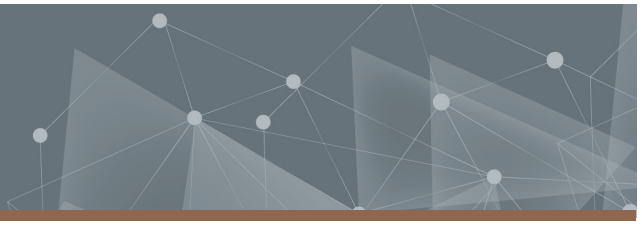
CHALMERS
UNIVERSITY OF TECHNOLOGY



Development of a Standardized Fluid Management Dashboard with Predictive Analytics using Machine Learning

Degree project report in Production Engineering

Julius Johansson
Erik Roman Kilander

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project report 2025

Development of a Standardized Fluid Management Dashboard with Predictive Analytics Using Machine Learning

Julius Johansson
Erik Roman Kilander

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Chalmers University of Technology
Gothenburg, Sweden 2025

Development of a Standardized Fluid Management Dashboard with Predictive Analytics Using Machine Learning

JULIUS JOHANSSON
ERIK ROMAN KILANDER

Supervisor: Silvan Marti, Industrial and Materials Science
Examiner: Björn Johansson, Industrial and Materials Science
Industrial supervisor: Anirudha Das, AB SKF

Degree project report 2025
Department of Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Abstract

This master's thesis explores the design and evaluation of a real-time monitoring dashboard for industrial oil filtration systems, with a focus on integrating predictive analytics to support data-driven maintenance strategies. The primary goal was to develop dashboards that provide operators with useful forecasts and visualizations to better understand system performance. To assess different implementation approaches, two cloud-based platforms, Microsoft Fabric and Databricks, were used to build proof-of-concept solutions. This allowed for a comparative analysis of their strengths in managing real-time data streams and supporting AI integration.

The analytical part of the project centered on two key predictive tasks: estimating the remaining useful life of the filtration systems and forecasting future sensor values. While both platforms successfully supported real-time dashboard functionality, the performance of the predictive models was limited. This was largely due to data-related constraints, specifically insufficient historical data, limited laboratory sampling, and weak correlations between sensor readings and actual filter conditions. In particular, attempts to predict filter condition from the available sensor data proved unreliable, highlighting the need for more comprehensive and representative datasets.

Despite these challenges, the project revealed what worked well and what fell short when adding predictive features. The findings emphasize the importance of improving both the quantity and quality of data to enhance predictive accuracy. Although current constraints affect the reliability of the predictive components, the developed dashboard solutions offer a solid foundation for improving operational visibility and enabling smart maintenance practices. If improved, these systems could support SKF's predictive maintenance work and broader sustainability goals more effectively, while also strengthening the company's competitive edge.

Keywords: Real-time monitoring, Predictive analytics, Machine learning, Remaining useful life estimation, Condition-based maintenance, Dashboard systems, Industrial oil filtration, Microsoft Fabric, Databricks, Sustainability

Acknowledgements

We would like to sincerely thank our Chalmers supervisor, Silvan Marti, for his valuable feedback, ideas, and continuous encouragement throughout the course of this project.

We also extend our gratitude to our industrial supervisor, Anirudha Das, for the opportunity to carry out this thesis at SKF and for his continuous support in this project.

Finally, we would like to thank all the team members and subject matter experts at SKF who supported us during this project. Their support, feedback and guidance have been instrumental in shaping and refining our thesis work.

Julius Johansson, Gothenburg, June 2025

Erik Roman Kilander, Gothenburg, June 2025

Contents

1
Introduction

1.1 Background

SKF is a global leader in manufacturing and supplying bearings, seals, and lubrication systems, focusing on enhancing the reliability and performance of rotating equipment. Serving industries such as automotive, aerospace, and industrial manufacturing, SKF provides solutions to improve efficiency, reduce downtime, and achieve sustainability goals.

SKF uses various fluids in operations like turning, forming, heat treatment, grinding, and washing. The SKF Fluid Management initiative focuses on the goal of keeping fluids clean longer and catch issues earlier, which can save costs and reduce waste. This will offer benefits in production, economic efficiency, and environmental impact.

Currently, fluid condition monitoring at SKF is performed via manual sampling and adjustments. This process is time-consuming, reactive, and prone to human error. Lab analyses may face delays due to workload. This master thesis hypothesizes that automating fluid management through real-time monitoring and predictive maintenance can improve fluid quality, and make fluid systems more efficient and environmentally friendly.

1.1.1 Industry 4.0

In the new era of industrial development, the capabilities and performance in terms of efficiency and maintenance has had much to gain from new technologies. Industry 4.0 has brought new technological advancements, such as Internet of Things(IoT) and artificial intelligence(AI), to industrial manufacturing (Lu, 2017). This is the continuation of the third industrial revolution and enhances the autonomous and intelligence aspects, amongst others. This is highly relevant since these capabilities are utilized in making factories smarter, which is seen in the field of maintenance (Lu, 2017). Maintenance is now more equipped to function in a proactive way using ML prediction capabilities. In the context of fluid management, this development is something to be leveraged; IoT and data analytics can be seen in the use of smart sensors and cloud computing in order to achieve an automated management of fluids using these data-driven decisions to get insights into system health.

1.1.2 Predictive maintenance

When it comes to predictive maintenance capabilities, these have emerged as a result of increased application of ML models and more computational possibilities in terms of managing large scale data in various formats. The benefit in the field of fluid management in particular is the ability to provide estimates for Remaining Useful Life (RUL) and causes of failures (Jardine et al., 2006). In fluid management, and this project specifically, these new technological advancements are used for monitoring fluid system parameters which can be indicative of the health of fluid cleaning systems.

1.1.3 Sustainability

Sustainability aspects like reduced waste, lowered need for large fluid usage, and overall energy efficiency improvements are all part of the Clean Manufacturing concept at SKF (SKF Group, 2021). The way it is approached in fluid management is to focus on working proactively so that fluids like oil can be kept clean for longer, and thereby extend the lifetime of said fluids (SKF Group, 2021). In addition to financial benefits, the ecological and environmental dimensions are just as relevant and important to consider as the positively impacted areas of optimized fluid management. By reducing both oil leakage into the environment, as well as the frequency with which the fluids need to be replaced, the hazardous waste and environmental strain is kept low. This way of working aligns with global sustainability goals and the principles of circular economy (Geissdoerfer et al., 2017).

1.1.4 Competitiveness

Today's manufacturers need to adopt modern technologies to stay competitive. This can be achieved by using predictive models to support data-based decisions. Using data-driven decision making and smart machines is becoming increasingly popular and manufacturers, like SKF, who are quick to adopt such tools are likely to be improving their operational efficiency and overall competitiveness (Lee et al., 2015).

1.2 Aim and research questions

The primary aim of this thesis project is to develop a standardized, data-driven fluid management system for SKF by integrating real-time sensor data (e.g., flow, pressure, and temperature) into a centralized GUI dashboard. This system will incorporate predictive maintenance capabilities through data analytics to predict fluid filter changes and detect abnormal behavior.

1.2.1 Research questions

This project aim to address the following research questions:

- How can a standardized GUI dashboard be designed to integrate and visualize real-time sensor data for fluid management?

- How can fluid filter changes be predicted by analysis of sensor data?
- What are the best practices for configuring and deploying the solution across SKF operations?

1.2.2 Scope and delimitations

The following delimitations define the scope and boundaries of this project:

- The project is confined to a sandbox environment and does not cover the deployment phase of the final solution.
- The project is constrained to a time frame of 22 weeks of full-time work.
- The focus is limited to specific oil filtration systems and the findings does not necessarily apply to other systems or sustainability applications within SKF or the manufacturing sector.

1.3 Structure of the Report

To help the reader follow the development of this project, the report is organized into six chapters. After this introduction, Chapter 2 presents the theoretical background, including key concepts in predictive maintenance, data science, and dashboard development. Chapter 3 describes the methodology used for data processing, modeling, and dashboard implementation. Chapter 4 presents the results from the analysis and evaluates the performance of the tools. Chapter 5 discusses these findings in relation to the initial goals and practical challenges that were encountered. Finally, Chapter 6 concludes the report and suggests possible suggestions for future work.

2
Theory

In this chapter, underlying theory for the methodology and results for this project will be presented.

2.1 Data Science

Data science is a multi-disciplinary field that combines statistics, computer science, and domain knowledge to identify patterns and trends in the data. By applying techniques like machine learning, it enables the analysis and prediction of patterns across various industries, supporting innovation and informed decision-making (Aves, 2025). In this project, data science was the primary field applied to support machine learning, which was a major component of the work.

2.1.1 Exploratory data analysis

The following chapter discusses Exploratory Data Analysis (EDA), a key process and starting point for any data science project. EDA is explained on a high level and its importance is emphasized, so as to give the reader an initial understanding of the concept in the project specific contexts in subsequent chapters.

2.1.1.1 Data driven vs hypothesis driven science

The concept of exploratory data analysis (EDA) is highly relevant in data science projects and constitutes one of the starting points when data is to be analyzed and worked on, or utilized with a scientific purpose (Skiena, 2017). This is becoming increasingly relevant due to the vast quantities of data that is generated at an ever increasing rate. As such, there is often a data-driven method in a scientific project that starts by studying given data and develops hypotheses and theories based on facts and observations. This is different from hypothesis driven science, where the method is to first develop a hypothesis which is then hopefully confirmed or rejected when data is studied and analyzed (Skiena, 2017).

2.1.1.2 Understanding data through EDA

To look at data and then find potential trends and patterns, or lack thereof, are main tasks done in exploratory data analysis(EDA). This applies to most cases where data science is leveraged, since the purpose of performing EDA is to gain an initial and high level understanding of a new dataset or multiple sets of data (Skiena,

2017). The starting points for this type of exploration may constitute addressing general questions about the nature of the dataset. These may be questions related to aspects such as, for instance the source of the data, the size and shape of the data, or whether the data points are categorical or numerical values (Skiena, 2017).

2.1.1.3 Summary statistics

Further steps to better understand data is to find the summary statistics for the data in question (Cooksey, 2020). Summary, or descriptive statistics is one part within the field of statistics that is used for description, as the name suggests. What this typically refers to, is to describe some characteristic of a large number of collected data points in a summarized format, which will then offer quick and concise insights into the dataset as a whole. Common characteristics that are often useful to find include statistical measurements such as average values, standard variations, percentiles, and peak values for a certain variable measurement (Cooksey, 2020). Such descriptive statistics are a good way to understand how the data behaves, as opposed to manually observing each individual data point and from that draw similar conclusions about the average values etc (Cooksey, 2020).

2.1.1.4 Data visualization techniques

In addition to descriptive statistics, another fundamental concept that facilitates exploration of data is to utilize visualization techniques (Skiena, 2017). The limiting aspects of using statistical measurements, and comparisons of said values in tabular formats are best addressed by means of combining them with graphical plots of a selection of the data. Graphs can show trends clearly without requiring users to study raw data. (Skiena, 2017).
Looking at the benefits of visualizing data in different ways, it is important to note that the results obtained depend not only upon the data quality itself, but also on what is displayed and not, in other words, what information the visualization clearly communicates.(Skiena, 2017). To exemplify, when plotting discrete time series values of a quantitative nature, this can be done using different types of charts. Barcharts are often used to plot aggregated data points in distinct intervals over time, while line charts generally aim to plot a variable over time as a continuous line (Skiena, 2017). If there is a need to interpolate values, which are then to be plotted, this may cause the visualization to contain values that are derived from the real values. While this is often an effective way to show trends for a variable, it should be considered as a general insight and not necessarily be taken as exact behavior. To ensure the graph is still displaying valid data, the initial data points from which interpolated values are generated should be included in the plot so that key values are still shown (Skiena, 2017).

2.1.1.5 Mange irregular time series data

To successfully and accurately study datasets from different sources, often in different formats, there may be a need to manage complex aspects such as that of having multiple datasets in different time series, with asynchronous relationships. Such ir-

regular time series are characterized by having large gaps between consecutive data points, and the length of the gaps also tend to be of differing sizes (Hartvigsen et al., 2022). This situation arises, for instance, when the data is coming from time discrete measurements, such as lab results, which may be combined with automatically generated data in real time. The problematic aspect of irregular time series is evident when datasets with two distinct timeseries should be compared or combined (Hartvigsen et al., 2022).
To manage the irregularity of having different time series, there are several methods for dealing with this so that the comparison between different data sets becomes possible. In general, the approach can be simplified somewhat when the goal is to perform initial exploration of the data. In these cases, the details or exact values are not necessarily crucial, as opposed to a model building task, where exact values might be more important (Hartvigsen et al., 2022). One approximation that allows a simple comparison is to aggregate data into bins of suitable sizes, which may be done, for example, by taking the average value for the selected time interval, and then letting this mean value represent the value for this entire interval (Hartvigsen et al., 2022). When two datasets are then plotted together, the aggregate values may be easier to plot, since data points may be allocated to the same points in time, even if one or more sets contain missing values as a result of the different number of data points in total between the sets.

2.1.1.6 EDA in practice

As previously established, one of the main purposes of performing EDAs is to quickly gain insights in the form of patterns and behaviors of the data. As far as the visualization goes, in the case of dealing with time series data, it is often useful to plot the data points over time as a starting point. This can give an overall view of how the values change over time, and if variable values are stable around their mean, or highly fluctuating etc. Additionally, it might be of interest to find and plot the trends of the data, which then reveals if a certain variable changes with time and if this change is consistently decreasing, increasing, or has a cyclic behavior.
One vital insight these early explorations may bring is to reveal whether the data sets as a whole actually align with the expected assumptions and prior knowledge of its source and thus what the data points represent (Skiena, 2017). Specifically, this could mean that relationships between variables are logically reflected in their statistics or in the visual representations. Together, statistical insights and visualizations can often complement each other (Skiena, 2017). For example, summary statistics allows for quantifying the correlation between two variables, and these relationships may similarly be shown by means of creating correlation matrices that reveal this potential correlation by plotting two variables in one graph.

2.1.2 Data preprocessing

In combination with general observation and exploring of data, there often arises a need to perform modifications of the dataset to get the desired formats and selection of meaningful information. This process is commonly referred to as data preprocessing, and involves cleaning, transformation, reduction and integration of

data (Han et al., 2011). Hence, this chapter describes these different dimensions of data preprocessing, and what impact they have on data analytics projects in terms of end results and overall success.
The need for preprocessing of data is related to data quality, which refers to whether or not the data is useful for its intended purpose and context in which it is used (Han et al., 2011). Preprocessing of data is thus often a necessary task that makes subsequent analytics steps possible.

2.1.2.1 Data cleaning

Preprocessing, as it pertains to the task of data cleaning, often addresses the commonly observed issue of having missing values in a large dataset (Han et al., 2011). This could involve a few missing records for a measured variable which is due to measurement errors, for instance. It could also mean having significant amounts of NaN values in a dataset with fields otherwise containing only numerical expressions. Another example of a situation which requires similar forms of data cleaning is when dealing with duplicate values, which may occur when combining multiple datasets (Han et al., 2011). Depending on why the erroneous values are present, and their criticality to the actual context at hand, they are treated in the preprocessing step so as to avoid undesired impact on the analysis.

2.1.2.2 Data integration

The common issue that prompts data integration tasks is the emerging inconsistencies in the process of combining multiple datasets (Han et al., 2011). To allow data from multiple sources or tables to be combined into a common set may be more or less trivial to achieve. Aspects to consider is how the data is structured, and if this allows immediate integration, or if there are certain steps to be done in order to achieve one single homogeneous data set (Han et al., 2011). Columns for the same variable may exist with different names in two separate data sets to be merged, and this must then be adjusted so that indexing will later make sense. The previously mentioned situation with multivariate irregular time series (Hartvigsen et al., 2022), and how to manage this situation when merging data sets is also a task of data integration (Han et al., 2011).

2.1.2.3 Dimensionality reduction

In practice it may not be necessary to work with the entire data set when performing further analysis, and to instead work with only a selection of the total number of records in the complete set. This is often the case when the initial data is vast in size, which may be time consuming to process and thus require unnecessary computation time (Han et al., 2011). Rather than merely cutting out a segment of the data, which may in itself cause unwanted problems, as one can imagine, the size may effectively be reduced using more deliberate approaches. Either the dimensionality is reduced, where certain features or variables redundant for the actual analysis are excluded (Han et al., 2011). The other approach is to reduce the number of records by performing aggregation with respect to the time, for example, and then using

aggregate values as representation for one time interval(Han et al., 2011).

2.1.2.4 Data transformation

The fourth dimension of preprocessing data prior to performing mining or other analytics tasks is that of transforming or consolidating data sets into the required formats to fit the mining process (Han et al., 2011). Transformation may be performed by means of aggregating data into larger time intervals, thereby raising the abstraction level (Han et al., 2011). It may also be of interest to adjust the range of the values that data points are allowed to take, such as keeping values within the range of 0 to 1. This concept is known as normalization of data, and the ranges are then reduced to the most optimal range with respect to further analytics or mining tasks (Han et al., 2011).

2.1.3 Machine learning

The descriptive and observational steps described above are all meant to produce a modified and refined version of the given data, so that the actual analytics processes can be done based on the data. Machine learning models are described in this chapter, where several different techniques, models, and their applications are discussed. There are models that have different forms of underlying logic, and they each apply to specific applications.
Machine learning (ML) on a conceptual level deals with the task of having a computer learn from example data, so that the model parameters are optimized (Alpaydin, 2021). This may be executed by first defining parameters that define the model, and then defining some variable as the target variable to be optimized (Alpaydin, 2021). Optimized means that either the ML model makes predictions on the variable values to be expected, or that it is more descriptive and historically focused, in nature.

2.1.3.1 Supervised learning

Two main distinct categories, for labeled data, of ML models are based on the task they perform, namely classification or regression models (Alpaydin, 2021).

2.1.3.1.1 Classification Classification in and of itself is concerned with the task of categorizing inputs into predefined classes (Alpaydin, 2021), which may be binary or multiple discrete classes. In the binary case, the input should be assigned one of two classes based on one or more conditions. The model is given what is called training data, where the input variable has labels, or assigned classes, that the model is then trained on, using suitable features (Alpaydin, 2021). The model then finds a set of rules that aligns with the training data, and which is then used in the next step, that of making the predictions. The model, or the rules the computer has found, which explains the data, is then subjected to new data sets, for which the label is not known (Alpaydin, 2021).
The task is for the model to assign the target variable its correct label (Alpaydin, 2021), that is, assign it the right class. Under the assumption that a good set

of features are chosen to enable the model to be trained, and the training data is representative for other unknown data as well, the model may be able to make predictions with some degree of accuracy (Alpaydin, 2021). The output from the model will then predict future variable values, and when comparing the actual output class, with the predicted classes, a measure of the model's accuracy may be found by comparing the number of correct predictions to the incorrect ones.

2.1.3.1.2 Regression The ML task is different if the target is a numerical value, the model should then predict this value, rather than by means of the features assign it a specific class or category. This type of problem falls under the category of regression problems (Alpaydin, 2021), and can be compared to the mathematical analytics case of finding a function that takes input values, which are the strategically chosen features, and produces the output with a certain level of precision. The difference in ML is that this function is not known, only a finite set of known in- and corresponding output values are known. For new input values, the model then tries to use its experience in learning from the training data, and interpolate an output value for what the output would be (Alpaydin, 2021).
Similarly to the classification problem, the model is first trained on a subset of data allocated for this purpose, where the input and output is labeled. The model evaluation then relates directly to how close the predicted values are to the true values, rather than, as in classification, compare the number of correct and incorrect predictions. The distance, or error, is the deviation from the true output to the predicted output, and gives a measurement of the performance of the model (Alpaydin, 2021).

2.1.3.1.3 Commonly used ML models There are a vast number of commonly used machine learning models and algorithms for equipment maintenance purposes. This chapter covers four such models to get a good representation of the different characteristics and strengths that may be obtained for this project, depending on the type of ML model selected.
Linear regression models are often beneficial to include in a general model performance comparison since its workings are easy to understand, and it is often used as a benchmarking model (Hastie et al., 2009). Similarly, both gradient boosting and random forest models are well established as being robust and effective for eliminating or minimizing errors due to overfitting or from only using single model outputs, as opposed to sequentially eliminating errors, which the gradient boosting methods are doing (Hastie et al., 2009). Prophet is another ML model that has predictive capabilities in the form of forecasting future unseen values. Prophet is an open source package that manages time series data formats effectively. Its strengths include the ability to effectively deal with time series data characteristics such as missing values, outliers and provide insights into trends and cycles on a weekly, or daily basis (Korstanje, 2021).
The models, linear regression, random forest, gradient boosting, and Prophet are described on a conceptual level to give a high level comparison.

2.1.3.1.4 Linear regression This model tries to find a relationship between input and target, where the input is made up of a selection of independent variables or

features, and the target variable is a dependent variable (Montgomery et al., 2012). As the name hints at, the model is based on the hypothesis that this relationship is linear, and that the dependent variable is a function of the independent variables. The correlation is thus expressed as a linear function where the coefficients for each input variable are found by training the model so that these coefficients may be estimated to a certain level of precision (Hastie et al., 2009). Linear regression models are easy to interpret and the computation times are relatively short compared to more complex model algorithms, but are not suited to solving problems that involve more complex relationships which can not be described using linear models (Hastie et al., 2009).

2.1.3.1.5 Random forest Random forest is based on the decision tree algorithm, which is an algorithm that uses branching and bounding, where the branches may be seen as paths down to a certain target class, or in the case of regression trees, a quantitative output value (Breiman, 2001). Random forest models utilize this principle, but take it one step further by combining multiple such models, which is called ensemble learning (Studer et al., 2011). This model uses a concept called bagging (bootstrap aggregating), and essentially deals with the issue of overfitting to improve the generalization of the model with respect to variance in data (Breiman, 2001).

2.1.3.1.6 Gradient boosting Another category within the field of machine learning models, gradient boosting machines (GBMs), are also ensemble models. In contrast to the random forest models, the GBMs are not utilizing combined output from several decision trees or other models, but rather improves its predictive accuracy by successively performing model training in sequence (Friedman, 2001). Common variants of GBMs are xgboost, catboost and adaboost.
Boosting models differ from random forest models in how the individual models are combined, or their outputs, more precisely. The distinction is in the parallel versus sequential approach, although both methods aim to improve the models performance in terms of how accurate the results are (Friedman, 2001). Since a sequential training of each model is applied for these types of ML models, the computation time is longer, and must be weighed against the pros, which include a possibility to manage errors made by the previous model which ultimately is used to minimize the errors in the predictions made in subsequent individual models (Friedman, 2001).

2.1.3.1.7 Prophet Prophet is an open source package, and having user-friendly functionalities to analyze time series data in an intuitive way makes it quite well suited to users who may lack extensive knowledge in data analytics or ML (Korstanje, 2021). Some of the main strengths of the model is its capability to handle missing data points, sudden trend changes, or outliers. As such, it offers business users and other non-data savvy people an efficient way to perform forecasting on time series data. The model is set up and the input and output variables are easy to define.
One particular useful function that the prophet model allows for is to visualize data components such as trends, cycles, and seasonality in data. This can be visualized

in separate plots for clarity, or it can be visualized in a combined plot so that actual values, trend lines, and outliers, for example, may be shown using plots with shared y and x axes (Korstanje, 2021). By means of these graphical presentations, analysts are able to draw conclusions and be more extensively informed in order to make decisions derived from these insights. Another useful option in prophet is that it is possible to customize the components of the time series data, for example by adjusting the frequency of detected changepoints to display in plots (Korstanje, 2021).

3
Methods

3.1 Methodological Framework

The project workflow follows the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, which remains the most widely adopted framework for structuring data mining initiatives. This choice helps structure the work into clear, manageable steps. CRISP-DM provides a structured and logical sequence of tasks, enabling effective planning and execution of project activities (Wirth & Hipp, 2002). The methodology is designed to support data mining processes across various domains by offering a flexible framework that can be adapted in greater detail as needed for each phase (Wirth & Hipp, 2002). In this context, the methodology guides the exploration and evaluation of data mining and analytical techniques best suited for deriving actionable insights (Han et al., 2011). A key objective is to identify forecasting methods that could support timely maintenance decisions.

CRISP-DM emphasizes that data mining is rarely a strictly linear process. Instead, it is often iterative, where earlier steps must be revisited based on insights or challenges encountered later in the process (Wirth & Hipp, 2002). The methodology breaks down the project lifecycle into six phases, each contributing to a coherent progression from business understanding to deployment. Before modeling or generating value through predictions, foundational work is required to ensure the project builds on a well-understood business context and reliable data.

The process begins with the development of a comprehensive understanding of the business problem. This involves assessing the current situation and clearly defining the objectives of the project. In parallel, the data is explored to gain a preliminary understanding of its content, structure, and relevance. This stage typically includes initial exploratory data analysis (EDA) to evaluate data quality and determine how well the available data supports the business case (Han et al., 2011; Wirth & Hipp, 2002).

The following phase, data preparation, builds upon the insights gained during the data understanding stage. Here, the data is cleaned and transformed into a format suitable for modeling (Wirth & Hipp, 2002). Tasks may include handling missing values, restructuring data tables, and creating derived variables. It is important to note that the combined efforts of understanding and preparing the data often represent a significant portion of the total project time. As such, the project timeline must account for this and avoid assuming that modeling can begin immediately.

Once the data is adequately prepared, the focus shifts to model selection and devel-

opment, followed by evaluation (Wirth & Hipp, 2002). These phases often require multiple iterations, as suboptimal results may reveal the need to revisit earlier stages. For example, weak model performance might indicate gaps in the original business understanding or the need for further data refinement. Thus, although the CRISP-DM framework is often presented sequentially, it is inherently iterative and adaptive to the evolving needs of the project.

The final phase is deployment, where the results are delivered in a usable form for the end user. Depending on the project scope and stakeholder needs, deployment may range from the integration of predictive models into operational systems to the presentation of results in the form of detailed reports. The complexity of deployment depends largely on the business requirements and the intended audience.

Following the implementation of the dashboard and its core functionality, the project addresses a final research question: the scalability of the proposed solution across SKF's global operations. This step involves additional literature review and interviews with subject matter experts (SMEs) to gain insight into the operational differences across various SKF plants. Based on these findings, the solution is re-evaluated to identify best practices for deployment on a broader scale and to ensure its adaptability to different factory contexts.

3.2 Exploratory Data Analysis

This chapter describes the methodology applied for exploratory data analysis (EDA) and explains the rationale behind each step. The purpose of EDA is to establish a foundational understanding of the dataset prior to feature engineering and machine learning (Cooksey, 2020; Han et al., 2011). This includes assessing data quality and identifying key patterns, trends, or anomalies that may influence subsequent stages of the project. EDA was performed using Python, employing a range of libraries for data processing and visualization. The primary packages used included:

- Pandas (for data manipulation and statistical summarization)
- Plotly (for creating interactive and high-quality visualizations)

Additional libraries, such as Numpy, Scipy, and Statsmodels, were also used for deriving additional statistical measures.

3.2.1 Data Understanding and Preparation

The first step in the process of exploratory data analysis involves getting a basic understanding of the dataset and performing basic data preparation. This step includes the following tasks:

- Identifying the number of records
- Understanding data types (e.g., numerical, categorical, datetime)
- Handling missing, duplicate or invalid records
- Converting data to a suitable format (i.e. date to datetime)
- Aligning data from multiple sources (sensor data with lab measurements)
- Standardizing measurement units and scales

- Removing unused or redundant columns
- Computing derived columns, such as total oil cleaned and time intervals between measurements

This step was crucial for providing a foundation for future statistical analysis, visualisations and feature engineering. It also ensured compatibility between different datasets (sensor data with lab measurements). After this initial step, the dataset was correctly structured and cleaned, to make it more suitable for statistical analysis and visualisation.

3.2.2 Descriptive Statistics

Once the data was cleaned, both basic and more advanced descriptive statistics were computed to gain a deeper understanding of each sensor's behavior (Cooksey, 2020; Montgomery et al., 2012). The analysis consisted of calculating statistical measurements including:

- Count (number of observations)
- Mean (average value)
- Standard deviation (variability)
- Minimum and maximum values
- Quartiles (25th, 50th/median, and 75th percentiles)
- Stationarity tests (e.g., Augmented Dickey-Fuller) to assess temporal consistency
- Signal-to-noise ratio (SNR) calculations to evaluate data quality and reliability (Hastie et al., 2009)

These metrics helped identify outliers, inconsistencies, and trends, and informed further decisions regarding feature engineering, data transformation, and modeling strategies.

3.2.3 Visualisations

In addition to the summary statistics, visualization is another key component of EDA, which helps interpret patterns that would be hard to see in tables (Cooksey, 2020). Multiple types of plots were generated to assess sensor signal behavior over time and in relation to target variables. They include the following:

- Raw time-series plots to show unfiltered sensor data
- Smoothed time-series plots using the Hodrick-Prescott (HP) filter to separate trend and noise
- Boxplots to visualize distributions and detect outliers
- Kernel Density Estimation (KDE) plots to show value distributions
- Composite plots overlaying all sensor signals (with normalization as needed) alongside key target variables such as MPC and particle counts

These visualizations helped to:

- Identify temporal trends and seasonality
- Assess correlations between sensors and target variables

- Detect anomalies or sudden shifts in behavior
- Support statistical findings with visual evidence

This step provided valuable insights and data understanding as preparation for performing effective feature engineering and model development. The valuable insights refer to a solid understanding of the distribution and behavior of the data over time, and potential dependencies between sensor readings and lab measurements.

3.3 Machine Learning

In this project, two main machine learning approaches were used, where the distinction lies in the objective. The first focuses on estimating the Remaining Useful Life (RUL) of the filter by predicting the time remaining until filter change is necessary. This involves restructuring the raw data into a format suitable for supervised learning. The modeling was carried out in Python using the Scikit-learn and XGBoost libraries.

The machine learning models used for this task were Random Forest Regression, Extreme Gradient Boosting (XGBoost), and Support Vector Regression (SVR). These models were chosen for their robustness, ability to handle nonlinear relationships, and proven effectiveness in regression problems. Although additional models could have been explored, the delimitation of exploring the above three models was set due to time and computational constraints. The assumption here is that if none of the selected models are able to achieve reasonable performance, it is unlikely that alternative models would yield significantly better results when subjected to the same or similar data.

As this project does not cover deployment of the solution, instead the goal is exploratory: to assess to what extent current sensor data can predict the remaining useful life of the filter. The prediction is framed as a regression task, which considers four key metrics as targets: MPC and particle counts at 4µm, 6µm, and 14µm. Since filter replacement decisions depend on both the absolute values and the trends of these indicators, an effective regression model must accurately predict not just the values, but also the temporal shape of these signals.

The second approach in the machine learning component of this project focuses on forecasting the sensor signals, which means treating them as target variables to predict. This is intended to provide insight into how the sensor readings change over time and to detect potential trends or anomalies. When such analytics are implemented in the dashboard, the forecasts can be used to trigger alerts if sensor values are projected to exceed or fall below predefined thresholds. This can then effectively provide an early warning system, an important tool to enhance proactive rather than reactive forms of system maintenance. The forecasting task also acts as a supplement to the RUL estimation, to support the broader goal of enabling predictive analytics and help users decide when maintenance is needed.

3.3.1 Feature Engineering

Following the exploratory data analysis, the next project step was feature engineering, carried out to prepare the dataset for modeling. Feature engineering involves identifying, creating, and selecting appropriate features that can effectively capture patterns and relationships within the data relevant to the target variables. This was conducted in parallel with initial model testing to evaluate the predictive power of various features. This was an iterative process, where different types of features were derived and evaluated using baseline, untuned models. The goal was to assess how well these features performed in relation to the target variables, meaning how effective the feature is as ml input information given the objective of predicting the current target variables. This iterative process was performed and refined the feature set accordingly.

Important to note is that this project is limited by a shortage of data, with only four filter-life cycles worth of historical batch data available. Furthermore, three of these cycles were deemed unsuitable for analysis due to atypical conditions: one involved a premature filter change, another contained data from a system used to clean hydraulic oil instead of the intended cutting fluid, and the third came from a system which used a special type of filter with different performance requirements. As a result, only one cycle of data from one system is considered in this project for exploring whether sensor data can accurately predict MPC and particle counts.

Since the target variables, i.e. MPC and particle count, were provided as daily values, it was necessary to aggregate the sensor data to the same time resolution. The point is to merge the data sets, sensor data with lab measurements, which involved this aggregation in order to manage the irregular timeseries problem. This aggregation also allowed the computation of multiple daily aggregated features per sensor. These aggregations were made to be able to extract meaningful patterns and relationships between the sensor data and the target variables. As an initial step in the daily feature engineering, for each sensor, a variety of statistical measurements were computed for each day in the cycle. These aggregated features were intended to summarize the daily behavior of each sensor and provide the model with a comprehensive representation of the system's state on a per-day basis:

- Mean
- Median
- Mode
- Minimum
- Maximum
- Standard deviation
- Variance
- Quantiles (25% and 75%)
- Skewness
- Kurtosis

In addition to sensor aggregations, several domain-specific features were engineered. These features were included to capture operational conditions that could influence the target variables:

- Total oil cleaned (provided in the data)
- Oil cleaned (computed as the difference in the total oil column between consecutive time stamps)
- Time elapsed (in days)
- Production pause indicator (a binary feature where 1 represents a production pause, 0 otherwise)

Finally, temporal features were derived to account for patterns in the data that concerns the time of occurrence and how it relates to the sensor readings. These include: lagged and rolling window features, specifically rolling mean and daily mean as a lag feature. These features were introduced with the intent of capturing both trends and delayed effects that might influence the targets, which ultimately aims to improve the model's ability to recognize temporal dependencies:

- Rolling features (mean)
- Lag features (mean)

After these features had been created, the combined comprehensive set of features, including daily aggregates, domain-specific and temporal features, was used as input to the model to see the results. Then, a feature selection process using Scikit-learn's SelectKBest method was applied to identify the most informative features based on the one-way ANOVA F-test value. This helped reduce dimensionality and eliminate redundant or non-informative features.

Despite experimenting with various feature combinations and applying feature selection techniques to identify the most informative variables, no specific feature or set of features demonstrated a strong or consistent predictive relationship with the target variables. Given the limited size of the dataset, some degree of overfitting might have been expected during model training. However, even this was not observed, regardless of the features used. This suggests that the features, based on sensor data, is likely a poor proxy for accurately modeling the target outputs. In the end, the following feature set was selected for more extensive modeling and hyperparameter tuning:

- **Daily mean values** for all sensors, to capture the overall daily behavior of each signal.
- **Rolling mean values** over the past seven days for each sensor, to incorporate recent trends and smooth out short-term fluctuations.
- **Lag features** representing the sensor values from each of the previous seven days, to account for potential delayed effects and temporal dependencies.
- **Domain-specific features** representing operational conditions that could influence the target variables

This feature set was configured to balance simplicity with temporal consideration, to ensure that the model had access to both current and historical data for each sensor variable.

3.3.2 Predictors of remaining useful life

Predicting the targets from the available sensor data was framed as a time-series forecasting task. The goal was not to estimate the current state of the system, but to forecast values of key target variables, namely MPC and particle counts. Predicting these variables was done so that the forecasts as such would then provide enough lead time to anticipate and prepare for a filter change. This forward-looking technique aligns with the broader objective of predictive maintenance.

In practice, the machine learning method involved using a direct multi-step forecasting strategy. This technique refers to the process of having a separate model trained for each forecast horizon, which means one model for predicting the target one day ahead, another for two days ahead, and so on. The maximum forecast horizon was set to 7 days, to match one full week's worth of advance notice. Consequently, seven individual models were trained for each week long window of data, where each model corresponds to a specific day in the forecast window. For each model, the target variable was shifted upward in time to align future values of the target with current and past sensor data. This setup allows the models to learn from patterns in the sensor readings together with tomorrows target value and forecast future outcomes.

3.3.2.1 Alternative Target Considerations

Another modeling approach that was briefly considered involved predicting other metrics that could be used to assess the filter condition, instead of relying on the laboratory data, specifically:

- **Days Running:** How many days the filter has been active.
- **Total Oil Cleaned:** The cumulative volume of oil processed by the filter.

The decision not to model these targets was based on two key considerations. First, these metrics are not directly linked to the actual operating conditions of the system and do not accurately reflect fluid condition. Contamination levels in the fluid are primarily influenced by the machine's operational load, e.g. when it is actively working on products, which is the source of the contaminants in the fluid. While these aspects may correlate with days running or total oil cleaned in actual, there is no direct causality between these metrics.

Second, both of these alternative targets have inherent limitations. A countdown of "days remaining" would typically assume a linear degradation rate, which does not hold true in practice, as machines do not operate continuously. Similarly, the "total oil cleaned" continues to accumulate even during machine idle periods or operational pauses, since the oil filtration system is always running, regardless if the machine that is it connected to is active or idle. This means that the metric would increase no matter if new contaminants are being introduced or not, which makes it an unreliable proxy for actual oil condition. As a result, these alternative approaches to the filter RUL estimation were not adopted in this project. However, these metrics could potentially form the basis of simpler, schedule-based maintenance protocols, an idea discussed later in Chapter 5 (Discussion).

3.3.3 Model Validation

To evaluate model performance in a way that reflects real-world deployment, two validation techniques were employed: *Visual inspection* and *time-series cross validation* (Refaeilzadeh et al., 2009).

Visual Inspection
Since the decision to change a filter is influenced not only by absolute values but also by the trend of the data, visual inspection showed whether the model correctly predicted filter changes. By examining the predicted versus actual values over time, one can assess whether the model captures the relevant patterns that would inform real-time decision making.

Time-Series Cross Validation

In addition to visual inspection, time series cross-validation was used to quantitatively evaluate performance (Refaeilzadeh et al., 2009). Two methods were applied: the *sliding window* and the *expanding window* approaches, each simulating different deployment scenarios.

- **Sliding Window:** The model is trained on a fixed-size rolling window of past data and used to predict future values. This approach is suited for dynamic environments where recent data is most indicative of future behavior.
- **Expanding Window:** The training set grows over time by continuously adding new data, using the entire historical record up to the current point. This method is beneficial when long-term trends and accumulated information are valuable for accurate prediction.

To quantitatively assess model performance during time series validation, a selection of commonly used evaluation metrics was used. These metrics provide a comprehensive overview of prediction accuracy, robustness, and overall fit (Chicco et al., 2021; Montgomery et al., 2012).

- **Mean Squared Error (MSE):** MSE calculates the average of squared differences between the predicted and actual values. It penalizes larger errors more heavily, making it sensitive to outliers (Chicco et al., 2021):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE):** RMSE is the square root of MSE, providing an error measure in the same unit as the target variable (Chicco et al., 2021; Montgomery et al., 2012):

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

- **Mean Absolute Percentage Error (MAPE):** MAPE expresses prediction accuracy as a percentage. It can become large if actual values are small (Chicco et al., 2021):

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- **Symmetric Mean Absolute Percentage Error (sMAPE):** sMAPE normalizes the absolute error by the average of the actual and predicted values, which helps mitigate issues with small denominators (Chicco et al., 2021):

$$\text{sMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

- **Coefficient of Determination (R^2 Score):** The R^2 score measures the proportion of variance explained by the model. It ranges from negative infinity to 1, where 1 indicates a perfect fit (Chicco et al., 2021; Montgomery et al., 2012):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

In terms of statistical metrics as well as visual inspection, evaluating model performance requires a mixed approach. Quantitative metrics such as MSE, RMSE, MAPE, sMAPE, and R^2 provide objective measures of accuracy and error, while visual inspection adds important context by revealing how well the model captures trends and patterns in the data (Chicco et al., 2021).

3.3.4 Changepoint Prediction

In addition to using sliding and expanding window validation, another prediction validation approach called "changepoint prediction" was used. This method involves training the model using an expanding window up to a series of predefined changepoints, in this case specific dates when laboratory measurements were taken in order to analyze the models ability in detecting its trend immediately afterward, either upward or downward. The core idea is to evaluate the model's ability to anticipate a change in the trend of the target variables without having seen the actual inflection. By training the model up to just before each such changepoint, it is possible to assess whether the model is capable of anticipating the imminent shift in trend without training on data after the actual inflection.

3.3.5 Sensor Forecasting

The second component of this project focuses on forecasting sensor values. The primary objective here is not to predict precise absolute values, but rather to provide an indication of the system's overall behavior by means of forecasting sensor trends over time. This method would provide predictive insights into the behavior of key metrics, e.g. pressure, flow, temperature, on a continuous basis, so as to give users a clearer picture of overall sensor trends, and whether they are increasing, decreasing, or remaining stable.

The primary objective of this feature is to improve situational awareness by allowing operators to monitor the direction of sensor values over time. This becomes particularly valuable when used in combination with threshold-based alerts, since many sensors are tied to specific operational limits. Forecasting these trends enables the system to anticipate potential operational problems and generate early warnings,

which would allow appropriate personnel to take preventive action before conditions deteriorate.

To achieve this, the Prophet model, a time series forecasting model developed by Facebook, was used. Prophet is particularly suitable for this task because it decomposes time series data into trend, seasonality, and residual components, so it is thus effective for capturing behavioral patterns. The model is fed raw sensor data in a simplified format: it only requires two columns, one for timestamps and one for the corresponding sensor values. This minimal input format simplifies pre-processing and allows the model to focus entirely on temporal patterns. As such, one model per sensor was trained and validated on, in this project. For the dashboard side of things, Prophet was also included. In this case, it provides trend forecasting for the sensors of two oil filtration systems that are currently connected to the API.

3.4 Data Architecture

In this project, two data platforms were utilized: **Microsoft Fabric** and **Databricks**. In this chapter, the data ingestion, storage, and processing workflows will be presented for both platforms, to highlight how data were handled in each case in preparation for machine learning and dashboard development.

Microsoft Fabric is a unified cloud-based platform that integrates data engineering, data storage, real-time analytics, and more, into a single environment. As it is currently under internal evaluation at SKF, one of the key considerations of this project was to assess its suitability for dashboard development, when taking into account all steps of data management and machine learning modeling.

Databricks, on the other hand, is a data analytics platform that is already in use within SKF. Unlike Microsoft Fabric, it does not come with integrated data storage capabilities. Since Databricks is already in use at SKF, its inclusion in this project served two important roles: One, it was included to verify that the dashboard and analytics could be implemented using existing tools. Secondly, it acts as a benchmark for comparing different aspects, such as performance and scalability, with Microsoft Fabric in this context of building a dashboard with machine learning analytics.

3.4.1 Data Source

The data used in this project is sourced from an API, which is the primary interface for retrieving real-time operational data. This is not the same data that the machine learning model were trained on, as that is based on older, historical readings. In contrast to historical data files of previous cycles, the API data includes a range of information: timestamps, sensor values, upper and lower thresholds for each sensor, and other operational metrics such as the number of days the filter has been active, cumulative volume of oil cleaned, identification numbers, alarm logs, etc. In the current set-up, three oil filtration units are connected to the API. However, only two of these units are actively and continuously updating their data. The

API updates with new information roughly every hour, which creates a delay of about one hour. Because of the scope of this project, which focuses more on RUL estimation and long-term sensor forecasts, this delay in readings is not considered a critical limitation, although it is something that should be considered if a real-time alarm system is a priority, discussed later in Chapter 5 (discussion). Depending on the platform used (Microsoft Fabric or Databricks), data retrieval takes place every 7 and 30 minutes, respectively. As a result, some duplicate records are present in the raw data that is stored and must be cleaned prior to visualization and analytics.

3.4.2 Microsoft Fabric

The primary platform used in this project is Microsoft Fabric, a unified, end-to-end, cloud-based data platform that integrates various aspects for data management, such as data engineering, data warehousing, and real-time analytics within a unified environment. Although Databricks was also utilized as an alternative tool in this project, Microsoft Fabric served as the main platform for this project. Currently, Microsoft Fabric is under internal evaluation at SKF, and its inclusion in this project fulfills a dual purpose. Firstly, it provides the foundational infrastructure and tools to carry out the project workflow. Secondly, this project acts as a practical case study to assess Microsoft Fabric's capabilities and suitability for similar use cases involving real-time data integration and machine learning workflows within SKF.

The ingestion of data by a fixed interval (e.g., every few minutes) occurs via an API. This raw data from the sensors is ingested and stored in a bronze table in the KQL database, specifically in an Eventhouse in Microsoft Fabric. This means that no external data storage is required; instead the Eventhouse is part of the Microsoft Fabric ecosystem, and so data stored there can be queried directly in an effective manner. This option can be compared to Databricks, where data warehousing takes place in an external database, which may have implications in terms of query latency or similar performance criteria.

Materialized views are used to compare or filter files for real-time performance optimization purposes. In Microsoft Fabric, what is called a materialized view is a way to store the results of a query from a queryset so that the data is readily available without having to run the entire query each time new data is ingested or otherwise updated. This function is especially useful when working with large streaming datasets where such calculations that a query performs can become time-consuming and resource-intensive if it must be re-executed manually. Compared to a regular queryset where the querying tasks are defined, a materialized view stores the result of the query. It is then updated automatically as the input data is renewed, which is a key aspect when working with real-time analytics and, in extension, interactive dashboards.

A materialized view was created to provide the necessary cleaned and structured data layer (referred to as the silver layer, or silver formatted data table). This silver table is the restructured and filtered version of the ingested raw data (referred to as the bronze layer) and is used for downstream analysis like machine learning modeling and output visualization. The silver layer simplifies the initial format of the data

by removing invalid or missing values, and renaming columns to make the names interpretable and clear. This makes the data more reliable and easier to work with when it comes to modeling and visualization.

In Microsoft Fabric, a pipeline is needed in order to be able to make the Queryset run automatically on a schedule. The pipeline is used to run the Queryset, and then to write the output into a newly created table, or alternatively a materialized view. Without these pipelines, the transformations would not be automated. This also means that the predictions made by the ML model are scheduled to be updated at regular intervals. The predicted forecasts are stored in a new table in the KQL database, as well as a table for historical data. Both tables are updated periodically (e. g., hourly) so that downstream applications always have the latest insights to work with.

KQL Querysets are used to achieve the necessary preprocessing (e. g., removing negative or invalid data points, duplicates, or calculated columns which are derived from initial columns (i. e., delta oil cleaned)). This way, only the relevant data is placed in the silver layer, and so that it also gets the right format with regard to naming of columns. This is in line with best practices in stream processing pipelines to obtain good reliability and results.

A Microsoft Fabric pipeline orchestrates and automates data movement and transformation from the bronze layer to a cleaned-up silver-format table. In this case, the data pipeline performs simple preprocessing tasks such as standardizing timestamps to datetime formats, removing missing values (called NaN in programming contexts), and removing irrelevant records (e.g., deleting rows where variable values fall below prespecified thresholds). This structured transformation aligns with the preprocessing step in the CRISP-DM methodology, which emphasizes data cleaning and preparation as essential for successful analysis projects(Wirth & Hipp, 2002),(Han et al., 2011).

Using Microsoft Fabric Notebooks, additional features are created from the silver table to enhance the predictive capacity of machine learning models. These include: Rolling averages or windowed statistics to follow local trends, time-based consecutive data point comparisons(delta oil cleaned), sudden changes in observations, and categorical encodings for operational states(e. g., production pauses). All of these engineered features are a large and significant part of the process of developing robust and accurate predictive models (Skiena, 2017), (Hartvigsen et al., 2022).

In Microsoft Fabric, two types of dashboard tools were utilized for this project: *Power BI* and *Real-time dashboard*. Power BI is a well-established business intelligence application that now offers seamless integration with Microsoft Fabric. It allows a direct connection to data within Microsoft Fabric and can be hosted as dashboard items on the platform. The Real-time dashboard, on the other hand, is exclusive to Microsoft Fabric and designed specifically for real-time data streaming. It uses data from KQL tables hosted on Microsoft Fabric workspace for querying data and is optimized for fast streaming data queries. In this project, both dashboard options were explored and evaluated.

Power BI connects directly to the silver layer and the machine learning results table

hosted on Microsoft Fabric via DirectQuery, something which enables real-time data refresh. The dashboard was developed within the Power BI desktop application and subsequently published to the Microsoft Fabric cloud environment for access and sharing of the generated reports and dashboard.

The Real-time dashboard is hosted natively within the Microsoft Fabric workspace and also connects to both the silver layer and the machine learning results table. It queries data dynamically using various dashboard elements to provide live updates and interactive visualizations tailored for real-time monitoring. This querying uses the KQL language, and as such the creation of tiles and other dashboard elements are generated with ease compared to the cumbersome process when Power BI is used.

3.4.3 Databricks

The other platform used in this project is Databricks, a cloud-based data analytics platform built on Apache Spark, designed for large-scale data processing, machine learning, and collaborative data science. In this project, Databricks was included primarily to assess the feasibility of implementing the solution using SKF's existing tools and to provide a baseline for comparison against Microsoft Fabric. Due to its later integration into the project and this comparative focus, the implementation details of Databricks are less extensive. In this case it may be noted that data storage and processing is relatively limited.

Data ingestion is handled through scheduled notebook runs. The notebook uses the request library, as well as API keys, to access and store the raw data in a predefined table. This notebook runs every 30 minutes, and with the API itself updating every hour, this means that there are going to be duplicate records that need to be handled.

Data ingestion in Databricks is managed through scheduled notebook executions. These notebooks utilize the Python `requests` library, and with API keys, to retrieve raw data from the source API and to store in a predefined table. The notebooks run every 30 minutes, while the API updates data approximately every hour. This mismatch in update frequency results in duplicate records, which require subsequent handling during data processing.

Due to the absence of dedicated processed data storage or integrated machine learning workflows, data processing in Databricks is quite limited. Tasks such as removing duplicate records and renaming columns are performed within Databricks' dashboard querying feature rather than in separate transformation pipelines.

Databricks' built-in dashboard capabilities are used to visualize and analyze the data by querying raw data directly from the raw data stored in ADLS using SQL. The dashboard features an overview panel that provides a historical summary of alerts, along with a detailed view of sensor readings, the ability to filter between different oil filtration systems and displays various metadata, including the last data update, total oil cleaned, last oil filter change, etc.

3.5 Dashboard Development

In this section, the methodology for dashboard development will be presented. In the end, three dashboards were created. Two in Microsoft Fabric using Power BI and Real-time Dashboard, and the third in Databricks. This multi platform approach was chosen to evaluate various dashboarding alternatives, with Databricks serving as a baseline point of comparison as it is currently in use at SKF.

As part of the project methodology, a structured process was followed to develop the dashboards. Several stages of prototype development were gone through prior to the final implementation, with each serving specific purposes. These stages facilitated early feedback collection, supported idea generation, and informed later design choices within the different tools. See below for the lists of dashboard development steps:

- **Requirement Specification**
  The first step was to list the dashboard's expected features and design elements. This helped define what was needed and made it easier to get feedback from stakeholders early on. This list was mainly generated through brainstorming sessions and discussions with project stakeholders and subject matter experts(SMEs).
- **Graphical Dashboard Mockup**
  A rough sketch of the dashboard was created to show layout ideas and possible components. This helped align expectations and generate further feedback.
- **Prototype with Historical Data (Power BI)** A working prototype was built using historical sensor data in Power BI. It was used to test if the design ideas were practical and to try out layouts, metrics, and slicers in one of the tools used in this project.
- **Prototype with Real-Time Data (Power BI)**
  Another prototype was made using real-time data from a KQL database in Microsoft Fabric. This was primarily done to evaluate how real-time data integration works in a continuously updating dashboard.
- **Final Dashboard Implementation**
  The final dashboards were built using Power BI, Microsoft Fabric Real-Time Dashboard, and Azure Databricks. They included lessons learned from the earlier versions. Forecasting was not added to the Databricks version because it was introduced later, and the predictions were not needed for comparisons sake anyway. However, this capability can be added easily in the future if needed.

Each of these stages builds progressively upon the previous stage and allows for the integration feedback from SKF to ensure better alignment with user needs and technical feasibility.

3.5.1 List of Requirements

The initial phase involved collecting and refining a comprehensive set of requirements that would define the scope and functionality of the dashboards. This list was

generated through a combination of brainstorming sessions, consulting with SMEs at SKF, and analyzing dashboards from other internal projects. The purpose was to capture all relevant ideas that could enhance dashboard functionality within the project's context. Once drafted, the list was reviewed and iteratively refined based on feedback from key stakeholders. This process remained dynamic, with continued updates and validation even as subsequent development steps commenced.

3.5.2 Graphical Prototype

Following the requirements gathering, a visual prototype was created to conceptualize the dashboard layout and appearance. This prototype aimed to translate abstract ideas into a tangible interface, allowing stakeholders to visually assess and critique the proposed structure. Canva, a user-friendly graphic design tool, was employed for this task, to enable rapid assembly of mockups that included overview panels and detailed component views. These visualizations were then presented to SKF experts for feedback. Their input led to several design refinements, including adjustments to chart types, navigation structure, and visual clarity. The prototype functioned as an essential bridge between conceptual planning and technical implementation.

3.5.3 Historical-Data Prototype (Power BI)

With a visual direction established, a functional prototype was developed using historical data in Power BI. This prototype allowed for the practical testing of layout concepts and the exploration of technical capabilities within the Microsoft Fabric environment. Historical sensor data, previously used in the project's machine learning models, served as the input. This approach avoided the complexities of real-time data processing while enabling effective evaluation of the dashboard's responsiveness, visual interactivity, and data modeling capabilities. During this phase, particular focus was placed on optimizing Power BI's data ingestion processes, designing interactive visuals aligned with the graphical prototype, and validating analytical computations. The historical data prototype served as a sandbox for refining visual elements and creating calculated metrics using the DAX syntax used in Power BI.

3.5.4 Real-Time-Data Prototype (Power BI)

The next stage involved integrating real-time data into Power BI to test its suitability for live monitoring scenarios. The objective was to replicate the functionality of the historical data prototype, but now using live data inputs by means of streaming capabilities. This included configuring data pipelines through Azure Event Hubs and Stream Analytics, connecting them to Power BI, and setting up live visuals with sub-minute refresh rates. This prototype helped identify challenges specific to real-time applications, such as refresh limitations, data latency, and functional constraints in DirectQuery mode. Although the machine learning components were not included in this stage, the prototype played a crucial role in understanding the technical demands and operational readiness required for a live data environment.

3.5.5 Final Dashboards

In the final stage, three dashboards with all features were developed and deployed: a *Power BI* dashboard, a *Real-time dashboard*, and a *Databricks* dashboard. Each had a design of a consistent two-page layout. The first page provided a high-level overview featuring key performance indicators, alarm summaries, and maintenance scheduling data. The second page allowed for detailed examination of individual oil filtration systems, including time series visualizations of sensor data and event history. These dashboards were built with scalability, usability, and maintainability in mind. Emphasis was placed on standardizing visual components, ensuring secure access through role-based authentication, and configuring automated data refresh mechanisms. The design and implementation of the final dashboards were based on all prior phases, with the end result being robust, user-friendly solutions suitable for SKF's operational environment.

3.6 Scalability and Deployment

After the dashboard and its functionality are finalized, the remaining research question is addressed. This involves analyzing the scalability of the solution and its broader applicability across multiple SKF factories. The process includes reviewing results and findings, and engaging in discussions with subject matter experts at SKF to gain a deeper understanding of the operational characteristics of SKF's global plants. Based on these insights, the potential solutions are reevaluated to identify best practices for deployment and implementation on a global scale.

3.7 Use of AI tools

Part of the thesis writing process, ChatGPT was used as a language support tool to help enhance phrasing and clarity. Specifically, the authors wrote content in bullet points or draft prose, which was then refined using the ChatGPTs rewriting features. This was carried out primarily for tone improvement, structure, and grammar. Use of the tool was only for the purposes of rewording and refining the authors' own writing style. No content, interpretations, or ideas were generated by ChatGPT.

The use of AI was limited to polishing the written text and was not used for any part of analysis, design decision, data interpretation, methodology structuring, or discussion of results. For example, AI tools were not utilized for building predictive models, selecting machine learning techniques, or comparing platform performance. The authors developed all such decisions and analyses on their own. The authors reviewed all AI-assisted text, and edited or rewrote it as necessary to preserve intended meaning for the project. Final editorial judgment was always reserved for the authors, who are solely responsible for the content and conclusions of this thesis.

No confidential SKF information was used as input to ChatGPT or other AI tools. The use of AI tools was conducted in accordance with ethical academic conduct and

the demands by Chalmers University of Technology for ethical use of AI in thesis work.

4
Results

This chapter highlights all the results and findings from the project, which includes the implementation and evaluation of the data architecture, machine learning modeling, and dashboard development. The aim of this chapter is to in detail provide a comprehensive overview of how each key component of the project was realized during this project.

4.1 Data Architecture

In this section of the results, details regarding the data architecture implementation for each of the two two platforms evaluated in the project: Microsoft Fabric and Databricks will be presented. It covers and compares how each tool handles critical aspects, such as: data ingestion, transformation, modeling, and storage. The comparison highlights the strengths and limitations of each platform in supporting the project and its objectives. For more detailed comparison and discussion, see Chapter 5.

4.1.1 Microsoft Fabric

This project was primarily performed within Microsoft Fabric, which served as the platform for data ingestion, transformation, storage, and dashboard visualization. This section describes how data management within Microsoft Fabric. For an overview of the Microsoft Fabric data model, see Figure 4.1
Data ingestion is handled through Microsoft Fabric's **Data Pipeline** item, which allows connections to various data sources. In this case, the data pipeline is connected to an external API, with two oil filtration systems connected to the API, updating approximately every hour. The pipeline is configured to run every seven minutes, retrieving data and sending it to an Eventhouse item inside Microsoft Fabric, where the raw data is stored in a KQL database. Since the pipeline updates more often than the API, the data include duplicate records that are handled in the transformation stage.

The data storage follows the medallion architecture, which organizes data into bronze, silver, and gold layers. Initially, raw and possibly duplicate data is stored in the bronze layer, in this case a KQL table. A materialized view is then created to form the silver layer, where key cleaning steps are applied: duplicates and invalid entries (i.e., negative values) are filtered out, column names are clarified, and calculated fields, such as oil cleaned, are derived too.

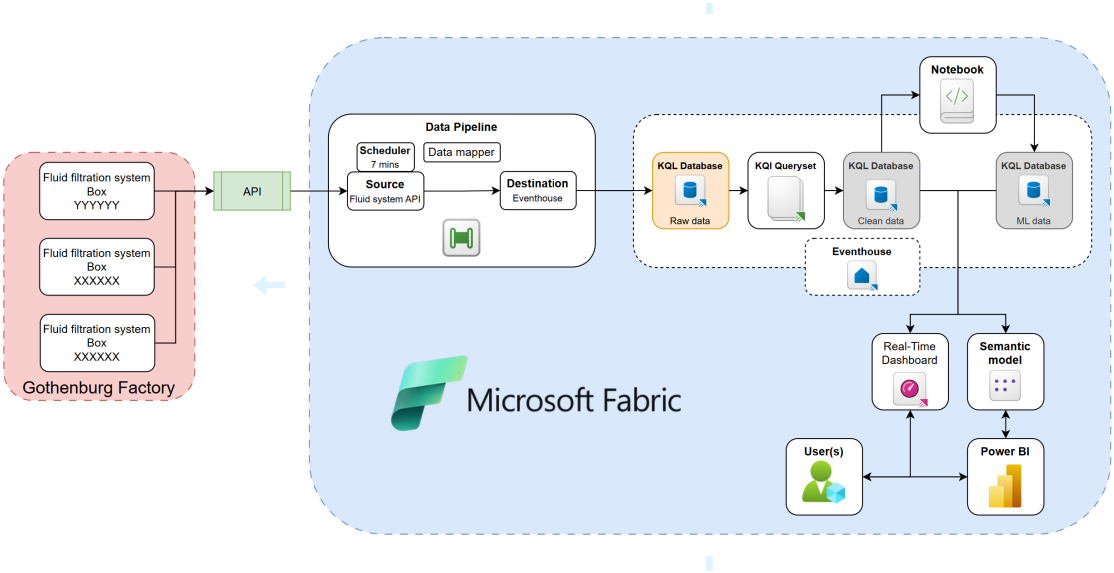


Figure 4.1: Data Model of Microsoft Fabric

Data from the silver layer is loaded into PySpark notebook, where forecasting modeling is performed using the Prophet model. The model output is appended to a KQL table. To simplify access for dashboards, another materialized view is created for this table that always shows the latest model run. This design improves dashboard performance by avoiding queries over large historical datasets of forecasting results. While in this project the volume of data is low, this method ensures scalability.

The data is mainly stored as KQL tables, with materialized views utilized as the silver layers. The bronze layer consists of raw, unprocessed data stored in KQL tables. The silver layer contains cleaned and transformed data, ready for analysis and visualization. Forecasting results are stored in a separate KQL table, with an additional materialized view for this too, with the latest run as the view. This method is mainly used as a practical means of querying the data inside the dashboards, e.g. both Power BI and the Real-time dashboard, improving performance of the dashboards.

KQL storage was chosen over alternatives such as Lakehouses for the modeling results due to its fast query performance, strong integration with Fabric's analytical tools, and support for real-time querying via materialized views. And while Power BI supports Lakehouse tables, the Real-time dashboard only supports data from KQL tables, making it a necessity. This approach ensures standardized, scalable, and reliable access to raw and modeled data for analytics and reporting within both types of items used as dashboards.

4.1.2 Databricks

While the primary objective of this project focused on Microsoft Fabric, Databricks was also evaluated to establish a baseline for comparison, as it is currently in use at SKF. One of the aims of using multiple platforms was to assess the viability of Microsoft Fabric's capabilities in this project, by benchmarking them against a

proven platform. Databricks is already in use for other workloads at SKF, making it a suitable candidate for evaluation in terms of its data processing and dashboarding capabilities. Databricks can also connect to other tools for visualization, such as Snowflake then to Grafana, if needed. This type of set-up is used for other pojects at SKF. However, for the purpose of this project, using the native dashboard feature was enough. For an overview of the Databricks data model, see Figure 4.2

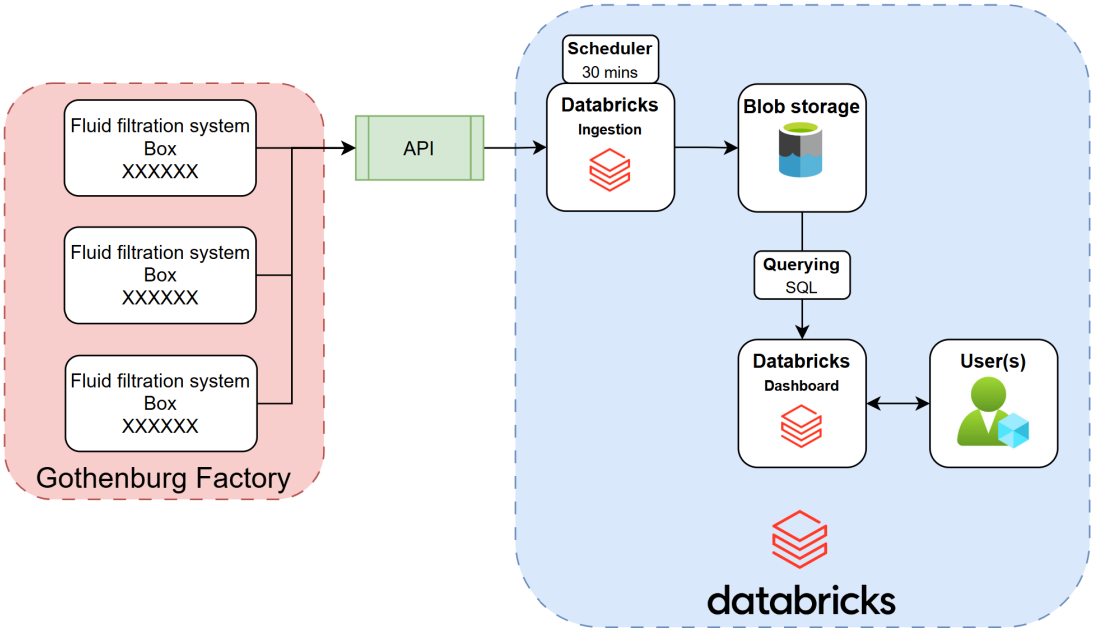


Figure 4.2: Data Model of Databricks

In Databricks, data ingestion was done by the same API as used in Microsoft Fabric. However, the data ingestion steps in Databricks were handled through Databricks PySpark notebooks. The data in the API was fetched using the `requests` package, and API keys, with the notebook scheduled to run every 30 minutes. Since the connected oil filtration systems update approximately once every hour, this setup produces some duplicate records, although fewer than those observed in Microsoft Fabric due to its higher refresh rate. This scheduling behavior is configurable on both platforms.

Due to the late introduction of Databricks into the project and constraints in both access and computational resources, virtually no data transformations were implemented. Instead, transformations were applied dynamically while querying the data inside the dashboard itself using SQL. Performing basic processing steps such as sorting out duplicates and renaming columns can be done in SQL. Even though the typical medallion architecture (bronze, silver, gold), as used in Microsoft Fabric, was not implemented for Databricks for this reason, it remains a fully viable approach within Databricks. Should this platform be considered as the platform for the continuation of this project, adopting the medallion architecture would be straightforward and beneficial.

RUL estimation modeling and sensor forecasting were not carried out within Databricks as part of this project due to resource limitations and due to the fact that it was not necessary for the comparison. However, it is important to note that notebooks within Databricks supports machine learning workflows in the same way as Microsoft Fabric. Consequently, machine learning, including time series forecasting using Prophet could be implemented with minimal adjustments. The key difference lies in the storage and retrieval of data, not in the modeling process itself, due to the usage of PySpark notebooks in both Databricks and Microsoft Fabric.

One of the key differences between Microsoft Fabric and Databricks is their approach to data storage. Microsoft Fabric is an end-to-end platform that includes integrated storage, such as Lakehouses and KQL databases, whereas Databricks requires an external storage solution. In this evaluation, data fetched in Databricks was stored in an Azure Blob Storage container. These tables effectively represent a bronze layer in the medallion architecture. For Databricks, silver and gold layers which were not created in this case, however, their implementation is fully supported within Databricks should SKF want to continue this project. Databricks dashboards query the data directly using SQL. Despite requiring external components for storage, Databricks offers a high degree of flexibility and scalability, making it a strong candidate for larger and more resource-intensive data and analytics projects.

4.2 Exploratory Data Analysis

As previously mentioned, in this project the CRISP-DM methodology was used as the guiding framework for data analysis and modeling, including EDA. Due to this, extensive effort was invested in data understanding, which included exploring and cleaning the raw data through comprehensive exploration and analysis.

4.2.1 Data Understanding and Preparation

The historical sensor data was provided as a CSV (comma separated values) file, accompanied by a separate CSV file for the laboratory results. In addition to this main CSV file, there were additional three "filter cycles" recorded. However, several factors makes these other cycles less suitable for training. For one, they were on a different oil filtration system, connected to a machine serving a different purpose (cutting fluid from grinding operation versus hydraulic oil). For this different system, its three cycles had different parameters. For example, one used a special type of filter that is not representative of typical operational performance, another was recorded using a different type of oil, and all but one cycle had limited laboratory data available. As a result, the decision was made to focus on a single cycle that is considered the most representative of actual production conditions and also has the most comprehensive data available. As a result, all the historical data results presented, in both *Exploratory data analysis* and *Machine learning* chapters, were based on one cycle, the most complete dataset. For transparency sake, this information is disclosed here.

Before conducting any EDA, several data preparation steps were undertaken to ensure data quality and consistency. This included correcting the schema to align with expected formats, handling negative and missing values, and renaming columns for clarity. Additionally, the date range was carefully set to focus on the relevant timeframe. New features were engineered, such as the time interval between measurements and the amount of oil cleaned during these intervals.

This pre-processing stage was essential to eliminate inconsistencies, improve data integrity, and enrich the dataset with features. Proper data preparation is crucial in any machine learning project, as it directly impacts the model's ability to learn patterns and make accurate predictions. Without this groundwork, the presence of invalid or misaligned data could lead to misleading results.

4.2.2 Descriptive Statistics

After cleaning the data, statistical analysis was conducted to gain an in-depth understanding of the behavior and distribution of the sensor data. Basic statistical measures, including count, mean, standard deviation, minimum, maximum, and quartiles, were calculated for each sensor. These summary statistics provide an overview of the data. See Table 4.1.

Table 4.1: Summary Statistics of the Sensor Dataset

	Flow	**Pressure**	**Relative humidity**	**Temperature**	**Oil cleaned**
count	4897	4897	4897	4897	4897
mean	0.75	2.50	1.03	36.11	44.21
std	0.18	0.02	0.61	5.63	10.64
min	0.35	2.39	0	22.5	1
25%	0.73	2.49	0.53	36.3	43
50%	0.82	2.50	0.97	38.7	48
75%	0.86	2.51	1.38	39.7	51
max	1.02	2.63	3.76	41.9	92

In addition to these basic descriptive statistics, the dataset was further evaluated using the Augmented Dickey-Fuller (ADF) test and the Signal-to-Noise Ratio (SNR) analysis. The ADF test is employed to assess the stationarity of the time series data to determine whether the sensor readings exhibit consistent statistical properties over time. The aim of the SNR analysis is to quantify the clarity of the signal relative to the variability/noise in the measurements. See Table 4.2.

In summary, the ADF results suggest that the sensor data can be considered stationary. However, visual inspection of the data reveals a more nuanced picture, with certain sensors exhibiting behaviors that challenge the stationarity assumption given by the ADF. This will be explored in greater detail in the following Section 4.2.3, (Visualisations). The SNR values are generally low across most sensors, implying considerable noise relative to the signal strength. However, the pressure sensor readings stands out as the sensor with the highest SNR, appearing more stable and potentially more predictable than the others. This aligns well with visual inspection

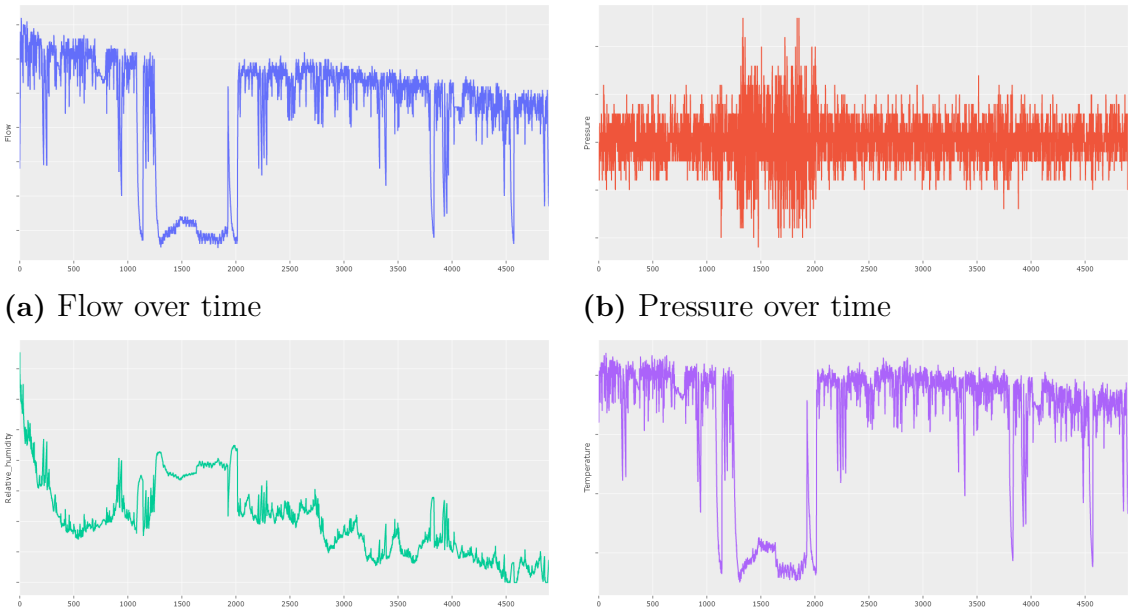
Table 4.2: ADF test and SNR Results

Sensor	Test Statistic	p-value	Stationary	SNR
Flow	-3.38	0.01	True	4.23
Pressure	-31.09	0.00	True	107.54
Relative_humidity	-3.48	0.01	True	1.70
Temperature	-3.46	0.01	True	6.41
Oil_cleaned	-3.60	0.01	True	4.16

of the data, and operational realities, as the pressure sensor is set as the control variable for the oil filtration system.

4.2.3 Visualisations

As part of the EDA, various visualizations were created to better understand the distribution, temporal dynamics, and interrelationships of the sensor readings. The first step involved plotting each sensor's measurements over time in order to identify trends and overall behavior across the observed period. These time series plots provide a view of the stability and variability of each sensor signal over time. The resulting visualizations are presented in Figure 4.3.



(a) Flow over time

(b) Pressure over time

(c) Relative humidity over time

(d) Temperature over time

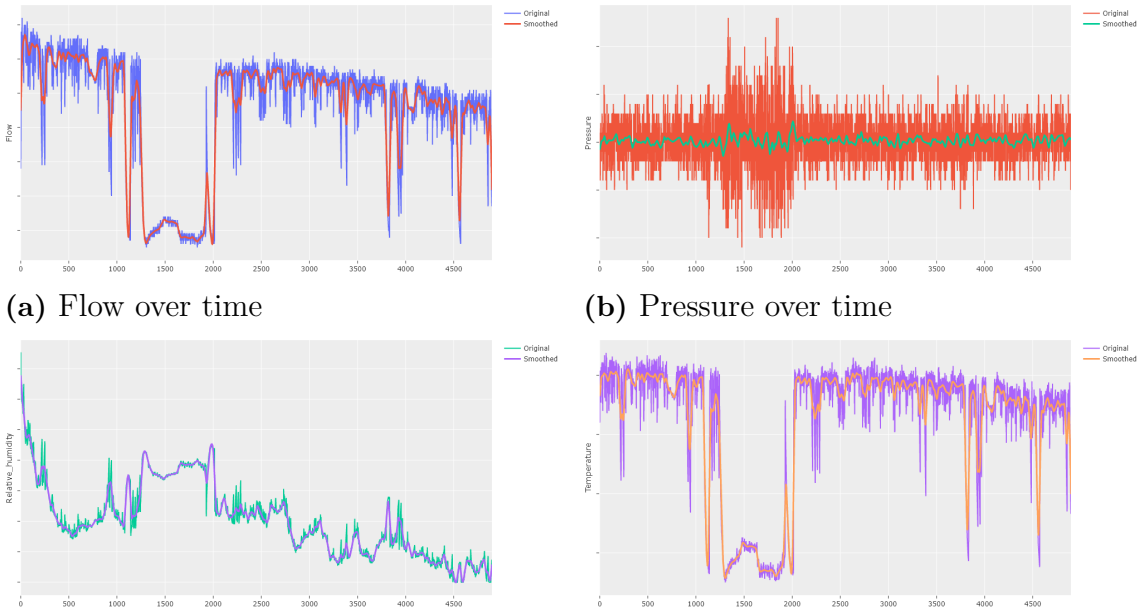
Figure 4.3: Time series plots of sensor measurements

Initially, a significant discrepancy in the sensor readings is observed at the beginning of the dataset. A noticeable shift in sensor behavior occurs, which corresponds to a production break of approximately one month during the summer. During this period, the system was placed into a low-power or standby mode to reduce operational stress while keeping the fluid moving to avoid settling of contaminants.

This results in a marked change in all sensor readings, particular in flow. Once production resumed, the sensors returned to their normal operating ranges. This behavior is clearly visible in all sensor readings, where the mean values temporarily change. An exception to this trend is observed in the pressure sensor; its mean remains relatively stable, but the variation or range of its values grows during this break.

In addition to this break, numerous abrupt dips are present in the data. These irregularities are attributed to routine shutdowns of the system, such as over the weekends, holidays and periods with little to no production. Both the one-month production break during the summer and these periodic shutdowns are important considerations when performing feature engineering and building predictive models. These aspects will be addressed in more detail in the following sections.

To complement the raw sensor readings and provide a clearer understanding of the underlying trends, a smoothing technique was applied to each time series using the Hodrick–Prescott (HP) filter. This filter was applied to separate the short-term component from the long-term trend in time series data. The goal was to reduce the impact of noise and fluctuations and make it easier to observe general behavioral patterns and long-term tendencies in the sensor data. The smoothed versions of the time series are presented in Figure 4.4.



(a) Flow over time

(b) Pressure over time

(c) Relative humidity over time

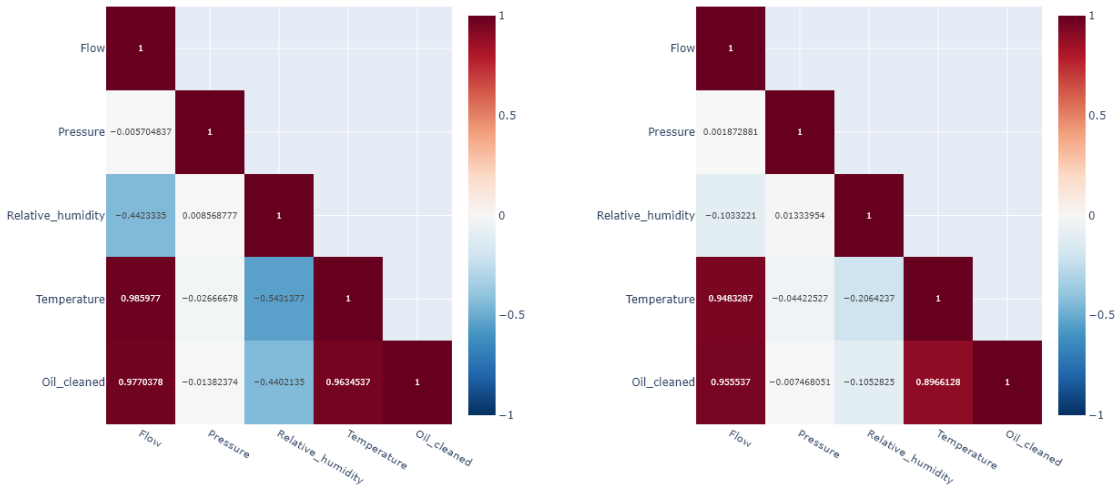
(d) Temperature over time

Figure 4.4: Time series plots of smoothed sensor measurements

To gain a deeper understanding of the distribution of the data, additional visualization methods were used in this project, focusing in particular on Kernel Density Estimates (KDE) and boxplots. These methods provide complementary views of the shape, spread, and potential outliers in the data. For each sensor, two sets of visualizations were created: one including the entire dataset with the one-month production break, and another excluding this break period. This comparison aims

to discover valuable insights into how the extended downtime affects the distribution and shape of of the sensor readings. Specifically, it highlights changes in variability and shifts in central tendency. It also highlights the presence of outliers that may be masked or exaggerated when the break period is included. Understanding these effects is critical for accurate feature engineering and model development. See Appendix A for KDE and boxplots.

Further visual examinations were carried as part of the EDA gain to better understand the relationships within the data. Both Pearson's and Spearman's correlation coefficients were calculated for the sensors to evaluate the strength and correlation between sensor readings. These coefficients provided complementary perspectives when compared to visual inspection using, with Pearson's measuring linear relationships and Spearman's capturing monotonic trends, allowing for a more comprehensive assessment of how the sensors interact with each other. See Figure 4.5.



(a) Pearson correlation coefficients **(b)** Spearman correlation coefficients

Figure 4.5: Correlation coefficients for all sensor data using (a) Pearson and (b) Spearman methods.

In addition to these correlation values, a correlation matrix encompassing all sensor measurements was generated and visualized. This matrix is an important visualization tool, since it offers detailed information regarding the interrelationships between the sensors and reveals patterns of similarity, redundancy, or independence. By examining the correlation matrix, deeper insight into the exact nature and strength of the connections between different sensors was achieved, which is crucial for understanding sensor behavior, identifying potential dependencies, and guiding feature selection in the following section. See Figure 4.6.

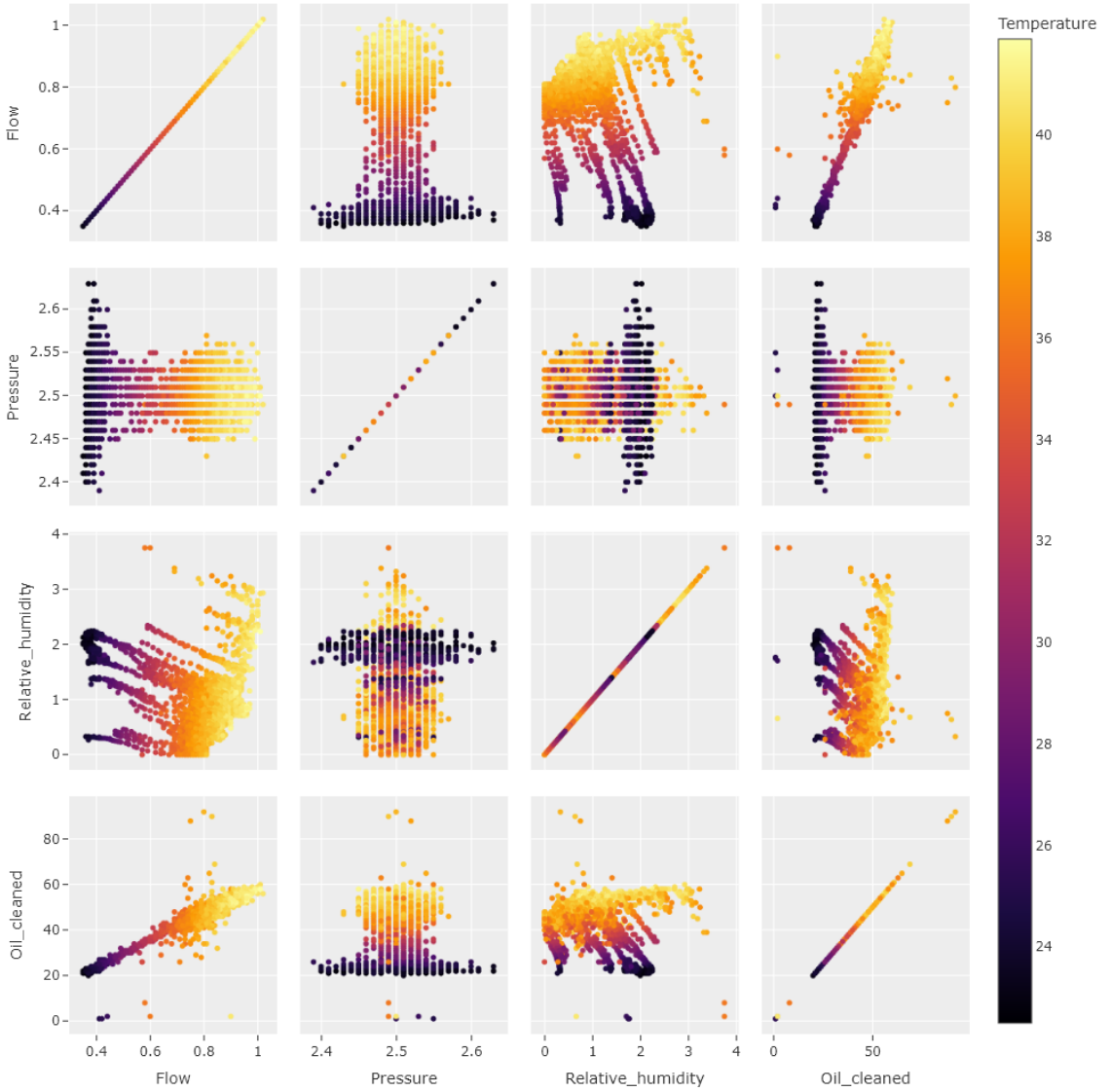


Figure 4.6: Correlation matrix for all sensors

By closely examining the correlation plot, several patterns become apparent and revealed meaningful connections between the sensors. For instance, an increase in flow corresponds with a rise in temperature, suggesting a correlation between these two parameters. Additionally, flow exhibits a strong correlation with the variable representing oil cleaned, which aligns logically since the oil cleaned measurement is derived from the total amount of oil cleaned.

Although the findings presented in this section may not reveal groundbreaking insights, the exploratory data analysis has nonetheless been very valuable. By analyzing and visualizing the data, a deeper understanding of the sensors' behavior and the relationships among the measured variables has been achieved. This comprehensive insight plays a critical role in guiding subsequent feature engineering and modeling workflows by ensuring that the underlying data patterns and behaviors are clearly understood before moving on to more advanced methodologies.

4.3 Machine Learning

In this section, the results of all machine learning aspects of this project will be presented. In total three machine learning models were used in this project: Random Forest Regressor (RFR), Extreme Gradient Boosting (XGB), and Support Vector Regression (SVR). These models were chosen because they are well.established methods that are widely used and known for their strong predictive performance and versatility across various regression tasks.

The models were trained and validated separately for each target variable using a direct multi-step forecasting approach over a 7-day forecasting horizon. This involved creating seven different models, with each model predicting the target variable at a specific future day, ranging from 1 day ahead up to 7 days ahead. The individual predictions from these models were then combined to form the complete 7-day forecast. This method was employed to capture temporal dependencies more accurately and to improve prediction robustness across multiple forecast horizon steps.

Additionally, two different time-series training and validation techniques were applied: sliding and expanding windows. The sliding window method involves using a fixed-size training set that moves forward in time, allowing for consistent data length but adapting to temporal changes. In contrast, the expanding window approach incrementally increases the training data as time progresses, which can help the models learn from a larger historical context. Employing both techniques provides a comprehensive assessment of model performance under varying temporal data availability scenarios.

4.3.1 Feature Engineering

Prior to applying machine learning models, the dataset underwent several preprocessing and feature engineering steps to ensure it was well suited for modeling. First, to align with the daily frequency of the lab data, the sensor readings were aggregated to daily aggregates. Second, lag features representing a weeks delay were created for each sensor variable, capturing temporal dependencies and allowing the models to account for recent historical trends. Additionally, a week of rolling mean was computed for each sensor to smooth out short-term fluctuations and reduce noise in the data. These transformations were performed to enhance the quality and clarity of the input features, aiming to improve the models' ability to learn meaningful patterns from the data and ultimately boost the accuracy of the predictions. In addition, total oil cleaned per day, a binary marker for summer production breaks, and elapsed time were incorporated as features.

Four target variables were considered in this project: MPC and particle counts of three different sizes: 4, 6, and 14 µm. The particle count corresponds to the ISO 4406 reporting standard for oil cleanliness, which is widely used as a standardized test to assess the contamination levels in oils.

4.3.2 Predictors of Remaining Useful Life

In this section, the results for all examined models; Random Forest Regressor (RF), Extreme Gradient Boosting (XGB), and Support Vector Regression (SVR) are presented, using two validation techniques: Sliding and Expanding windows. Each model underwent hyperparameter tuning, with the tuning process focused on optimizing performance for the MPC target variable. The various prediction targets act as proxies for deciding the condition of the filter, making the results indicative of model performance in determining RUL.

4.3.2.0.1 MPC In predicting MPC, the models generally performed well from a metrics perspective, except for the SVR model using the expanding window, which showed poor results. Among the models, XGB achieved the best performance, while SVR performed the worst. See Table 4.3.

Table 4.3: Prediction results for MPC

Metric	RF		XGB		SVR	
	Sliding	**Expanding**	**Sliding**	**Expanding**	**Sliding**	**Expanding**
MSE	0.343	0.505	0.165	0.220	0.514	0.913
RMSE	0.586	0.711	0.406	0.469	0.717	0.955
MAPE	6.53%	8.00%	4.34%	4.86%	7.38%	11.12%
R^2	0.883	0.828	0.944	0.925	0.825	0.689

However, certain anomalies were observed across all models and cross validation windowing validation methods. Despite using direct multi-step forecasting to align future target variables with sensor data, and generally achieving good metric scores, the predictions appeared "shifted" in time. This is true for almost all models and validation techniques. For example, when the MPC trend was decreasing, the models tended to overestimate values; conversely, when the MPC trend was increasing, they tended to underestimate the absolute values compared to the target. In addition to this, both RF and XGB exhibited a characteristic “stepwise” pattern in their prediction results. This behavior is presented in Figures 4.7 and 4.8.

4.3.2.0.2 Particle Counts In addition to MPC, particle count were also trained and validated against as target columns. As the measurements are of three different sizes (4µm, 6µm and 14µm), a total of nine models were trained and assessed in their capabilities in determining particle count. See Table 4.4, 4.5, and 4.6 for an overview of the models performance.

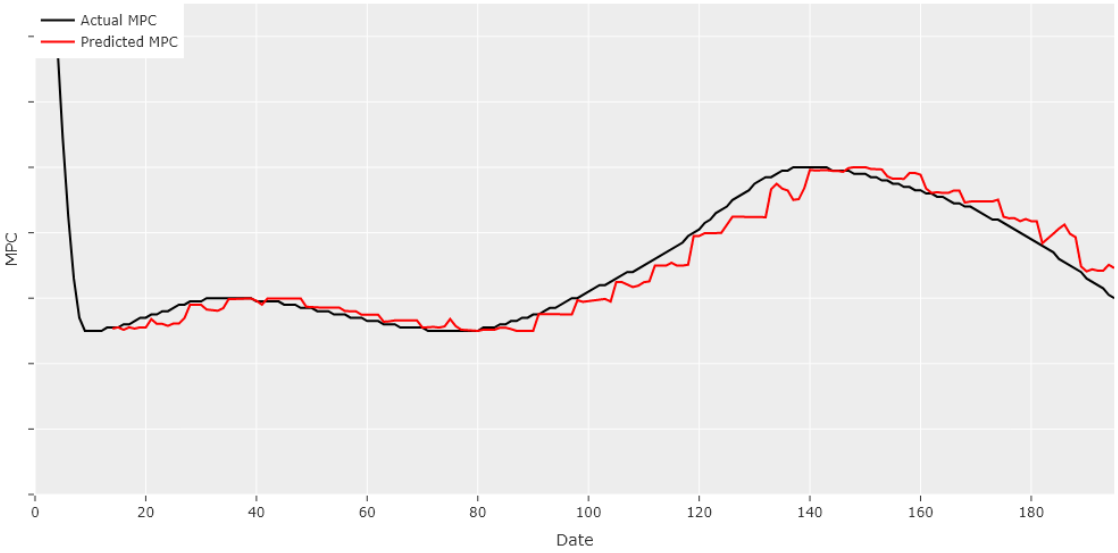


Figure 4.7: Prediction for MPC using XGB and sliding window

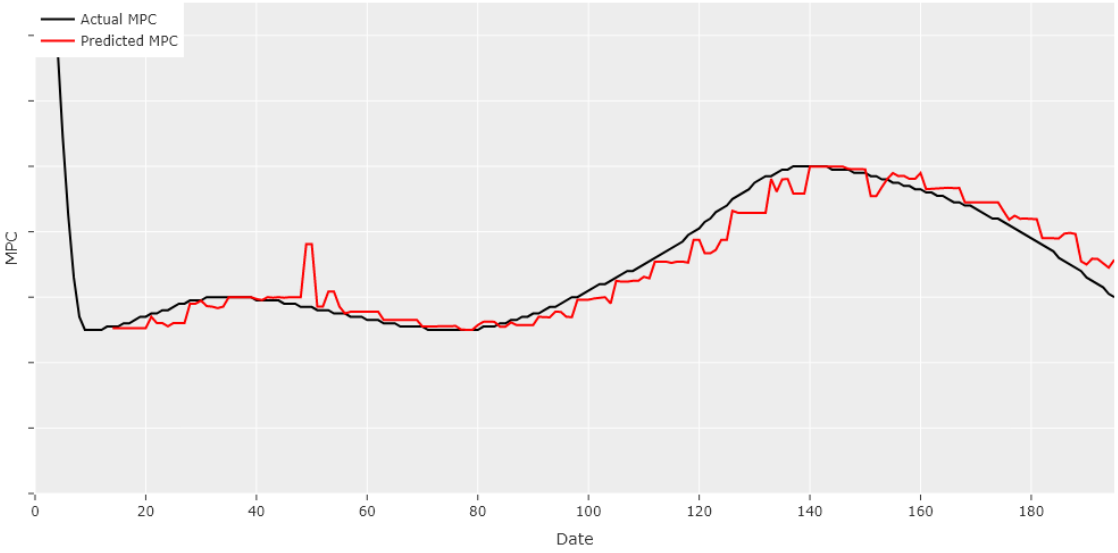


Figure 4.8: Prediction for MPC using XGB and expanding window

Table 4.4: Prediction results for Particle count (>4µm)

Metric	RF		XGB		SVR	
	Sliding	**Expanding**	**Sliding**	**Expanding**	**Sliding**	**Expanding**
MSE	186.690	345.874	102.445	133.527	243.651	1481.861
RMSE	13.663	18.598	10.122	11.555	15.609	38.495
MAPE	7.69%	10.91%	5.21%	6.63%	8.68%	26.86%
R^2	0.871	0.761	0.929	0.908	0.831	-0.026

Table 4.5: Prediction results for Particle count (>6µm)

Metric	RF		XGB		SVR	
	Sliding	**Expanding**	**Sliding**	**Expanding**	**Sliding**	**Expanding**
MSE	44.401	82.067	23.407	31.079	42.799	377.030
RMSE	6.663	9.059	4.838	5.575	6.542	19.417
MAPE	14.94%	25.17%	10.70%	14.34%	15.00%	38.63%
R^2	0.871	0.762	0.932	0.910	0.876	-0.095

Table 4.6: Prediction results for Particle count (>14µm)

Metric	RF		XGB		SVR	
	Sliding	**Expanding**	**Sliding**	**Expanding**	**Sliding**	**Expanding**
MSE	0.678	1.310	0.404	0.549	0.756	2.437
RMSE	0.823	1.144	0.635	0.741	0.869	1.561
MAPE	16.93%	28.64%	11.87%	16.88%	16.79%	39.73%
R^2	0.701	0.423	0.822	0.758	0.667	-0.074

For all particle count as target variables, model performance was generally worse when compared to MPC as target. Specifically, SVR using the expanding window performed particularly poorly, even receiving negative R-squared scores for all particle count targets. Similar to the MPC results, both RF and XGB exhibited a characteristic stepwise pattern, while SVR's predictions were very erratic. For RF and XGB, when the target values were decreasing, the models tended to overestimate, and when the targets were increasing, they tended to underestimate the values. Among the particle counts, predictions for the 14 µm target were by far the least accurate, altough all sizes showed poor accuracy.

4.3.3 Changepoint Prediction

In addition to validating the forecasting using sliding and expanding time-series validation, another method for evaluating the model's performance was applied to

validate the model's performance in detecting trend changes in the target variables. In this case, the MPC target was subjected to a "changepoint trend analysis". The XGBoost model, the overall best performer, was trained up to each MPC sample date and then used to forecast the following 7 days. This was done using an expanding window time series validation approach combined with the direct multi-step forecasting method used previously. See Figure 4.9 for an overview of these changepoint predictions.

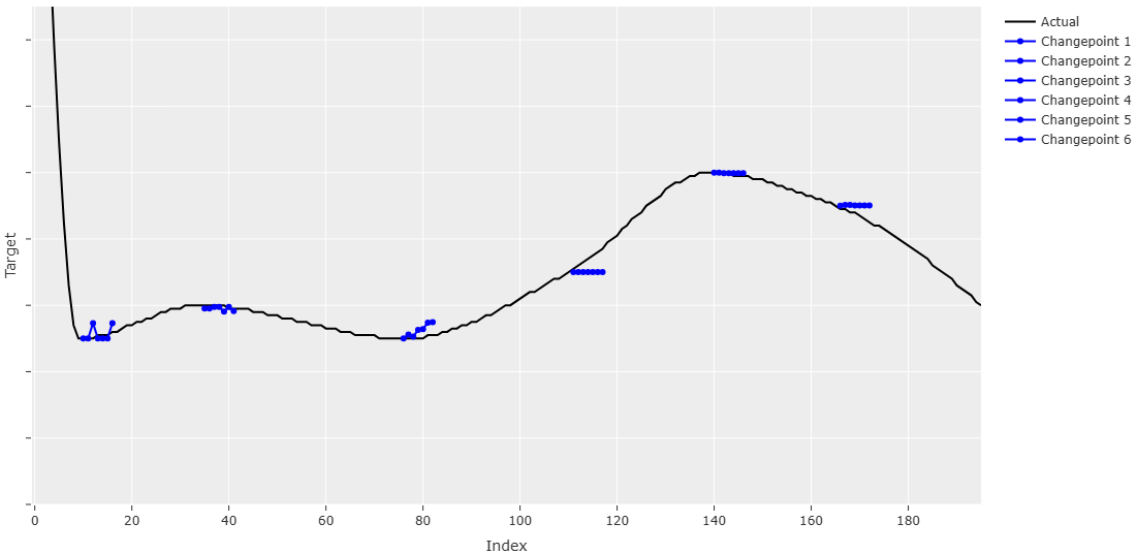


Figure 4.9: Overview of all changepoint predictions

In evaluating the results shown in Figure 4.8, the focus is not on the absolute accuracy of the predicted values, but rather on whether the model captures the correct target trend direction, whether it is increasing, decreasing, or remaining stable. This is a somewhat subjective visual assessment, but clear trend alignment is observed in only two of the six cases.

4.3.4 Sensor Forecasting

To model sensor forecasting, the Prophet model, developed by Facebook, was utilized. Each sensor was initially trained under two scenarios: First, using an expanding window time series validation with a 7-day forecast horizon. Second, on the full dataset to evaluate the model's performance with the maximum available data. These approaches help assess the model's ability to adapt to newly available data and understand how well it fits the training data over time, essentially showing how the model accuracy "converges" to as it is exposed to more data.

As highlighted earlier, a notable issue in the dataset was a one-month production break during the summer. While this period was initially handled using Prophet's holidays parameter, it led to the model overcompensating for the break, ultimately worsening overall predictions. Consequently, the decision was made to exclude data during the summer break interval during training.

In the previous section, MAPE was used as one of the evaluation metrics. However, in this case, MAPE yielded disproportionately high error values for certain sensors. As a result, SMAPE was chosen in place of MAPE since it provides a more balanced representation of forecast accuracy. For the sensor forecasting results, see Table 4.7 and 4.8.

Table 4.7: Prophet sensor forecasting using expanding window validation

Metric	Flow	Pressure	Relative Humidity	Temperature
MSE	0.04	0.00	0.12	46.10
RMSE	0.21	0.02	0.34	6.79
SMAPE	15.20%	0.57%	47.65%	10.08%
R^2	-3.35	-0.05	0.32	-4.54

Table 4.8: Prophet sensor forecasting using all data

Metric	Flow	Pressure	Relative Humidity	Temperature
MSE	0.01	0.00	0.03	6.38
RMSE	0.08	0.02	0.18	2.53
SMAPE	7.21%	0.54%	22.81%	4.32%
R^2	0.35	0.01	0.88	0.23

From a metrics standpoint, the sensor forecasting results for expanding window validation are generally poor, with negative R-squared scores across all sensors except for the relative humidity sensor. This indicate that the models struggle to generalize predictions when trained on limited historical data. However, the performance improve when the model is trained on the entire dataset, indicating that Prophet benefits significantly from larger volumes of data.

In the case of the relative humidity sensor, the R-squared score increases from 0.32 under expanding window validation to 0.88 when trained on the full dataset, which highlights the model decent performance when enough data is available. Despite this improvement, the overall performance for all sensors remains quite poor, with low or negative R-squared values for other sensors, such as flow and temperature. This trend indicates the model's limited ability to capture the complex dynamics or noise present in the sensor data, particularly in a real-world, evolving context where new data is continuously introduced.

However, despite the improved metric, a visual inspection reveals a clear mismatch between the predicted and actual values This indicates that even for the best performing sensor model, the model requires several months of data before its predictions become reasonably accurate with reality. Other sensors, in particularly the pressure and temperature sensor exhibit even weaker performance, as shown in Table 4.8. These models struggle to capture meaningful patterns, limiting their prac-

tical forecasting value. Additionally, the models, for the most part, fail to capture the variance, or range, of the data accurately.

For a visual comparison, see Figure 4.10 and Figure 4.11, which illustrate the relative humidity forecasts for expanding window validation when compared to training on the full dataset.

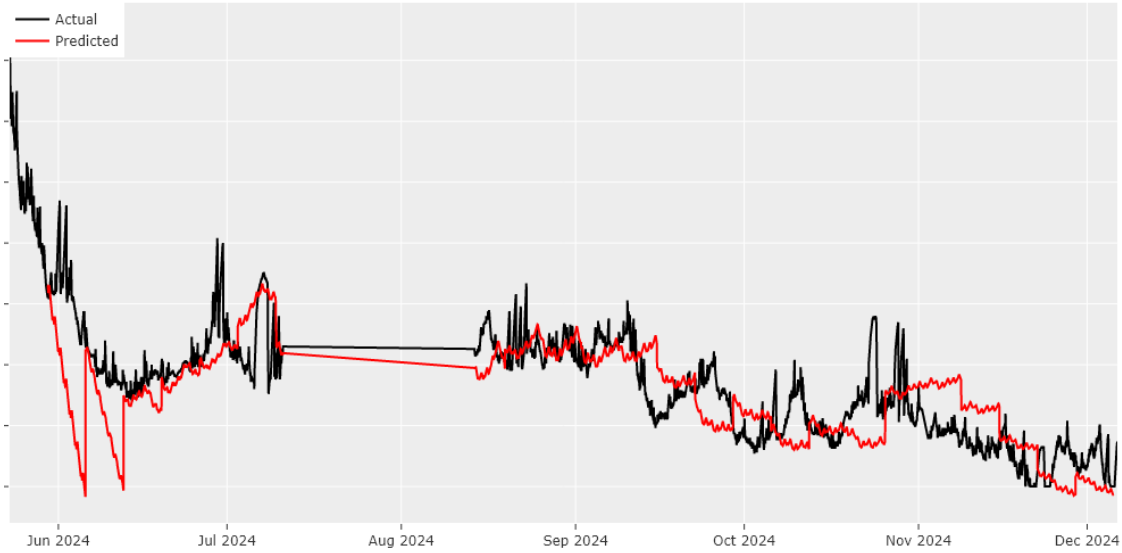


Figure 4.10: Forecasting of relative humidity sensor using expanding window

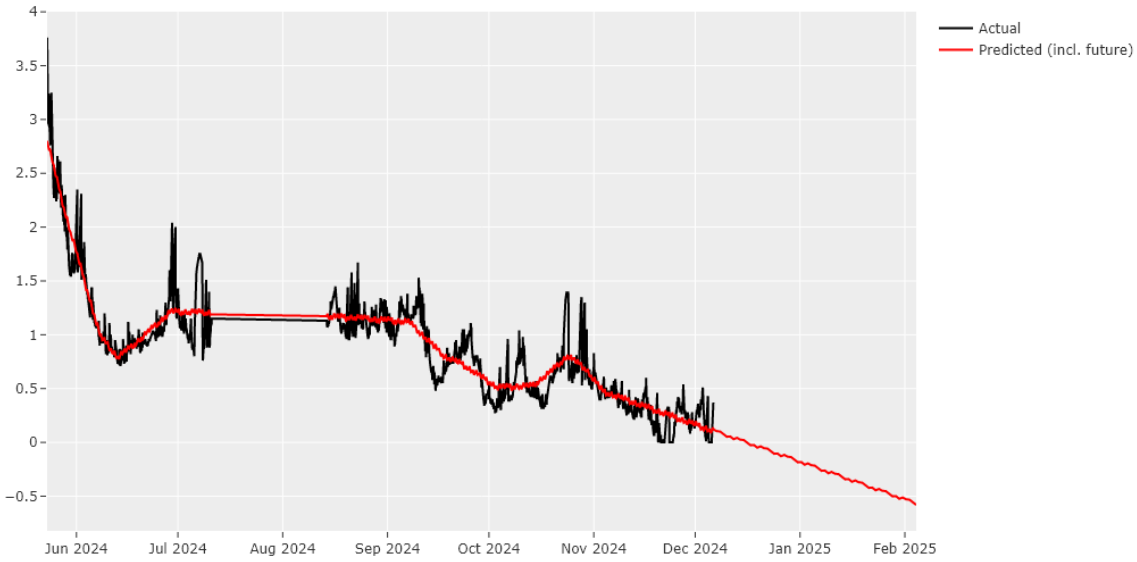


Figure 4.11: Forecasting of relative humidity sensor using all data

The difference highlights the significant improvement in stability and pattern recognition once the model has access to the complete data history.It is important to note that the goal of these forecasts is not solely to achieve precise point predictions. In many operational contexts, the underlying trend and drift of the data are of equal, if not greater, importance. Recognizing this, the "trend" component of the additive

Prophet model was extracted and used for visual display in the dashboard to better communicate long-term behavior. An example of this can be seen in Figure 4.12. Consequently, the trend component of the model prediction was chosen for display in the dashboard, rather than the full Prophet prediction (yhat-value), to better highlight the long-term trend of the sensor data.

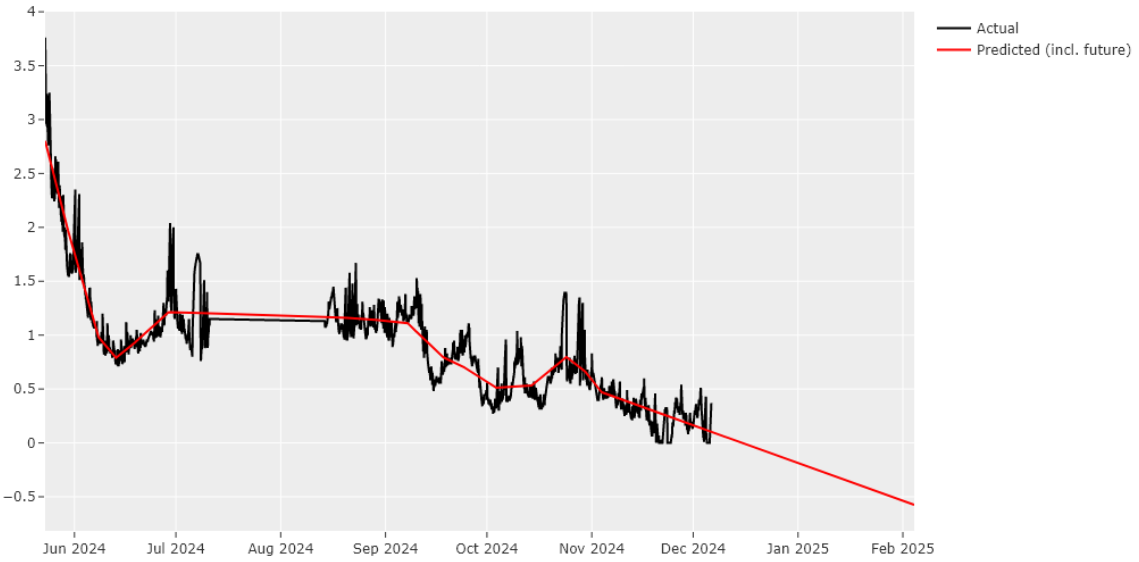


Figure 4.12: Trend component of relative humidity sensor forecasting

4.4 Dashboard Development

During the course of the project, multiple stages of dashboard prototypes were developed prior to the actual implementation in the final tools, see below. These development stages served several important purposes. Most importantly, they facilitated early feedback, supported idea generation, and supported later dashboard developments.

4.4.1 List of Requirements

The following list outlines the initial requirements and ideas gathered as a foundation for developing the first dashboard prototype. These requirements were compiled through brainstorming sessions and discussions with SKF subject matter experts. The goal was to ensure that both technical and operational needs were addressed from the outset, enabling early alignment and meaningful iterations throughout development.

- Display historical sensor data (Flow, Pressure, Temperature, Humidity) without ML forecasting.
- Show predicted sensor trends using ML (e.g., Prophet trend lines).
- Include total oil cleaned as a numeric counter.
- Indicate the last filter change and estimated time until next required change.
- Display timestamp of last received data transmission.

- Provide real-time alerts for abnormal sensor values (e.g., high pressure).
- Notify users if sensors or machines are offline or sending duplicate/no data.
- Show the current status of the ML model (Active/Idle).
- Allow selection of custom time intervals and reporting granularity.
- Display predicted vs. actual values clearly (with visual distinction).
- Include forecasted MPC (Membrane Patch Colorimetry), and particle count.
- Provide notifications for filter maintenance and data drift.
- Display data distribution using histograms, box plots, etc.
- Use control charts to track whether values stay within normal limits.
- Visualize historical maintenance activity logs.
- Notify if the data pipeline is inactive or delayed.
- Color-code urgency for maintenance (e.g., remaining useful life).
- Allow manual filtering by sensor type or time range.
- Present data summaries using pie charts or other overview visuals.

4.4.2 Graphical Dashboard Mockup

The second stage in the dashboard development process involved the creation of visual mockups to help conceptualize the layout and content of the final dashboards. This was accomplished using the diagramming tool *draw.io*, which enabled the design of schematic overviews that illustrate what the dashboards could look like, including layout structure, number of pages, key components, and their arrangement.

The purpose of these visual mockups was to translate the initial list of functional requirements into tangible representations, which allows for early-stage feedback from stakeholders. They served as low-fidelity prototypes that helped communicate design intentions and align expectations before any actual implementation in Power BI or other tools began. See Figures 4.13 and 4.14 for an overview of the initial mockups.
While the mockup incorporated several of the core ideas outlined in the requirement specification, it should be noted that due to it being a static image, it does not support interactive elements. The sensor readings, plots, alerts, and other visual elements were manually added as placeholders. These elements were included solely for the purpose of demonstrating what the user interface and features could look like to SKF in order to gather more detailed feedback on what they considered essential for the dashboard's functionality.

Additional mockups were developed to incorporate further elements from the initial list of requirements, along with new features identified through feedback and further brainstorming sessions. These designs can be seen in Figures 4.15 and 4.16.
These graphical mockups played a crucial role in the following development stages, helping to evaluate which elements were considered essential and most insightful for inclusion in the final dashboards. By presenting design concepts to SKF, the mockups facilitated productive discussions with project stakeholders, thus allowing for clearer prioritization of features and refinement of the overall dashboard based on needs and expectations.

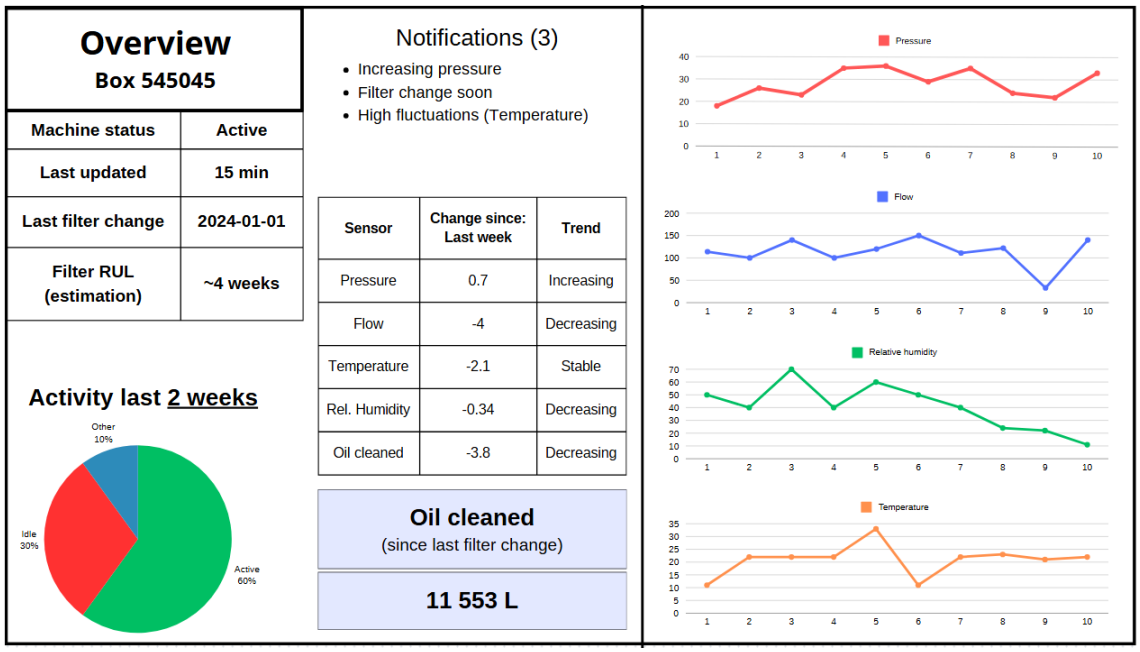


Figure 4.13: Dashboard mockup to visualize layout, structure, and key components.

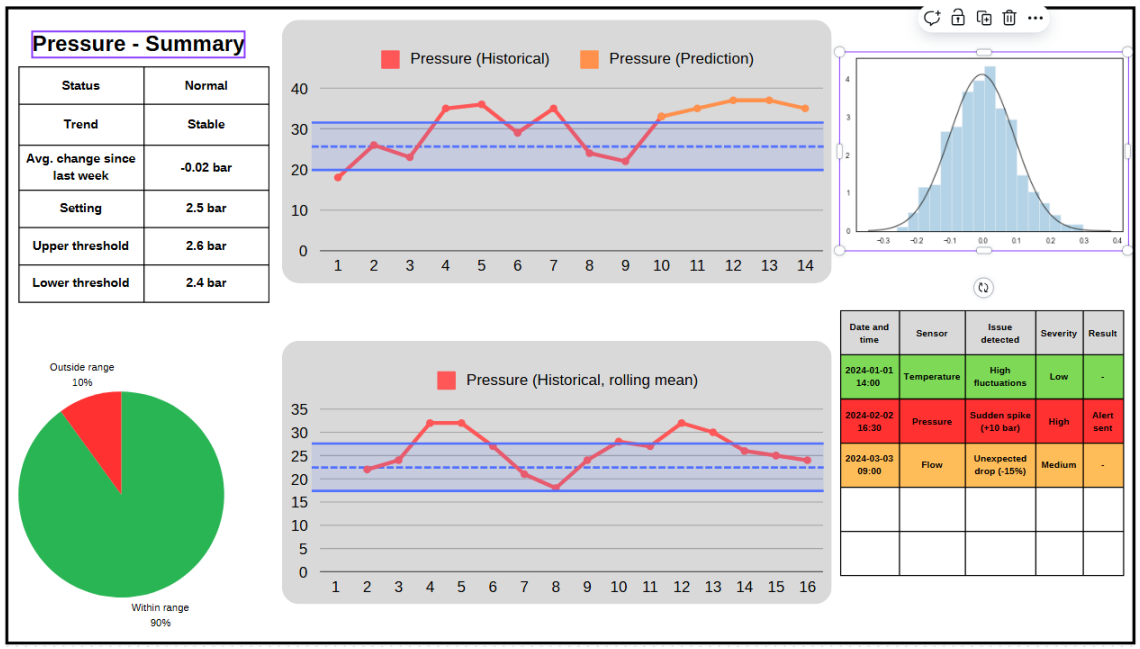


Figure 4.14: Dashboard mockup to visualize layout, structure, and key components.

4.4.3 Dashboard Prototype using Historical Data (Power BI)

The third dashboard prototype was developed using **Power BI**, focusing on historical data. This prototype served as a foundational step in evaluating Power BI's capabilities and limitations within the context of the project. The primary objective

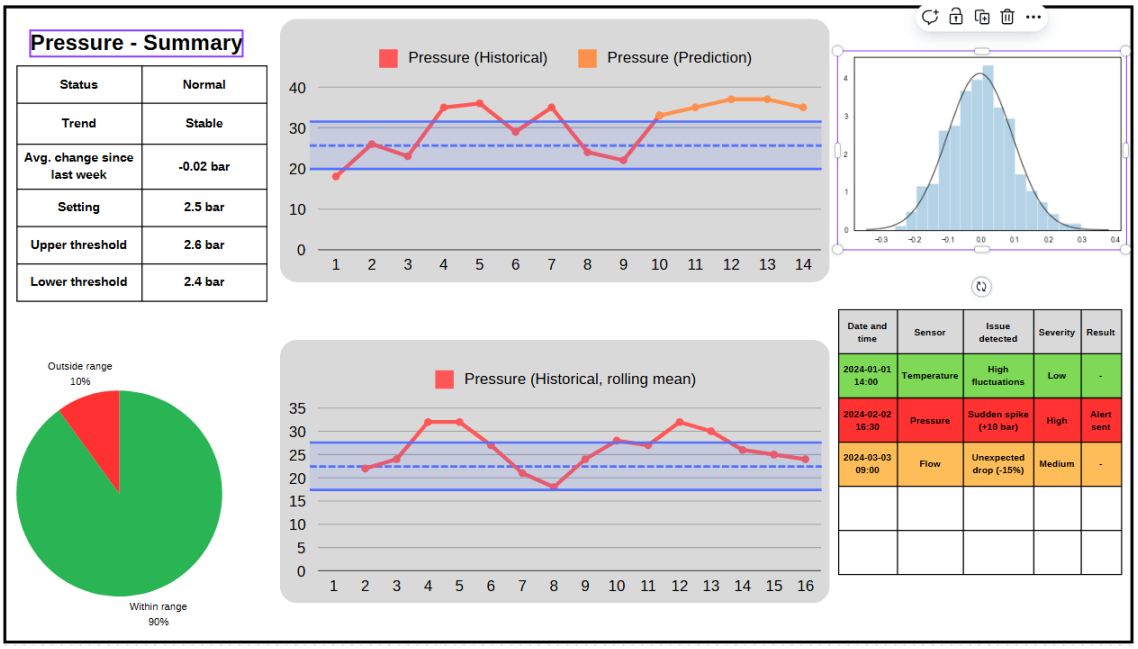


Figure 4.15: Dashboard mockup, detailed view with forecasting

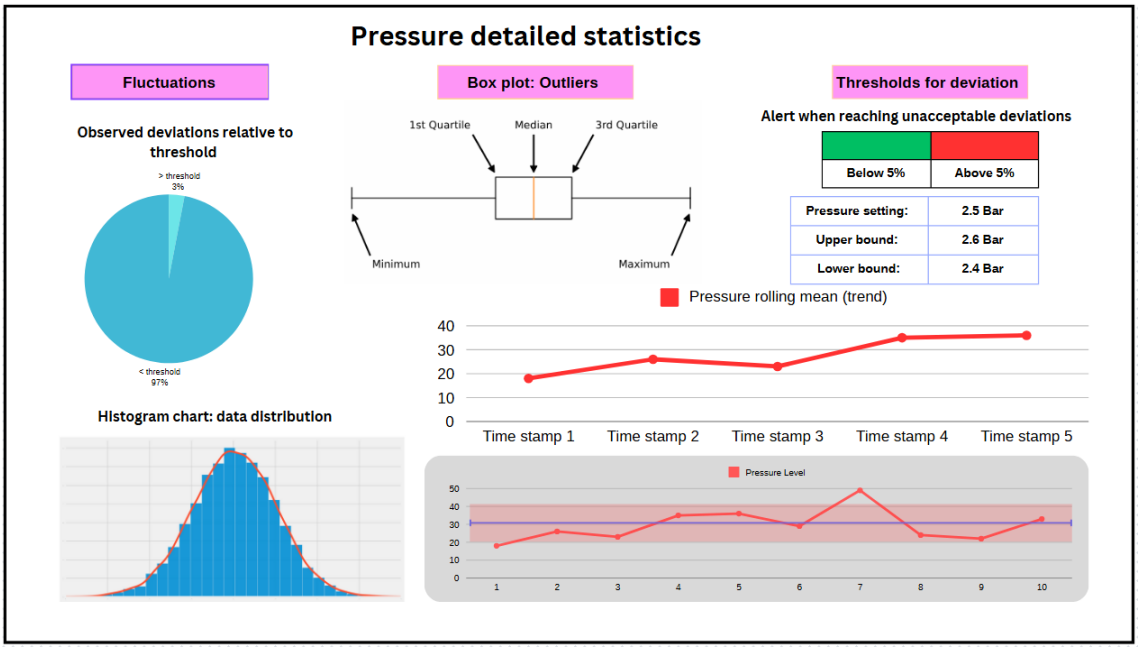


Figure 4.16: Dashboard mockup, detailed view with forecasting, second version

was to explore the platform's functionality, specifically its capability in data querying, table construction, visual customization, and interactive plotting. Other than technical exploration, this stage was also used to gather further **feedback** from the project stakeholders. The Power BI prototype functioned as a *sandbox environment*, offering a preview of how the final dashboard might be structured before moving forward with real-time data implementation. See Figure 4.17, and 4.18.

4.4.4 Dashboard Prototype using Real-time Data (Power BI)

The third and final dashboard prototype was also developed using **Power BI**, but this time with a focus on real-time data integration. This stage marked a critical step in assessing Power BI's capabilities and limitations in handling streaming data, especially when compared to the previous phase, which emphasized historical data analysis. As with previous stages, this stage also included collecting valuable user feedback. The key aspects examined during this stage included: Integration and visualization of streaming data Responsiveness, performance and scalability of using real-time data.
Overall, this prototype provided critical insights into the potential of Power BI for real-time analytics, laying the groundwork for future enhancements and confirming the tool's viability in dynamic, data-driven environments.
In summary, the Power BI prototypes served as a **crucial validation step**, allowing the project team to evaluate design concepts, gather stakeholder input, and refine dashboard requirements in a low-risk environment before transitioning to the real dashboards.

4.4.5 Final Dashboards

This section describes the final dashboards created, and explains the design, and content as well as the methods of data ingestion that differed for each version. Three tools were used to create dashboards, and the key difference between these and the previous prototypes lie in their multiple pages, the forecasting capability, and their use of real time data.

4.4.5.1 Microsoft Fabric

Two tools to create dashboards are part of the Microsoft Fabric environment, namely Power BI and the Real-Time dashboard. The solutions using these tools are presented below.

4.4.5.1.1 Power BI The initial version of the dashboard was developed in Power BI, leveraging insights gained during the prototype stage, particularly from testing historical data within the platform. This experience greatly informed the design and functionality decisions in the project. The dashboard consists of two primary pages. The first page serves as the main overview panel that summarizes system alerts/alarms. It also provides a quick visual summary of turnovers per box, enabling users to easily assess how many turnovers each box has completed in its current filter cycle. This high level view enables users to monitor key operational metrics at a glance without drilling down into specific oil filtration systems. Should the user want more detailed information per system basis, the second page offers a more detailed panel, presenting sensor readings alongside other metrics and metadata such as turnovers, oil cleaned, location, production channel, and the timestamp of the last data update. This detailed insight allows users to investigate specific conditions or anomalies in the production environment. See Figure 4.17 and 4.18.

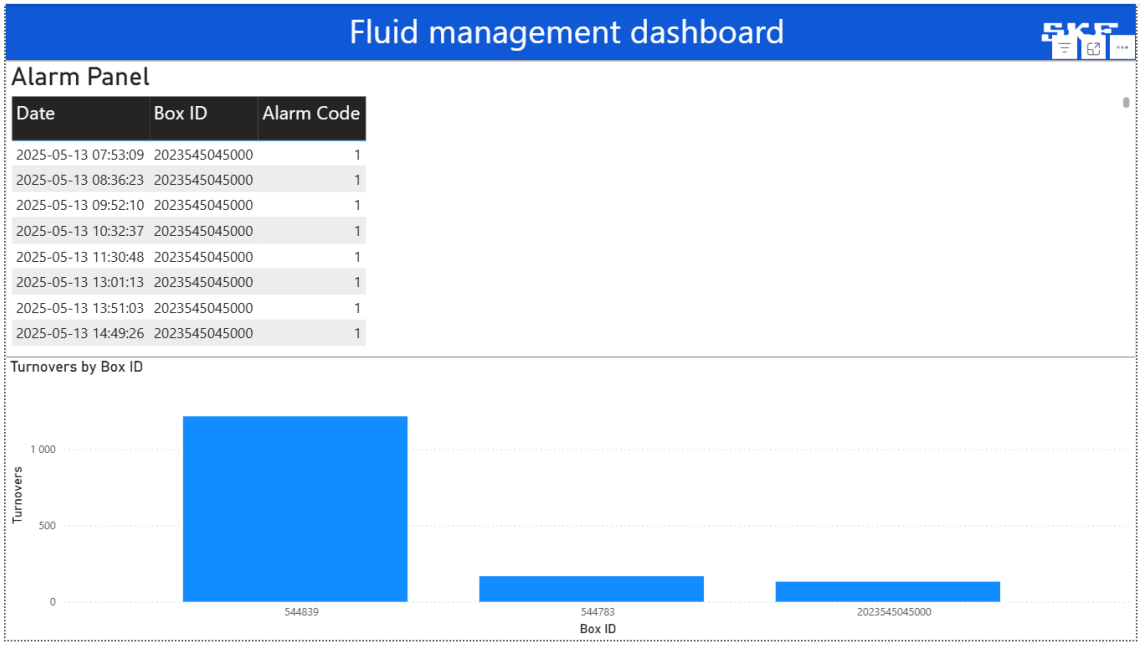


Figure 4.17: Overview page of the dashboard developed in Power BI

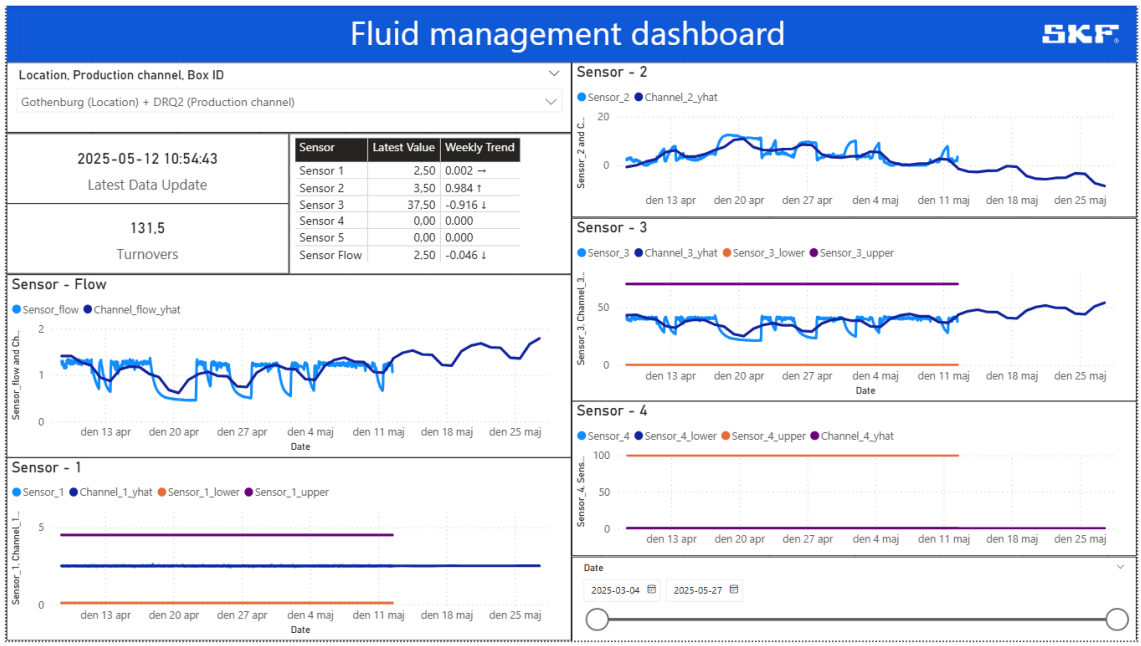


Figure 4.18: Overview page of the dashboard developed in Power BI

Despite its strengths, several limitations became apparent during development using Power BI. Performance issues emerged, particularly related to querying and data processing. Tasks that are typically straightforward when using SQL or KQL (for Databricks and Real-Time Dashboard respectively), such as combining columns, summarizing metrics, or applying more complex data processing, in Power BI required writing DAX formulas, which proved less intuitive and efficient. This complexity made the creation and management of tables cumbersome within Power BI.

Nevertheless, Power BI's primary strength lies in its more in-depth visualization capabilities. It offers a broad selection of chart types, settings of charts, and themes. This rich set of options enables the creation of visually compelling and highly customizable dashboards.

4.4.5.1.2 Real-time Dashboard In addition to the Power BI Desktop application, Microsoft Fabric's Real-Time Dashboard was used in this project as a key tool for visualizing data. Unlike Power BI, which relies on DirectQuery to connect to KQL databases, the Real-Time Dashboard enables direct querying via KQL. This approach allows for better performance in terms of lower latency and greater scalability, particularly when handling high-frequency and high-volume data in real-time.

The dashboard developed using this tool consists of three interactive pages. The first page provides an overview panel displaying system alerts and the date of the most recent filter change, which is dynamically detected via the oil cleaned data column. The second page is a detailed view, where users can filter between oil filtration systems by selecting a specific location, production channel, and then the desired system. The third page replicates this detailed view but integrates a sensor forecasting feature, which can be toggled on or off through user-defined parameters. See Figure 4.19, 4.20, and 4.21 for an overview over the various pages of the Real-Time Dashboard.

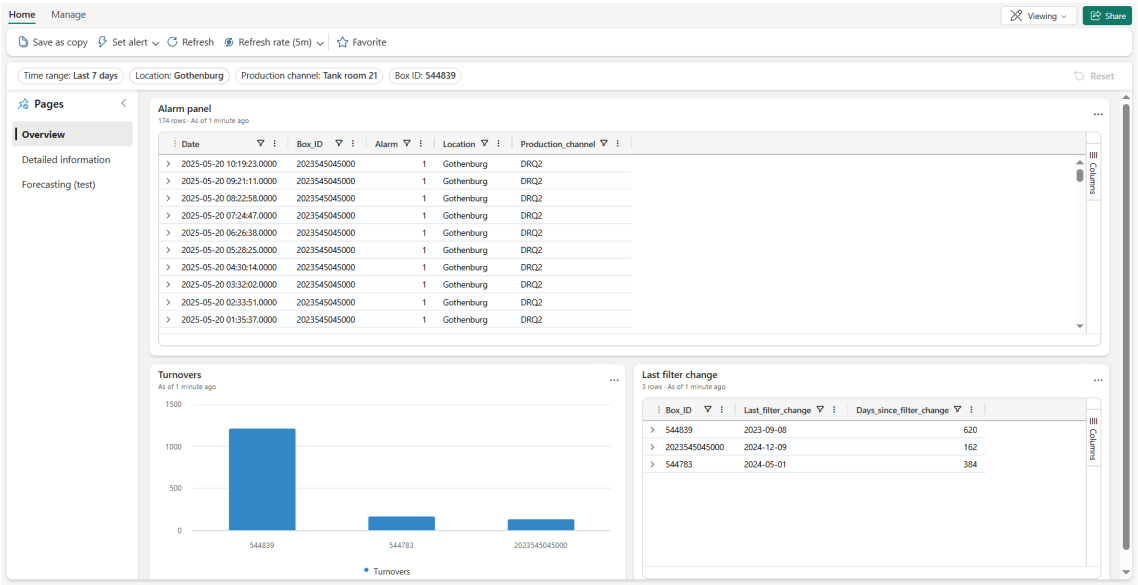


Figure 4.19: Overview page of the Real-Time Dashboard in Microsoft Fabric

One of the key strengths of the Real-Time Dashboard is its native integration with Fabric's Eventhouse, allowing users to query streaming data directly via KQL. This capability enables data updates to be reflected immediately in the visualizations, without significant delays (usually sub 100 milliseconds), making it a highly suitable option for systems that require low latency and scalability. However, while the Real-Time Dashboard excels in speed and responsiveness, it comes with certain trade-offs. Its visualization capabilities are limited to standard chart types, such as time series

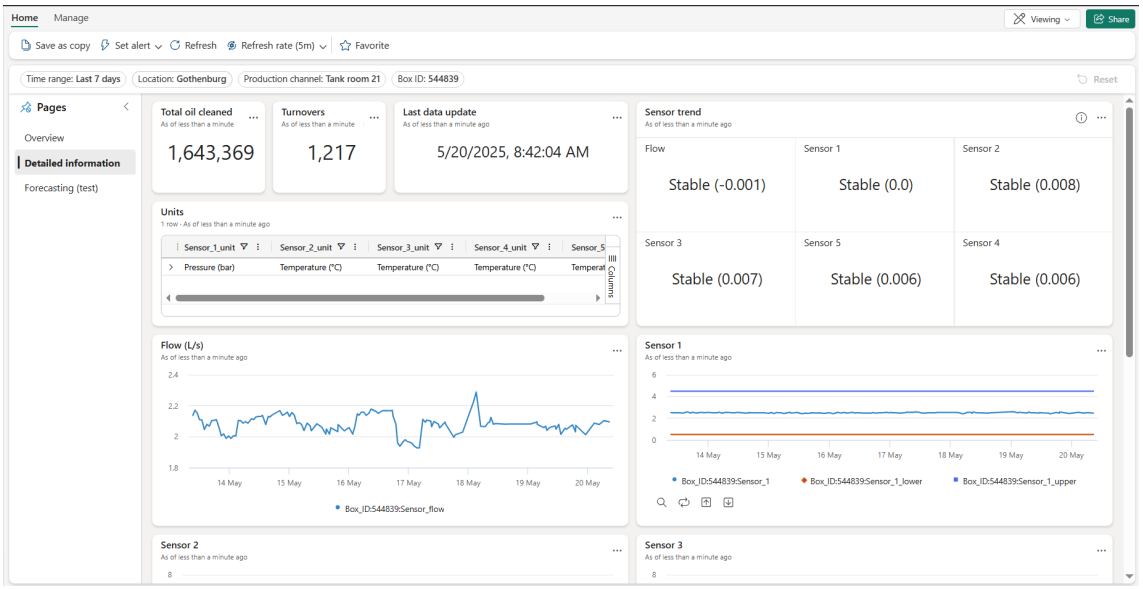


Figure 4.20: Detailed page of the Real-Time Dashboard in Microsoft Fabric

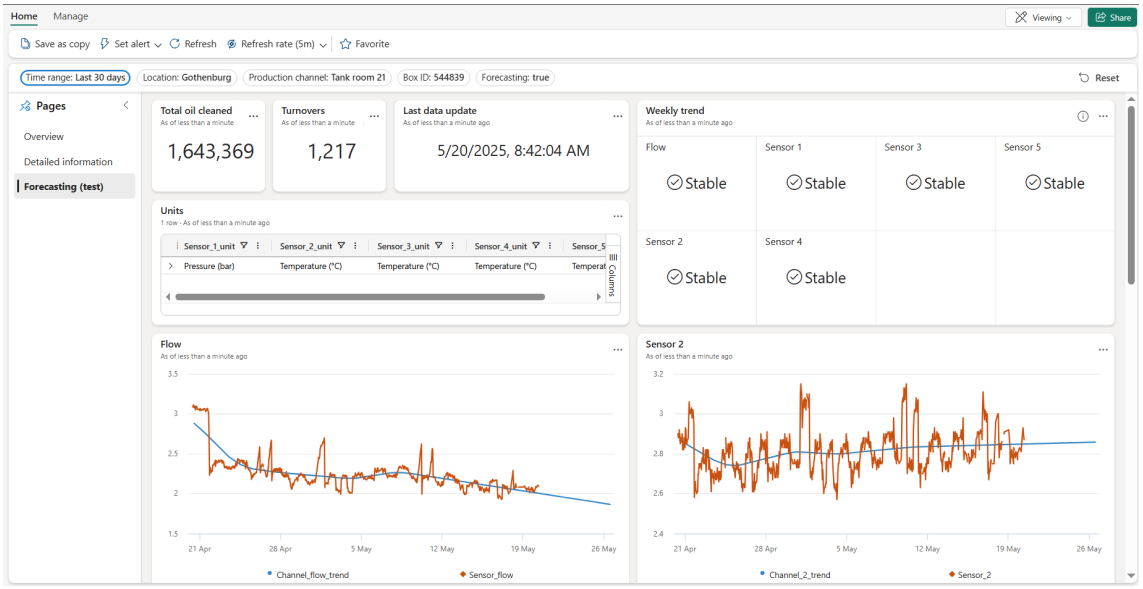


Figure 4.21: Detailed page of the Real-Time Dashboard in Microsoft Fabric with trend forecasting

plots, pie charts, and bar chart. And these charts have limited to no customization, with no way to change colors of data. Compared to Power BI, which offers a richer set of graphical customization options and thematic controls, the Real-Time Dashboard is more streamlined, prioritizing performance over design customizability.

The built-in alert system in Real-Time Dashboard another valuable feature, allowing users to define thresholds for metrics. When those thresholds are exceeded, alerts can be automatically sent through Microsoft Teams or Outlook. However, the system does not support dynamic thresholds, such as comparing one column's value against another, which limits its use in more complex monitoring scenarios. Overall,

the Real-Time Dashboard serves as a highly effective tool for scenarios demanding immediate visibility into streaming data, with strong potential for scalability. Its KQL language simplifies the creation of interactive, real-time visualization elements and thus making it a compelling alternative to Power BI.

4.4.5.2 Databricks

The final dashboard was developed using Azure Databricks, a tool that is currently in use at SKF. This makes Databricks a highly relevant alternative that aligns with existing workflows and infrastructure. The dashboard accesses data stored in Azure Data Lake Storage (ADLS) through SQL queries, enabling efficient retrieval and processing of structured datasets. In terms of visualization, Databricks offers slightly better capabilities compared to Microsoft Fabric's Real-Time Dashboard. While it does not quite match Fabric's performance for live data queries, it still significantly outperforms Power BI and represents a viable option for real-time analytics. The structure of the Databricks dashboard closely mirrors that of the Real-Time Dashboard in Microsoft Fabric, featuring both an overview panel and a detailed information panel. However, this implementation does not include the sensor forecasting panel, primarily due to time and resource constraints. This functionality could be added later using Databricks notebooks, which support seamless integration of machine learning models and predictive analytics in the same way as Microsoft Fabric does. See Figure 4.22, and 4.23 for a detailed look over the various pages of the dashboard in Databricks.

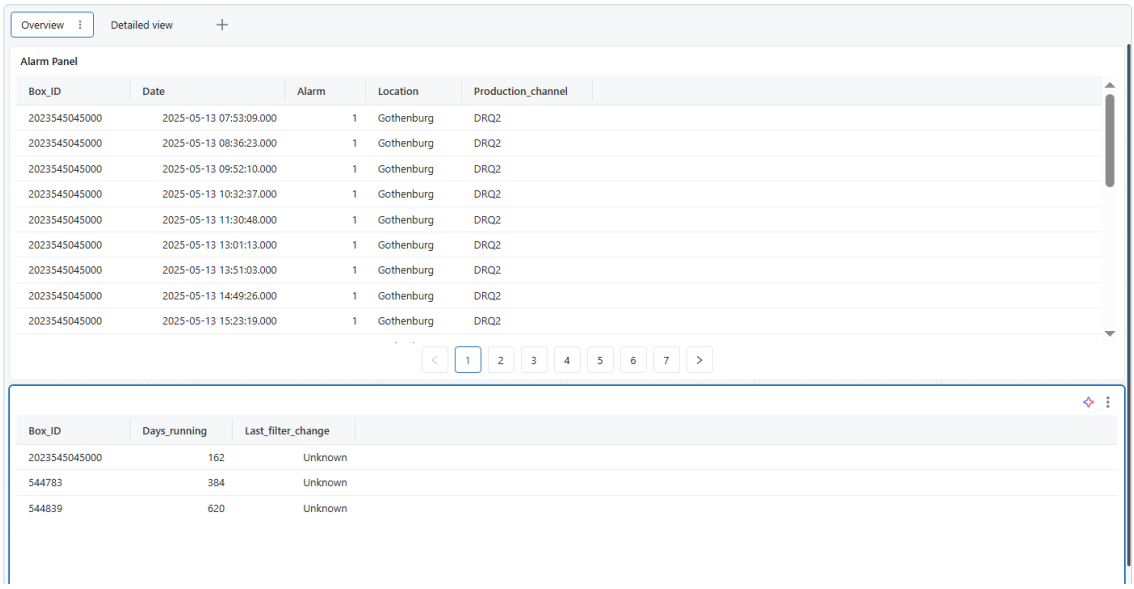


Figure 4.22: Overview panel in the Databricks dashboard

One of the key advantages of Databricks is its dynamic alerting functionality. Using notebooks, users can implement conditional logic to trigger alerts, for example when one column exceeds another threshold column. This is especially notable for this project, as the thresholds are defined in their own separate column. Through Notebooks, Databricks can notify users via email or SMS, although this requires

Figure 4.23: Detailed panel in the Databricks dashboard

integration with external services, such as SMTP. This level of flexibility is not available in Microsoft Fabric, which relies on its built-in alert system, and is limited to static values. As a result, Databricks provides more powerful options for automated monitoring and notification.

5
Discussion

In this chapter, we discuss the key conclusions drawn from the project, with a focus on platform selection and modeling approaches. Additionally, we offer our reflections and recommendations for the project's future development.

5.1 Platform

In our evaluation of platforms, we focused on two primary ecosystems: Microsoft Fabric and Databricks. Within Microsoft Fabric, both Power BI and the Real-Time Dashboard were tested as candidate tools for building the dashboard. Each option presented distinct advantages and drawbacks. Power BI's main strengths lie in its high degree of customizability and versatility in data presentation. However, both the Real-Time Dashboard and Databricks outperformed Power BI in terms of performance and scalability, which are critical factors if the project were to scale up to monitor additional boxes.

Currently, the system integrates data from only two boxes connected via the API, with several months of historical data stored for Microsoft Fabric and a shorter window of a few weeks for Databricks (as Databricks was introduced later in the project). Data ingestion frequency differs between platforms: Microsoft Fabric receives new data approximately every 7 minutes, while Databricks updates roughly every 30 minutes.

Despite the limited dataset, Power BI exhibited noticeably poor performance, with refresh times reaching up to five seconds. In contrast, the Real-Time Dashboard frequently achieved sub-100 millisecond query response times, with queries being executed in parallel. Performance when using Databricks was slightly worse than the Real-Time Dashboard, likely due to its reliance on external blob storage rather than integrated storage, but it still outperformed Power BI substantially.

When considering functionality, particularly the alerting system, Databricks provided the most robust solution. Alerts can be configured dynamically within notebooks, to trigger actions as new data arrives, and is not restricted to statically defined thresholds. While Microsoft Fabric also supports notebooks, the current implementation ingests data through its "Data Pipeline" tool directly into Eventhouse. This architecture restricts dynamic alert management, instead it requires thresholds to be manually defined within the Real-Time Dashboard interface. Unlike

Databricks, where alert thresholds can reference other data columns (for example, setting a threshold dynamically based on values in another column), Fabric's alerts are static. Additionally, Fabric's system lacks the ability to set alerts on tables directly, which means that alerts must be set manually on individual sensors, which further limits the flexibility.

Overall, we believe Databricks to be the preferred platform for this project, closely followed by the Real-Time Dashboard in Microsoft Fabric. Databricks offers nearly the same performance and scalability as the Real-Time Dashboard, but provides far greater customization, especially for alert configuration. Moreover, being an independent platform rather than an end-to-end solution like Microsoft Fabric, Databricks can be integrated with additional complementary tools such as Snowflake and Grafana if its native dashboard capabilities should prove insufficient or unintuitive.

Nonetheless, Microsoft Fabric remains a viable option, particularly for projects with less demanding alerting requirements. It benefits from being a fully integrated solution with a smoother learning curve and slightly better raw performance. However, the current alert system's limitations are a significant drawback. We view it as a work in progress that Microsoft is likely to improve over time, as enhanced alert functionality would greatly strengthen the platform's suitability for projects like this one.

5.2 Machine learning

For the machine learning component of the project, it was determined that predicting RUL through estimating lab data, e.g. MPC and particle counts, was not feasible given the available data. The model predictions were not sufficiently accurate and did not consistently capture the trend. There are two primary issues contributing to this limitation: First, the current sensor suite are not strongly correlated to the condition of the fluid, thus making them unreliable proxies for estimating RUL. Second, there is a data problem: The dataset lacks the volume required to support generalizable modeling. This challenge is twofold: not only are there too few operational cycles to effectively train a robust model, but the associated lab data for the sensor data is also very limited.

A model that aims to estimate RUL should ideally be trained on data for many cycles to allow it to learn generalized behavior rather than memorizing patterns from a small sample of data. In addition, each of these cycles should include more frequent lab measurements, preferably on a weekly basis, to accurately track the condition of the fluid and reduce reliance on interpolated targets. To facilitate this, future work of the project should focus on integrating sensors that more directly measure fluid condition, and data collection should encompass around 20 to 30 complete operational cycles, alongside more frequent lab data samples, to enable meaningful modeling.

Regarding the sensor forecasting component of this project, it was observed that per-

formance is generally acceptable when ample historical data is available, although it fails to capture large deviations in the data. However, during the dashboard development when limited streaming data was accessible, limitations became apparent, particularly in the model's ability to accurately capture and track fluctuations and sudden changes, for instance when the system is shut down unexpectedly. These shutdowns are inconsistent, sometimes the system stops on a Thursday and resumes on Friday, other times it remains off throughout the weekend, other times it's not powered down during the week at all. This erratic operational pattern makes it harder for model to detect or learn any clear seasonality, which is important for forecasting accuracy using Prophet. As a result, short-term trend predictions can deviate significantly from actual sensor readings, which can potentially lead misinterpretation or loss of trust in the results.

Although only Prophet was used in this project for sensor forecasting, based on a request from SKF subject matter experts, other machine learning techniques, including neural networks, could potentially be explored if more data had been available. However, the broader question remains: is forecasting sensor trends truly valuable in this context? From our perspective, the answer is nuanced. While it is ultimately SKF's decision on how beneficial such insights are, we believe that forecasting sensor values may not offer significant added value. There are two main reasons for this:

1. **Inconsistent Predictive Capability:** With the current modeling setup, forecasts are often unreliable, particularly when it comes to capturing system shutdowns or operational irregularities, seeing as the system is frequently powered down in unpredictable manners. For example, if the flow rate drops to a new stable level, the model sometimes incorrectly predict that it will keep falling, at least until more data becomes available. This inconsistency can reduce practical usefulness and trust in model outputs.

2. **Redundancy with Visual Inspection:** The sensor trends that the model attempts to project can often be inferred just by visually inspecting the data on the dashboard. For example, if average sensor values is slowly declining over the span of weeks or months, a separate forecast line which shows the same downward trend projected into the future may not necessarily provide additional insights.

We still believe that incorporating predictive capabilities, particularly RUL estimates, in the dashboard is a valuable pursuit, provided more data becomes available. However, we do not consider forecasting sensor values to offer the same level of benefits. In this project, sensor value forecasting was explored mostly as an alternative modeling approach once it became clear that the existing data was insufficient for accurate RUL prediction. This allowed us to explore a different form of predictive analytics and incorporate it into the dashboard. While this approach was technically interesting, its practical value is deemed by us to be quite limited given the operational realities.

One exception to this is the pressure sensor, because it is set as the control variable. Its readings remain tightly clustered around a defined setpoint, which makes long-

term trend forecasting more predictable. However, even here we don't know if much insight can be gained from forecasting the pressure readings, because we reason that if an unexpected event were to happen that would cause pressure to spike or drop, for example, a sudden blockage in the piping, the change would occur too rapidly for any long-term forecast model to anticipate it and would already have happened by the time the model "realizes" something is wrong. On the other hand, long term forecasting is not very meaningful for the pressure sensor either, since it being the control variable, it always stays within a tight range under normal operating conditions.

5.3 Future work

This project was an early attempt to combine digital tools and sustainability goals for the oil filtration system through dashboard development and predictive analytics. From this phase, we have uncovered several key insights:

Microsoft Fabric has proven to be a viable platform for monitoring the system, although its current alert system presents notable limitations that may affect real-time responsiveness and dynamic alerting capabilities. If more advanced alerting capabilities are of highest priority, we recommend using Databricks instead.

A significant challenge lies in accurately estimating the RUL of the filtration system. This difficulty comes from both insufficient data volume, as there are not enough operational cycles or lab measurements, and data quality issues, since the existing sensors do not effectively capture the fluid condition parameters needed for reliable RUL prediction. To address these challenges and guide the continuation of the project, we propose the following actions for SKF:

- **Enhance Sensor Deployment:** Introduce sensors that better represent the fluid condition indicators (i.e., MPC and particle counts). Recommended sensor types could include particle counters, viscosity sensors, dielectric sensors, or optical sensors. Further work would be required to evaluate these types of sensors, but we believe that these types of sensor could act as more accurate proxies for fluid condition than the current sensors.
- **Expand and Improve Data Collection:** Initiate a data collection project involving multiple oil filtration systems, capturing data throughout entire filter change cycles. This should include associated laboratory measurements, preferably with increased sampling frequency.

Regarding dashboard and visualization tools, if the solutions evaluated in this project are found to be insufficient or lack critical functionality, a different approach would be to integrate a platform like Azure Databricks with specialized visualization tools, e.g. Grafana, which is already in use in other parts of SKF. This hybrid solution combines Databricks' powerful data processing and alerting capabilities with more advanced features and customization of a dedicated visualization tool in order to offer a better user experience.

We believe the next steps outlined above will help advance the project by strength-

ening predictive analytics, thus enabling true condition-based monitoring and improving both sustainability and operational efficiency.

5.3.0.1 Additional considerations

In addition to continuing the project with respect to predictive modeling through AI and dashboards, we offer two additional suggestions to the current process that can be implemented immediately, without the need for extensive data collection projects or the installation of new sensors. These measures can be deployed immediately, or alongside a long-term machine learning and dashboard project, and will help streamline both filter change maintenance and data management. These suggestions are: *Scheduled Maintenance Protocols* and *Centralized Laboratory Data Integration*.

Scheduled Maintenance Protocols:
Currently, filter replacements rely on manual review of lab results, which is time consuming and dependent on consistent access to laboratory measurements. While the upper limit for filter duration is set at 12 months, this guideline is not always followed in practice. As more oil filtration systems are introduced into production, maintaining compliance with these intervals will become increasingly difficult. As an alternative to manual review of lab data and machine learning, we propose introducing a simpler, maintenance schedule set by historical system behavior and/or manufacturer guidance. This maintenance strategy can be based on:

- A standard 12-month replacement cycle as a baseline.
- A more conservative 8 to 10-month interval.
- An average lifespan calculated from existing maintenance records.

By configuring this logic within the analytics platform (Microsoft Fabric or Databricks), the system can automatically determine upcoming maintenance dates, send advance reminders (e.g., 30 days in advance), and track compliance. This reduces the risk of failures or quality issues caused by filters running too long, while at the same time streamlining maintenance scheduling.

Centralized Laboratory Data Integration:
Currently, lab measurement results are stored as local Excel files and reviewed manually. To complement this process and support future analytics, we propose standardizing and storing lab measurements in a centralized database, for example in Microsoft Fabric or Databricks. This would involve:

- Defining a consistent data template with fields for key parameters (e.g., MPC, particle counts), as well as metadata (e.g., machine ID, sample date, filter type).
- Automating uploads of this data, for example, through APIs or scheduled batch imports, into a centralized database.
- Integrating lab data metrics into the existing dashboard(s) to provide additional insights.

A centralized database for lab measurements will streamline routine inspections, offer a scalable, cloud-based solution for managing laboratory data across multiple oil filtration systems, and lay the groundwork for future machine learning projects

by ensuring that all relevant data is clean, consistent, and accessible.

By implementing these two relatively simple suggestions outlined in this section, SKF can immediately enhance maintenance consistency, reduce the risk of extended filter usage, and establish a foundation for future predictive analytics projects. These two suggestions can be implemented relatively easily to enhance the existing process.

6
Conclusion

The aim of this thesis was to explore how real-time monitoring dashboards combined with predictive analytics could support condition-based maintenance in industrial fluid systems. Two cloud-based platforms, Microsoft Fabric and Databricks, were used to develop and evaluate dashboards that visualize sensor data and provide early warnings based on machine learning predictions. While the dashboards were successfully implemented and had a good potential for real-time transparency, the predictive models had limited performance. This was mainly due to challenges such as limited historical data, limited lab data, and weak correlations between sensor readings and actual filter conditions. This means that while trend forecasting was possible, accurately estimating the remaining useful life of the filter proved to be more difficult.

Despite these limitations, the project provided valuable insights into integrating data pipelines, machine learning, and visualization for fluid management at SKF. For future work, improving data quality and expanding the dataset, especially with more labeled maintenance events, would be key steps toward better model performance. For future work, increasing the data volume and the frequency of laboratory sampling, as well as exploring additional sensors to integrate in the oil filtration systems, would be key steps toward improving model performance.

Additional research into domain-specific features may also improve the reliability of predictions and make the dashboards more actionable in actual operation scenarios. Since the project was limited in time and divided between dashboard development and predictive analytics, neither area could receive full attention. This means that there may be additional features or information that could serve as better proxies for the laboratory data, which might be identified through deeper analysis. However, we are confident that the limitations in the predictive capability stems from the data rather than the modeling approach.

Bibliography

Alpaydin, E. (2021). *Introduction to machine learning* (4th). MIT Press. Retrieved April 21, 2025, from https://research.ebsco.com/c/lu54te/search/details/j63i7tsfpz

Aves, M. (2025). A review paper on data science & its fundamentals.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. Retrieved April 15, 2025, from https://research.ebsco.com/c/lu54te/search/details/orofrgk5wz

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, *7*, e623. Retrieved April 23, 2025, from https://doi.org/10.7717/peerj-cs.623

Cooksey, R. W. (2020). *Illustrating statistical procedures: Finding meaning in quantitative data*. Retrieved April 9, 2025, from https://doi.org/10.1007/978-981-15-2537-7_5

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232. Retrieved April 20, 2025, from https://research.ebsco.com/c/lu54te/search/details/4of57t5ttr

Geissdoerfer, M., Savaget, P., Bocken, N. M. P., & Hultink, E. J. (2017). The circular economy – a new sustainability paradigm? *Journal of Cleaner Production*, *143*, 757–768. Retrieved April 25, 2025, from https://www.sciencedirect.com/science/article/abs/pii/S0959652616321023

Han, J., Kamber, M., & Pei, J. (2011). *Data mining concepts and techniques* (3rd). Morgan Kaufmann. Retrieved March 18, 2025, from https://research.ebsco.com/c/lu54te/search/details/z7jq6m5zer

Hartvigsen, T., Gerych, W., Thadajarassiri, J., Kong, X., & Rundensteiner, E. (2022). *Stop&hop: Early classification of irregular time series*. Retrieved March 27, 2025, from https://arxiv.org/abs/2208.09795

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer. Retrieved March 29, 2025, from https://research.ebsco.com/c/lu54te/search/details/pnakopdb25

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*(7), 1483–1510. Retrieved April 5, 2025, from https://www.sciencedirect.com/science/article/pii/S0888327005001512

Korstanje, J. (2021). The prophet model. In *Advanced forecasting with python* (pp. 253–272). Springer. Retrieved May 5, 2025, from https://link.springer.com/chapter/10.1007/978-1-4842-7150-6_19

Lee, J., Bagheri, B., & Kao, H. A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18–23. Retrieved March 22, 2025, from https://www.sciencedirect.com/science/article/abs/pii/S221384631400025X

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications, and open research issues. *Journal of Industrial Information Integration*, *6*, 1–10. Retrieved March 30, 2025, from https://www.sciencedirect.com/science/article/pii/S2452414X17300043

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to linear regression analysis*. Wiley. Retrieved April 12, 2025, from https://research.ebsco.com/c/lu54te/search/details/ujnaqvls6f

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. In L. Liu & M. T. Özsu (Eds.), *Encyclopedia of database systems* (pp. 532–538). Springer. Retrieved April 28, 2025, from https://doi.org/10.1007/978-0-387-39940-9_565

SKF Group. (2021, June). *A clean start for bearing performance* [Retrieved May 1, 2025, from https://evolution.skf.com/a-clean-start-for-bearing-performance/]. https://evolution.skf.com/a-clean-start-for-bearing-performance/

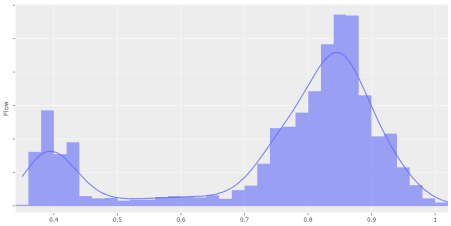
Skiena, S. S. (2017). *The data science design manual*. Retrieved May 12, 2025, from https://link.springer.com/book/10.1007/978-3-319-55444-0

Studer, M., Ritschard, G., Gabadinho, A., & Müller, N. S. (2011). Discrepancy analysis of state sequences. *Sociological Methods & Research*, *40*(3), 471–510. Retrieved May 8, 2025, from https://www.researchgate.net/publication/227575298_Discrepancy_Analysis_of_State_Sequences

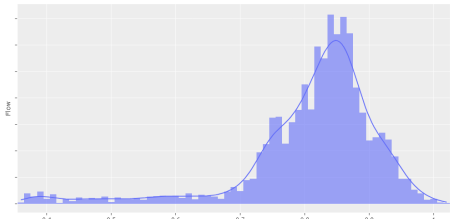
Wirth, R., & Hipp, J. (2002). *Crisp-dm: Towards a standard process model for data mining*. Retrieved March 15, 2025, from https://cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf

A

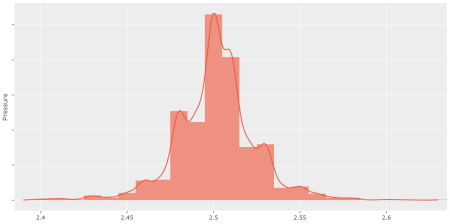
Data distribution



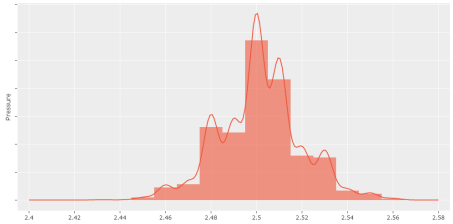
(a) Flow (with outliers)



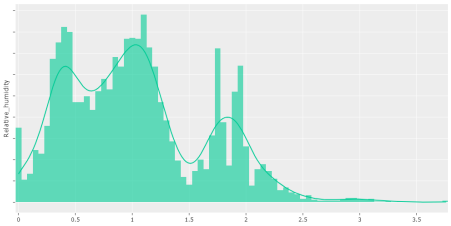
(b) Flow (no outliers)



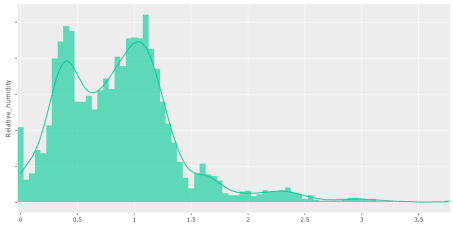
(c) Pressure (with outliers)



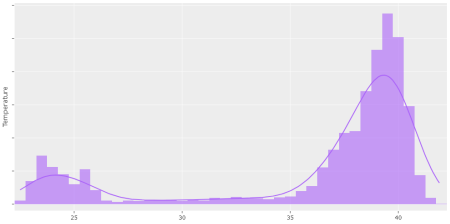
(d) Pressure (no outliers)



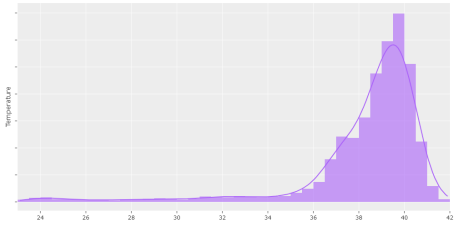
(e) Relative humidity (with outliers)



(f) Relative humidity (no outliers)

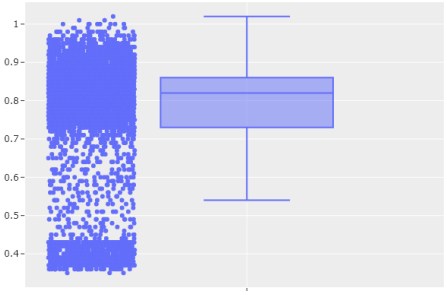


(g) Temperature (with outliers)

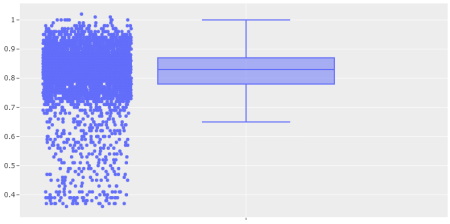


(h) Temperature (no outliers)

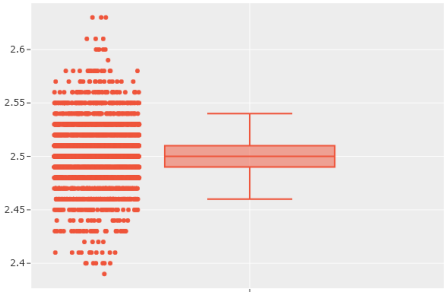
Figure A.1: Kernel Density Estimates (KDE) for all sensor readings, shown with and without outliers.



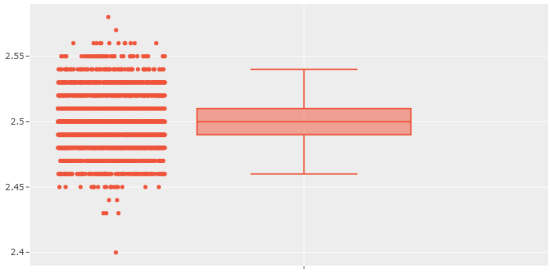
(a) Flow (with outliers)



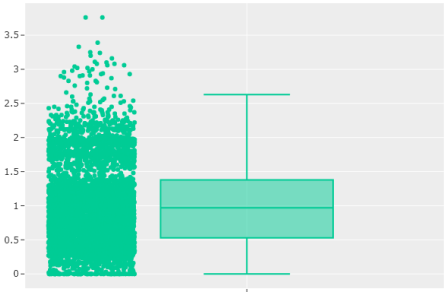
(b) Flow (no outliers)



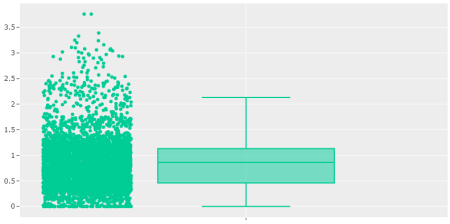
(c) Pressure (with outliers)



(d) Pressure (no outliers)

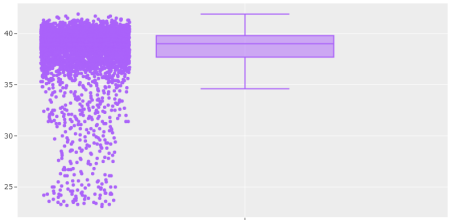


(e) Relative humidity (with outliers)



(f) Relative humidity (no outliers)

(g) Temperature (with outliers)



(h) Temperature (no outliers)

Figure A.2: Boxplots of all sensor readings, shown with and without outliers.

B

MPC prediction plots

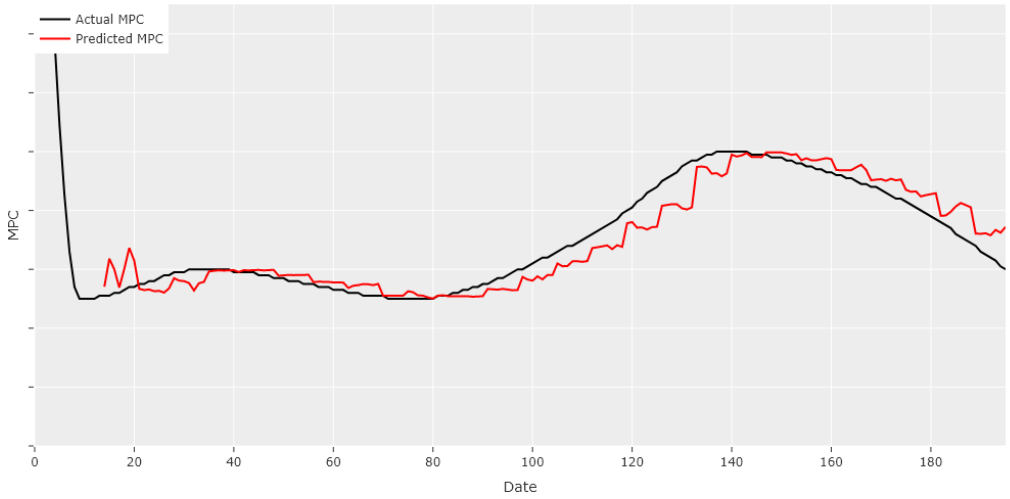


Figure B.1: MPC prediction using Random Forest Regression with sliding window approach

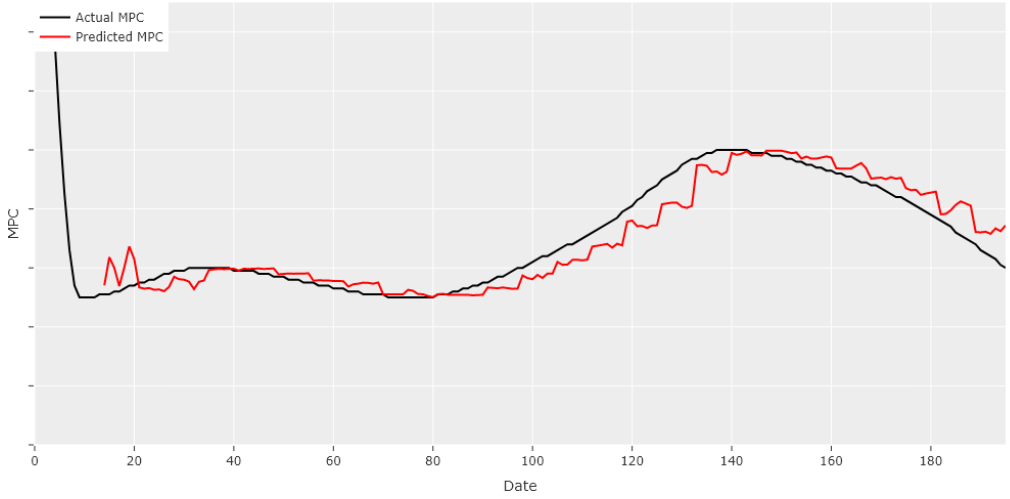


Figure B.2: MPC prediction using Random Forest Regression with expanding window approach

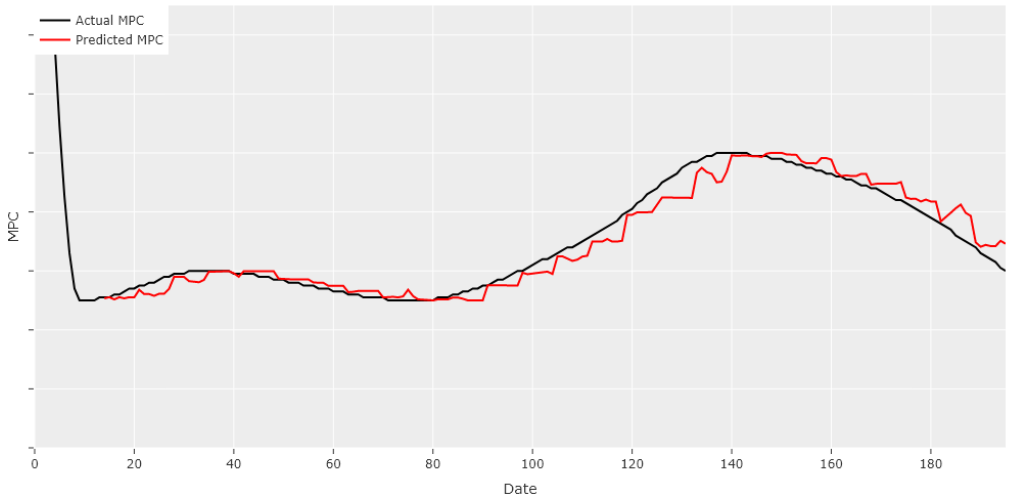


Figure B.3: MPC prediction using Extreme Gradient Boosting with sliding window approach

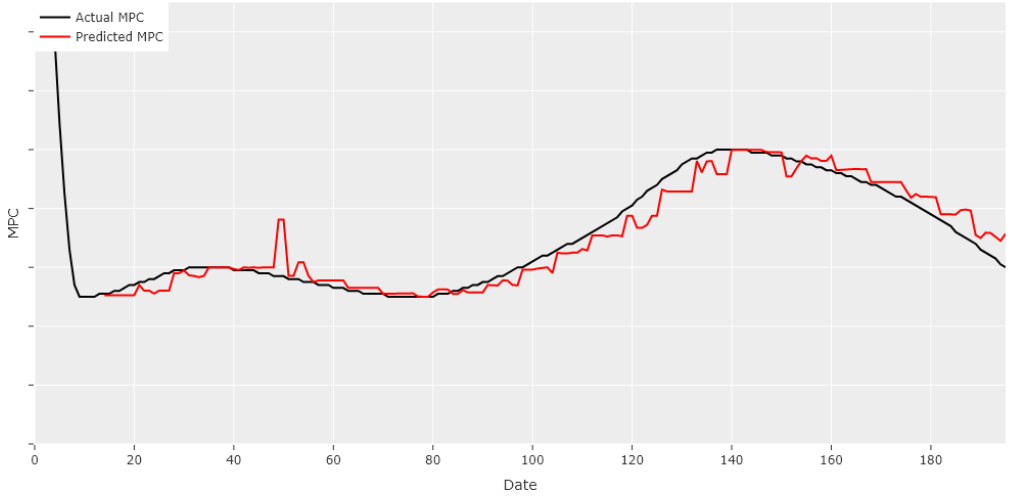


Figure B.4: MPC prediction using Extreme Gradient Boosting with expanding window approach

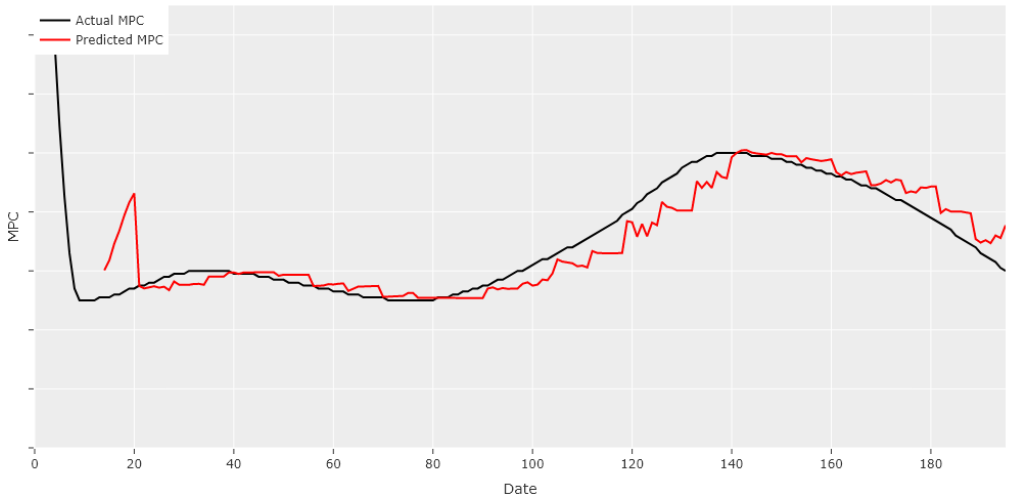


Figure B.5: MPC prediction using Support Vector Regression with sliding window approach

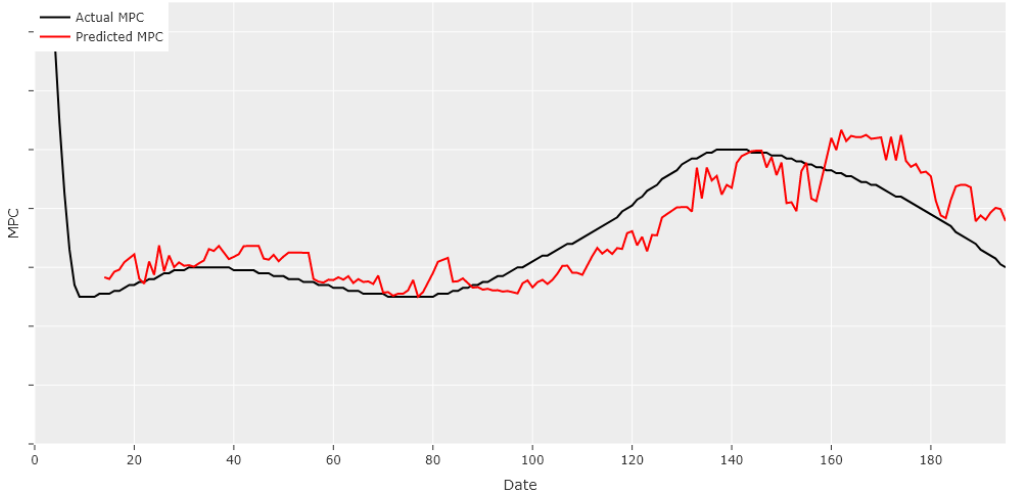


Figure B.6: MPC prediction using Support Vector Regression with expanding window approach

C

Particle count(>4µm) prediction plots

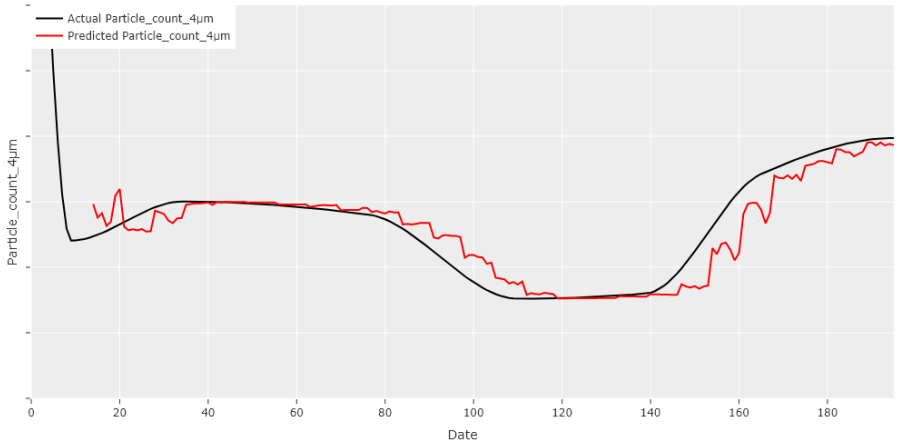


Figure C.1: Particle count(>4µm) prediction using Random Forest Regression with sliding window approach

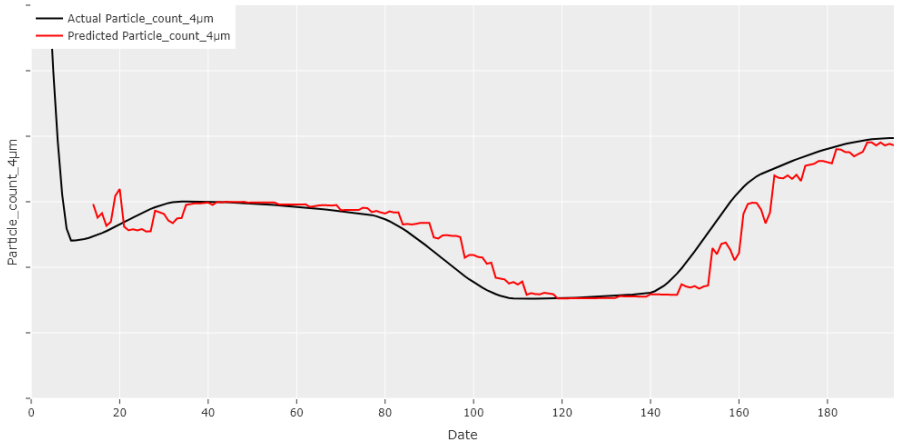


Figure C.2: Particle count(>4µm) prediction using Random Forest Regression with expanding window approach

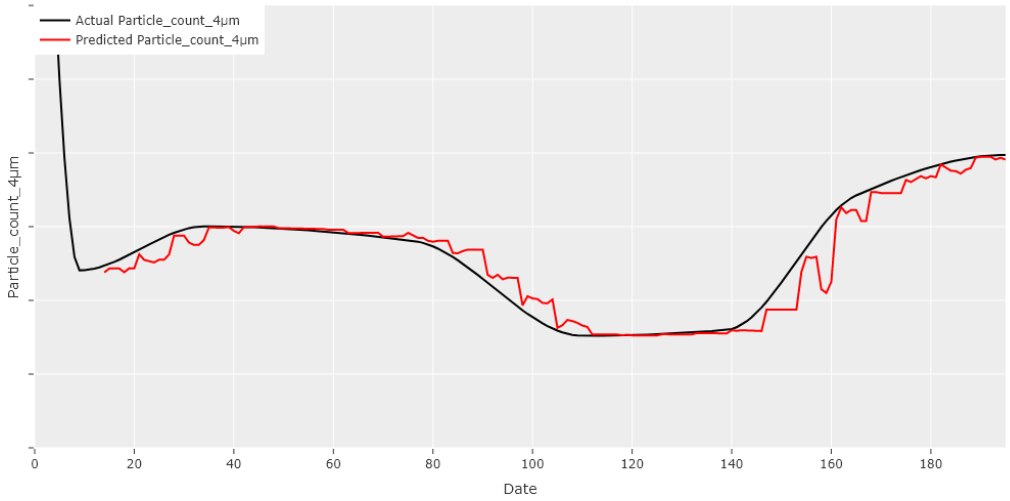


Figure C.3: Particle count(>4µm) prediction using Extreme Gradient Boosting with sliding window approach

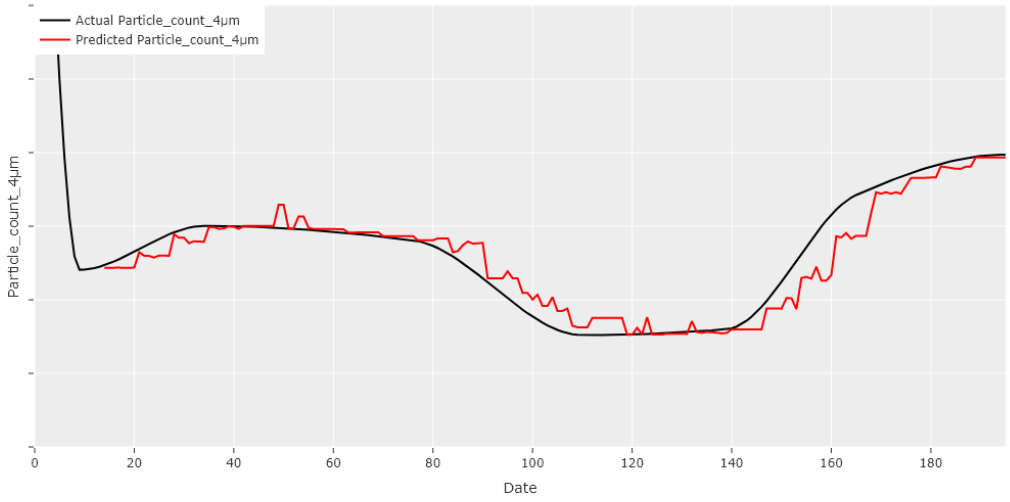


Figure C.4: Particle count(>4µm) prediction using Extreme Gradient Boosting with expanding window approach

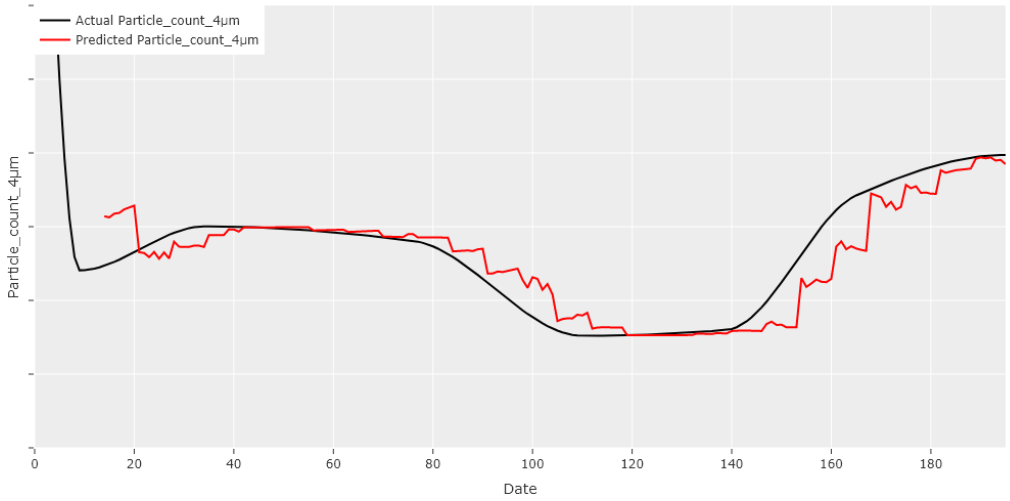


Figure C.5: Particle count(>4µm) prediction using Support Vector Regression with sliding window approach

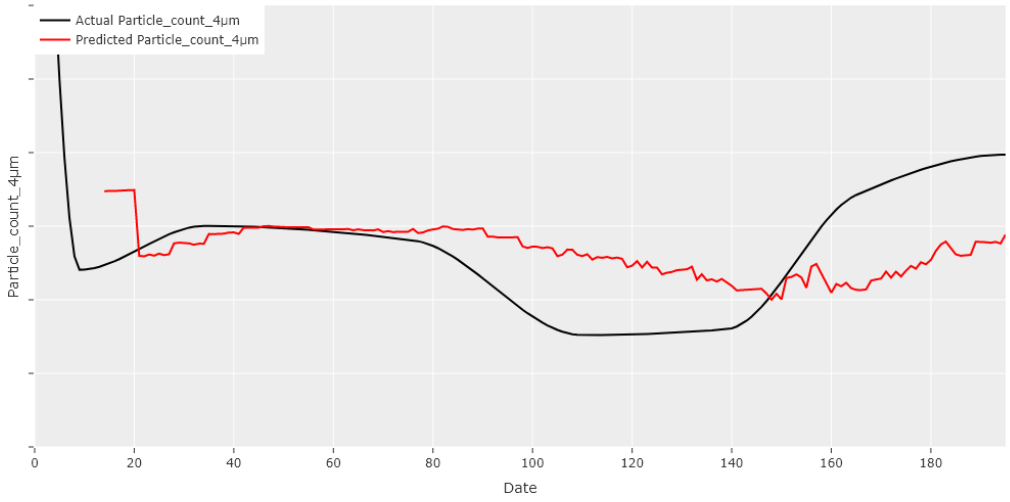


Figure C.6: Particle count(>4µm) prediction using Support Vector Regression with expanding window approach

D

Particle count(>6µm) prediction plots

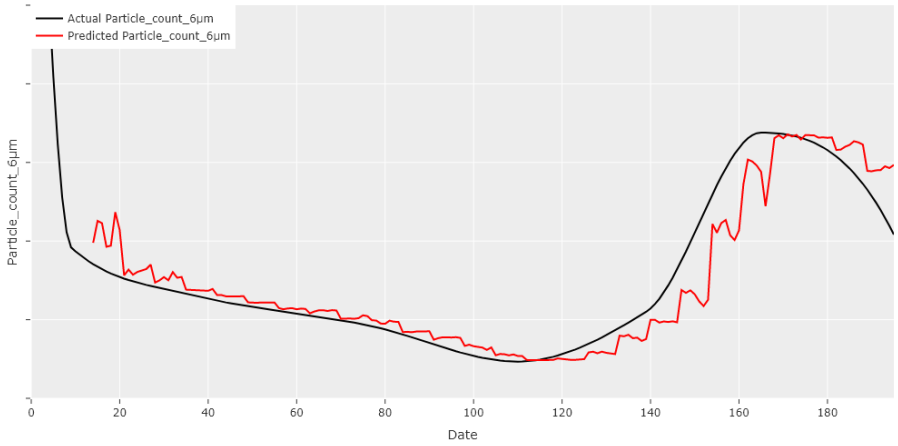


Figure D.1: Particle count(>6µm) prediction using Random Forest Regression with sliding window approach

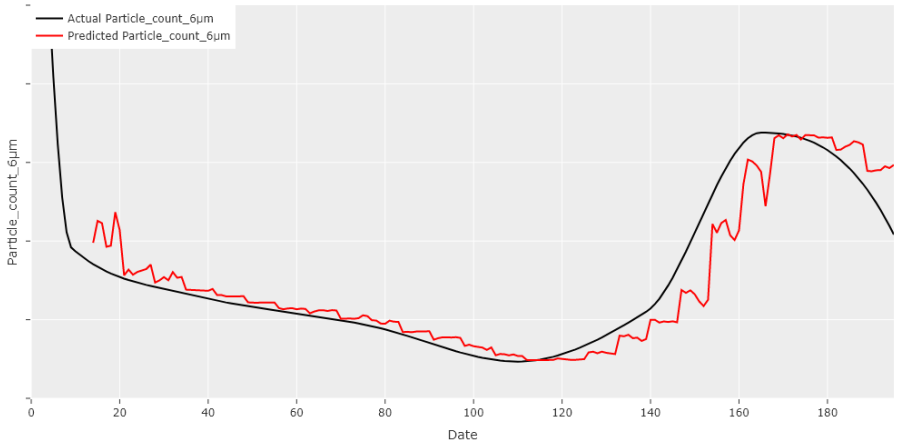


Figure D.2: Particle count(>6µm) prediction using Random Forest Regression with expanding window approach

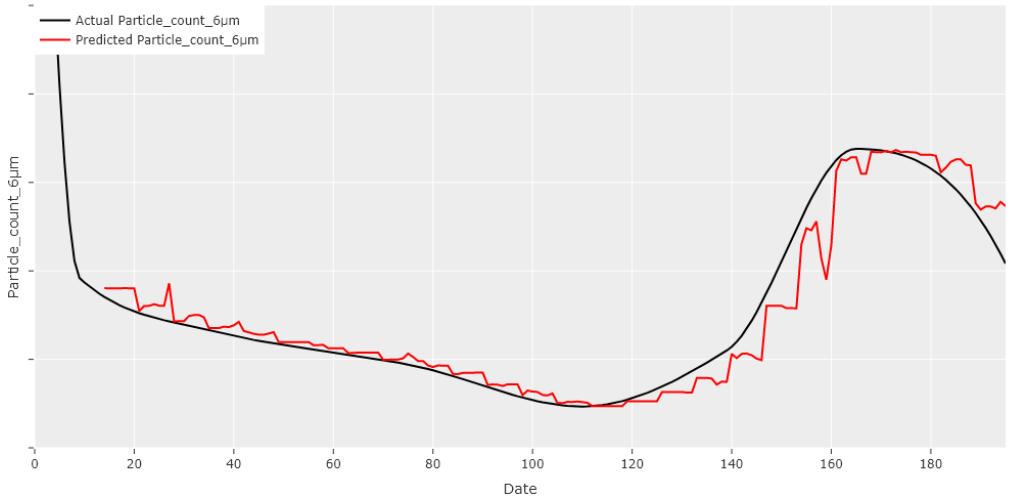


Figure D.3: Particle count(>6µm) prediction using Extreme Gradient Boosting with sliding window approach

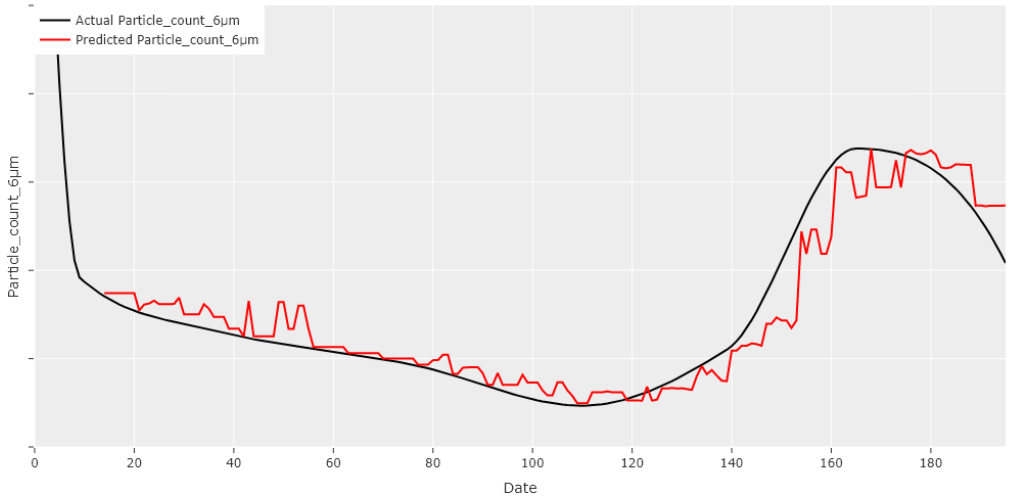


Figure D.4: Particle count(>6µm) prediction using Extreme Gradient Boosting with expanding window approach

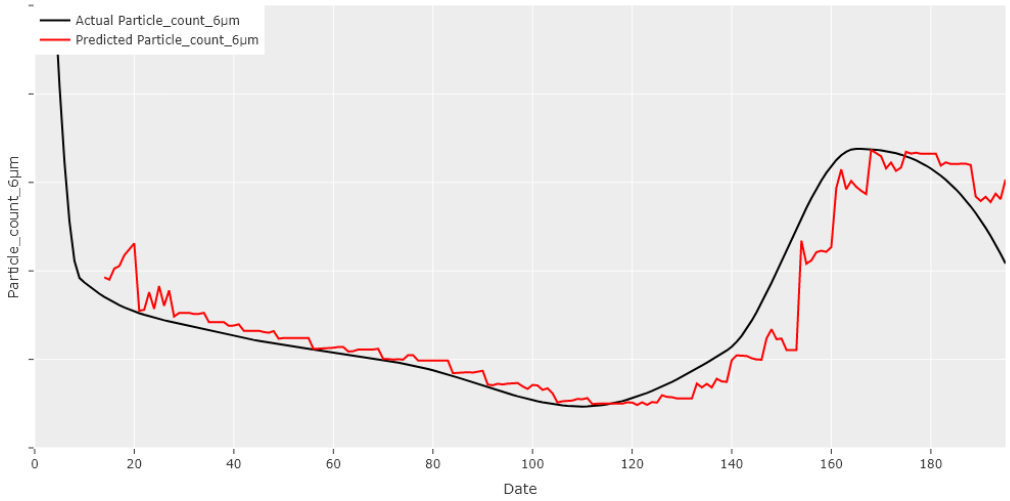


Figure D.5: Particle count(>6µm) prediction using Support Vector Regression with sliding window approach

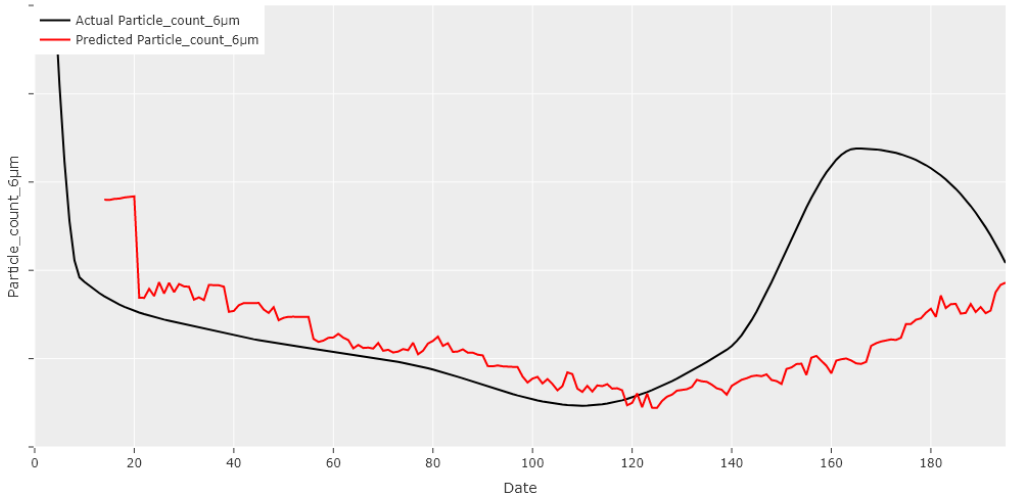


Figure D.6: Particle count(>6µm) prediction using Support Vector Regression with expanding window approach

E

Particle count(>14µm) prediction plots

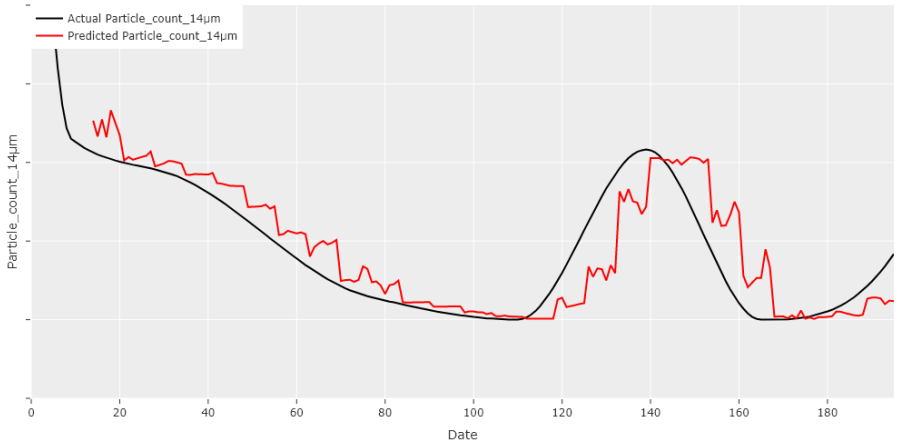


Figure E.1: Particle count(>14µm) prediction using Random Forest Regression with sliding window approach

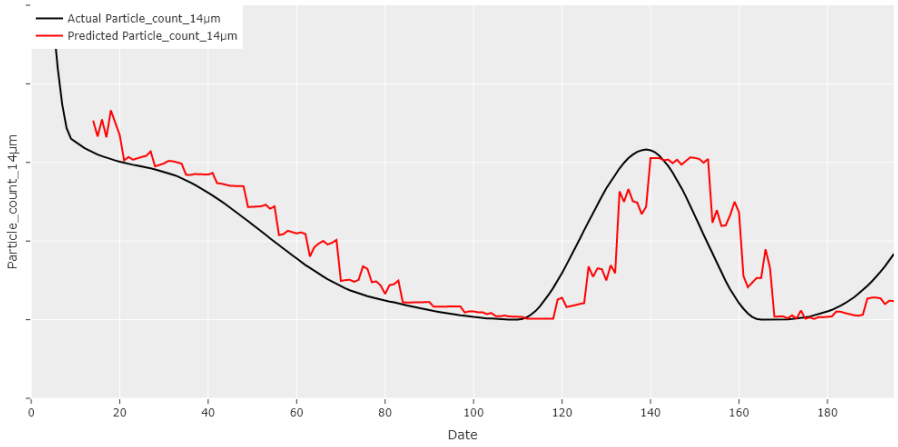


Figure E.2: Particle count(>14µm) prediction using Random Forest Regression with expanding window approach

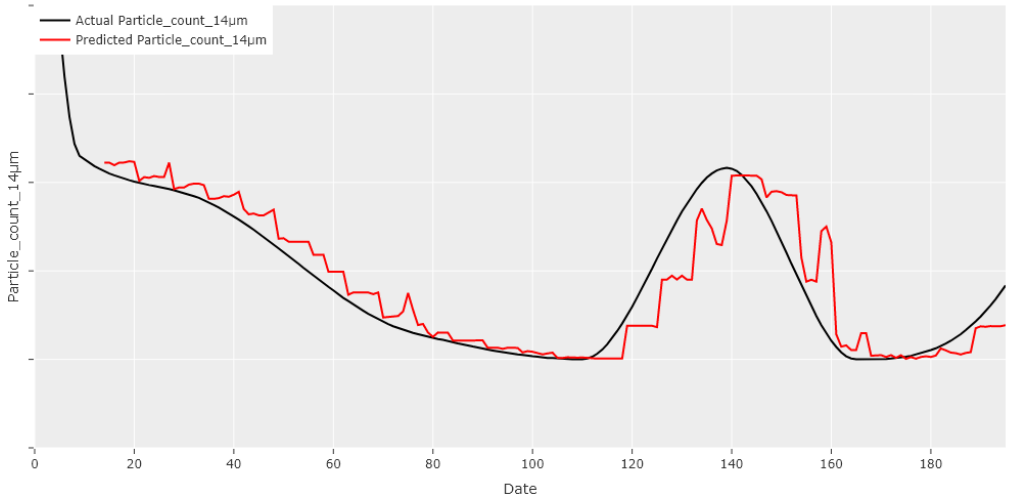


Figure E.3: Particle count(>14µm) prediction using Extreme Gradient Boosting with sliding window approach

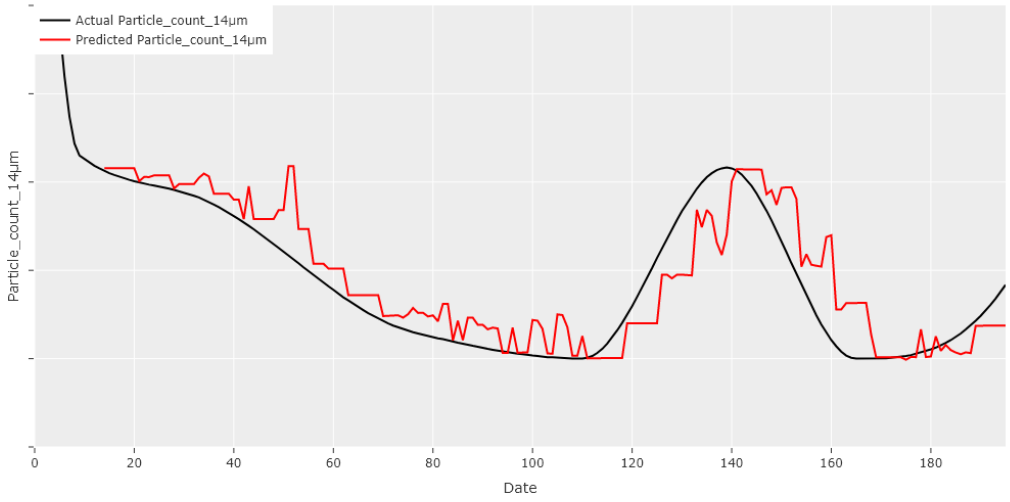


Figure E.4: Particle count(>14µm) prediction using Extreme Gradient Boosting with expanding window approach

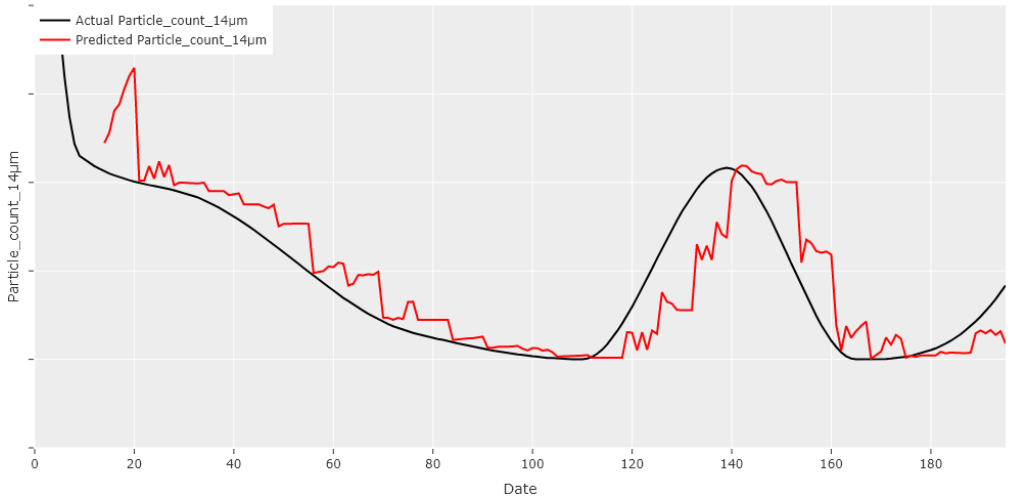


Figure E.5: Particle count(>14µm) prediction using Support Vector Regression with sliding window approach

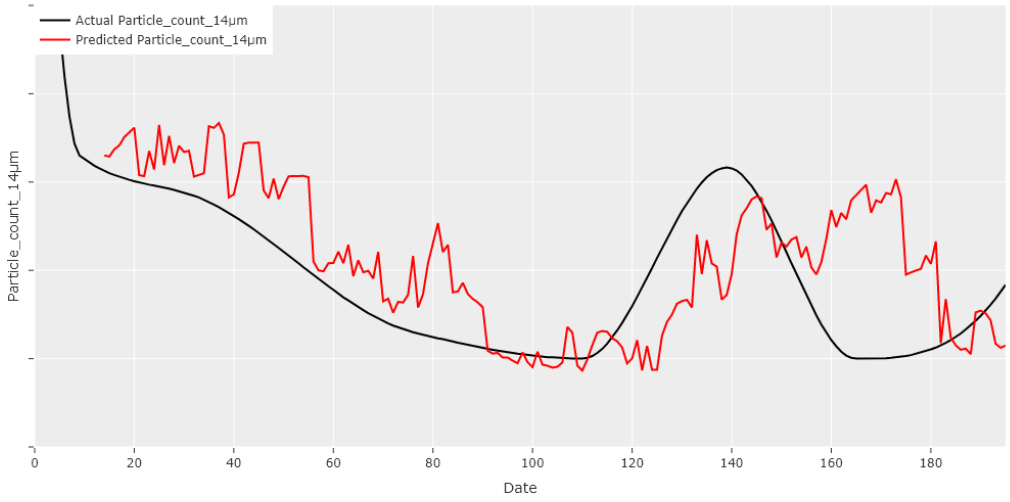


Figure E.6: Particle count(>14µm) prediction using Support Vector Regression with expanding window approach

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY