty array to store distances
6:         for $x_k$ in $X$ that is not $x_i$ do
7:             $d(i,k) \leftarrow DTW(x_i, x_k)$
8:         end for
9:         Find $k$ nearest neighbors of $x_i$ based on $d(i,k)$
10:        Select a random neighbor $x_k$ from the $k$ nearest neighbors
11:        $\pi^* \leftarrow$ optimal warping path between $x_i$ and $x_k$ using DTW
12:        $x_s \leftarrow$ empty sequence to store synthetic sample
13:        for each $(i,j) \in \pi^*$ do
14:            $\delta \leftarrow \mathcal{U}(0,1)$
15:            $x_s[p] \leftarrow x_i[p] + \delta \cdot (x_k[q] - x_i[p])$
16:        end for
17:        $X_s[j] \leftarrow x_s$
18:    end for
19: end while
```

4 | Proposed Feature Engineering and Failure Prediction Methods

In this chapter the proposed feature engineering and failure prediction methods are described, alongside the data pipeline for training the models. The methods described in this chapter will both be used and applied on the accumulator failure data and methanol changeover data. The feature engineering will be based upon a feature selection algorithm, which aims to reduce dimensionality, deal with spurious correlations, while, emphasizing feature importance. The outlined method will ultimatemly be used to answer the second inquiry *"How can feature engineering be used to create a feature space suited for prediction models, while giving insight into the system failures?"*.

The prediction methods proposed in this chapter are two fold, where one uses binary classification and a Prediction Window, and the second, Remaining Useful Life prediction. The first prediction problem is formulated as follows;

Can a system failure be predicted through binary classification, by predicting if a failure happens within a Prediction Window, given data read from a Reading Window?.

From the literature it is found that this method of binary classification works well for time series data, given appropriate window sizes. The second prediction problem is formulated as;

Can the Remaining Useful Life of a system be predicted through regression, given data read from a Reading Window?

This method is also pursued in literature but the same effect of RW size has not been explored in the same capacity as the binary classification problem.

For the two prediction tasks, different models are used, including Random Forest with lags, 1D CNN, and LSTM networks. The model choices are fundamental to answer the inquiry *"Can ML and AI models be used to predict marine propulsion system failures?"*, as well as the research question *"to what extent can ML and AI be used to predict and prevent total system failure in MAN Energy Solution's maritime propulsion systems?"*.

4.1 Data pipelining for failure prediction

The pipeline for training and validating failure prediction models include reading the data, windowing the data based on the reading window (RW), labeling the data according to the prediction window (PW) or the remaining useful life (RUL), generating augmented samples using SMOTE, training and validating the model, and finally testing the model with unseen data. Each part of the pipeline, summarized in figure 4.1, is described in this section.

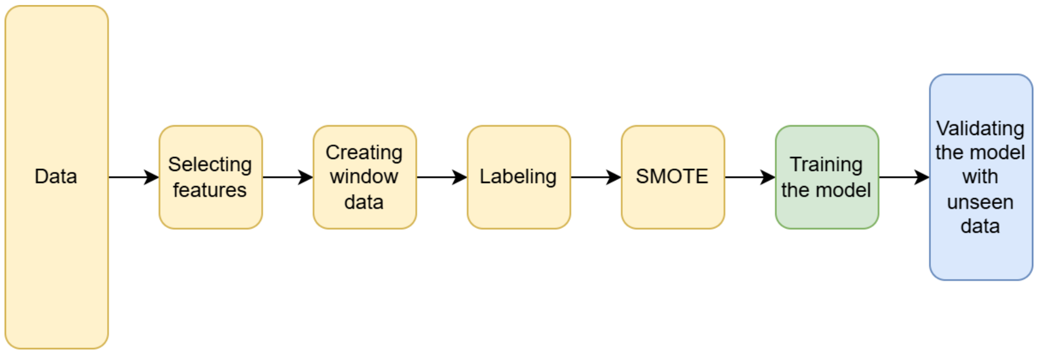


Figure 4.1: Diagram showing the pipeline for the data to be ready for one of the failure prediction models.

4.1.1 Data description

The data from the Posorja Express is labeled according to previous work as described section 5.1.1. The data is segmented into smaller time sequences which are labeled accordingly. The segments are 8000 samples long, corresponding to 2.2 hour, ensuring sufficient RW sizes for finding temporal dependencies, and data diversity. In total 100 stable segments and 15 failure segments are used. The windowing and labeling of the segments is described in section 4.1.3. To reduce dimensionality, as well as remove spurious correlations, the algorithm described in section 4.1.2 is used.

In order to find a solution to the problem of knocking and damage of supply pipes mentioned in section 1.2.2, service tests were conducted. Here measurement setups were setup, in order to measure the system and identify the failure cause. It is this data which will be used for developing changeover failure prediction models. In this case the data is pre-labeled, with the files obtained from the tests containing either a successful or failed changeover sequence.

The methanol changeover data has been measured with 20 kHz for several different tests, containing seven successful changeovers, and eight which failed. Both are used when training failure prediction within a prediction window. In case of RUL prediction, only the failed changeovers are used for training. Additionally, for RUL prediction only six of the eight failed changeovers are used, due to faulty data loading. Exact windowing and labeling of the data is described in section 4.1.3. Due to the high sampling frequency, it was decided in section 5.2.3 to only use the maximum value per injection. Furthermore, to reduce dimensionality and remove spurious correlations, the algorithm in section 4.1.2 i used. For classification with a PW, sequences of 78 injections are used, as this is the smallest number of injections required for the human operator to detect a failure. For RUL prediction, the full sequence until failure is used, meaning varying sequence lengths.

4.1.2 Algorithm based feature selection

In order to select an appropriate subset of features, redundant information should be removed, as mentioned in section 1.4. As also mentioned in section 1.4, features should also have a high relation to the class. While methods like greedy elimination, and Lasso regression deal well with elimination feature-feature correlations, they do not consider each individual features relation to class. On the other hand, CFS considers both feature-feature correlation, as well as feature-class correlation, however this comes at the cost of computation time, as all possible subsets of features need to be computed. In an effort to eliminate high feature-feature correlations, while emphasizing high feature-class relationship, algorithm 9 is developed based on recursive feature elimination (RFE).

Algorithm 9

```
1: Input: Features $F$, feature-feature correlation matrix $C_{ff}$, feature-class importance
   $I_{cf}$, threshold $\varepsilon$
2: Output: Feature subset $F^*$
3: $F^* \leftarrow F$
4: high correlation $\leftarrow$ True
5: while high correlation = True do
6:     high correlation $\leftarrow$ False
7:     for each feature pair $(i, j) \in F$ do
8:         if $|C_{ff}(i, j)| \geq \varepsilon$ then
9:             high correlation $\leftarrow$ True
10:            if $I_{cf}(i) > I_{cf}(j)$ then
11:                Replace $j$ $i$ in $F^*$
12:            end if
13:            if $I_{cf}(j) \geq I_{cf}(i)$ then
14:                Replace $i$ with $j$ in $F^*$
15:            end if
16:        end if
17:        Re-evaluate $C_{ff}$ for $F^*$.
18:        Sort feature pairs $(i, j) \in F^*$ from largest to smallest according to $C_{ff}(i, j)$
19:    end for
20: end while
21: return $F^*$
```

Although algorithm 9 lends from RFE, taking into account feature importance, it diverges from traditional RFE by using feature-feature correlation threshold rather than continually considering smaller and smaller feature subsets until a desired number of features is reached[28]. The measure of feature importance will depend on the task, and will be described for the accumulator failure, and methanol changeover failure in sections 5.1.4 and 5.2.1 respectively.

4.1.3 Windowing and labeling

In general, data for failure classification, using a prediction window, is a binary classification with labels (1) and (0), representing a failure and stable conditions respectively. Data is windowed according to the RW, which slides along the data sequence. If the reading window is outside the PW the window is labeled as (0), if the RW is within the PW it is labeled as (1). This windowing and labeling is illustrated in figure 4.2

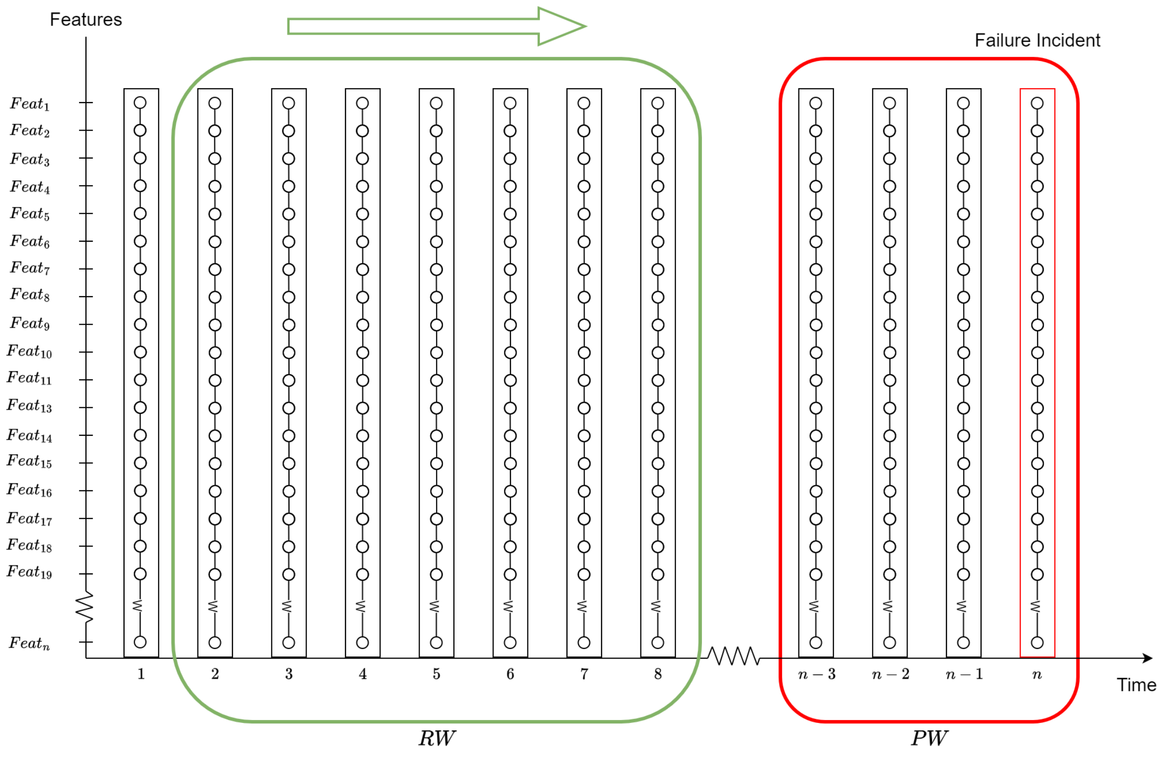


Figure 4.2: Illustration of a sliding reading window over all features at each time step with a prediction window at the end of the sequence where system failure occurs.

In cases where $PW < total\, sequence\, length$ there will be windows of data corresponding to the lead up to a failure, which are labeled as (0). A window is labeled as (1) when the RW enters the PW, as described in figure 4.2.

Data used for RUL prediction uses only failure sequences of varying length L_S leading up to and including a failure. The label of each window is the distance from the end of the RW until the failure occurs. Again a sliding window with size RW is used to window the data. The windowing and labeling of RUL is described in figure 4.3.

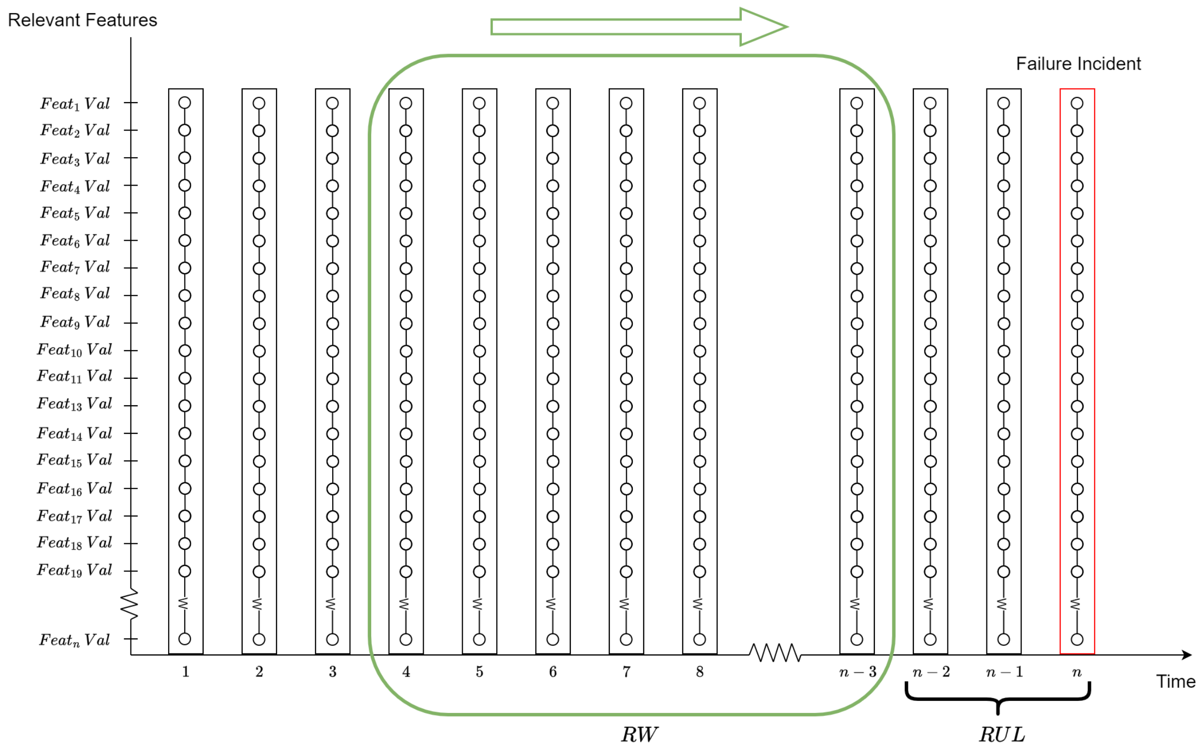


Figure 4.3: Plot showing the reading window over the features in each time stamp moving towards a failure incident and deciding the remaining useful life.

4.1.4 SMOTE and dataset balancing

Due to the limited data, an issue arises when training a failure prediction models, as there is very little data variance. To avoid this, augmented data is generated using SMOTE as described in section3.6. The SMOTE algorithm generates new class samples with its nearest neighbors, creating augmented data within the class distribution. In the general the datasets should be balanced, with an equal amount of data labeled (1) and (0). SMOTE is used to generate augmented failure samples, increasing the class by N fold. In cases where there is lead up data, the dataset is balanced such that $N_{stable} = N_{lead\ up} = N_{failure}/2$, using either random sampling or SMOTE. In cases where $PW = L_S$, meaning no lead up, the dataset is balanced such that $N_{stable} = N_{failure}$, again using random sampling or SMOTE.

In the case of Posorja Express the data is heavily skewed towards stable data. Therefore, SMOTE, as seen in algorithm 7, is used to oversample failure instances, and random sampling is used to downsample stable data. In the methanol changeover data the dataset is not as unbalanced. However, it contains very few sequences. Therefore, SMOTE is used to generate both stable and failure samples, to increase the amount of total data and data variance. Given the uneven sequence lengths in methanol changeover RUL prediction, the DTW-SMOTE algorithm 8 is used to augment full sequences.

4.2 Proposed failure prediction models

In this section the proposed models for failure prediction are presented. Random forest with lags, 1D CNN, and LSTM networks are chosen for their failure prediction performance for time sequential data, as demonstrated by the literature review in chapter 2. Transformer are not pursued based on strong recommendations from MAN ES from prior experience.

4.2.1 Random forest with lagged data

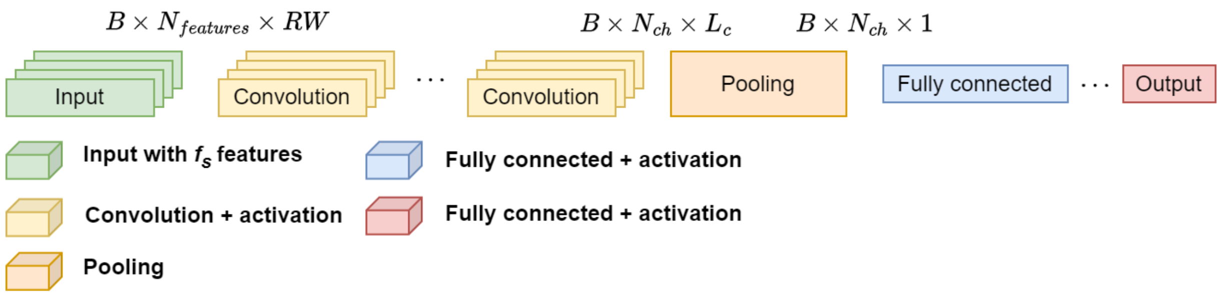
As mentioned in section 3.4.2 Random Forest are not inherently meant for multivariate time sequential data. However, they can be adapted for it using lagged features. In practise, this is done by flattening the multivariate time sequences such that that shape goes from $(N_{features},\ RW) \rightarrow (1, N_{features} \cdot RW)$. The model works by sliding the RW and making a prediction based on the data in the RW as described in section 4.1.3. As mentioned, the tasks involving a prediction window are classification tasks, and the RUL prediction is a regression task. For the two different tasks classification and regression trees are used respectively. In essence this means the use of different impurity measures, best suited for the purpose. Gini impurity is used for classification, due to its reduced computational complexity compared to cross entropy. For regression the mean square error is used, treating the regression as a least squares problem. The random forest models take in the hyperparameter n_estimators, which dictates the number of decision trees used in the random forest. The specific tree type, loss function, and n_estimators for each task are described in chapter 6.

4.2.2 1D Convolutional Neural Network

As described in section 3.5, a 1D CNN finds temporal features by convolving across one dimension, and extracting features. The data shape input in a 1D CNN is
$(B,\ N_{channels},\ N_{time\,steps})$, for a batch size of B. When applying this to the machine failure prediction problems outlined, the shape becomes $(B,\ N_{features},\ RW)$, with channels equal to the selected features, and the number of time steps corresponding to reading window size. Each convolutional layer changes the length of the original sequence with length L_S according to equation 4.1, where the convolutional layer has a kernel size of k_s, padding of p, and stride s.

$$L_c = \frac{L_S + 2p - k_s}{s} + 1 \tag{4.1}$$

The convolutions are then pooled to extract features, either through max or average pooling. After convolution and pooling, fully connected layers are used in order to obtain the desired output shape. Both convolutional and fully connected layers can use activation functions such as ReLU to model nonlinearities, or Sigmoid or softmax to fit the output for classification. Furthermore, the layers may use dropout to avoid overfitting. All attributes, as well as network depth and size, will depend on the prediction task, and are described in chapter 6. In figure 4.4 the general architecture of the 1D CNN can be seen. The input size depends on the number of features $N_{features}$ from the feature selection process, as well as the reading window size RW. The last layer is a pooling layer, which ensures flattening of the of the time sequence. The original feature size is scaled from $N_{features}$ to a number of channels N_{ch}.



$B \times N_{features} \times RW$
$B \times N_{ch} \times L_c$
$B \times N_{ch} \times 1$
Input
Convolution
...
Convolution
Pooling
Fully connected
...
Output
Input with f_s features
Convolution + activation
Pooling
Fully connected + activation
Fully connected + activation

Figure 4.4: General 1D CNN architecture for failure prediction.

1D CNNs are well suited for both classification and regression, depending on the loss function and output layer. Binary cross entropy is used as the loss function binary classification, and mean square error loss is used for regression tasks. For the output layer, Sigmoid activation is used for binary classification, and no activation is used on the output layer for regression models. In chapter 6 the loss function, as well as the optimizer, activation functions, learning rate, and batch size, used for training each specific model, are described.

4.2.3 LSTM networks

As discussed in section 3.5.2 LSTM networks capture long term dependencies. They take in sequential data in the shape $(B, N_{time\ steps}, N_{features})$ which equates to $(B, RW, N_{features}$ for batch size B, and are suited for both classification as well as for regression tasks. In this case they are used for both in cases of longer reading windows, as they can capture long term dependencies that models like 1D CNN or random forest with lags might miss. LSTM networks are constructed from one or more LSTM blocks with input, output, and forget gates. Each LSTM block has a hidden size h_s, corresponding to the number of output features of the block. The blocks are often combined with fully connected linear layers, to output the desired prediction. The linear layers may also use activation functions such as ReLU to capture non-nonlinearities, or Sigmoid for binary classification. Furthermore, the linear layers may also use dropout to avoid overfitting. The LSTM output has the same length as the input sequence L_s, in order to vectorize it for the fully connected layers two main techniques are used. Firstly, the last time step of the LSTM output can be used, as it has captured context from previous steps. Alternatively, an attention mechanism can be used, to capture positional importance in the time sequence as described in section 3.5.2. The general architecture for the LSTM models used for the tasks in this thesis can be seen in figure 4.5.

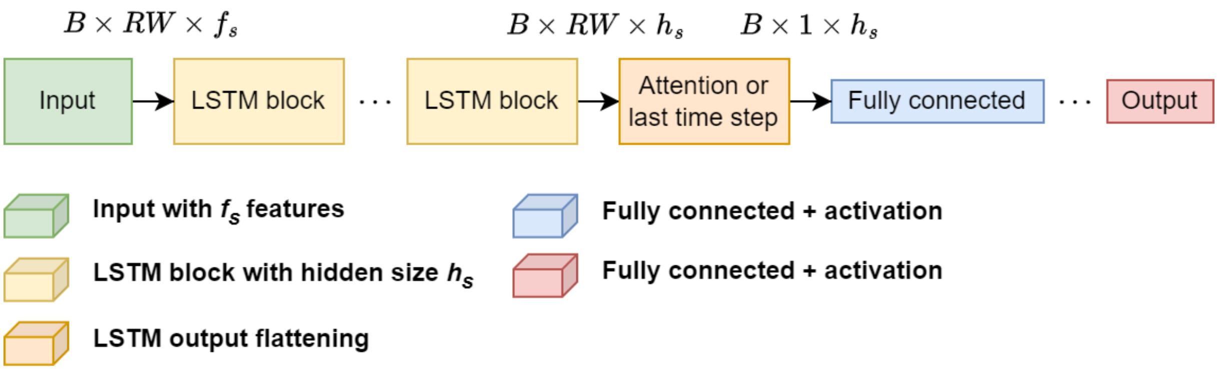

$B \times RW \times f_s$
$B \times RW \times h_s$
$B \times 1 \times h_s$
Input
LSTM block
LSTM block
Attention or last time step
Fully connected
Output
Input with f_s features
LSTM block with hidden size h_s
LSTM output flattening
Fully connected + activation
Fully connected + activation

Figure 4.5: General LSTM network architecture

LSTM networks can be trained with different loss functions depending on the task. Binary cross entropy is used as the loss function binary classification, and mean square error loss is used for regression tasks. Model training performance also depends on the optimizer used, as well as the hyperparameters batch size, learning rate, dropout, and RW. The choice of hyperparameters greatly impact model training and performance. The specific dimensions, depth, activation functions, hyperparameters, loss functions, and optimizers used are described in chapter 6 for each task.

4.2.4 Model training and validation

Once data has been read, features have been selected, data has been windowed, labeled, and synthetic samples have been generated, the model can be trained. For the Random Forest this is done according to algorithm 10, where trees in the random forest are trained according to algorithm 6.The random forest algorithm takes in training and validation data, together with the number of decision trees used n_estimators. The training and validation loop works by the model generating a random forest with n_estimators trees, fits the model using a specified impurity measure, and validating the model on validation data.

Algorithm 10 Random Forest Training and Validation

```
1: Input: Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, model $M$, impurity measure $\mathcal{L}$, number of trees $n\_estimators$
2: Output: Validation prediction $(y_{prediction},\, y_{true})$
3: Initialize model $M$ with $n\_estimators$
4: Train model $M(\mathcal{D}_{\text{train}}, \mathcal{L})$
5: Validate model $y_{prediction} \leftarrow M(\mathcal{D}_{\text{val}})$
```

For the AI models (CNN, and LSTM), the training loop described in algorithm 11 is used. Batch training is used for parallel computing and generalization with a specified batch size. The model is trained over several epochs with early stopping to avoid overfitting. Early stopping works by stopping training, and restoring the best model, if further training hurts validation results. The model is fitted using a specified loss function and optimizer, depending on the task.

Algorithm 11 AI Training and Validation with Early Stopping

```
1: Input: Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, model $M$, loss function $\mathcal{L}$, optimizer $O$, max epochs $E_{\max}$, patience $p$
2: Output: Trained model $M$
3: Initialize model parameters $\theta$
4: $best\_val\_loss \leftarrow \infty$
5: $patience\_counter \leftarrow 0$
6: for $epoch = 1$ to $E_{\max}$ do
7:     Shuffle $\mathcal{D}_{\text{train}}$
8:     for each batch $(x, y)$ in $\mathcal{D}_{\text{train}}$ do
9:         $y_{\text{pred}} \leftarrow M(x; \theta)$
10:        $loss \leftarrow \mathcal{L}(y_{\text{pred}}, y)$
11:        Update $\theta$ using $O$ to minimize $loss$
12:    end for
13:    $val\_loss \leftarrow 0$
14:    for each batch $(x_{val}, y_{val})$ in $\mathcal{D}_{\text{val}}$ do
15:        $y_{val}^{\text{pred}} \leftarrow M(x_{val}; \theta)$
16:        $val\_loss \leftarrow val\_loss + \mathcal{L}(y_{val}^{\text{pred}}, y_{val})$
17:    end for
18:    $val\_loss \leftarrow val\_loss / |\mathcal{D}_{\text{val}}|$
19:    if $val\_loss < best\_val\_loss$ then
20:        $best\_val\_loss \leftarrow val\_loss$
21:        $best\_weights \leftarrow \theta$
22:        $patience\_counter \leftarrow 0$
23:    else
24:        $patience\_counter \leftarrow patience\_counter + 1$
25:        if $patience\_counter \geq p$ then
26:            break
27:        end if
28:    end if
29: end for
30: Return model $M$ with weights $best\_weights$
```

To test the models performance on unseen data, a four fold cross validation is used for certain hyperparameters.

5 | Data Analysis and Feature Engineering Results

In this chapter the data for the two tasks outlined in section 1.2 are analyzed in order to determine whether failure is detectable and predictable. The analysis will also lay a basis for feature selection and feature engineering of the datasets, resulting in a feature subset well suited for prediction, while being highly interpretable. To aid in this feature selection, the algorithm described in section 4.1.2 is used. Ultimately, this chapter aims to answer the secondary inquiry *"how can feature engineering be used to create a feature space suited for prediction models, while giving insight into the system failures?"*.

5.1 Data analysis for accumulator failure data on the Porsorja Express

The dataset from the line recorder files for the Porsorja Express contains 1719 parameters, some of these are measured for multiple cylinders, or other components that there are multiples of, this results in a total of 3392 features. In the following section the best features for classifying the difference between engine stability and accumulator failure are found. Since the dataset contains a high dimensional feature space with high correlations and irrelevant features useful to look at, feature selection will be used to eliminate these. The goal is to obtain a reduced feature subset for the prediction models.

5.1.1 Accumulator failure labeling

In order to label incident times of accumulator failure, an algorithm developed in previous work will be used[64]. The algorithm was developed using a ratio between the engine load and swash plate position while looking for an additional L_{WIVA} feedback error warning. The algorithm developed in the previous work [64] can be seen in algorithm 12.

Algorithm 12 Algorithm which detects accumulator bursts based on the swash plate position, engine load, and L_{WIVA} feedback error.

```
1: inputs:
       $engine\,load, swash\,plate\,position$
2: initialize:
       $threshold \leftarrow descision\,boundary$
3: $N \leftarrow$ length of $engine\,load$
4: for $i = 1$ to $N$ do
5:     $r \leftarrow \frac{engine\,load[i]}{swash\,plate\,position[i]}$
6:     if $(r \geq threshold)$ AND $(engine\,load[i] > 1)$ AND ($L_{WIVA}$ feedback error) then
7:         An accumulator has burst
8:     end if
9: end for
```

The best threshold found in the precious work [64] is 1.96. The algorithm looks for cases where the ratio exceeds the threshold, while the engine load is above 1 and when a feedback error warning occurs. If all conditions are satisfied, then, with high certainty, an accumulator has failed.

Algorithm 12 is used on the linerecorder files from the Porsorja Express to find instances of accumulator failure. A linerecorder file continuously spans several days and measures all features with a frequency of 1 Hz. The two linerecorder files in the dataset contains data from 08/02-2024 to 22/03-2024 and 10/04-2024 to 08/05-2024. By applying the algorithm to the linerecorder files the incidents in table 5.1 were found at the corresponding times.

Entry	Description	File
[97473:97477]	New failure	File 1
[486906:488000]	Failure on cyl 3	File 1
[519140:519150]	Failure on cyl 4	File 1
[527344]	Failure on cyl 4	File 1
[794348:794353]	Failure on cyl 6	File 1
[912551:912556]	Failure on cyl 8	File 1
[1078984:1078989]	Failure on cyl 1	File 1
[1155595:1155601]	Failure on cyl 7	File 1
[1691149:1691153]	Failure on cyl 5	File 1
[2030484:2030495]	Failure on cyl 2	File 1
[2030484:2030495]	Failure on cyl 2	File 1
[2030850:2030860]	Failure on cyl 2	File 1
[2964215:2964224]	Failure on cyl 4	File 1
[3215102:3215105]	Failure on cyl 3	File 1
[3217855:3217870]	Failure on cyl 3	File 1
[3285035:3285036]	Failure on cyl 6	File 1
[196704:196706]	Failure on cyl 5	File 2
[508707:508716]	Failure on cyl 4	File 2
[1416347:1416354]	Failure on cyl 8	File 2
[1619576:1619578]	Failure on cyl 5	File 2

Table 5.1: Indices all the incidents of accumulator failure in the two linerecorder files with cylinder number.

The entries in table 5.1 are the times found by the detection algorithm. Since the linerecorder files span more days than the dewesoft data obtained directly from Porsorja Express, new incidents are found, which is a good indication that the algorithm 12 works on linerecorder data.
These timestamps are used to know when exactly in the linerecorder files an accumulator burst has occurred. The failure found in file 1 are made into a list of intervals with a stable segment before and after each incident and with the incident segment extending 3000 samples before the actual burst to also capture the lead up to an incident. This also makes it so that the data is not heavily skewed towards stable data since this is a very small part of the total dataset. The intervals used for classification can be seen below.

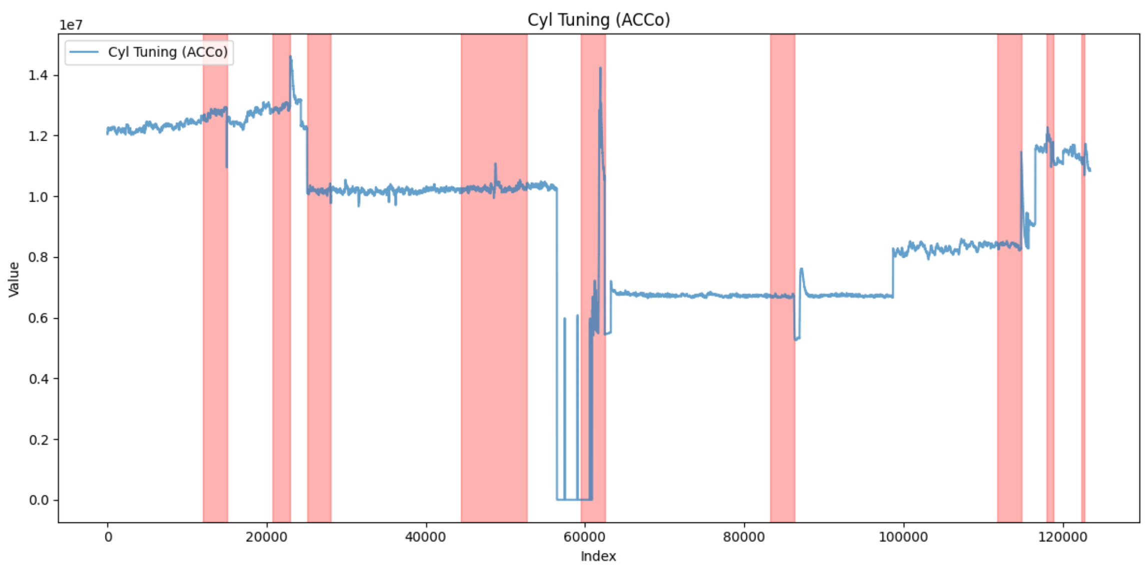


Figure 5.1: Graph of the Engine Load in the chosen intervals, the red indicates the interval leading up to an incident.

The intervals with stable conditions are found by a series of equation, from which MAN ES determines engine stability. These new intervals are then to classify when an incident is happening and when the engine is running as it should.

5.1.2 Feature-class correlation

To determine how closely correlated the features are to the failure, a binary classification is setup with failure and stable data. The feature-class correlation is computed by calculating the Pearson correlation coefficient, as described in section 3.2, between the feature samples and corresponding class labels. The highest correlated features to the class are listed in order of most correlated to least. The 30 features with the highest absolute correlation value can be seen in table 5.2.

Correlation	Feat. Name
-0.3210	Seconds since midnight (UTC)
-0.3210	Seconds since midnight (local)
0.2786	Cyl Tuning (ACCo)->Fuel Oil Pi Adjustment
0.2786	p(i) (ACCo), FO adjustment
0.2632	ECU->IO->CCUs Feedback->Pi average FO Adjustment
-0.2625	Liner Wall Monitoring->CCU1 LinerWall Temp.1 Dev.[Deg C]
-0.2625	Liner Wall Monitoring->CCU1 LinerWall Temp.1 Dev.[Deg C]
0.2621	ECU->IO->CCUs Feedback->Pi average FO Adjustment MOP
0.2621	Avg. p(i) (ACCo), FO adj.
0.2566	ECU->IO->CCUs Feedback->Pi average FO Adjustment
0.2556	ECU->IO->CCUs Feedback->Pi average FO Adjustment MOP
0.2520	p(i) (ACCo), FO adjustment
0.2520	Cyl Tuning (ACCo)->Fuel Oil Pi Adjustment
-0.2483	IO->AI (8615), Compressor Inlet Press
-0.2483	TC air intake pressure - ducted inlet, PT 8615
-0.2473	Cyl Tuning (ACCo)->Cylinder Prise Tuning->Regulator Out Adjustment
-0.2473	p(rise) adj. (ACCo)
0.2457	Current msec adjustment (ACCo)
0.2457	Cyl Tuning (ACCo)->Current Msec Adjustment
0.2453	Liner Wall Monitoring->CCU1 LinerWall Temp.2 Dev.[Deg C]
0.2452	Liner Wall Monitoring->CCU1 LinerWall Temp.2 Dev.[Deg C]
-0.2436	Air temperature before cyl. (fire box), TE 8610
-0.2436	IO->AI (8610), Scav.Air Fire Alarm
0.2388	Autotuning V2->SF Avg Adjustment
0.2388	ECU->IO->CCUs Feedback->Msec average Adjustment
0.2385	Autotuning V2->FO Avg Adjustment
-0.2383	p(rise) adj. (ACCo)
-0.2383	Cyl Tuning (ACCo)->Cylinder Prise Tuning->Regulator Out Adjustment
0.2359	Autotuning V2->SF Avg Adjustment
0.2359	ECU->IO->CCUs Feedback->Msec average Adjustment

Table 5.2: Top 30 Features with the Highest Correlation to Incident Class

The results of the feature-class correlation are generally low and it is suspected this might be due to non-linear dependencies. For this reason other methods of determining feature-class relation are pursued.

Random forest is a classic classification method, with built in feature selection and importance as described in section 3.4.2, and is therefore suitable in this case. In order to determine feature-class relations in a non-linear manner a random forest is used. This is done because a random forest can handle nonlinearities, whereas Pearson's correlation coefficient cannot. To help the model understand past behavior and capture trends in each feature, lagged data is used in the random forest model. When choosing the number of lags to use, it is important not to use too few lags as this may lead to the model missing important information. However, using too many lags risks overfitting the model, while also being computational heavy to calculate a lot of lags. Since the purpose of this random forest is to features important to classifying a failure occurrence, it is not crucial to find far back trends in the dataset at this point. Therefore, it has been chosen to use 90 lags in the model. The feature importance of the resulting random forest can be seen in figure 5.2.

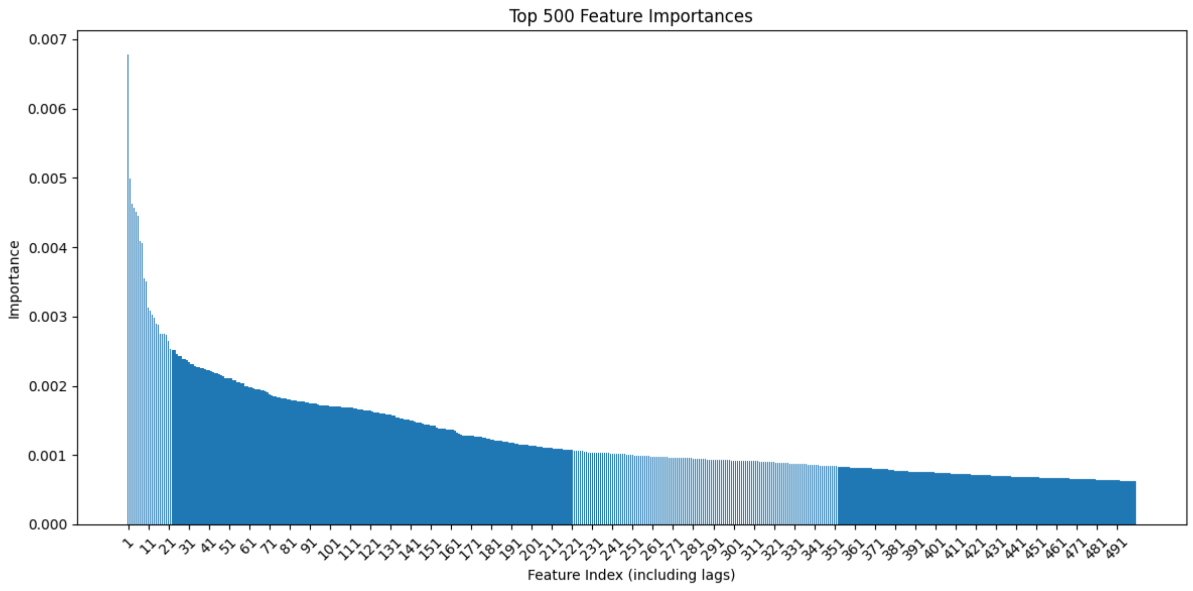


Figure 5.2: Shows a graph the importance level of 500 most important features.

The results in figure 5.2 shows the 500 most important features. Looking at the graph it can be seen that it has an exponential shape. Therefore, as you move to the right the importance of each additional feature decreases quickly. This indicates that is only necessary to select features from the highest performing group. For now, by looking at the graph, it has been chosen to focus on the 25 most important features, which are can be found in table 5.3.

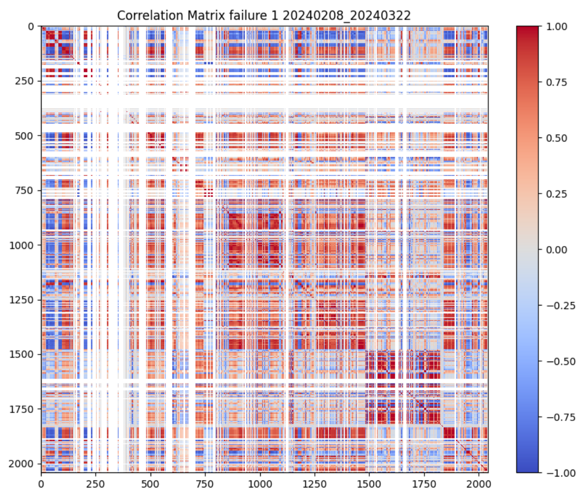
Feat.	Org. Feat.	Lag	Feat. Name
56305	1104	1	IO->AI (8408), Jacket CW Outlet pr. Cyl
10405	204	1	Exhaust valve position
5635	110	25	PSP Safety Margin Min Action nr. 3
21115	414	1	Hydraulic Pressure LP Filter->Filtered Pressure [bar]
23995	470	25	Cyl Tuning (ACCo)->Pcomp Measured
75287	1476	11	Ordered CCU msec
11832	232	0	Tacho->Max filtered speed
26316	516	0	Pmax Bearing Limit
39372	772	0	Cyl Tuning (ACCo)->Pc/Psc Measured
24187	474	13	Cyl Tuning (ACCo)->Pcomp Measured
19405	380	25	luboil_cyl_in_feedrate
24797	486	11	Cyl Tuning (ACCo)->Prise Measured
15606	306	0	Engine Tuning (ACCo)->PMI Telegram Recv
31645	620	25	TC LO inlet pressure, PT 8103
16345	320	25	Exhaust Valve Supervision->Close Supervision
44687	876	11	Ordered Pcomp/Pscav Ratio
56891	1115	26	IO->AI (8410), Cylinder CW Outlet
30295	594	1	p(max) (PMI)
54595	1070	25	IO->AI (8414), CW jacket outlet temp meas
27832	545	37	IO->AI (8421), CW_InletScavAirPress [bar]
1225	24	1	Exh. Valve Close Request
1045	20	25	Fuel Injection Sync. Angle
22645	444	1	CylTuning(ACCo)->CylPriseTuning
20196	396	0	Estimated effective engine power
39475	774	1	Cyl Tuning (ACCo)->Pc/Psc Measured

Table 5.3: UpdatedFeatureData

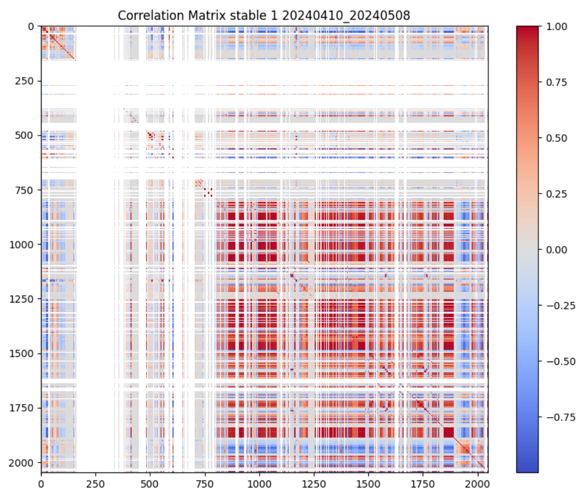
The resulting features in table 5.3 suggests that lagged data is used for failure classification. For this reason, longer lags are pursued for failure prediction with the reduced dataset in chapter 6.

5.1.3 Feature-feature correlation

As already established, a lot of features are highly correlated to each other. Therefore, in order to reduce dimensionality and remove spurious correlations, redundant features should be eliminated. In order to determine which features are highly correlated a correlation matrix of the feature space is computed and displayed. First, two correlation matrices are found, one where an failure occurs and one during stable engine operation. From this is can be seen how correlations look in each scenario. The correlation matrices are seen in figure 5.3.



(a) Feature-Feature correlation matrix for all relevant features over a failure segment.



(b) Feature-Feature correlation matrix for all relevant features over a stable segment.

Figure 5.3: Different Feature-Feature correlation matrices that shows the difference in behavior between system failure and stable engine conditions.

The first thing to notice from figure 5.3 is that features correlate more with each other during failure. This, however, makes sense since many features have somewhat the same behavior during a burst of an accumulator, some might spike at the same time and some might turn off. It can also be seen in both correlation matrices that features in the lower right corner are more correlated to each other than other features. This can be explained by looking further into which features lies here. Many features in that area are the same measurement measured on different cylinders. For instance, different temperature measurements for each cylinder lie in this area of the correlation matrix. Logically, these features relate to each other. A correlation matrix for the full dataset can be seen in figure 5.4.

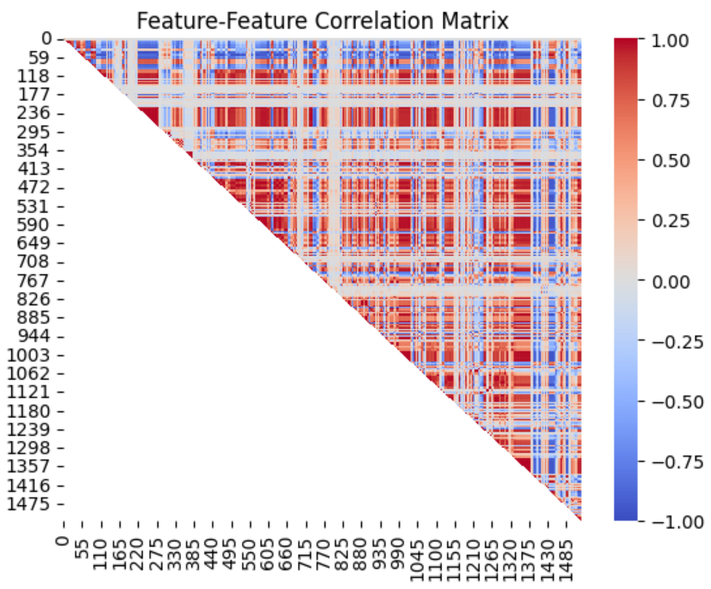


Figure 5.4: Feature-Feature correlation matrix for 1530 features.

It is clear from the presented correlation matrices, that significant feature reduction should be used in order to eliminate potential spurious correlations. This is done in section 5.1.4.

5.1.4 Feature selection

In order to reduce the 3392 features, to remove highly correlated features, and features unrelated to classifying failure, algorithm 9 is used. The algorithm will use the correlation matrix seen in figure 5.4 as C_{ff}, and the importances from the random forest in section 5.1.2 are used as I_{cf}. The feature-feature correlation threshold is set as 0.7, as this is generally considered the threshold for high correlation [65][66][67]. Using algorithm 9 with these parameters will ensure a feature subset where features are uncorrelated to each other and have a high correlation to the class, as described in CFS. Applying algorithm 9 results in the correlation matrix, with 51 features, seen in figure 5.5.

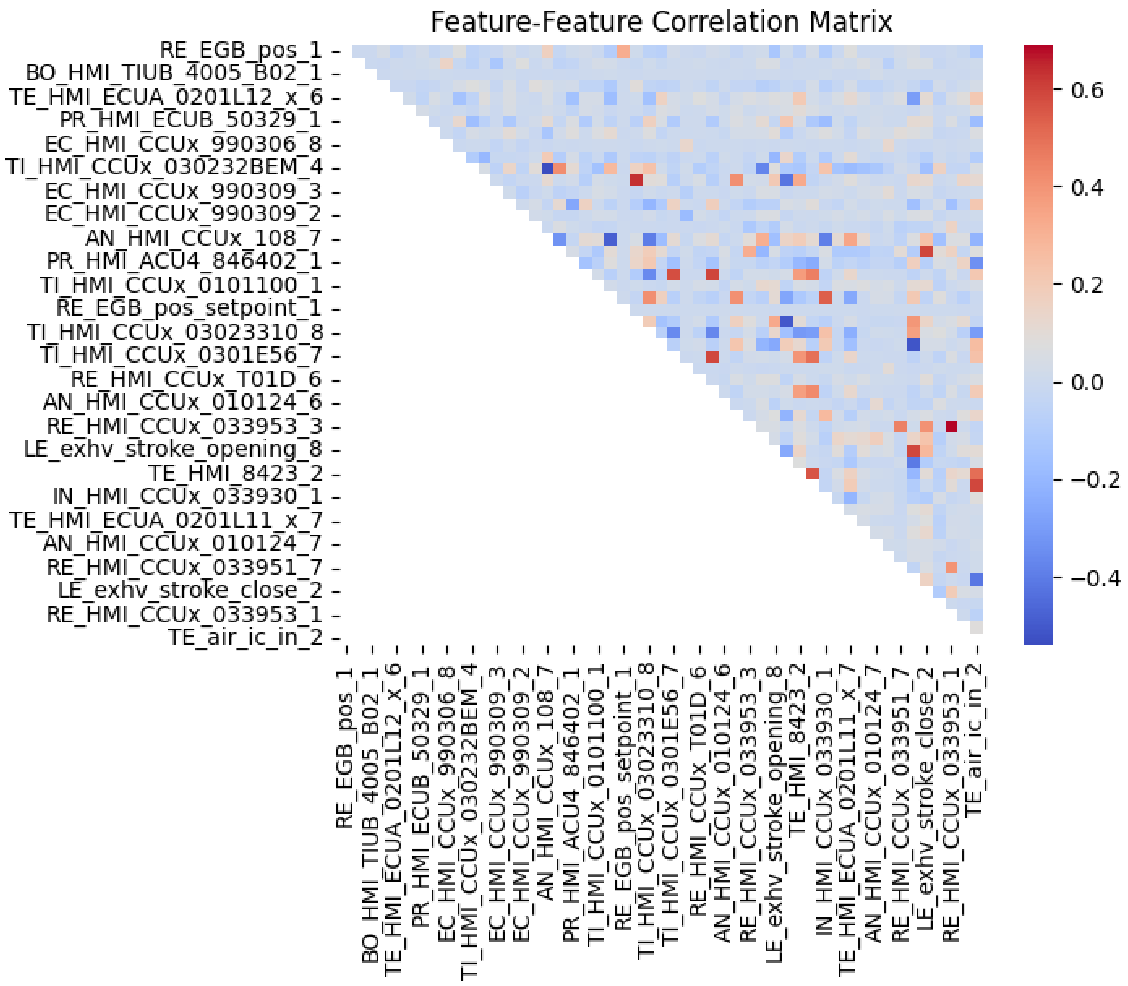


Figure 5.5: Feature-Feature correlation matrix the reduced feature subset.

Figure 5.5 shows the 51 selected features from algorithm 9. Furthermore, it can be seen that feature-feature correlations have been significantly reduced with only a few nearing the correlation threshold of 0.7. In terms of using these features in prediction models the feature subset has to be expanded. Many of the features are features which are measured for all cylinders. As accumulators can fail on all cylinders, measurements on all cylinders need to be used. Adding the additional features results on a feature subset containing 209 total features, based on the results seen in figure 5.5.

Looking at the features selected in this section, they primarily relate to four things, exhaust temperature, air intake pressure, liner wall temperature, and Adaptive Cylinder Control (ACCo). These results are logical, as poor injection performance negatively impacts cylinder health as seen by the temperature. The fact that the air intake pressure is higher correlates well with the increase in exhaust temperature, since a turbo charger uses the energy from the exhaust gasses to generate intake pressure. The ACCo is a mechanism that adjusts various engine parameters to adapt to changes and trying to maintain a desired combustion. Changes in these adaptive parameters could indicate that the control system finds deviations from normal conditions and tries to correct this.

5.2 Data analysis for methanol change over measurements

The data collected from the methanol changeover is measured in tests. For each test it is known if the changeover procedure fails, and when during the process it fails. In order to determine whether prediction of changeover failure is feasible, a simple classification of the tests is conducted. Each test is split into injection cycles in order to capture only relevant data, as in between injections, most features have very little variance. Each injection cycle is labeled according to the test, a summary of the labeled dataset used for classification can be seen in table 5.4. Note that a failure is labeled as '1' and stable as '0'.

Test	T007	T008	T010	T11	T020	T030	T035	T039	T044
Sample size	101	18	11	17	32	59	328	63	68
Label	1	1	1	1	1	0	0	0	0
Engine load	25%	25%	40%	25%	50%	25%	57%	60%	75%

Table 5.4: Summary of changeover data used for preliminary analysis.

In order to develop a failure prediction model, this data is analyzed to gain a better understanding of what differentiates system failure from stable conditions. This includes determining whether failure is classifiable, analyzing which features are related to failure occurrence, selecting an appropriate feature subset, and determining if a failure is predictable.

5.2.1 Failure classification

Given that the tests in table 5.4 are conducted at different engine loads, the injection cycles are not the same length, as injection is based on the crankshaft angle and not time. For this reason DTW-nn is used for classification. This is also a non-parametric learning algorithm, meaning it is simply measured how closely features align within each class. DTW-nn is conducted for each individual feature in the dataset, in order to determine which features relate to the occurrence of failure. Each feature is evaluated through the classification accuracy and F1 score. A high accuracy for a feature indicates a strong relation to the occurrence of failure, and may therefore be useful for failure prediction.

Early experiments with the DTW-nn algorithm yielded an average accuracy of 100% for four fold cross validation. It was found that this was due to tests being mixed with injections within a test are very similar, and the algorithm would always assign to the same test. To avoid this, training and testsets were redefined to ensure generalization. The testset was redefined as containing all injections from one failure test, and one non-failure test. The trainingset is then the remainder of the tests. The algorithm was run for each possible combination of testsets, resulting in a total of 16 test folds. Furthermore, the dataset was balanced such that 18 injections, or data points, were randomly selected from each test, except for test T010 and T011 which were combined with 9 random samples from each. The resampled dataset can be seen in table 5.5.

Test	T007	T008	T010 & T011	T020	T030	T035	T039	T044
Sample size	18	18	18	18	18	18	18	18
Label	1	1	1	1	0	0	0	0
Engine load	25%	25%	25% &40%	25%	50%	25%	60%	75%

Table 5.5: Summary of data used for DTW-nn classification of methanol injections during changeover.

The results of the 16 test folds can be seen in appendix I.1. The accuracy and F1-score is calculated for each fold, and summarized by the average across folds. The top performing features, and their average accuracy and F1 score can be found in table 5.6.

Feature name	**Avg. Accuracy**	**Avg. F1-score**
P886-8 [PM886-8.FOR]	0.95	0.96
PExhBnd8 [PM8702-8]	0.95	0.93
SG3Aft	0.88	0.75
LFBIV_Plunger8_Man_mm	0.87	0.91
SG2Aft	0.86	0.74
PGasCh8 [PT6409-8]	0.82	0.85
LFBIV8Man_mm	0.81	0.85
LFuelV8Aft	0.79	0.81
P886-8 [PM886-8. AFT]	0.78	0.72
LFBIV_Plunger8_Aft_mm	0.78	0.72
PCylDAU8 [PT1422-8]	0.77	0.72
LFBIV_Plunger8_Aft	0.77	0.72
SG1Man	0.77	0.61
LELFI8_mm	0.77	0.73
CELFI8_mm	0.77	0.73
P886-8 [PM886-8.MAN]	0.76	0.70
LFuelV8Man	0.76	0.78
LFuelP8	0.75	0.76
PScav8 [PM8601-8]	0.74	0.69
LExhValveFor8 [ZT4111-8.FOR]	0.74	0.73
PCylDAU6 [PT1422-6]	0.74	0.68
PCyl8 [PM1422-8.1]	0.74	0.68
LFBIV8 [ZM-8.MAN]	0.73	0.67
LFBIV8Man_filter	0.73	0.67
LELFI-L8 [ZT6424-8]	0.72	0.65
CELFI8	0.71	0.70
SG1Aft	0.71	0.66
CELFI-L8 [XC6429-8]	0.71	0.64
PCylDAU5 [PT1422-5]	0.71	0.63
LELFI-L8_mm	0.71	0.63
PLubHole8 [PM1422-8.4]	0.70	0.63
PCylDAU1 [PT1422-1]	0.70	0.74
LFBIV_Plunger8_Man	0.69	0.75
PGasSupply	0.69	0.75

Table 5.6: Summary of results of DTW-nn classification for high performing features.

5.2.2 Feature correlations for changeover

The results from the DTW-nn algorithm have shown that several features are well suited for failure classification. However, the current feature set might contain redundancies in the form of highly correlated features. As mentioned in section 2.2, this can cause issues when training regression models or AI networks. To reduce redundancy, the correlation between the features are summarized in the correlation matrix seen in figure 5.6 for high performing features. A correlation matrix for the full feature space can be seen in appendix I.2.

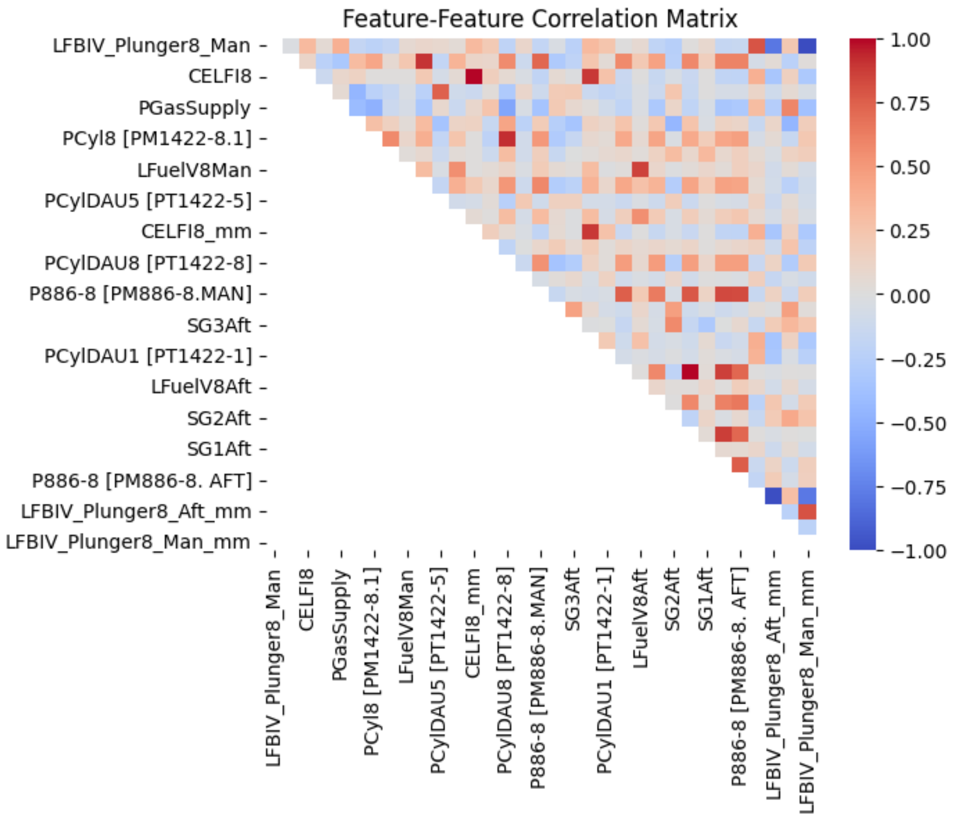


Figure 5.6: Correlation matrix over feature-feature correlations, for the high performing features seen in table 5.6

Some of the correlation pairs such as *CELFI8* and *CELFI8_ mm* have a correlation coefficient of 1, as they are the same signal, just one has been converted from a voltage to a mm measurement. Other cases of high correlation pairs include *CELFI-L8* and *LELFI-L8*, where C indicates a control signal and L a feedback signal. The control and feedback signal are naturally highly correlated. A full list of highly correlated features, with a numerical correlation coefficient above 0.7, can be seen in appendix I.3.

5.2.3 Feature selection for the changeover problem

In order to eliminate redundancies from high feature-feature correlations, the algorithm described in section 4.1.2 is applied on the feature subset seen in table 5.6, as these features perform well in terms of classifying system failure. The algorithm is used with DTW-nn accuracy as the importance measure I_{cf}, and with the feature-feature correlation matrix in figure 5.6 as C_{ff} again with a threshold of $\varepsilon = 0.7$. The algorithm yields the feature subset and corresponding feature-feature correlation matrix seen in figure 5.7.

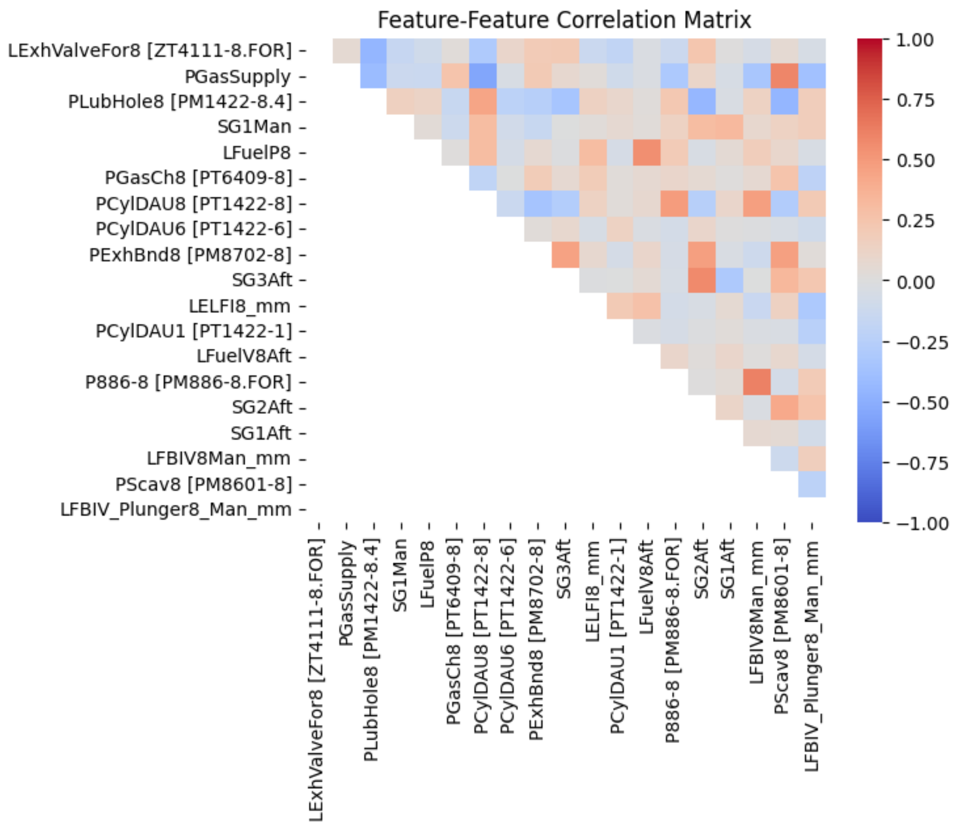
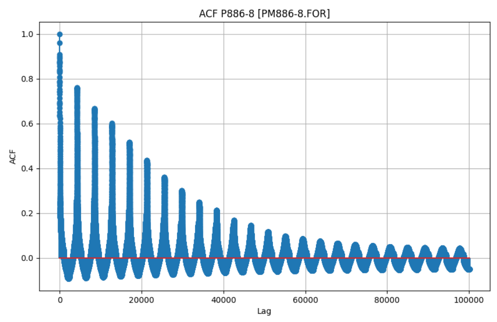


Figure 5.7: Feture-feature correlation matrix of feature subset determined by algorithm 9.

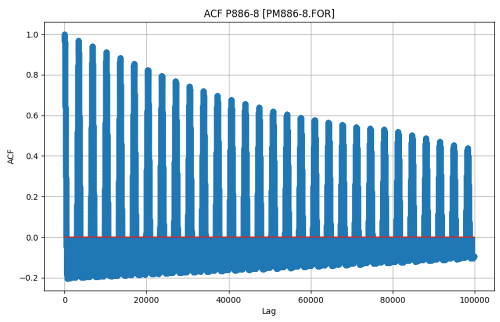
5.2.4 Failure predictability based on selected features

In order to determine whether it is possible to predict a failure auto-correlation is used, as it allows to check for time dependencies in the features. In order to predict a failure beforehand the features should demonstrate time dependency.

The auto-correlation is computed for one minute of data, starting from a changeover. The data is segmented into injection cycles and concatenated. In this way, redundant information in between injections is removed. This helps with computation and provides a clearer picture of relevant feature information. For the algorithmically selected features summarized in figure 5.7, most features show a time dependency and share a similar pattern to the ACF for *P886-8 FOR* seen in figure 5.8.



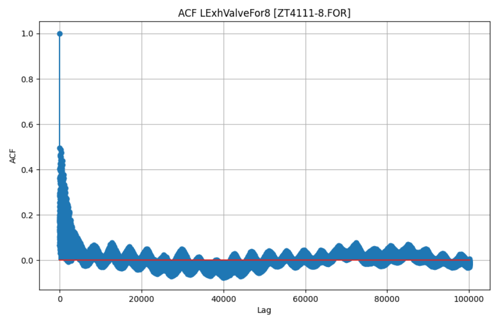
(a) Acf of *P886-8 FOR* for test 13, which is a failure.



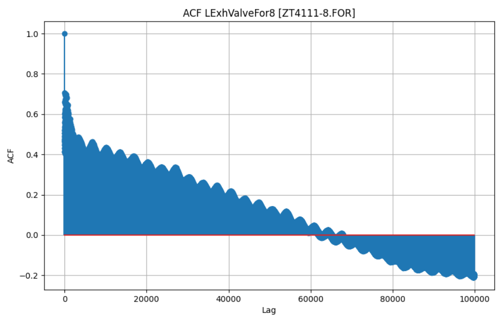
(b) Acf of *P886-8 FOR* for test 31, which is a non-failure.

Figure 5.8: Acf for *P886-8 FOR* for a failure and non-failure test.

The spikes in correlation correspond with injection cycles. Meaning that for the lags where injection cycles line up, there is a high degree of correlation. This indicates that each injection cycle is dependent on the previous one, and thereby time dependent. The only feature that does not follow this pattern is *LExhValveFor8*, as seen in figure 5.9. It is suspected that this is because the exhaust valve react to bad injections when the system is already failing.



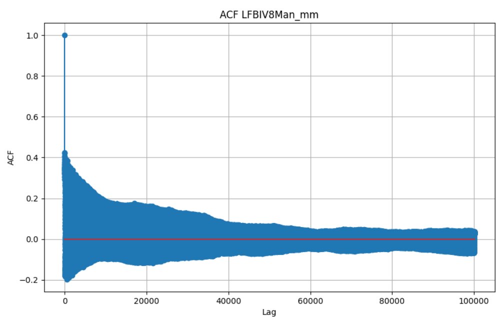
(a) Acf of *LExhValveFor8* for test 13, which is a failure.



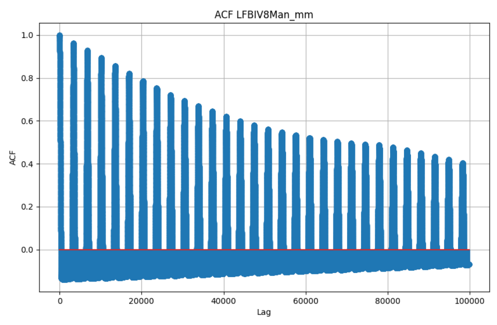
(b) Acf of *LExhValveFor8* for test 31, which is a non-failure.

Figure 5.9: Acf for *LExhValveFor8* for a failure and non-failure test.

Lastly, *LFBIV8_mm* has cases where the auto-correlation is low, as seen in figure 5.10. These cases have been investigated, and has been linked to instances where the sensor fails, and becomes extremely noisy.



(a) Acf of *LFBIV8Man_ mm* for test 13.



(b) Acf of *LFBIV8Man_ mm* for test 31.

Figure 5.10: Acf for *LFBIV8Man_ mm* for test 13 and 31.

Based on the auto-correlation results, feature *LExhValveFor8* is removed from the feature space, due to its lack of time dependency. Furthermore, the noisy measurements of *LFBIV8Man_ mm* should be considered when training a prediction model. The acf results also show that the coefficients generally fall off quicker for failure tests, compared to non-failure tests. This is also an indicator that failure is predictable, as the time dependencies do not hold leading up to system failures.

5.2.5 Feature engineering for the changeover problem

The results in sections 5.2.3 and 5.2.4 yield a significantly reduced feature subset, reducing it from 68 to 17 features, based on their relation to failure occurrences, as well as there time dependencies. The feature subset is summarized in table 5.7.

Features
LELFI8_mm
LExhValveFor8 [ZT4111-8.FOR]
LFBIV_Plunger8_Man_mm
LFBIV8Man_mm
LFuelP8
LFuelV8Aft
P886-8 [PM886-8.FOR]
PCylDAU1 [PT1422-1]
PCylDAU6 [PT1422-6]
PCylDAU8 [PT1422-8]
PExhBnd8 [PM8702-8]
PGasCh8 [PT6409-8]
PGasSupply
PLubHole8 [PM1422-8.4]
PScav8 [PM8601-8]
SG1Aft
SG1Man
SG2Aft
SG3Aft

Table 5.7: Selected feature subset for mehtanol changeover failure prediction.

The features are sampled with 20 kHz, resulting in long sequences, which will cause issues when training. For this reason, the temporal dimensionality should also be reduced. As seen from the autocorrelation, the time dependencies occur from injection to injection. Therefore, summarizing each injection cycle will help reduce temporal dimensionality, while maintaining temporal dependencies. Inspecting the autocorrelation results reveal that the high correlation occur when the peaks of the injections meet. Furthermore, the spiky pattern of the injections are well suited for using maximum values. For these reasons the injection maximum for each feature is used. It is checked that all features have an injection maximum, and not a minimum. An example of injection maximum values can be seen for *P886-8 [PM886-8.FOR]* in figure 5.11 for a stable and failing changeover to methanol for tests T035 and T008 respectively.

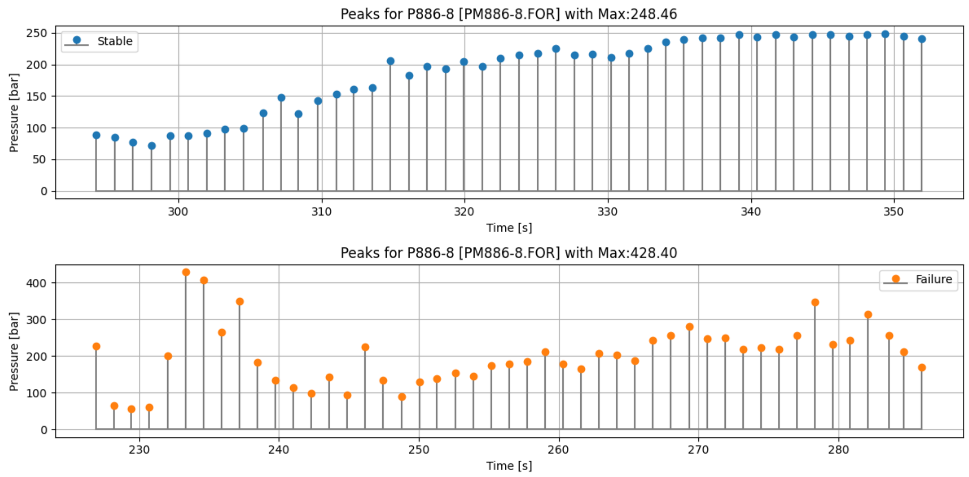
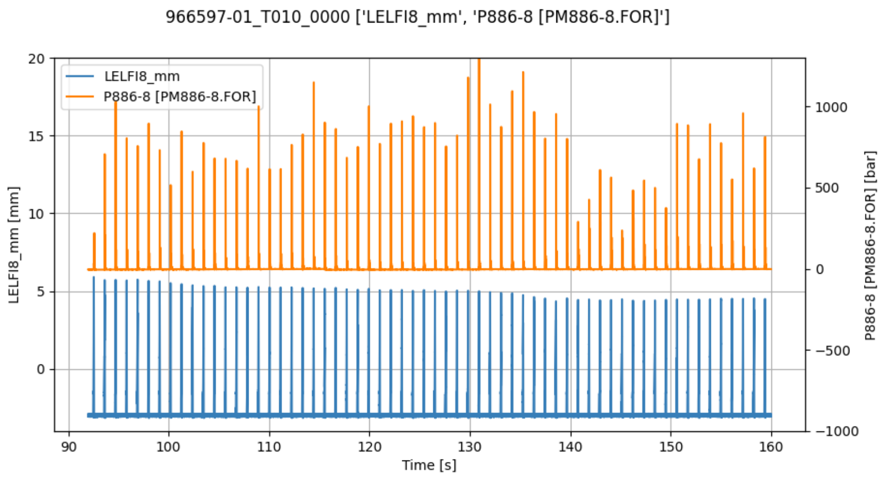
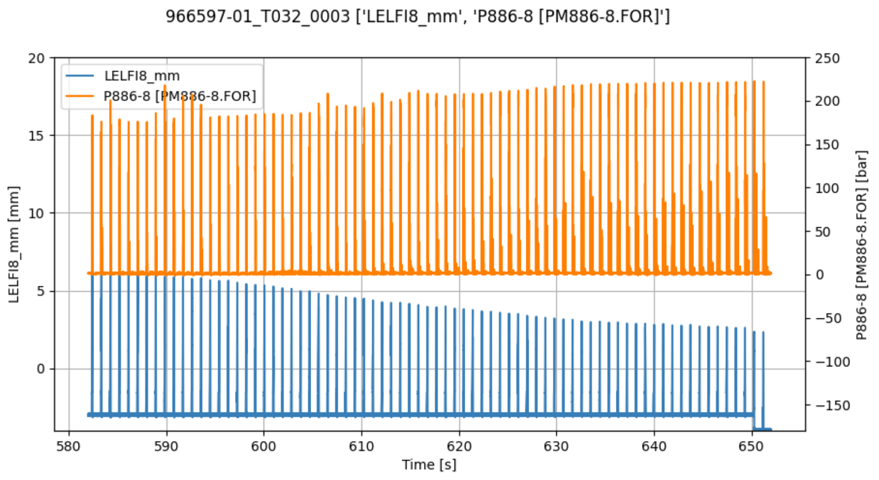


Figure 5.11: Stem plot of max values of *P886-8 [PM886-8.FOR]* per injection cycle for tests T035 and T008.

It can clearly be seen the temporal development of *P886-8 [PM886-8.FOR]*, where the stable changeover ramps up and stabalizes, whereas the failed changeover is more sporadic, and with higher pressure spikes. The selected features and maximum values for the feature subset have been presented to experts on the injection system. Their analysis of the results, and their expert knowledge leads to following conclusions; *LELFI8_mm*, which controls diesel injection, decreases as the methanol injections are ramped up. Therefore, *P886* should increase, to increase injection pressure. However, during a failure that *P886* do not follow *LELFI8_mm* as it is supposed to as seen in figure 5.12. Furthermore, large pressure spikes in *P886* occur.



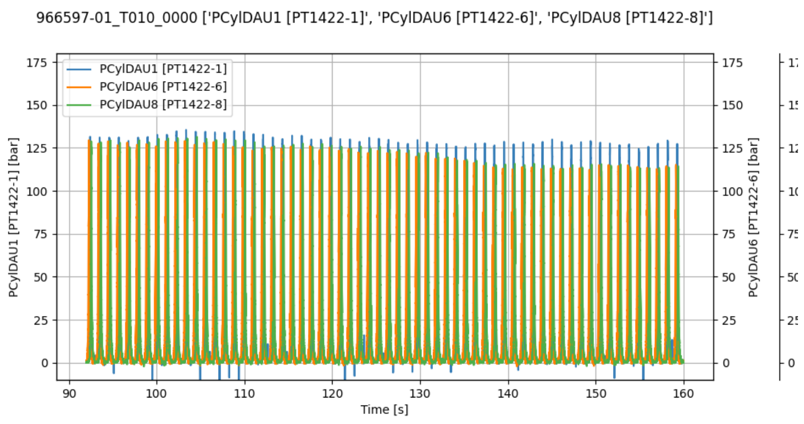
(a) Failed changeover in test T010.



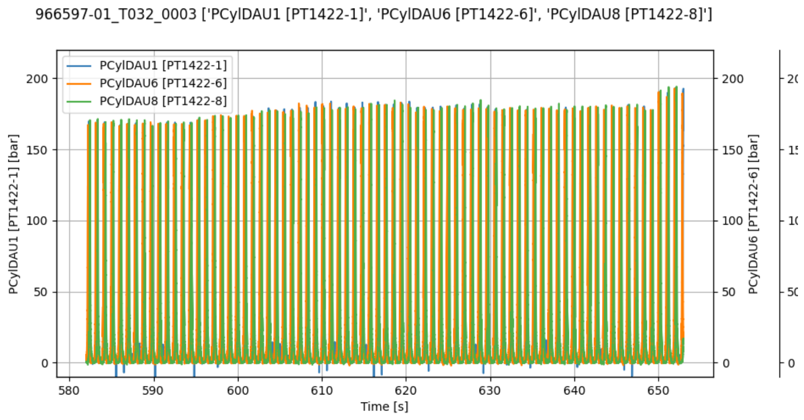
(b) Successful changeover in test T032.

Figure 5.12: $P886\ FOR$ and $LELFI8\_mm$, for a failed and successful changeover.

This results in poor injections, and therefore the cylinder pressures on later cylinders, such as cylinder $PCylDau6$ and $PCylDau8$, have decreased cylinder pressure compared to the setpoint, and that seen on $PCylDau1$ which can be seen in figure 5.13.



(a) Failed changeover in test T010.



(b) Successful changeover in test T032.

Figure 5.13: Cylinder pressure on cylinder 1, 6, and 8, for a failed and successful changeover.

The bad injections can also be seen by poor exhaust in $LExhValve8$, and a compensation in air inlet pressure $PScav8$. The knocking which occurs in the piping during a failing changeover is detected by the strain gauges $SG1Aft$, $SG1Man$, $SG2Aft$, and $SG3Aft$. Lastly, looking at the $FBIV$ measurements, it can be seen that the plunger seems to bounce during failure as seen in figure 5.14.

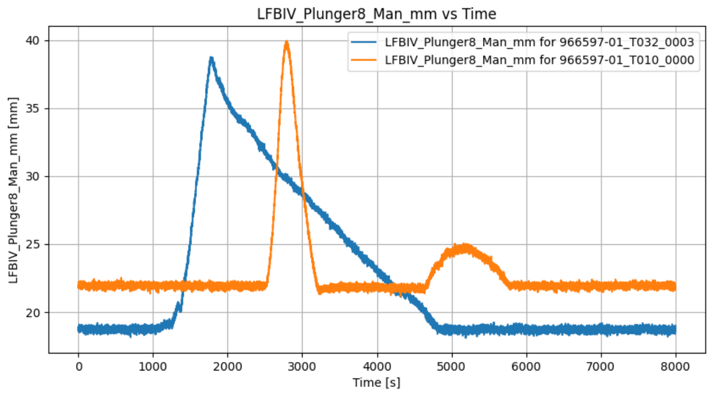


Figure 5.14: FBIV plunger position for failed changeover T010 seen in orange, and successful changeover T032 in blue.

It can also be seen from figure 5.14, that during failure, the FBIV plunger peaks at a higher max value. These conclusions strengthen the reason for using max values to reduce temporal dimensionality. Although, this comes at the cost of loosing the secondary, smaller, spike caused by the FBIV plunger bounce.

6 Prediction Model Results

In this chapter results for all prediction models developed are presented. All models are described by a their program assumptions, hyperparameters are summarized in a table, and the model is evaluated using a four fold cross validation. In some cases a preliminary hyperparameter sweep is done before the cross validation to avoid long training and validation times. The results from this chapter will answer the inquiry *"can ML and AI models be used to predict marine propulsion system failures?"*.

6.1 Results for accumulator failure prediction

In this section accumulator failure prediction models are evaluated, addressing the first prediction problem *"can a system failure be predicted through binary classification, by predicting if a failure happens within a Prediction Window, given data read from a Reading Window?"*. This includes random forest with lags, 1D CNN, and LSTM classifiers. Each model is evaluated for different prediction window and reading window sizes.

6.1.1 Random forest accumulator failure prediction results

The program assumptions for the random forest model used for accumulator failure prediction can be seen in table 6.1.

Component	**Details**
Model Type	Random Forest classifier with lags
Hyperparameters	Number of trees (n_estimators), reading window (RW)
Input Shape	(1, RW·209 features)
Model description	Random forest containing n_estimators trees that conduct majority vote
Output	Shape: (1, 1), Value: $out \in [0; 1]$
Impurity measure	Gini index

Table 6.1: Summary of the random forest model used for accumulator failure prediction.

An initial hyperparameter sweep was conducted by training on SMOTE data, and validating on real data. The failure class was increased five fold using SMOTE with 5 nearest neighbors. The stable samples were randomly split into training and validation in a 0.5/0.5 split, the samples were then randomly undersampled to balance the number of failure, stable, and lead up samples in the training set. The tested hyperparameters are summarized in table 6.2.

Parameters	**Values**						
RW stride	100	150					
n_estimators	50	100	150	200			
RW	600	900	1200	1500	1800		
PW	600	1200	1800	2400	3600	5400	7200

Table 6.2: Hyperparameters for random forest accumulator failure prediction.

The results from the sweep has multiple runs with an F1-score of 1.0, meaning no misclassification in validation, the results can be seen in appendix II. Due to these results, the model was tested on unseen data, using a four fold cross validation. As mentioned in section 4.1.1 the data comes from multiple segments of a full dataset. The folds are split into segments before windowing and using SMOTE. This ensures that SMOTE training data has not been generated on data used for validation. The hyperparameters are reduced to include a PW of 600, a RW of 600, and 100 estimators to see how well the cross validation performs compared to the initial run which had an F1-score of 1.0. The F1-score for failure and stable classes are seen in table 6.3 together with the fold accuracies.

Metrics	**Folds**				
	1	2	3	4	**Avg.**
F1 failure	0.0	0.15	0.0	0.0	0.038
F1 stable	0.67	0.69	0.67	0.67	0.675
Accuracy	0.5	0.54	0.5	0.5	0.51

Table 6.3: Four fold cross validation results for random forest accumulator failure prediction.

When performing a four fold cross validation the model performance significantly decreases. There is only one fold where a failure is predicted, for all other folds the model only predicts 'stable', as can be seen in the confusion matrices found in appendix II.1. This indicates that the model does not perform well on unseen data.

6.1.2 1D CNN classifier accumulator failure prediction results

The program assumptions for the 1D CNN classifier used for accumulator failure prediction can be seen in table 6.22, with description of convolutional and fully connected layers.

Component	**Details**
Model Type	1D Convolutional Neural Network (CNN)
Hyperparameters	Learning rate, batch size (B), dropout, reading window (RW)
Input Shape	(B, RW, 209 features)
Convolutional Layers	
2 x 1D Conv layer + ReLU	Channels: 128, Kernel: 5, Padding: 2, Stride: 1
Max Pooling	Kernel: 2, Stride: 2
2 x 1D Conv layer + ReLU	Channels: 256, Kernel: 3, Padding: 1, Stride: 1
Adaptive mean pooling	Output size: (B, 1, 256)
Fully Connected Layers	FC: $256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 1$
Activation	ReLU activation in intermediate layers, sigmoid activation in final layer
Output	Shape: (B, 1, 1), Value: $out \in [0; 1]$
Loss Function	Binary Cross Entropy (BCE)
Optimizer	Adam

Table 6.4: Summary of the 1D CNN architecture used for accumulator failure prediction.

Given the results of the random forest classifier in section 6.1.1, a reduced hyperparameter set is used in combination with a four fold cross validation. A smaller and larger RW is used, combined with a smaller PW, to see if it is possible to predict accumulator failures. The network is tested with and without droput, and with a batch size of 16 which should be sufficient for generalization while avoiding computing large matrices. The hyperparamteters are summarized in table 6.5.

Hyperparameters	**Values**	
Batch size	16	
PW	600	
RW	600	1800
Dropout	0.0	0.3
Learning rate	0.0001	0.001

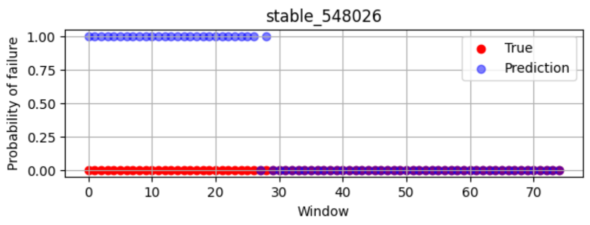
Table 6.5: Hyperparameters for 1D CNN used for accumulator failure prediction.

The model is trained and validated in a four fold cross validation, using SMOTE with 5 nearest neighbors to increase the failure data by five fold. The data is balanced with failure, stable and lead up data. The model was trained using early stopping with a max epochs of 20, and patience of 5. The result with the smallest validation loss, and the result with the largest accuracy and F1 scores for both classes are seen in table 6.6. Both runs have a RW of 600, and a learning rate of 0.001, the only difference is the dropout. The remaining results can be found in appendix II.2.

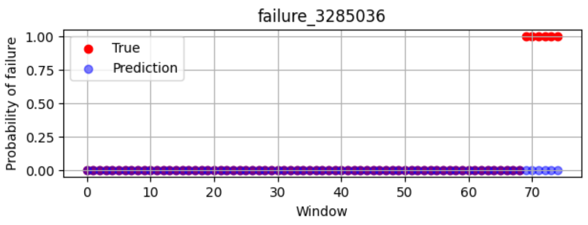
Parameters	**Metrics**			
Dropout	**Loss**	**Accuracy**	**F1 failure**	**F1 stable**
0.0	1.045	0.71	0.72	0.69
0.3	0.795	0.59	0.71	0.32

Table 6.6: Four fold cross validation results for 1D CNN accumulator failure prediction.

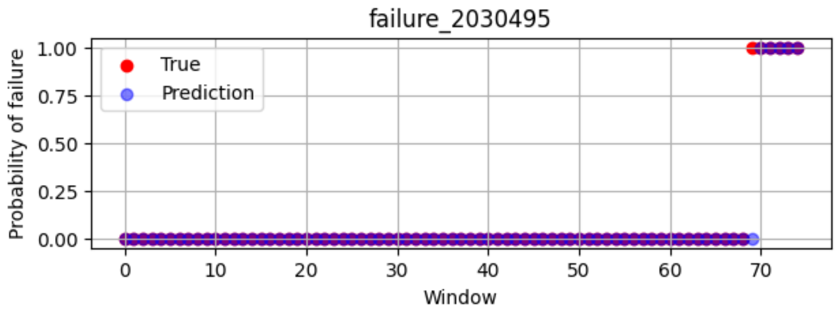
Despite the higher loss, the run with a dropout of 0.0 performs significantly better with regard to accuracy and F1 score. Example segments have been passed through the trained model in order to see how the model classifies them. All examples are from fold 4, and include a correctly predicted failure, an incorrectly predicted failure, and a misclassified stable segment. These examples can be seen in figure 6.1.



(a) Misclassified stable segment.



(b) Incorrectly predicted failure segment.



(c) Correctly predicted failure segment.

Figure 6.1: Example segments from accumulator failure data passed through the 1D CNN model.

6.1.3 LSTM classifier accumulator failure prediction results

The program assumptions for the LSTM classifier used for accumulator failure prediction can be seen in table 6.7, with description of LSTM blocks and fully connected layers.

Component	**Details**
Model Type	Long-short term memory
Hyperparameters	Learning rate, batch size (B), dropout, reading window (RW), attention
Input Shape	(B, 209 features, RW)
LSTM layers LSTM block Output	 Number of blocks: 2, Hidden size: 128 Last time step or attention block, shape: (B, 128, 1)
Fully Connected Layers Activation	FC: $128 \rightarrow 1$ Sigmoid
Output	Shape: (B, 1, 1), Value: $out \in [0; 1]$
Loss Function	Binary Cross Entropy (BCE)
Optimizer	Adam

Table 6.7: Summary of the LSTM architecture used for accumulator failure prediction.

Given the results from the 1D CNN, the same combination of hyperparameters were tested as for the validation run with the best accuracy and F1-score, and tested both with and without attention. The hyperparameters are summarized in table 6.8.

Hyperarameters	**Values**	
Batch size	16	
Learning rate	0.001	
Dropout	0.0	
RW	600	
PW	600	
Attention	True	False

Table 6.8: Hyperparameters for LSTM used for accumulator prediction.

The model is trained and validated in a four fold cross validation, using SMOTE with 5 nearest neighbors to increase the failure data by five fold. The data is balanced with failure, stable and lead up data. The model was trained using early stopping with a max epochs of 20, and patience of 5. The results for the four fold cross validation without attention can be seen in table 6.9, with performance measured by average loss, accuracy, F1-score for the failure and stable class, across folds. Remaining results can be found in appendix II.3.

Loss	**Accuracy**	**F1 failure**	**F1 stable**
0.693	0.56	0.36	0.66

Table 6.9: Four fold cross validation results for LSTM accumulator failure prediction.

The results show very poor performance, especially when trying to classify failure data. Due to the poor results, no further hyperparameters or model training and validation is pursued.

6.2 Results for methanol changeover failure prediction

In this section the results for methanol changeover failure prediction are presented in order to solve the first prediction problem *"can a system failure be predicted through binary classification, by predicting if a failure happens within a Prediction Window, given data read from a Reading Window?"*. This is done using random forest with lags and 1D CNN. Due to the relative short sequence lengths, and the performance of the other models, it is deemed unnecessary to pursue an LSTM model for this task. Both models are evaluated for different prediction windows and reading windows.

6.2.1 Random forest methanol changeover failure prediction

The program assumptions for the random forest model used for changeover failure prediction can be seen in table 6.10.

Component	**Details**
Model Type	Random Forest classifier with lags
Hyperparameters	Number of trees (n_estimators), reading window (RW)
Input Shape	(1, RW·19 features)
Model description	Random forest containing n_estimators trees that conduct majority vote
Output	Shape: (1, 1), Value: $out \in [0; 1]$
Impurity measure	Gini index

Table 6.10: Summary of the random forest model used for changeover failure prediction.

The hyperparameters and PW tested for the random forest classifier are summarized in table 6.11. This is done for a broad range of RW and PW, to test their impact. Early experiments showed that the number of estimators only performs poorly if there are "few" or "many". The default number for Sci-Kit Learns random forest is 100 estimators[68], therefore, 75 and 100 are tested, as a smaller number would simplify the model.

Parameters	**Values**					
n_estimators	75	100				
PW	8	32	64	78		
RW	1	2	4	8	16	32

Table 6.11: Hyperparameters for random forest changeover failure prediction.

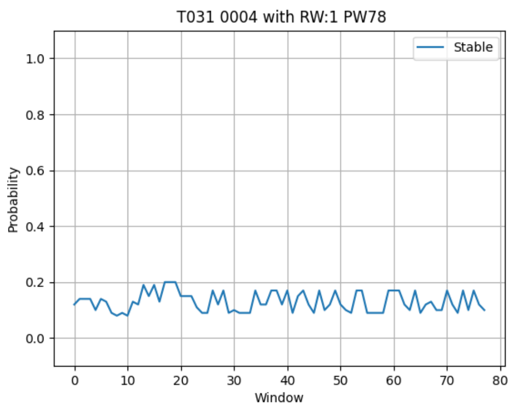
The hyperparameter performance is measured through training and validation. Failure, lead up and stable data was used to generate synthetic data using SMOTE by increasing the failure data five fold and balancing. The Random Forest was trained on synthetic data and validated on the original data. The results of the hyperparameter sweep can be found in appendix II.4. The hyperparameter sweep is evaluated by the log loss, and shows that a PW size of 78 performs the best, followed by a PW of 64. The reading windows all perform well for a high PW, in general a RW of 32 and 8 perform with lower loss for a PW of 78 and 64. A RW of 1 also performs well for a PW of 78. All combinations of these RW and PW have an F1-score of 1. The average difference in loss between 100 and 75 estimators is small, however, 100 outperforms slightly.

In terms of selecting model parameters, a PW of 78 is chosen, as this yields the best results, and allows for prediction furthest prior to failure. Although a RW of 32 performs best, smaller RW are desired to allow earlier prediction, and requires less data to be collected and stored in order to make a prediction. Therefore, reading windows of 32, 8, and 1 are tested in a secondary hyperparameter sweep. This secondary sweep will use a four fold cross validation split into tests. As mentioned in section 4.1.1 there are eight changeover tests which fail, and seven which succeed. These tests are split into four folds. The results of the sweep can be seen in table 6.12, where performance is measured by the F1-score for the failure and stable class respectively. The cross validation will also indicate the models performance and robustness to unseen data.

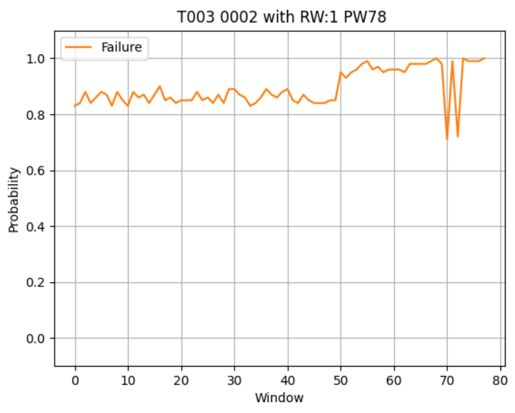
Parameters		**F1-scores per fold**				
PW	**RW**	1	2	3	4	**Avg.**
78	32	1.0/1.0	0.90/0.88	0.87/0.82	0.93/0.89	0.93/0.90
78	8	0.98/0.98	0.80/0.67	0.80/0.67	0.94/0.89	0.88/0.80
78	1	1.0/1.0	0.97/0.96	0.80/0.67	0.94/0.89	0.93/0.88

Table 6.12: Four fold cross validation results for random forest changeover failure prediction evaluated by the F1 for the failure and stable class respectively.

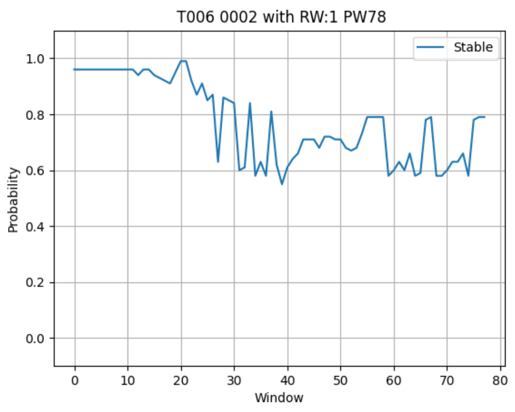
A large reason for the under performance of fold 2 and 3 is that the stable test T006 is in the validation fold. A plot of the predicted probability of failure can be seen in figure 6.2, alongside the predicted probability of test T010 which is an actual failure from fold 3, as well as a failed and stable changeover from fold 1 which is the best performing fold.



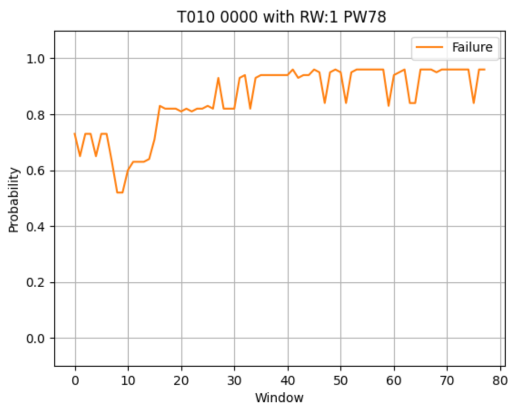
(a) Failure probability of successful changeover T031 from validation fold 1.



(b) Failure probability of failed changeover T003 from validation fold 1.



(c) Failure probability of successful changeover T006 from validation fold 3.



(d) Failure probability of failed changeover T010 from validation fold 3.

Figure 6.2: Probability of a failure happening within the PW for test T031, T003, T006, and T010. Orange indicates a failed changeover, and blue a successful changeover.

It can be seen in figure 6.3 that *P886* as noisy spikes, even though it can be seen that the changeover procedure stabilizes. It is likely due to these spikes that the changeover procedure is predicted as a failure.

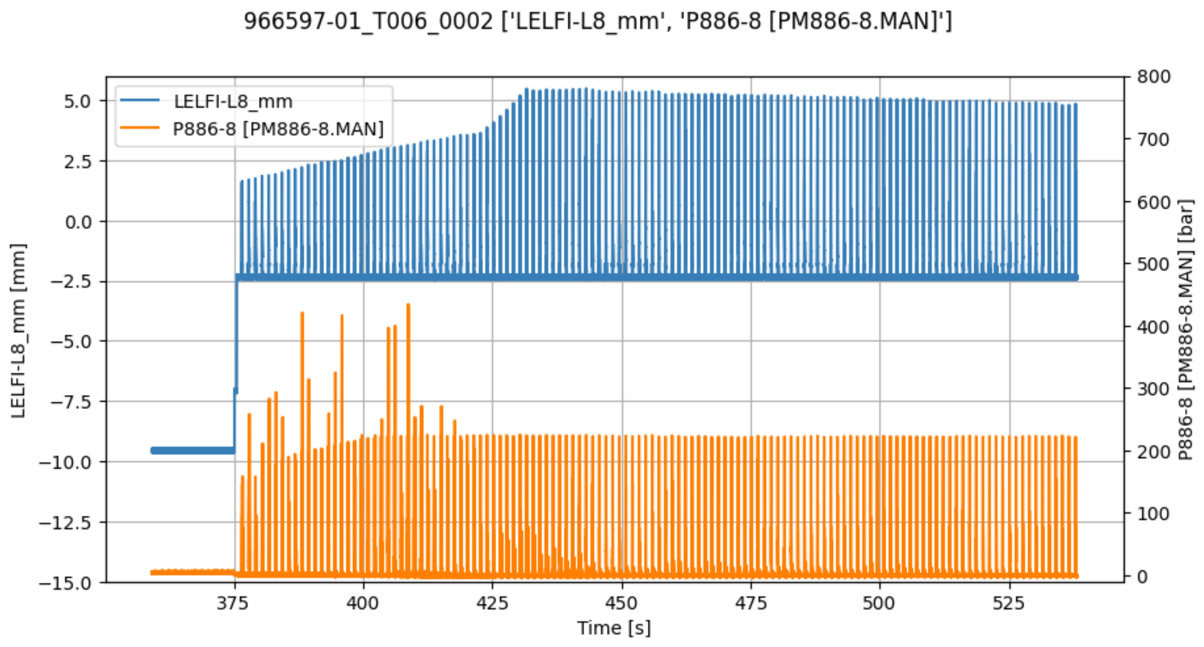


Figure 6.3: *P886-8 MAN* plotted alongside *LELFI-L8_ mm* for test T006.

The noisy spikes are suspected to be related to a secondary problem on the injection system. This problem relates to air being caught in the hydraulic fluid, which causes such pressure spikes in *P886*. This problem has already been resolved, by updating the piping. Therefore, this type miss-classification is unlikely to happen on data from an updated system.

6.2.2 Results for 1D CNN changeover failure prediction with prediction window

The program assumptions for the 1D CNN classifier used for changeover failure prediction can be seen in table 6.22, with description of convolutional and fully connected layers.

Component	**Details**
Model Type	1D Convolutional Neural Network (CNN)
Hyperparameters	Learning rate, batch size (B), dropout, reading window (RW)
Input Shape	(B, RW, 19 features)
Convolutional Layers	
1 x 1D Conv layer + ReLU	Channels: 64, Kernel: 3, Padding: 1, Stride: 1
1 x 1D Conv layer + ReLU	Channels: 128, Kernel: 3, Padding: 1, Stride: 1
Adaptive mean pooling	Output size: (B, 1, 128)
Fully Connected Layers	FC: $128 \to 1$
Activation	Sigmoid
Output	Shape: (B, 1, 1), Value: $out \in [0; 1]$
Loss Function	Binary Cross Entropy (BCE)
Optimizer	Adam

Table 6.13: Summary of the 1D CNN architecture used for changeover failure prediction.

As mentioned in section 4.2.2, the 1D CNN takes in RW, PW, learning rate, batch size, and dropout as hyperparameters. The Random Forest results indicate that larger PW perform well, therefore, only larger PW are tested for the 1D CNN. Furthermore, smaller RW of 1 and 2 do not make sense to test, as this leaves to few time steps to convolve over. The network is both tested with and without dropout. The chosen dropout is smaller, as the network is shallow. Lastly batch sizes of 16 and 32 are tested, to test the balance between generalization when learning, and reducing computation. The hyperparamters are summarized in table 6.14.

Parameters	**Values**		
Batch size	16	32	
Learning rate	0.0001	0.001	
Dropout	0.0	0.3	
PW	32	64	78
RW	4	8	16

Table 6.14: Hyperparameters for 1D CNN changeover failure prediction.

Augmented data was generated using SMOTE with 5 nearest neighbors to increase the data amount. The failure data was increased three fold and balanced with stable and lead up data. A four fold cross validation was performed, using early stopping with a maximum of 20 epochs, with a patience of 5. The performance is evaluated on the average loss, F1-score and accuracy. The best performing results can be seen in table 6.15 with hyperparameters RW=16, PW=64, batch size=32, learning rate=0.0001, and dropout=0.3. Results for the remaining hyperparameters are found in appendix II.5.

Loss	Accuracy	F1 failure	F1 stable
0.319	0.91	0.94	0.74

Table 6.15: Four fold cross validation results for 1D CNN changeovr failure prediction.

Again, it is seen that the F1-score for stable data is lower. The poorest performing fold is fold 3, containing test T006, which is noisy and therefore being misclassified as mentioned in section 6.2.1.

6.3 Changeover remaining useful life prediction

In this section the results for methanol changeover remaining useful life prediction are presented, answering the second prediction problem *"can the Remaining Useful Life of a system be predicted through regression, given data read from a Reading Window?"*. This includes regression models using random forest with lags, 1D CNN, and LSTM. All models are evaluated for different reading windows and hyperparameters.

6.3.1 Results for random forest changeover RUL prediction

The program assumptions for the random forest model used for changeover RUL prediction can be seen in table 6.16.

Component	**Details**
Model Type	Random Forest regressor with lags
Hyperparameters	Number of trees (n_estimators), reading window (RW)
Input Shape	(1, RW·19 features)
Model description	Random forest containing n_estimators trees that conduct majority vote
Output	Shape: (1, 1), Value: $out \in [0; 1]$
Impurity measure	Mean Square Error (MSE)

Table 6.16: Summary of the random forest model used for changeover RUL prediction.

The hyperparameters are tested for a broad range of RW and n_estimators as summarized in table 6.17.

Parameters	**Values**				
n_estimators	50	100	150		
RW	1	4	8	16	32

Table 6.17: Hyperparameters for random forest changeover RUL prediction.

The six failure sequences were used to generate 30 additional full sequences using DTW-SMOTE with 3 nearest neighbors. A four fold cross validation was performed on the

SMOTE and original samples in a hyperparameter sweep. The best performing hyperparameters are a RW of 32, and 150 estimators. The results can be seen in table 6.18, evaluated through the coefficient of determination R^2 for all folds, and the full results can be found in appendix II.6.

Fold				
1	2	3	4	**Avg.**
0.696	0.862	0.871	0.696	0.781

Table 6.18: Four fold cross validation results for random forest RUL prediction evaluated by the R^2 coefficient.

6.3.2 Results for 1D CNN changeover RUL prediction

The program assumptions for the 1D CNN regressor used for changeover RUL prediction can be seen in table 6.22, with description of convolutional and fully connected layers.

Component	**Details**
Model Type	1D Convolutional Neural Network (CNN)
Hyperparameters	Learning rate, batch size (B), dropout, reading window (RW)
Input Shape	(B, RW, 19 features)
Convolutional Layers	
2 x 1D Conv layer + ReLU	Channels: 64, Kernel: 3, Padding: 1, Stride: 1
Adaptive mean pooling	Output size: (B, 2, 128)
1 x 1D Conv layer + ReLU	Channels: 128, Kernel: 3, Padding: 1, Stride: 1
Adaptive mean pooling	Output size: (B, 1, 128)
Fully Connected Layers	FC: 128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 16 $\rightarrow$ 1
Activation	ReLU on all intermediate layers, none on final layer
Output	Shape: (B, 1, 1), Value: $out \in \mathbb{R}$
Loss Function	Mean Square Error (MSE)
Optimizer	Adam

Table 6.19: Summary of the 1D CNN architecture used for changeover RUL prediction.

The model is trained and validated in a four fold cross validation for the hyperparameters seen in table 6.20. Again, 30 augmented changeover sequences are generated using DTW-SMOTE with 3 nearest neighbors.

Parameters	**Values**			
Batch size	16			
Learning rate	0.0001	0.001		
Dropout	0.0	0.3		
RW	4	8	16	32

Table 6.20: Hyperparameters for 1D CNN changeover RUL prediction.

The best performing hyperparameters are a RW of 32, a learning rate of 0.0001, and with a dropout of 0.0. The results for of the four fold cross validation for these hyperparameters are found in table 6.21.

Fold				
1	2	3	4	**Avg.**
0.738	0.921	0.905	0.739	0.826

Table 6.21: Four fold cross validation results for 1D CNN changeover RUL prediction evaluated by the R^2 coefficient.

The 1D CNN outperforms the random forest regressor, with all folds having an R^2 coefficients above 0.7, with two folds having R^2 coefficients above 0.9. The tests used in the validation fold are passed through the model individually, to see how the model performs on unseen changeover sequences. The results for test T003 and T005 are plotted in figures 6.4 and 6.5.

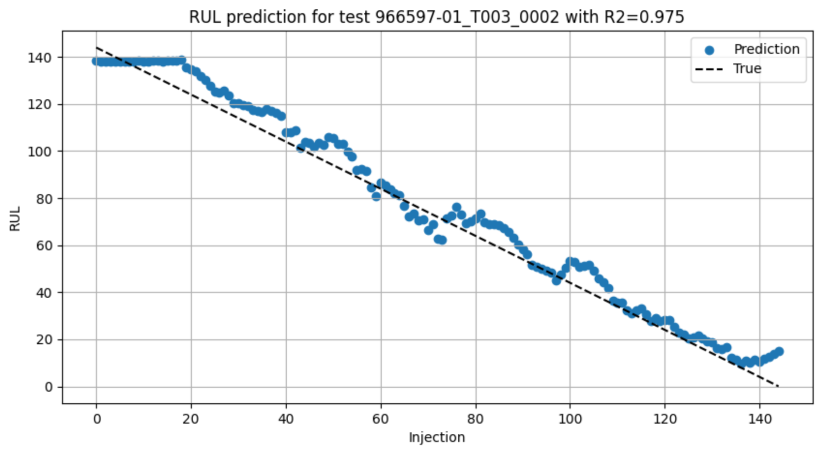


Figure 6.4: True and predicted RUL of test T003 in validation fold 2.

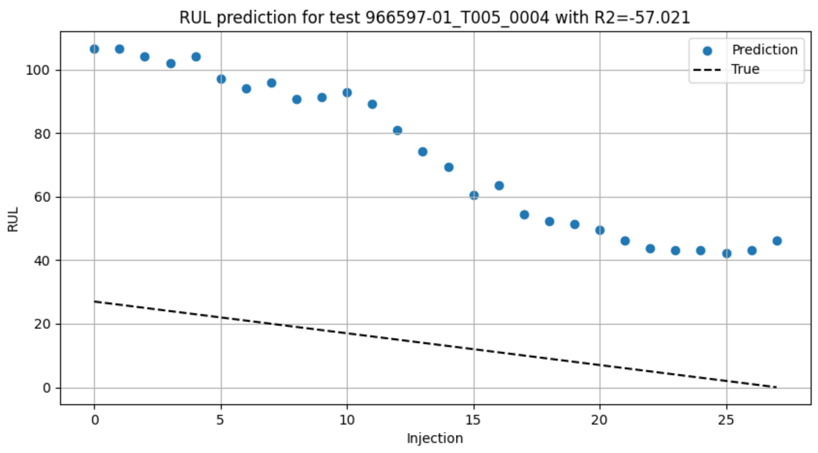


Figure 6.5: True and predicted RUL of test T005 in validation fold 3.

It can be seen that the model is able to predict the RUL for test T003 well with $R^2 = 0.975$. The RUL prediction for test T005 on the hand is poor with $R^2 = -57.021$. It can also be seen that test T005 only has 27 injections, meaning fewer than the RW of 32, which might explain the poor performance. In general it is found that changeovers with fewer injections before failure have significantly worse RUL predictions. Furthermore, the model in general has a hard time predicting smaller RUL values, and it can be seen that the predictions flatten or curve up slightly at the end of the sequence. Plots for all other tests and folds can be found alongside the full hyperparameter sweep results in appendix II.7.

6.3.3 Results for LSTM changeover RUL prediction

The program assumptions for the LSTM regressor used for changeover RUL prediction can be seen in table 6.22, with description of LSTM blocks and fully connected layers.

Component	**Details**
Model Type	Long-short term memory
Hyperparameters	Learning rate, batch size (B), dropout, reading window (RW), attention
Input Shape	(B, 19 features, RW)
LSTM layers	
LSTM block	Number of blocks: 3, Hidden size: 64
Output	Last time step or attention block, shape: (B, 64, 1)
Fully Connected Layers	FC: $64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 1$
Activation	ReLU on all intermediate layer, none on final layer
Output	Shape: (B, 1, 1), Value: $out \in \mathbb{R}$
Loss Function	Mean Square Error (MSE)
Optimizer	Adam

Table 6.22: Summary of the 1D CNN architecture used for changeover RUL prediction.

Given that LSTMs are suited for finding long term dependencies, RUL prediction is tested for longer reading windows to test if it can outperform the 1D CNN. The hyperparameters used in the sweep can be sen in table 6.23. The model is tested both with and without attention layer.

Parameters	**Values**	
Batch size	16	
Attention	True	False
Learning rate	0.0001	0.001
Dropout	0.0	0.3
RW	16	32

Table 6.23: Hyperparameters for LSTM changeover RUL prediction.

As with the two previous models, 30 augmented changeover sequences are generated using DTW-SMOTE with 3 nearest neighbors. Overall the model performs poorly with a maximum average R^2 coefficient over folds of 0.477, and largest R^2 coefficient for a single fold being 0.788. Both of these runs are without attention, have a learning rate of 0.0001, and a RW of 32. The only difference is the dropout. The results for these runs can be seen in table 6.24, and the full results can be found in appendix II.8.

Hyperparameters	**R^2 coefficient per fold**				
Dropout	1	2	3	4	**Avg.**
0.3	0.333	0.421	0.783	0.370	0.477
0.0	0.530	0.306	0.788	-0.132	0.373

Table 6.24: Four fold cross validation results for LSTM changeover RUL prediction evaluated by the R^2 coefficient.

6.4 Result summary

The model results are summarized in table 6.25 in order to answer the outlined prediction problems. The models are evaluated through the best metric scores, averaged over folds in a four fold cross validation. For classification the models are evaluated through the F1-score for failure, and F1-score for stable. The regression models are evaluated by the coefficient of determination R^2, and root mean square error RMSE.

Dataset	**Prediction method**	**Model**	**Evaluation metrics**	
			F1-score failure	**F1-score stable**
Accumulator failure	PW	Random Forest classifier	0.038	0.675
Accumulator failure	PW	1D CNN classifier	0.72	0.69
Accumulator failure	PW	LSTM classifier	0.36	0.66
Changeover failure	PW	Random Forest classifier	0.93	0.90
Changeover failure	PW	1D CNN classifier	0.94	0.74
			R^2	**RMSE**
Changeover failure	RUL	Random Forest regressor	0.781	19.64
Changeover failure	RUL	1D CNN regressor	0.826	17.18
Changeover failure	RUL	LSTM regressor	0.477	27.37

Table 6.25: Summary of failure prediction model results.

From the results it can be concluded the the methanol changeover failures are easier to predict, compared to the accumulator failure. The best performing model for accumulator failure prediction is the 1D CNN, whereas for the changeover failure the random forest classifier has the best performance. For changeover RUL prediction the 1D CNN regressor performs better than both the random forest and LSTM. In general the LSTM models have poor performance compared to other models.

7 Discussion and conclusion

In this chapter the results from chapters 5 and 6 are discussed and concluded upon, considering the the outlined inquiries and research question in section 1.5.

7.1 Discussion

In this section the results from the feature engineering in chapter 5, and the prediction results in chapter 6 are discussed. This discussion aims to provide the foundation for concluding and answering the inquiries and research question outlined in section 1.5

7.1.1 Discussion of accumulator failure results

The challenge with developing models which can predict accumulator failures on the Porsorja Express is the sparsity of system failures within the dataset. Two types of data was acquired from the Posorja Express, similarly to the methanol changeover system, specific measurements of the hydraulic system has been measured in a test setup using Dewesoft, capturing data with high frequency, which has been analyzed in previous work [64]. The data used in this thesis is linerecorder data collected directly from the ship computer, sampling with 1 Hz. The reason for using this is, with respect to a solution implementation, it would be easier to implement since it does not require additional equipment. Furthermore, even though the datasets do not span the exact same time period, the number of failures were almost equal and mostly the same were found in both cases. The linerecorder files contain 20 failures, while the Dewesoft files contain 21 failures.

The low number of failures make it challenging to train parametric models, such as transformers, sufficiently. Data augmentation was used to synthetically generate failure class samples for model training. Early results using this method showed potential. However, once unseen data was introduced in a four fold cross validation the model collapsed. The lack of data for this project is mainly due to time restrictions, as MAN ES was not able to board the ship to gather additional data. However, newly collected data may have posed new challenges, as significant modifications have been made to the cylinder heads in the meantime. These modifications could potentially effect measurements, severely challenging model robustness. If further work is to be done, this lack of data should be addressed. While the early results indicated good results, the introduction of unseen data collapsed the model. It is suspected that this is due to the engine conditions for the specific segment being classified rather than the actual failure. Thus when introducing new data, the engine conditions appear too different to be able to classify.

7.1.2 Discussion of methanol changeover failure results

For the methanol changeover problem generally good results are achieved. For binary failure prediction the random forest performs best, as seen in chapter 6. The random forest results, in section 6.2.1, show that both a large Reading Window of 32 and a smaller RW of 1 perform well. While a RW of 1 has slightly worse performance, it does come with several benefits. A RW of 1 implies that a system failure can, in some cases, be predicted from the first methanol injection. This allows for very early warning, as well as simplifying the model and storage requirements, which can be beneficial in implementation. Similarly, for the random forest regressor, used for changeover remaining useful life prediction, a RW of 32 perform the best as concluded in section 6.3.1. However, in this case the random forest is outperformed by the 1D CNN, which clearly demonstrates superior performance for the larger RW as demonstrated in section 6.3.2. These results indicate that the changeover failure is highly predictable, given the engineered feature set found in section 5.2.5. Furthermore, the conclusions reached, based on the feature engineering, demonstrate the importance of highly interpretable feature selection, as this allows to reach conclusions regarding system failures, how they can be seen, and concretizes the lead up to a failure.

Although the methanol changeover failures are predictable, the models and methods lack implementation feasibility. The reason for this is the aforementioned specified test setup used to collect the data. These measurements are not standard and implementing the prediction model would require extra equipment. Therefore, it is recommended to correlate the findings from the prediction models in this thesis, to the 1Hz sampled linerecorder data during changeover procedures. This would allow to implement the prediction model based on measurements already collected by the ship's computer, without requiring extra equipment. The feature engineering in section 5.2.5 provides a good basis for this possibility, as the data has already been downsampled, using only peak values for each injection.

7.1.3 Prediction model and window size comparison

From section 6.4 it is found that the best performing model for accumulator failure prediction and changeover RUL prediction is the 1D CNN, whereas for changeover failure prediction the random forest classifier performs best. In all cases, the LSTM has the poorest performance. The 1D CNN performs better in cases where a longer RW is needed for prediction, with a RW of 600 for accumulator failure classification, and 32 for changeover RUL prediction. While the random forest classifier used for changeover failure classification has a lower 'stable' class F1-score, for a RW of 32, the model performance for a RW of 1 indicates that failure can be predicted on a single time measurement. This indication implies that time dependencies are not necessary for changeover failure classification, which can explain why the random forest outperforms. For accumulator failure classification and changeover RUL prediction, longer RWs yield better results. This indicates that time dependencies are necessary for failure prediction, in these cases. The 1D CNNs ability to extract temporal features may result in better model predictions, when learning time dependencies is required. The poor performance of the LSTM networks is suspected to be related to the amount of training data, with few failure sequences to train on resulting in low data variance. Furthermore, in for methanol changeover the PW is large, which aligns with the finding sin [8], which also concludes that LSTMs perform worse for larger PWs.

A shorter RW performs better in the case of predicting accumulator failures. Furthermore, as seen in section 6.2.1 for changeover failure prediction, a RW of 1 has the second highest performance. While intuition might dictate that longer RWs provide better results, it can be seen that this is not necessarily the case. The results for different reading windows indicate that the window size has to be appropriate for the task and data. For instance, the cause of an accumulator failure may happen within a short time period. In this case, increasing the window size will outweigh the relevant information, as indicated by the results in section 6.1.2. In the case of changeover failure classification, it seems that either most of- or all of the sequence should be taken into consideration, or failure prediction should be instance based. The medium sized windows likely lead to a higher degree of 'stable' misclassifications, as noisy instances have a greater impact over several windows. This is not an issue for a RW of 1, as the noise is only misclassified once, and it is not a problem for a large RW, as the noise is outweighed. Lastly, for changeover RUL prediction seen in section 6.3.2, there is a clear relation between longer RWs and better prediction results. Furthermore, it seems that longer RWs are necessary compared to binary classification. This can be due to the added complexity of predicting when a failure occurs, rather than if it occurs.

Lastly, the choice of Prediction Window also has a great impact on the model. As with the RW the appropriate size is very task dependent. As seen for accumulator failure in section 6.1.2, a shorter PW is needed, as it is not possible to predict failures for larger windows. This indicates that the the cause or signs of failure are only detectable close to a failure. On the other hand, for the changeover problem, a longer PW yields improved results. This is likely due to the fact that the cause of the failure happens prior to the start of methanol injections. Therefore, the failure is detectable already from the start.

7.1.4 Problem and dataset comparisons

In general, the results in section 6.4 indicate that changeover failures are easier to predict compared to the accumulator failures. This comes down to the nature of the problem and data. As mentioned in section 1.2.2 the preliminary analysis of the measurements from methanol changeover procedures, resulted in a hypothesis regarding captured nitrogen air causing the failures. No similar hypothesis has been able to be made for the accumulator failure problem. While it is possible for ML and AI models to find patterns undetectable to humans, knowing that experts are able to extract useful information from the dataset creates a much stronger foundation to build these models upon.

Furthermore, considering that hydraulic injection systems are high-frequency systems, the higher sampling rate of the methanol changeover data makes it easier to extract relevant features and use feature engineering to find relevant information. Additionally, the measurements for the methanol changeover procedures are much more direct, compared to those of the accumulator failure. The first is data from specified setups measured directly on the hydraulic system, whereas the latter are general monitoring values from the ship's computer. While a data analysis was conducted of high frequency hydraulic measurements on the Porsorja Express in [64], these yielded only discernible patterns up to 250 injections prior to failure, using DTW. This prediction time-frame is too narrow to be useful for taking preventative action. While the prediction time-frame for the changeover problem is similar, it is more useful in this case, as the system can stop a changeover from injection to injection. The engine can always switch back to diesel, meaning no downtime or man power required to prevent system failure.

7.2 Conclusion

The resulting feature sets found in chapter 5 are heavily based on feature engineering, focusing on removing spurious correlations within the feature space, while maintaining feature importance. This has resulted in significantly reduced dimensionality, reducing the curse of dimensionality. In the case of the accumulator failure dataset from the Porsorja Express, the feature space has been significantly reduced by removing the numerous multiples of measurements. It was found that the same measurements are measured on multiple subsystems, resulting in a large number of redundant features. Generally, the found features relate to cylinder health, such as the exhaust temperature, air intake pressure, liner wall temperature, and cylinder control measurements and adjustments. This indicates that a failure can be seen by poor injections in the cylinder which experiences accumulator failure. For the methanol changeover dataset from the LGIM G95, features have been selected mainly based on feature importance with regard to good or bad injections. The auto-correlation of the selected features show clear and significant time dependency between injections. For this reason it was found that looking at the maximum injection values for these features demonstrated a clear pattern for methanol changeover sequences with identifiable system failures. Based on these results, it was possible to describe the failure sequence based on *P886* pressure, cylinder pressure, exhaust valve measurements, air intake pressure, FBIV plunger and needle position, and strain gauge measurements. The results from feature engineering demonstrate the benefits of highly interpretable features, as they provides key insights into the failures.

Several models were tested for failure prediction, with a focus on time sequence data. This included using random forest with lagged data, 1D CNNs, and LSTMs. Notably, transformers were not pursued in this thesis based on recommendation from MAN ES. However, given the results in this thesis, they might be worth testing in future work. To improve model training and performance, SMOTE and DTW-SMOTE was used to augment data. In general the changeover failures were easier to predict, compared to accumulator failure. The best performing model for accumulator failure prediction is the 1D CNN, performing with an accuracy of 71%. For changeover failure the best performing model is the random forest classifier with an accuracy of 91% for a RW of both 32 and 1, with a worse F1-score for stable data with a RW of 1. Lastly, for predicting changeover Remaining Useful Life, the 1D CNN outperforms the other models for the regression task, with an R^2 coefficient of 0.831 for a RW of 32. These results indicate a high degree of predictability for methanol changeover failure, with a lesser degree of predictability for accumulator failure. In general LSTM networks perform poorly, likely due to the limited amount of data.

From conducting hyperparameter sweeps it is found that the Reading Window and Prediction Window size are heavily dependent on the task. In the case of accumulator failure, large RWs reduce model performance, and smaller PWs are necessary, indicating that failures occur without much warning. In the case of changeover failure prediction using random forest classification, small and large RWs perform well, while using medium sized RWs hurt performance. Here, large PWs are required, as it is suspected that the cause of failure occurs prior to methanol injection start. Lastly, larger RWs are required for the regression task of predicting methanol changeover RUL.

Based on these conclusion the research question is answered;

"To what extent can ML and AI be used to predict and prevent total system failure in MAN Energy Solution's maritime propulsion systems?"

ML and AI models can be used to predict methanol changeover failures to a high degree. Furthermore, the applied ML techniques, such as DTW-nn, have given a great insight into the failure sequence. Accumulator failures are predictable to a lesser degree, using the chosen models. It seems that the lack of data makes it difficult to find a repeatable pattern. Overall this thesis has contributed to the research field of predictive maintenance by successfully applying ML and AI models to predict failures on hydraulic injection systems within maritime propulsion. Additionally, the high interprebility of the feature selection and engineering process is not only able to deal with high dimensionality and spurious correlations but also provide valuable insights into system failures. Lastly, it has been highlighted that the window sizes, for binary failure classification and Remaining Useful Life, have a great impact on model performance, and the choice of these should be task specific. In spite of these contributions, there is room for improvement and future work. In terms of methanol changeover failure, it is recommended to correlate the results of this thesis with data collected on the ship's computer through linerecorder data. Developing prediction models fit for linerecorder data would eliminate the requirements for extra equipment. Furthermore, the models should be expanded to include additional engines, other than the LGIM G95, that experience the same issue. Regarding accumulator failure prediction, there is significant room for improvement of the prediction model. To achive this, it is recommended to either collect more data including from other ships, pursue unsupervised methods, re-review the high-frequency data, or combine real data with simulation models. This might provide valuable insights into the problem, more clearly defining the problem and task. Finally, these recommendations open the possibilities for MAN ES to further pursue predictive maintenance and machine failure prediction, for their maritime propulsion systems, based upon the valuable insights and results of this thesis.

Bibliography

[1] International Chamber of Shipping. *Shipping and World Trade: Driving Prosperity.* https://www.ics-shipping.org/shipping-fact/shipping-and-world-trade-driving-prosperity/. Accessed: 30/05/2025. 2022.

[2] European Commission. *Reducing emissions from the shipping sector.* https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-sector_en. Accessed: 12/12/2024.

[3] International Maritime Organization. *Nitrogen Oxides (NOx) – Regulation 13.* https://www.imo.org/en/OurWork/Environment/Pages/Nitrogen-oxides-(NOx)-âĂŞ-Regulation-13.aspx. Accessed: 12/12/2024.

[4] MAN Energy Solutions. *About MAN Energy Solutions.* https://www.man-es.com/company/about-us. Accessed: 12/12/2024.

[5] Clean Air Technology Center (MD-12). *Nitrogen Oxides (NO_x): Why and How They Are Controlled.* Tech. rep. EPA-456/F-99-006R. Accessed: 12/12/2024. Research Triangle Park, NC: United States Environmental Protection Agency, 1999. URL: https://www3.epa.gov/ttncatc1/dir1/fnoxdoc.pdf.

[6] MAN Energy Solutions. *Methanol fuel for the maritime industry.* Accessed: 2025-05-29. -. URL: https://www.man-es.com/marine/strategic-expertise/future-fuels/methanol.

[7] The Editorial Team. *How a vessel switches to low sulfur fuel when entering ECAs.* Accessed: 2025-05-29. 2016. URL: https://safety4sea.com/how-a-vessel-switches-to-low-sulfur-fuel-when-entering-ecas/.

[8] Nicolò Oreste Pinciroli Vago, Francesca Forbicini, and Piero Fraternali. "Predicting Machine Failures from Multivariate Time Series: An Industrial Case Study". In: *Machines* 12.6 (May 2024), p. 357. ISSN: 2075-1702. DOI: 10.3390/machines12060357. URL: http://dx.doi.org/10.3390/machines12060357.

[9] Devendra K. Yadav, Aditya Kaushik, and Nidhi Yadav. "Predicting machine failures using machine learning and deep learning algorithms". In: *Sustainable Manufacturing and Service Economics* 3 (2024), p. 100029. ISSN: 2667-3444. DOI: https://doi.org/10.1016/j.smse.2024.100029. URL: https://www.sciencedirect.com/science/article/pii/S2667344424000124.

[10] Tiago Zonta et al. "Predictive maintenance in the Industry 4.0: A systematic literature review". In: *Computers Industrial Engineering* 150 (2020), p. 106889. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106889. URL: https://www.sciencedirect.com/science/article/pii/S0360835220305787.

[11] Sigurd S. Petersen et al. "An AI-Powered RIS Technology for Hand Gesture Recognition in the Radiating Near Field". In: *2025 IEEE International Workshop on Antenna Technology, iWAT 2025*. IEEE Communications Society, 2025, pp. 1–4. ISBN: 979-8-3315-2737-2. DOI: 10.1109/iWAT64079.2025.10931217.

[12] Dehua Peng, Zhipeng Gui, and Huayi Wu. *Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect*. 2024. arXiv: 2401.00422 [cs.LG]. URL: https://arxiv.org/abs/2401.00422.

[13] Eda Kavlakoglu Jacob Murel. *What is multicollinearity?* https://www.ibm.com/think/topics/multicollinearity. Accessed: 13-02-2025. 2023.

[14] Wenqian Ye et al. *Spurious Correlations in Machine Learning: A Survey*. 2024. arXiv: 2402.12715 [cs.LG]. URL: https://arxiv.org/abs/2402.12715.

[15] Gustavo Batista, Ronaldo Prati, and Maria-Carolina Monard. "A Study of the Behavior of Several Methods for Balancing machine Learning Training Data". In: *SIGKDD Explorations* 6 (June 2004), pp. 20–29. DOI: 10.1145/1007730.1007735.

[16] Seba Susan and Amitesh Kumar. "The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the Art". In: *Engineering Reports* 3.4 (2021), e12298. DOI: https://doi.org/10.1002/eng2.12298. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12298.

[17] Nongnuch Poolsawad, Chandra Kambhampati, and J.G.F. Cleland. "Balancing Class for Performance of Classification with a Clinical Dataset". In: *Proceedings of the World Congress on Engineering* 1 (July 2014), pp. 237–242.

[18] A. Abu-Samah et al. "Failure Prediction Methodology for Improved Proactive Maintenance using Bayesian Approach". In: *IFAC-PapersOnLine* 48.21 (2015). 9th IFAC Symposium on Fault Detection, Supervision andSafety for Technical Processes SAFEPROCESS 2015, pp. 844–851. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.09.632. URL: https://www.sciencedirect.com/science/article/pii/S2405896315017619.

[19] Anjali Parashar et al. *Failure Prediction from Limited Hardware Demonstrations*. 2024. arXiv: 2410.09249 [cs.RO]. URL: https://arxiv.org/abs/2410.09249.

[20] Daeil Kwon et al. "IoT-Based Prognostics and Systems Health Management for Industrial Applications". In: *IEEE Access* 4 (2016), pp. 3659–3670. DOI: 10.1109/ACCESS.2016.2587754.

[21] IBM. *What is principal component analysis (PCA)?* https://www.ibm.com/topics/principal-component-analysis. Accessed: 06/11/2024. 2023.

[22] Jonathon Shlens. *A Tutorial on Independent Component Analysis*. 2014. arXiv: 1404.2986 [cs.LG]. URL: https://arxiv.org/abs/1404.2986.

[23] F. Pedregosa et al. *Manifold learning.* Accessed: 17/02/2025. URL: https://scikit-learn.org/stable/modules/manifold.html.

[24] Mark A. Hall. "Correlation-based Feature Selection for Machine Learning". PhD thesis. The University of Waikato, 1999.

[25] Bahavathy Kathirgamanathan and Padraig Cunningham. "Correlation Based Feature Subset Selection for Multivariate Time-Series Data". In: *CoRR* abs/2112.03705 (2021). URL: https://arxiv.org/abs/2112.03705.

[26] R Muthukrishnan and R Rohini. "LASSO: A feature selection technique in predictive modeling for machine learning". In: *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*. 2016, pp. 18–20. DOI: 10.1109/ICACA.2016.7887916.

[27] Fabiana Camattari et al. *Greedy feature selection: Classifier-dependent feature selection via greedy methods.* 2024. arXiv: 2403.05138 [stat.ML]. URL: https://arxiv.org/abs/2403.05138.

[28] F. Pedregosa et al. *Feature selection.* https://scikit-learn.org/stable/modules/feature_selection.html. Accessed: 05/05/2025. 2011.

[29] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://dx.doi.org/10.1613/jair.953.

[30] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. *TSGM: A Flexible Framework for Generative Modeling of Synthetic Time Series.* 2024. arXiv: 2305.11567 [cs.LG]. URL: https://arxiv.org/abs/2305.11567.

[31] "Dynamic Time Warping". In: *Information Retrieval for Music and Motion.* Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-74048-3. DOI: 10.1007/978-3-540-74048-3_4. URL: https://doi.org/10.1007/978-3-540-74048-3_4.

[32] Karl Bringmann et al. *Dynamic Dynamic Time Warping.* 2023. arXiv: 2310.18128 [cs.CG]. URL: https://arxiv.org/abs/2310.18128.

[33] François Petitjean, Alain Ketterlin, and Pierre Gançarski. "A global averaging method for dynamic time warping, with applications to clustering". In: *Pattern Recognition* 44.3 (2011), pp. 678–693. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2010.09.013. URL: https://www.sciencedirect.com/science/article/pii/S003132031000453X.

[34] Daren Yu et al. "Dynamic time warping constraint learning for large margin nearest neighbor classification". In: *Information Sciences* 181.13 (2011). Including Special Section on Databases and Software Engineering, pp. 2787–2796. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2011.03.001. URL: https://www.sciencedirect.com/science/article/pii/S0020025511001204.

[35] Hossein Pishro-Nik. “Introduction to Probability, Statistics, and Random Processes”. In: Kappa Research LLC, 2014. Chap. 5. URL: https://www.probabilitycourse.com.

[36] Joshua Noble and Eda Kavlakoglu. *What is Autocorrelation?* https://www.ibm.com/think/topics/autocorrelation. Acessed: 05/05/2025.

[37] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python — Feature Importance.* Accessed: 2025-05-28. 2023. URL: https://scikit-learn.org/stable/modules/permutation_importance.html.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. “The Elements of Statistical Learning: Data Mining, Inference, and Prediction”. In: 2nd. New York, NY: Springer, 2009. Chap. 3.

[39] Zahira Marzak et al. “Forecasting Multivariate Time Series with Trend and Seasonality: A Random Forest Approach”. In: *Innovative Intelligent Industrial Production and Logistics.* Ed. by Michele Dassisti, Kurosh Madani, and Hervé Panetto. Cham: Springer Nature Switzerland, 2025, pp. 128–144.

[40] L. E. Peterson. “K-nearest neighbor”. In: *Scholarpedia* 4.2 (2009). revision #137311, p. 1883. DOI: 10.4249/scholarpedia.1883.

[41] Oliver Kramer. “K-Nearest Neighbors”. In: *Dimensionality Reduction with Unsupervised Nearest Neighbors.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 13–23. ISBN: 978-3-642-38652-7. DOI: 10.1007/978-3-642-38652-7_2. URL: https://doi.org/10.1007/978-3-642-38652-7_2.

[42] Aaryan Ohekar. *What is the difference between a Decision Tree Classifier and a Decision Tree Regressor?* https://medium.com/@aaryanohekar277/what-is-the-difference-between-a-decision-tree-classifier-and-a-decision-tree-regressor-36641bd6559c. Accessed: 2025-05-27. 2023.

[43] Lior Rokach and Oded Maimon. “Decision Trees”. In: vol. 6. Jan. 2005, pp. 165–192. DOI: 10.1007/0-387-25465-X_9.

[44] Cynthia Rudin. *Decision Trees, MIT 15.097 Course Notes.* https://ocw.mit.edu/courses/15-097-prediction-machine-learning-and-statistics-spring-2012/f5678de0e329ce097fc6ec6182ebaea2_MIT15_097S12_lec08.pdf. Accessed: 12/05/2025. 2012.

[45] Floriana Esposito et al. “A Comparative Analysis of Methods for Pruning Decision Trees”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19 (June 1997), pp. 476 –491. DOI: 10.1109/34.589207.

[46] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python.* https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation. Accessed: 2025-05-21. 2011.

[47] IBM. *What is random forest?* https://www.ibm.com/think/topics/random-forest. Accessed: 03/05/2025. 2024.

[48] Leo Breiman. *Random Forests*. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. Accessed: 03/05/2025. 2001.

[49] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE]. URL: https://arxiv.org/abs/1511.08458.

[50] Serkan Kiranyaz et al. "1D convolutional neural networks and applications: A survey". In: *Mechanical Systems and Signal Processing* 151 (2021), p. 107398. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398. URL: https://www.sciencedirect.com/science/article/pii/S0888327020307846.

[51] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[52] MathWorks. *Long Short-Term Memory Networks*. Accessed: 06/05/2025. 2024. URL: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html.

[53] Davi Guimarães da Silva and Anderson Alvarenga de Moura Meneses. "Comparing Long Short-Term Memory (LSTM) and bidirectional LSTM deep neural networks for power consumption prediction". In: *Energy Reports* 10 (2023), pp. 3315–3334. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2023.09.175. URL: https://www.sciencedirect.com/science/article/pii/S2352484723014208.

[54] Benjamin Lindemann et al. "A survey on long short-term memory networks for time series prediction". In: *Procedia CIRP* 99 (2021). 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020, pp. 650–655. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2021.03.088. URL: https://www.sciencedirect.com/science/article/pii/S2212827121003796.

[55] Manu Chauhan. *A simple overview of RNN, LSTM and Attention Mechanism*. https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b. Accessed: 2025-05-27. 2021.

[56] Stefania Cristina. *The Attention Mechanism from Scratch*. https://machinelearningmastery.com/the-attention-mechanism-from-scratch/. Accessed: 2025-05-27. 2023.

[57] Qingsong Wen et al. *Transformers in Time Series: A Survey*. 2023. arXiv: 2202.07125 [cs.LG]. URL: https://arxiv.org/abs/2202.07125.

[58] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[59] Zuo X. Li-R. et al. Su L. "A systematic review for transformer-based long-term series forecasting". In: *Artificial Intelligence Review* 58 (2025). DOI: https://doi.org/10.1007/s10462-024-11044-2. URL: https://link.springer.com/article/10.1007/s10462-024-11044-2.

[60] Jordan Hoffmann et. al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556.

[61] Li Shen et al. *On Efficient Training of Large-Scale Deep Learning Models: A Literature Review*. 2023. arXiv: `2304.03589 [cs.LG]`. URL: `https://arxiv.org/abs/2304.03589`.

[62] Dina Elreedy, Amir F. Atiya, and Firuz Kamalov. “A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning”. In: *Machine Learning* 113 (2024), pp. 4903–4923. DOI: `10.1007/s10994-022-06296-4`. URL: `https://doi.org/10.1007/s10994-022-06296-4`.

[63] Xinyu Yang et al. “A Time Series Data Augmentation Method Based on Dynamic Time Warping”. In: *2021 International Conference on Computer Communication and Artificial Intelligence (CCAI)*. 2021, pp. 116–120. DOI: `10.1109/CCAI50917.2021.9447507`.

[64] Emil Lytje-Dorfman Jacob V. Køpke. *Machine Learning Based Signal Analy- sis for Detecting Accumulator Failures in Hydraulic Systems*. Tech. rep. Aalborg University, 2025.

[65] Keith G. Calkins. *Applied Statistics - Lesson 5 - Correlation Coefficients*. `https://www.andrews.edu/~calkins/math/edrm611/edrm05.htm`. Accessed: 03/06/2025. 2005.

[66] Haldun Akoglu. “User’s guide to correlation coefficients”. In: *Turkish Journal of Emergency Medicine* 18.4 (2018), pp. 91–93. DOI: `10.1016/j.tjem.2018.08.001`.

[67] *Correlation - Statistics Resources*. `https://resources.nu.edu/statsresources/correlation`. Accessed: 03/06/2025. 2025.

[68] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. Accessed: 2025-05-27. 2023. URL: `https://scikit-learn.org/stable/`.

I Methanol changeover data analysis full results

In the appendix all results from the methanol changeover data analysis are found.

I.1 Results of DTW-nn for the methanol changeover injection classification

All results for DTW nearest neighbor injection classification for all features and all folds.

Feature/Test	7&35	7&30	7&44	7&39	8&35	8&30	8&44	8&39	10 11&35
PM886-8.FOR	0.75	1	1	1	0.75	1	1	1	0.94
PExhBnd8	1	1	1	1	0.94	0.75	0.75	0.75	1
SG3Aft	0.5	0.5	0.5	0.5	1	1	1	1	1
PlungerMan_mm	0.5	1	1	1	0.5	1	1	1	0.47
SG2Aft	0.5	0.5	0.5	0.5	1	1	1	1	1
PGasCh8	0.5	1	1	1	0.5	1	1	1	0.25
LFBIV8Man_mm	0.5	0.97	0.97	0.97	0.5	0.94	0.94	0.94	0.5
LFuelV8Aft	0.5	0.94	0.94	0.94	0.5	1	1	1	0.25
PM886-8. AFT	0.5	1	1	1	0.5	1	1	1	0.69
PlungerAft_mm	0.5	1	1	1	0.5	1	1	1	0.5
PCylDAU8	0.5	1	1	1	0.5	1	1	1	0.47
Plunger_Aft	0.5	1	1	1	0.5	1	1	1	0.47
SG1Man	1	1	1	0.94	0.64	0.64	0.64	0.58	0.97
LELFI8_mm	0.5	0.53	0.53	0.53	0.5	1	1	1	0.5
CELFI8_mm	0.5	0.53	0.53	0.53	0.5	1	1	1	0.5
PM886-8.MAN	0.5	0.94	0.94	0.94	0.5	1	1	1	0.5
LFuelV8Man	0.5	0.94	0.94	0.94	0.5	0.94	0.94	0.94	0.25

Feature/Test	10 11&30	10 11&44	10 11&39	20&35	20&30	20&44	20&39	Avg
PM886-8.FOR	1	1	1	0.75	1	1	1	0.95
PExhBnd8	1	1	1	1	1	1	1	0.95
SG3Aft	1	1	1	1	1	1	1	0.88
PlungerMan_mm	0.97	0.97	0.97	0.5	1	1	1	0.87
SG2Aft	1	1	1	1	0.94	0.94	0.94	0.86
PGasCh8	0.94	0.75	0.75	0.5	1	1	1	0.82
LFBIV8Man_mm	0.83	0.83	0.83	0.5	0.92	0.92	0.92	0.81
LFuelV8Aft	0.86	0.75	0.75	0.5	0.89	0.89	0.89	0.79
PM886-8. AFT	0.89	0.89	0.89	0.67	0.5	0.5	0.5	0.78
PlungerAft_mm	0.97	0.97	0.97	0.5	0.5	0.5	0.5	0.78
PCylDAU8	0.97	0.97	0.97	0.5	0.5	0.5	0.5	0.77
Plunger_Aft	0.97	0.97	0.97	0.5	0.5	0.5	0.5	0.77
SG1Man	0.97	0.97	0.92	0.5	0.5	0.5	0.58	0.77
LELFI8_mm	0.92	0.92	0.92	0.5	1	1	1	0.77
CELFI8_mm	0.89	0.89	0.89	0.5	1	1	1	0.77
PM886-8.MAN	0.89	0.89	0.89	0.69	0.5	0.5	0.5	0.76
LFuelV8Man	0.81	0.75	0.75	0.5	0.81	0.81	0.81	0.76

Feature/Test	7&35	7&30	7&44	7&39	8&35	8&30	8&44	8&39	10 11&35
LFuelP8	0.5	1	1	1	0.5	0.92	0.92	0.92	0.25
PScav8	0.39	0.97	0.89	0.89	0.5	1	1	1	0.5
LExhValveFor8 FOR	0.5	0.67	0.69	0.69	0.5	0.97	1	1	0.17
PCylDAU6	0.5	0.5	0.5	0.5	0.5	0.86	0.86	0.86	0.47
PCyl8	0.5	1	1	1	0.5	1	1	1	0.36
LFBIV8 MAN	0.5	1	1	1	0.5	0.86	0.86	0.86	0.36
LFBIV8Man_filter	0.5	1	1	1	0.5	0.86	0.86	0.86	0.36
LELFI-L8	0.5	1	1	1	0.5	0.94	0.94	0.94	0.25
CELFI8	0.5	0.67	0.67	0.67	0.5	0.94	0.94	0.94	0.25
SG1Aft	0.75	1	0.5	1	0.75	1	0.5	1	0.86
CELFI-L8	0.5	1	1	1	0.5	0.92	0.92	0.92	0.25
PCylDAU5	0.5	0.5	0.5	0.5	0.5	1	1	1	0.47
LELFI-L8_mm	0.89	0.56	0.56	0.56	0.5	1	1	1	0.33
PLubHole8	0	0.5	0.5	0.5	0.5	1	1	1	0.5
PCylDAU1	0.5	0.5	1	1	0.5	0.5	1	1	0.25
PFuelP8	0.5	0.25	0.53	0.53	0.5	0.72	1	1	0.33
Plunger8_Man	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25

Feature/Test	10 11&30	10 11&44	10 11&39	20&35	20&30	20&44	20&39	Avg
LFuelP8	0.86	0.75	0.75	0.5	0.72	0.72	0.72	0.75
PScav8	0.92	0.92	0.92	0.5	0.5	0.5	0.5	0.74
LExhValveFor8 FOR	0.83	0.67	0.67	0.5	0.97	1	1	0.74
PCylDAU6	0.92	0.92	0.92	0.5	1	1	1	0.74
PCyl8	0.81	0.81	0.81	0.5	0.5	0.5	0.5	0.74
LFBIV8 MAN	0.94	0.86	0.86	0.5	0.5	0.5	0.5	0.73
LFBIV8Man_filter	0.94	0.86	0.86	0.5	0.5	0.5	0.5	0.73
LELFI-L8	0.92	0.72	0.72	0.5	0.5	0.5	0.5	0.72
CELFI8	0.97	0.75	0.75	0.5	0.78	0.78	0.78	0.71
SG1Aft	0.89	0.89	0.89	0.33	0.5	0	0.5	0.71
CELFI-L8	0.92	0.72	0.72	0.5	0.5	0.5	0.5	0.71
PCylDAU5	0.92	0.92	0.92	0.5	0.69	0.69	0.69	0.71
LELFI-L8_mm	0.81	0.81	0.81	0.5	0.67	0.67	0.67	0.71
PLubHole8	0.75	0.75	0.75	0.5	1	1	1	0.7
PCylDAU1	0.67	0.64	0.67	0.5	0.5	1	1	0.7
PFuelP8	0.97	0.81	0.81	0.5	0.72	1	1	0.7
Plunger8_Man	0.75	0.75	0.75	0.5	0.5	1	1	0.69

Feature/Test	7&35	7&30	7&44	7&39	8&35	8&30	8&44	8&39	10 11&35
PGasSupply	0.5	0.5	1	1	0.5	0.44	0.94	0.94	0.25
LExhValveFor8_filter	0.5	0.44	0.69	0.69	0.5	0.72	0.97	0.97	0.17
CELBI8	0.5	1	1	1	0.5	0.86	0.86	0.86	0.17
CELFI-L8_mm	0.83	0.53	0.53	0.53	0.5	1	1	1	0.42
PLPS8	0.5	1	1	1	0.5	0.72	0.72	0.72	0.25
VFBIV8Man	0.83	0.86	0.5	0.86	0.5	0.53	0.5	0.53	0.75
LFBIV8Aft_mm	0.19	0.69	0.97	0.69	0.5	1	0.97	1	0.47
PLPS_ReturnOilValve8	0.5	0.5	0.5	0.5	0.5	0.92	0.92	0.92	0.39
PFuelV8Aft	0.5	0.03	0.53	0.53	0.5	0.5	1	1	0.31
PCylDAU2	0.5	0.53	0.94	0.94	0.5	0.58	1	1	0.25
PCylDAU3	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.42
CELFI8_pilot	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
EngLoad_calc	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.31
LELFI8	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
P455-8	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
PExhVDamper8	0.5	0.5	1	1	0.5	0.5	1	1	0.25
LELBI8	0	0.64	0.44	0.5	0.5	0.86	0.86	0.92	0.17

Feature/Test	10 11&30	10 11&44	10 11&39	20&35	20&30	20&44	20&39	Avg
PGasSupply	0.78	0.75	0.75	0.5	0.39	0.89	0.89	0.69
LExhValveFor8_filter	0.86	0.67	0.67	0.5	0.72	0.97	0.97	0.69
CELBI8	0.78	0.67	0.67	0.5	0.53	0.53	0.53	0.68
CELFI-L8_mm	0.83	0.83	0.83	0.5	0.53	0.53	0.53	0.68
PLPS8	0.92	0.75	0.75	0.5	0.5	0.5	0.5	0.68
VFBIV8Man	0.75	0.89	0.75	0.61	0.64	0.5	0.64	0.66
LFBIV8Aft_mm	0.72	0.72	0.72	0.5	0.5	0.47	0.5	0.66
PLPS_ReturnOilValve8	0.81	0.81	0.81	0.5	0.64	0.64	0.64	0.65
PFuelV8Aft	0.89	0.78	0.78	0.5	0.5	1	1	0.65
PCylDAU2	0.94	0.72	0.72	0.31	0.11	0.53	0.53	0.63
PCylDAU3	0.92	0.83	0.83	0.5	0	0.5	0.5	0.63
CELFI8_pilot	0.92	0.75	0.75	0.5	0.11	0.61	0.61	0.63
EngLoad_calc	0.97	0.81	0.81	0.5	0.03	0.53	0.53	0.63
LELFI8	0.92	0.75	0.75	0.5	0.08	0.58	0.58	0.63
P455-8	0.92	0.75	0.75	0.5	0.08	0.58	0.58	0.63
PExhVDamper8	0.75	0.75	0.75	0.5	0	0.5	0.5	0.63
LELBI8	0.58	0.58	0.64	0.5	0.92	0.92	0.97	0.63

Feature/Test	7&35	7&30	7&44	7&39	8&35	8&30	8&44	8&39	10 11&35
PFuelV8Man	0.5	0.53	0.53	0.53	0.5	0.89	0.89	0.89	0.25
Frequency	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
PCylDAU4	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.33
PSeal_FBIV	0.5	0.5	1	1	0.5	0.39	0.89	0.89	0.25
PScavAbs	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
PAirspring8	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
PHPS8	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.25
VFBIV8Aft	0.86	0.86	0.83	0.86	0.5	0.5	0.5	0.5	0.58
SG3Man	0.36	0.5	0.5	0.5	0.75	0.69	0.69	0.69	0.75
PCylDAU7	0.5	0.58	1	0.81	0.5	0.58	1	0.81	0.17
CELVA8	0.5	0.5	1	1	0.5	0.36	0.86	0.86	0.17
CSync8	0.5	0	0.5	0.5	0.5	0.5	1	1	0.06
KnockSensor1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.47
LFBIV8 AFT	0	0	0.5	0.5	0.5	0.31	0.81	0.81	0.25
LFBIV8Aft_filter	0	0	0.5	0.5	0.5	0.31	0.81	0.81	0.25
SG2Man	0	0	0.5	0.5	0.5	0.31	0.81	0.81	0.25
PIndCock8	0	0.5	0.5	0.53	0.5	0.5	1	0.5	0.5

Feature/Test	10 11&30	10 11&44	10 11&39	20&35	20&30	20&44	20&39	Avg
PFuelV8Man	0.97	0.72	0.72	0.5	0.5	0.5	0.5	0.62
Frequency	0.97	0.75	0.75	0.5	0.03	0.53	0.53	0.62
PCylDAU4	0.92	0.78	0.78	0.5	0	0.5	0.5	0.62
PSeal_FBIV	0.97	0.75	0.75	0.5	0	0.5	0.5	0.62
PScavAbs	1	0.75	0.75	0.5	0	0.5	0.5	0.61
PAirspring8	0.86	0.75	0.75	0.5	0.03	0.53	0.53	0.61
PHPS8	0.92	0.75	0.75	0.5	0	0.5	0.5	0.61
VFBIV8Aft	0.58	0.58	0.58	0.5	0.5	0.5	0.5	0.61
SG3Man	0.69	0.69	0.69	0.39	0.67	0.53	0.53	0.6
PCylDAU7	0.89	0.67	0.67	0.5	0.08	0.5	0.31	0.6
CELVA8	0.78	0.67	0.67	0.5	0.03	0.53	0.53	0.59
CSync8	0.56	0.56	0.56	0.5	0.47	0.97	0.97	0.57
KnockSensor1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
LFBIV8 AFT	0.86	0.75	0.75	0.5	0	0.5	0.5	0.47
LFBIV8Aft_filter	0.86	0.75	0.75	0.5	0	0.5	0.5	0.47
SG2Man	0.75	0.75	0.75	0.5	0	0.5	0.5	0.46
PIndCock8	0.25	0.75	0.25	0.5	0	0.5	0	0.42

I.2 Full correlation matrix

Full correlation matrix for all features in the methanol changeover dataset can be seen in figure I.1

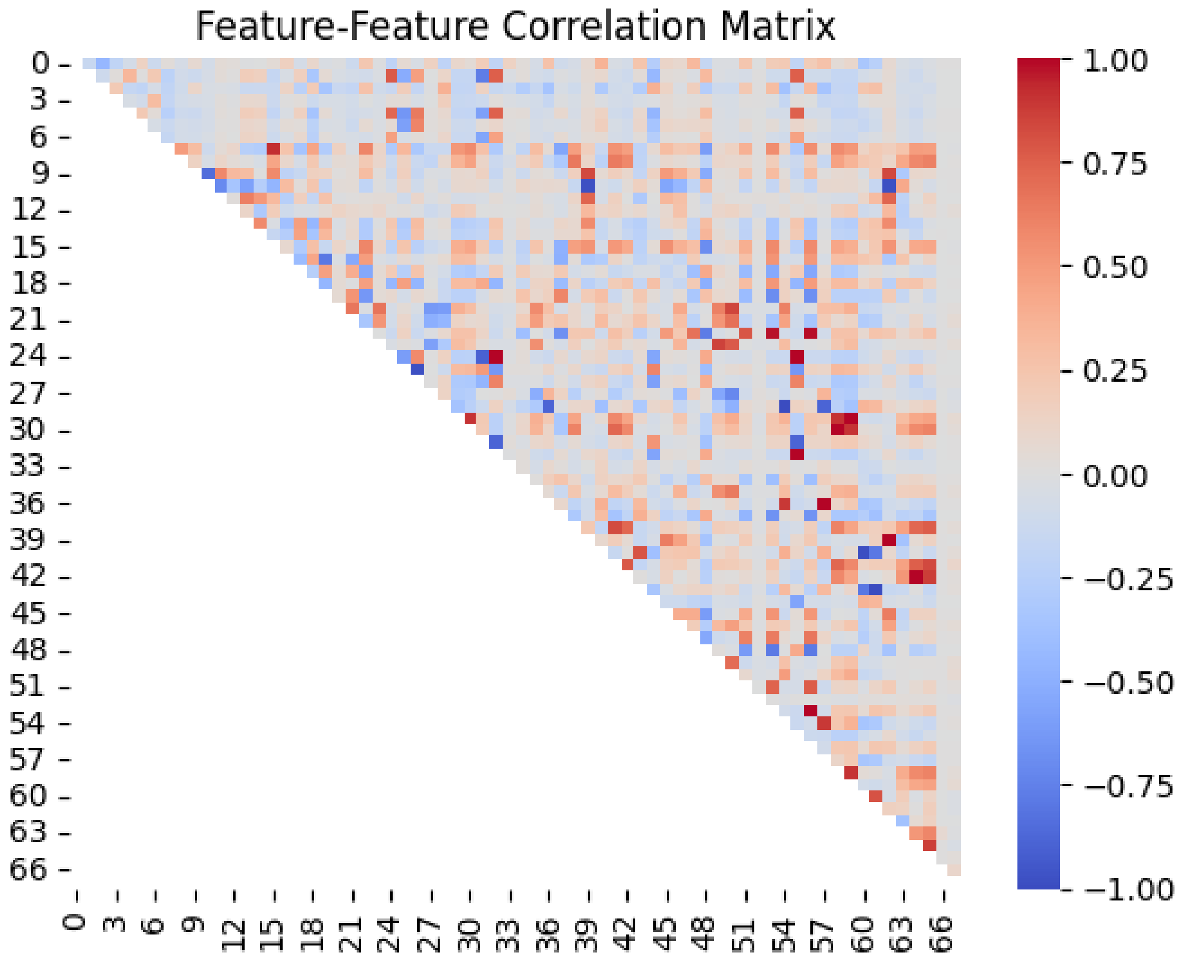


Figure I.1: Correlation matrix of methanol changeover features.

I.3 Highly correlated feature pairs

Feature 1	Feature 2	Correlation
CELFI8	CELFI8_mm	1
LELFI-L8 [ZT6424-8]	LELFI-L8_mm	1
LFBIV8 [ZM-8.MAN]	LFBIV8Man_filter	0.992479969
PCylDAU8 [PT1422-8]	PCyl8 [PM1422-8.1]	0.917223897
CELFI-L8 [XC6429-8]	LELFI-L8 [ZT6424-8]	0.903615151
CELFI-L8 [XC6429-8]	LELFI-L8_mm	0.90361515
CELFI8	LELFI8_mm	0.886462346
LELFI8_mm	CELFI8_mm	0.886462346
LFBIV8 [ZM-8.MAN]	LFBIV8Man_mm	0.872129944
LFBIV8Man_filter	LFBIV8Man_mm	0.861296673
LFuelV8Man	LFuelV8Aft	0.852549421
P886-8 [PM886-8.MAN]	LFBIV8Man_mm	0.833140269
P886-8 [PM886-8. AFT]	P886-8 [PM886-8.MAN]	0.826149059
LFBIV_Plunger8_Aft_mm	LFBIV_Plunger8_Man_mm	0.803732477
LFBIV_Plunger8_Aft	LFBIV_Plunger8_Man	0.790930369
P886-8 [PM886-8.MAN]	LFBIV8 [ZM-8.MAN]	0.776855916
P886-8 [PM886-8. AFT]	LFBIV8Man_mm	0.761389152
P886-8 [PM886-8.MAN]	LFBIV8Man_filter	0.755006896
PCylDAU5 [PT1422-5]	LExhValveFor8 [ZT4111-8.FOR]	0.74464756
P886-8 [PM886-8. AFT]	LFBIV8 [ZM-8.MAN]	0.724883878
P886-8 [PM886-8. AFT]	LFBIV8Man_filter	0.719970284
P886-8 [PM886-8.MAN]	LELFI-L8_mm	0.719780144
LELFI-L8 [ZT6424-8]	P886-8 [PM886-8.MAN]	0.719780141
LFBIV_Plunger8_Aft	LFBIV_Plunger8_Man_mm	-0.792911922
LFBIV_Plunger8_Man	LFBIV_Plunger8_Aft_mm	-0.802817853
LFBIV_Plunger8_Aft	LFBIV_Plunger8_Aft_mm	-0.984810613
LFBIV_Plunger8_Man	LFBIV_Plunger8_Man_mm	-0.999872518

Table I.1: List of feature pairs with a numerical correlation coefficient above 0.7.

II | Validation results

In this appendix all results from validating models in chapter 6 can be found.

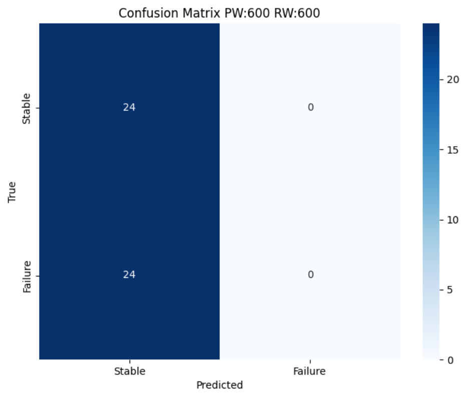
II.1 Random forest accumulator failure prediction

The initial hyperparameter sweep results conducted, by training on SMOTE data and validated on real data, can be seen in table II.1 for all hyperparameters resulting in no misclassifications.

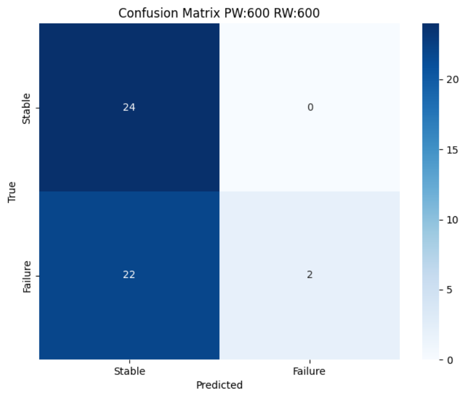
Parameters					**Metrics**	
n_estimators	log_loss	pw	rw	stride	accuracy	f1-score
150	0.129	1200	600	150	1	1
150	0.0936	600	600	100	1	1
50	0.092	600	600	100	1	1
50	0.080	600	1500	100	1	1
100	0.083	600	1500	100	1	1
200	0.131	600	1500	150	1	1
50	0.097	600	1800	100	1	1
150	0.153	600	600	150	1	1
150	0.092	600	1800	100	1	1
100	0.154	600	600	150	1	1
50	0.138	600	1500	150	1	1
200	0.083	600	1500	100	1	1
100	0.130	600	1500	150	1	1
50	0.154	600	600	150	1	1
150	0.082	600	1500	100	1	1
100	0.093	600	600	100	1	1
50	0.101	600	900	100	1	1
100	0.129	1200	600	150	1	1
200	0.096	600	600	100	1	1
100	0.092	600	1800	100	1	1
200	0.131	600	1500	150	1	1
200	0.154	600	600	150	1	1
150	0.130	600	1500	150	1	1

Table II.1: Table of initial hyperparameter sweep validation results for random forest accumulator failure prediction, for all hyperparmeters resulting in no misclassifications.

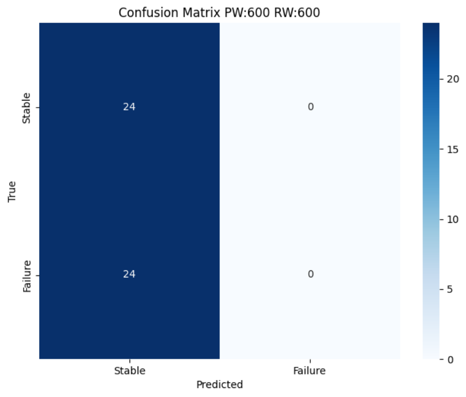
Confusion matrices for all folds in the four fold cross validation for a reading window and prediction window of 600, with a learning rate of 0.001, and a dropout of 0.0.



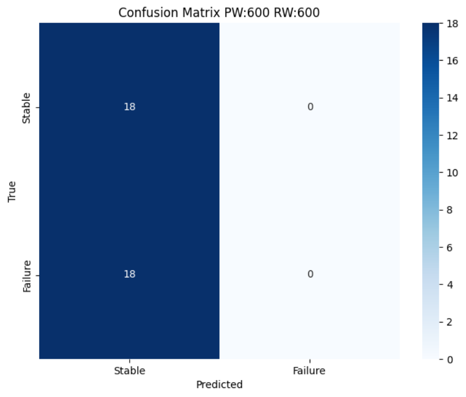
(a) Confusion matrix for fold 1



(b) Confusion matrix for fold 2



(c) Confusion matrix for fold 3



(d) Confusion matrix for fold 4.

Figure II.1: Confusion matrices for all four folds in the four fold cross validation.

II.2 1D CNN accumulator failure prediction

Results of the hyperparameter sweep conducted for 1D CNN accumulator failure prediction can be seen in table II.2. The hyperparameters are evaluated by accuracy F1-score for failure, F1-score for stable, and loss. These metrics are averaged across a four fold cross validation.

Parameters				**Evaluation metrics**			
dropout	lr	pw	rw	accuracy	f1 failure	f1 stable	loss
0.300	0.001	600	600	0.594	0.711	0.316	0.795
0.000	0.001	600	600	0.706	0.719	0.693	1.045
0.300	0.000	600	1800	0.562	0.533	0.588	1.308
0.300	0.001	600	1800	0.475	0.323	0.571	1.774
0.300	0.000	600	600	0.537	0.362	0.637	1.995
0.000	0.001	600	1800	0.558	0.243	0.676	2.284
0.000	0.000	600	600	0.487	0.000	0.655	2.657
0.000	0.000	600	1800	0.500	0.000	0.667	3.309

Table II.2: Table of four fold cross validation hyperparameter sweep results for 1D CNN accumulator failure prediction. Metrics are the averaged across folds.

II.3 LSTM accumulator failure prediction

Results of the hyperparameter sweep conducted for LSTM accumulator failure prediction can be seen in table II.3. The hyperparameters are evaluated by accuracy F1-score for failure, F1-score for stable, and loss. These metrics are averaged across a four fold cross validation.

Parameters			**Evaluation metrics**			
pw	rw	attention	accuracy	f1 failure	f1 stable	loss
600	600	True	0.5	0	0.667	0.693
600	600	False	0.556	0.360	0.660	0.693

Table II.3: Table of four fold cross validation hyperparameter sweep results for LSTM accumulator failure prediction. Metrics are the average across folds.

II.4 Random forest changeover failure prediction

The results of random forest changeover prediction, for initial hyperparameter sweep trained on SMOTE data and validated on real data, can be found in tables II.4 and II.5.

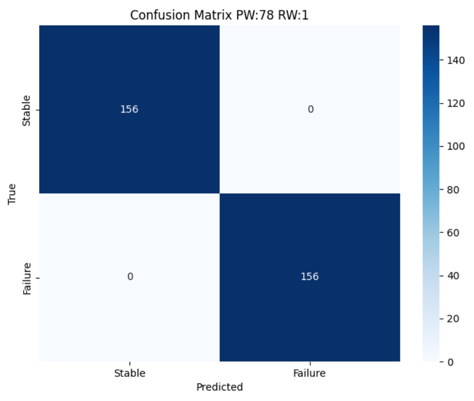
Parameters			**Evaluation metrics**		
n_estimators	pw	rw	accuracy	f1-score	log_loss
75	78	8	1.000	1.000	0.006
100	64	32	1.000	1.000	0.006
100	78	32	1.000	1.000	0.006
75	64	32	1.000	1.000	0.006
75	78	32	1.000	1.000	0.006
100	78	8	1.000	1.000	0.006
100	64	16	1.000	1.000	0.007
100	78	16	1.000	1.000	0.007
75	78	16	1.000	1.000	0.007
75	64	16	1.000	1.000	0.007
100	78	4	1.000	1.000	0.007
75	78	4	1.000	1.000	0.007
75	78	1	1.000	1.000	0.008
100	78	1	1.000	1.000	0.008
100	78	2	1.000	1.000	0.008
75	78	2	1.000	1.000	0.009

Table II.4: Table of initial hyperparameter sweep results for random forest changeover failure prediction, for all hyperparameters resulting in no misclassifiactions.

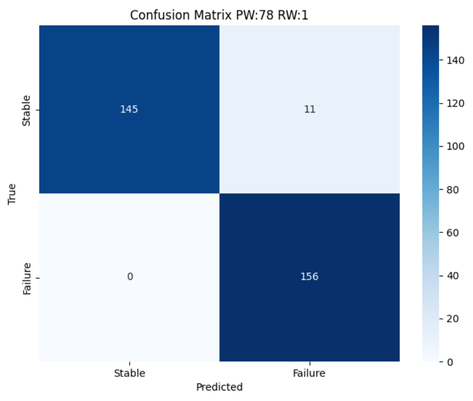
Parameters			Evaluation metrics		
n_estimators	pw	rw	accuracy	f1-score	log_loss
100	32	8	0.993	0.995	0.046
75	32	8	0.993	0.995	0.047
100	32	4	0.993	0.995	0.057
75	32	2	0.993	0.996	0.057
100	32	2	0.993	0.996	0.057
75	32	4	0.989	0.993	0.058
100	64	4	0.993	0.994	0.060
75	64	4	0.993	0.994	0.060
75	64	8	0.981	0.982	0.061
100	64	8	0.981	0.982	0.061
100	32	16	0.987	0.991	0.062
75	32	32	0.989	0.991	0.062
75	32	16	0.983	0.988	0.062
100	32	32	0.989	0.991	0.062
100	64	1	0.986	0.988	0.063
75	32	1	0.976	0.985	0.065
100	32	1	0.976	0.985	0.066
75	64	1	0.983	0.985	0.066
100	64	2	0.990	0.991	0.066
75	64	2	0.993	0.994	0.066
75	8	32	0.989	0.994	0.080
100	8	32	0.989	0.994	0.081
75	8	4	0.965	0.981	0.109
100	8	4	0.965	0.981	0.110
100	8	16	0.966	0.982	0.121
75	8	16	0.958	0.977	0.123
100	8	8	0.948	0.971	0.136
75	8	8	0.955	0.976	0.136
100	8	1	0.932	0.963	0.162
75	8	1	0.932	0.963	0.163
100	8	2	0.924	0.958	0.173
75	8	2	0.927	0.960	0.174

Table II.5: Table of remaining results of initial hyperparameter sweep for random forest changeover failure prediction.

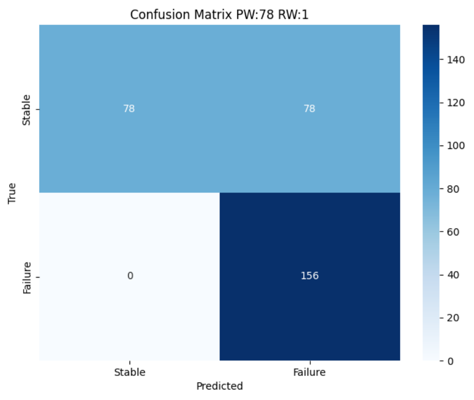
Results of the secondary hyperparameter sweep for $RW = \{1, 8, 32\}$, $PW = 78$, and $n\_estimators = 100$ can be seen in the cofusion matrices in figures II.2, II.3, and II.4.



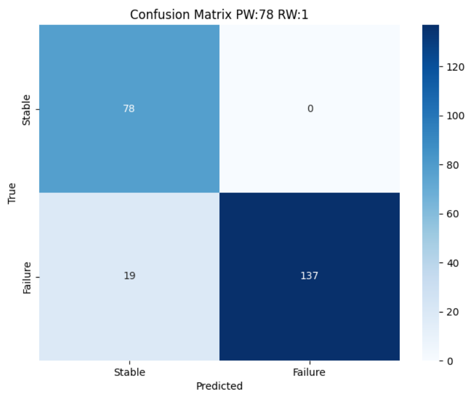
(a) Fold 1



(b) Fold 2

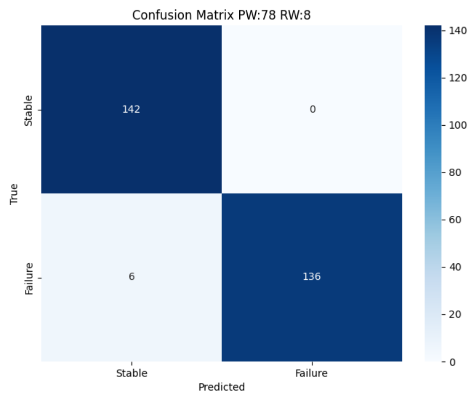


(c) Fold 3

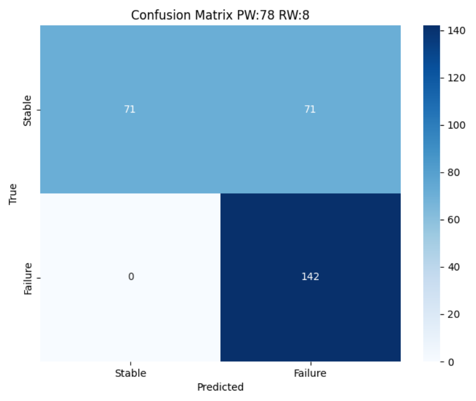


(d) Fold 4

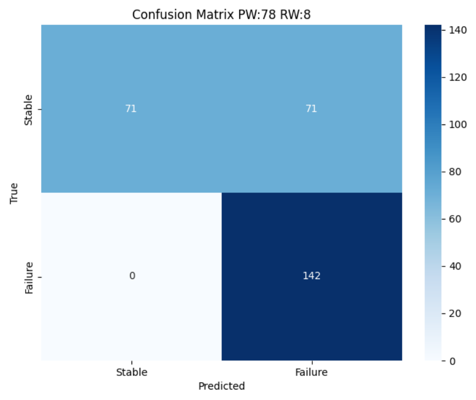
Figure II.2: Confusion matrices from four fold cross validation with $PW = 78$, $RW = 1$, $n\_estimators = 100$.



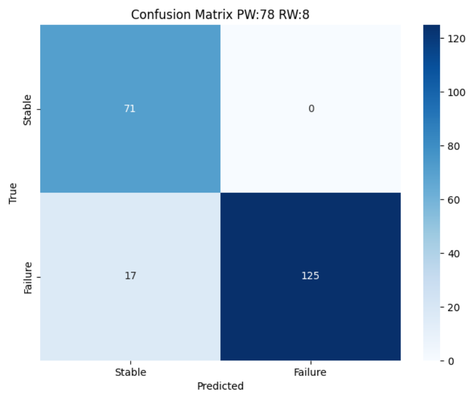
(a) Fold 1



(b) Fold 2

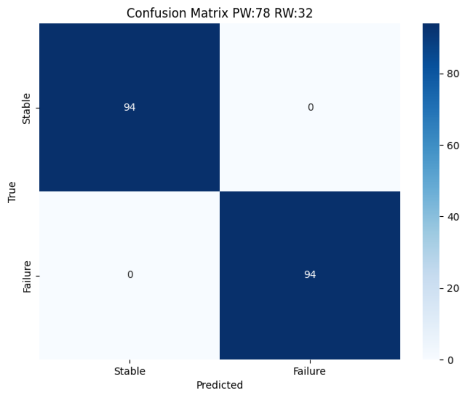


(c) Fold 3

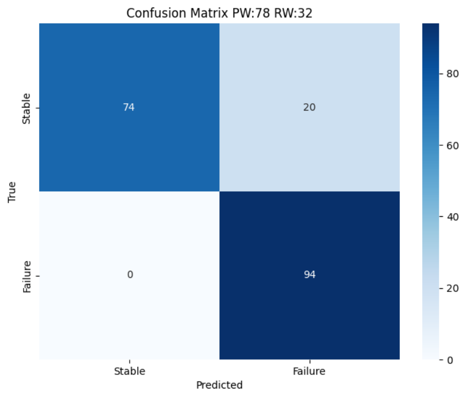


(d) Fold 4

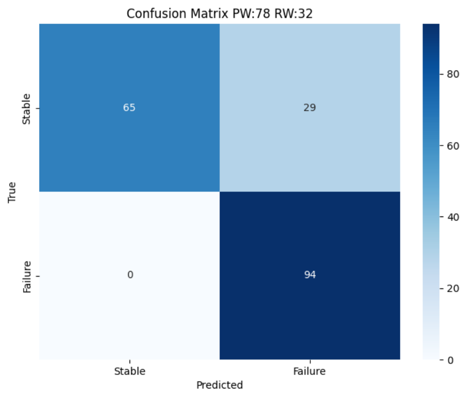
Figure II.3: Confusion matrices from four fold cross validation with $PW = 78$, $RW = 8$, $n\_estimators = 100$.



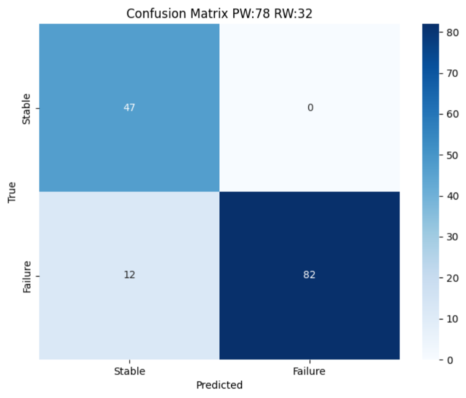
(a) Fold 1



(b) Fold 2



(c) Fold 3



(d) Fold 4

Figure II.4: Confusion matrices from four fold cross validation with $PW = 78$, $RW = 32$, $n\_estimators = 100$.

II.5 1D CNN changeover failure prediction

Results of the hyperparameter sweep conducted for 1D CNN changeover failure prediction can be seen in table II.6. The hyperparameters are evaluated by accuracy F1-score for failure, F1-score for stable, and loss. These metrics are averaged across a four fold cross validation.

Parameters					**Metrics**			
batch_size	dropout	lr	pw	rw	accuracy	f1 failure	f1 stable	loss
16	0.0	0.0001	64	16	0.882	0.922	0.707	0.429
32	0.3	0.0001	64	16	0.906	0.941	0.737	0.319
16	0.0	0.0001	64	16	0.897	0.933	0.726	0.498
32	0.0	0.0001	64	16	0.892	0.93	0.72	0.388
32	0.0	0.0001	78	16	0.887	0.924	0.716	0.424
32	0.3	0.0001	78	16	0.883	0.92	0.712	0.364
32	0.0	0.001	64	16	0.882	0.919	0.711	0.768
16	0.3	0.0001	78	8	0.882	0.92	0.717	0.511
16	0.3	0.0001	78	16	0.874	0.913	0.698	0.486
32	0.0	0.0001	78	4	0.867	0.906	0.704	0.491
16	0.3	0.0001	64	16	0.865	0.905	0.688	0.468
32	0.3	0.001	78	16	0.865	0.901	0.695	0.765
32	0.3	0.001	78	8	0.863	0.9	0.699	0.922
32	0.0	0.0001	78	8	0.862	0.899	0.698	0.45
32	0.0	0.001	78	16	0.861	0.895	0.693	0.577
16	0.0	0.001	78	4	0.857	0.896	0.693	1.054
16	0.3	0.001	78	8	0.852	0.889	0.688	1.19
32	0.3	0.0001	78	4	0.847	0.886	0.685	0.449
32	0.3	0.0001	78	8	0.847	0.886	0.681	0.48
16	0.0	0.0001	78	4	0.846	0.884	0.685	0.621
16	0.3	0.001	78	16	0.843	0.896	0.637	1.042
16	0.0	0.0001	78	8	0.838	0.881	0.667	0.54
32	0.3	0.001	64	16	0.837	0.89	0.633	0.825
16	0.0	0.0001	78	16	0.829	0.881	0.631	0.555
16	0.0	0.001	64	16	0.828	0.875	0.642	0.974

batch_size	dropout	lr	pw	rw	accuracy	f1 failure	f1 stable	loss
16	0.3	0.001	78	4	0.827	0.873	0.655	1.176
32	0.0	0.001	78	4	0.825	0.877	0.637	0.923
16	0.0	0.001	78	16	0.824	0.877	0.624	1.035
32	0.3	0.001	78	4	0.818	0.875	0.619	1.131
32	0.0	0.001	78	8	0.818	0.872	0.622	1.121
16	0.3	0.0001	78	4	0.816	0.872	0.623	0.601
16	0.3	0.001	64	16	0.813	0.865	0.616	0.891
16	0.0	0.001	78	8	0.811	0.865	0.62	1.029
16	0.3	0.001	32	8	0.692	0.499	0.761	1.477
32	0.3	0.001	32	4	0.684	0.363	0.773	1.463
16	0.0	0.0001	64	4	0.681	0.633	0.689	0.66
32	0.0	0.0001	32	4	0.676	0.48	0.756	0.68
32	0.3	0.0001	32	8	0.675	0.475	0.736	0.686
32	0.0	0.001	32	16	0.674	0.373	0.751	1.793
32	0.3	0.001	64	8	0.674	0.691	0.623	1.593
16	0.0	0.0001	32	4	0.673	0.464	0.744	0.723
32	0.0	0.001	32	8	0.672	0.47	0.748	1.467
32	0.0	0.0001	32	16	0.67	0.502	0.702	0.816
16	0.0	0.0001	32	16	0.666	0.461	0.732	0.793
32	0.3	0.001	32	8	0.666	0.486	0.738	1.611
16	0.3	0.0001	32	8	0.665	0.46	0.745	0.781
16	0.3	0.0001	32	16	0.663	0.509	0.699	0.865
32	0.0	0.0001	32	8	0.66	0.503	0.714	0.752
16	0.3	0.001	32	16	0.658	0.451	0.731	1.474
32	0.0	0.0001	64	8	0.653	0.595	0.627	0.756
16	0.3	0.0001	32	4	0.653	0.331	0.733	0.825
16	0.0	0.001	32	4	0.652	0.297	0.734	1.46
16	0.0	0.0001	32	8	0.651	0.476	0.704	0.716
32	0.3	0.0001	32	16	0.649	0.484	0.706	0.709
16	0.0	0.001	32	16	0.649	0.482	0.721	1.683
32	0.0	0.0001	64	4	0.641	0.526	0.671	0.786

batch_size	dropout	lr	pw	rw	accuracy	f1 failure	f1 stable	loss
32	0.3	0.0001	64	8	0.641	0.542	0.657	0.787
16	0.3	0.001	32	4	0.639	0.33	0.719	1.369
32	0.3	0.0001	32	4	0.639	0.342	0.713	0.686
32	0.3	0.0001	64	4	0.635	0.531	0.641	0.747
16	0.3	0.0001	64	4	0.632	0.523	0.645	0.734
32	0.3	0.001	32	16	0.627	0.331	0.692	1.649
16	0.3	0.001	64	4	0.626	0.522	0.601	1.743
16	0.3	0.0001	64	8	0.622	0.544	0.605	0.808
16	0.0	0.0001	64	8	0.62	0.517	0.62	0.834
32	0.0	0.001	32	4	0.617	0.377	0.712	1.333
32	0.0	0.001	64	4	0.598	0.565	0.53	1.926
16	0.0	0.001	64	4	0.597	0.53	0.556	1.346
16	0.3	0.001	64	8	0.592	0.571	0.512	2.039
32	0.3	0.001	64	4	0.592	0.506	0.563	1.711
32	0.0	0.001	64	8	0.59	0.531	0.538	1.474
16	0.0	0.001	64	8	0.587	0.53	0.508	1.732
16	0.0	0.001	32	8	0.587	0.308	0.656	2.161
32	0.0	0.0001	78	4				

Table II.6: Table of four fold cross validation hyperparameter sweep results for 1D CNN changeover failure prediction. Metrics are the averaged across folds.

II.6 Random forest changeover RUL prediction

Results of the hyperparameter sweep conducted for random forest changeover RUL prediction can be seen in table II.7. The hyperparameters are evaluated by mean absolute error, R^2, and root mean squared error. These metrics are averaged across a four fold cross validation.

Parameters		**Metrics**		
n_estimators	rw	MAE	R2	RMSE
150	32	10.999	0.781	19.635
100	32	11.071	0.778	19.7734
50	32	11.122	0.772	19.971
150	16	13.668	0.730	23.801
100	16	13.708	0.725	24.033
50	16	13.540	0.7296	23.803
150	8	15.303	0.706	25.721
100	8	15.297	0.706	25.712
50	8	15.721	0.691	26.376
150	4	14.221	0.755	23.785
100	4	14.180	0.755	23.785
50	4	14.117	0.759	23.676
150	1	14.406	0.756	24.014
100	1	14.551	0.753	24.184
50	1	14.367	0.758	23.857

Table II.7: Table of four fold cross validation hyperparameter sweep results for random forest changeover RUL prediction. Metrics are the averaged across folds.

II.7 1D CNN changeover RUL prediction

Results of the hyperparameter sweep conducted for 1D CNN changeover RUL prediction can be seen in table II.8. The hyperparameters are evaluated by mean absolute error, R^2, and root mean squared error. These metrics are averaged across a four fold cross validation.

Parameters			**Metrics**			
dropout	lr	rw	Loss	MAE	R2	RMSE
0.0	0.0001	32	300.512	10.867	0.831	16.995
0.0	0.0001	32	300.512	10.867	0.831	16.995
0.0	0.0001	32	307.534	11.602	0.826	17.182
0.3	0.0001	32	325.226	13.009	0.817	17.854
0.0	0.001	32	348.57	11.531	0.803	18.213
0.0	0.0001	16	417.254	13.572	0.801	20.068
0.3	0.001	32	360.83	13.311	0.787	18.851
0.3	0.0001	16	459.025	15.874	0.78	21.308
0.0	0.001	8	507.23	15.942	0.775	22.402
0.0	0.0001	4	544.543	16.764	0.77	23.148
0.3	0.001	16	481.031	15.946	0.762	21.689
0.0	0.001	16	461.419	14.482	0.76	20.891
0.0	0.0001	8	550.846	16.381	0.76	22.975
0.0	0.001	4	552.71	15.84	0.749	22.995
0.3	0.0001	8	567.159	17.96	0.743	23.664
0.3	0.0001	4	619.539	19.132	0.737	24.838
0.3	0.001	8	605.478	18.782	0.721	24.238
0.3	0.001	4	658.011	19.103	0.72	25.588

Table II.8: Table of four fold cross validation hyperparameter sweep results for lstm changeover RUL prediction. Metrics are the averaged across folds.

The predicted and true RUL are plotted for all sequences in all folds as seen in the figures below. This is done for the best performing hyperparameters $RW = 32$, $learning\,rate = 0.0001$, $drop\,out = 0.0$.

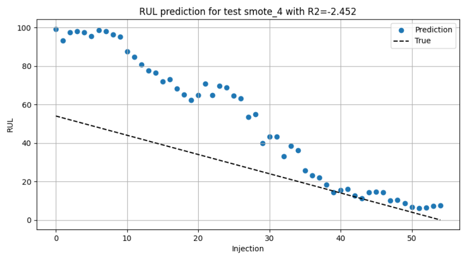


Figure II.5: Smote 4, $R^2 = -2.45$

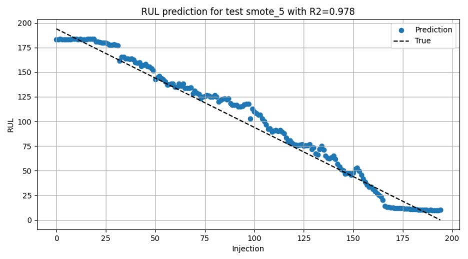


Figure II.6: Smote 5, $R^2 = 0.978$

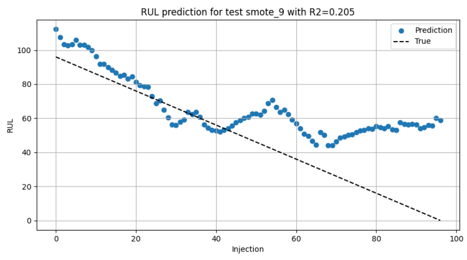


Figure II.7: Smote 9, $R^2 = 0.205$

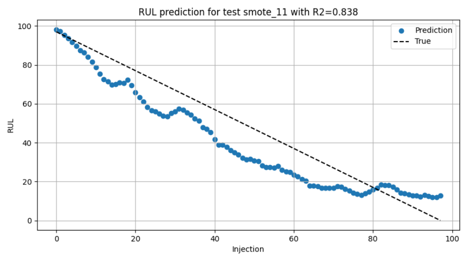


Figure II.8: Smote 11, $R^2 = 0.838$

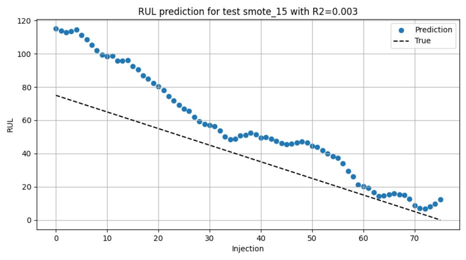


Figure II.9: Smote 15, $R^2 = 0.003$

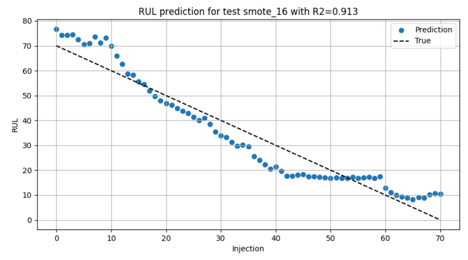


Figure II.10: Smote 16, $R^2 = 0.913$

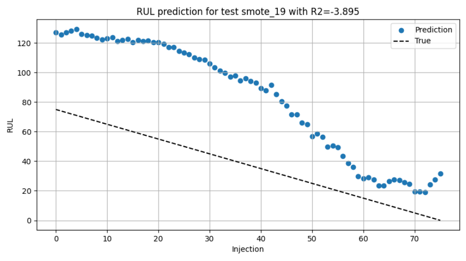


Figure II.11: Smote 19, $R^2 = 0. - 3.895$

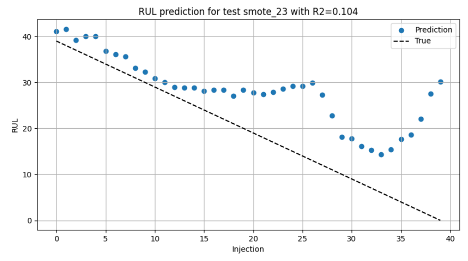


Figure II.12: Smote 23, $R^2 = 0.104$

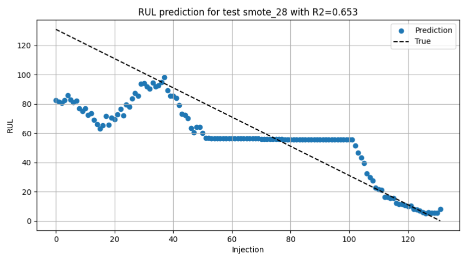


Figure II.13: Smote 28, $R^2 = 0.653$

Figure II.14: Fold 1

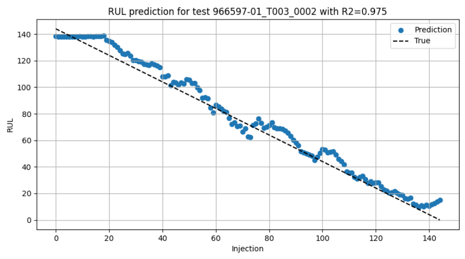


Figure II.15: T003, $R^2 = 0.975$

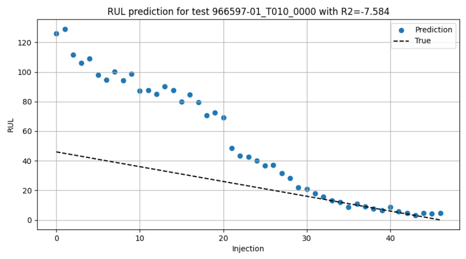


Figure II.16: T010, $R^2 = -7.584$

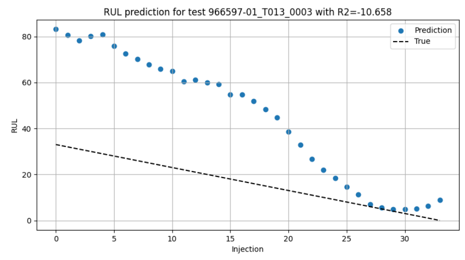


Figure II.17: T013, $R^2 = -10.7$

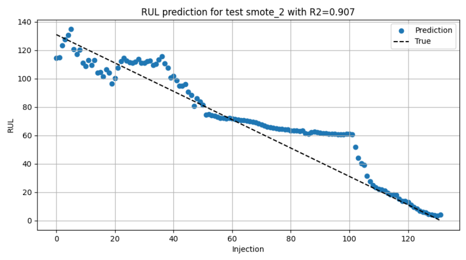


Figure II.18: Smote 2, $R^2 = 0.907$

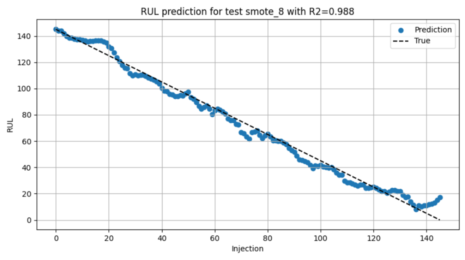


Figure II.19: Smote 8, $R^2 = 0.988$

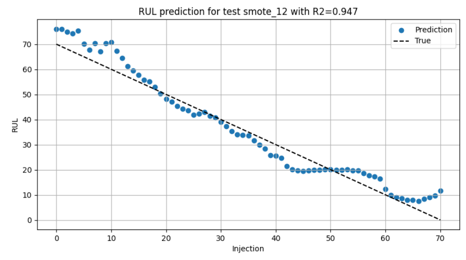


Figure II.20: Smote 12, $R^2 = 0.947$

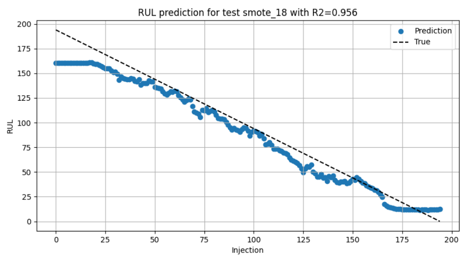


Figure II.21: Smote 18, $R^2 = 0.956$

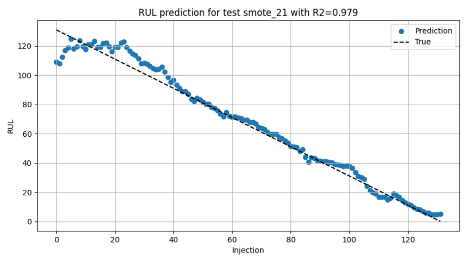


Figure II.22: Smote 21, $R^2 = 0.979$

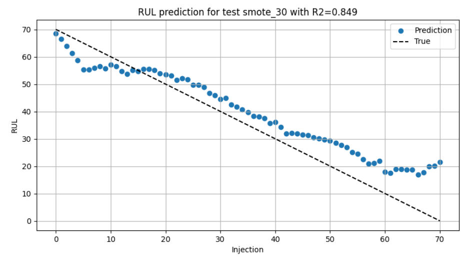


Figure II.23: Smote 30, $R^2 = 0.849$

Figure II.24: Fold 2

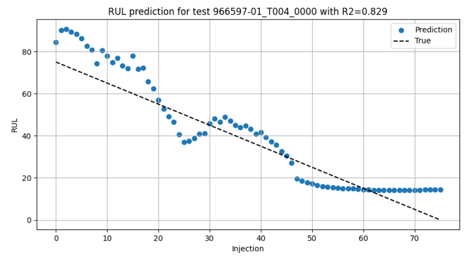


Figure II.25: T004, $R^2 = 0.829$

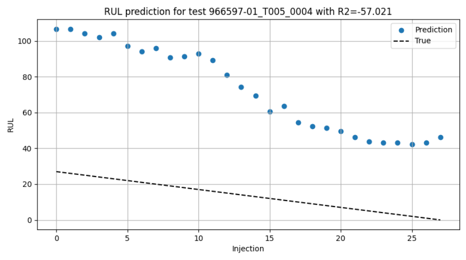


Figure II.26: T005, $R^2 = -57.0$

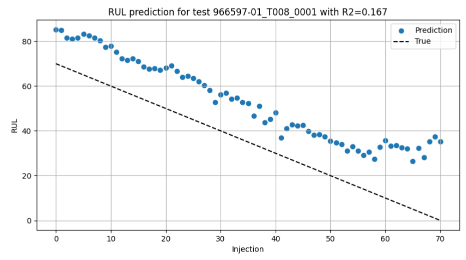


Figure II.27: T008, $R^2 = 0.167$

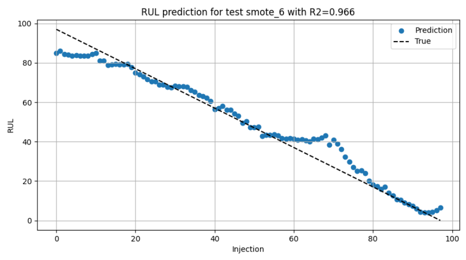


Figure II.28: Smote 6, $R^2 = 0.966$

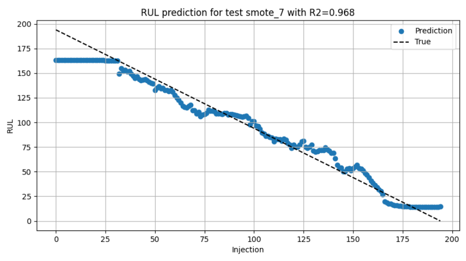


Figure II.29: Smote 7, $R^2 = 0.968$

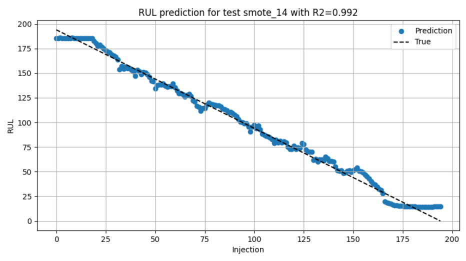


Figure II.30: Smote 14, $R^2 = 0.992$

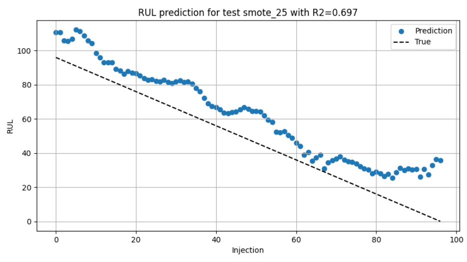


Figure II.31: Smote 25, $R^2 = 0.697$

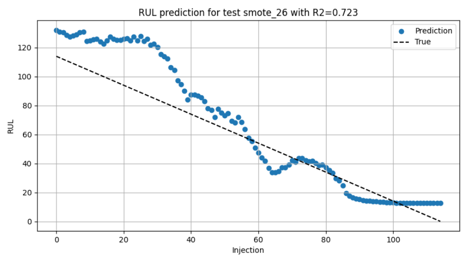


Figure II.32: Smote 26, $R^2 = 0.723$

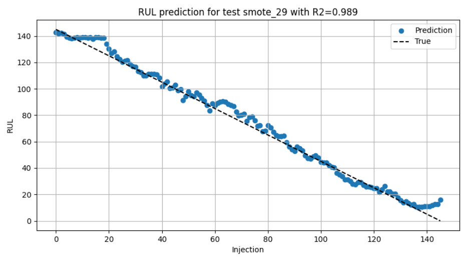


Figure II.33: Smote 29, $R^2 = 0.989$

Figure II.34: Fold 3

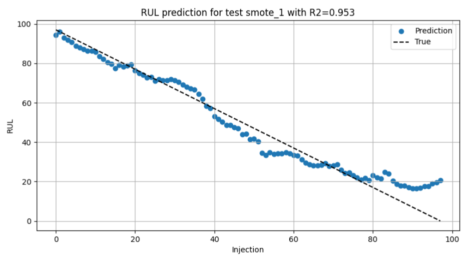


Figure II.35: Smote 1, $R^2 = 0.953$

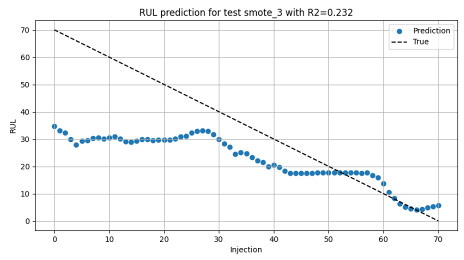


Figure II.36: Smote 3, $R^2 = 0.232$

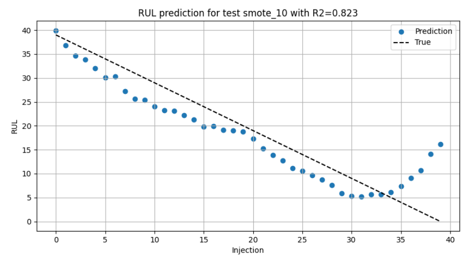


Figure II.37: Smote 10, $R^2 = 0.823$

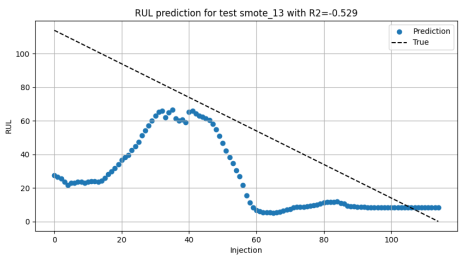


Figure II.38: Smote 13, $R^2 = -0.529$

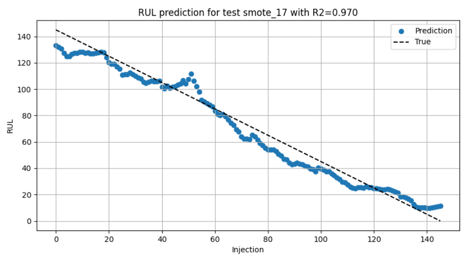


Figure II.39: Smote 17, $R^2 = 0.970$

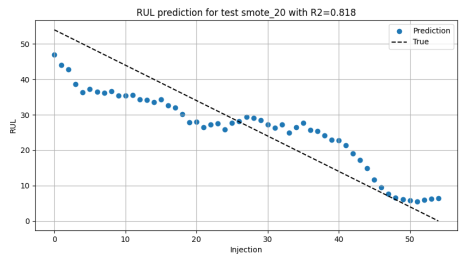


Figure II.40: Smote 20, $R^2 = 0.818$

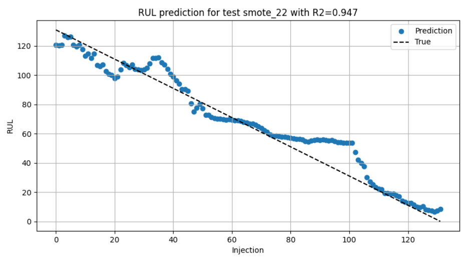


Figure II.41: Smote 22, $R^2 = 0.947$

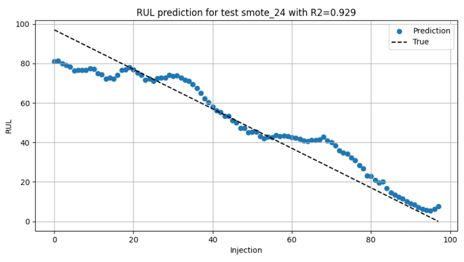


Figure II.42: Smote 24, $R^2 = 0.929$

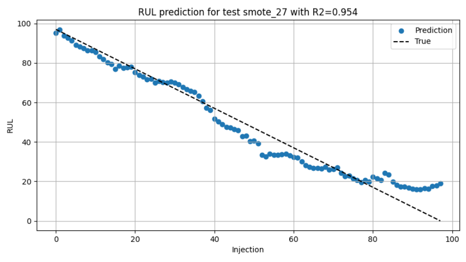


Figure II.43: Smote 27, $R^2 = 0.954$

Figure II.44: Fold 4

II.8 LSTM changeover RUL prediction

Results of the hyperparameter sweep conducted for LSTM changeover RUL prediction can be seen in table II.9. The hyperparameters are evaluated by mean absolute error, R^2, and root mean squared error. These metrics are averaged across a four fold cross validation.

Parameters					**Metrics**			
attention	batch_size	dropout	lr	rw	Loss	MAE	R2	RMSE
true	16	0.3	0.0001	32	1029.186	19.216	0.419	30.539
false	16	0.3	0.0001	32	793.428	20.533	0.477	27.366
true	16	0	0.0001	32	809.946	19.311	0.410	28.062
false	16	0	0.0001	32	802.007	20.387	0.373	28.140
true	16	0.3	0.001	32	1466.079	22.842	0.027	36.873
false	16	0.3	0.001	32	929.787	19.793	0.387	29.496
true	16	0	0.001	32	1092.440	22.136	0.202	32.712
false	16	0	0.001	32	1228.876	22.523	0.088	34.949

Table II.9: Table of four fold cross validation hyperparameter sweep results for lstm changeover RUL prediction. Metrics are the averaged across folds.

III | Network architecture diagrams

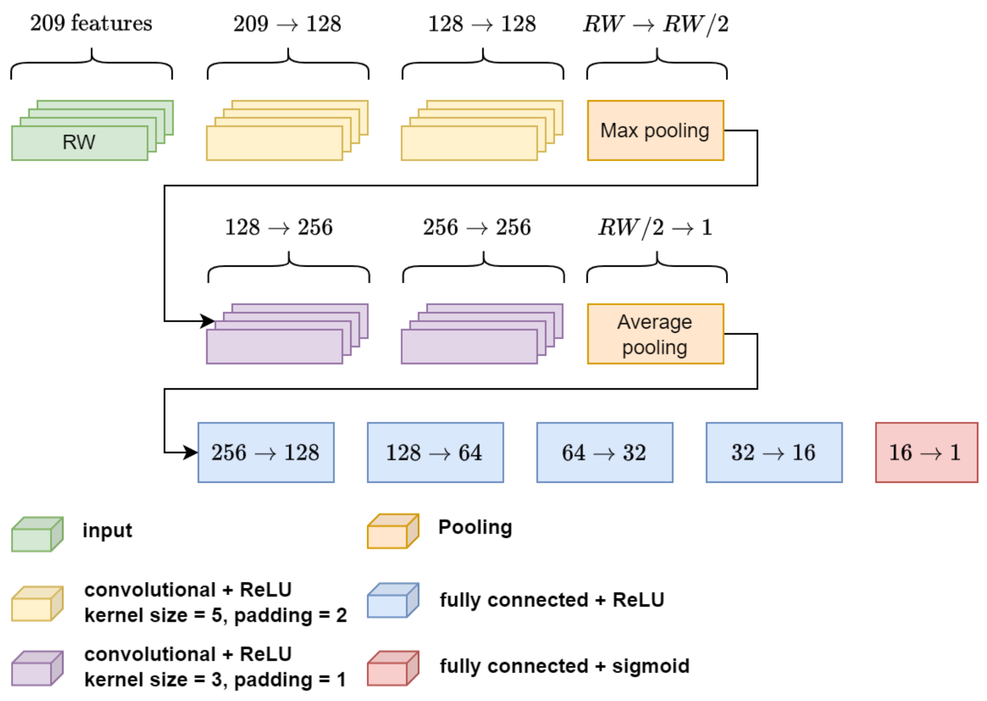


Figure III.1: Proposed network architecture for accumulator failure prediction with a reading window RW and prediction window PW.

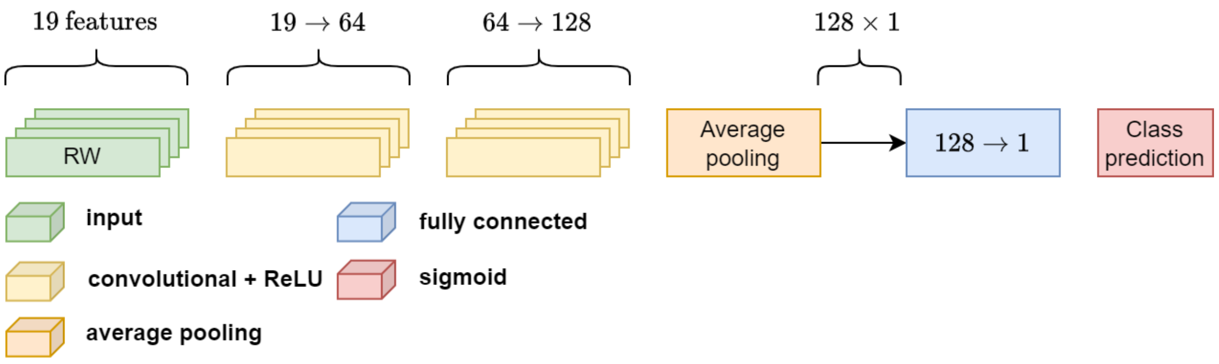


Figure III.2: Proposed network architecture for changeover failure prediction with a reading window RW and prediction window PW.

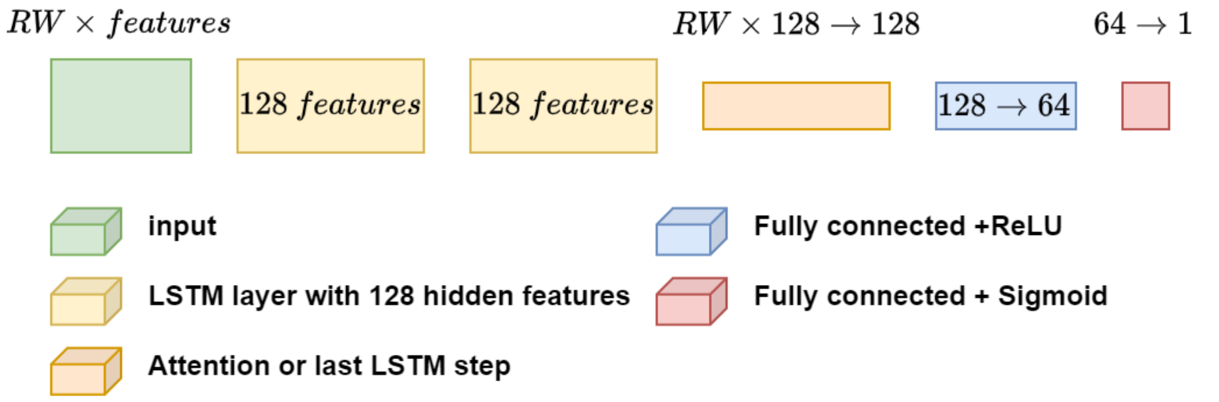


Figure III.3: Proposed LSTM classification network for accumulator failure prediction for reading window RW and prediction window PW.

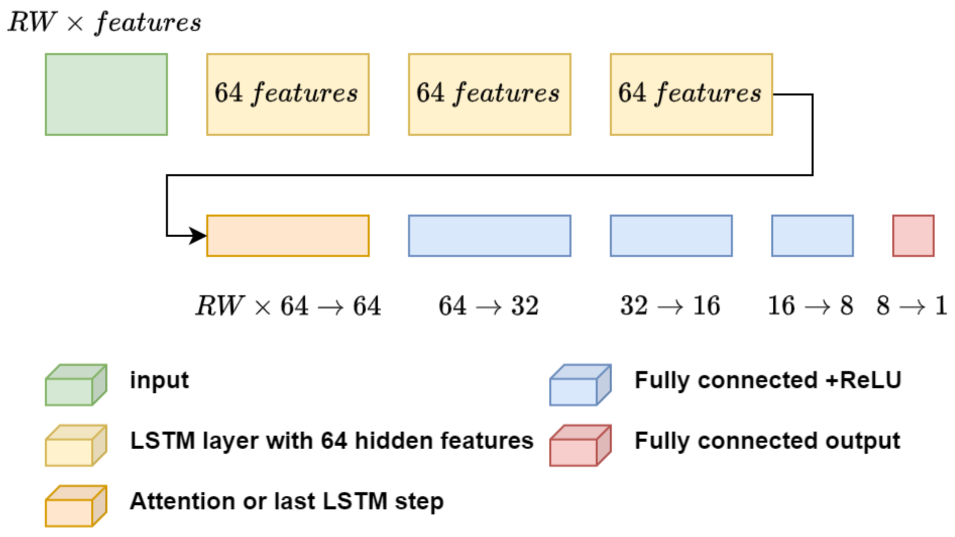


Figure III.4: Proposed LSTM regression network for methanol changeover RUL prediction for reading window RW.