Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

BHAGYA LAKMINI KARUNARATHNE RANBANDI DEWAGE

Detecting and Explaining Performance Differences and Cycle Time Variability in Industrial Robots Using Multivariate Machine Learning Models

DEGREE PROGRAMME IN DATA ENGINEERING
2025

ABSTRACT

Ranbandi Dewage Bhagya Lakmini Karunarathne: Detecting and Explaining Performance Differences and Cycle Time Variability in Industrial Robots Using Multivariate Machine Learning Models
Bachelor’s thesis
Data Engineering
July 2025
Number of pages: 112

Modern material handling relies heavily on automation, so it is important to understand why some robots complete tasks faster and more efficiently than others. This thesis looks at performance differences and cycle time variation in customized gantry robots built by Cimcorp Oy. The analysis uses machine learning models and real-world data collected from eight robots over a long period of operation.

The full dataset was divided into eight groups based on different working conditions, and two operating states were chosen for detailed study. Each dataset included numeric signal data from individual robot cycles. Time-based features were removed to focus on internal signals, and all signal names were anonymised to protect the client’s confidentiality.

A supervised XGBoost regression model was used to predict cycle times in both selected conditions. The model gave high predictive accuracy, with R^2 values of 0.92 and 0.77. To understand which signals had the biggest impact on cycle time, SHAP (SHapley Additive exPlanations) and Sobol sensitivity analysis were used. PCA (Principal Component Analysis) was also used to find patterns in robot behaviour and show how different robots performed under the same conditions.

These results show that machine learning, combined with explainable AI methods, can help predict and understand differences in robot performance. The model's reliability was further confirmed by testing on new unseen data, and an additional efficiency ranking method was developed using predicted versus actual cycle times to highlight robot level performance differences. This approach supports improved monitoring, decision making, and operational efficiency in industrial automation systems.

Keywords: Industrial automation, gantry robots, Cimcorp Oy, robot performance, cycle time prediction, machine learning, XGBoost, SHAP, Sobol sensitivity analysis, PCA, explainable AI

PREFACE

Working on this thesis has been one of the most meaningful and challenging experiences of my studies in Data Engineering at Satakunta University of Applied Sciences (SAMK). It brought together everything I have learned over the past few years and gave me the opportunity to apply that knowledge to a real-world industrial problem.

The topic, which focuses on analysing performance differences and cycle time variability in customized gantry robots, was developed in close collaboration with Cimcorp Oy. From the very beginning, their team was actively involved in the process. We held weekly meetings where I received feedback, suggestions, and motivation that consistently pushed the work forward. That continuous support helped me maintain focus and aim for a higher quality of work than I could have achieved on my own.

This project gave me a realistic view of how data is used in the industrial world not just to build models, but to solve practical problems, improve systems, and support decision-making. It also helped me grow personally. I learned how to manage long-term work, communicate technical findings clearly, and stay motivated through complex challenges.

I am deeply grateful to Cimcorp Oy for their trust, collaboration, and support throughout the project. I also want to thank my thesis supervisor at SAMK for providing clear guidance and encouragement whenever needed, and my data analytics teacher, whose advice helped sharpen the analytical side of the work.

Looking back, this project has been a key step in shaping my identity as a data engineer. I leave this experience with more confidence, a deeper understanding of real-world data work, and excitement for what lies ahead.

CONTENTS

LIST OF SYMBOLS AND TERMS

Term / Symbol	Meaning
ta	Cycle time. The time it takes for a robot to complete one full work cycle.
Anonymized input features	gu, gg, grbr, gk, gs, ga, gp, gw, gamp, tw, tq, tf, ex, ev, ey, ew, eo, eh, ei, el, em, eq, en, es, gx, gy, ep, ej, tc, te, td, tm, tp, ts, ed These represent internal signals, sensor values, actuator states, or system parameters. Their meanings are hidden to protect client confidentiality.
ghi	A derived feature calculated as gi + 40, used in place of two similar original signals.
floor_pick	A boolean flag showing whether the robot performed a floor-picking task.
simultaneous_operation	A boolean flag indicating if the robot executed multiple actions during the same cycle.
ev_positive	A derived flag showing whether the ev feature was greater than zero.
all_true	A configuration where floor_pick, simultaneous_operation, and ev_positive are all True. Represents synchronized task execution.
simul_false	A configuration where floor_pick and ev_positive are True, but simultaneous_operation is False. Represents more independent task execution.
XGBoost	A machine learning algorithm used to predict cycle time and evaluate feature importance.
SHAP	SHapley Additive exPlanations. Explains how each input feature affects the model prediction.
Sobol indices	Measures used in global sensitivity analysis to assess how much each input feature affects the output.
PCA	Principal Component Analysis. A method to reduce dimensionality and compare robot behaviours visually.
Change Point Detection	A technique for detecting when the robot's performance trend (cycle time) changes over time.
PELT	Pruned Exact Linear Time. An algorithm used for efficient change point detection.
ruptures	A Python library used to perform change point detection on time-series data.

1 INTRODUCTION

1.1 Background and Motivation

Modern factories and warehouses use robots to move items, handle materials, and complete tasks efficiently. These robots help reduce labour costs, improve safety, and increase productivity. One common type used in material handling is the gantry robot. These robots are built to move along fixed paths and can handle large or heavy items. Cimcorp Oy builds customized gantry robots used in different industries such as tire manufacturing and logistics.

A keyway to measure how well these robots perform is by looking at their cycle time. This means the time it takes for a robot to complete one full task. Shorter cycle times can lead to higher output. However, trying to make cycle times too short can also increase the wear and tear on the robot, or lead to energy waste. Faster cycles can reduce cost per product, but they also increase the risk of damage to key robot parts when the robot is handling heavier loads (Stuhlenmiller et al., 2021).

Machine learning is now being used in many areas of robotics to help improve performance. For example, time-series models have been used to predict forces in robotic machining. This approach helped track performance in real time, which is important in fast-changing industrial settings (Wu et al., 2024). These tools can also be used to estimate how long a task will take and find problems early.

However, applying these models to real industrial robots is still challenging. Robots may operate in different conditions, making it difficult to compare their performance. Also, machine learning models often act like black boxes, meaning their predictions can be hard for engineers to understand. Many

companies want tools that are not only accurate, but also clear and easy to explain.

This thesis aims to solve these problems by using real data collected from eight Cimcorp gantry robots. It uses a model called XGBoost to predict cycle time and explainable AI tools like SHAP and Sobol to show which robot signals are most important. The goal is to find out why some robots perform better than others and help improve decision-making for robot settings and maintenance.

By understanding these differences, companies can better monitor their robots and make smarter choices to improve efficiency and reduce downtime.

1.2 Company Overview: Cimcorp Oy

Cimcorp Oy is a company based in Finland that designs and builds automation systems. It was founded in 1975 and has its main office in Ulvila, near the city of Pori (Cimcorp, 2025). Cimcorp focuses on making robotic systems that help companies move and manage goods more efficiently.

The company mainly works with customers in the grocery retail and tire industries. For example, it helps grocery warehouses move food quickly and accurately, and it supports tire manufacturers with fully automated handling systems. Cimcorp’s solutions are used in many parts of the world.

Cimcorp is part of a larger company called Murata Machinery, which is based in Japan (Cimcorp, 2025). This allows Cimcorp to work internationally while keeping its development and production in Finland. The company also provides support and maintenance for the systems it installs, helping customers keep their automation running smoothly for years.

Figure 1.1: Cimcorp's automated gantry robot system used for tire handling in a warehouse environment. The system is an example of the advanced industrial robotics developed by the company (Image source: Cimcorp Oy).

1.3 Research Objectives and Questions

Understanding performance variation in industrial robots is becoming increasingly important as automation becomes more complex and widespread. Robots often work for many years, during which their performance can change. These changes may be due to technical issues, wear and tear, changes in the working environment, or differences in robot design and configuration. Even when robots are installed in similar environments, they may not perform the same way. This makes it necessary to monitor performance carefully and look for the reasons behind such differences.

Several studies have highlighted the importance of analysing robot performance over time. Robot energy consumption and mechanical wear can increase with higher cycle speeds and payloads, which in turn affects overall resource efficiency (Stuhlenmiller et al., 2021). Additionally, robot assembly times can vary significantly, even under controlled conditions, due to hidden or

unobserved factors (LI et al., 2021). These findings suggest that simply tracking the average cycle time is not enough. Instead, multivariate data analysis is needed to capture deeper patterns. Tools like XGBoost and SHAP are useful in this context because they not only predict performance but also explain the reasons behind it (Guo et al., 2021; Kostov & Hristov, 2023).

The goal of this thesis is to apply machine learning methods to analyse long-term performance data from multiple industrial gantry robots at Cimcorp Oy. Each robot has collected thousands of data points over months or years. These include sensor values, actuator readings, and internal signals from every cycle. By examining these signals with advanced models, it is possible to detect when performance changes and explain why two robots may behave differently even under similar tasks.

The study aims:

- To detect and analyse changes in the performance of individual robots over long time periods using multivariate sensor data and machine learning.

- To identify and explain differences in performance between similar robots working under the same or comparable tasks.

- To find the key input features that best explain the detected changes or differences in robot cycle time.

Research Questions

Based on the above goals, this thesis aims to answer:

- How to detect and analyse changes in the performance of individual robots over long time periods using multivariate sensor data and machine learning.

- How to identify and explain differences in performance between similar robots working under the same or comparable tasks.

- How to find the key input features that best explain the detected changes or differences in robot cycle time.

1.4 Scope and Limitations

This thesis focuses on detecting and explaining performance differences in industrial robots using real operational data collected from Cimcorp Oy. The main output variable in this work is cycle time, which measures how long a robot takes to complete one work cycle. This was the only confirmed and labelled performance variable available in the dataset. Other potential output variables, such as energy consumption, stress levels, or mechanical wear, were not included due to limitations in the available data and confidentiality agreements.

The input data is made up of multivariate time-series signals gathered from robots during real operations. These signals reflect the robot's behaviour, environment, and task conditions. However, the exact names and technical details of these signals are anonymized for confidentiality. As a result, while the analysis uses all available features, specific signal labels or descriptions are not shown in the report.

The results of this study are based on data from a limited group of robots working in one company. Therefore, the findings may not apply directly to all types of industrial robots or other work environments. For example, robots used in welding, painting, or healthcare may behave differently and would require separate analysis. The conclusions are most relevant for robots working in material handling and warehouse automation, which are Cimcorp's focus areas.

In this study, the machine learning models used are based on patterns in the data. They can help show which input features are most linked to changes in performance. However, the models cannot prove clear cause-and-effect relationships. There may be other hidden factors, such as human operator actions, software updates, or maintenance schedules, that are not included in the data.

Another important limitation is that the dataset does not include labelled fault events, downtime records, or maintenance logs. This means that the goal of the study is to understand performance variation over time, not to predict failures or detect faults.

Despite these limitations, the thesis provides useful insights into how machine learning can be used to support robot performance monitoring and help engineers and managers understand what factors are most related to changes in cycle time.

Use of AI Tools:

During the preparation of this thesis, artificial intelligence (AI) tools were used to support writing and research. OpenAI's ChatGPT (version released March 14, 2024) helped with brainstorming, improving the clarity of technical explanations, organizing some sections, and refining grammar. ResearchRabbit was used to explore and track related academic literature. These tools were not used to generate original research results, conduct analysis, or replace the author’s critical thinking. All AI-assisted content was carefully reviewed and edited by the author to ensure academic integrity and originality. The author takes full responsibility for the final content.

1.5 Thesis Structure

This thesis is divided into seven main chapters.

Chapter 1 introduces the topic, explains the background and motivation, gives a short overview of the company Cimcorp Oy, presents the research objectives and questions, defines the scope and limitations of the study, and describes how the thesis is structured.

Chapter 2 reviews previous research and background literature. It covers topics such as industrial automation, gantry robot systems, performance metrics, machine learning in robotics, and explainable AI techniques like SHAP, Sobol sensitivity analysis, and Principal Component Analysis (PCA).

Chapter 3 explains the data used in this study and how it was prepared. It describes the operational data from Cimcorp robots, how the data was grouped into different operating conditions, how it was cleaned, and how features were selected and anonymized for analysis.

Chapter 4 presents the machine learning methods used. It explains the modelling pipeline, gives an overview of the XGBoost regression model, describes how the model was trained and evaluated, and shows the results of the performance predictions.

Chapter 5 focuses on explainability and pattern analysis. It uses SHAP to show which features were most important, applies Sobol analysis for sensitivity checking, uses PCA for dimensionality reduction, and visualizes how different robots behave under similar tasks.

Chapter 6 discusses the key findings, what they mean for real-world robot monitoring, the limitations of the methods, and suggestions for future work.

Chapter 7 gives the final conclusions. It summarizes the results, highlights the research contributions, and reflects on the work done.

At the end, the thesis includes a list of references and appendices with supporting information.

2 LITERATURE REVIEW

2.1 Industrial Automation and Material Handling

Industrial automation has become a major part of modern manufacturing and logistics. Automated systems help factories produce goods faster, more safely, and more efficiently. Material handling, which includes moving, storing, and managing materials inside a factory or warehouse, is one of the key areas where automation has made a big difference. Robots are widely used for handling parts, transporting items, and supporting other machines or humans in repetitive tasks.

In material handling, the use of robotic systems like gantry robots, robotic arms, and autonomous vehicles is increasing rapidly. Robotic compact storage and retrieval systems (RCS/RS) are growing in popularity due to their modular design and ability to scale based on demand (Trost & Eder, 2024). These systems often include a robot that performs pick-up and drop-off tasks within a grid structure, and their performance is measured using factors like cycle time, response time, and reliability.

Automation brings many benefits such as reduced labour costs, better product quality, and the ability to operate 24/7. However, the performance of these systems can vary depending on the robot, task, and conditions. Robot performance, including energy use and maintenance needs, should be considered over the whole life cycle (Stuhlenmiller et al., 2021). For example, robots that operate at faster cycle times may reduce cost per unit but could also require more maintenance and spare parts. While automation is helpful, it needs to be monitored and optimized for long-term efficiency to ensure sustained performance and value.

The importance of measuring and predicting robot performance has also led to the use of simulation and data-driven tools. Simulation-based prediction models are becoming more common, especially in tasks like robot welding,

where knowing the exact cycle time is important for planning and safety (Cho et al., 2024). These models help companies avoid guesswork and improve planning accuracy.

Industrial automation is also moving toward smarter systems that adapt and improve over time. One example is a method for robot condition monitoring using data from program-position cycles. This kind of predictive maintenance can detect problems before they cause breakdowns. In this way, the data collected from robots during normal operations can be used to improve performance and reduce downtime (Guo et al., 2021).

Optimizing the coordination and implementation of robots in factories can help reduce cycle times and improve efficiency (Spensieri et al., 2021). Good coordination between robots and clear task planning makes sure that resources are used well and no time is wasted during operations.

In summary, industrial automation plays a critical role in modern material handling systems. The use of robots helps companies handle more work with less effort, but it also brings new challenges in performance monitoring and optimization. Understanding how and why robot performance changes over time is important to get the most value from these automated systems.

2.2 Gantry Robots and Performance Metrics

Gantry robots are widely used in material handling, warehousing, and automated manufacturing environments because of their ability to move in large workspaces and perform repetitive pick-and-place or transport tasks. They are built on a fixed frame structure with linear movement along X, Y, and Z axes, making them suitable for high-precision and high-speed operations. In automated storage systems, for example, gantry robots often manage containers stacked in vertical racks, enabling fast and efficient storage and retrieval (Trost & Eder, 2024).

When measuring the performance of gantry robots, cycle time is one of the most important metrics. Cycle time refers to the time it takes for a robot to complete one full task or cycle, such as picking an item and placing it in a new location. Shorter cycle times usually mean better efficiency, but reducing cycle time too much can increase wear and maintenance needs (Stuhlenmiller et al., 2021). Their research showed that robots performing short cycle tasks with higher payloads had increased chances of joint failures, especially over long operating periods. This highlights the need to balance speed and reliability when optimizing gantry robot operations.

Performance metrics in industrial robotics often include:

- Cycle time
- Payload handling
- Accuracy and repeatability
- Energy consumption
- Downtime or maintenance frequency

In the context of robot welding, accurate cycle time prediction is important not only for planning but also for maintaining consistent production quality (Cho et al., 2024). Their simulation-based approach helps estimate cycle times more reliably, which is essential in high-precision tasks like spot welding. These kinds of approaches are also valuable for gantry systems that follow fixed movement patterns and deal with changing payloads and timing constraints.

Another important factor is trajectory planning, which includes how the robot moves through its cycle. Poor planning can increase energy usage and wear. Optimizing the task order and robot configurations can help minimize cycle time in robots that have multiple feasible arm positions (Bottin et al., 2022). This kind of optimization is especially useful in gantry robots, where there may be many ways to reach a target.

As robotic systems grow more complex, predictive methods become necessary to estimate and monitor performance. One such approach involves predictive maintenance, where robot signal data such as motor currents are compared over time (Guo et al., 2021). This type of system allows early identification of anomalies in gantry robot motion, even before failures occur.

In summary, gantry robot performance is usually evaluated through a mix of time-based, energy-based, and reliability-focused metrics. Cycle time remains the most commonly used measure, but it must be considered along with factors like payload, movement efficiency, and mechanical health. Real-world data and predictive modelling, as shown in several studies, are crucial tools for improving and monitoring the performance of gantry robots.

2.3 Machine Learning in Industrial Robotics

Machine learning (ML) is becoming more common in industrial robotics, especially when it comes to improving robot performance, reducing errors, and making production systems smarter. Instead of relying only on fixed rules and programming, ML models can learn patterns from data, make predictions, and adjust to changes in operations. In robotic manufacturing, this helps with tasks like predicting cycle time, detecting unusual behaviour, and improving reliability.

For example, a time-series-based machine learning method was developed to predict milling force in industrial robots (Wu et al., 2024). This method used Long Short-Term Memory (LSTM) models with optimization to predict performance changes in real-time. By doing this, the robot could monitor its own operations and adjust to avoid defects like vibration or tool wear. This shows that ML models can work with time-based sensor data to track robot conditions during operation.

In another study, simulation and data-driven models were used to estimate not only the energy consumption but also the reliability of industrial robots

(Stuhlenmiller et al., 2021). They showed that short cycle times combined with heavy payloads could increase the risk of wear in robotic joints. With predictive models, it's possible to analyse how robot settings affect long-term performance and maintenance needs.

Several machine learning models have also been tested for predicting task durations in complex robotic systems. One predictive maintenance approach uses historical signal patterns, such as motor currents, to detect early signs of mechanical failure in robots (Guo et al., 2021). Instead of waiting for a breakdown, their method allows early alerts by tracking changes in signal behaviour using statistical models.

Another useful application of machine learning in robotics is cycle time estimation. A simulation-based machine learning model was introduced to predict cycle time in robot welding (Cho et al., 2024). Their research showed that smart models could estimate production time more accurately than traditional methods, which often rely on rough averages or manual input.

Furthermore, a study explored how tuning machine learning model parameters for robotic systems can affect performance (Farrahi & Mahmood, 2023). Their work showed that choosing the right cycle time and model settings is important for real-world learning. If the parameters are not tuned properly, even good models like PPO and SAC may perform poorly.

Overall, machine learning enables robotic systems to:

- Predict performance changes or failures in advance,
- Optimize cycle times by learning from past data,
- Adjust to different operating conditions without manual reprogramming.

These capabilities make ML especially useful in systems where robot behaviour changes over time or when robots are used in different ways for similar tasks.

2.4 Explainable AI: SHAP, Sobol, and PCA

In many machine learning applications, it is important not just to make accurate predictions, but also to understand why the model makes certain predictions. This is especially true in industrial robotics, where engineers and decision-makers need to know which factors affect robot performance, and how. This is where Explainable AI (XAI) methods come in. Techniques like SHAP (SHapley Additive exPlanations), Sobol sensitivity analysis, and PCA (Principal Component Analysis) help us see which features or variables are most important in a prediction model.

SHAP is one of the most widely used methods for explaining predictions made by machine learning models. It is based on game theory and gives each feature an importance value for a particular prediction. For example, when predicting robot cycle time, SHAP can show whether a specific feature like payload or travel distance has a strong positive or negative influence on the result. SHAP has been successfully applied in industrial robotics for tasks like battery life prediction, where understanding feature contributions helps improve performance (Rao et al., 2024).

Sobol sensitivity analysis is a useful method for understanding how different input variables affect the output of a model. It shows which inputs have the biggest impact, both on their own and when they interact with other inputs. This is especially helpful in robotics, where many signals like velocity, torque, and acceleration can work together to influence performance. One study applied this method to a 5-axis hybrid robot to find the most important variables, simplify the model, and improve the accuracy of predicting joint forces and torques (Luo et al., 2024). Using a similar approach for cycle time analysis can help identify which input signals are most important to watch for early signs of performance issues.

Principal Component Analysis (PCA) is a way to make a large set of data easier to understand. It takes many related variables and turns them into a smaller number of new variables that still keep the most important information.

These new variables are called principal components. PCA helps to spot patterns in the data, such as strange or unusual robot behaviour. For example, researchers used PCA to match music features with robot joint movements so the robot could dance. Even though this study focused on performance, it shows how PCA can help make complex robot data simpler and more useful (Saviano et al., 2024).

Combining SHAP, Sobol, and PCA gives a strong foundation for making machine learning models more transparent and trustworthy. In the context of this thesis, these methods help answer key questions, such as:

- Why did the robot performance change?
- Which signals or sensor values are most responsible for the differences?
- How do multiple factors interact over time?

By using XAI tools, this study does not just predict changes or differences in cycle time, but also explains them clearly, which can help with maintenance, diagnostics, and system improvement.

2.5 Related Work and Research Gaps

Recent studies have explored how machine learning models can be applied to understand robot performance in industrial settings, focusing on factors such as energy consumption, task duration, and operational efficiency. For example, (Stuhlenmiller et al., 2021) analysed how robot cycle time and payload affect energy efficiency and reliability in the long run, showing that performance is not just about speed but also resource usage and wear over time. Similarly, (Cho et al., 2024) presented simulation-based approaches for predicting cycle time in robotic welding, emphasizing how accurate prediction models can improve production planning.

Several papers also focus on modelling variations in robot cycle time due to internal or external factors. For instance, (LI et al., 2021) proposed a Bayesian model to capture heterogeneous assembly times caused by material differences or unknown conditions. Their study emphasizes the importance of including both observed and unobserved factors when modelling robot performance, which aligns well with this thesis's focus on multivariate data.

In terms of using machine learning, (Farrahi & Mahmood, 2023) investigated how policy optimization in reinforcement learning is affected by different cycle times. They found that performance varies with action timing, which means cycle time tuning is crucial even when using standard ML models like PPO or SAC. This supports the idea that input signal timing and structure can significantly influence prediction accuracy.

While many of these studies show strong model performance and provide good insights, they often focus on either offline prediction, simulation environments, or narrow use cases like welding or milling. Few studies examine long-term changes in real robots performing real tasks over months of operation. For example, (Deng et al., 2022) proposed a digital twin for real-time operational prediction, but did not fully address robot-to-robot variability or explain feature impact using interpretable models like SHAP or Sobol.

Identified Gaps

Lack of long-term performance monitoring: Most prior research focuses on short-term or simulation-based analysis. This thesis addresses long-term monitoring using real operational data.

Limited interpretability: Many models are treated as black boxes. This study adds explainability using SHAP, Sobol, and PCA, helping identify which signals affect cycle time the most.

Few studies on robot-to-robot differences: While some works examine system-level efficiency, they do not compare multiple robots doing similar tasks under similar conditions.

Client-specific data structure: Existing research does not reflect the custom gantry robot setup used by Cimcorp, meaning this study fills an industry-specific need with real data.

3 DATA AND PREPROCESSING

3.1 Description of Operational Data

This thesis uses real-world data collected from industrial gantry robots operating in Cimcorp Oy's automated material handling systems. These robots work in tasks like picking, moving, and placing items in logistics environments. The data was recorded continuously during live production.

The focus of the analysis is the cycle time, which is the time a robot needs to complete one task cycle. This variable is used as the target for machine learning models. The goal is to detect changes in robot performance over time and find differences between similar robots.

The dataset contains 46 columns. Among these:

- ta is the target variable, representing cycle time
- deviceid and cmd_time are metadata columns and are dropped during preprocessing
- There are 2 Boolean features: floor_pick and simultaneous_operation
- A third Boolean feature, ev_positive, is created during preprocessing to indicate whether the ev signal is greater than zero

- The remaining columns are numerical input features, used by the model to learn performance patterns

Even though the real feature names are not known due to anonymization, machine learning models can still learn from the patterns and relationships in the numeric data. This practice is common in industrial applications where data confidentiality is important. In such cases, variables are anonymized or encoded to protect sensitive information about processes or equipment, but models are still able to identify trends and make accurate predictions based on statistical behaviour. For example, a study modelled robotic assembly performance using anonymized and heterogeneous data to ensure confidentiality while preserving the ability to analyse system performance. The researchers used a full Bayesian approach to capture the effects of both observed factors and unobserved heterogeneity, such as material variability or unknown environmental influences. Even without knowing the exact feature names, their model successfully predicted assembly cycle times and identified key influencing factors. This shows that anonymized data can still be highly useful in predictive modelling when appropriate statistical techniques are applied (LI et al., 2021).

The data includes performance logs from eight different robots, each operating in varied conditions. To better understand the context and behaviour, Boolean features are used to define operational states. In this study, two specific states are selected for focused analysis:

- all_true: where floor_pick, simultaneous_operation, and ev_positive are all True
- simul_false: where floor_pick and ev_positive are True, but simultaneous_operation is False

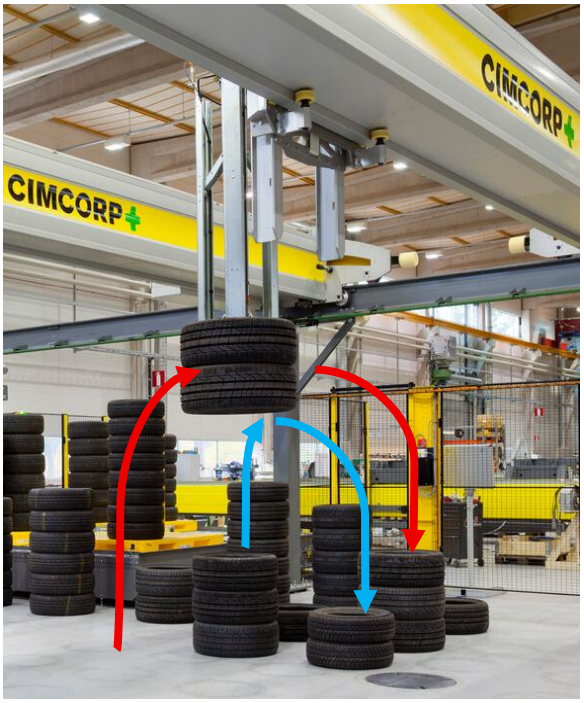


Figure 3.1: Cimcorp gantry robot performing floor picking, simultaneous operation, and ev activity (all_true state). Arrows added by author to indicate key actions (Image source: Cimcorp Oy).

These states are defined using structured filtering, which enables consistent comparisons across robots and over time. A similar idea of parameter-based grouping for analysing robot cycle time was explored using simulation models in storage and retrieval systems (Trost & Eder, 2024). The importance of analysing task-specific cycle time variations to support robot system design was also discussed in a review of human-robot work systems (Komenda et al., 2021).

Because the data spans a long operational period, it is well-suited for analysing both time-based performance changes and differences between individual robots. This is similar to work using simulation and real-world data to predict cycle times for robotic welding under different configurations (Cho et al., 2024). Other studies also emphasize the importance of modelling time-based variability and system configurations to better understand cycle time performance (Komenda et al., 2021; Stade et al., 2025).

In summary, the dataset includes:

- One known target variable: cycle time (ta)
- Several anonymous numeric features
- Three known Boolean input signals
- Eight robots, observed over a long period
- Two operational states selected for detailed analysis

These characteristics make the dataset a strong foundation for modelling robot performance using multivariate machine learning.

3.2 Data Collection Period and Robot Context

The data used in this thesis was collected between December 1, 2024, and March 23, 2025, from eight customized gantry robots deployed in real industrial environments at Cimcorp Oy. These robots are part of advanced material handling systems that operate continuously in high-demand logistics and manufacturing settings.

Each robot performs a sequence of operations, such as picking and placing goods, which are recorded as task cycles. The robots worked under different operational conditions, allowing for detailed comparisons. Although they all performed similar functions, the environments and settings varied slightly, which helps in understanding how these differences affect robot performance.

In related research, robot cycle time in robotic compact storage systems was shown to depend heavily on layout and stacking height, even if the robot hardware remains the same (Trost & Eder, 2024). Similarly, cycle time and payload changes were found to directly affect robot resource efficiency, especially over long use periods (Stuhlenmiller et al., 2021). These studies support the importance of analysing robot behaviour across different setups, even when tasks appear the same.

The variety in settings during the data collection period offers valuable context. As shown in prior work, robot performance often varies due to both observed and hidden factors, and long-term monitoring is necessary to capture these trends (LI et al., 2021). Similarly, this thesis uses nearly four months of data to examine how performance changes over time and across different robots.

Furthermore, cycle time prediction becomes more accurate when the robot's working context is well understood and time-based data is used to reflect changing system conditions, as shown in recent research (Cho et al., 2024). This approach fits the current thesis, which uses a structured time period and robot-specific data to model and explain cycle time behaviour.

In summary:

- Data was collected from eight gantry robots
- Time period: 2024-12-01 to 2025-03-23
- Robots worked in real industrial settings under different operational conditions
- Similar robot tasks allow meaningful performance comparisons
- Time coverage supports trend analysis and change detection, as backed by recent research

3.3 Data Grouping and Operational States

To better understand robot performance under different working situations, the full dataset was divided into eight groups. These groups were based on three binary input features that describe specific operational settings for each cycle:

- floor_pick (True/False): whether the robot picked items from the floor,
- simultaneous_operation (True/False): whether the robot was operating with other equipment at the same time,
- ev_positive (True/False): a signal related to event status.

Each group in the dataset represents a unique combination of these three features, resulting in 2 × 2 × 2 = 8 distinct operational groups. This grouping allowed a structured comparison of robot behaviour across clearly defined conditions.

From these eight, two operational states were selected for detailed analysis:

- all_true: where all three flags (floor_pick, simultaneous_operation, ev_positive) are True.
- simul_false: where floor_pick and ev_positive are True, but simultaneous_operation is False.

These states were chosen to focus on specific differences caused by simultaneous operations while keeping the other conditions constant. By controlling two variables and changing only one, the impact of that change becomes clearer.

This grouping method is supported by studies that show how setup and task design affect robot performance. For example, understanding task-specific configurations is important for analysing cycle time, especially in human-robot systems (Komenda et al., 2021). Robot cycle time can also vary a lot depending on the order of tasks and how the robot is set up. Separating different cases makes the analysis more accurate and meaningful (Touzani et al., 2021).

The grouping also helps manage different robot behaviours, a challenge noted by researchers who used Bayesian models to handle variation from both known and hidden factors (LI et al., 2021). In this study, we use flags like simultaneous_operation to separate the data into clear groups. This makes the analysis easier to understand.

By comparing groups such as all_true and simul_false, we can see how running other operations at the same time affects performance, without getting distracted by unrelated factors.

3.4 Data Cleaning and Feature Engineering

Before building machine learning models, the raw data needed to be cleaned and prepared. The original dataset contained 46 columns, including a mix of numeric, Boolean, and timestamp features. The main goal of data cleaning was to remove irrelevant, duplicated, or misleading information while keeping the core signals that affect robot cycle time (ta) (For the full preprocessing code, see Appendix 2.1).

The first step was to remove unnecessary columns:

- deviceid was dropped because it only identifies the robot, and models should not learn from labels.
- cmd_time was also removed, as this is a timestamp and all time-based features were excluded.

Next, a detailed analysis was done to identify duplicate and highly correlated columns:

- Columns such as go, gj, ek, and gh were found to be exact duplicates or near-perfectly correlated with gg, ej, el, and gi. These redundant columns were dropped to reduce noise and dimensionality.
- A new column ghi was created to replace the highly correlated pair gh and gi, using the formula ghi = gi + 40.

Then, rows with invalid or irrelevant conditions were removed:

- Rows where tm > 0 or tp > 0 were excluded, since these features signalled operational states not relevant to the intended analysis.
- Rows where ta (cycle time) exceeded 20,000 were also removed. These extreme values were treated as probable logging errors or exceptional cases.

Outlier detection is a common step in data analysis, especially when trying to clean noisy data. Some widely used methods include interquartile range (IQR), median absolute deviation (MAD), standard deviation filters, and Z-score thresholds. These techniques are helpful when working with general-purpose datasets where extreme values often result from errors. However, when working with robot signal data, these methods must be used carefully. In industrial settings, unusual values in the data may still represent real robot behaviours, such as sudden stops, high torque movements, or rare task events. Automatically removing them can erase useful information.

For this reason, global outlier filtering was avoided in this study. Instead, filtering was applied only when there was a clear operational reason, such as sensor faults, logging anomalies, or known irrelevant operating conditions. This approach follows recommendations from studies that focused on robotics and manufacturing data, where high-frequency data often contains meaningful variation even at the extremes (LI et al., 2021). Similarly, a digital twin-based model preserved outlier behaviour to better reflect system changes over time (Deng et al., 2022). In another example, simulation and real-world data were both used to train models that retained unusual signals for accurate performance prediction (Cho et al., 2024). Task-specific conditions in human–robot systems also caused variation that should not be filtered without a clear reason (Komenda et al., 2021).

To better structure the data, a new Boolean column called ev_positive was added. It indicates whether the ev signal is greater than zero and helps define different operational states of the robot. This column was used later to group the data for more meaningful comparisons between robots and over time.

Finally, the feature tb was dropped after checking its distribution. It was close to zero in nearly all rows and showed no meaningful variation or contribution to predicting cycle time.

After the cleaning process, the dataset was reduced and clarified:

- 40 columns remained, including only numeric and Boolean features
- The target variable was ta (cycle time)
- Key Boolean inputs for later grouping were floor_pick, simultaneous_operation, and ev_positive

This structured data cleaning follows common best practices in industrial robot modelling. For example, cleaning out repeated or unhelpful sensor features before building models can help improve accuracy (Wu et al., 2024). Also, it is important to use specific cleaning steps based on how the robotic system works, especially when digital twin models are involved (Deng et al., 2022). These examples support the idea that cleaning should be based on clear reasons from the data, not just automatic methods that might remove useful patterns.

These steps helped prepare the dataset for accurate, fair, and interpretable machine learning analysis.

3.5 Anonymization and Confidentiality Considerations

Since the dataset and project were provided by Cimcorp Oy, all work on the data had to respect client confidentiality. This was especially important because the dataset contained real operational data from industrial robots in use at customer sites.

To protect sensitive information, several steps were taken to anonymize the data before any modelling or analysis was done:

All signal names were anonymized by the client before analysis. This means that while each column contains valid numeric or Boolean values, the true meaning or source of each signal is not known to the researcher. This protected the technical and commercial details of the robot systems.

Robot identifiers such as deviceid were removed early in the preprocessing phase. While device IDs helped confirm the diversity of sources (eight different robots), they were not used during modelling to prevent any chance of reverse identification.

Time-based data, such as cmd_time, was also removed. This ensured that no specific timeframes or patterns related to client operations could be reconstructed from the dataset.

Operational states and internal flags, like floor_pick or simultaneous_operation, were used only in a generalized way to form grouped conditions. These groupings helped the analysis but did not expose specific processes, configurations, or workflows.

By anonymizing feature names and removing metadata, the analysis focused entirely on numerical patterns and general behaviour, rather than any sensitive or identifiable details. This approach aligns with ethical practices in industrial data science. For example, responsible handling of real-world robot data, especially in collaboration with companies (Deng et al., 2022; Komenda et al., 2021).

The results and interpretations in this thesis are therefore based on abstracted signal data and general trends, without compromising the confidentiality or proprietary information of Cimcorp Oy or its customers.

3.6 Conceptual Framework

This conceptual framework was developed based on the literature reviewed in earlier sections. It brings together key ideas from industrial automation, gantry robot performance, machine learning, and explainable AI. The purpose of the framework is to show how the main components of the study are connected and how the research is structured to address the research questions.

The framework begins with real-world sensor data collected from multiple gantry robots operating in industrial environments. This raw data includes numeric signals and Boolean indicators that reflect different working conditions. Once collected, the data undergoes cleaning and grouping. This involves removing irrelevant or duplicated features, managing missing values, and segmenting the data based on robot operational states.

After data preparation, machine learning models are used to predict cycle time, which is the target variable in this study. The primary model is XGBoost regression, selected for its high accuracy and ability to handle structured data effectively. The objective is not only to generate accurate predictions but also to understand which features influence robot performance.

To support this, explainable AI techniques are applied to the trained models. Methods such as SHAP and Sobol sensitivity analysis are used to identify which input signals most impact the predicted cycle time. These insights support engineering decisions and provide a deeper understanding of robot behaviour.

Overall, the conceptual framework as depicted by figure 3.2 guides the entire research process. It connects the data sources, the modelling steps, and the interpretation tools. It also reflects the study’s goal of combining predictive performance with interpretability in a real industrial context.

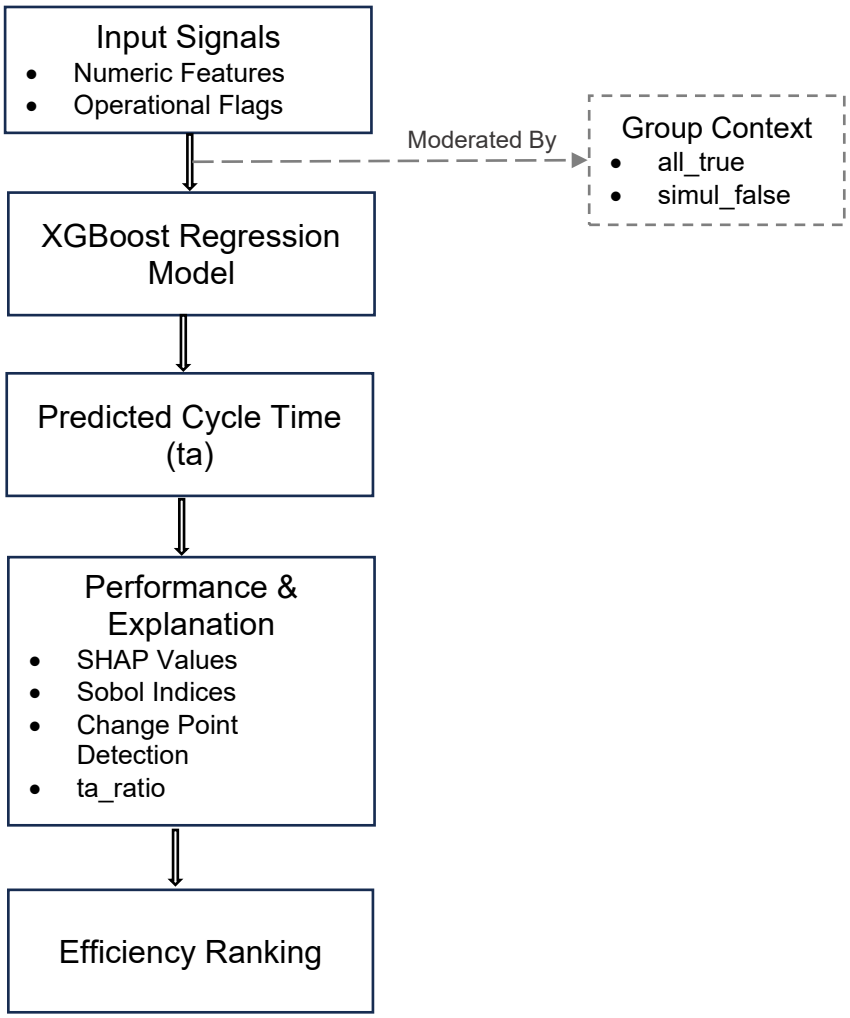


Figure 3.2: Conceptual Framework of the Study (Developed by the Author)

4 MACHINE LEARNING METHODOLOGY

4.1 Overview of the Modelling Process

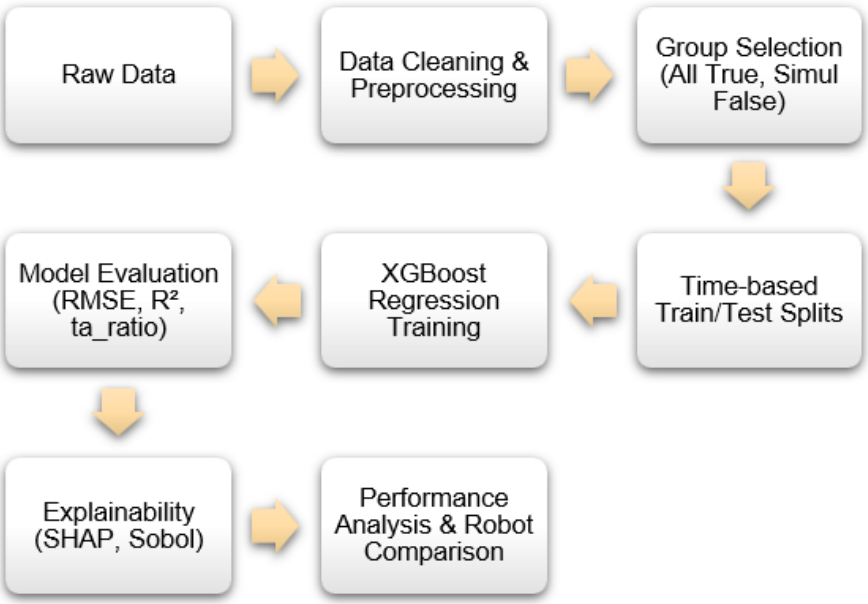


Figure 4.1: Machine Learning Pipeline (Developed by the Author)

The core goal of this thesis is to model and explain variability in robot performance, particularly cycle time, using real-world multivariate sensor data collected over nearly four months. The modelling process shown by figure 4.1 was carefully designed to ensure that results are not only accurate but also interpretable and actionable in an industrial context. The dataset contained 46 original columns including timestamps, device identifiers, Boolean operational flags, and numerous numeric signals. After detailed preprocessing, 43 features were retained for modelling, and the only known output variable was ta, which represents cycle time.

The data preparation phase was followed by the construction of a supervised regression pipeline. Two key groups were defined based on the Boolean features floor_pick, simultaneous_operation, and ev_positive. The first group, named all_true includes rows where all three features are true. The second

group, named simul_false, includes rows where floor_pick and ev_positive are true but simultaneous_operation is false. These specific groupings were chosen in collaboration with the client to examine how simultaneous operation affects robot performance when the other two conditions are held constant.

To build fair and reliable models, the data for each robot was split into time-based blocks using a five-day rolling segmentation method. Within each robot's timeline, consecutive five-day windows were alternated into training and testing subsets. This method ensured that the models were evaluated on different time segments than they were trained on, helping to avoid overfitting and to capture temporal dynamics in robot behaviour. Such a time-aware cross-validation approach is especially important in robotic environments where performance may shift gradually due to wear, task variations, or external influences.

All models were trained on cleaned data where only meaningful features were retained. Duplicate and highly correlated columns were removed to reduce noise. In addition, features such as ex and ey were converted to their absolute values to account for directional symmetry, which has no bearing on cycle time in this analysis. The three Boolean grouping variables and the metadata fields deviceid and cmd_time were excluded from model training to ensure generalization and to prevent the model from learning unintended biases. After these adjustments, 38 features remained as numeric input variables for prediction.

The main model used in this research was XGBoost, a tree-based boosting algorithm known for its efficiency and accuracy in working with structured tabular data. XGBoost was chosen because it handles outliers well, can learn complex interactions between features, and has built-in support for missing data and feature importance ranking (Farrahi & Mahmood, 2023). This makes it suitable for real-world signal datasets like those generated by industrial robots, where input features can have different scales, patterns, and behaviours.

The model was first used to predict ta values, and then to help identify which input features most influenced the predictions. In addition to building accurate models, a major focus of the modelling process was to provide insights that could be interpreted by engineers and stakeholders. To support this, explainability tools such as SHAP and Sobol sensitivity analysis were applied after training. SHAP values were used to rank the importance of features and to show how changes in input values influence the predicted cycle time. These techniques made it easier to identify which features were most associated with increases or decreases in performance across different robots and time windows (Luo et al., 2024; Trost & Eder, 2024).

In addition to building and interpreting the XGBoost regression models, further analyses were conducted to deepen the understanding of robot performance differences over time and between devices. These analyses were designed to detect temporal shifts, compare robots quantitatively, and identify root causes of performance variation.

First, change point detection was applied to the cycle time series of individual robots. Using the PELT algorithm from the ruptures library, significant shifts in daily average cycle time were identified. This allowed the detection of when a robot's behaviour changed significantly, which is essential for understanding trends, anomalies, or operational shifts. Robots with at least 50 days of valid data were included in this analysis to ensure stable detection. The change points provided concrete time markers where deeper feature investigations could be focused.

To understand what might have caused these shifts, a method was implemented to analyse changes in feature importance across time segments. Each robot's timeline was segmented based on the detected change points, and the correlation between input features and the target variable ta was recalculated for each segment. The features whose correlation with cycle time changed the most between segments were identified as the most likely contributors to the performance shifts. This step helped reveal which specific

signals were responsible for cycle time variability and how their influence evolved over time.

Next, a comparative analysis was performed between robots from the all_true and simul_false groups. For each robot, the top features most correlated with cycle time were extracted, and these lists were compared across groups. The symmetric difference between the top features of the two groups revealed which input signals behaved differently under varying operational states. This helped isolate the impact of simultaneous operation and other context-dependent behaviours.

In a separate classification task, a Random Forest model was trained to distinguish whether a data point belonged to the all_true or simul_false group based on the selected features. This not only tested whether the operational states influenced feature behaviour but also allowed the identification of features most important for distinguishing between the two groups. The model achieved high classification accuracy, reinforcing the validity of the group-level behavioural differences.

SHAP values were also computed at the individual robot level to better understand which features most influenced predictions per robot. This provided detailed insight into robot-specific behaviour, showing, for example, whether one robot relied more on signal ex while another was more affected by gg. The SHAP summary plots and bar charts allowed engineers to compare machines not only by their average cycle times but by the specific feature interactions driving those outcomes.

To complement SHAP, Sobol sensitivity analysis was extended to individual robots as well. Using the SALib framework and Saltelli sampling, Sobol indices were calculated for each robot’s model to quantify the influence of each input feature on the cycle time predictions. These results were visualized as feature-by-robot heatmaps, making it easier to identify which features were consistently important and which varied in importance across devices. Sobol

analysis added an extra layer of global interpretability to complement the local insights from SHAP (Trost & Eder, 2024).

Temporal performance trends were also visualized using weekly PCA scatter plots and boxplots of cycle time per robot. These visualizations helped to explore how robots clustered in signal space and how their performance distributions shifted over time. Weekly comparisons highlighted not only differences between robots but also potential drift or degradation within robots, which could indicate maintenance needs or adaptation issues.

Finally, an efficiency ranking system was created to assess each robot's performance relative to model expectations. For each robot, the predicted cycle time was compared to the actual cycle time using a ratio metric (ta_ratio = actual / predicted). Robots were ranked based on their average cycle time and average ta_ratio, producing an interpretable efficiency score that combined speed and consistency. This ranking approach helps prioritize robots for further investigation or optimization, especially in high-volume production settings.

Together, these additional modelling and analysis steps provided a detailed and actionable understanding of robot behaviour over time and across operational conditions. They enabled not just prediction, but also explanation, comparison, and diagnosis of performance differences, all grounded in explainable machine learning and sensitivity theory (Cho et al., 2024; Guo et al., 2021; Luo et al., 2024).

4.2 XGBoost Regression Model

The main algorithm used for modelling cycle time in this thesis is XGBoost (Extreme Gradient Boosting). XGBoost is a type of decision tree–based machine learning algorithm that is well-known for its strong performance on structured tabular data. It builds an ensemble of trees by adding new trees that

correct the errors made by previous ones. This process is called boosting, and it helps the model learn complex patterns in the data more effectively.

One of the main reasons XGBoost was chosen is its ability to handle noisy, real-world sensor data from industrial robots. It works well even when input features are on different scales or have missing values. It can also manage outliers better than some other models, which is important when working with operational data that may include rare but extreme measurements. These capabilities make XGBoost suitable for manufacturing environments, where the input signals can vary over time or between machines (Farrahi & Mahmood, 2023; Hung et al., 2018).

In this thesis, XGBoost is used as a regression model. That means it predicts a continuous output. In this case, the robot cycle time (ta). The model takes as input a set of numeric features that were cleaned and selected through preprocessing. To ensure fairness and generalization, no labels like robot ID or timestamps were included in the model training, and directional features like ex and ey were converted to their absolute values to avoid misleading the model.

During training, XGBoost learns to minimize the error between its predictions and the actual ta values. It does this using a loss function, usually the squared error. The model also uses regularization, which prevents it from overfitting the training data by keeping the model from becoming too complex. XGBoost also supports early stopping, which means it can stop training when performance stops improving on a validation set.

XGBoost has been successfully applied in several recent robotics and industrial analytics studies, where it was used to predict energy consumption, detect failures, and estimate process times (LI et al., 2021; Stuhlenmiller et al., 2021). Its ability to provide feature importance scores also supports explainability, which is a key goal of this thesis. These scores show which input signals are most influential in the model’s predictions, making it easier for engineers to understand what is driving changes in robot performance.

Overall, XGBoost provides both strong prediction power and useful interpretability, which makes it a good fit for detecting and explaining performance variation in industrial robots using multivariate signal data.

4.3 Model Training and Evaluation Metrics

After selecting features and preparing the dataset, the XGBoost regression models were trained to predict the cycle time (ta) using 38 numeric features. The training and evaluation process was designed to ensure fairness and generalization, especially because the data came from real industrial operations over several months.

Each robot's timeline was divided into five-day blocks. These blocks were alternately assigned to training and testing sets, so that the model was evaluated on time periods that were not seen during training. This time-based splitting method helped simulate a realistic deployment scenario, where future performance is predicted from past behaviour. Such an approach helps address concept drift and ensures fair assessment in time-sensitive applications (Wu et al., 2024).

To find the best model settings, a randomized grid search was performed over key XGBoost hyperparameters, including number of estimators, learning rate, maximum tree depth, and subsampling rate. Each combination was evaluated using 3-fold cross-validation on the training data. This method allowed for faster optimization while avoiding overfitting.

Model performance was measured using two main metrics:

- Root Mean Squared Error (RMSE): RMSE calculates the average magnitude of the error between predicted and actual cycle times. It is sensitive to large errors and is widely used for evaluating regression models. RMSE has been shown to be effective for evaluating prediction

accuracy in robot task performance, as demonstrated in prior studies such as (LI et al., 2021).

- R-squared (R^2): This metric indicates how well the model explains the variability in the output variable. An R^2 score close to 1 means the model explains most of the variation in cycle time. It is a standard regression metric used in many industrial machine learning studies, where robot system performance was evaluated using both RMSE and R^2 (Luo et al., 2024).

These metrics were calculated not only globally across the entire dataset but also per robot. Evaluating per robot helped to understand how consistent the model performance was across different machines. This is important in real-world applications, where sensor quality or hardware differences may affect prediction quality.

In addition, a performance ratio (ta_ratio) was computed for each sample. This ratio is defined as the actual cycle time divided by the predicted cycle time. Ratios above 1 indicate slower-than-expected performance, while ratios below 1 indicate faster performance. This ratio was used later to compare robot efficiency over time and across groups.

This evaluation process combined standard machine learning practices with domain-specific insight into robot performance. It enabled both accurate prediction and actionable insights into how and why performance differs between robots (Guo et al., 2021; Luo et al., 2024).

4.4 Cross-validation Strategy

To make sure the models built in this thesis can be trusted and work well on new data, a careful cross-validation strategy was used. Since the dataset comes from real robots working over time, random splitting of the data would mix early and late data together. This could lead to misleading results, as the

model might learn patterns from future data while predicting past behaviour. To avoid this, a time-aware splitting method was used.

Each robot's timeline was divided into consecutive five-day blocks, starting from its earliest timestamp. These blocks were then split into training and testing sets by alternating them. For example, the first block was used for training, the second for testing, the third for training again, and so on. This ensured that the model was always tested on future data that it had not seen during training. This type of temporal cross-validation is important when working with systems where performance might slowly change over time due to wear, environment, or workload (Wu et al., 2024).

Using this method also made it possible to fairly compare robots by evaluating each one on its own timeline, without mixing data across robots. It respected the time order of events, which is especially important when modelling industrial processes or sensor-driven systems. A similar approach was used in predictive maintenance and robotic behaviour modelling, where the importance of keeping the temporal structure of data intact was highlighted (Trost & Eder, 2024).

This five-day block method provided enough variation between training and testing segments, making sure the model learned general patterns and not just memorized the data. It also allowed performance to be compared across different robots and time periods in a consistent way. For the full model training and tuning code, see Appendix 2.2.

By combining this time-based cross-validation with clean and structured inputs, the overall modelling process became more robust, giving more reliable and interpretable results.

4.5 Performance Results

After training the XGBoost regression models on both robot groups (all_true or simul_false), their predictive performance was evaluated using Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2). These metrics were chosen because they are widely used in machine learning applications where the target variable is continuous, such as in cycle time prediction.

The models showed strong predictive accuracy across both robot groups. In the all_true group, the average RMSE values per robot were consistently low, while R^2 scores were high, indicating that the model was able to explain a significant portion of the variability in cycle time. Similar results were observed for the simul_false group, confirming that the modelling approach generalised well across different operating conditions.

To make the results more interpretable, actual versus predicted cycle times were plotted for individual robots. These scatter plots revealed that most predictions closely followed the ideal diagonal line, with some expected variance at higher or more irregular cycle times. Additionally, robot-specific metrics such as the mean of the actual-to-predicted ratio (ta_ratio) were computed. This helped identify whether the model systematically overestimated or underestimated cycle time for certain robots.

A further analysis was done to compare model performance between robots. Robots with higher average RMSE and lower R^2 were flagged as having more unpredictable performance patterns. In contrast, robots with tighter prediction accuracy suggested more stable operations. This kind of interpretability is especially important in industrial settings, where engineers must understand not just whether a model works, but also why and for whom it works best (Deng et al., 2022; Luo et al., 2024).

These results demonstrate that XGBoost, when combined with time-aware validation and proper data preprocessing, can effectively model cycle time

variations in industrial robots. It provides not only accurate predictions but also supports performance monitoring and robot comparison in production environments (Komenda et al., 2021).

5 EXPLAINABILITY AND PATTERN ANALYSIS

5.1 Visualizing Behavioural Differences Between Robots

To investigate the temporal evolution of robot performance, this section applies change point detection to segment cycle time trends. The Python library ruptures was used with the PELT algorithm and an L2 cost function. This method is designed to detect points in a time series where the statistical properties (e.g., mean) of the signal change. A penalty parameter is used to control model complexity and avoid overfitting by limiting the number of detected segments.
This type of approach aligns with prior studies that used the ruptures library for time-series segmentation to detect performance shifts and behavioural changes in dynamic systems (Truong et al., 2020).

Change point detection is applied to each robot's daily average cycle time to identify performance shifts that may reflect mechanical adjustments, wear-and-tear, process changes, or environmental influences. The method does not assume major breakdowns; rather, it is sensitive enough to detect subtle but consistent changes in behaviour, as shown in Appendix 2.3.

5.1.1 Case Study: Robot 21

This section compares Robot 21 in all_true and simul_false groups to show how group context affects cycle time and feature importance.

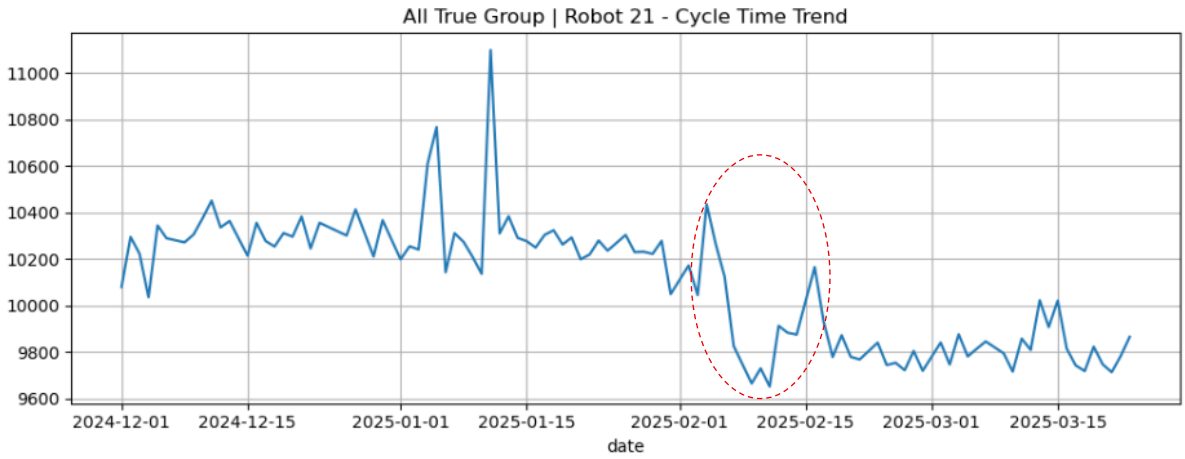


Figure 5.1a: Robot 21 – Cycle Time Trend under all_true Group.

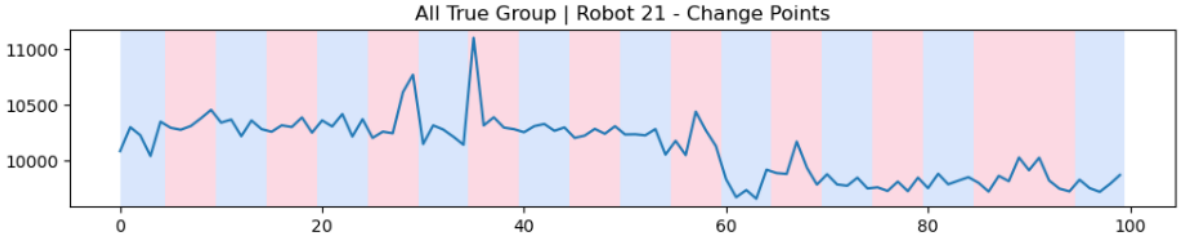


Figure 5.1b: Robot 21 – Change Points under all_true Group.

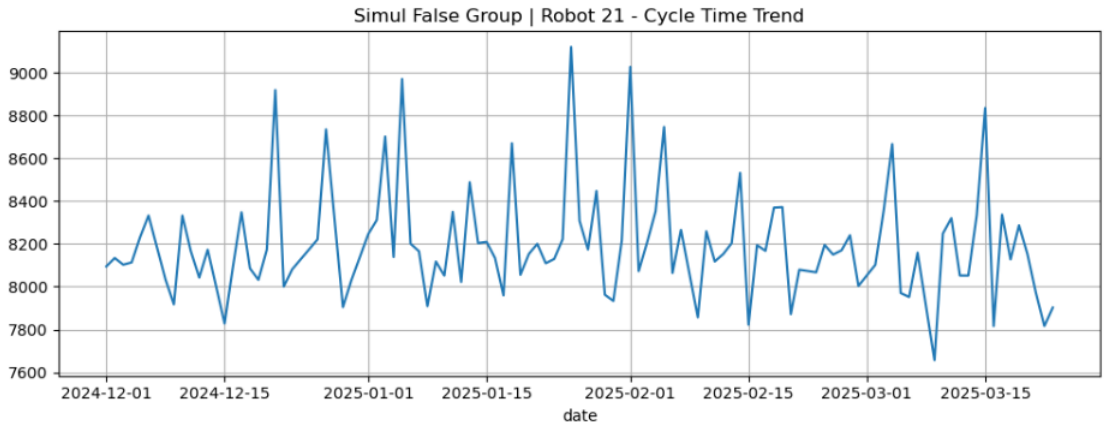
In all_true configuration, Robot 21 operates under synchronized and stable conditions. The cycle time trend (Figure 5.1a) shows moderate fluctuation, with a relatively steady performance until early February 2025. After this point, a noticeable decline in cycle time occurs, suggesting a phase of increased efficiency or external process optimization. This observation is consistent with studies linking cycle time and payload with resource efficiency and reliability in industrial robots (Stuhlenmiller et al., 2021).

The change point analysis (Figure 5.1b) supports this observation. The time series is divided into distinct segments, each marked by a different colour. These segments correspond to periods with relatively stable average cycle time. Transitions between segments align with visible changes in the trend. This methodology aligns with techniques used in modelling heterogeneous robotic performance over time to identify behavioural shifts (LI et al., 2021).

It is important to note that these shifts are not necessarily abrupt or caused by failures. The PELT method can detect even minor but persistent changes in the data, which could be due to changes in operational pattern, routine maintenance, environmental conditions, or scheduling changes. The penalty parameter in the algorithm ensures the segmentation remains meaningful without becoming overly sensitive to noise. While this analysis provides insight into how performance changes over time, domain knowledge is still required to assess the practical significance of these detected shifts.

```
Top features for robot 21:
 ta    1.000000
tw     0.734142
ts     0.374347
el    -0.331305
gg    -0.331305
ei    -0.327242
Name: ta, dtype: float64
```

Figure 5.1c: Top Correlated Features with Cycle Time




```
Top features for robot 21:
 ta    1.000000
ex     0.580163
ev     0.533498
ed     0.399623
ts     0.371183
gg    -0.351875
Name: ta, dtype: float64
```

Figure 5.1d: Cycle Time Trend, Change Points, and Top Correlated Features under simul_false Configuration

For Robot 21 operating under all_true group, correlation analysis (Figure 5.1c) shows that certain input signals have a strong statistical relationship with cycle time. The feature tw has a high positive correlation, which suggests it may be a key factor influencing performance in this configuration. Other features like ts also show a moderate positive relationship, while el, gg, and ei have negative correlations. This means that higher values in these features are often linked with shorter cycle times. These initial results point to clear behavioural patterns in the robot's operation. However, correlation does not confirm causation, so further analysis using explainable AI methods such as SHAP and sensitivity analysis is needed to better understand the role of each feature.

Under the simul_false operational mode, Robot 21 exhibits (Figure 5.1d) a more volatile cycle time trend compared to its all_true counterpart. As shown in the cycle time plot, frequent peaks and troughs are present throughout the observation period, indicating a less stable working environment. The change point analysis reveals numerous short segments, with alternating patterns suggesting repeated shifts in operational dynamics. These changes may be caused by subtle variations such as inconsistent task scheduling, external disturbances, or less synchronized robot coordination.

The top influencing features for this configuration include variables like ex, ev, and ed, each showing moderate correlation with cycle time. The presence of multiple moderate contributors, rather than one dominant factor, supports the interpretation that cycle time variability in this mode is driven by a broader mix of influences. This pattern is shown in Figure 5.2, where feature differences between configurations are highlighted. Overall, Robot 21 under simul_false reflects a system with higher performance variability, where predicting or controlling cycle time may be more complex.

	All True	Simul False	Shifted Features
21	[tw, ts, el, gg, ei]	[ex, ev, ed, ts, el]	[ev, ed, ei, gg, tw, ex]
22	[tw, gg, el, ei, gp]	[ex, ev, ed, ts, em]	[ts, ex, ed, gp, em, el, ev, ei, gg, tw]
31	[tw, td, el, gg, ei]	[ex, ev, em, es, en]	[en, ex, em, es, el, ev, ei, gg, tw, td]
32	[tw, ts, td, gg, el]	[ex, ev, em, es, en]	[en, ts, ex, em, es, el, ev, gg, tw, td]
41	[tw, td, ts, gg, el]	[ex, ev, es, em, en]	[en, ts, ex, em, es, el, ev, gg, tw, td]
42	[tw, td, ts, el, gg]	[ex, ev, em, es, ts]	[ex, em, es, el, ev, gg, tw, td]
51	[tw, td, el, gg, ts]	[ex, ev, em, es, eq]	[ts, ex, em, es, el, ev, gg, tw, eq, td]
52	[tw, el, gg, gp, ei]	[ex, ev, em, es, ts]	[ts, ex, gp, em, es, el, ev, ei, gg, tw]

Figure 5.2: Comparison of Top Correlated Features per Robot in all_true and simul_false Groups, and the Corresponding Shifted Features (i.e., features that appear in one group but not the other for the same robot)

5.1.2. Patterns Across All Robots – all_true Group

After examining Robot 21 in detail, the remaining robots under all_true configuration were also analysed using the same methodology. These included Robots 22, 31, 32, 41, 42, 51, and 52. Each robot's daily average cycle time was visualized over time and change point detection was applied to capture transitions in performance.

- Robot 22 experienced a rise in cycle time around mid-December 2024, stabilizing by late January 2025. The change points aligned with this shift. After stabilization, performance was relatively flat. Top correlated features were similar to Robot 21, but with a slightly stronger link to gg.

- Robot 31 showed frequent short-term fluctuations, with change points appearing almost every 7–10 days between December and February. These segments suggest operational volatility or task switching. The cycle time had alternating increases and drops.

- Robot 32 mirrored Robot 31's pattern but with less intense variability. Cycle time generally trended downward, suggesting improving

performance. Several change points occurred in late December 2024 and mid-January 2025.

- Robot 41 had a stable trend with only two major shifts, both resulting in a slight decrease in cycle time. The key shifts occurred around early January 2025 and late February 2025.

- Robot 42 showed a similar behaviour to Robot 41, but one of its segments in early February had a short spike in cycle time, possibly due to task-related disturbance. The overall trend was still downward.

- Robot 51 had consistent decrease in cycle time with very minor fluctuations. One clear shift happened in mid-January 2025, after which the robot maintained a lower average cycle time.

- Robot 52 followed Robot 51’s trend, though it showed a slightly longer stable phase before dropping. The change points were fewer but clearly aligned with dips in cycle time around late January 2025.

These variations highlight that even under synchronized operational states (all_true), robot performance is not uniform. The number of change points detected per robot also varied, reinforcing the idea that individual robot behaviour evolves differently over time. This variability reflects findings in studies of human–robot and multi-robot systems where cycle time differences arise from heterogeneous task conditions (Farrahi & Mahmood, 2023). A visual summary of these patterns is provided in Appendix 1.

While the change point analysis reveals meaningful shifts in performance trends, interpreting their operational or mechanical causes requires collaboration with domain experts who have contextual knowledge of robot tasks and system conditions.

5.1.3. Patterns Across All Robots – simul_false Group

In contrast, the simul_false group showed more similar behaviour across all robots. Most robots, including Robot 21, had a cycle time that either increased slowly or changed slightly over time. The change point detection found several shifts, but these happened at similar times and with less variation than in all_true group.

This means that when robots work without doing tasks at the same time, their behaviour becomes more regular and predictable. This may be because they follow a more standard task routine with fewer changes.

Robots 22, 31, 32, 41, and others also showed these steady patterns. The cycle time changes were smaller and more alike across different robots. This supports the idea that how robots are grouped affects how they perform.

5.2 Scatter Plot: Model Fit Visualization

This section evaluates the predictive performance of the trained XGBoost models using visualizations of the actual versus predicted cycle time (ta). These visual tools help to assess how well the model captures robot behaviour patterns across different configurations and robots.

Two types of plots are used for each prediction set:

- Scatter plots, which show a sample of individual predictions and their alignment with the ground truth. Points that lie close to the diagonal red dashed line indicate strong predictive performance. The tightness of clustering reflects prediction consistency, while wider dispersion suggests greater uncertainty or noise.

- Density plots, represented as two-dimensional histograms, visualize the distribution of prediction errors across the dataset. These plots highlight

regions with high concentration of prediction points, offering a more holistic view of model performance beyond individual data samples.

By combining these two views, it becomes possible to both visually detect prediction bias or skewness and understand where the model performs best or struggles. These evaluations are performed both at the group level (Section 5.2.1) and for individual robots (Section 5.2.2), allowing deeper insight into how model performance varies with operational context and robot-specific patterns.

5.2.1 Group-Level Prediction Performance

To evaluate the predictive accuracy of the XGBoost model across robot groups, both scatter plots and density views were generated for the two operational configurations. Similar efforts in robotic cycle time prediction using simulation based approaches have shown improved accuracy and stronger interpretability (Cho et al., 2024). These visualizations are complemented by quantitative performance metrics, including the coefficient of determination (R^2), root mean squared error (RMSE), and the actual-to-predicted cycle time ratio (ta_ratio), as summarized in Tables 5.3b and 5.3d.

The ta_ratio provides a normalized measure of model alignment, defined as the actual cycle time divided by the predicted cycle time. Its interpretation is straightforward:

- If ta_ratio ≈ 1.00 → model predictions closely match actual performance.
- If ta_ratio > 1.00 → actual cycle time is longer than predicted (robot is slower than expected).
- If ta_ratio < 1.00 → actual cycle time is shorter than predicted (robot is faster than expected, i.e., more efficient).

To optimize model performance, hyperparameter tuning was performed separately for each configuration using randomized search with 3-fold cross-

validation over 10 candidate settings, resulting in 30 fits per group. The optimal parameters selected were:

- all_true: n_estimators = 300, max_depth = 7, learning_rate = 0.05, subsample = 0.8, colsample_bytree = 0.8

- simul_false: n_estimators = 200, max_depth = 5, learning_rate = 0.1, subsample = 0.8, colsample_bytree = 0.7

These configurations were used in the final training process before performance evaluation.

To improve the clarity and interpretability of the scatter plots, a random sample of 10,000 predictions was used instead of the full dataset, which contains hundreds of thousands of records per group. This approach preserves the distributional properties while avoiding overplotting that could obscure important trends. In contrast, the density plots include the full test set, providing a more complete visualization of prediction concentration across the range of cycle times.

all_true Configuration

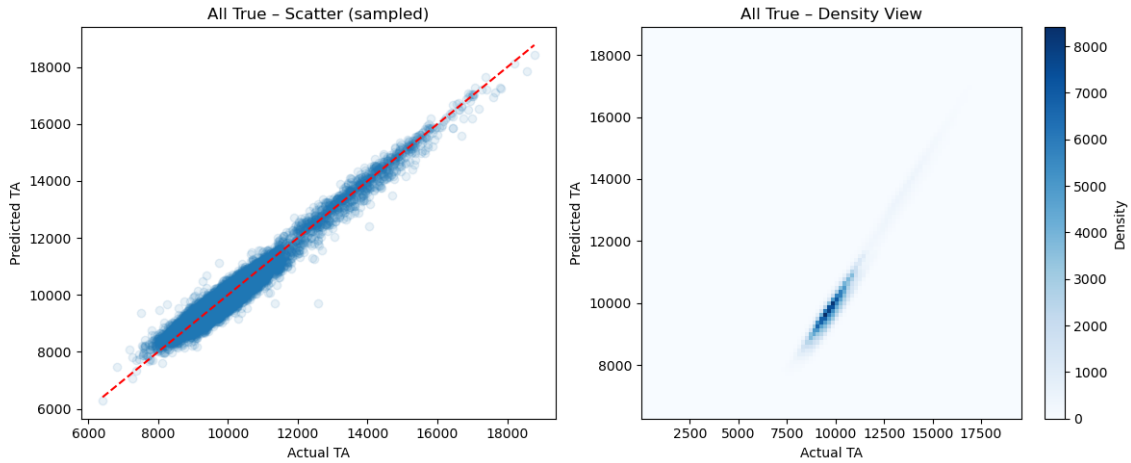


Figure 5.3a: Actual vs. predicted cycle time (ta) for the all_true group. Left: scatter plot (sampled); right: 2D density plot showing prediction concentration.
Note: The axis labels in the plots use "TA" to represent the target variable ta.

=== All True – 5-day split with tuned XGBoost ===

	robot_id	samples	R2	RMSE	ta_ratio_mean
0	21	48326	0.961	268.6	1.000
1	22	50180	0.963	284.1	0.998
2	31	56426	0.962	276.1	1.002
3	32	54315	0.956	299.0	1.002
4	41	50503	0.965	267.8	0.998
5	42	49631	0.963	273.9	0.997
6	51	57277	0.957	270.0	1.001
7	52	56523	0.949	297.5	1.000

Figure 5.3b: Model performance metrics for the all_true group. Includes robot-wise R^2, RMSE, and mean actual-to-predicted ratio.

Figure 5.3a presents the model predictions for the all_true group. The scatter plot (left) shows a tight clustering of predicted values along the diagonal reference line, especially in the 9000–11000 range. The high alignment and minimal spread indicate that the model successfully captures robot behaviour under synchronized conditions.

The density plot (right) further supports this, with a sharply concentrated region along the diagonal, highlighting prediction consistency across the dataset. As shown in Figure 5.3b, the model achieves R^2 values between 0.949 and 0.965, and RMSE values between 267 and 300. Although RMSE values may appear numerically high, this is expected given the absolute scale of the cycle time (ta), which typically ranges from 8000 to 15000. In relative terms, the error remains low.

Furthermore, the ta_ratio values hover near 1.000 for all robots, indicating that the model is neither systematically over- nor under-predicting cycle times. This level of calibration suggests high model fidelity across synchronized robot operations.

simul_false Configuration

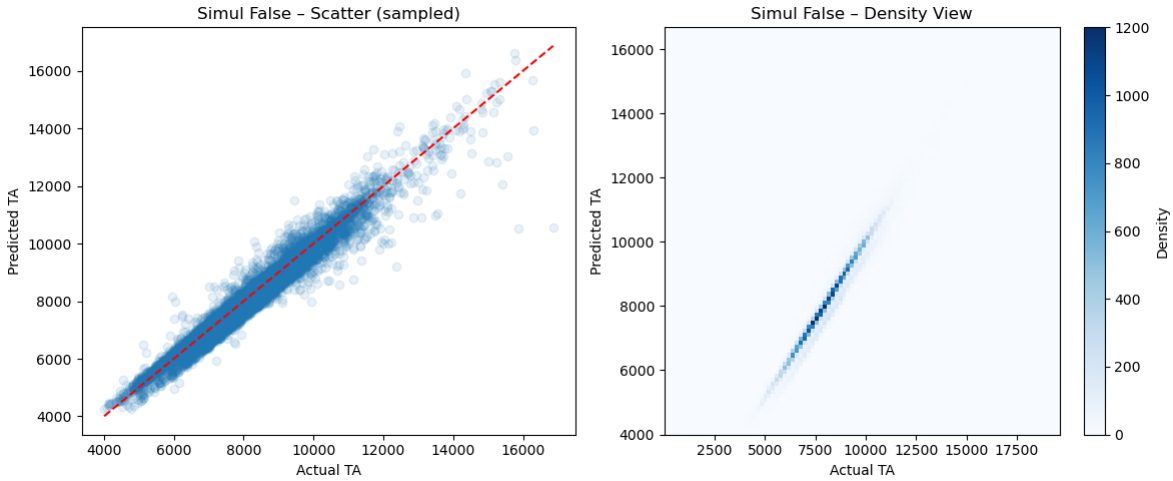


Figure 5.3c: Actual vs. predicted cycle time (ta) for the simul_false group. Left: scatter plot (sampled); right: density view.
Note: The axis labels in the plots use "TA" to represent the target variable ta.

=== Simul False – 5-day split with tuned XGBoost ===

	robot_id	samples	R2	RMSE	ta_ratio_mean
0	21	10768	0.953	336.1	1.000
1	22	10044	0.953	345.7	0.999
2	31	7764	0.961	336.2	1.000
3	32	7229	0.951	354.6	1.000
4	41	10197	0.967	268.0	0.999
5	42	9718	0.963	294.2	1.000
6	51	6748	0.971	267.0	1.000
7	52	6981	0.961	314.9	1.001

Figure 5.3d: Model performance metrics for the simul_false group, including robot-wise R^2, RMSE, and mean actual-to-predicted ratio (ta_ratio).

Figure 5.3c shows the corresponding results for the simul_false group. The scatter plot displays a slightly broader spread around the diagonal, especially at extreme values of ta, reflecting increased variability in robot performance when tasks are executed asynchronously.

The density plot illustrates a less concentrated prediction distribution compared to the all_true configuration. However, Figure 5.3d reveals that model performance remains strong, with R^2 values between 0.951 and 0.971 and RMSE values ranging from 267 to 354. As with the all_true group, the magnitude of RMSE reflects the large scale of ta, not model inaccuracy.

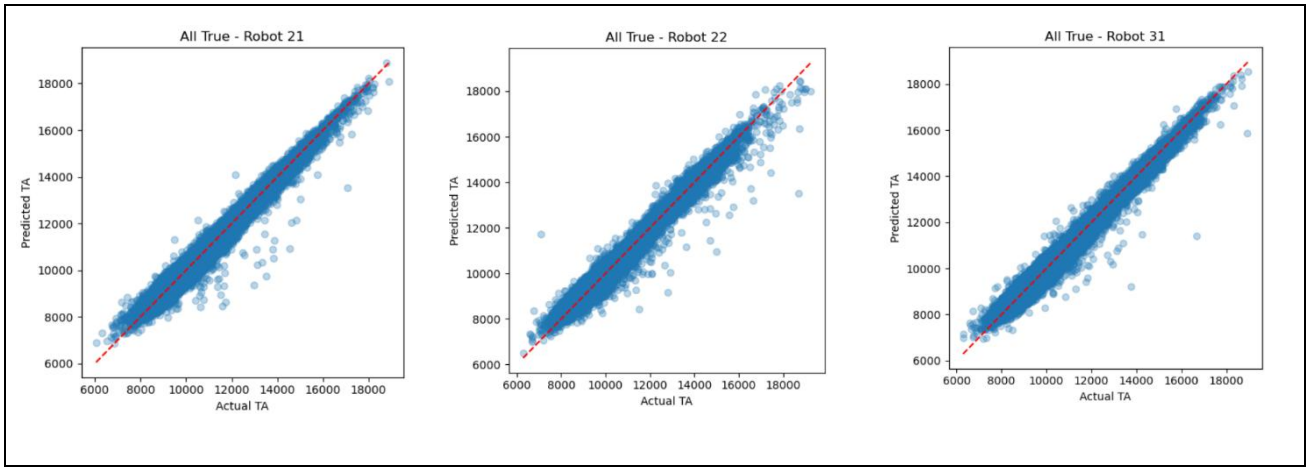
Importantly, the mean ta_ratio values remain close to 1.000, confirming that the model maintains unbiased estimates even under more dynamic, less predictable operational conditions.

Overall, the model demonstrates robust predictive performance across both configurations. While both groups show high accuracy, the all_true configuration yields tighter visual clustering and slightly lower RMSE variability. This aligns with earlier findings that synchronized robot operations result in more consistent cycle time behaviour, which improves the model's ability to generalize.

5.2.2 Per-Robot Prediction Accuracy

To further understand model performance across different robotic systems, scatter plots of predicted vs. actual ta values were generated for each individual robot. These plots are shown in Figure 5.4a for the "all_true" condition and Figure 5.4b for the "simul_false" condition.

Each subplot represents a single robot, displaying the alignment between predicted and true values. The dashed red line indicates the ideal prediction line (y = x), serving as a visual benchmark for accuracy.



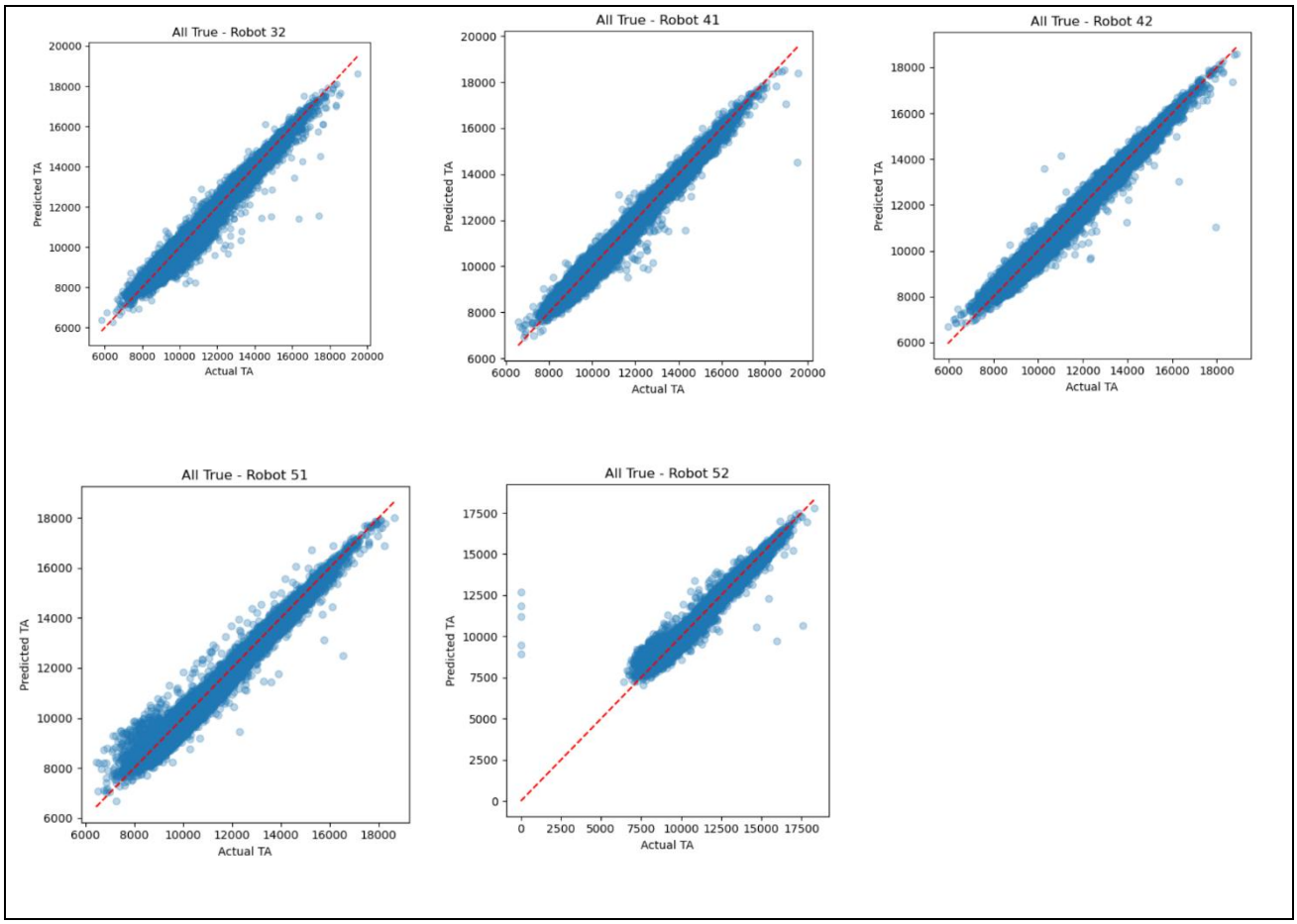


Figure 5.4a: Per-Robot Prediction Accuracy – all_true Condition
Note: The axis labels in the plots use "TA" to represent the target variable ta.

all_true Condition (Figure 5.4a)

Figure 5.4a: Under the "all_true" condition, most robot predictions closely follow the identity line, indicating strong model performance.

Key observations:

- For Robots 21 and 22, the models show very accurate predictions. The data points are extremely tightly grouped along the diagonal line, forming a very narrow and dense band. While the majority of points are highly concentrated, there are a few individual data points that are visibly slightly outside this densest core, though they remain relatively close to the main predictive trend.

- Robots 31 and 32 also show good overall agreement between predicted and actual values. However, compared to Robots 21 and 22, the spread of data points increases slightly, especially when 'Actual TA' values are higher (around 14,000 and above). In this upper range, the band of points around the diagonal becomes visibly wider.

- Robots 41 and 42 display a generally good fit. However, compared to Robots 21 and 22, the band of data points is slightly wider and more consistent around the diagonal line across their entire observed 'Actual TA' range. No individual points are noticeably far from the main cluster.

- Robot 51 shows a more noticeable increase in the overall scattering of its data points compared to the previous robot groups. The band of predictions around the diagonal is visibly broader, particularly clear in the middle range of 'Actual TA' values (around 8,000 to 14,000).

- Finally, Robot 52 shows the most significant difference from perfect prediction. Its data points are widely scattered across a large area, with many predictions far from the diagonal line. A prominent group of outliers is seen where 'Actual TA' is very low (around 0-1000) but 'Predicted TA' is much higher (around 10,000 to 12,500). Robot 52's 'Actual TA' range uniquely includes values as low as 0, distinguishing its operational profile.

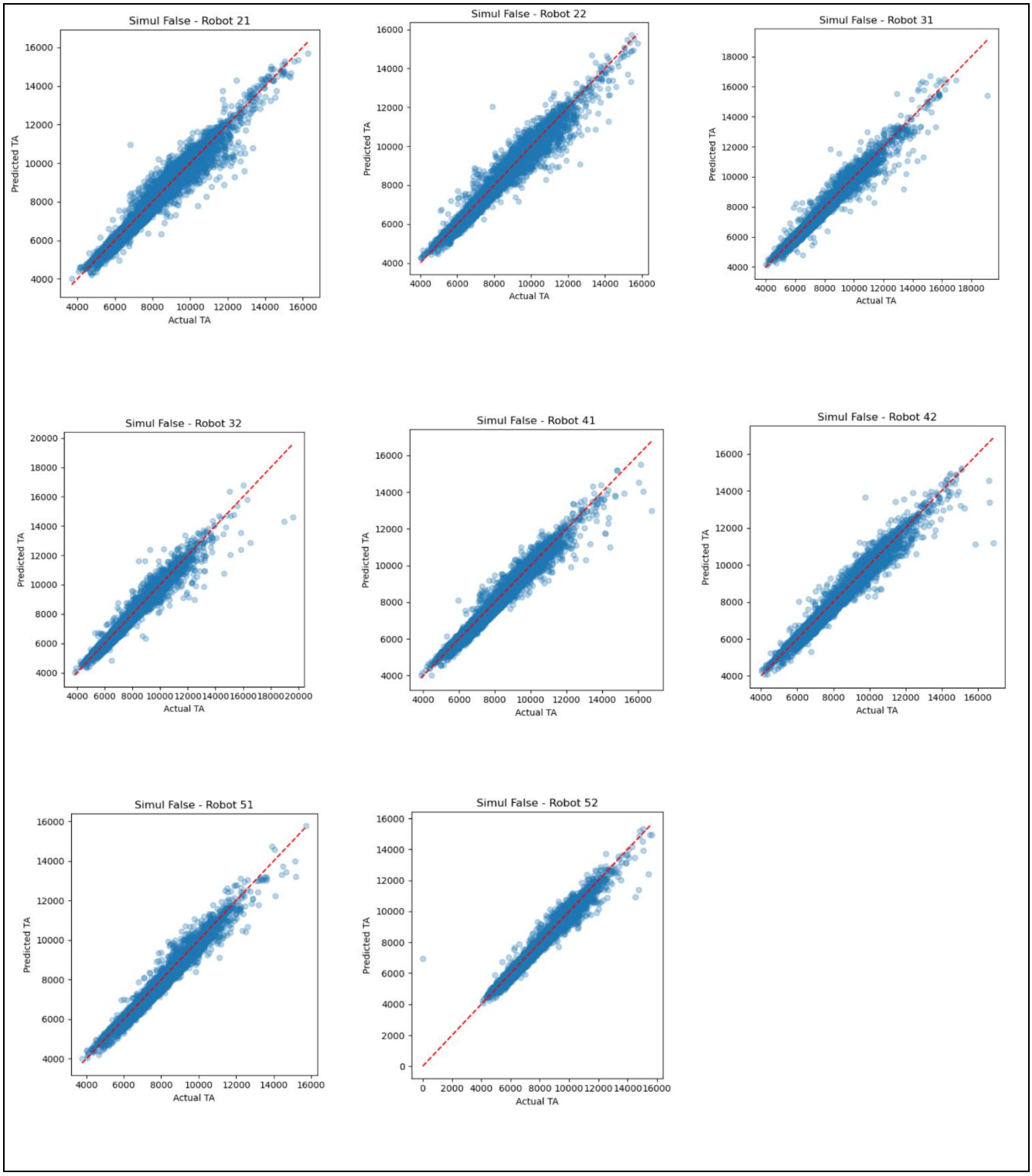


Figure 5.4b: Per-Robot Prediction Accuracy – simul_false Condition
Note: The axis labels in the plots use "TA" to represent the target variable ta.

simul_false Condition (Figure 5.4b)

Figure 5.4b shows the scatter plots of predicted versus actual ta values for each robot when the simultaneous_operation condition is set to False. These

plots help evaluate how the model performs under this specific operational context.

Key observations:

- For Robots 21 and 22, predictions are very accurate, forming an extremely tight and dense band along the diagonal line. A few individual points are slightly outside this core but remain close to the main trend.

- Robots 31 and 32 show good overall agreement, but prediction precision noticeably decreases at higher 'Actual TA' values (around 14,000 and above), where the band of points around the diagonal visibly widens.

- Robots 41 and 42 display a generally good fit, characterized by a slightly wider and more consistent spread of data points around the diagonal across their entire 'Actual TA' range compared to Robots 21 and 22.

- Robot 51 exhibits a more noticeable increase in overall scattering, with the prediction band visibly broader, especially in the middle range of 'Actual TA' values (around 6,000 to 12,000). This wider spread indicates less consistent and precise predictions.

- Finally, Robot 52 shows the most significant deviation from perfect prediction. Its data points are widely scattered across a large area. A prominent outlier exists where 'Actual TA' is very low (around 0-1000) but 'Predicted TA' is much higher (around 7,000). Robot 52's 'Actual TA' range uniquely includes values as low as 2.00, and its predictions are substantially less reliable.

Overall, the comparison between the all_true and simul_false conditions shows that the model generalizes well across different operational settings for

most robots. However, Robot 52 uniquely displays an 'Actual TA' range extending to very low values in both conditions and consistently exhibits the most significant prediction scatter and prominent outliers.

5.2.3 Model Evaluation on New Data

To assess the generalization capability of the trained regression model, it was evaluated on a new dataset collected from the client between April 1 and June 2, 2025. The same preprocessing steps were applied, including feature selection, column pruning, and grouping logic.

Two previously defined groups were selected for evaluation:

- all_true: where floor_pick, simultaneous_operation, and ev_positive are all True.
- simul_false: where floor_pick and ev_positive are True but simultaneous_operation is False.

For each group, features were extracted using the same structure as in training, ensuring consistency. Predictions were generated using the trained XGBoost regression model, and the results were evaluated using R^2 and RMSE metrics.

For the all_true group:

R^2 Score: 0.92
RMSE: 420.44

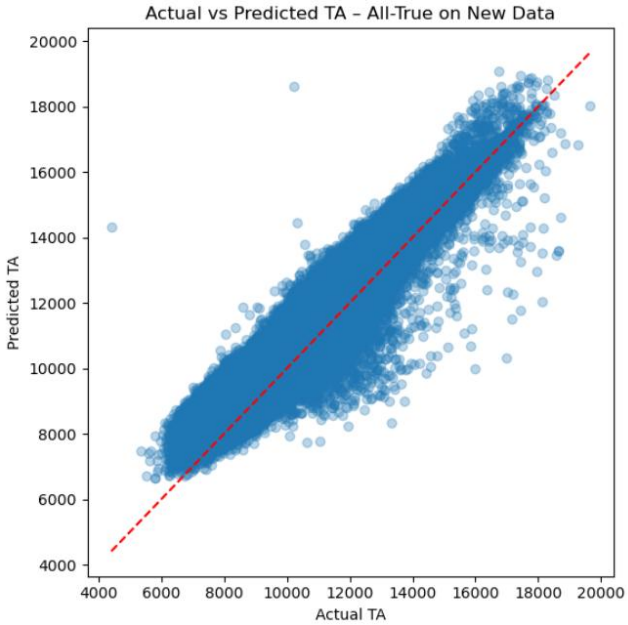


Figure 5.5a: Actual vs Predicted ta – all_true on New Data
Note: The axis labels in the plots use "TA" to represent the target variable ta.

For the simul_false group:

R^2 Score: 0.77

RMSE: 860.65

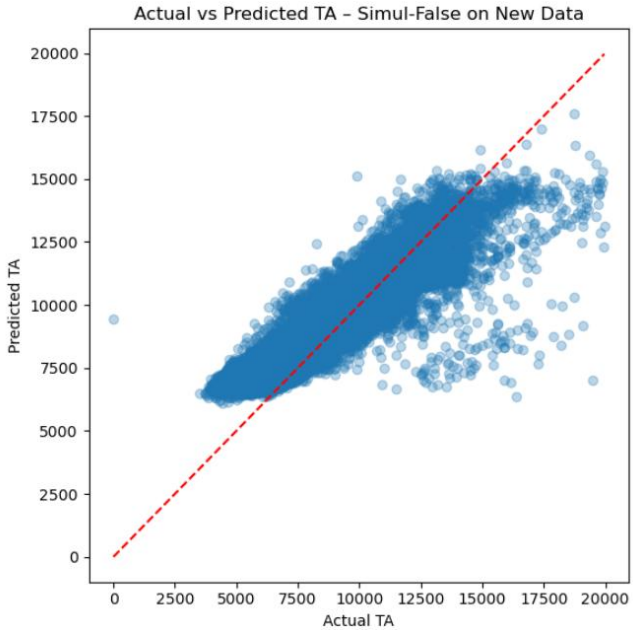


Figure 5.5b: Actual vs Predicted ta – simul_false on New Data
Note: The axis labels in the plots use "TA" to represent the target variable ta.

Figures 5.5a and 5.5b show how well the model predicted cycle times by comparing actual values with predicted ones. In both plots, each point represents a prediction, and the closer it is to the diagonal line, the better the prediction.

In the all_true group (Figure 5.5a), the points are closely packed around the diagonal. This means the model predicted the cycle times very accurately. The high R^2 score of 0.92 and the low RMSE of about 420 milliseconds confirm that the model works well when robots are operating with all conditions active.

In the simul_false group (Figure 5.5b), the points are a bit more spread out. The R^2 score is lower at 0.77, and the RMSE is higher at about 861 milliseconds. Even though the predictions are slightly less accurate, the model still follows the overall pattern. This shows that the model can still perform well even when robots are not working in simultaneous operation.

These results show that the model is strong and reliable when tested on new data. It can be used to monitor robot behaviour in real settings, even when conditions change. This supports its use in real factory environments where robots work under different situations.

5.3 SHAP Values: Feature Importance

To interpret the robot classification model and understand the contribution of each input feature, SHAP (SHapley Additive exPlanations) values were used. SHAP provides a consistent and interpretable measure of how much each feature impacts the model's predictions, both for individual samples and on average across the dataset. The SHAP value computation code is provided in Appendix 2.4.

This section analyses feature importance using two methods:

- Global SHAP analysis, which evaluates overall feature influence across the entire dataset.
- Per-robot SHAP heatmaps, which show how feature importance varies between robots.

Two operational conditions are considered: all_true and simul_false.

5.3.1. Global Feature Importance Across Conditions

Figure 5.6 presents two SHAP plots for the all_true condition. In this condition, all operational flags are set to True, and the model is trained to classify robots using all available signals.

- The plot on the left shows the SHAP summary, where each point represents one data sample. The horizontal axis shows how much a feature pushed the model's prediction higher or lower. The colour represents the feature’s actual value in that sample (red for high, blue for low).

- The plot on the right is a bar chart showing the average importance of each feature across all predictions, using the mean absolute SHAP value.

all_true Condition

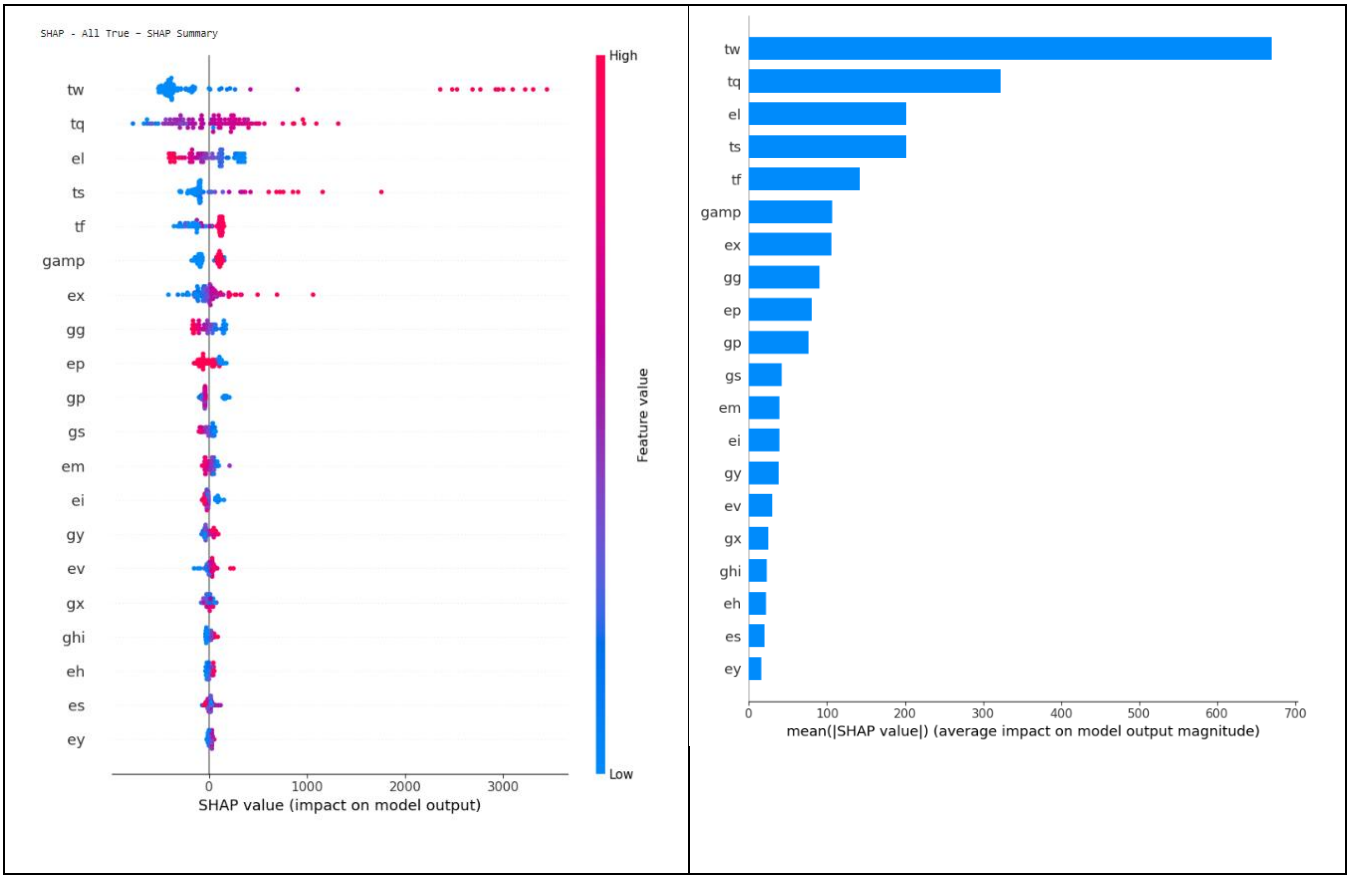


Figure 5.6: SHAP summary and bar plots for all_true

Together, these plots reveal which features the model uses most heavily to make decisions.

tw

This feature has the highest SHAP values overall, both in terms of spread (left plot) and average importance (right plot). Its values range widely, and high tw values often push predictions strongly in one direction, while low values pull them the other way. The model is highly sensitive to this feature, suggesting that different robots have clear patterns associated with it.

tq

The second most important feature. The red and blue points are spread across a wide SHAP range, meaning both high and low tq values influence predictions significantly. This indicates that tq varies meaningfully between robots and helps the model distinguish them.

el

This feature also plays a key role. The SHAP points are tightly grouped, but the feature still shows notable influence on the model. Compared to tw and tq, its effect is more balanced, contributing consistently but with less variation.

ts

This feature shows moderate impact. The SHAP summary plot shows that a few high values push the model's prediction up sharply, but for most samples, its contribution is small. It might be useful only for certain robots or certain scenarios.

tf

While still relevant, this feature has a smaller and more compact SHAP distribution. Its influence is limited to a smaller number of samples, suggesting that it plays a secondary role in model prediction.

gamp

This feature stands out because its SHAP values are balanced around zero and consistently small. The points are tightly clustered, and its average impact is moderate. This suggests that gamp is a stable and evenly contributing feature across robots. It does not push predictions strongly in either direction, but its presence helps the model maintain accuracy.

ex

This feature shows scattered influence. Some red and blue points are far from zero, indicating occasional strong impact, but overall, it ranks lower in average SHAP value. Its influence seems to depend on specific conditions.

gg

The feature gg has an average effect similar to ex. Its SHAP points suggest that it contributes more to certain samples but does not dominate model behaviour across the board.

ep

This feature shows a tight SHAP spread, with moderate importance. It likely plays a specific role for certain robots rather than being globally significant.

gp

This feature has a smaller range of SHAP values and ranks lower in average importance. It has a consistent but minor contribution.

gs

Very low SHAP values are seen for gs, both in the summary plot and in the average plot. It is likely not a key feature for prediction under the all_true condition.

em, ei

Both of these features rank near the bottom. Their SHAP values are small and tightly grouped, indicating that they have little influence on the model's output.

Remaining features (gy, ev, gx, ghi, eh, es, ey)

These are included in the SHAP summary plot but show very little influence on predictions. Their SHAP values are close to zero, and the average impact is minimal. The model likely does not use them meaningfully when all three flags are True.

Figure 5.7 shows SHAP plots for the simul_false condition, where floor_pick and ev_positive are True but simultaneous_operation is False. This setting represents a different operational mode, allowing analysis of how the model adjusts its behaviour when one condition is removed.

- The left plot is a SHAP summary plot, showing how each feature influences predictions across individual samples.
- The right plot is a SHAP bar plot, displaying the average absolute SHAP value for each feature.

Simul_false Condition

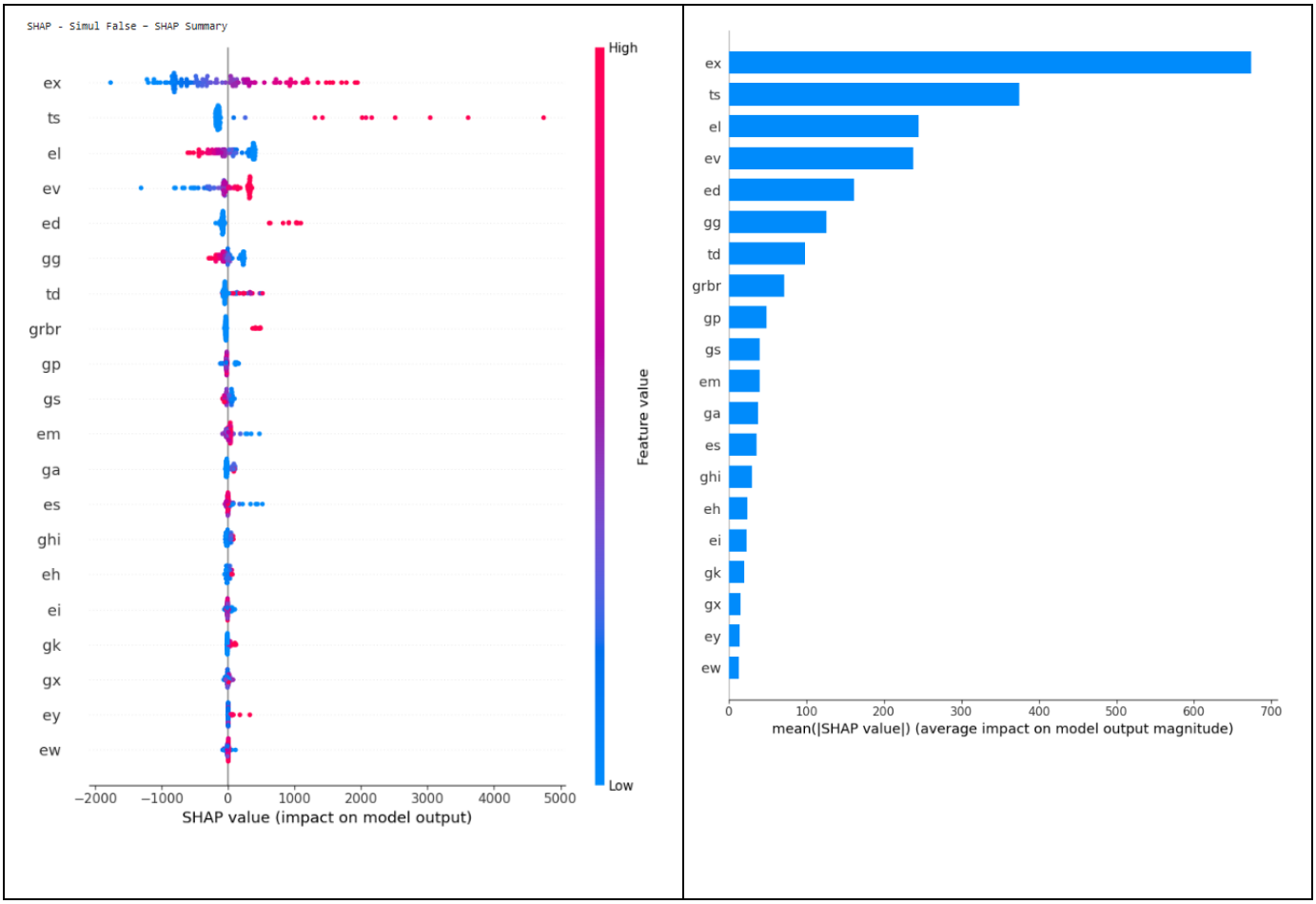


Figure 5.7: SHAP summary and bar plots for simul_false

ex

This is the most important feature in the simul_false condition. The SHAP points for ex are widely spread on both sides of the x-axis, meaning that both high and low values strongly affect the model's predictions. It has the largest average SHAP value, indicating the model depends on it the most when simultaneous_operation is False.

ts

The second most influential feature. It shows a clear horizontal spread, especially for high values, suggesting that it plays a major role in driving the model's output higher in some cases.

el

Ranks third in importance. Although its SHAP values are more tightly packed than ex and ts, it still contributes consistently. The model uses it regularly, but with less variation.

ev

ev has a more noticeable impact here than it did under all_true. It shows a balanced spread and appears among the top features, suggesting its importance increases when simultaneous operation is disabled.

ed

This feature has moderate importance. Its SHAP points are mostly close to zero but appear in a wide range of samples. It provides a small but consistent contribution to the model.

gg

Appears in the middle of the ranking. While its SHAP values are not large, they are visible in both plots. It likely helps the model for certain robots or specific cases.

td

Contributes modestly. There is some spread in its SHAP values, especially on the left side of the plot, meaning low values might be reducing predicted outputs in some cases.

grbr

This feature appears with small SHAP values in a few samples. Its influence is very limited and likely relevant only in rare conditions.

gp, gs

Both features show low SHAP values. They appear in the plot but with little horizontal spread. Their role is minor and likely specific to certain samples.

em, ga

These features have small and tightly grouped SHAP values. Their average impact is low, suggesting the model rarely relies on them.

es, ghi, eh, ei, gk, gx, ey, ew

All these features appear near the bottom of the SHAP summary plot with very small values. Their contribution is minimal, and the model likely ignores them in most predictions.

5.3.2. Per-Robot Feature Importance

Figures 5.8 and 5.9 present SHAP heatmaps showing the average contribution of each feature toward identifying individual robots under two conditions: all_true and simul_false. Each column corresponds to a robot, and each row to a feature. Brighter cells indicate that a feature had a stronger impact on the model's ability to recognize that robot based on its cycle time (ta). This per-robot breakdown reveals both dominant and supporting features that the model depends on, providing insight into how robot behaviour varies across operational contexts.

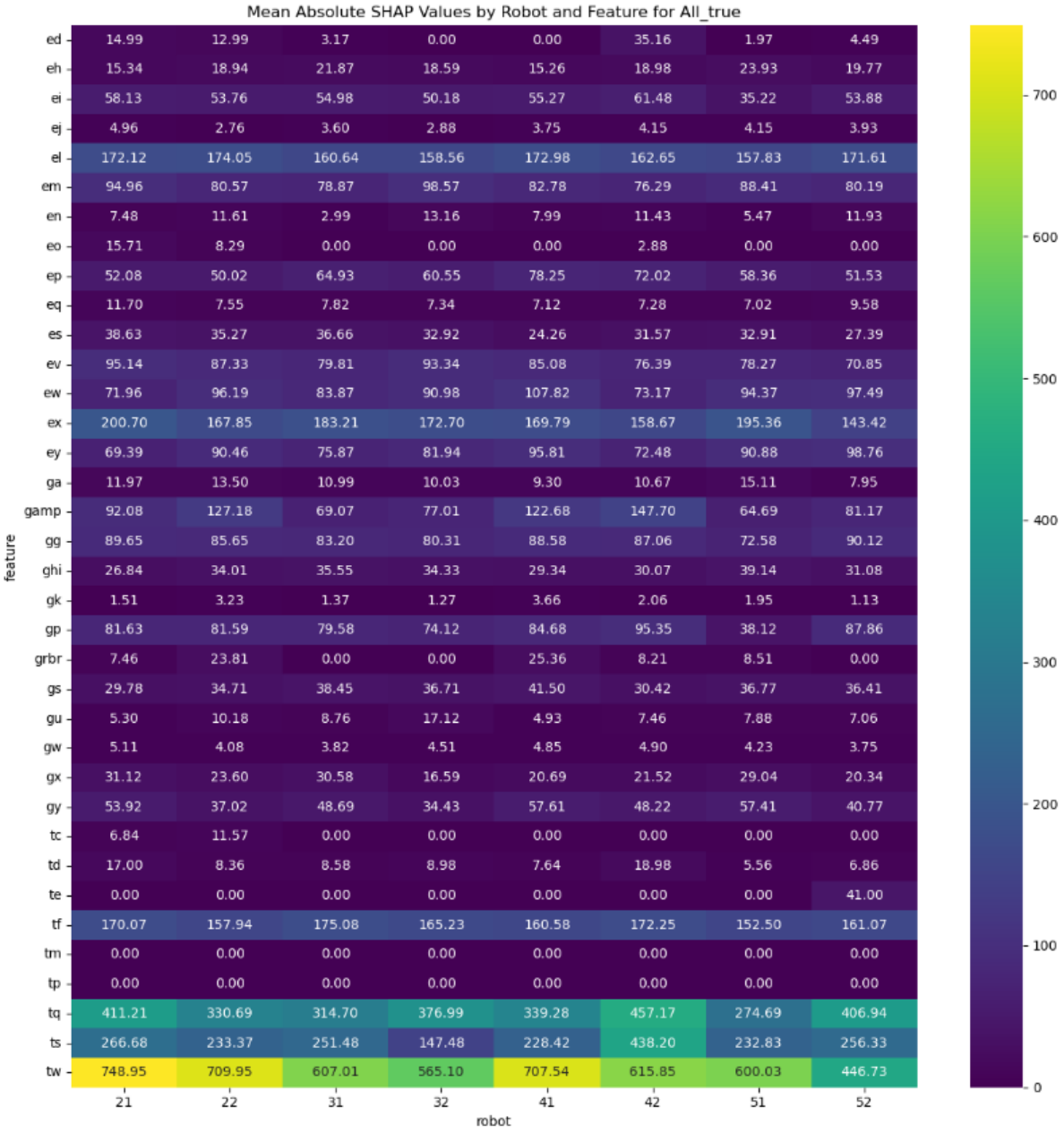


Figure 5.8: Mean Absolute SHAP Values by Robot and Feature for all_true.

All_true Condition (Figure 5.8)

In this setting, where all operational flags are True, the feature impact is dominated by three consistently high-contributing features across all robots particularly:

- tw stands out as the most dominant feature, consistently showing the highest SHAP values across all robots. Its impact is exceptionally strong, ranging from 446.73 to 748.95.

- tq is a consistently major contributor, often ranking as the second most impactful feature. Its high influence is evident with values ranging from 274.69 to 457.17. Notably, Robot 21 has a value of 411.21, Robot 42 has 457.17, and Robot 52 has 406.94.

- ts also plays a significant role, demonstrating a strong influence with values ranging from 147.48 to 438.20. Its highest impact is observed for Robot 42 (438.20).

These three features form a reliable pattern the model uses to separate robots under normal task conditions. Their consistently high values show they are highly informative for recognizing cycle time differences across devices.

Beyond these top contributors, several other features also exert a significant, high-moderate influence on the model's predictions:

- ex contributes at a significant moderate level across all robots, with values ranging from 143.42 to 200.70.

- tf contributes moderately across all robots, with values typically ranging from 152.50 (Robot 51) to 175.08 (Robot 31).

- el consistently shows a notable impact across all robots, with SHAP values generally ranging between 157.83 and 172.12.

- Features like gamp (values 64.69-147.70), ew (values 71.96-107.82), ey (values 69.39-98.76), em (values 76.29-98.57) are also consistently significant secondary contributors across most robots.

On the other hand, features like tm and tp have no effect on the model's predictions, as their SHAP values are zero for all robots. Similarly, te, eo, tc, and grbr generally has a minimal or no impact, with only a couple of robots showing slightly higher values.

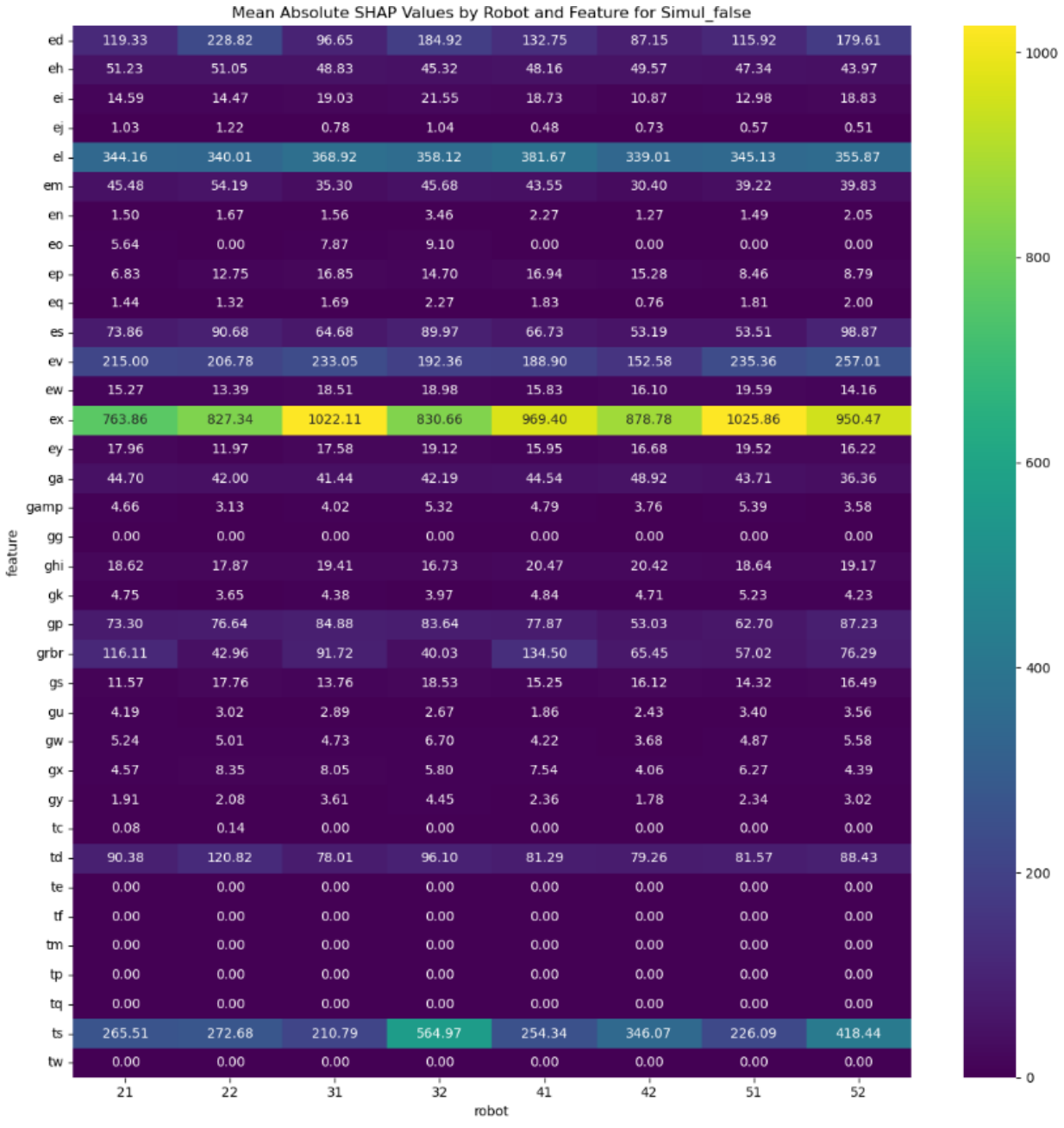


Figure 5.9: Mean Absolute SHAP Values by Robot and Feature for simul_false.

simul_false Condition (Figure 5.9)

When simultaneous_operation is False, the model’s behaviour changes significantly:

- ex becomes the primary distinguishing feature, with extremely high values ranging from 827.34 to 1025.86 This feature clearly dominates model decisions under this setting.

- ts also plays a very significant role, demonstrating strong influence with values ranging from 210.79 to 564.97. Notably, it shows a particularly high impact for Robot 32 (564.97), making it a crucial differentiator.

- el is another consistently major contributor, often ranking as one of the most impactful features alongside 'ex'. Its high influence is evident with values consistently in the 300s (e.g., 339.01 to 381.67).

Beyond these top three, several other features also exert a significant, high-moderate influence on the model's predictions:

- ev consistently contributes at a significant moderate level across all robots, with values typically ranging from 152.58 to 257.01.

- ed also shows consistent moderate to high values, generally ranging from 87.15 to 228.82.

- grbr and td show moderate impact, with 'grbr' ranging from 40.03 to 134.50 and 'td' from 78.01 to 120.82.

Conversely, features such as gg, te, tf, tm, tp, tq, and tw consistently show zero SHAP values, meaning they have no impact on the model's predictions. tc, eo, and ej also exhibit minimal to no impact, with values either at or very near zero for most robots.

This reveals a clear feature shift: under changed task conditions, the model depends on a completely different feature set to maintain its predictive accuracy.

A comparative analysis of the SHAP results under the all_true and simul_false conditions highlight several key differences in how the model interprets robot behaviour:

- In the all_true condition, the model relies on one set of core features, whereas in simul_false, an entirely different group emerges as dominant.

- Features that were central to prediction during coordinated operation (e.g., tw, tq) become completely irrelevant under isolated task settings.

- Conversely, features like ex and el, which had moderate roles under all_true, take over as primary indicators in simul_false.

- This indicates that robot behaviour expresses itself differently depending on whether robots are operating collaboratively or independently.

Some features may appear unimportant in the simul_false setting not because they lack predictive value, but because they are inactive, constant, or constrained by the operational mode. In such cases, their signals may not vary meaningfully, preventing the model from leveraging them for differentiation. This reinforces that SHAP values reflect more than just feature strength. They also capture whether a feature is contextually active and informative. Understanding this distinction is essential when interpreting model behaviour across different task conditions.

The SHAP heatmaps show that the model relies on different features depending on operational conditions, confirming its adaptability. Robots such as 32, 42, and 52 show more distinct feature patterns, making them easier to

identify. While some features contribute strongly across all robots, others only play a role under specific conditions or for specific devices. This variability reflects the complex nature of robot behaviour and suggests that both dominant and supporting features are valuable. Features with consistently low importance may be inactive in certain settings or offer limited differentiation and can be reviewed for potential simplification in future models.

To further support the finding that robots show distinct behavioural patterns, an additional classification experiment was performed using the same anonymized feature set. An XGBoost classifier was trained to predict robot identity based only on internal signal data. Time-based and task-related fields were excluded to focus solely on the robots' internal behaviour. Before training, all numeric features were scaled using standard normalization, and the target robot ID was label-encoded. The dataset was split into training and test sets using stratified sampling to ensure balanced representation of all robots, with 70 percent for training and 30 percent for testing. The classifier achieved a high accuracy of 99.04 percent on the test set. This confirms that each robot has a consistent and unique signature in its sensor patterns. The result reinforces the SHAP findings and shows that robot-specific behaviours are reliably distinguishable from signal data alone, even without explicit time or task context.

5.3.3 Robot Efficiency Ranking Based on Cycle Time Estimation

To compare the operational efficiency of individual robots, each robot’s actual cycle time (ta) was compared to model-predicted values using an XGBoost regression model. For every robot cycle, a new feature called ta_ratio was computed by dividing the actual ta by the predicted ta. A ratio close to 1 means the model's prediction closely matches the robot's actual cycle time, which suggests consistent and expected behaviour. Large deviations might indicate inefficiencies or unexpected conditions (See Appendix 2.6 for the efficiency ranking code).

The ta_ratio was calculated for each robot under both all_true and simul_false conditions. Then, robots were ranked using two key metrics:

- Average cycle time (avg_ta): Lower average times imply faster operation.
- Mean ta_ratio: A value close to 1 implies the robot is behaving as predicted.

These two ranks were combined into a single efficiency score by averaging the individual rankings. Figure 5.10 shows the results, sorted by overall efficiency.

	deviceid	avg_ta	std_ta	mean_ta_ratio	count	rank_by_avg_ta	rank_by_mean_ta_ratio	efficiency_rank
0	21	9729.958789	1572.418739	0.999967	117178	1.0	2.0	1.5
1	52	9790.419410	1478.461024	0.999927	125319	2.0	1.0	1.5
2	51	9845.930260	1491.417451	0.999974	126255	3.0	4.0	3.5
3	22	9912.076767	1659.135860	0.999971	119427	5.0	3.0	4.0
4	31	9897.080957	1586.202934	0.999981	127500	4.0	7.0	5.5
5	32	10012.159665	1614.072561	0.999975	122726	8.0	5.0	6.5
6	41	9929.534053	1613.268754	0.999978	118784	7.0	6.0	6.5
7	42	9925.257158	1591.870648	0.999984	115781	6.0	8.0	7.0

Figure 5.10: Robot Efficiency Ranking Based on Average Cycle Time and Prediction Alignment

The analysis reveals that robots 21 and 52 are the most efficient, with low cycle times and strong alignment with the model's predictions. Robots 22, 31, and 51 also rank highly. These insights help identify well-performing robots and can guide performance audits, maintenance planning, or deeper investigations into underperforming systems.

5.4 Sobol Sensitivity Analysis

To evaluate the global influence of each feature on model predictions, Sobol sensitivity analysis was conducted using total-order (ST) indices. This method provides a robust measure of how variation in each feature contributes to the variance in predicted cycle time (ta). Unlike SHAP, which explains individual

predictions, Sobol indices capture average feature importance across the entire input space, offering a complementary global perspective.

The analysis was performed for two operational settings: all_true (where all control signals are active) and simul_false (where simultaneous_operation is disabled). For both, the analysis included global and per-robot sensitivity investigations.

5.4.1 Global Sensitivity Patterns

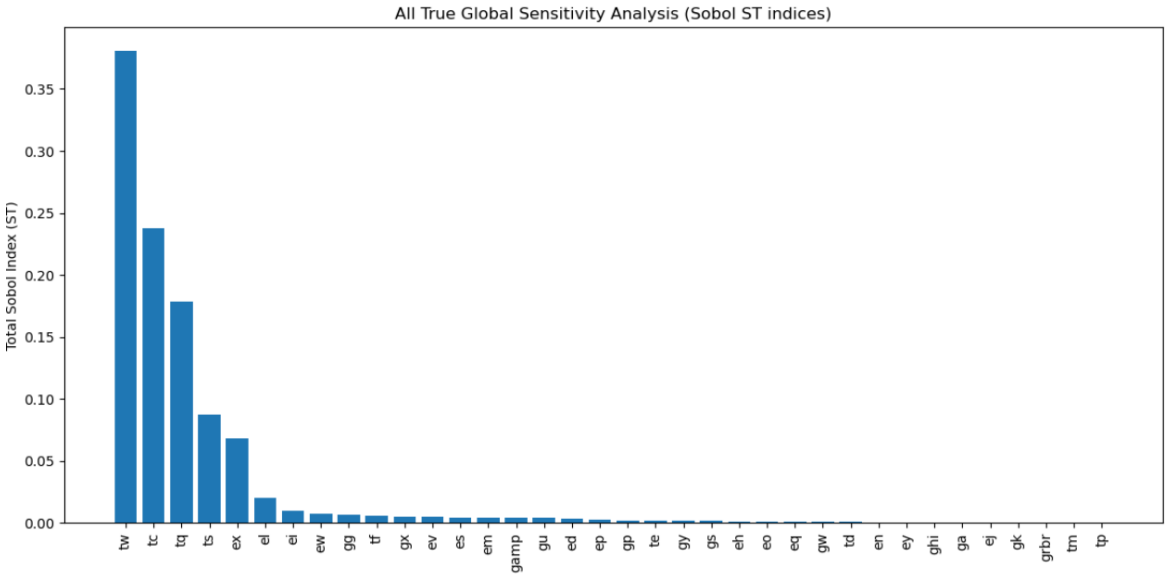


Figure 5.11: Sobol Total Sensitivity (ST) Indices – all_true (Global)

Figure 5.11 shows the total Sobol indices for all_true. The feature tw (cycle time) emerges as the most influential variable with an index around 0.38, indicating it alone accounts for over a third of the total output variance. Other important features include tc (~0.24), tq (~0.18), and ts (~0.09). Together, these four features contribute to the majority of the model's global sensitivity. Features like ex, el, and ei have low but non-negligible contributions, while the remaining features have minimal influence (ST < 0.02).

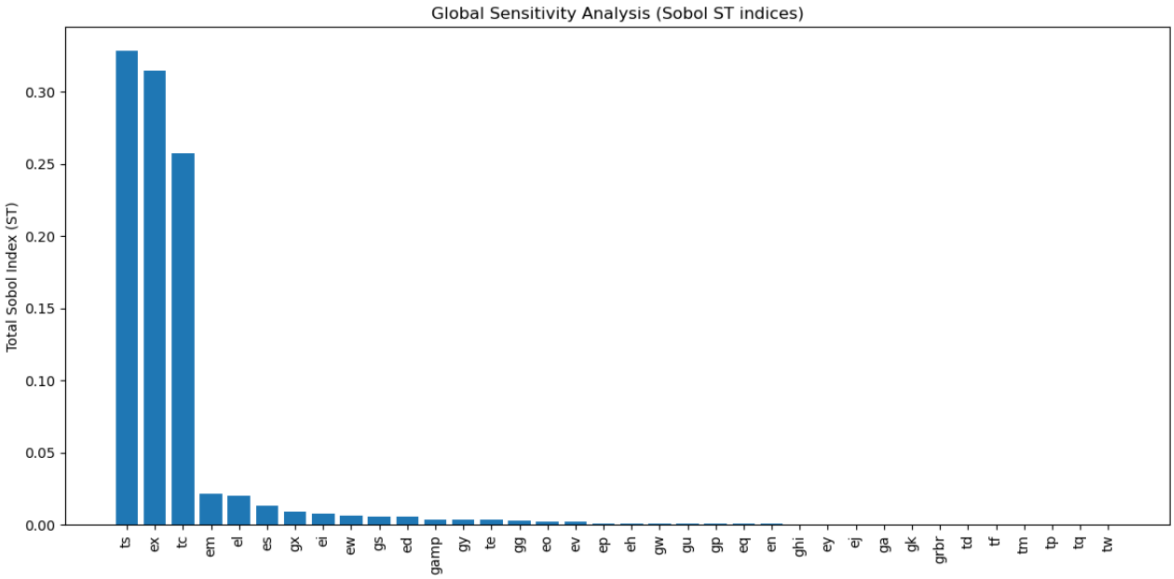


Figure 5.12: Sobol Total Sensitivity (ST) Indices – simul_false (Global)

In Figure 5.12, for the simul_false setting, a different pattern emerges. The dominance of tw disappears (ST ≈ 0), and features ts, ex, and tc become the primary drivers of output variance, with ST values of approximately 0.33, 0.32, and 0.26, respectively. This shift indicates that when simultaneous operation is disabled, the model no longer relies on tw and instead prioritizes task-related or alternative signal features.

This contrast confirms the model's context-aware adaptability, where sensitivity focus shifts in response to operational mode changes.

5.4.2 Per-Robot Sensitivity Insights

Figures 5.13 and 5.14 display Sobol heatmaps per robot under both settings. Each cell in the heatmaps represents the ST index of a feature for a specific robot, allowing a fine-grained analysis of feature influence per unit.

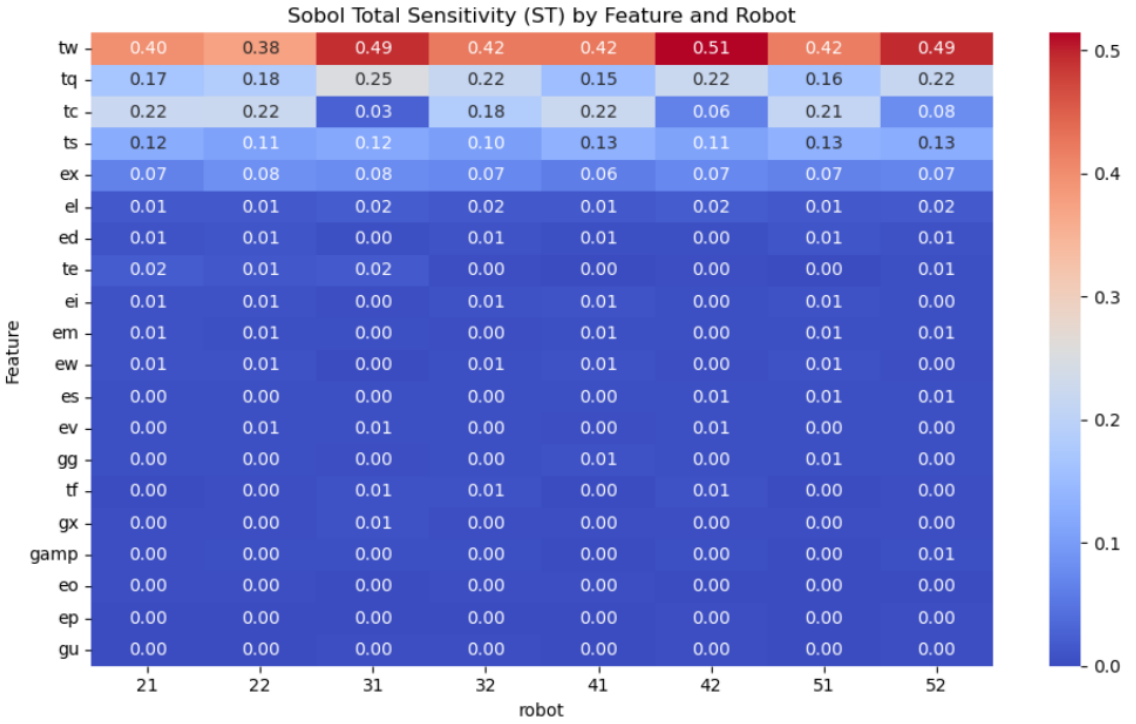


Figure 5.13: Sobol Total Sensitivity (ST) by Feature and Robot – all_true

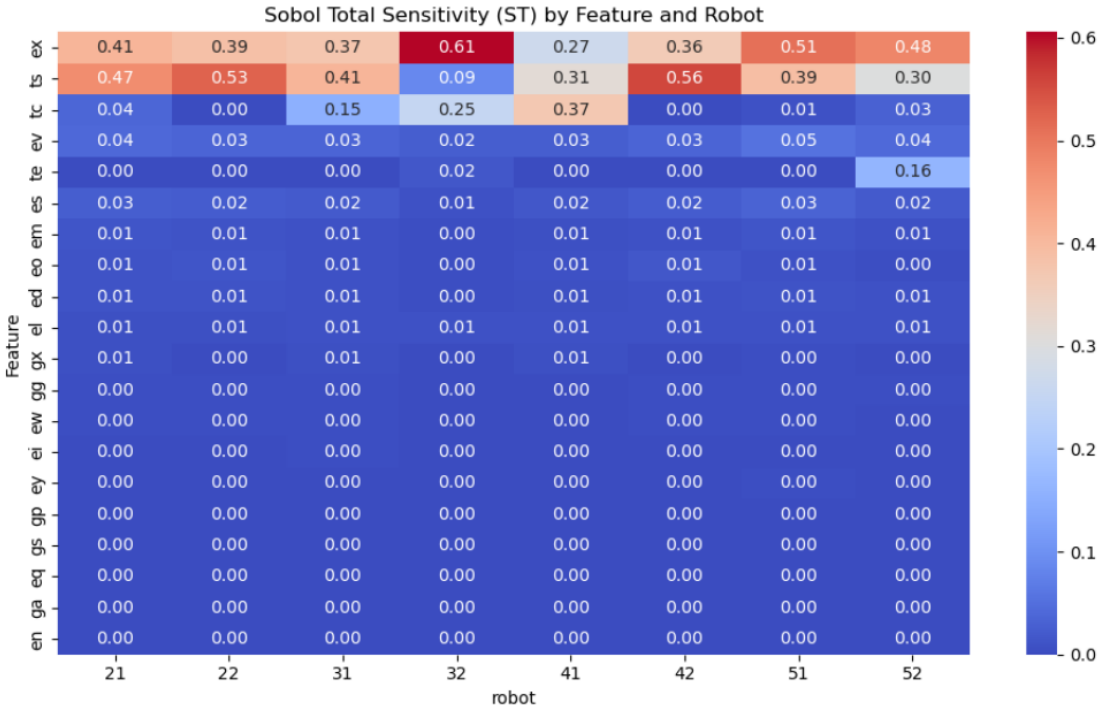


Figure 5.14: Sobol Total Sensitivity (ST) by Feature and Robot – simul_false

Robot 21:

In all_true, the most sensitive features are tw (0.40), tc (0.22), and tq (0.17). In simul_false, importance shifts to ts (0.41) and ex (0.41), with tc (0.04) playing a minor role. This shows that Robot 21 has a highly distinctive behaviour signature that the model adjusts to depending on the setting.

Robot 22:

Under all_true, similar influence is seen for tw (0.38), tc (0.22), and tq (0.18). In simul_false, ts (0.39) and ex (0.39) dominate, showing a clear re-prioritization. The model's attention is redirected away from tw, which becomes nearly zero in this condition.

Robot 31:

Displays the highest sensitivity to tw in all_true (0.49), indicating strong reliance on this feature. However, in simul_false, the key features are ts (0.37), ex (0.37), and tc (0.15.

Robot 32:

Also, highly dependent on tw in all_true (0.42). In simul_false, the model assigns more importance to ex (0.27) and tc (0.31), while ts (0.36) becomes the dominant driver. The consistent redistribution of importance highlights the model’s flexibility.

Robot 41:

Has the highest ST for tw in all_true (0.51). Under simul_false, ex becomes the most sensitive (0.36), followed by ts (0.27), and tc (0.37), indicating balanced feature reliance.

Robot 51:

In all_true, the top features are tw (0.42), tc (0.21), and tq (0.16). In simul_false, importance shifts strongly to ex (0.51) and ts (0.30), with very low influence from tw.

Robot 52:
In all_true, tw remains important (0.49), similar to Robot 31. In simul_false, ex (0.48) and ts (0.30) dominate, while tw again becomes negligible.

Top features in both contexts:
Only a few features (tw, ts, ex, tc) consistently show high ST values, although their ranking and presence change significantly depending on the condition.

Conditionally inactive features:
Some features, like tw, are active in all_true but show zero importance in simul_false. This may indicate that their corresponding signals are disabled or constant under certain operational settings.

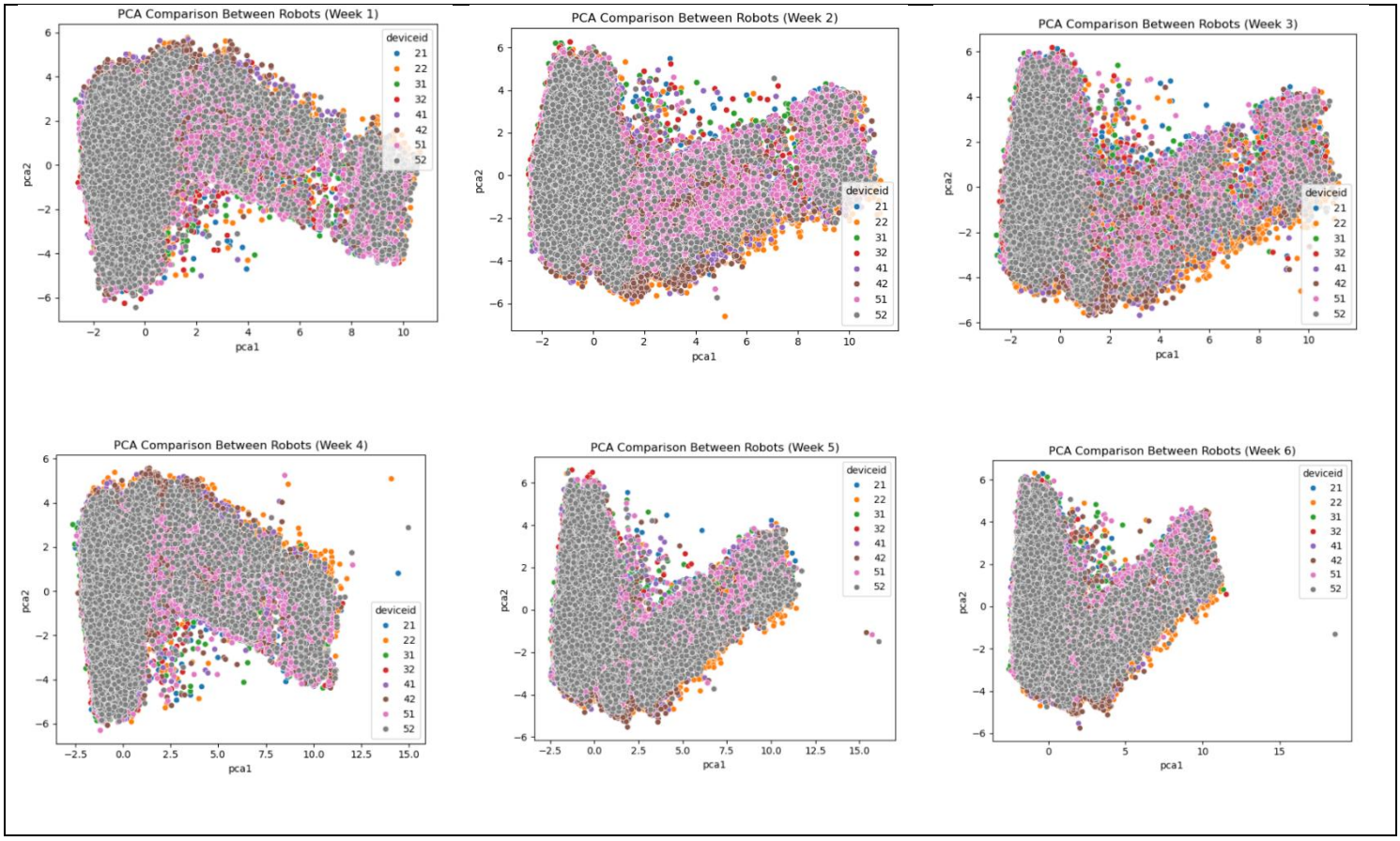
Globally negligible features:
Features such as gy, gs, gx, gamp, ep, eo, gu, gp, ey, eq, ga, and grbr show ST values consistently near zero across all robots. These variables can be considered redundant in the current modelling context and may be candidates for removal or revision.

Sobol sensitivity analysis complements the SHAP findings by offering a global view of feature influence. It reinforces the model's strong dependence on a few dominant features while also uncovering subtle, robot-specific variations. Both methods consistently highlight a small set of key features, such as tw, tq, and ts under the all_true condition, and ex and ts under the simul_false condition, as critical to accurate cycle time (ta) prediction. While SHAP captures local, instance-level effects, Sobol quantifies overall sensitivity across the entire input space. Together, they show that the model adapts flexibly to different operational contexts by shifting its reliance depending on whether simultaneous operation is enabled. This dynamic adjustment supports robust robot identification under varying task conditions. Additionally, the agreement between SHAP and Sobol enhances trust in the model's interpretability, while features with consistently low importance in both analyses point to potential areas for simplification in future model development.

5.5 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a tool that helps reduce complex information into a simpler form, making it easier to compare different robots. Instead of looking at all sensor signals at once, PCA turns this high-dimensional data into two main values: PCA1 and PCA2. These are the horizontal and vertical axes in the plots and show the biggest patterns in robot behaviour.

Each point in the plots in Figure 5.15 represents one cycle from one robot. The colours show which robot each point belongs to. If a group of points from one robot stays in a small area, it means that robot is behaving in a steady and similar way. If the points are spread out or move from one week to another, it may indicate changes in tasks, usage conditions, or performance. The PCA implementation used to generate these visualisations is detailed in Appendix 2.5.



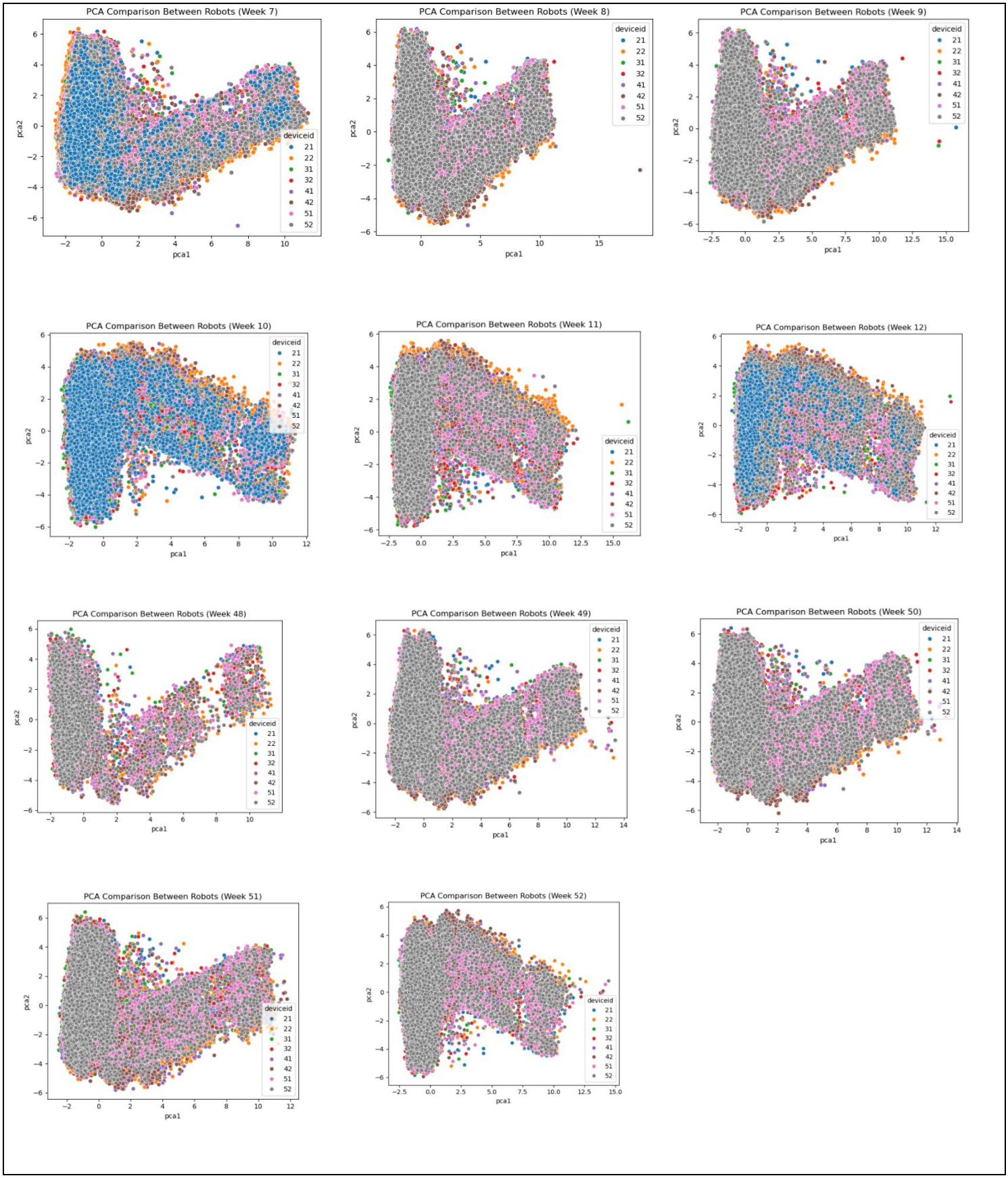


Figure 5.15: PCA Comparison of Robot Behaviours Across 17 Weeks (Weeks 1–12 and 48–52)

Timeline and Weekly Structure

The data used in this study covers the time period from December 1, 2024 to March 23, 2025. The weekly breakdown shown in the plots follows the ISO calendar system. This means:

- Week 1 starts from the first full week of January (even if some data is from late December).
- The last weeks (like Week 48, 49, 50, etc.) actually come from December 2024.
- Weeks do not appear in strict date order in the dataset because ISO week numbers loop back at the end of the year.

Because of this, plots for Week 48 to Week 52 appear after Weeks 1 to 12, even though those weeks come from earlier calendar dates. This is important when analysing robot behaviour over time, since some shifts may look like they happen later, but actually occur earlier in the timeline.

Key Observations from the PCA Plots

The PCA plots (Figure 5.13) provide useful insights into the behaviour of individual robots across different weeks. The following patterns were observed based on visual analysis:

Robot 52 shows up in nearly every week and typically has the largest number of data points. Its cluster is usually dense and positioned centrally in the PCA plots. This suggests that Robot 52 was operating regularly and under stable conditions throughout the recorded period. Because of its consistency, it serves as a useful reference or baseline for comparing other robots' behaviours.

Certain robots demonstrate shifts in their PCA positions from week to week. For example:

- Robot 21 displays a significant expansion in Week 7, indicating more varied activity or a change in workload compared to previous weeks.

- Robot 31 and Robot 32 show shifting positions in Weeks 3 and 48, possibly reflecting a change in task type, environment, or operation schedule. These movements away from usual locations in the PCA space suggest the model is detecting real behavioural changes in those weeks.

The shape and compactness of robot clusters vary across weeks:

- In earlier weeks (Weeks 1-4), clusters are relatively tight, indicating consistent robot operation with little variation.

- In later weeks like Week 5, Week 7, and especially Week 49, the points become more spread out. Robots such as 22 and 41 show larger dispersion, possibly due to increased variability in their tasks or changes in system configuration.

In several weeks (such as Weeks 2, 10, and 12), multiple robots cluster close together in the PCA space. This suggests those robots were performing similar types of work or operated under similar conditions during those weeks. These overlaps imply shared task structures or coordinated usage patterns.

In Weeks 48, 49, and 52, some robots deviate clearly from the main clusters. These outliers may reflect irregular events such as:

- Temporary system changes
- Maintenance or recalibration periods
- New task assignments or unusual workload distribution

These deviations are important, as they may help detect changes in performance over time or highlight moments when a robot operated outside its

normal behaviour range. The use of PCA and clustering to detect behavioural patterns in robot systems is supported by recent robotics literature, including PCA-based analysis for mapping motion behaviours (Saviano et al., 2024).

6 DISCUSSION

6.1 Interpretation of Key Findings

This study explored how industrial robots behave over time using machine learning, sensor data, and explainability tools. The main findings from the analysis are explained below.

Robot performance changes over time

Most robots showed shifts in their cycle time during the study period. These changes were not always large or sudden. In many cases, the trends evolved slowly. The use of change point detection helped identify when these shifts happened. For example, Robot 21 had a clear performance change in early February 2025, while others like Robot 52 remained steady for longer. These findings suggest that even when there are no obvious faults, robot behaviour can still change due to operational pattern, or routine maintenance.

Task settings influence robot behaviour

Robots responded differently under the two tested operational settings. In the all_true condition, where floor_pick, simultaneous_operation, and ev_positive were all active, performance was more variable. In the simul_false condition, where only floor_pick and ev_positive were active, the robots worked more steadily. This may be because simultaneous tasks (where one robot does more than one job during the same cycle) introduce more complexity and variation in how the robots perform.

Similar robots behave differently

Although the robots were of the same type and shared the same workspace, their behaviour was not identical. Some robots showed consistent performance over time, while others showed more fluctuations. For example, Robot 52 operated regularly across all weeks with few changes, but Robots 31 and 32 showed more variation. These differences may be due to how each robot was used, its specific task, or maintenance schedule.

Machine learning models predicted performance well

The trained XGBoost models predicted robot cycle time accurately across both operational settings. The prediction quality was strong at both group level and per-robot level. Visualisations showed tight alignment between predicted and actual values for most robots. Slightly more error appeared under the simul_false setting, likely due to changes in behaviour when simultaneous tasks were turned off.

Explainability tools revealed useful insights

Using SHAP and Sobol sensitivity analysis, the study identified which sensor signals most influenced model predictions. Under all_true, features like tw, tq, and ts were most important. Under simul_false, features like ex, ts, and el became more relevant. This shift shows that the model adapts to different conditions and uses the features that matter most in each setting. Both SHAP and Sobol agreed on the importance of the key features, which increases confidence in the results.

Visual analysis helped compare behaviour over time

PCA was used to project robot behaviour into two dimensions for each week. This made it easier to compare how robot activity changed over time. Some weeks showed clear changes in behaviour patterns or task conditions,

especially in Week 7 and Week 49. Robots like 21 and 41 moved further away from their usual behaviour, possibly due to changes in schedule or task type.

Different methods worked together

Each method gave a different view of the robot system. Change point detection showed when robot behaviour changed. SHAP and Sobol explained which signals influenced performance the most. PCA visualised trends over time. Together, these tools gave a full picture of robot behaviour and how it evolved under different conditions.

In summary, the study showed that robot performance is shaped by both task settings and time-based changes. Even when robots do similar work, their behaviour can still differ. The use of explainable machine learning and visual tools can help detect these changes and support better robot monitoring.

6.2 Practical Implications for Robot Monitoring

The findings from this study offer several useful insights for improving how robots are monitored in real industrial environments.

Early detection of performance changes

By using change point detection on cycle time trends, it is possible to identify when a robot's behaviour starts to shift. These changes might not be visible during routine checks but can still affect performance over time. Detecting them early allows for more informed maintenance planning or investigation before larger issues develop.

Understanding how task settings affect behaviour

The results show that robot behaviour is affected by operational settings. Robots working under the all_true condition, which includes simultaneous

tasks, tend to have more variation in their cycle time. In contrast, those under the simul_false condition behave more consistently. This means that monitoring systems should take task context into account when analysing robot performance. Comparing robots without considering their operating mode may lead to wrong conclusions.

Feature monitoring can support diagnostics

The study identified specific sensor features that are closely linked to cycle time. Features like tw, tq, and ts were important under one setting, while ex, ts, and el became more relevant under another. Monitoring these key signals in real time can help explain why a robot's performance is changing and provide more targeted feedback to engineers or operators.

Better model design for real-world use

The machine learning models used in this study were able to predict robot performance with high accuracy. Importantly, the models also adjusted to different task settings by focusing on different features. This shows that flexible and explainable models can be used in practice to track performance across changing conditions. They can help identify when robots behave differently from usual, even if there are no technical errors.

Visual tools make robot data easier to understand

Tools like PCA helped simplify complex sensor data into patterns that can be visually analysed. This makes it easier for engineers and managers to compare robots and detect unusual behaviour over time. Such tools can support decision-making without needing advanced technical skills.

Overall, the methods used in this study can support more proactive and informed robot monitoring. By combining predictive models with explainability and visual analysis, companies can better understand how robots are performing and respond more quickly when changes occur.

6.3 Methodological Limitations

This study has provided valuable insights into robot performance, but several limitations should be noted.

Limited scope and generalisation

The findings are based on data from a specific group of robots in one industrial setting. These results may not apply to other types of robots, tasks, or factory environments. To confirm broader relevance, similar analyses would need to be tested in different contexts.

Reliance on predefined group labels

The full dataset was divided into eight groups based on working conditions defined by three binary input signals. Two specific groups, called all_true and simul_false, were selected for focused analysis. These groupings depended on signal flags provided in the dataset. If those flags were inaccurate or incomplete, the analysis may not fully reflect true task conditions.

Anonymised and reduced feature set

Each dataset included numeric signal values collected from robot cycles. However, all signal names were anonymised to protect the client's confidentiality. Time-based features were also removed to keep the focus on internal signals. While this makes the data safer to use, it limits the ability to fully interpret what each signal represents or how it links to physical processes. This can make model explanations harder to connect to real-world operations.

Focus on cycle time only

Cycle time was used as the main performance indicator. While useful, it does not show other aspects such as energy use, task quality, mechanical stress,

or safety margins. A more complete evaluation would include other performance factors alongside cycle time.

Model interpretation limitations

The study used SHAP values and Sobol sensitivity analysis to explain model behaviour. These tools are powerful, but they do not prove cause and effect. For example, a feature with high importance in the model may not directly influence performance, especially if it is correlated with another hidden factor. More direct analysis or expert validation is needed to confirm practical meaning.

Risk of data noise or bias

Despite efforts to clean and preprocess the data, real-world industrial signals may contain noise, missing values, or inconsistent measurements. These issues can affect both model accuracy and interpretability. The quality of the results depends heavily on the quality of the input data.

Subjectivity in visual interpretation

Techniques like PCA and scatter plots support visual analysis, but patterns seen in plots can be interpreted differently by different people. This adds some subjectivity to conclusions drawn from visual tools. These insights are most reliable when paired with expert review.

In summary, this study offers a practical and interpretable view of robot performance, but the conclusions should be seen as initial findings. Stronger evidence would come from future studies that include more task types, performance measures, and domain knowledge.

6.4 Suggestions for Future Research

This study explored performance differences in industrial robots using sensor data and machine learning models. While the results were promising, there are several ways future research could improve and expand this work.

Include more performance measures

This study focused on cycle time as the main measure of robot performance. Future work could include other indicators such as energy consumption, task quality, maintenance history, or mechanical load. These additional measures would give a more complete picture of robot behaviour and help explain why performance changes occur.

Use real-world feature names and descriptions

In this study, feature names were anonymised to protect confidentiality, which made it harder to connect model results with real machine actions. Future research using fully labelled data would help link input signals to specific physical components, making explanations easier to understand and more useful for engineers and operators.

Explore more operational settings

Only two operational groups (all_true and simul_false) were studied in detail. Future studies could analyse all eight group combinations or create new groupings based on clustering, task type, or shift schedules. This would allow researchers to explore how different working conditions affect robot performance more deeply.

Apply to other factories or robot types

The current study focused on eight similar gantry robots in a single factory. Future research should test these methods on different robot types, industries,

or use cases. This would show whether the findings can be generalised and help identify robot-specific or task-specific behaviours.

Combine with expert knowledge

While machine learning models can find patterns in data, understanding the causes behind these patterns often requires domain expertise. Future studies could involve regular collaboration with robot operators, engineers, or system designers to better interpret model outputs and validate insights.

Develop early warning tools

The change point detection methods used here could be turned into monitoring tools that automatically alert users when a robot's performance shifts. Future work could develop real-time dashboards or integration with existing factory systems to support early detection of problems and improve maintenance planning.

Use more advanced explainability methods

While SHAP and Sobol analysis provided useful insights, future studies could explore other explainability techniques such as counterfactual explanations, partial dependence plots, or causal inference tools. These methods may offer deeper or more flexible ways to understand model decisions and robot behaviour.

Link behaviour changes to real events

Some performance shifts may be caused by known events such as software updates, mechanical tuning, or environmental changes. Future work could combine data analysis with event logs, maintenance records, or production schedules to better explain why certain change points occur.

7 CONCLUSIONS

7.1 Summary of Main Results

This study explored how industrial robot performance changes over time and under different working conditions using multivariate sensor data and machine learning. The main goal was to detect, explain, and visualise cycle time variability across eight gantry robots operating in a real-world industrial setting.

Change point detection using the PELT algorithm successfully identified performance shifts in each robot's cycle time. These shifts were often linked to possible changes in scheduling, maintenance, or task conditions. Robots did not behave the same way even under the same working configuration, especially in the all_true group where tasks were more complex.

Principal Component Analysis (PCA) helped visualise weekly behaviour patterns and revealed periods where some robots changed their operation noticeably. Robot 52 was the most stable across weeks and provided a good baseline for comparison.

XGBoost models were able to predict cycle time accurately in both operational groups. Visual tools such as scatter plots and density plots showed that the all_true condition produced more consistent and predictable results. In contrast, the simul_false condition showed slightly more variability but still achieved strong performance.

SHAP and Sobol analyses identified the most important input features driving robot cycle time. Feature importance changed depending on the operational setting. For example, tw and tq were key in the all_true condition, while ex and el became more important when simultaneous_operation was turned off.

Together, these results show that robot performance is influenced by both internal signal changes and external working conditions. The combined use of

time-series analysis and explainable machine learning helped uncover hidden patterns and provided a deeper understanding of individual robot behaviour over time.

7.2 Research Contributions

This thesis contributes to the field of industrial analytics by combining data-driven methods with domain understanding to detect and explain robot performance changes over time.

The key contributions are:

- A complete analysis pipeline that uses time-series change point detection to identify shifts in robot cycle time. This allowed the detection of subtle performance changes without requiring manual labels or prior assumptions about failure events.

- A machine learning model that predicts robot cycle time with high accuracy using multivariate sensor signals. The model showed consistent performance across different operational settings, confirming its robustness.

- The integration of explainable AI methods, including SHAP and Sobol analysis, to identify the most important features influencing robot behaviour. These tools helped translate complex model behaviour into understandable insights.

- A comparison of robot performance under different task coordination settings (all_true vs. simul_false), revealing how synchronization affects variability and predictability.

- A visual behavioural analysis using PCA, which allowed robot behaviour to be compared across weeks. This helped identify consistent performers and outliers over time.

These contributions provide practical tools for analysing robot behaviour in manufacturing environments and support future efforts in predictive maintenance, performance benchmarking, and process optimisation.

7.3 Final Reflections

This thesis has been a valuable and personal journey. Working closely with Cimcorp Oy and applying the tools and methods I learned at SAMK gave me a practical understanding of how data can be used to solve real industrial problems.

I have learned that data projects are not only about building models. They are also about meeting practical needs, solving everyday challenges, and delivering useful information that supports better decisions. I also gained important skills such as managing a long-term project, explaining technical ideas clearly, and staying focused through uncertainty.

This experience has helped shape me as a data engineer. I now feel more confident in working with real-world data, communicating insights, and contributing to improvements in industrial systems. I am thankful for the support and feedback I received along the way, and I am looking forward to applying what I have learned in future opportunities.

REFERENCES

Bottin, M., Boschetti, G., & Rosati, G. (2022). Optimizing Cycle Time of Industrial Robotic Tasks with Multiple Feasible Configurations at the Working Points. *Robotics*, *11*(1), 16. https://doi.org/10.3390/robotics11010016

Cho, S., Kim, D., & Park, S. (2024). Simulation Based Cycle Time Prediction for Robot Welding. *2024 Winter Simulation Conference (WSC)*, 2775–2784. https://doi.org/10.1109/WSC63780.2024.10838830

Cimcorp. (2025). *About Us*. Cimcorp Oy. Retrieved from Https://Cimcorp.Com/about-Us/.

Deng, Z., Tian, S., Xu, W., Lu, J., & Hu, Y. (2022). A digital twin-driven dynamic prediction method for robotized production line operational performance. *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, 1222–1227. https://doi.org/10.1109/ICIEA54703.2022.10006259

Farrahi, H., & Mahmood, A. R. (2023). Reducing the Cost of Cycle-Time Tuning for Real-World Policy Optimization. *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN54540.2023.10191867

Guo, D., Zhang, Y., & Chen, X. (2021). *Robot Predictive Maintenance Method Based on Program-Position Cycle* (pp. 78–88). https://doi.org/10.1007/978-3-030-89095-7_8

Hung, A. J., Chen, J., Che, Z., Nilanon, T., Jarc, A., Titus, M., Oh, P. J., Gill, I. S., & Liu, Y. (2018). Utilizing Machine Learning and Automated Performance Metrics to Evaluate Robot-Assisted Radical Prostatectomy Performance and Predict Outcomes. *Journal of Endourology*, *32*(5), 438–444. https://doi.org/10.1089/end.2018.0035

Komenda, T., Brandstötter, M., & Schlund, S. (2021). A comparison of and critical review on cycle time estimation methods for human-robot work systems. *Procedia CIRP*, *104*, 1119–1124. https://doi.org/10.1016/j.procir.2021.11.188

Kostov, B., & Hristov, V. (2023). Optimizing Cycle Time of Industrial Robot for Loading Molding Machine: A Comprehensive Analysis and Optimization Approach. *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 01–05. https://doi.org/10.1109/HORA58378.2023.10156771

LI, M., SUN, X., LIANG, G., SHEN, Y., ZHANG, Q., & FENG, Y. (2021). Bayesian heterogeneous assembly time modeling for robotic

performance prediction. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, *15*(1), JAMDSM0007–JAMDSM0007. https://doi.org/10.1299/jamdsm.2021jamdsm0007

Luo, Z., Xiao, J., Liu, S., Wang, M., Zhao, W., & Liu, H. (2024). A dynamic parameter identification method for the 5-DOF hybrid robot based on sensitivity analysis. *Industrial Robot: The International Journal of Robotics Research and Application*, *51*(2), 340–357. https://doi.org/10.1108/IR-08-2023-0178

Rao, V. V, Ratan, K. G., Bewoor, M., & Benni, R. (2024). Leveraging Boosting Algorithms for Accurate Battery Life Prediction in Autonomous Robots. *2024 IEEE Conference on Engineering Informatics (ICEI)*, 1–9. https://doi.org/10.1109/ICEI64305.2024.10912333

Saviano, G., Villani, A., & Prattichizzo, D. (2024). Mapping Music onto Robot Joints for Autonomous Choreographies: PCA-Based Approach. *2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI)*, 232–237. https://doi.org/10.1109/RTSI61910.2024.10761717

Spensieri, D., Åblad, E., Bohlin, R., Carlson, J. S., & Söderberg, R. (2021). Modeling and optimization of implementation aspects in industrial robot coordination. *Robotics and Computer-Integrated Manufacturing*, *69*, 102097. https://doi.org/10.1016/j.rcim.2020.102097

Stade, D., Spoor, J. M., Manns, M., & Ovtcharova, J. (2025). Process time distribution simulation in robotic assembly line balancing. *International Journal of Production Research*, *63*(10), 3467–3484. https://doi.org/10.1080/00207543.2024.2416570

Stuhlenmiller, F., Weyand, S., Jungblut, J., Schebek, L., Clever, D., & Rinderknecht, S. (2021). Impact of Cycle Time and Payload of an Industrial Robot on Resource Efficiency. *Robotics*, *10*(1), 33. https://doi.org/10.3390/robotics10010033

Touzani, H., Hadj-Abdelkader, H., Seguy, N., & Bouchafa, S. (2021). Multi-Robot Task Sequencing & Automatic Path Planning for Cycle Time Optimization: Application for Car Production Line. *IEEE Robotics and Automation Letters*, *6*(2), 1335–1342. https://doi.org/10.1109/LRA.2021.3057011

Trost, P., & Eder, M. (2024). An analytical performance approach for RCS/RS with one robot serving multiple stack heights under a one-path relocation strategy. *Scientific Reports*, *14*(1), 3593. https://doi.org/10.1038/s41598-024-53884-6

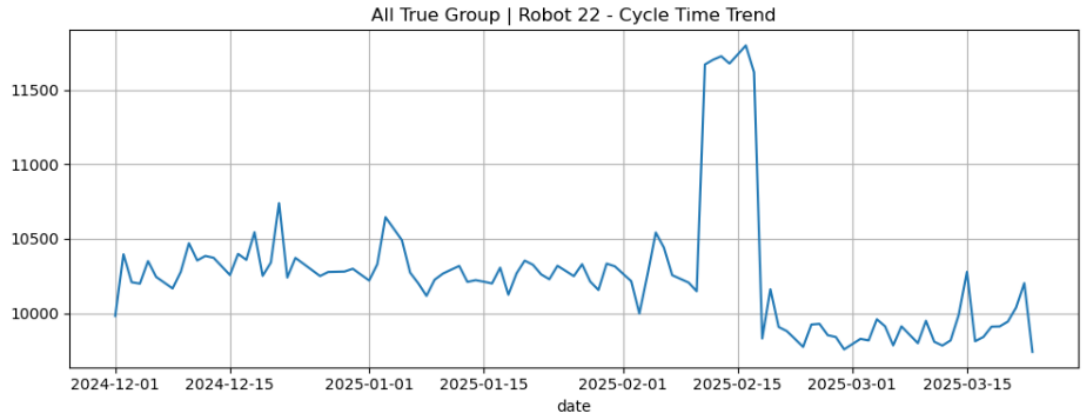
Truong, C., Oudre, L., & Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, *167*, 107299. https://doi.org/10.1016/j.sigpro.2019.107299

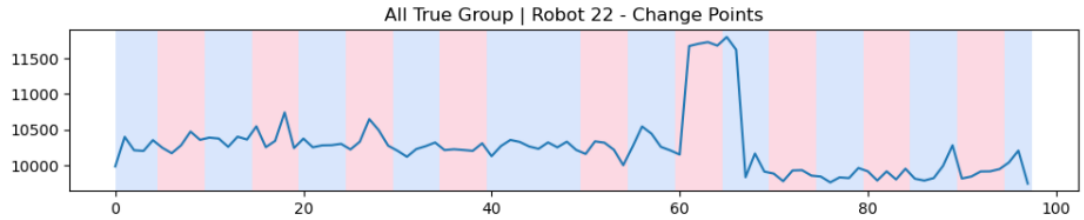
Wu, K., Lu, Y., Huang, R., Kuhlenkötter, B., & Li, W. (2024). A Time-Series Data-Driven Method for Milling Force Prediction of Robotic Machining. *IEEE Transactions on Instrumentation and Measurement*, *73*, 1–12. https://doi.org/10.1109/TIM.2024.3376018

APPENDICES

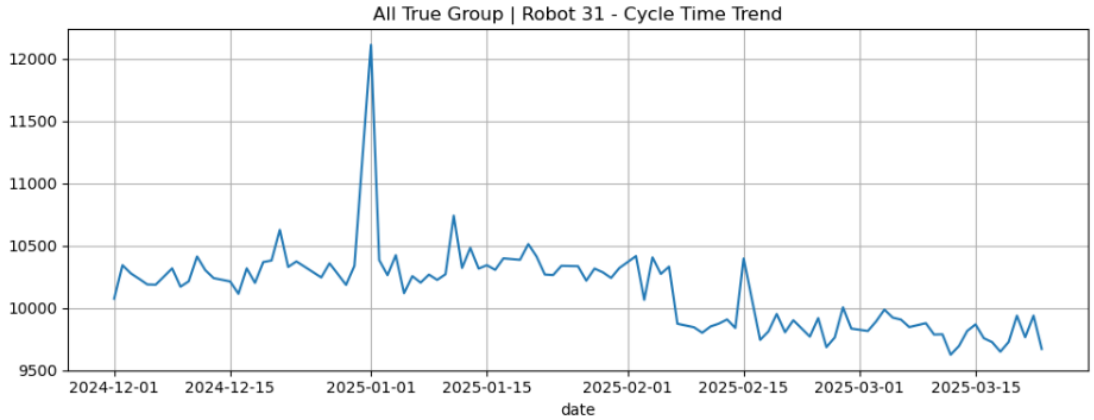
Appendix 1 – Change Point Detection: all_true & simul_false

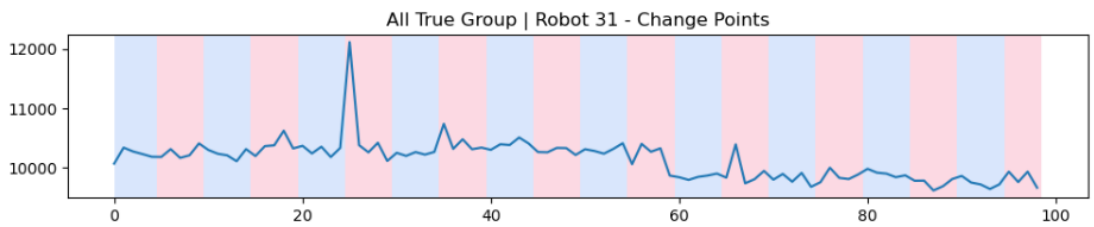
all_true group (Robots 22–52)



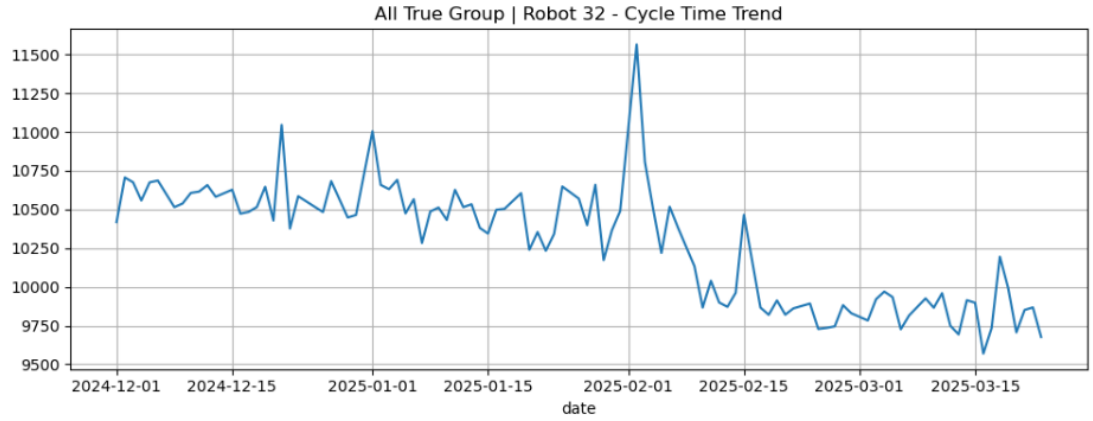


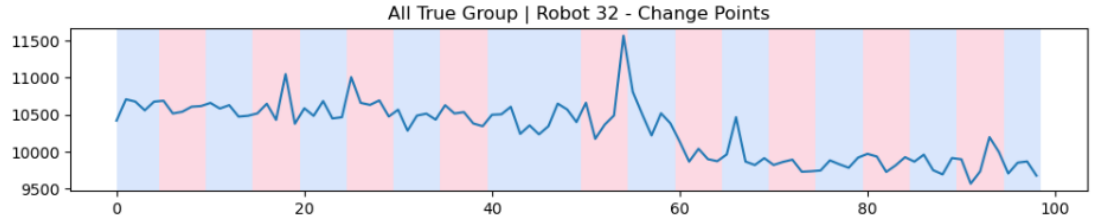

```
Top features for robot 22:
 ta    1.000000
tw     0.778838
gg    -0.297539
el    -0.297539
ei    -0.290148
gp    -0.290031
Name: ta, dtype: float64
```



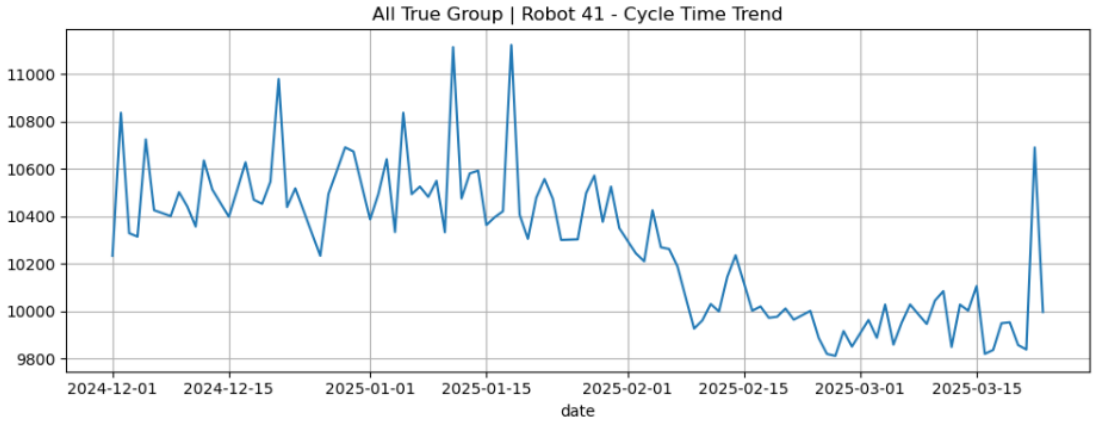


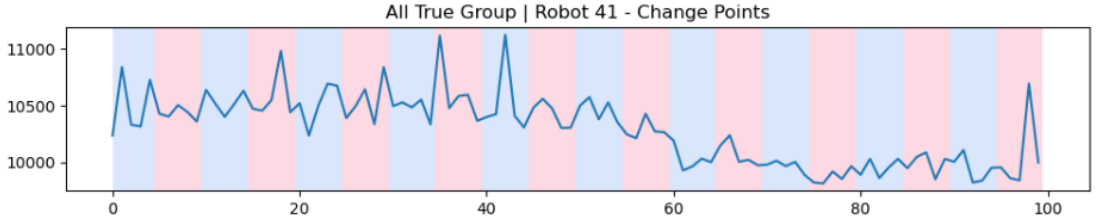

```
Top features for robot 31:
 ta    1.000000
tw     0.760173
td     0.334274
el    -0.331232
gg    -0.331230
ei    -0.327260
Name: ta, dtype: float64
```



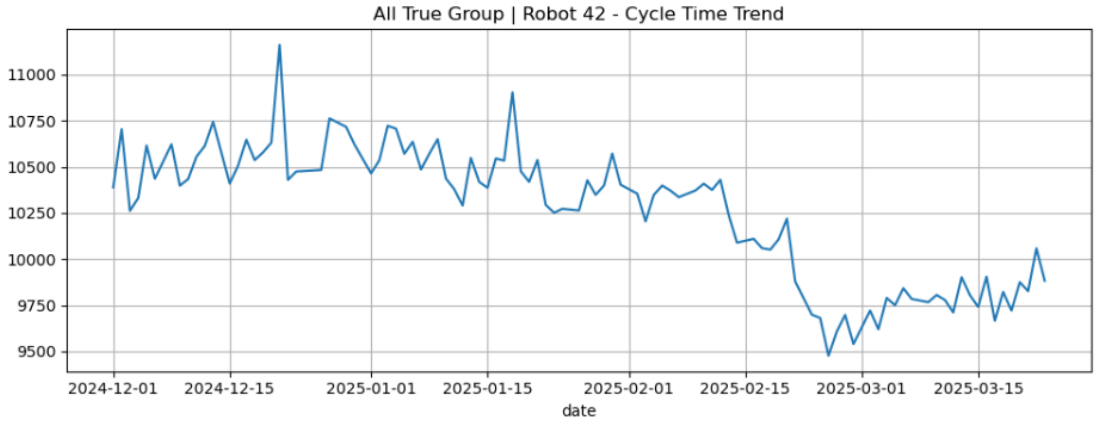


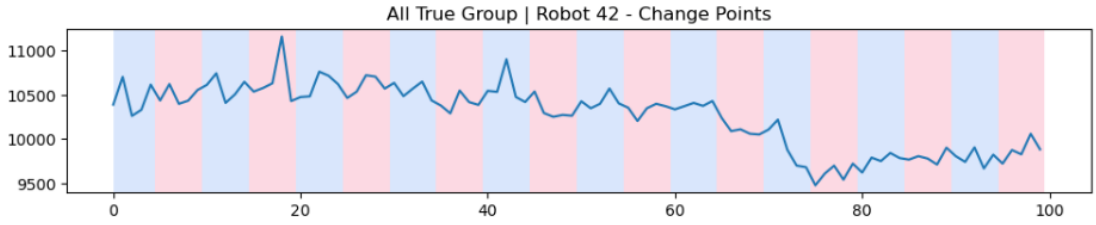

```
Top features for robot 32:
 ta    1.000000
tw     0.744129
ts     0.363831
td     0.358468
gg    -0.333288
el    -0.333288
Name: ta, dtype: float64
```



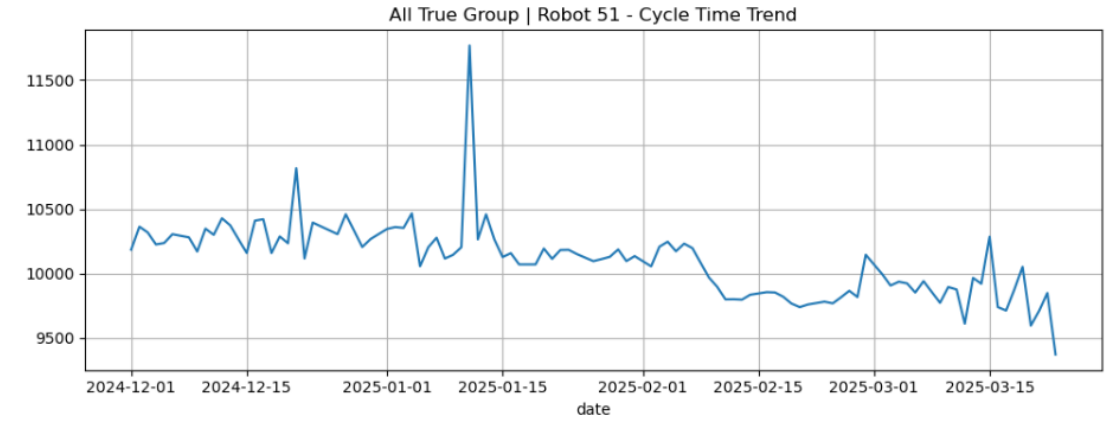


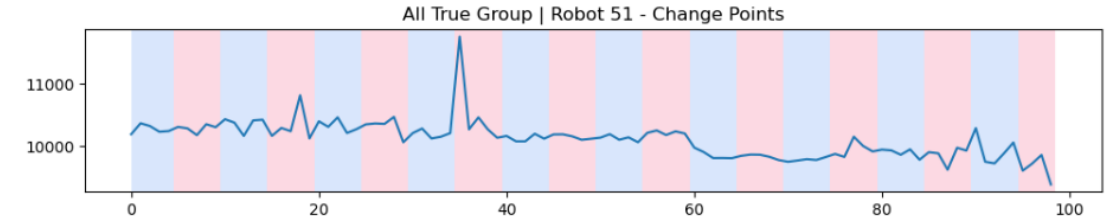

```
Top features for robot 41:
 ta    1.000000
tw     0.758828
td     0.416099
ts     0.371728
gg    -0.321193
el    -0.321193
Name: ta, dtype: float64
```



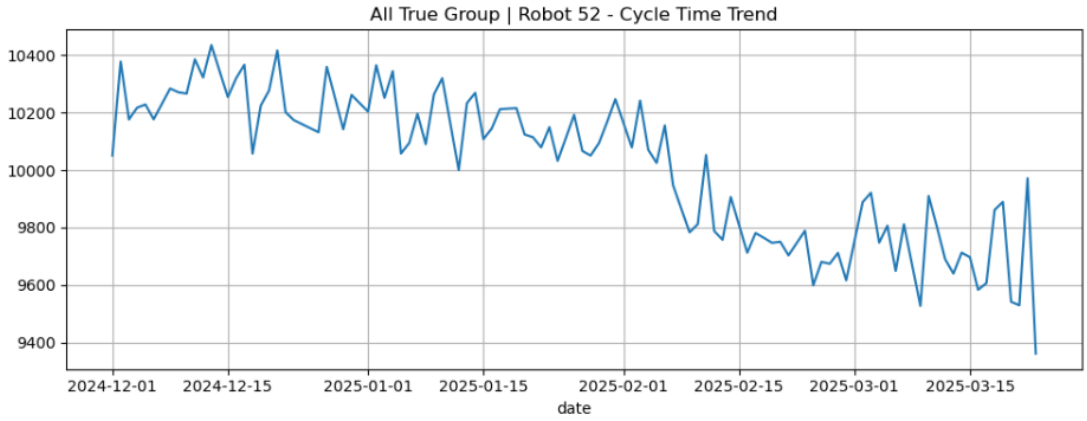


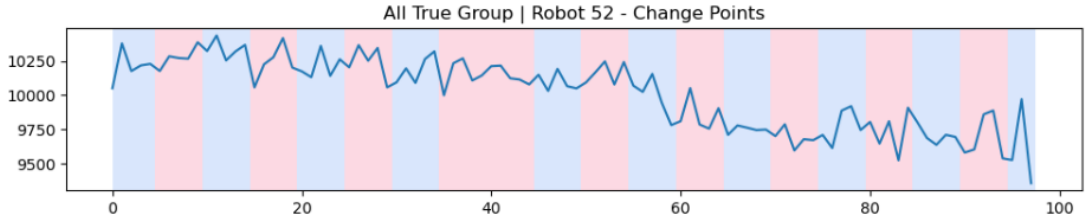

```
Top features for robot 42:
 ta    1.000000
tw     0.733300
td     0.342537
ts     0.336974
el    -0.324172
gg    -0.324172
Name: ta, dtype: float64
```





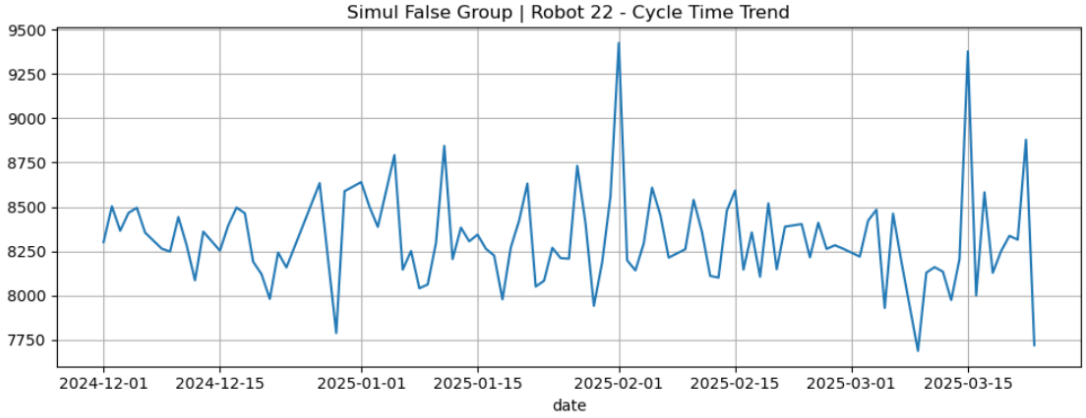

```
Top features for robot 51:
 ta    1.000000
tw     0.740394
td     0.356891
el    -0.336521
gg    -0.336521
ts     0.335442
Name: ta, dtype: float64
```

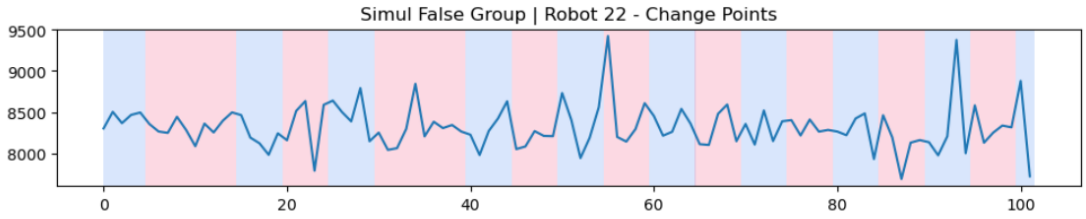




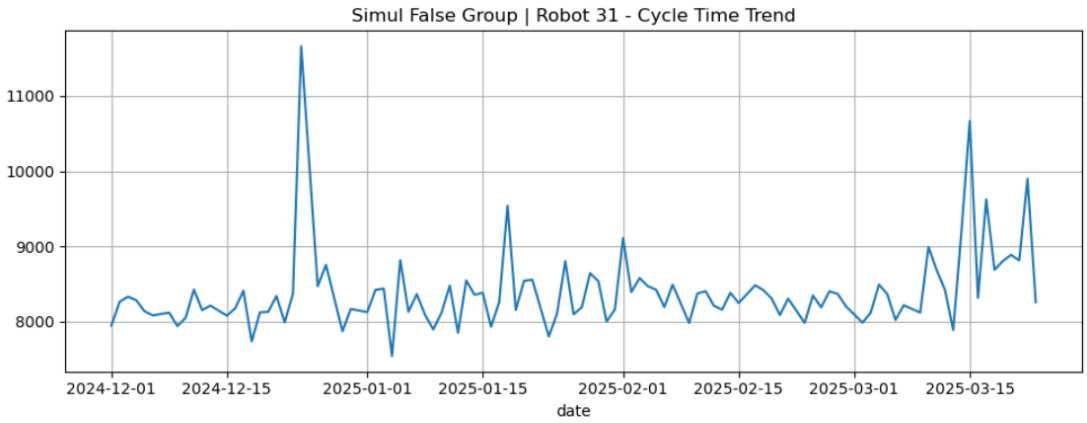

```
Top features for robot 52:
 ta    1.000000
tw     0.737431
el    -0.338708
gg    -0.338708
gp    -0.332991
ei    -0.332634
Name: ta, dtype: float64
```

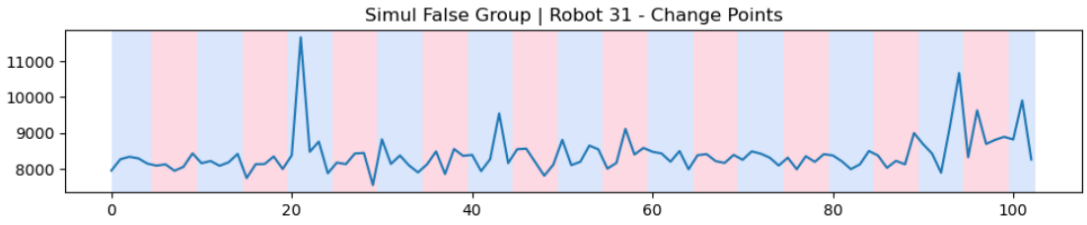
simul_false group (Robots 22–52)



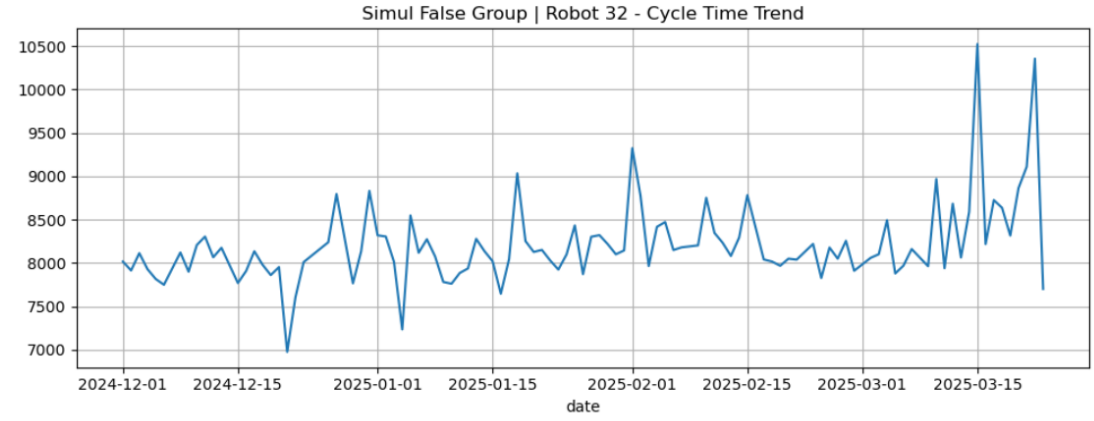


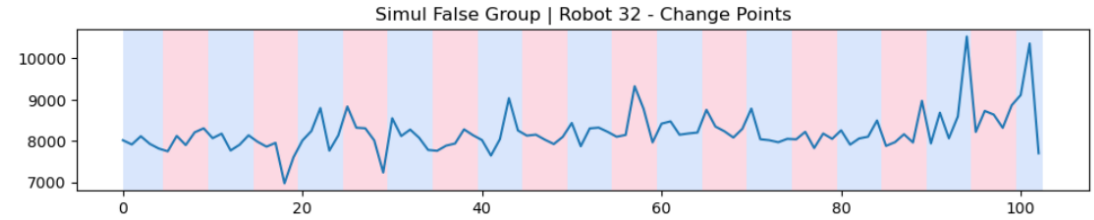

```
Top features for robot 22:
 ta    1.000000
ex     0.570858
ev     0.526072
ed     0.409857
ts     0.388167
em     0.358902
Name: ta, dtype: float64
```



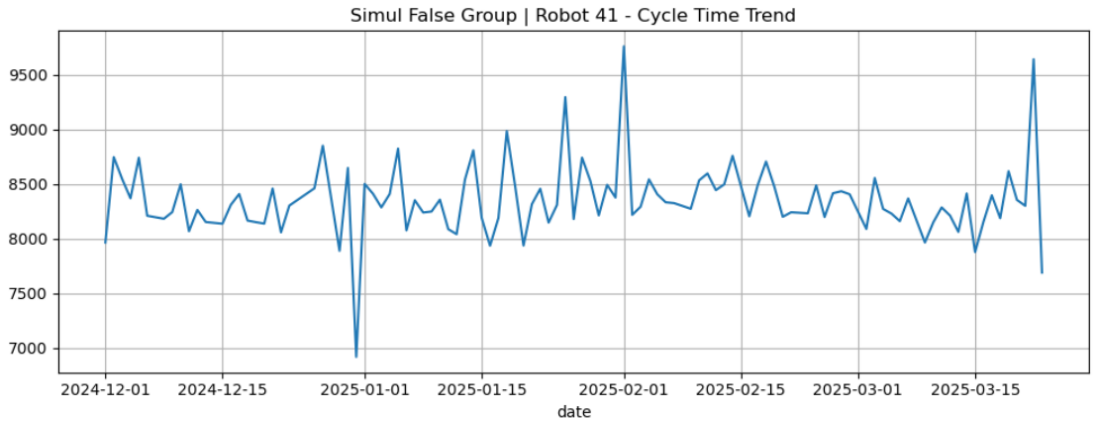


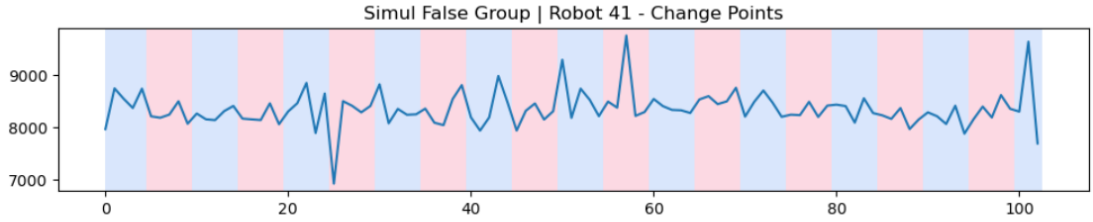

```
Top features for robot 31:
 ta    1.000000
ex     0.737586
ev     0.617692
em     0.493458
es     0.485573
en    -0.409116
Name: ta, dtype: float64
```



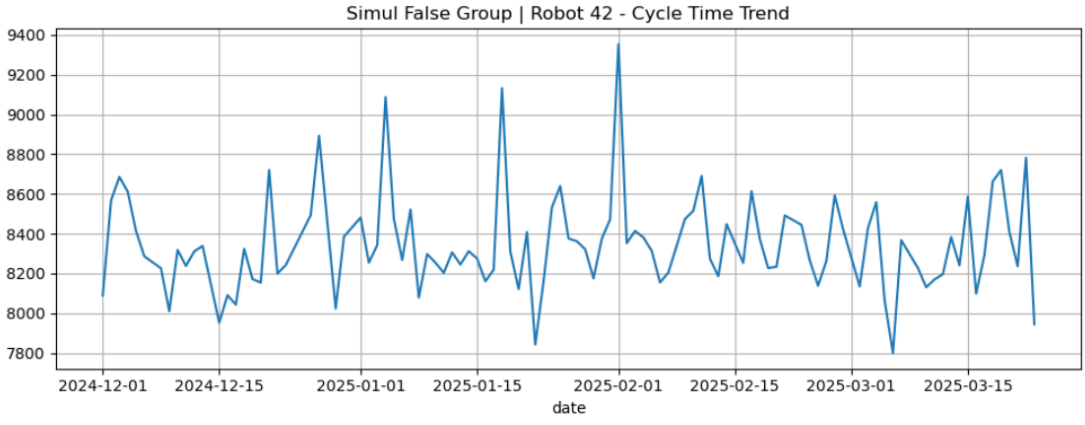


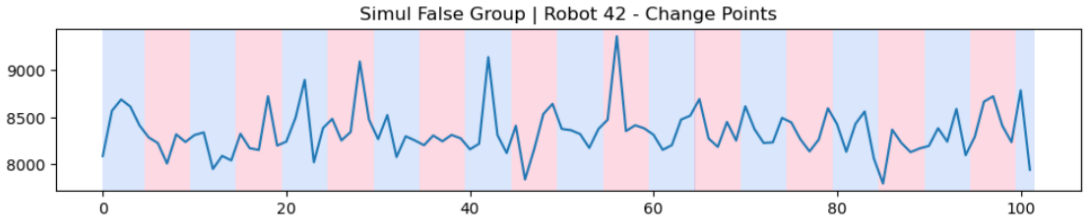

```
Top features for robot 32:
 ta    1.000000
ex     0.670294
ev     0.587090
em     0.498569
es     0.446650
en    -0.382991
Name: ta, dtype: float64
```



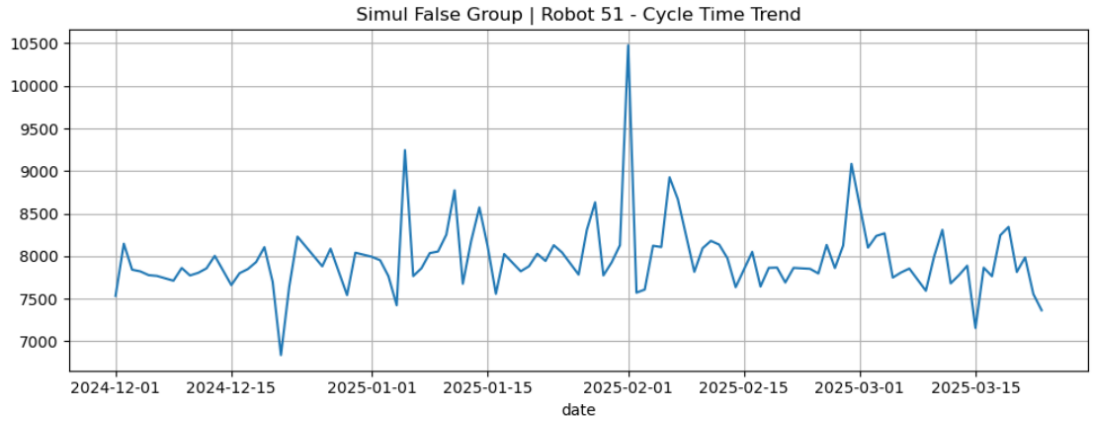


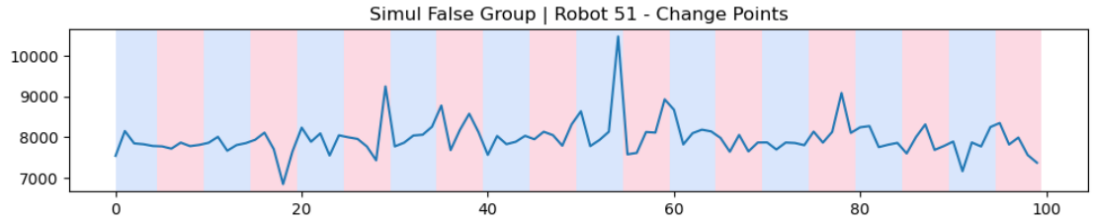

```
Top features for robot 41:
 ta    1.000000
ex     0.726165
ev     0.643715
es     0.544430
em     0.522580
en    -0.383628
Name: ta, dtype: float64
```



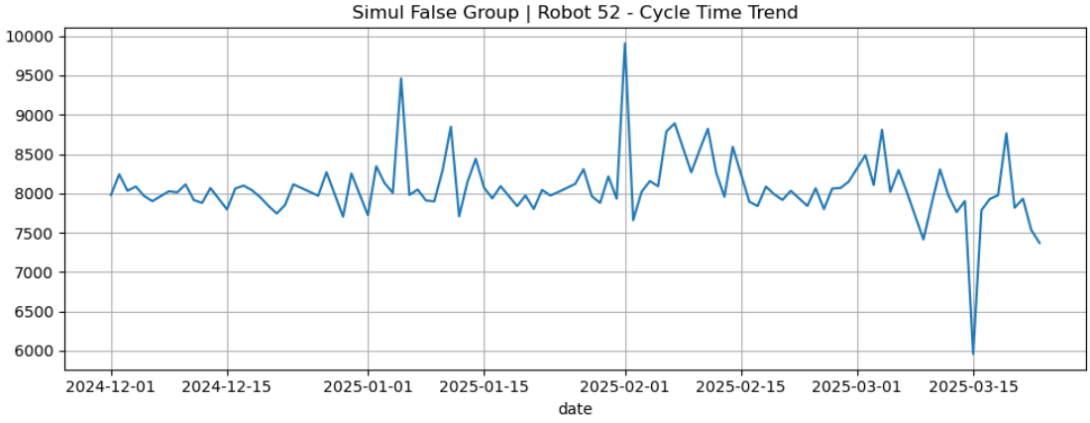


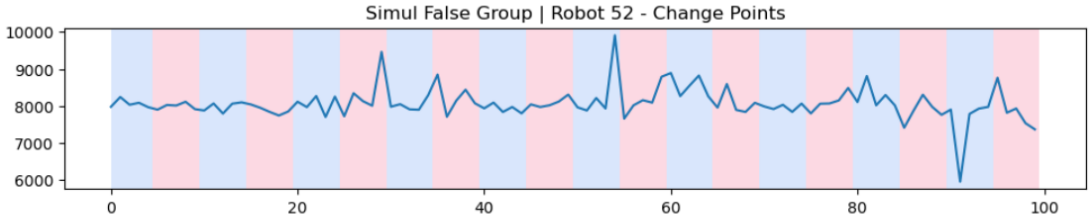

```
Top features for robot 42:
 ta    1.000000
ex     0.659110
ev     0.593786
em     0.479705
es     0.476948
ts     0.395997
Name: ta, dtype: float64
```






```
Top features for robot 51:
 ta    1.000000
ex     0.766616
ev     0.688753
em     0.621082
es     0.611004
eq     0.427292
Name: ta, dtype: float64
```






```
Top features for robot 52:
 ta    1.000000
ex     0.676272
ev     0.621594
em     0.531084
es     0.524516
ts     0.446416
Name: ta, dtype: float64
```

Appendix 2 – Code Listings for Preprocessing, Modelling, and Analysis

This appendix presents key Python code snippets used during the analysis, covering data preparation, modelling, feature importance, and sensitivity analysis.

2.1 Data Preprocessing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ruptures as rpt

# Load anonymized data
data = pd.read_csv('robot_data.csv')
data['cmd_time'] = pd.to_datetime(data['cmd_time'])

# Drop duplicates
duplicates = []
for col1 in data.columns:
    for col2 in data.columns:
        if col1 != col2 and
data[col1].equals(data[col2]):
            duplicates.append((col1, col2))

# Manual investigation and cleanup
data['ghi'] = data['gi'] + 40
data.drop(columns=['go', 'gj', 'ek', 'gh', 'gi'],
inplace=True)

# Remove invalid rows
data = data[(data['tm'] <= 0) & (data['tp'] <= 0)]
data = data[data['ta'] <= 20000]
data['ev_positive'] = data['ev'] > 0
data = data.drop(columns=['tb'], errors='ignore')

# Grouping
groups = {}
for pf in [True, False]:
    for so in [True, False]:
        for ev_pos in [True, False]:
            name = f"group floor_pick {pf}
simultaneous_operation {so} ev_positive {ev_pos}"
            group = data[
                (data['floor_pick'] == pf) &
                (data['simultaneous_operation'] == so) &
                (data['ev_positive'] == ev_pos)
```

```
            ]
            groups[name] = group

all_true = groups["group floor_pick True
simultaneous_operation True ev_positive True"]
simul_false = groups["group floor_pick True
simultaneous_operation False ev_positive True"]
```

2.2 Model Training

```
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_squared_error
import random

# Select features
def select_features(df):
    df = df.copy()
    df = df.drop(columns=['cmd_time'], errors='ignore')
    df[['ex', 'ey']] = df[['ex', 'ey']].abs()
    features =
df.select_dtypes(include='number').columns.difference(['t
a','deviceid']).tolist()
    X = df[features].fillna(0)
    y = df['ta'].values
    ids = df['deviceid']
    return X, y, ids

# Split data
from sklearn.model_selection import train_test_split

def five_day_split(df, train_even_block=True):
    df = df.copy()
    df['cmd_time'] = pd.to_datetime(df['cmd_time'])
    train_mask = pd.Series(False, index=df.index)
    test_mask  = pd.Series(False, index=df.index)
    for rid, g in df.groupby('deviceid'):
        g = g.sort_values('cmd_time')
        first_midnight =
g['cmd_time'].iloc[0].normalize()
        block_no = ((g['cmd_time'] -
first_midnight).dt.days // 5).astype(int)
        if train_even_block:
            in_train = block_no % 2 == 0
        else:
            in_train = block_no % 2 == 1
        train_mask.loc[g.index] = in_train
        test_mask.loc[g.index]  = ~in_train
    return train_mask, test_mask
```

```
# Train model
def train_with_tuning(X_train, y_train):
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [3, 5, 7],
        'learning_rate': [0.01, 0.05, 0.1],
        'subsample': [0.7, 0.8, 1.0],
        'colsample_bytree': [0.7, 0.8, 1.0]
    }
    model = XGBRegressor(random_state=42, n_jobs=-1)
    search = RandomizedSearchCV(model, param_grid,
n_iter=10, cv=3, scoring='neg_root_mean_squared_error')
    search.fit(X_train, y_train)
    return search.best_estimator_
```

2.3 Change Point Detection

```
def analyze_robots(group_df, group_name):
    for robot_id in group_df['deviceid'].unique():
        robot_data = group_df[group_df['deviceid'] ==
robot_id].copy()
        robot_data['date'] =
pd.to_datetime(robot_data['cmd_time']).dt.date
        daily_avg =
robot_data.groupby('date')['ta'].mean()
        if len(daily_avg) < 50:
            continue
        signal = daily_avg.values
        algo = rpt.Pelt(model="l2").fit(signal)
        change_points = algo.predict(pen=10)
        rpt.display(signal, change_points)
        plt.title(f"{group_name} | Robot {robot_id} -
Change Points")
        plt.show()
```

2.4 SHAP Analysis

```
import shap

def explain_with_shap(model, X_test, label):
    sample = X_test.sample(n=100, random_state=42)
    explainer = shap.Explainer(model)
    shap_values = explainer(sample)
    print(f"\n{label} - SHAP Summary")
    shap.summary_plot(shap_values, sample)
    shap.summary_plot(shap_values, sample,
plot_type="bar")
```

2.5 PCA and Classification

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

# PCA visualization
for week in data['week'].unique():
    week_df = data[data['week'] == week]
    if week_df['deviceid'].nunique() < 2:
        continue
    features = week_df.drop(columns=[
        'ta', 'deviceid', 'week', 'floor_pick',
        'simultaneous_operation', 'ev_positive',
'cmd_time', 'date'
    ], errors='ignore')
    scaled = StandardScaler().fit_transform(features)
    pca_result =
PCA(n_components=2).fit_transform(scaled)
    week_df['pca1'], week_df['pca2'] = pca_result[:, 0],
pca_result[:, 1]

# Classification
features = data.drop(columns=[
   "ta", "deviceid", "week", "floor_pick",
   "simultaneous_operation", "ev_positive",
   "cmd_time", "date"
], errors='ignore')
features = features.select_dtypes(include='number')
features_scaled =
StandardScaler().fit_transform(features)
X = features_scaled
y = data["deviceid"]
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_encoded = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(
   X, y_encoded, stratify=y_encoded, test_size=0.3,
random_state=42
)
clf = XGBClassifier(use_label_encoder=False,
eval_metric="mlogloss", random_state=42)
clf.fit(X_train, y_train)
print(f"Robot ID Classification Accuracy (XGBoost):
{accuracy_score(y_test, clf.predict(X_test)) *
100:.2f}%")
```

2.6 Robot Efficiency Ranking

```
from xgboost import XGBRegressor

def add_ta_ratio(group_df, robot_id):
    robot_df = group_df[group_df['deviceid'] ==
robot_id].copy()
    robot_df['cmd_time'] =
pd.to_datetime(robot_df['cmd_time'])
    robot_df.sort_values('cmd_time', inplace=True)
    for col in ['ex', 'ey']:
        if col in robot_df.columns:
            robot_df[col] = robot_df[col].abs()
    X = robot_df.drop(columns=['deviceid', 'cmd_time',
'ta', 'floor_pick',
                               'simultaneous_operation',
'ev_positive'], errors='ignore')
    X = X.select_dtypes(include='number').fillna(0)
    y = robot_df['ta'].values
    model = XGBRegressor(n_estimators=100,
random_state=42, n_jobs=-1)
    model.fit(X, y)
    y_pred = model.predict(X)
    y_pred = np.where(y_pred == 0, np.nan, y_pred)
    robot_df['ta_estimated'] = y_pred
    robot_df['ta_ratio'] = robot_df['ta'] / y_pred
    return robot_df

def process_group(group_df, group_name):
    frames = []
    for rid in group_df['deviceid'].unique():
        df = add_ta_ratio(group_df, rid)
        df['group'] = group_name
        frames.append(df)
    return pd.concat(frames, ignore_index=True)

all_true_with_ratio = process_group(all_true, 'All True')
simul_false_with_ratio = process_group(simul_false,
'Simul False')

def rank_robot_efficiency(all_true_df, simul_false_df):
    combined = pd.concat([all_true_df, simul_false_df],
ignore_index=True)
    summary = (
        combined.groupby(['deviceid'])
        .agg(
            avg_ta=('ta', 'mean'),
            std_ta=('ta', 'std'),
            mean_ta_ratio=('ta_ratio', 'mean'),
```

```
            count=('ta', 'count')
        )
        .reset_index()
    )
    summary['rank_by_avg_ta'] =
summary['avg_ta'].rank(method='min')
    summary['rank_by_mean_ta_ratio'] =
summary['mean_ta_ratio'].rank(method='min')
    summary['efficiency_rank'] = (
        summary['rank_by_avg_ta'] +
summary['rank_by_mean_ta_ratio']
    ) / 2
    return
summary.sort_values(by='efficiency_rank').reset_index(dro
p=True)

robot_ranking =
rank_robot_efficiency(all_true_with_ratio,
simul_false_with_ratio)
```