Universiteit Leiden

Master Computer Science

Pattern Mining Enabled Scanner Event-based Quantitative Analysis of Lithography Scanner Productivity Performance

Name:	Chang Liu
Student ID:	s2560712
Date:	02/08/2022
Specialisation:	Computer Science: data science
1st supervisor:	Matthijs van Leeuwen
2nd supervisor:	Wojtek Kowalczyk

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Currently, there is a global shortage of chips. To increase the chip production in the same amount of time, we need to improve the productivity of the lithography scanner machine. Time and productivity are negatively correlated. To improve productivity, we want to shorten the chip production time, starting by knowing what lead to this time value.

In this project, we focus on a transition time range in the chip production of the lithography scanner, called lot overhead(LOH). This transition stage is composed of many events happening in chronological order. We carry out a quantitative analysis using coalition game theory, machine learning, and pattern mining methods. This enables us to identify event codes or patterns that can indicate productivity impact and quantify their impact on LOH.

With several considerations and expectations, we adopt Shapley Value from coalition game theory as the impact's quantification measurement and call it the time influence. The machine learning prediction models selected to support Shapley Value calculation are Random Forest and Gradient Boosting. By doing so, we can find which events and patterns are playing a big role in the LOH of a certain lot, or generally of all lots of a machine within a certain period of time.

The pattern here is composed of a sequence of events. To find patterns with large time influence, we propose a three-stage method. In the first two stages, we find candidates with a large potential to have a big impact, then we carry out quantification analysis to get the final top patterns with large time influence. We conduct experiments on four sequential pattern mining algorithms to find pattern candidates. Besides this, we also conduct a pre-experiment on finding itemset patterns as a test for time consumption and the underlying regression model performance.

We compare the results with domain experts' experience and see if the events and patterns we find make sense. We find the method can identify some events with the potential to play a big role in reality based on domain experts' experience, while some others of them are unknown or are regarded as unimportant event codes by experts and need to be checked further. The patterns we find tend to be reasonable and bring us useful insights.

Acknowledgement

Coming to the Netherlands to study is like starting a different chapter of the journey in my life. This chapter is full of memorable moments. This also applies to this master thesis internship project, which is definitely a voyage full of happiness, surprise, and growth. I want to thank everyone who helped, supported, and cared for me on this journey.

I have always wanted to apply the knowledge I learned during my academics to solve real-world problems, or simply to get some echo from the world. This is the reason I decided to finish my graduation thesis project in the industry. I thank LIACS in Leiden University and ASML to provide me with the opportunity, the good platform, and the strong support to do the thing I really wanted to.

I would like to express my gratitude to my thesis supervisor Matthijs van Leeuwen, not only because of his guidance on the research direction in general but also for his way of working and thinking, which also influence me positively. I also thank my second supervisor Wojtek Kowalczyk to review my thesis. I would like to thank my daily supervisor Lincen Yang, we have frequent discussions and I am surprised that he can always understand my questions well and fast, sometimes even before I show the complete materials I prepared to back up my questions, and his help is always to the point.

I would also like to express my gratitude to my mentors in the company, Hasini, and Hrishi, who irrespective of their busy schedules took out time to have helpful discussions with me. I would also like to thank my coach Ignacio from Development & Engineering department. He is always warm-hearted to help, both technically and mentally, giving a lot of encouragement to new joiners. I would also like to thank Jos from D & E too, he provided me with many different perspectives and insights from D & E side.

This project was finished in the Customer Support department, besides my mentors, I'd also like to thank many people there. I really want to thank Fred, an experienced domain expert who provided many insightful ideas and opinions. Also my Productivity domain background knowledge trainers and people who helped me with the arrangement and the review of plans and results: Sicco, Lin, Tingting, Nikesh, Marteen, Nihan, Ranjani, Tom, and Dennis R.

I'd also like to thank all my family members, and my friends Bonan, Xiaofeng, Xun, and Shuwen, who support me going through some tough times during this project.

In this internship thesis project journey, I grow a lot, both academically and mentally. These will definitely continue to significantly influence me in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

ASML is a company specializing in producing photolithography systems, which gives the world's chip-makers the power to mass produce microchips that are ubiquitous in today's world. ASML's lithography scanner systems (such as EUV and DUV) fabricate the circuitry patterns on silicon wafers covered by a photoresist. This process is called Expose, which is a critical step in the chip manufacturing process. The patterns link together as a single integrated circuit that can offer computing or memory function [4]. The processed wafers are then cut into a pre-defined size to create micro chips [8].

The chip is an essential part of all kinds of modern devices. There has been a global chip shortage since 2020 due to the snow-ball effect of the COVID-19 pandemic, affecting more than 169 industries [3]. There are several ways to relieve this problem on the ASML side, like delivering new lithography systems or improving the availability and efficiency of the current systems [4]. With the latter, we come to the topic of productivity improvement to increase chip production (wafer per hour).

Customers always customize their scanner usage to print different custom chip patterns accurately. These customizations often impact the throughput capability(productivity) of the scanner. Often, the scanner cannot reach its maximum throughput capability. Typically, productivity domain experts analyze customer scanner productivity performance data using internal data analysis tools. However, due to the large data volume and complexity of the process, it is challenging and labor-intensive to identify all possible productivity issues. Thus in this project, we experiment with taking the advantage of data science, to make the analysis more efficient, and with the expectation to discover new insights.

In chip production, many events are happening in chronological order and logged in the system. An event can be of the following types: normal operation, warning, issue, or error(but we don't know it in advance if without abundant domain knowledge). Each of the events gets a corresponding code, named as *Event Code*. Each event code reports the content of a certain event and the event's occurrence time inside a scanner. They are used for diagnostics purposes.

In mass production the lithography scanner produces the chips in batches, each batch is named a "lot" in industry. We focus on a specific time range in chip production, called lot overhead(LOH). On a high level, LOH is the duration between the processing of two consecutive lots in the scanner. We analyze these event codes in LOH duration and aim at identifying the event codes or patterns(composed by a sub-sequence of events) that can indicate productivity impact and quantify their impact on LOH duration.

We adopt methods from coalition game theory [28], machine learning, and pattern mining to solve the problem.

Due to the consideration of the interactive effect among events and with several expectations from domain experts, we adopt Shapley Value[29] from coalition game theory as the quantification measurement of impact and name it the time influence. Shapley Value breakdowns each lot's total LOH time based on the contribution of the events and/or patterns in the LOH.

In addition to the analysis of the single event's impact on the LOH, we also want to find patterns with a big impact on LOH time. To the best of our knowledge, currently, there is no method for mining sequential patterns with a numerical target. So we propose a three-stage method. In the first two stages, we find pattern candidates. Firstly, we use sequential pattern mining algorithms [49] as the tool to find pattern candidates. Then we do a selection and keep only the best candidates. After this, we send these pattern candidates together with all single events into the quantification method mentioned above to evaluate their time influence, thus being able to find events and patterns with a big impact. We conduct experiments on four sequential pattern mining algorithms [49] to find pattern candidates. Besides this, we also conduct a pre-experiment on finding itemset patterns as a test for time consumption and underlying regression model performance. Meanwhile, itemset patterns themselves could also be interesting.

After all these, we invite domain experts to review the result of pattern mining and exploration of events/patterns with a big impact on the LOH and see if the results are in line with their empirical experience. We find the method can identify some events with the potential to play a big role in reality based on domain experts' experience, but many of them are currently unknown by experts and need to be checked further. The patterns we find tend to be reasonable and bring us useful insights.

The rest of the thesis is organized as follows. In chapter 2 we give a general background introduction of the company, department, and data scope. In chapter 3 we list all concepts and mathematical notations we used, together with their brief explanations. In chapter 4, we give an overview of the datasets we are dealing with. In chapter 5 we give a formal statement of the problem we are going to solve and the motivations behind it. In chapter 6 we mention the previous work that is related to our project. In chapter 7, we describe the methods and algorithms we applied to solve the problem in detail. In chapter 8, we carry out the experiments, where the set-up is listed and the results are presented and discussed. In chapter 9 we conclude the whole project, and in chapter 10 the potential future work and new research directions are given.

Chapter 2

Background

The process that lithography scanners use a light source to print the microcircuits on the wafers is called *Expose*. A revolutionary invention of ASML on lithography scanner is TWINSCAN, a unique lithography system platform with two complete wafer table modules. When the wafer on table one is being exposed, another wafer is loaded on table two and then aligned and mapped. The tables then swap positions so that the wafer on table two is exposed while the wafer on table one is unloaded [4]. This is called *Swapping*.

A *wafer* itself is a circular-shaped disc made of silicon on which the printing of the circuit takes place. A collection of wafers supplied to the machine for processing at a given period of time is known as a *lot* (sometimes people also call it a *batch*). Each lot consists of a certain amount of wafers, which is usually 25 but could be less than 25. The *mask* or *reticle* contains the structure information we want to transfer to the wafer.

To achieve better mass production, we aim to increase the productivity of the scanner. The Productivity team divides the time in wafer production into 3 parts.

- Wafer expose time (WET)
- Wafer swap time (WST)
- Lot overhead (LOH)

Wafer processing time WPT is equal to WET + WST. We mainly aim at analyzing the lot overhead (LOH) time in this project, while the analyzing method can be easily generated for WET and WST analysis. The relative relationship between LOH and the other two is shown in Figure 2.1

The productivity team's goal is to increase the lithography scanner's productivity (while keeping good quality), letting the process be streamed and standardized as much as possible. However, the machine does not always run as expected, errors may occur, customers choose different machine configuration settings to optimize the pattern they want to print, etc. These could all influence the LOH time. Also, whenever there is an update in lithography, the new procedure is implemented, then the LOH could change accompanied by the updating. Under these circumstances, experts in ASML want to inspect and diagnose the machine's running conditions efficiently.

Previously domain experts are doing the analysis of the events (codes) using internal tools. The analysis highly relies on delicately trained domain knowledge

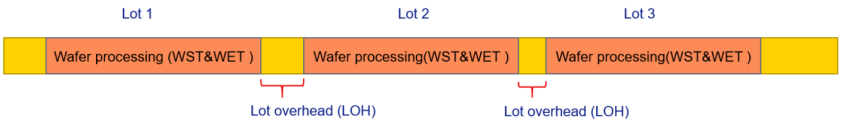


Figure 2.1: Three parts of productivity
WST and WET make up the wafer processing time(areas in orange), and we emphasize the transition period called LOH here(areas in yellow).

or even several years of experience accumulation. Even though the number of event codes is too many(at least more than 12k) and it is difficult for experts to learn about all the event codes, especially since these codes could come from completely different domains and may keep increasing. Additionally, one may check single events, but with the increase in data volume, it may be difficult to easily explore the relationship between them and extract patterns from it. Thus in this project, we are taking the advantage of data science techniques, to deal with a huge amount of information and overcome the domain knowledge gap.

Chapter 3

Basic concept and Notation

In this chapter, we list the definition of several concepts and mathematical notations we use to better define and solve our problem.

Firstly, we denote the collection set of all possible unique event codes e_i as E, and the number of unique event codes is $|E|$, i.e., $E = \{e_1, e_2, ..., e_{|E|}\}$.

An event (code) sequence s is an ordered list of event codes(ordered in their occurrence time), i.e., $s = [a_1, a_2, ...a_n]$, where n is the length of s, and a_i takes the value from E. Each sequence s corresponds to a certain lot. Further, given two event code sequences s and s' that $s = [a_1, a_2, ...a_n]$, $s' = [b_1, b_2, ...b_m]$. We denote $s' \sqsubseteq s$ if s' is contained in s, i.e., $s' \subseteq s$ if and only if there exist integers $i_1, ...i_n$ that $1 \leq i_1 < i_2 < ... < i_n \leq n$ s.t $b_1 = a_{i_1}, b_2 = a_{i_2}, ...b_n = a_{i_n}$. We call s' is a sub-sequence of s. For example, $[a, d]$ is a sub-sequence of $[a, b, c, d]$.

In the later pattern mining section, a sequential pattern candidate is firstly a sub-sequence of an event (code) sequence and then satisfies other requirements. We denote the sub-sequence pattern as p.

An event (code) itemset I is a set of items such that $I \subseteq E$. Each I corresponds to a certain lot. For a sequence $[a, b, c, c]$, the corresponding itemset is $\{a, b, c\}$. Similarly as for sequence, in the later pattern mining section, an (sub)itemset pattern candidate is firstly a subset of the event (code) itemset and then satisfies other requirements. We denote the (sub-)itemset pattern as q.

We denote P as a database(list) of event sequences, i.e., $P = [s_1, s_2, ...s_n]$ where n is the number of sequences. we define the *support* of a pattern p, denoted as $Supp(p, P)$, which means the number of elements of the set $\{s : s \in P, p \sqsubseteq s\}$.

Similarly, we define a database of itemsets as Q, i.e., $Q = [I_1, I_2, ...I_n]$ where n is the number of sequences. Then the support of a (sub-)itemset pattern q, denoted as $Supp(q, Q)$, which means the number of elements of the set $\{I : I \in Q, q \subseteq I\}$.

A concept directly related to *support* is the $Minsup$, the minimum support of a pattern. If we set the *minsup* threshold, only patterns with *support* value bigger than this threshold will be selected, while others are discarded. The $Minsup\ ratio$ is equal to $\frac{Minsup}{|P|}$ or $\frac{Minsup}{|Q|}$ for a pattern in sequence database P and itemset database Q respectively. In our case, $|Q| = |P|$ and the value is the total number of lots.

The *gap* in a sub-sequence pattern in this project is defined as the number of

events between two consecutive events belonging to a sub-sequence pattern. For example, for a sequence$[a_1, a_2, a_3]$, if we regard $[a_1, a_3]$ as a pattern, then a_2 belongs to the gap of this pattern and the length of the gap for this sequential pattern is 1 ($gap = 1$).

NB: For simplicity, we may directly use "event" to refer to the "event code" in the rest of the thesis.

Chapter 4

Dataset

We get two types of datasets, one is on the detailed event level, and another dataset documents more general information on the lot/batch level.

4.1 The Event dataset

The first one is the *Event dataset* extracted from the company's internal cloud database system. In the Event dataset, the start time of the events happening in the chip production is continuously logged (in timestamp ascending order). For the detailed information of columns see Table 4.1.

Attributes	Description	Format
Equipment Nr.	The equipment number.	machine code from specification
Time Stamp	The timestamp records the start time of an event in the format	"MM/DD/YYYY hh:mm:ss"
Event Code	Every unique event is assigned a unique code in the log.	"XX-XXXX" (alphanumeric combination)
User Description	Acknowledgment of an event from the user side with a short description of the action recorded.	plain text
Developer Text	Description of the event and additional remarks set by the developers to analyze and track the events.	plain text

Table 4.1: Columns in Event dataset

NB: In the latter phase of the project we get information from experts' domain knowledge that not all events (codes) related to LOH are included in the Event dataset. Thus further data sources need to be checked and added to the analysis in the future.

4.2 The Lot dataset

Another is the *Lot dataset*, which documents the basic information(identity, configurations, and time information) for each lot. There are many more columns in the Lot dataset, but we only choose the ones that are needed in this project: the lot identifier (the Lot ID), the columns related to timestamps calculation, and two Y/N judgment conditions that have a big impact on LOH time selected by the domain experts(which will be used in pre-processing), shown in Table 4.2.

Attribute	Description
Lot ID	A unique identifier assigned to each lot.
Lot Start Date	The date on which the processing of the lot starts
Lot Start Time	The time at which the processing of the lot starts.
Lot End Time	The time at which the processing of the lot ends.
First wafer start	Since each lot consists of a varying number of wafers, this denotes the start of the first wafer.
Last wafer end	The time at which the processing of the last wafer ends.
Mask Exchange	Can be either Y/N. This denotes whether the current lot includes a mask exchange process.
Illumination Variation	Can be either Y/N. This denotes whether the current lot includes an illumination Change process.

Table 4.2: Main columns in Lot dataset

The data volume of the Event dataset is around 2.6 million (2,672,428), with 1337 unique events. In the Lot dataset, the number of lots is 5,659.

Before we carry out further analysis, we first need to build the connection between continuously documented events and the LOH-related information we can calculate from the Lot dataset.

Chapter 5

Problem Statement

We want to improve the scanner productivity, starting by knowing what is happening behind LOH time. We want to find the most interesting events and patterns with a big impact on LOH. This means either the events and patterns themselves are taking a long time or their appearances are the reasons/signals of the delay in LOH.

5.1 Motivation

The first idea is to directly calculate the influence of events and patterns. Then for each lot we have $LOH = \sum_{i=1}^{m} e_i$, where there are m unique events in a lot. By doing this we change the problem into solving an equation set. This is under the assumption that the impact of each event on LOH time is fixed and independent – then naturally the independence holds for a pattern(composed of several events), the impact is simply the summation of every single event within a pattern.

However, the impact of events and patterns in different situations may vary. Firstly, due to the nature of the machine and manufacturing, the time taken for a certain event(like the event refers to a certain physical action in the scanner) could vary a bit under different conditions. Secondly, we notice that the events happening in the machine are potentially highly correlated. For example, a certain event may raise some errors, then several events happening behind it may be triggered automatically to solve this error, thus the correlation of these events is high. The correlation will lead to our impact estimation for these events or patterns being inaccurate. Last but not least, the events may have an interactive effect on the LOH time. In other words, the appearance of a certain event may influence the time taken for some other events, so that the time taken by these events happening together is not equal to simply adding up each event's time together. These corrupt the fixed event influence assumption. Thus, we think it is reasonable to keep the analysis with some granularity and differentiate different circumstances.

Besides the consideration of the uncertainty above, we also have some expectations on the measurement of the influence of events and patterns: 1) we expect it is in the context of the time (so in the same unit as LOH, in second); 2) we expect to achieve the breakdown of LOH and distribute the influence. That is we can break the LOH into different small pieces (events and/or patterns). Every small piece is able to be explained quantitatively while adding the impact of every small piece together we can recover the total LOH.

With the considerations and expectations above, we define the impact of a certain event or pattern as its Shapley Value [29] in a lot and call the impact the *time influence*, denoted as ψ.

The main idea of Shapley Value [29] is to distribute the final result for features involved in leading to this result based on each of their contributions fairly. So for a certain event or pattern s, its time influence is:

$$\psi_s = Shapley\,Value(s) \tag{5.1}$$

Shapley Value starts analysis locally on every lot so that we can differentiate the circumstances and keep the granularity and flexibility. The ψ is in the same unit of the LOH, in second. With the ψ of all events and patterns in a lot, we can recover the LOH time for this lot.

For the result of the time influence quantification analysis of a lot (using the Shapley Value), its meaning and accuracy are built on the form and the quality of the data input, and the quality of the underlying machine learning prediction model. Hereby we formalize our sub-problems in achieving the final goal as follows.

5.2 Sub-problem 1

Sub-problem 1:
Mine interesting patterns. Create a feature matrix with proper pattern mining algorithms.

For the interestingness, we mean the pattern that is taking a long time compared with other events and patterns (for example, it represents many extra measurements and calibrations), or it is the reason or the signal of the delay in the LOH (for example, it indicates an idle period in the system).

For feature matrices with proper patterns, we hope to include the patterns: 1) they can potentially improve the predictive performance of the machine learning model; 2) they are interesting for us to investigate its Shapley Value. The interestingness here is mainly based on the consideration of the correlation and interactive effect between events. From the perspective of correlation, we hope to find events that their happening together would reveal a high-level root cause. From the perspective of interaction, we hope to find events that their happening together would probably have a larger time influence than they appear alone.

We need to translate the data into the languages that the machine learning model can understand, by creating a feature matrix. To evaluate the time influence of the patterns we need to include them as features in the feature matrix. Meanwhile, we can expect the addition of pattern features could improve the predictive performance of the machine learning model.

The feasibility of adding patterns as new features and the expectation of model performance improvement are rooted in considering the interaction effect in the events. As we discussed in the motivation part above, the appearance of a certain event could influence another. Thus the combined influence on the LOH might not be equal to the sum of the two. Thus, by adding patterns, we are likely to provide new information that could reveal the interaction under the mechanism of the industrial process and consequently help the machine learning models to learn more and have better prediction performance.

5.3 Sub-problem 2

Sub-problem 2:
Train a predictive model based on the created feature matrix in sub-problem 1 that minimizes the squared error between prediction and real LOH value.

$$\textbf{minimize} \ \sum(predicted\ LOH\ - true\ LOH)^2 \tag{5.2}$$

The calculation of Shapley Value is built on a prediction model. One can intuitively understand as we are using the machine learning model to understand the internal mechanism of the industrial process, through predicting LOH time.

In this project, our main goal is not to pursue the prediction performance to the extreme, as our final goal is the time influence result. However, the calculation of Shapley Value is built on the prediction model, thus we still hope its performance is as good as possible. Meanwhile, by design hope and usability of the tool, we also need to consider the time consumption.

5.4 Sub-problem 3

Sub-problem 3:
Rank and interpret the Shapley Value time influence quantification result.

By sorting events and patterns' time influence value in descending time influence, we can find the ones that have a large influence on the LOH that we care about the most. With the help of the analysis from a domain perspective, we can have a deeper understanding and gain insights into what their impact is in reality.

It is worth noticing that we start from the local perspective, to evaluate the time influence of each event in every lot. Then, by integrating all the local results, we can also easily get global insight in general.

Chapter 6

Related Work

6.1 Scanner event data analysis

When analyzing the relationship between events and LOH time, the previous intern started by using the unsupervised method K-means [7] to get several clusters and labeled each lot by the cluster it belongs to [8]. Then the analysis task is transferred into a classification problem when he uses events to predict LOH and analyze their relationship. However, changing the target from LOH value to group label leads to the loss of accurate numerical information. If we do so, we can no longer achieve the goal of quantitatively analyzing the features in terms of time(in the unit of a second). Also, there lacks an explanation of the meaning in different classes/clusters. With regard to this, we decide to approach the prediction task from the regression angle. The last intern also tried using the association mining rule to find the patterns (in this case the order information is neglected). Meanwhile, there is no further quantitative analysis of these patterns' influence with regard to time.

6.2 Data blending/Event bucketing

The data blending method used by the last intern is to create a workflow in Knime [1]. However, this method takes more than 20 hours to run on the company computer [2] [8] (in our test run, it takes around 25 hours), making the model unable to be really used by the domain experts. The main reason is several "hard" *For* loops are used. In our experiment, we resort to python and shorten the time significantly.

6.3 Feature influence quantification

We want to quantify the (time) influence of features (here the features are events or patterns). Traditionally, when mentioning feature influence, one could think of many existing feature importance measures. They assign scores to input features in a predictive model that indicates the relative importance of each feature when making a prediction [9]. The scores can tell us which features may be the most or least relevant to the target. However, they usually only provide global insights into features.

[1] A graphical programming platform `https://www.knime.com/`.
[2] Configuration: Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz 2.21 GHz

6.3.1 Explainable AI and Model agnostic

When we consider the problem from a domain practical perspective, the relationship between events is complicated. Meanwhile, due to the nature of the mechanical machine and industrial process, the time influence of events is likely to vary in different situations (or in different lots). So it is reasonable to keep the analysis with some granularity and differentiate different circumstances. We thus think about local explanation power in model agnostic [32] methods.

Explainable AI (XAI)
The model agnostic method belongs to the research area Explainable AI (XAI) [10]. As machine learning becomes a crucial component of an ever-growing number of user-facing applications, XAI has become an increasingly important area of research [11]. It helps to build trust for humans to make decisions and take actions based on the machine learning model results.

The XAI can be mainly divided into two groups: the interpretable models and model agnostic ones.

Interpretable models
The easiest way to achieve interpretability in XAI is to directly use the interpretable models. The common methods are Linear Regression and Decision Tree. These methods are widely used and accepted due to their ability to directly inspect the machine learning model results by returning the decision path in the tree or the weight of features [11].

Model agnostic
Model agnostic is contrary to model specific(the interpretable model is always model specific by definition [32] for example), where the interpretability power does not limit to a specific model type.

Some machine learning models are regarded as black box models that humans do not understand what is happening inside the model. This leads us to the model agnostic, which is designed to explain any model to answer the question of why did the model make this prediction in a human understandable format, as illustrated in Figure 6.1. By giving understandable and persuasive answers, model agnostic helps humans to trust the result brought by the black box model (while the model agnostic method can also be applied to interpretable models).

The model agnostic is a post hoc method, as it is applied after the machine learning model has been trained [32]. The types of model agnostic methods can be mainly divided into global model agnostic methods (which are used to get a general understanding) and local ones (which are used to explain individual predictions).

Shapley Value and breakDown
We choose one model agnostic method called Shapley Value [29]. While there exists another similar model agnostic method also with local explainability power called *breakDown* [32]. The main difference between breakDown and Shapley Value can be intuitively understood in the following scenario — suppose all features to be evaluated are waiting in line to get tested before entering the room to form a team. Then for breakDown, each time the model picks the feature(an event or a pattern) that would contribute the most to the regression prediction result and lets it enter the room to hold the tested features and group them as a team. Then the selection and testing process iterate until all event codes are added. For Shapley value, instead of performing the selection, it lets feature enter in the team in random order [31]. So

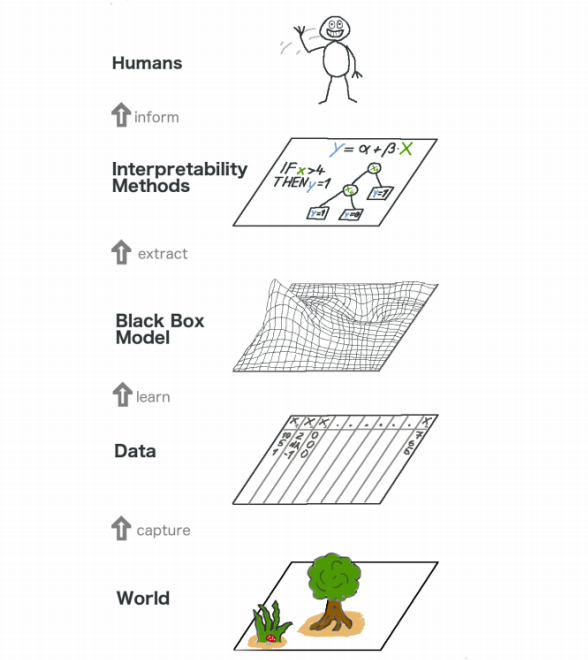


Figure 6.1: Model agnostic illustration [32].
The lowest layer is the real *World*, we capture the world and prepare analysis starting by translating the world phenomena or observations to *Data*, which is the second lowest layer. Then we use machine learning models to find the structure of the real world (data). This is known as the *Black Box Model* layer, as many black box models tend to have better predictive performance than other machine learning models [37]. Above this is the *Interpretability Methods* layer to help us unbox the black box model and present the explainable result to the last layer, us, the *Human*.

for the breakDown [32] method, the contribution of each feature it evaluates depends on the respective feature values that are already in the team.

6.3.2 ANOVA

If we think about estimating the influence of features(events) in the context of time from a global perspective, *ANOVA* [16] could be a choice. ANOVA stands for analysis of variance, designed to analyze the difference between the group means. It is like linear regression but with categorical variables (features), while the target variable is numerical. After fitting an ANOVA model, one can use the coefficient as the estimation of feature influence. However, the result of ANOVA also depends on the order the features are evaluated, like the breakDown [32] method we mentioned above.

6.4 Pattern Mining

Pattern mining is a research area to discover interesting, useful, unexpected patterns in a database [6]. The mined patterns can be useful for understanding the data and decision-making.

6.4.1 Pattern mining in general

In our case, patterns describe interesting sub-itemsets(denoted as q) or sub-sequences(denoted as p). Both itemset pattern q and sequential pattern p are formally defined in Chapter 3. We refer to the definition from [5] to describe our pattern mining task. Given database $\mathcal{D}$, a language $\mathcal{L}$ defining sub-itemsets or sub-sequences of the data, and a selection mechanism $\mathcal{M}$ that determines whether an element $\gamma \in \mathcal{L}$ describes an interesting sub-itemset or sub-sequence of $\mathcal{D}$ or not [45]. The task is to find all interesting sub-itemsets or sub-sequences:

$$\mathcal{T}(\mathcal{L}, \mathcal{D}, \mathcal{M}) = \{\gamma \in \mathcal{L} \mid \mathcal{M}(\mathcal{D}, \gamma) \; is \; true\} \tag{6.1}$$

The classical task of pattern mining is in fact an enumeration problem. One can enumerate all pattern possibilities of $\gamma \in \mathcal{L}$ and assess them with $\mathcal{M}$, thus only a single correct answer should be found. In the early development of the field, this naive enumeration method is used. However, the time and memory consumption increase explosively, thus people are devoted to optimizing the searching process by such as using different data structures to represent the database [6] or optimizing the combinatorial searching [12] techniques to reduce the search space [6].

6.4.2 Sequential pattern mining algorithms

One of the major problems in pattern mining is the explosion of the number of returned patterns [45]. Naive enumeration is unrealistic with the increase in the number of features in solving the real-world problem.

Researchers are devoted to optimizing the search process. *AprioriAll* is the first sequential pattern mining algorithm [17], and *GSP* [18] is an improved version proposed by the author which applies breadth-first [22] search and uses the *horizontal database* structure to represent the original sequence database [6].

However, algorithms adopting the breadth-first principle suffer from time consumption problems (multiple scans of database) and memory consumption problems (it must keep all sequences of length k in memory to generate patterns of length $k + 1$). Another data structure proposed by researchers to represent the sequence database is the *vertical database* (or so-called *IDLists*) [6]. Under this data structure, the researchers utilize depth-first [23] search, which avoids the two drawbacks faced by breadth-first algorithms mentioned above. Representative of the sequential pattern mining algorithm is *Spade* [19].

The search algorithms that use *IDLists* data structure need to frequently do the join operation on *IDLists*, which is costly. To further optimize this, another improvement is to encode the IDLists as bit vectors [6]. The representative algorithm is *Spam* [20].

To further reduce the number of join operations, researchers proposed a method called *co-occurrence pruning* [21]. It requires an initial scan of the sequence database to create a *co-occurrence map*($CMAP$) [6] that stores all frequent sequences composed of two items(events) in E. So in the searching phase, one pattern can be directly eliminated if its last two elements do not belong to $CMAP$, which reduces

the join operations and accelerate the searching process. The algorithms with utilizing *CMAP*, like *CM-Spam* [21] and *CM-Spade* [21], outperform aforementioned algorithms like *GSP* and *Spam* significantly [6].

CM-Spade is claimed to be the fastest algorithm [21], which is the reason we choose *CM-Spade* in our experiments.

6.4.3 Sequential pattern mining algorithms with variations

A huge amount of patterns may be returned by algorithms with regard to original sequence database characteristics and parameters (like minsup threshold) set by users [6]. To avoid the overwhelming of patterns, researchers are also devoted to finding concise representations (a subset of all sequential patterns that provides a meaningful summarization) [6] instead of returning all patterns simply to satisfy the minsup threshold.

There are three main variations [6]:

- Closed sequential patterns: the set of sequential patterns that are not included in other sequential patterns having the same support,
- Maximal sequential patterns: the set of sequential patterns that are not included in other sequential patterns,
- Generator sequential patterns (aka sequential generators): the set of sequential patterns that have no sub-sequence having the same support.

The method we choose in our experiments for finding closed sequential patterns is *CM-ClaSP* [21], for finding the maximal sequential patterns is *VMSP* [24], and for finding the generator patterns is *VGEN* [25]. They all adopt *vertical representations*(*IDLists*) of the original sequence database.

6.4.4 No previous work in mining sequential patterns with numerical target

In our case, the interestingness of patterns lies in that either they are taking a long time compared with others, or are the reasons leading to the delay in LOH. Our target LOH is numerical (thus the pattern mining task falls within the regression domain). However, to the best of our knowledge, there is no previous work on mining and analyzing sequential patterns with a numerical target.

There exists previous work that is dealing with patterns and with a numerical target. In Pattern-Aided Regression Modeling and Prediction Model Analysis [13], the authors are devoted to finding *contrast patterns*. The authors firstly use a baseline model to divide the instances in the dataset into two classes, LE and SE (LE is the group composed of instances where small prediction errors are made, while SE is the opposite). The *contrast patterns* are the patterns that get very different *support* values in different classes.

They then build local regressions models based on the patterns. The weighted average of the local regression models is taken as the final prediction result, with the goal of improving the regression prediction performance. Meanwhile, from the explainability perspective, the local regression models on contrast patterns are easy to interpret and thus can be used to represent the predictor-target relationship [13].

The difference between this job and our scenario lies in three points: 1) the main goal. They want to achieve higher regression prediction performance, while we are more interested in explaining features/patterns themselves(so the prediction is just a

middle goal of our task). 2) the type of patterns we want to find. The final interesting patterns we are interested in are the patterns that have a big time influence on the target variable, while the patterns the author in this project want to find are the patterns that optimize a quality measure called total residual reduction (trr)[13]. The residual is the difference between the predicted and true/observed target variable values. 3) the order. the patterns they find are the weighted combination of a set of features(so without order), while we want our patterns to keep the order information.

Previously there also exist methods to find patterns with the consideration of the order, like in paper [14] and [15]. However, the task type is limited to the classification domain, instead of regression.

In [14] the authors follow the following heuristic to search for patterns as features for classification task: 1)pattern should be frequent; 2)patterns should be distinctive of at least one class; 3)pattern feature set should not contain redundant patterns. In [15] the authors raised a two-stage methodology to mine sequential patterns for the classification tasks. In the first stage, they also start by using the sequence mining technique to extract sequential pattern candidates before further proceeding to the second stage.

Due to the final quantification goal we want to achieve, we have to keep the numerical nature of our target variable (thus a regression task). But we can get inspiration from the similarity in these previous works to find sequential patterns as features and as potential candidates for the patterns that may have a big time influence on the LOH time.

Chapter 7

Method and Algorithm

In this chapter, we explain the methods and algorithms we use in detail. We will first start with LOH-related calculation methods from the industry, then comes to the methods we used to translate datasets into the format that computers can process. After that, we explain the methods and algorithms we design or choose to quantify the time influence of events and patterns. After this, we introduce the algorithms and methods we apply to find pattern candidates.

7.1 Solution pipeline

Before diving into the details we first show the general pipeline in this project to help understand (Figure 7.1). The blocks in orange squares are methods/processing steps, while the content in the purple ellipse is the (middle/final) output of previous methods.

We first start with the pre-processing of data. We format the time information into timestamps and remove the lots with the unqualified logs. Then we use the pre-defined methods by the department and experts to filter out the events we are not interested in. We then calculate the LOH time and the start and end time of LOH. After this preparation, we bucket the LOH to its corresponding lot's LOH duration. At last, we do some data engineering to format the bucketed event sequence into the required format for further analysis – the basic occurrence feature matrix containing only single event features, the event sequence lists database for sequential pattern mining, and the event set lists database for itemset pattern mining.

Then we design a two-stage method to mine pattern candidates. In the first stage, we carry out experiments on either using Krimp [45] to mine itemset patterns or using one of the four algorithms(CM-SPADE [21], CM-ClaSP [21], VMSP [24] and VGEN [25]) to mine sequential patterns. After getting pattern candidates, we carry out a further selection to get the final candidates. For each pattern mining method, we add its corresponding pattern candidates separately to the original feature matrix (which contains only single events as features). By doing so we get six different feature matrices. The performances of the pattern mining results under different methods are compared.

We then send six different feature matrices separately into the rest time influence quantification (sub)pipeline: a combination of a regression model (either the random forest [46] or the gradient boosting [46]) and Treeshap [37] to calculate the time influence results – the Shapley Value for events and/or patterns. Since we prepare six different feature matrices and apply two underlying prediction models, we will get twelve results based on different set-ups.

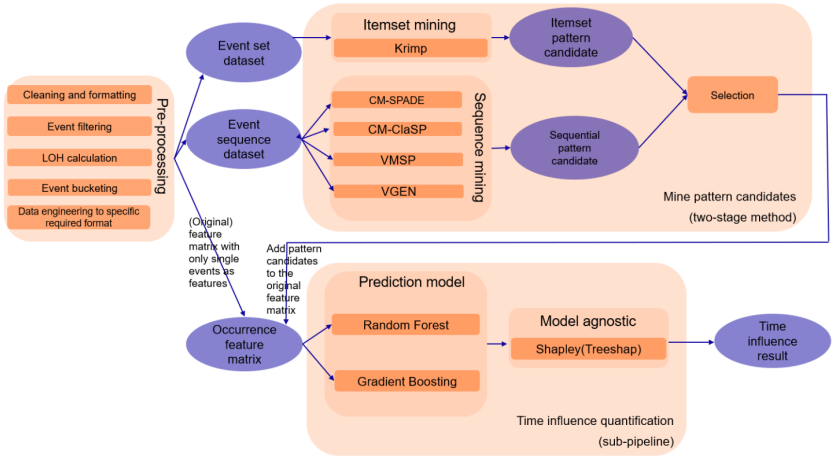


Figure 7.1: Project solution pipeline
The pipeline starts from pre-processing of datasets to prepare the feature matrix to send into the "Time influence quantification" sub-pipeline in the bottom orange square. The feature matrix can contain only original events as features from the original dataset or add extra pattern features. If adding pattern features, we need to apply the two-stage "Mine pattern candidates" methods in the upper orange square.

7.2 Cleaning and formatting

For the Event dataset pre-processing, we transfer the time information into standard DateTime to support the later bucketing task. The Lot dataset contains many more columns than the columns we presented in the Dataset chapter. We only choose the columns we need as discussed in the Dataset chapter (the lot identifier, time calculation related, and important judgment factors selected by domain experts). Then we did multiple checks on data validity (more details in Appendix B).

7.3 LOH calculation

In the previous chapter, we introduced the role of LOH in the scanner's wafer production. LOH is not existing information we can find in the dataset. In reality, many detailed calculations(in Appendix C) are needed to calculate the LOH time based on the lot's streaming condition using the Lot dataset.

At a high level, LOH is from the end of the last wafer expose of the previous lot to the start of the first wafer expose of the next lot (shown at a high level in Figure 2.1). The productivity team analyses LOH in two parts:

$$LOH = front\ LOH\ (FLOH) + back\ LOH\ (BLOH) \tag{7.1}$$

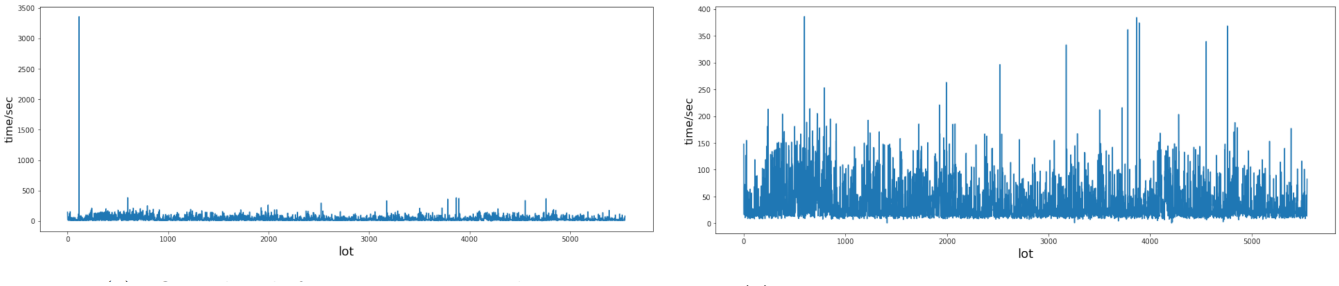
Calculation
The real value of FLOH and BLOH depends on the lot streaming condition. We use the formula pre-defined(in Appendix C) by the company to do the detailed

calculation. The input is several columns related to the lot's and wafer's start and end time in the Lot dataset, and the output is the LOH time.

In the middle of the calculation, we also get the start and end time of both front and back LOH duration for each lot, which we can use to bucket the events to the lot it belongs to.

Rounding
After calculation, we round the LOH value to one decimal place (in line with the last intern, to simplify the further analysis later).

Basic exploratory analysis
After calculating the total LOH value and getting the FLOH and BLOH time range, we did a basic exploratory analysis to check LOH time by plotting the LOH value.



(a) LOH values before removing wrongly logged lots

(b) LOH values after removing wrongly logged lots

Figure 7.2: LOH value barplot
The lot with the LOH value larger than 3000 seconds is regarded as wrongly logged and thus removed.

From Figure 7.2a we can see there is one lot with the LOH value extremely high(larger than 3000 seconds). After consulting domain experts, it may be because the data is logged wrong, so we decide to remove this lot. We thus add a filter, to remove lots with LOH values bigger than 3000 seconds. After removing the wrong lots the LOH values are plotted in Figure 7.2b. We can see at present all LOH values are below 400 seconds, which is more reasonable.

We further plot the histogram of the LOH value with the bin size of 200(this can be changed by the user). The x-axis scale range is set to (0, 400) as the maximum LOH value shown in Figure 7.2 is around 400s (due to the automated design of *matplotlib* plotting package [26], the shown x-axis scale range is (0, 260) in Figure 7.3). The x-axis is the LOH value, the y-axis is the number of lots whose LOH value falls into a certain bin. We can see that the majority of lots are with the LOH value around 14s(big peak), which we regard as good streamed lots.

7.4 Event filtering

Not all events in E are useful in our analysis, but we have limited knowledge of what are they. After discussing with domain experts we try to set up two filtering mechanisms to filter out events we are not interested in our analysis beforehand.

- the "pollution" event codes pre-excluded by domain experts when they do the

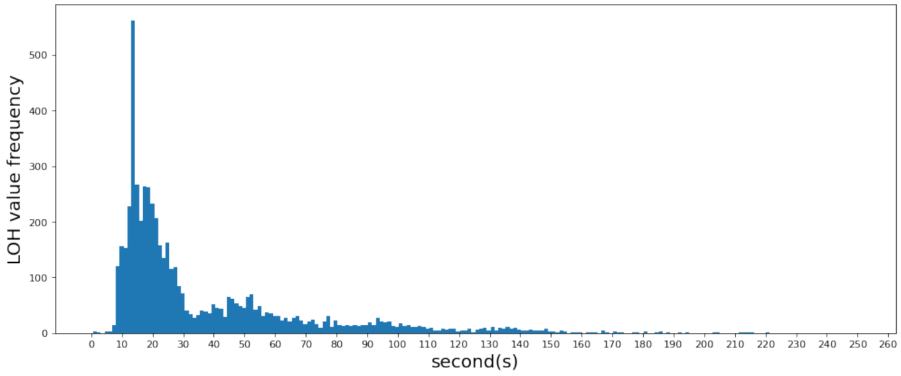


Figure 7.3: LOH value histogram
The majority of lots are with LOH value around 14s(big peak), which we regard as good streamed lots.

analysis manually (only a handful of events).[1]

- the "basic events" codes

The reference table that lists event codes that are usually pre-excluded by domain experts when they are doing analysis is in Appendix A. For the "basic event codes", we mean the code for an event that belongs to the standard wafer-producing procedure. While we don't have such a complete list, we thus define the filtering methods ourselves after consulting domain experts:

1. Find out lots without Mask Exchange and Illumination Variation (details see Table 4.2 in Dataset introduction section).
2. Calculate the 0.1 quantiles of lots' LOH value found in step 1. Find out lots in 1 with LOH less than 0.1 quantile LOH value.
3. Find the union of events in lots found in step 2, known as the "basic events list".
4. Remove the event code starting with a certain letter prefix from the "basic events list" (see Appendix B). The rest of the codes are regarded as final "basic events", and we store them.

Finding out lots under certain conditions is easy. Then to get the union of the events belonging to "basic lots", we first bucket the Event codes to the LOH range of these lots, choosing the occurrence matrix fashion. We then do the join operation to the occurrence matrix based on columns to get the union of the events in these lots as "basic events". After this, we remove all the event codes starting with a certain letter combination from the "basic events list". We append the "basic events list" to the pre-excluded code list got from the domain expert, and get the final excluding event codes.

We remove the "all_ex_codes" from the Event dataset, meanwhile, remove the lots that are composed of only events in our excluding "all_ex_codes" list, before carrying out further experiments. For these removed lots, they got a mean of 12.47 seconds

[1]This list can be further improved by asking more domain experts to report the polluted event codes and we further remove them, to further refine the dataset we feed into the model.

and a standard deviation of 4.76 seconds. They are indeed among the LOH range we regarded as normal lots' LOH time.

After event filtering, the anonymized reference table is created.

7.5 Data blending/Event bucketing

In the calculation of the total LOH using the formula above on the Lot dataset, in the middle we also get a by-product – the start and end time of both front and back LOH. We can then use this time range to bucket each event in the Event dataset into the LOH time interval it belongs to. By doing so we blend two datasets, so for each unique Lot ID identifier, we get it's corresponding LOH event sequence list /event set list/ feature vector as discussed in section 7.6.

The data blending takes around 25 hours using the last intern's method, making the model unable to be really used by the domain experts. In our experiment, we resort to python and take the advantage of existing tools (mainly the ".between" function in the pandas series) in Pandas. By doing so, we shorten the time to around 6.5 minutes to achieve the same performance as the last intern's work, i.e., to directly bucket all events in the Event dataset to their corresponding LOH duration

After we decide to use the filtering mechanism mentioned above, we further change the processing order: we firstly only bucket the events belonging to normal lots(the lots that satisfy the step 2 among the four filtering methods mentioned in sections 7.4). After getting the basic event codes we exclude them from the Event dataset, and we remove the lots that only contain basic events. Then we do the bucketing to the rest of the events and to the rest of the lots. In this order, it takes less than 1 minute to finish the bucketing process.

A bit more discussion
When checking the Event dataset, we find that there seem to be several events happening at the same time. If so, when preparing data into a proper data structure for the sequence mining task, we need to do an extra pre-processing step to pack the simultaneously happening events as an itemset. So for each lot, it is composed of an ordered list of itemsets.

After doing a check on the Event dataset, we find from developer text that even for the same event code, the real message/action could vary. One detailed example of an event code querying result is in Figure D.1 in Appendix D. We can see that an event code could mean the reticle is moved, but the starting and ending place of the reticle movement could vary. Thus, we don't do the extra packing, and keep all the events as they are and bucket them into each lot's LOH in the order they were documented in the Event dataset (in time order).[2]

7.6 Data structures used for further analysis

To support the follow-up experiments we use three types of data structures. For the time influence evaluation subtask, we create the occurrence feature matrix data structure. For the sequential pattern mining subtask, we use the event sequence lists data structure. For the itemset mining method, we use event set lists data structure, a variation from event sequence lists data structure.

[2]This also makes the *event sequence dataset* we build in a later stage a special case in sequence mining academic area, that each sequence is composed of an ordered list of single items, instead of an ordered list of itemsets (a typical representative is customer transaction data in the supermarket).

Below we will also give an example of how each data structure is generated from the original demo Event dataset in Table 7.1 (suppose A, B, C, and D are all events in the Event dataset; the timestamp is the start time of the event, which is in line with the description in Dataset section), and a demo result of the start and end time of front lot overhead (FLOH) and back lot overhead (BLOH) (which we get while calculating the LOH time using the formula above) in Table 7.2.

	timestamp	event code
lot 1	2022-6-6 00:01:00	A
lot 1	2022-6-6 00:02:00	B
lot 1	2022-6-6 00:04:00	C
lot 1	2022-6-6 00:07:30	D
lot 2	2022-6-6 00:10:00	B
lot 2	2022-6-6 00:15:00	C
lot 2	2022-6-6 00:21:00	B
lot 3	2022-6-6 00:31:00	A
lot 3	2022-6-6 00:35:00	D

Table 7.1: Demo: the Event dataset

	FLOH start	FLOH end	BLOH start	BLOH end
lot 1	2022-6-6 00:00:00	2022-6-6 00:03:00	2022-6-6 00:07:00	2022-6-6 00:08:00
lot 2	2022-6-6 00:09:00	2022-6-6 00:15:00	2022-6-6 00:20:00	2022-6-6 00:24:00
lot 3	2022-6-6 00:30:00	2022-6-6 00:32:00	2022-6-6 00:35:00	2022-6-6 00:40:00

Table 7.2: Demo: start and end time of each lot's FLOH and BLOH (calculated using the Lot dataset)

7.6.1 Event sequence lists database

The first data format we create is the event sequence lists database. We get the database by doing a scan through the Event dataset and let the events get into each Front/Back LOH range bucket in the order based on their timestamp. The target data structure we want to get is as Table 7.3.

One thing to notice is we set the bucket to be left(start) inclusive, but not right(end) inclusive. For example, for event C in lot 2 in the Event dataset demo in Table 7.1, its starting timestamp is right at the end of the FLOH end time of lot 2, so we don't regard it as an event belonging to the LOH of lot 2. Meanwhile, for event D in lot 3, its starting timestamp is right on the BLOH start of lot 3, while we regard it to belong to the LOH of lot 3.

.

7.6.2 Event set lists database

This is a variation of the event sequence database. To transfer the data into this structure, the general process is the same as how we get the event sequence database. The difference is that instead of letting events go to each FLOH/BLOH bucket in time order, we take the union(set) of events. By doing so, we no longer consider order information in this data structure.

After taking the union of the event codes in each lot's LOH we get Table 7.4.

Lot ID	Event code series	LOH time (s)
lot 1	A, B, D	240
lot 2	B, B	600
lot 3	A	420

Table 7.3: Demo: event sequence lists database structure
Events are bucketed into the LOH duration it belongs to. Event C is removed as it does not fall into any BLOH or FLOH duration listed in Table 7.2.

Lot ID	Event code series	LOH time (s)
lot 1	A, B, D	240
lot 2	B	600
lot 3	A	420

Table 7.4: Demo: event set lists database structure
Events are bucketed into the LOH duration it belongs to. Compared with Table 7.3, the duplicated event B is removed in lot 2 as we are dealing with the set.

7.6.3 Occurrence feature matrix

Figure 7.4 shows our third data structure: the feature matrix based on occurrence. The entries take binary values, with 1 meaning presence and 0 absence.

Each row of the occurrence feature matrix represents a lot, while each column represents an event. If an event e_i(the $i-th$ event in E)[3] appears in the $j-th$ lot, then the entry (j, i) in the feature matrix will be set to 1. The information in the occurrence feature matrix here is consistent with the event sequence lists and event set lists data structure above. In the feature matrix each row is a lot's event vector representation.

$$\begin{array}{c} \\ \texttt{lot 1} \\ \texttt{lot 2} \\ \texttt{lot 3} \end{array} \begin{array}{c} \begin{array}{cccc} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 7.4: Demo: occurrence feature matrix structure
The event A appears in lot 1, thus the entry on the top left gets 1, while event D does not appear in lot 3, thus the bottom right entry get 0. The rest are similar.

7.6.4 Adding mined pattern candidates to the feature matrix

In a later section, we will apply several pattern mining techniques together with a selection mechanism to mine pattern candidates with the potential to lead to large LOH time. After getting the pattern candidates we need to add them as new features

[3]More precisely in our case, e_i is the $i-th$ event in our anonymized reference table, since after event filtering some events are filtered out from the original E

into the original feature matrix(which contains only single event features) and go through the rest of the pipeline to get the final quantified time influence evaluation results.

For sequential pattern mining, SPMF [49] can output the patterns together with the lot id the pattern appears. Thus for each sequential pattern, we just need to add another column to the end of the original feature matrix and set the row entry to one, if the pattern appears in that corresponding lot. For Krimp, we need to first create queries ourselves to find the lot where each pattern appears. After finding the lot id(row index), other operations to add pattern candidates as new features to the original feature matrix are the same as sequential pattern mining.

Continue with the above example, suppose we have a pattern that appears in lot 2, then we can update the feature matrix with an extra column at the end of the original feature occurrence matrix. The demo result is shown in the matrix in Figure 7.5:

$$\begin{array}{c} \\ \texttt{lot 1} \\ \texttt{lot 2} \\ \texttt{lot 3} \end{array} \begin{array}{c} \begin{array}{ccccc} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{pattern} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 7.5: Demo: occurrence feature matrix updated with a pattern feature
Similar to Figure 7.4, while with a newly added column at the end to represent the pattern feature we want to analyze. This pattern appears only in lot 2.

7.7 Shapley Value, Shap and TreeShap

With several considerations and expectations to evaluate the influence of events or patterns(discussed in section 5.1), we adopt Shapley Value as the time influence measurement. Shapley Value is built on other prediction models. We adopt Random Forest and Gradient Boosting as the underlying machine learning prediction models (discussed in section 7.8).

From another angle, we can understand the combination of the machine learning prediction model and Shapley Value in an intuitive way — Industrial process is a complex process, it is difficult for humans to understand the whole picture and every tiny detail. So we firstly use a machine learning model to understand the industrial process (by predicting the LOH time) and abstract it into "knowledge" implicitly entailed in the model. After this, we need to reveal what is inside of the black box and generate understandable explanations for the human users. By explaining the model, we achieve our goal to evaluate the time influence brought by certain events or patterns. The explanation step is also known as *model agnostic* [32] in academia, and Shapley Value itself is a model agnostic method. By doing so, we carry out a quantified analysis of the influence of events in the context of time. After this, the finding of the top 30 events or patterns that potentially have the biggest time influence on the LOH (which may lead to the delay and production decrease) is easily within reach.

Though later the implementation we use is Shap(Treeshap), it is still built on Shapley Value and the difference lies in the way to compute the Shapley Value. So

in this part, we are going to firstly introduce Shapley Value in detail. Below we will firstly introduce the mathematics behind Shapley Value, followed by an intuitive example of how Shapley Value is computed in a real-world example. After that, we explain its relationship with our use case and machine learning (models).

7.7.1 General introduction and connection with our use case

Shapley Values is coined by Shapley (1953)[29]. It is a method from coalitional Game Theory [28]. The main design idea is to distribute the payout for people involved in a game based on their contribution fairly.

The Shapley Values approach was first translated to the machine-learning domain by Strumbelj and Kononenko [51], [52]. By interpreting a model trained on a set of features as a value function, Shapley value provides a natural and principled way to explain the predictions of nonlinear models common in the field of machine learning [30]. Among many model agnostic algorithms, Shapley Value is the only method that has a solid theory base [32]. This gives practitioners more confidence to use it in real applications.

How is Shapley Value applied in our situation? In evaluating the time influence of the events or patterns belonging to the LOH range of the lithography scanner data? we map the concepts as below:

- each lot of microchip production is a "game",
- each event or pattern is a "player",
- the difference between the predicted final LOH time and the expected LOH (the mean of the LOH prediction results of all lots) is the "payout",
- the time influence of event or pattern is the payout distribution result based on their "contribution".

The idea is through distributing the difference between the prediction and the mean prediction (the "payout"), we achieve our goal of quantifying machine events' or pattern's time influence on the LOH.

7.7.2 Shapley formulation

Now we give the Shapley Value in mathematical format [53].

Two types of formulation
There are actually two ways to write the mathematical formula, one is from the perspective of combination(or so-called coalition; without order) shown in Formula 7.2, and another is from the perspective of permutation(with order) shown in Formula 7.3.

$$\phi_i = \sum_{S \subseteq F - \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} (v(S \cup \{i\}) - v(S)) \tag{7.2}$$

Shapley Value math formula from coalition perspective [53]

$$\phi_i = \frac{1}{|F|!} \sum_{P} (v(S \cup \{i\}) - v(S)) \tag{7.3}$$

Shapley Value math formula from permutation perspective [53]

where v is a value function to evaluate the payout of a combination(in formula 7.2), and of a permutation (in formula 7.3). F is the set of all unique events. S is the subset of F, and $S \subseteq F$, $F - \{i\}$ means S is all event subsets of F excluding the event i which we are interested in.

Weight
Although the first formula could reflect the "coalition" design idea behind Shapley Value and is used by the majority of the papers, the complicated fraction(which is a weight) in Formula 7.2 might be more intuitive to understand from a permutation perspective. If we are dealing with permutation, then we just need to treat each permutation equally. So after calculating the $v(S \cup \{i\}) - v(S)$ for all possible permutations, we just need to calculate their mean. The number of possible permutations of all unique events and patterns is $|F|$, thus there are $|F|!$ different permutations.

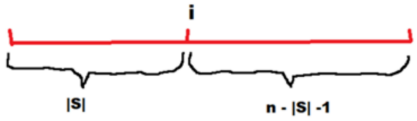


Figure 7.6: The origin of weight in Shapley Value formula [53]
Suppose all n features stand in a line, and there are $|S|$ features before our feature of interest i, thus there are $(n - |S| - 1)$ events behind it.

But since Shapley Value is designed in the idea of coalitional game theory (from a combination perspective), we know that one combination of events and patterns could have various different permutation ways. Thus, we can aggregate them. See Figure 7.6, suppose we have a lot with n unique features(events and patterns) in LOH duration and n features are standing in a line. Suppose there are $|S|$ events before our feature of interest i, thus there are $(n - |S| - 1)$ events behind it. The number of all different orders of events before feature i is $|S|!$, and the possible number of orders of events behind feature i is $(n - |S| - 1)!$. Thus we multiply $|S|!$ by $(n - |S| - 1)!$, to aggregate all the permutation possibilities here, as each of them would get the same value function result. Using the coalition form to carry out the calculation in practice will significantly shorten the computing time.

Shapley Value calculation demo
Here we present an intuitive example of how Shapley Value is calculated in a certain lot. Suppose we know a lot's LOH is composed of events A, B, and C, and the LOH value is 160s. Suppose we have also trained a predictive machine learning model on all lots in the dataset, so we get the value function v to evaluate the *(time) contribution* of each event in different cases. What we want to know is the event A, B, and C's time influence on the LOH in this certain lot, respectively.

We use the trained predictive model to predict the LOH time when only a subset of events is present in the lot. The predicted result is shown in Table 7.5.

Event	Predicted LOH time taken(s)
A	40
B	50
C	60
A, B	95
A, C	110
B, C	120
A, B, C	160

Table 7.5: Demo: predicted LOH time of all different coalitions(subsets) of events

All possible permutations	Permutation Occuring Probability	Time contribution in different cases(second)		
		A	B	C
A, B, C	1/6	A=40	A,B-A=95-40=55	A,B,C-A,B=160-95=65
A, C, B	1/6	A=40	A,C,B-A,C=160-110=50	A,C-A=110-40=70
B, A, C	1/6	B,A -B =95-50 =45	B=50	B,A,C-B,A=160-95=65
B, C, A	1/6	B,C,A-B,C=160-120=40	B=50	B,C-B=120-50=70
C, A, B	1/6	C,A-C=110-60=50	C,A,B-C,A=160-110=50	C=60
C, B, A	1/6	C,B,A-C,B=160-120=40	C,B-C=120-60=60	C=60
Shapley Value/time influence(second)	permutation	40*1/6+40*1/6+45*1/6+40*1/6+50*1/6+40*1/6 = 42.5	52.5	65
	combination	2 *40*1/6 + 2 *40*1/6 + 45*1/6 + 50*1/6 = 42.5	52.5	65

Figure 7.7: Demo: Shapley Value calculation example [50]
The yellow area shows the time contribution calculation of each event in different coalitions; the gray area shows the Shapley Value calculation of event A based on permutation(formula 7.3), while green area shows it based on a combination(formula 7.2). The two cases in orange can be aggregated if we use the combination formula, as shown in "2" highlighted in orange. This also applies to the two cases in blue areas.

Then we can calculate the time influence of each event as shown in Figure 7.7. In Figure 7.7, we can see the calculation process of event A's time influence from both the permutation perspective(in gray area) and the combination perspective(in green area). In the yellow part, the time contribution of a certain event in different cases is calculated.

Further, we can see when calculating the time contribution of event A in different cases, $v(B,C,A)-v(B,C)$ and $v(C,B,A)-v(C,B)$ get the same contribution value 40s, as B,C and C,B are formed by the same combination$\{B,C\}$ (marked as blue areas). Thus when using a combination form of Shapley Value time influence calculation, we can aggregate them(highlighted in the green area with "2" in blue) and reduce computing times of value function(make predictions using the trained machine learning model). The same aggregation can be applied to the permutation cases marked in orange areas, where they get the same calculation process: $v(A)-v(\emptyset)$. The aggregation in the calculation is highlighted in orange in the green area.

Property

There are several properties [32] of Shapley Value that are in line with our expec-

tations. 1) efficiency: the summation of the time influence of all events equals the difference between the model prediction and model predicted mean of all lots. This allows us to breakdown each lot completely, which satisfies the expectation from domain experts; 2) additivity(linearity): if there is more than one value function for evaluation, then the summation of time influence of an event or pattern under different value function is equal to the time influence calculation of this event with these value function combined together. This allows us to calculate the Shapley Values fast for tree-type prediction models we used, where the time influence is simply the (weighted) summation of all trees. This enables us to do the time and computing power-consuming Shapley Value calculation faster.

7.7.3 Relationship with machine learning

Now we explain the connection between Shapley Value and machine learning, taking our evaluation of time influence as an example.

Value function
In the real world, we don't usually have a perfect pre-existing evaluation mechanism (value function) as in the above calculation demo. Here we take the advantage of the machine learning model $f(x)$ and use it to predict the LOH time taken by different feature combinations (coalition) as our value function.

The feature could be either an event or a pattern. The definition of the value function v for machine learning model is $f(x) - E(f(x))$[4]. Thus we get the value function v to evaluate the *time contribution* of each event in different cases to support further Shapley Value (*time influence*) calculation.

Deal with missing value
In the implementation, we need to deal with missing features in the payout calculation(value function). We only have one prediction model that is built on all features and are not able to fit models on a subset of features. The solution to deal with this is to use the random sample to remove the influence of the features that should be "missing" when calculating the payout of the game on a subset of present features.

In the latter description, we call the lot we are analyzing the "target lot", the feature we are analyzing the "target feature" and the subset we are calculating the value function's value the "target subset".

To achieve the removal of the influence of the "missing" feature, each time we keep the features presented in the target subset unchanged. Meanwhile for each "missing feature", we randomly sample a lot in the dataset and use the sampled lot's corresponding feature value as a substitution for the target lot's missing feature value. By keeping the presented feature value unchanged and randomly sampling the "missing feature", we create a "carrier". After the carrier is ready, we again randomly sample the lots in the dataset and use the sampled lot's corresponding feature value as a substitution of our target lot's target feature value [32]. By doing this we get a "complete" lot with all the features present.

By doing the above sampling process several times, we get many "complete" lots. Among them, there will be lots with our target feature present or absent. Then we can calculate the mean of them respectively, to get the $v(S \cup \{i\})$ and $v(S)$ respectively. Then we just need to insert them back into the Shapley value calculation formula to get the final time influence quantification for each event or our pattern candidate.

[4]This is to ensure $v(\emptyset) = 0$, since the best estimator for $f(x)$ when all features are missing is the "mean", $E(f(x))$, the predicted average of the LOH value for all lots.

Sampling method
One thing to notice is here we use the random sample, while in [36] Lundberg also mentions one can choose various ways to carry out the sampling and substitution. For example, one can replace the "missing" feature with the mean value of features in the dataset, or with values imputed based on the data's correlation matrix.

Discussion on correlation and causation in our case
The predictive models capture the correlation of features and the target but are not causal models. If we hope to directly capture the true causal effects between them, the feature we are analyzing needs to be independent not only of other features in the model, but also of unobserved confoundings [33] (or at least not strongly redundant with any other measured features and unmeasured confoundings) [31]. This is a harsh requirement and does not often hold in real-world data.

Hereby we discuss how this potential problem influences the result and analysis of our project and how it can be solved(or minimized).

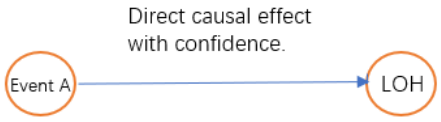


Figure 7.8: Demo: the confirmed causal effect
Event A is known to be completely independent or confirmed by domain judgment.

The first and the easiest situation is that the feature of interest satisfies the independence requirement so that nothing extra needs to be done. Meanwhile, in some scenarios, even if a certain feature of interest is not independent, if we read from the development text or based on previous root cause analysis, the domain expert can directly confirm with confidence its long-time consuming, and direct causal effect with the LOH (illustrated in Figure 7.8).

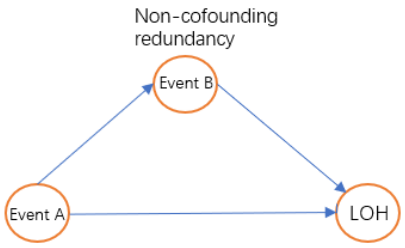


Figure 7.9: Demo: non-cofounding redundancy
Part of the causal effect brought by Event A may output in Event B.

In the second situation, feature A has a causal effect on the LOH, meanwhile, it is influencing another feature B which also has a causal time influence on the LOH (Figure 7.9), then the time influence evaluated by Shapley Value with the predictive model will spread among these two features [31], which is doable in our case. If we want to further accurately estimate the causal effect, we need to remove the

redundant variables [31].

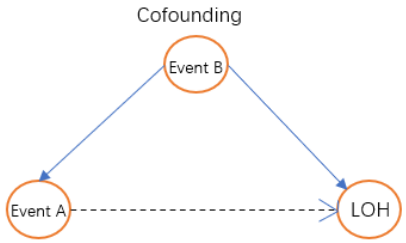


Figure 7.10: Demo: cofounding
Event A does not have a direct causal effect to LOH, but there exists another event B that correlated with both A and our target, the LOH.

The third situation is that feature A we find to have a large time influence on the LOH does not directly have a causal effect to the LOH, but there exist cofoundings [33] that correlated with both feature A and our target, the LOH (Figure 7.11). In this scenario, a potential future work to assist the domain experts to do in-depth analysis is to create a correlation table/heatmap. Whenever there are counter-intuitive or unknown relationships appearing, they can infer from the correlation result to dig deeper into the root cause which may be led by other features correlated with feature A. In academia, there also exists a method called *Double ML*(or debiased machine learning) [34]. It first de-confounds the feature of interest and then estimates the average causal effect of changing that feature [31].

Figure 7.11: Demo: unobserved cofounding
Not all important variables are measured or explicitly known in the analysis.

The fourth situation, which is the most cumbersome one, is that some features suffer from unobserved confounding because not all important variables are measured or explicitly known in the analysis. To know the true causal effect, the gold standard, in this case, is to use some randomization to break the correlation between the features of interest and the unmeasured confoundings [31]. In our experiment, we set the feature perturbation parameter to "intervention", which breaks the dependencies between features according to the rules dictated by casual inference [31], [56]. This setting can also be helpful in solving the problem here. Meanwhile, as discussed in the Dataset chapter, including a more complete data source could help in revealing the hidden variables, and potentially minimizing the problem here.

In the future, we can also involve more Development & Engineering experts to do further analysis based on this assisting tool we make and confirm the causal effect before making the decisions to change or upgrade the system from the domain perspective.

7.7.4 Shap and Treeshap

Shapley Value has been widely adopted after the publication of the paper by Lundberg and Lee (2017) [30] and Python's library for *SHapley Additive exPlanations*, *SHAP* (Lundberg 2019 [31]). The authors of *SHAP* also proposed an efficient algorithm for tree-based models (Lundberg, Erion, and Lee 2018). In later 2020, the same author proposed methods that enable the exact computation of Shapley value explanations for tree-based models named as *Treeshap* [37]. It overcomes the computing power problems due to the requirement to do calculations on all subsets of features, which was a major problem faced by the application of Shapley Values. In the past, computing limitations forced people to resort to approximation solutions.

The basic idea of Treeshap is to push all subset calculations down the tree at the same time[32]. The computation complexity for Treeshap is $O(TLD^2)$, where T is the number of trees, L is the maximum number of leaves in any tree, and D is the maximal depth of any tree [32].

7.8 Regression models for prediction

Machine learning regression models are helpers for us to understand the inner mechanism of the complicated process. Continuing from the previous section about Shapley Value, the regression model is a gadget in Shapley Value calculation, to help us evaluate the "payout" as part of the value function mentioned above.

We choose two tree-based methods in this project, and there are mainly two reasons for that: 1) Tree-based machine learning models such as random forests and Gradient Boosting (another famous version is XGBoost [47], one of the fastest implementations of Gradient Boosting [39]) are popular in non-linear prediction models and gained some reputation in their regression and classification power [37]. Tree-based tends to consistently outperform standard deep models on tabular-style datasets [47], [37]. 2) As mentioned in section 7.7.2, the Shapley Value calculation can be specifically optimized for tree-based methods, which enables us to do the calculation fast and make the pipeline we create more usable in reality.

The two models we choose are *Random Forest* [38] and *Gradient Boosting* [39]. Both of them are ensemble methods, which usually give us better prediction results(thus a potentially better understanding of the internal mechanism of the industrial process).

The building block of Random Forest and Gradient Boosting is the *decision tree* [40]. The theory behind the decision tree is pretty intuitive: it gradually approaches the final confident prediction by continuously splitting the features, and dividing and narrowing the search space. Since the decision tree has the risk of over-fitting [42] on training data, which will lead to bad performance when testing on other data, we also need to prune the tree in case of over-fitting. There exists pre-pruning and post-pruning. Intuitively, pre-pruning is to stop the growth of the tree in an early stage when the tree is building, and post-pruning is to cut the trees after the tree is completely built. One can start the pruning process from bottom-up or top-down [41].

The decision tree can be used to deal with both classification problems and regression problems. Here we apply it in the regression scenario. When dealing with a regression problem, the tree-growing training process is built on minimizing the mean squared error(MSE, which is similar to the formula of RMSE in section 7.8.3 but without taking the root square) while dividing the search space. In the prediction session, the final regression result for a new observation is equal to the average value of the observations falling into the same area.

7.8.1 Gradient Boosting

Gradient Boosting is a decision-tree-based ensemble machine Learning algorithm that uses a gradient boosting framework. In terms of "Boosting", is an ensemble technique where new models are added to correct the residuals or errors made by existing models. In terms of "gradient", the errors are minimized by the gradient descent algorithm. Models are added sequentially until no further improvements can be made. After that, all the models are added together to make the final prediction (see Figure 7.12).

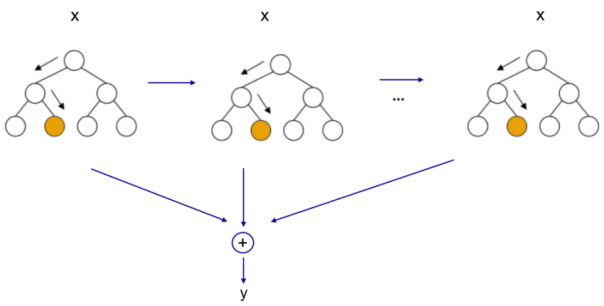


Figure 7.12: Gradient boosting principle [27]
Trees are sequentially added to correct the residuals or errors made by existing tree models.

Another more well-known name connected with Gradient Boosting is XGBoost. XGBoost stands for "Extreme Gradient Boosting". The "extreme" comes from a great combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time[48].

The reason we don't choose XGBoost is that in the implementation procedure, we met a problem of connecting XGBoost implementation [47] and Shapley Value calculation implementation by Lundberg [31]. The XGBoost implementation [47] requires the input data structure as Dmatrix. When we change the prepared feature matrix to Dmatrix format, if a feature completely does not appear in the training dataset sent into Dmatrix, the feature column will be automatically removed. This will cause the problem in a later stage when we are trying to send the trained XGBoost model into Shapley Value implementation [31]. Since the column for the missing feature is removed, while we are still passing the original train set with full feature preset, the code will prompt "shape not match error"[5].

In our dataset there are several features(i.e., events) that rarely happen, when doing the train test split, some features/events may completely not appear in the train set. Thus the reduction of shape in the column length and triggering the shape not match error. Considering this, we decide to switch to the Gradient Boosting wrapper in sci-kit learn library [46]. Also, due to the same reason, the final Treeshap analysis is carried out on the train set, not the original feature matrix which includes all lots.

[5]Bug discussion on Shap implementation author's Github: `https://github.com/slundberg/shap/issues/580`

7.8.2 Random forest

The decision tree may suffer the problem of huge variance if it grows very deep, as the tree tends to be very specific that a small change in the leave might completely change the final result. A common way to reduce the variance is by "bagging", to take many training sets from the population, build a separate prediction model using each training set and average the resulting predictions as our final prediction result (illustrated in Figure 7.13). This brings us to the random forest.

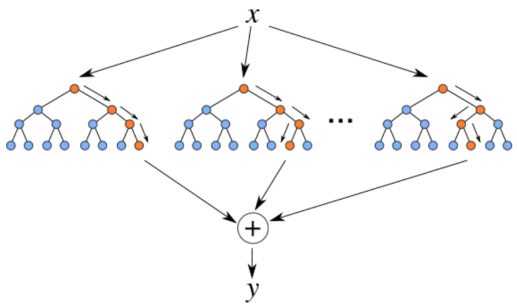


Figure 7.13: Random forest principle [27]
The results of all trees can be calculated in parallel and are added together to make the final prediction result.

Compared with simple bagging(or Bootstrap aggregation) methods to let the trees grow "naturally", random forest goes one step further to apply small tweaks to de-correlate the trees when they are growing. More specifically, when building these decision trees in the random forest, one can limit a tree/a split to use a fixed number or ratio of randomly sampled predictors [54].

7.8.3 Evaluation of regression models

We use two measurements to measure the performance of the Gradient Boosting and Random Forest prediction model, *root mean squared error* (*RMSE*) and *mean absolute error* (*MAE*) [46].

$$RMSE = \sqrt{\sum (predicted\ LOH - real\ LOH)^2} \tag{7.4}$$

$$MAE = \sum |predicted\ LOH - real\ LOH| \tag{7.5}$$

7.8.4 Baseline linear regression model

Except for the above two tree-based methods(as a gadget for time influence calculation), to assess the model performance we also choose simple linear regression with extra limitations to force the coefficient to be positive. Under the initial assumption that we can directly calculate the "fixed time influence"(discussed in section 5.1) of events and patterns, we can use coefficients to represent each event/pattern's "fixed time influence"(if no serious correlation), which can be either no time influence or positive time.

7.9 Pattern mining

Except for the single event, we are also interested in finding the interesting patterns (that either themselves are taking a long time or are the reasons leading to the delay in LOH). The considerations behind this are:

1) Sometimes the finding of a single event does not show enough evidence to lead to a delay in LOH. We want to find other related events to strengthen the diagnosis guess;

2) We want to explore the connections between events to find events connections and influences that were unknown before, or to conclude a higher level root cause. Thus mining sequential patterns could help domain experts to diagnose the machine's running conditions and potential problems better.

We carry out experiments on both itemset pattern mining and sequential pattern mining. For both of them, we start to approach it from the frequency perspective. There are mainly three reasons behind it:

1) the intuition that a pattern happens rarely tends to have less impact on the LOH in general(especially from a global perspective).

2) Development & Engineering experts prioritize the frequent problems over problems that rarely happen, as firstly solving the problems that happen frequently will bring bigger benefits.

3) sequence mining is a computing power-consuming and time-consuming task, while the expectation of this project is to design an applicable tool that can solve problems fast so we need to take the time into consideration. Frequency is a common attribute that has been researched and optimized intensively in sequence mining in academia, which provides us with plenty of tools that is realistic to solve problems for a short time.

7.9.1 Itemset pattern mining

While it is reasonable to guess sequential patterns(keeping the event order information) can provide us with more accurate and useful insights, we decide to start from itemset mining (which ignores the order information). There are several considerations:

1) there is a potentially strong interaction between event codes. Our final time influence evaluation results getting from *Shapley Value* calculation are built on the premise that the first prediction regression model gets a good result. So we hope by finding events frequently happening together and adding them to the occurrence feature matrix could improve the model prediction performance, thus improving the final time influence quantification result.

2) Since our interested sequence mining is a time-consuming task, the searching space is too huge and it is easy to get pattern explosion[6]. Thus we first apply the itemset mining as a test to pre-estimate the computing time and parameters and see if we can further proceed with the more time-consuming task sequence mining. An assumption is that if the frequent itemset pattern we find already cannot provide any insight, e.g., cannot find the itemset pattern that has a big time influence on the LOH, then the sequence mining also will not work.

Here we define the sub-itemset patterns we are going to find:

[6]The last intern also tried frequent itemset mining (association rule mining), however, at last, it was not possible to run an experiment on the whole dataset, which is also one of the reasons we decide to start with the relatively easy task instead of pushing things too hard

Itemset pattern candidate:
Search for event sub-itemset pattern q, such that $Minsup(q, Q) > u$, where u is user-defined parameters.

We use a frequent itemset mining algorithm *Krimp* to find itemset patterns. The key problem Krimp solves is to find the best set of patterns that compress the dataset best [45]. It innovatively applies the MDL(Minimum description length) principle [7] to achieve the goal, by learning through a data compression perspective. The advantage of this method compared with other itemset mining algorithms is that this one is designed under the MDL principle, so the final returned patterns are already selected patterns that contain the most information from the original event sequence database. This could help us filter out a potentially large amount of overlapping patterns that do not provide information of good quality to machine learning models and human analysis. We thus update our search from the above:

Itemset pattern candidate(update):
Search for event sub-itemset pattern q, such that $Minsup(q, Q) > u$, where u is user-defined parameters. Only keep (sub)itemset patterns provide smaller compressed size when they are added to the current itemset pattern set under defined encoding system [45].

$$\mathcal{T}(Q, E, u) = \{q \in Q \mid Minsup(q, Q) > u\} \textbf{ AND } \textit{keep only q that compresses more.} \tag{7.6}$$

The item patterns found by Krimp are to be further selected using *naive time influence* measurement discussed in section 7.10. Together with the below sequential pattern mining technique, they are parallel to the first stage in our two-stage pattern mining method illustrated in the pipeline.

7.9.2 Sequential pattern mining

To the best of our knowledge, currently, there is no method for directly mining and analyzing item sequential patterns with the numerical target. So we design a two-stage method to firstly find some potential candidates as shown in pipeline Figure 7.1. We start to approach it from the perspective of frequency, like in paper [13], [14] and [15]. Although either they are focusing on a classification task or the patterns they consider does not have order-keeping requirements. Secondly, we let it pass a selection procedure by the naive average time difference. The detailed selection measurement called *naive time influence* is discussed in section 7.10.

With the reasons discussed at the beginning of this chapter, we start to approach it from the frequency perspective. Here we define the sequential patterns we are going to find:

Sequential pattern candidate:
Search for all event sub-sequence pattern p, such that $Minsup(p, P) > u$, where u is user-defined parameters.

Domain experts also mention it might already be a nice try to find a sub-sequence

[7] It believes the best model should be able to use the shortest description to describe the data and is sometimes introduced as mathematical applications of Occam's razor[43].

$$\mathcal{T}(P, E, u) = \{p \in P \mid Minsup(p, P) > u\} \tag{7.7}$$

with small gaps or even a continuous sub-sequence, so a variation of the second sub-problem is:

Sequential pattern candidate(variation):
Search for all event sub-sequence pattern p, such that $Minsup(p, P) > u$, $gap < v$, where u, v are all user-defined parameters.

$$\mathcal{T}(P, E, u, v) = \{p \in P \mid Minsup(p, P) > u,\ gap < v\} \tag{7.8}$$

The gap limitation limits the gaps between two events belonging to a pattern. By setting a low gap value threshold v, the algorithms can work faster, which also enables us to lower the minsup threshold in finding candidate sequences with limited computing power and memory.

A problem faced by sequence mining is a huge amount of patterns may be returned. It is difficult for humans to analyze a huge amount of information. Also, many of them may be overlapping, thus does not provide better information for the machine learning model to digest. Based on these, we can get help from the research result on the variations of the original sequential pattern mining result, to mine a concise representation of the complete pattern results (discussed in section 6.4.3).

With the considerations above, we choose four types of sequential pattern mining algorithms: CM-SPADE [21](find all sequential patterns that satisfy the above searching requirement, claimed to be the fastest algorithm [21] in dealing with this task), CM-ClaSP [21] (find all closed sequential patterns), VMSP [24] (find all maximal sequential patterns) and VGEN [25] (find all generator sequential patterns).

Among them, VMSP and VGEN are two algorithms that enable us to set the gap limitation. By setting the max gap, we can specify if gaps are allowed or how many of the gaps are allowed in sequential patterns. If the max gap is set to N, a gap of N-1 event is allowed between two consecutive events of a pattern. For example, if the max gap is set to 1, no gap is allowed (i.e. each consecutive event of the candidate pattern must appear consecutively in a sequence).

Like the above itemset mining, sequential pattern mining is paralleled as the first stage in our two-stage pattern candidate mining method.

7.10 Naive time influence

After the first steps in our proposed two-stage methods to find pattern candidates, we also need to build pattern candidates' connection with LOH. We do this by calculating the difference between the average LOH time of lots with this candidate and without it in the dataset. We call it the naive time difference and denote it as ϕ.

So given any sequential pattern p, we can divide all lots (or all sequences s in P) into two subsets P_0, P_1 given the appearance of p, where $P_1 = \{s \in P, p \sqsubseteq s\}$, $P_0 = P \setminus P_1$. Then we calculate the naive time influence ϕ_p as the difference between the average LOH time of all lots in P_0 and P_1:

$$\phi_p \;=\; avgloh(P_1) - avgloh(P_0) \tag{7.9}$$

Similar calculation applied to (sub-)itemset patterns. Given any (sub-)itemset pattern q, we can divide all lots (or all itemset I in Q) into two subsets Q_0, Q_1 given the appearance of q, where $P_1 = \{I \in Q, q \subseteq I\}$, $Q_0 = Q \setminus P_1$. Then we calculate the naive time influence ϕ_q as the difference between the average LOH time of all lots in Q_0 and Q_1:

$$\phi_q \;=\; avgloh(Q_1) - avgloh(Q_0) \tag{7.10}$$

After calculating the *naive time influence* of all patterns, we select the patterns with ϕ rank on the top with the quantile threshold defined by the user.

Selection:
Search for event candidate patterns $x_i \in X$ (x_i could either be sequential pattern or itemset pattern, and X is the collection of all patterns) found either by itemset patterning or sequential pattern mining in the first-stage, such that $\phi_{x_i} > w\_quantile(\phi_X)$, where w is user-defined parameter.

$w \in (0, 1)$ and $w\_quantile$ means the w-th quantile value of all candidate patterns' LOH value.

Naive time influence selection is the second stage in our two-stage pattern candidate mining pipeline. After the selection, we add the final pattern candidates as new features into the original feature matrix(which contains only single event features) and continue with the rest of the analysis in our pipeline to get the final quantified time influence evaluation result.

7.11 Result evaluation

To check by model comparison
Since we have limited knowledge of the event codes, it makes our evaluation of results a bit difficult. When lacking existing knowledge, we can compare results generated by different algorithms to see the similarity of results. If the results are generated from different algorithms in achieving the same goal, then we can have more confidence the result is reasonable, instead of random calculations.

Here we evaluate the results we get by comparing the similarity of the results getting from different underlying machine learning algorithms when calculating Shapley Value. we define the similarity measurement as:

$$similarity\ top\ N = \frac{|A \cap B|}{N} \tag{7.11}$$

where A and B are lists of events/patterns found by both methods ranked on top N with regard to the time influence order and N a value to decide *topN* is set by the user. We evaluate the similarity of events/patterns found by two methods respectively and ranked among either top30, top20, and top10 (N= 30,20,10 respectively).

To check with domain knwoledge
We also check the results from the domain perspective and with the best of our current knowledge. To know if the top events and patterns we find are useful in a

practical situation, we ask domain experts to compare the results with their domain experience and give some empirical comments.

For the simplicity of analysis checking, in this project, we check the results of events and patterns ranked at the top30 with regard to their time influence in descending order. But one can easily set N in $topN$ to the value they want.

Chapter 8

Experiment

In this chapter we carry out the empirical evaluation of the solution we proposed in section 7 in the following steps:

Question 1 ***pattern***: Can we use the two-stage method we proposed to mine high-quality pattern candidates?

Specifically, we assess the mined patterns in the following aspects:

- Is it meaningful in the industrial context?
- Does it tend to be good candidates that have a large time influence on LOH?

Question 2 ***prediction***: Can we use pattern candidates we mined together with single events to predict LOH time? Does adding pattern candidates improve the performance of the prediction model? Which pattern mining algorithms produce patterns that improve the prediction most?

The evaluation of prediction performance here is based on the prediction performance on the test set we reserved by splitting the dataset (details in section 8.2.1).

Question 3 ***quantification***: What are the Shapley Values for each pattern candidate/individual event? Can use Shapley Value(together with our predictive model with feature matrices based on our pattern mining results) as time influence quantification measurement to find events and patterns that play a big role in LOH? How would the domain experts interpret the results, i.e., is it in accordance with their prior knowledge of (some) known patterns' time cost?

Specifically, for events or patterns that play a big role:

- Is itself taking a long time?
- Is it the reason or signal to lead to delay in LOH?

NB: For IP(intellectual property) reasons we anonymize all the event code names by mapping each unique event code to a unique integer. While we keep a dictionary (anonymized reference table) that can map anonymized integer numbers back to code names when needed.

8.1 Experiment on Question 1

We use our proposed two-stage method to mine the sequential patterns. We start with the mining itemset pattern using *Krimp* as a trial. After checking the feasibility we pursue our real goal to mine sequential patterns using four different sequential pattern mining methods: *CM-SPADE*, *CM-CLaSP*, *VMSP* and *VGEN*.

8.1.1 Set-up

We hereby give the experiment set-ups for Krimp and four sequence mining methods respectively.

Krimp set-up
In the experiment, we use the Krimp implementation from paper [44]. To run Krimp, the most important parameter is the *minsup* parameter. Less minsup means more possible event/pattern candidates, thus greatly increasing the search space. Conversely, if we raise the value of minsup value, then the computing time decreases significantly.

Algorithm	Type	minsup(absolute)
Krimp	itemset	100

Table 8.1: Itemset pattern mining set up on minsup threshold
The threshold 100 is chosen due to the consideration of memory and time.

The minsup level we choose for running Krimp is 100 as shown in Table 8.1. Here the minsup is the absolute minimum support value required to select itemset patterns (differentiate the minsup ratio adopted by sequence mining algorithms below, which is a ratio ranging between 0$\sim$ 1). The pruning strategy is set as post-pruning[45]. [1]

One consideration is to set minsup threshold to 30, as from domain knowledge there are some regular events happening once per day(and hence 30 per month, which is the time range of our dataset), like some conventional cleaning tasks. However, itemset mining is both a time and memory-consuming task. We are not able to run this setting under normal office laptop configuration. We also tried setting it to 60, and it takes *Krimp* 2116s to run under the configuration above, which is too long by tool design expectation on time limitation from the domain experts. Thus we finally set the minsup to 100 (around 0.02 of the total number of lots). Under this setting, the Krimp can get the result instantly.

The domain experts also mention there could be regular events that happen every hour. In this case, we can set the minsup threshold higher, which will make the algorithm finish getting results even faster.

Sequence mining set-up
After checking the feasibility of itemset mining, we further carry out experiments

[1] Another thing to notice is the application requires to convert item numbers(for us is the anonymized integer number of event codes) to default fic numbering(by setting 'dbOutTranslateFw' = True). We originally tried to set this as False, as we want to reserve the original anonymized code number. However, after trying so, the algorithm cannot work normally nor deliver results. We guess it is because, under the new fic-numbering fashion, the search can be done more efficiently as the fic-numbering already re-sort the event codes according to their frequency.

on sequence mining. We adopt 4 sequence mining algorithms – *CM-SPADE*, *CM-CLaSP*, *VMSP* and *VGEN*. In the experiment, we choose the implementation of these four algorithms from an Open-Source Data Mining Library SPMF [49].

The parameter setting for sequential pattern mining is listed in Table 8.2. One thing to notice is that in sequential pattern mining, the minsup is a ratio (so *support* divided by the total number of lots), instead of the absolute number of the *support* of a pattern as in Krimp.

Algorithm	Sequence type	minsup(ratio)	gap
CM-SPADE	frequent	0.04	\
CM-ClaSP	frequent closed	0.04	\
VMSP	frequent maximal	0.006	1
VGEN	frequent generator	0.006	1

Table 8.2: Sequence pattern mining set up on minsup and gap threshold
The thresholds are chosen due to the consideration of memory and time. VMSP and VGEN are two algorithms that enable us to set the gap limitation, which enables us to lower the minsup threshold. We set it to 1 to limit the algorithm search patterns composed of only consecutive events.

Besides, we also set the parameter "Show sequence ids" as True for all four methods, to store the index of the lot that contains the sequential patterns we found. This saves us a lot of work to update the feature matrix than when dealing with *Krimp*, which requires us to scan datasets several times to find the lots' index where these lots contain a certain pattern.

Just like in Krimp, we want to filter out regular event patterns(like conventional fixed cleaning procedures) happening every day. Under the consideration that there are usually 30 days in a month, we try to push the threshold of minsup to $30/5000 \approx 0.006$. However, the minsup 0.04 is the lowest value that we are able to set for CM-SPADE and CM-ClaSP, to carry out the analysis on the current office computer. Setting this value lower, the computer will report out of memory and the SPMF [49] tool will break. For all 4 algorithms, we push the computing power to the limit under the current laptop configuration. The reason we are able to set a lower minsup threshold in the latter two algorithms (VMSP and VGEN) is due to the extra gap constraints we get for these 2 algorithms. Limiting the gap enables us to do a more intensive search on patterns with lower minsup 0.006.

Naive time influence threshold

To further select the pattern candidates we get by pattern mining algorithms, we use naive time influence to pick the top candidates. The threshold of naive time influence quantile we set is shown in Table 8.3 in the "top" column. Meanwhile, we also want to know if the pattern candidates we find are reasonable. Due to the complexity of human manually checking, we raise the threshold to pick a smaller amount of the pattern candidates ranked highest with regard to its naive time influence value. The threshold set for human checking is listed in the "highest ranked" column in Table 8.3.

The thresholds for sequence patterns are higher than itemset patterns. This is

because we get more sequence pattern candidates in the first stage than itemset pattern candidates. Thus by rising the threshold for sequence patterns, we reduce the sequential candidates to be added to the original feature matrix.

	top	highest ranked
Itemset	0.5	0.9
Sequence	0.8	0.98

Table 8.3: Naive time influence threshold
Threshold used to select top patterns as final pattern candidates to add to occurrence matrix, and threshold to get the highest ranked patterns for domain analysis.

8.1.2 Result and Discussion: First-stage

Under $minsup = 100$ constraint, Krimp helps us find 23 itemset patterns. The basic statistics of sequential pattern mining result is in Table 8.4. Under the experiment parameter settings above, the pattern mining procedures for both itemset and sequential patterns can be done instantly.

Algorithm	# patterns found	# non-single event pattern
Krimp	246	23
CM-SPADE	776	763
CM-ClaSP	408	396
VMSP	83	75
VGEN	403	356

Table 8.4: Sequence mining result statistics
The number of sequential patterns found is in the order of CM-SPADE > CM-ClaSP > VGEN > VMSP. Sequential pattern mining algorithms return more patterns than the itemset mining algorithm.

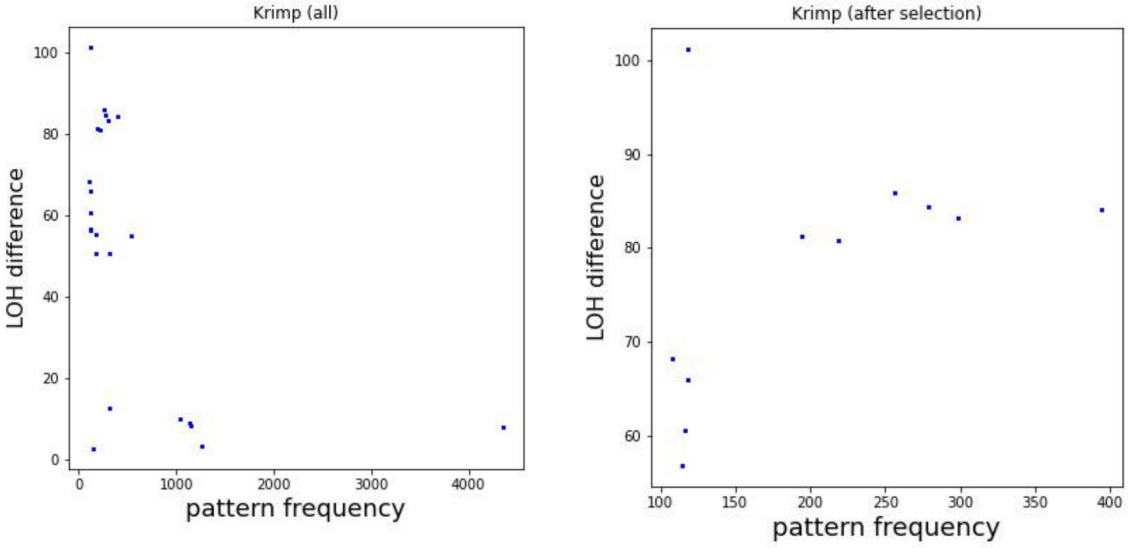
From the statistic result in Table 8.4 can see the number of patterns found by the first three sequential algorithms follows the order: CM-SPADE > CM-ClaSP > VMSP, which is in line with our expectation based on the definition of the type of patterns these algorithms are aiming to find. Sequential pattern mining algorithms return more patterns than the itemset mining algorithm, this is mainly due to the minsup threshold we set in the itemset pattern mining algorithm being more strict than that in sequential pattern mining.

The algorithms run unexpectedly fast. However, whenever we try to lower the minsup threshold a very little bit, even with the decision to sacrifice the length of the time, the memory does not allow it. So we can see the common problem in sequence mining, that with looser constraints, the amount of calculations increases explosively.

8.1.3 Result and Discussion: Second-stage for itemset patterns

After getting the frequent itemsets using Krimp, we calculate the average LOH difference between lots with a certain pattern and without it. We carry out an extra pattern selection procedure based on the event's naive average time influence ϕ.

We visualize the result in the scatter plot below (Figure 8.1). The x-axis is the frequency of a pattern, and the y-axis is the naive time influence result of a pattern. From Figure 8.1a we can see Krimp does find the patterns that tend to have a big time influence on the LOH in terms of naive average time influence ϕ.



(a) all

(b) after selection

Figure 8.1: Scatter plot of LOH difference between lots with and without Krimp itemset pattern
We select patterns with bigger LOH difference(*naive time influence*) as final itemset pattern candidates.

A: Top candidates selection result

Based on this graph we can let the user set the threshold to select the patterns they want to check manually. We set the threshold using the quantiles of these patterns' naive time influence ϕ.

Here we set the threshold to 0.5, then we calculate the 0.5 quantiles of these patterns' ϕ. Only the itemset patterns p with $\phi_p > 0.5$ quantile of all patterns' ϕ (all patterns means the patterns getting from stage 1) are regarded as our patterns of interest and to be picked. All the others are abandoned. This threshold is lower compared with the threshold used to do a domain check. Here we can let more patterns pass, as we don't need to do analysis manually and can take the advantage of algorithm's power.

The number of final itemset patterns after selection is 11, as shown in Table 8.5. Then we add these patterns into the original occurrence feature matrix (which only

contains single events as features in section 7.6.3) and get the new feature matrix in the shape of (4903, 236).

	Before	After selection
Krimp	23	11
CM-SPADE	763	152
CM-ClaSP	396	79
VMSP	75	15
VGEN	356	71

Table 8.5: Second stage pattern selection result statistics
The final number of sequential patterns follows the order of CM-SPADE > CM-ClaSP > VGEN > VMSP. The decrease of sequential patterns is more significant than of itemset patterns due to the looser selection threshold we choose for itemset patterns.

B: Highest ranked candidates manually checking

We set the threshold to a high value of 0.9, to get a small number of itemset patterns with the biggest ϕ. We ask domain experts to check if these highest-ranked patterns make any sense. We set this high value because we don't want to get overwhelmed with too many pattern candidates to check manually.

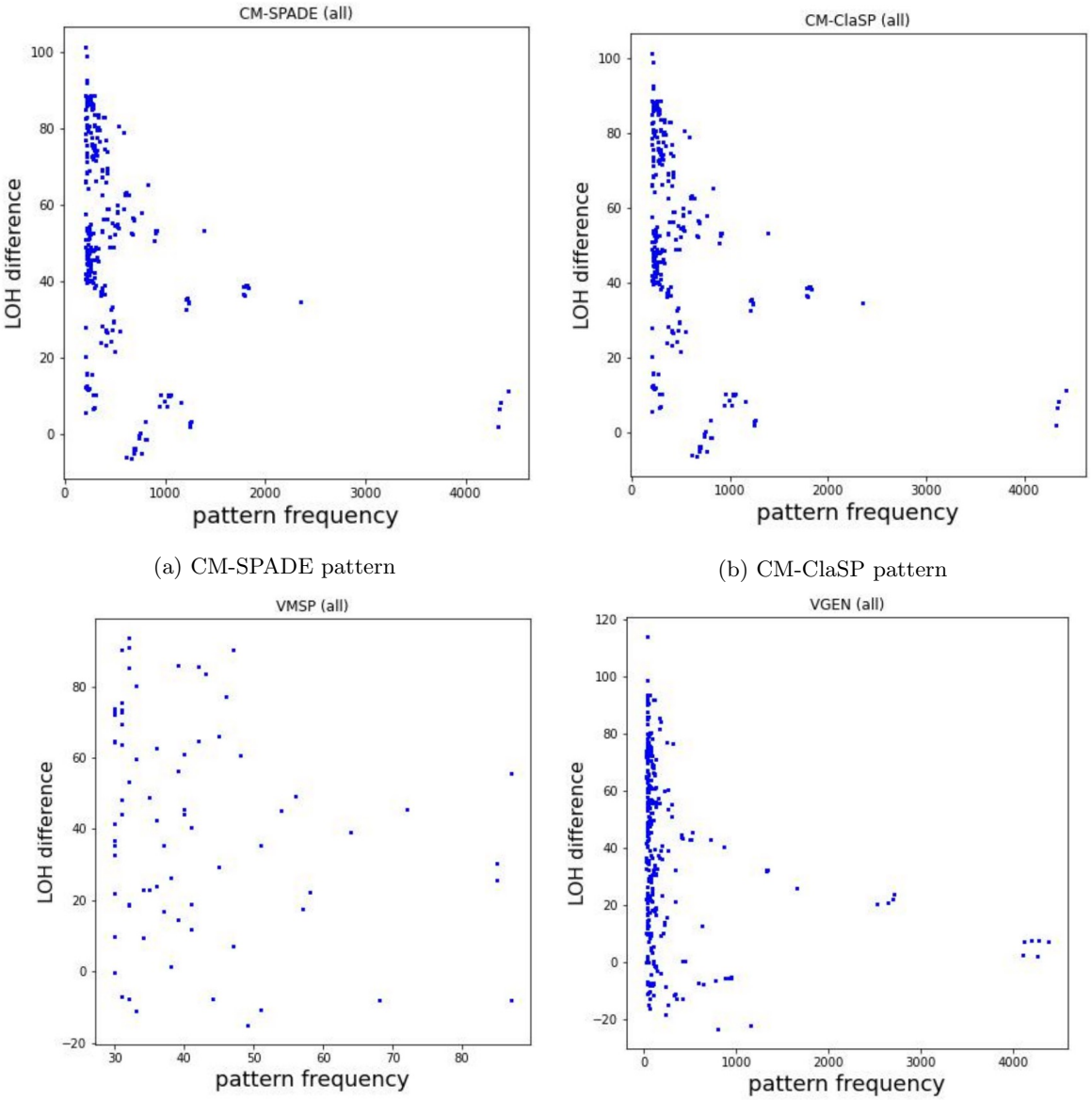
We notice the patterns found by four algorithms are composed of a small event set. We randomly pick two of them to ask the domain experts to check. One is [5, 18, 17, 24, 57, 56, 52], with ϕ big value 85.92s. The domain experts find they are indeed the events that often happen together to start the lot, although 17 is not important at all in analyzing the LOH. Another candidate we pick is [5, 18, 17, 16, 31, 21] with even bigger ϕ value 101.28s. This itemset pattern also tends to have an impact on LOH from the domain perspective, as it indicates the previous lot is not finished and no new wafer will be exposed.

8.1.4 Result and Discussion: Second-stage for sequential patterns

After getting the patterns using these four algorithms, we first need to remove the patterns composed of only a single event. We assume the pattern is composed of no less than two events, as if we add them to the feature matrix, they will become duplicated features with original ones.

Then like what we did when dealing with Krimp, we calculate the naive time influence (LOH difference in figures and visualize the results for each of the algorithms in the scatter plots below (Figure 8.2). The x-axis is the frequency of a pattern, and the y-axis is the *naive time influence*(named as LOH difference in scatter plot) result of a pattern.

A: Top candidates selection result



(a) CM-SPADE pattern

(b) CM-ClaSP pattern

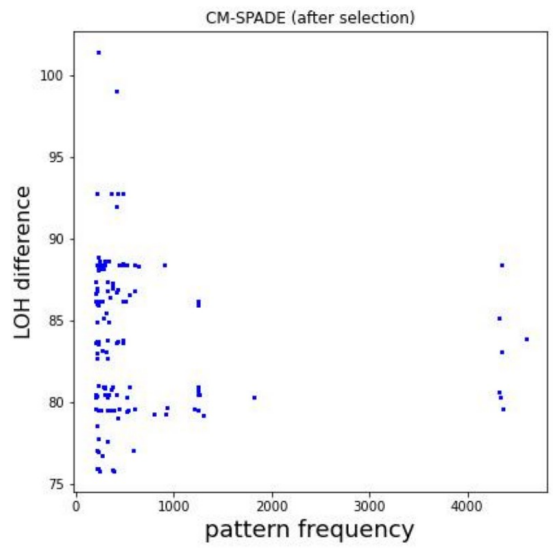
(c) VMSP pattern

(d) VGEN pattern

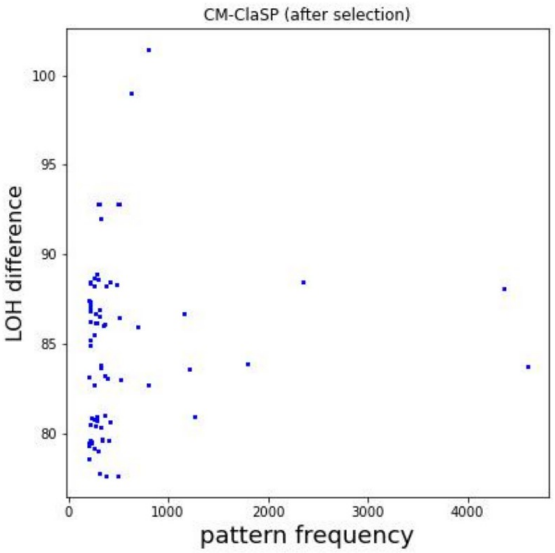
Figure 8.2: Scatter plot of LOH difference between lots with and without sub-sequence pattern
The scatter plot of CM-SPADE looks similar to that of CM-ClaSP.

The quantiles threshold with regard to the naive time influence we choose for sequential patterns is 0.8, higher than that for itemset patterns selection. The reason to set the higher threshold here is that we get more sequential patterns detected than itemset patterns detected (mainly due to the looser minsup threshold we set for sequential pattern mining algorithms in stage 1), so we need to further shrink the number of patterns.

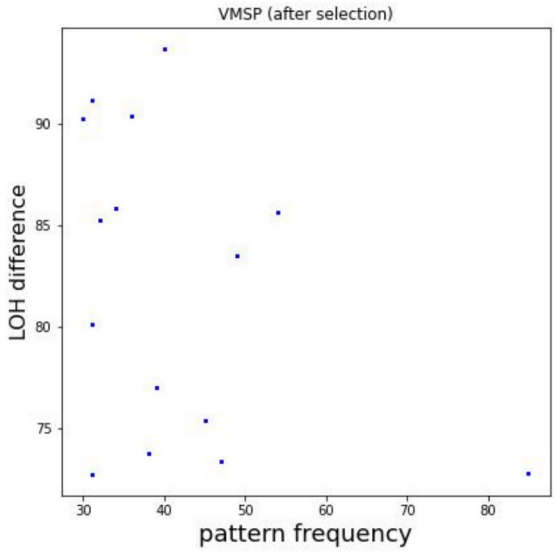
The final number of sequential patterns follows the order of CM-SPADE > CM-ClaSP > VGEN > VMSP. The decrease of the itemset patterns is less significant



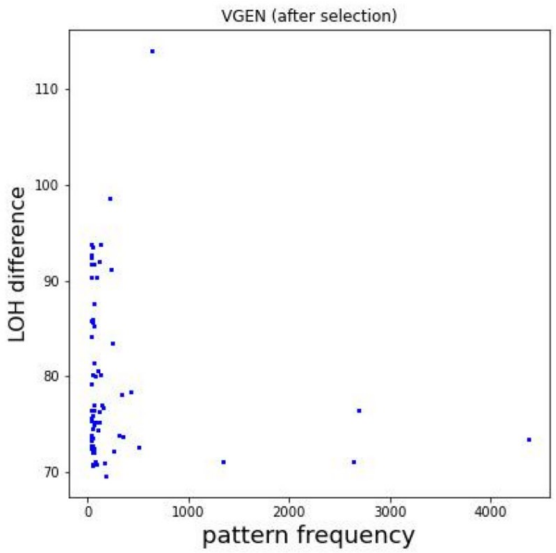
(a) CM-SPADE pattern



(b) CM-ClaSP pattern



(c) VMSP pattern



(d) VGEN pattern

Figure 8.3: Scatter plot of LOH difference between lots with and without a sub-sequence pattern (after selection)
We select patterns with bigger LOH difference(*naive time influence*) as final sequential pattern candidates.

than that of sequential patterns due to the lower selection threshold we choose for selection. From Figure 8.2a, 8.2b we can see the patterns found by CM-SPADE and CM-ClaSP are pretty similar, while the patterns we find using VMSP and VGEN are quite different (Figure 8.2c, 8.2d).

Due to the extra gap constraints, we get more patterns with lower frequency. In three cases among the four (CM-SPADE, CM-ClaSP, and VGEN), they all find patterns with very large frequency, but not VMSP. Considering the definition of

VMSP, we reason that this indicates there are some actions does not go fluently in the production process and require a lot of redoes (pattern composed of repetitive events).

The number of final sequential patterns after selection for four sequential pattern mining algorithms are 152, 79, 15, and 71, as shown in Table 8.5. We then add these patterns into created original occurrence feature matrix and get the new feature matrix in the shape of (4903, 377), (4903, 304), (4903, 240), (4903, 296) respectively for CM-SPADE, CM-ClaSP, VMSP, and VGEN. The predicted result using the new feature matrix is shown in Table 8.6. The events after selection with a threshold on their ϕ value are shown in Figure 8.3. After the selection, the patterns are all with ϕ bigger or at least very close to 70second. We increase the probability to find good pattern candidates to test if they get a bigger time influence ψ, with the help of ϕ.

B: Highest ranked candidates manually checking

We increase the quantile threshold to a very high value of 0.98. So only candidates that satisfy minsup constraints from the frequency perspective (and gap constants for VMSP and VGEN), and also with naive time influence ϕ ranks among the top 0.02 are to be manually inspected.

Three things we directly notice when we look at the sequence patterns top candidate results got by these 4 algorithms. The first thing we find is that the sequence patterns found by each algorithm still have very high similarities, for example [31, 5, 17, 5, 18], [31, 5, 17, 5, 5, 18] and [31, 5, 17, 17, 5, 18] found by CM-SPADE as highest ranked candidates. So in the future, we can further come up with a method to combine/shrink the sequence patterns we get at present. We can find a method to define the event sub-sequences similarities to do the post-pruning like in [14]. So that we only add the subsequential patterns after selection into the feature matrix for further analysis.

Secondly, we also see directly that the patterns found by VGEN – the generator patterns, show more variety than the patterns found by other algorithms. Also from the theoretical angle, generator patterns can be argued to be more preferable than other types of patterns according to the MDL (Minimum Description Length) principle [55]. As they are the smallest subsequences that characterize a group of sequences in a sequence database [6]. With this, we are more likely to avoid finding patterns overlapping with each other.

Thirdly, all the highest ranked candidates found by four sequential pattern mining algorithms are having similar *naive average time influence* ϕ, around 90, with a standard deviation of 2. Though the highest ranked candidates found by four algorithms differ a lot.

CM-SPADE tends to find a sequential pattern with different combinations composed of a small set {31,5,17}. CM-CLASP tends to find a sub-sequence pattern with many events 5. While the highest ranked candidate of VMSP and VGEN looks neater, with many unique events.

The highest ranked candidate [21, 31], [31, 5, 17, 18] found by CM-SPADE is meaningful, as they together mean switching off the laser, thus might lead to a gap. While we see event 17 happen frequently with other meaningful events, but itself is not interesting. It is more of subscribing action that bounded with other activities but does not has any effect with regard to machine production process time.

8.1.5 Conclusion of experiment on Q1

Thus we conclude that the two-stage pattern mining methods can help us to find reasonable candidates. The patterns are meaningful in the industrial context and tend to have a big influence on LOH that either itself is taking a long time, or is the reason leading to delay in LOH.

8.2 Experiment on Question 2

As mentioned in the Method chapter, the time influence quantification (sub)pipeline contains two steps, firstly we need to use a regression model to predict LOH. The later Shapley Value time influence calculation is built on the previous prediction model. It is intuitive that the better the first prediction model, the better the second Shapley Value calculation results. In this section, we do a grid search to find the best parameters for prediction results and find the most reasonable set-ups.

8.2.1 Set-up

With several considerations theoretically and practically discussed in the Method chapter, we use the implementation of *Random Forest* and *GradientBoosting* in *Scikit-learn* library [46], a free software machine learning library for the Python programming language. And for the *Shapley Value* calculation we use the Treeshap implementation from Su-In Lee's lab and Microsoft Research [30].

Train test split
Before the experiment, we split the dataset into train set and test set with test set size = 0.33 and random state 42.

Baseline
The first baseline we choose is the global average LOH in train set as the prediction result for every lot, and calculate the RMSE and MAE as the baseline. We choose this baseline method to simplify the comparison, as the mean prediction result is the same for all twelve different settings (six feature matrices and two prediction regression methods). Mean prediction is the best choice when we don't have extra information. The global average LOH value in the train set is 40.33. The RMSE and MAE are 36.62 and 26.21, which are listed in Table 8.6.

The second baseline model we choose is Linear regression. We limit the coefficient to be positive as if we want to interpret the result, the time influence(in the current context is comparing with the intercept) of an event will probably be either none or positive. The experiment is denoted as "positive lg" with results listed in Table 8.6.

Parameter tuning
Since the Shapley Value is built on the prediction model, we need to spend some effort to make sure the prediction model is as good as possible, by searching for the best parameter to make the prediction more accurate. To find the best parameter, we performed a grid search. We use the GridsearchCV from Sci-kit learn [46], to divide the train set into 3 folds and perform the cross-validation. The grid search is to do the exact search on all combinations of the parameters and inspect the prediction performance on the test set.

Due to the limitation of the current computing power we limit the number of parameters we searched. We try not to let the tree models grow too deep(by limiting the maximal depth of the trees), or too wide(by limiting the number of trees(or so-called estimators in the sci-kit learn implementation)).

Meanwhile, as mentioned in the Method chapter, the computation complexity for Treeshap is $O(TLD^2)$, where T is the number of trees, L is the maximum number of leaves in any tree, and D is the maximal depth of any tree [32]. Increasing the depth and number of trees will not only lead to a longer time in prediction model training but also increase the time taken to calculate the Shapley Value.

The parameters we tried in *Gradient Boosting* are:

- number of trees: 50, 100, 200, 300, 400, 500
- learning rate: 0.5, 0.1, 0.05,0.01, 0.005
- max depth of the trees: 3,5,7,9

The parameters we tried in *Random Forest* are:

- number of trees: 5, 20, 100, 200, 400
- max depth of the trees: 4, 8, 16, 32, 64

After experiments with the above parameter tuning settings, we notice for both regression models, the biggest parameter value in the number of trees reached (for gradient boosting, it is using the original feature matrix plus VMSP and VGEN patterns; for the random forest model it is using original feature matrix plus either CM-SPADE, CM-ClaSP or VGEN patterns. The detailed result can be found in Table 8.6). While for max depth for both models and learning rate for gradient boosting model, the biggest parameter value does not reach. We thus carry out extra experiments on increasing the number of trees, while keeping other parameters the same as the above settings and naming the experiments as "gb-xl" and "rf-xl".

Extra parameter tuning
The extra parameters we tried for the number of trees parameter in both *Gradient Boosting* and *Random Forest* are:

- number of trees: 600, 800

Except for the parameters we are tuning, for reproducibility, we set the random state to 0 for both regression models training, set n_job = -1 for random forest training, and leave all the other parameters as default settings from the implementations in sci-kit learn [46].

8.2.2 Prediction Result

In Table 8.6 we present the best training results for both models with different feature matrices. The *Random Forest* denoted as "rf" and *Gradient Boosting* denoted as "gb". The extra experiments on the "number of trees" are denoted as "rf-xl" and "gb-xl". The data column in the table refers to the feature matrix we used. The original feature matrix with only single events as features. For the feature matrix after adding patterns, we name it "ori plus $<$ *pattern mining method* $>$". For example, "ori plus Krimp" is the original feature matrix adding itemset pattern candidates found by Krimp(after selection). The evaluation results using both RMSE and MAE are listed. The column Expected Shapely is the result getting in the experiment on question 3, we can skip it for now.

Data	Model	RMSE	MAE	Expected Shapley
baseline		36.62	26.21	\
original feature mtx	positive lg	15.82	10.55	\
	gb	15.81	9.99	40.33
	rf	16.39	10.25	40.39
	gb-xl	15.78	9.90	40.33
	rf-xl	16.46	9.96	40.30
ori plus Krimp	positive lg	15.85	10.57	\
	gb	15.84	10.02	40.33
	rf	16.48	10.34	40.39
	gb-xl	15.85	9.98	40.33
	rf-xl	16.51	10.33	40.30
ori plus CM-SPADE	positive lg	13.24	9.21	\
	gb	<u>12.18</u>	8.01	40.33
	rf	12.36	<u>7.83</u>	40.34
	gb-xl	12.31	8.16	40.33
	rf-xl	12.35	<u>7.82</u>	40.30
ori plus CM-ClaSP	positive lg	13.23	9.20	\
	gb	12.22	8.05	40.33
	rf	12.36	<u>7.83</u>	40.34
	gb-xl	12.18	8.10	40.33
	rf-xl	12.35	<u>7.82</u>	40.30
ori plus VMSP	positive lg	14.44	9.87	\
	gb	14.59	9.32	40.33
	rf	15.35	9.40	40.39
	gb-xl	14.56	9.27	40.33
	rf-xl	15.12	9.28	40.30
ori plus VGEN	positive lg	12.52	8.70	\
	gb	12.46	8.28	40.33
	rf	13.27	8.18	40.34
	gb-xl	12.45	8.24	40.33
	rf-xl	13.28	8.18	40.30

Table 8.6: Regression model prediction result with different underlying machine learning predictive models and with different feature matrices.

In Table 8.7, we list the best parameters under different settings. The training time is documented too. For Total time it is the summation of prediction model training time and Shapley Value calculation in latter pipelines.

Data	Set up	# trees	max depth	learning rate	Train time(s)	Total time(s)
original feature mtx	gb	300	3	0.05	351	360
	rf	20	8	\	31	44
	gb-xl	600	3	0.05	295	309
	rf-xl	600	16	\	66	1109
ori plus Krimp	gb	400	3	0.05	330	338
	rf	20	8	\	33	47
	gb-xl	600	3	0.05	292	302
	rf-xl	600	8	\	72	203
ori plus CM-SPADE	gb	100	5	0.1	596	606
	rf	400	32	\	59	1564
	gb-xl	800	3	0.05	542	553
	rf-xl	600	32	\	112	2433
ori plus CM-ClaSP	gb	100	5	0.1	452	463
	rf	400	32	\	50	1549
	gb-xl	800	5	0.01	478	507
	rf-xl	600	32	\	96	2425
ori plus VMSP	gb	500	3	0.05	328	337
	rf	20	16	\	38	79
	gb-xl	600	3	0.05	297	307
	rf-xl	600	16	\	76	1134
ori plus VGEN	gb	500	3	0.05	401	410
	rf	400	32	\	44	1799
	gb-xl	600	3	0.05	395	405
	rf-xl	600	64	\	97	2823

Table 8.7: The best parameters and time information of different regression models and feature matrices

8.2.3 Discussion on comparison with baseline

We can directly see that in all the experiments on different models and feature matrices, the RMSE and MAE decrease a lot from our baseline mean prediction. Thus the prediction model is learning the internal mechanism of the industrial process and getting more information than the baseline. While comparing with linear regression, we find in all case the random forest and gradient boosting outperforms it.

In all the experiments the RMSE is above 12 and MAE above 7. Meanwhile, during the training we notice the RMSE and MAE do not differ much in the parameter tuning process, this corresponds with the caveat in the dataset section that not all Event related to LOH is included in the Event dataset. Thus we may still lack the information to make the prediction better.

8.2.4 Discussion on feature matrix

By comparing the results of using the original feature matrix (with only single events as features) and using the original feature matrix plus the itemset patterns in the Table 8.6, we can see that there are no significant changes in the prediction performance

than when using only single event features, for both models, in terms of RMSE and MAE. Although the patterns we sent in the prediction model tend to have big time influence, with regard to naive time influence and domain basic checking.

We originally guess due to the interaction effect of events, adding the itemset pattern candidates could provide more information to the model to improve the prediction performance. From the little changes in the prediction result(even slightly worse), we think it is because both of the prediction regression models we choose have already learned the interaction effect relatively well.

Meanwhile, from Table 8.6 we find that, by adding sequential pattern candidates, the prediction result improves, compared with the results using only single events or by adding itemset patterns. The RMSE decreases from $15.9 \sim 16.5$ to $12.1 \sim 13.3$, MAE decreases from $9.9 \sim 10.3$ to $7.8 \sim 8.3$, by adding three of four sequential pattern candidates except for by adding VMSP patterns. For adding VMSP sequential pattern candidates, the performance gets much less improvement compared with adding CM-SPADE, CM-ClaSP and VGEN, with RMSE decreasing from $15.9 \sim 16.5$ to $14.5 \sim 15.4$ and MAE decreasing from $9.9 \sim 10.3$ to $9.2 \sim 9.4$.

This is a good indicator that adding (appropriate)sequential patterns provides useful information to help with prediction performance. We will discuss why VMSP is not a good choice among sequential pattern mining techniques in our application scenario in the later section when we are analyzing the final results of pattern time influence quantification.

8.2.5 Discussion on model

Comparing two underlying prediction models, gradient boosting performs better than random forest under all feature matrix set-ups, in terms of both RMSE evaluations. While for MAE, random forest outperforms the gradient boosting after we add CM-SPADE, CM-ClaSP, and VGEN sequential pattern candidates.

The best result achieved wrt MAE is using random forest with feature matrix plus (selected) CM-SPADE and (selected)CM-ClaSP pattern candidates; the best result achieved wrt RMSE is using gradient boosting with feature matrix plus (selected) CM-SPADE pattern candidates.

8.2.6 Discussion on parameter

We will first discuss experiment results excluding extra experiments on increasing the parameter the "number of trees" (the extra experiments are denoted as "gb-xl" and "rf-xl").

From Table 8.6 we see that the gradient boosting model tends to choose shallower max depth(3 or 5), with a learning rate of 0.05. While the random forest tends to choose a deeper max depth of for tree-growing (32). When we only send in single events as features, random forest tends to understand information fast, with a smaller number of trees(only 20) and a shallower max tree depth(8) needed. This also applies to after adding (selected) Krimp itemset pattern candidates and (selected) VMSP sequential patterns. While after adding the other three types of sequential pattern candidates, the number of trees needed and max depth increase.

We find this situation corresponds with the final prediction results wrt RMSE and MAE. As we discussed above in feature matrix settings, the prediction performance is better after we add the three types of sequential patterns CM-SPADE, CM-ClaSP, and VGEN. We think this is because the information contained by the original feature matrix is limited and adding itemset pattern and VMSP pattern does not provide

much extra useful information. Thus the model can easily digest the limited information with fewer trees and shallower depth. While adding (selected)CM-SPADE, CM-ClaSP and VGEN patterns do bring in extra useful information. So that the model needs more trees and depth to digest, and produce better prediction results.

In all cases, the best parameters chosen under the original feature matrix plus CM-SPADE or CM-ClaSP patterns are the same, this also corresponds with our finding from Figure 8.2 that the patterns found by CM-SPADE and CM-ClaSP look very similar.

extra parameter experiments on increasing the number of trees
As illustrated in the above experiment setup section, we notice for both regression models, the biggest parameter value in the number of trees reached, while others did not. So we experiment on increasing the number of trees parameter, while keeping other parameters to tune the same as the previous experiments. We denote experiments as "gb-xl" and "rf-xl"

We see from Table 8.6 and Table 8.7 that for the original max tree number reached experiment, the prediction result increases a little bit. Under gradient boosting: for original feature matrix plus (selected)VMSP, RMSE decreases from 14.59 to 14.56 and MAE decreases from 15.35 to 15.12; for original feature matrix plus (selected)VGEN patterns, RMSE decreases from 12.46 to 12.45 and MAE decreases from 8.28 to 8.24. Meanwhile, they don't further choose bigger value 800 and prefer 600.

Under random forest: for original feature matrix plus (selected)CM-SPADE, RMSE decreases from 12.36 to 12.35 and MAE decreases from 7.83 to 7.82; for original feature matrix plus (selected)CM-ClaSP patterns, the changes are exactly the same, for original feature matrix plus (selected)VGEN patterns, RMSE and MAE keeps more or less the same(only RMSE increases 0.01).

8.2.7 Discussion on time

From Table 8.7 we can see that in all the basic experiments (without including the extra parameter tuning on "number of trees" in the "xl" experiments), the time is limited to 30min (the total time). While in extra parameter tuning experiments on increasing the "number of trees" to 600 and 800, we can see that the time on tuning these two parameters (together with the same set of other parameters to tune), the training time is already the same as previously when training on several other parameters for gradient boosting, and for random forest the train time is already doubled. However, the increase is doable.

Meanwhile, we find that the increase in Shapley Value calculating time on the random forest is not doable (Shapley Value calculating time = total time - train time). The time for the best-predicted setup by adding CM-SPADE, CM-ClaSP, and VGEN patterns, the total time is more than 40min(previously finished within 30min). By tool design expectation we hope the tool to give fast analysis. Thus considering the very little improvement mentioned when discussing extra parameters above and the significant increase in time, we think increasing the number of trees does not worth it.

Generally, the random forest has an advantage in training time, but the total time (mainly because of Shapley Value calculation) is much longer than gradient boosting in the best set-ups by adding three types of sequential pattern candidates.

As discussed in the Method chapter, theoretically the computation complexity for Treeshap is $O(TLD^2)$, where T is the number of trees, L is the maximum number

of leaves(in our experiments we didn't tune it, we allow an unlimited number of leaf nodes)in any tree and D the maximal depth of any tree [32]. The Shapley value calculation time increasing grows at a quadratic rate with respect to D the max depth. This explains why the random forest is taking a much longer time than using gradient boosting as the underlying prediction model. As random forest tends to choose deeper max depth in our scenarios while gradient boosting tends to choose shallower.

8.2.8 Conclusion of experiment on Q2

Considering the results we conclude that adding (appropriate)sequential patterns can improve the performance of the prediction model. The difficulty in further decreasing the RMSE and MAE indicate that potentially there are other factors on which we lack information. This may be because of the events that are not included in the Event dataset as the caveat we mentioned in the Dataset chapter.

From the above discussions, with the considerations of performance and time, the best setup is using gradient boosting as the underlying prediction model and original feature matrix plus (selected)CM-SPADE sequential pattern candidates, with 100 trees, max depth 5, and learning rate 0.1. The experiment can be finished in around 10min.

8.3 Experiment on Question 3

For the experiment on question 3, we visualize the results in different types of graphs, with three levels of granularity and different points of focus – the decision plot, the summary plot, and the (global) bar plot[2] and draw some conclusions purely from the graph. Except for the quantification result, we also plot the heatmap plot, which enables us to observe the relationships between the features and between features and LOH value intuitively from another perspective.

Due to the limited knowledge of a large number of event codes, we will use the pre-defined similarity measure in the Method section to measure the validity of the model by model results comparison. We also invite different domain experts to check the results and their empirical reviews are collected.

8.3.1 Set-up

To calculate the Shapley Value efficiently, we use the Treeshap implementation from Slundberg [37]. Since we have the expectation to have the final time influence in the unit of time (the same unit as our target variable LOH time), we set the model output argument in TreeExplainer as "raw".

According to the author, the calculation of Shapley Value using Treeshap "rely on conditional expectations", so that we need to decide how to handle correlated event(and/or pattern) features. We set the feature perturbation to "interventional" in our experiment, as this breaks the dependencies between features according to the rules dictated by casual inference [56].

Here for simplicity of analysis, we will carry out the analysis on the top30 events or patterns that has the biggest global influence. After getting the time influence of events and patterns we sort the results in descending order so that we can find which top30 events or patterns have the biggest time influence on the scanner's LOH time. One thing to notice is the final judgment on whether the single event is known to have(or not have) impact or unknown is based on productivity team experts' opinion since this project is finished within the Productivity team in the Customer Support department. Meanwhile, in the discussion part, we also include the opinions from departments' experts, which sometimes may have slightly different from the perspective of the Productivity team.

[2]To align the shape of bar plot and the other two, we actually visualized 31 events or patterns in decision plot and summary plot

8.3.2 Time influence decision plot

The decision plot is one of the graphs with the smallest granularity. There are several advantages it equips: 1)It is able to show a large number of feature results densely and clearly; 2) ability to visualize multi-output for each prediction; 3) Explore feature effects for a range of feature values; 4) Identify typical prediction paths; 5) Identify outliers [31].

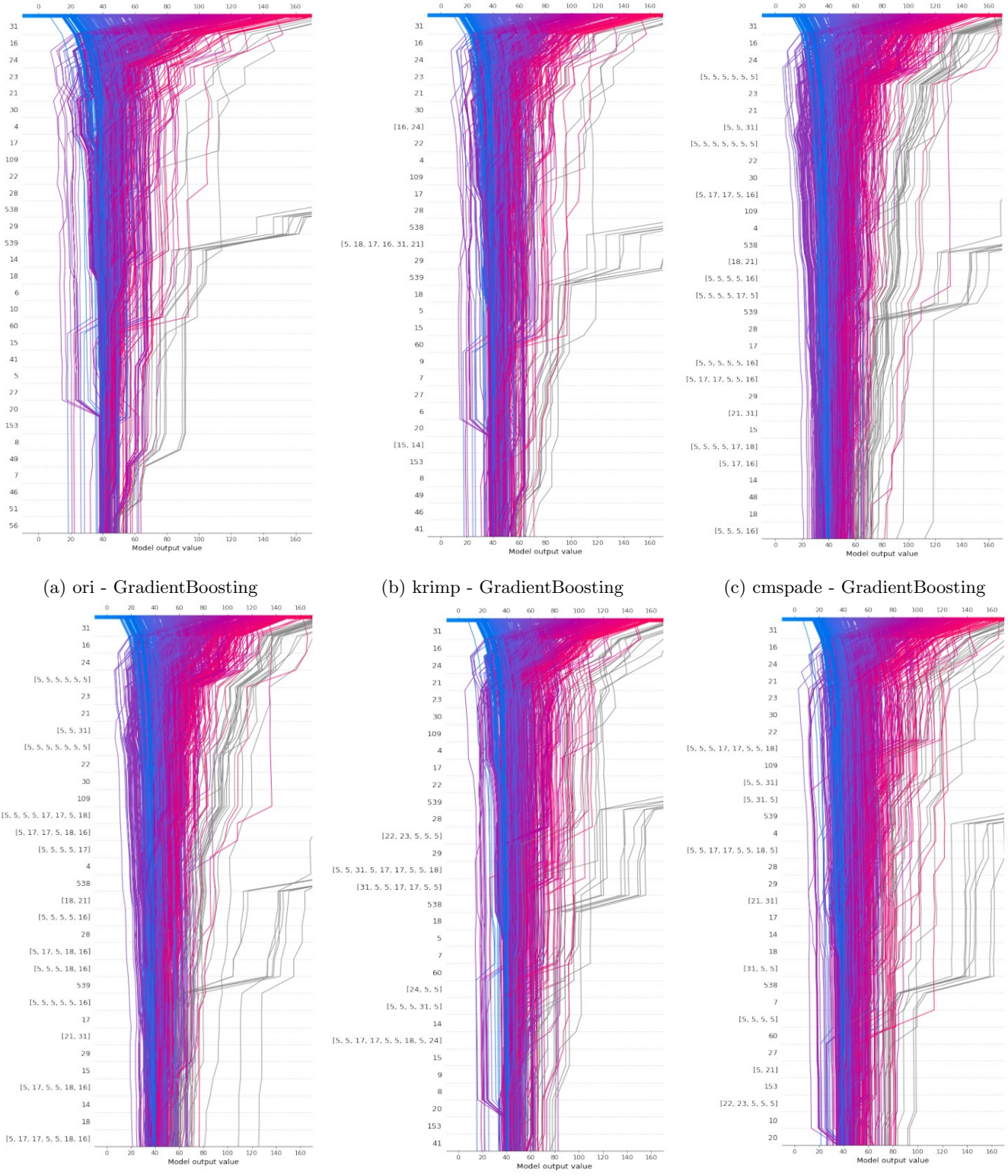
In the decision plot, in general, each line represents the explanation of a lot. On the y-axis, the features(either a single event or a pattern) are listed from top to bottom in descending order with regard to their general meantime influence. The x-axis represents the time influence result in the unit of second.

This true LOH value determines the color of the line on a spectrum. Here high LOH value gets the line redder, while a low value gets the line color bluer. The gray lines are extreme values that potentially remind us this lot is an outlier. All the line connects to the Shapley global predicted mean value at the bottom(as we only show the top30 events, so the bottom of the graph was cut). Each line interacts with the top x-axis at a point, the point value on that x-axis is the true observed LOH time of this lot. So from the bottom to the top, it is an accumulation process. The time influence of events is accumulatively adding to the more below events' time influence [37]. That is from the bottom to top, each line continuously moves right(the time contribution of a certain event is adding LOH time compared with the global predicted mean, so-called *Shapley expected Value* (like a linear model's effects are relative to the intercept). All the time influence of events adding up together, plus the global mean of the predicted LOH value of all lots, make up our LOH time. With the decision plot, we can see the detailed breakdown of the LOH time influence under different situations. This is a unique advantage of the decision plot.

The decision plot of the time influence results is in Figure 8.4 and 8.5, for the underlying prediction model Gradient Boosting and Random Forest respectively.

We can clearly see that, for lots with higher LOH value(present in more red line), especially for those with extreme LOH value (present in gray line), the time influence of events 31, 538, and 539 tends to be positively huge, as they stretch out long lines in the figure. 31 and 539 are pretty clear which tells us no laser is allowed for exposing, and 539 is an event code that means takes more than 200s. These two are both foreseeable events that have a big influence on the LOH. While 538 is recognized by domain experts as an event that should not have a direct impact on LOH time. However, we strongly doubt this event has a strong correlation to another event that has a big influence in LOH, and here 538 is just over-shadow it. Thus future work could be creating a correlation table of all the events so that we can potentially reveal the real event that is playing a big role. Although the defect of creating the correlation table is that we can only evaluate the correlation between two events, but not of multiple events(itemset or sequence).

Decision plot A: under gradient boosting

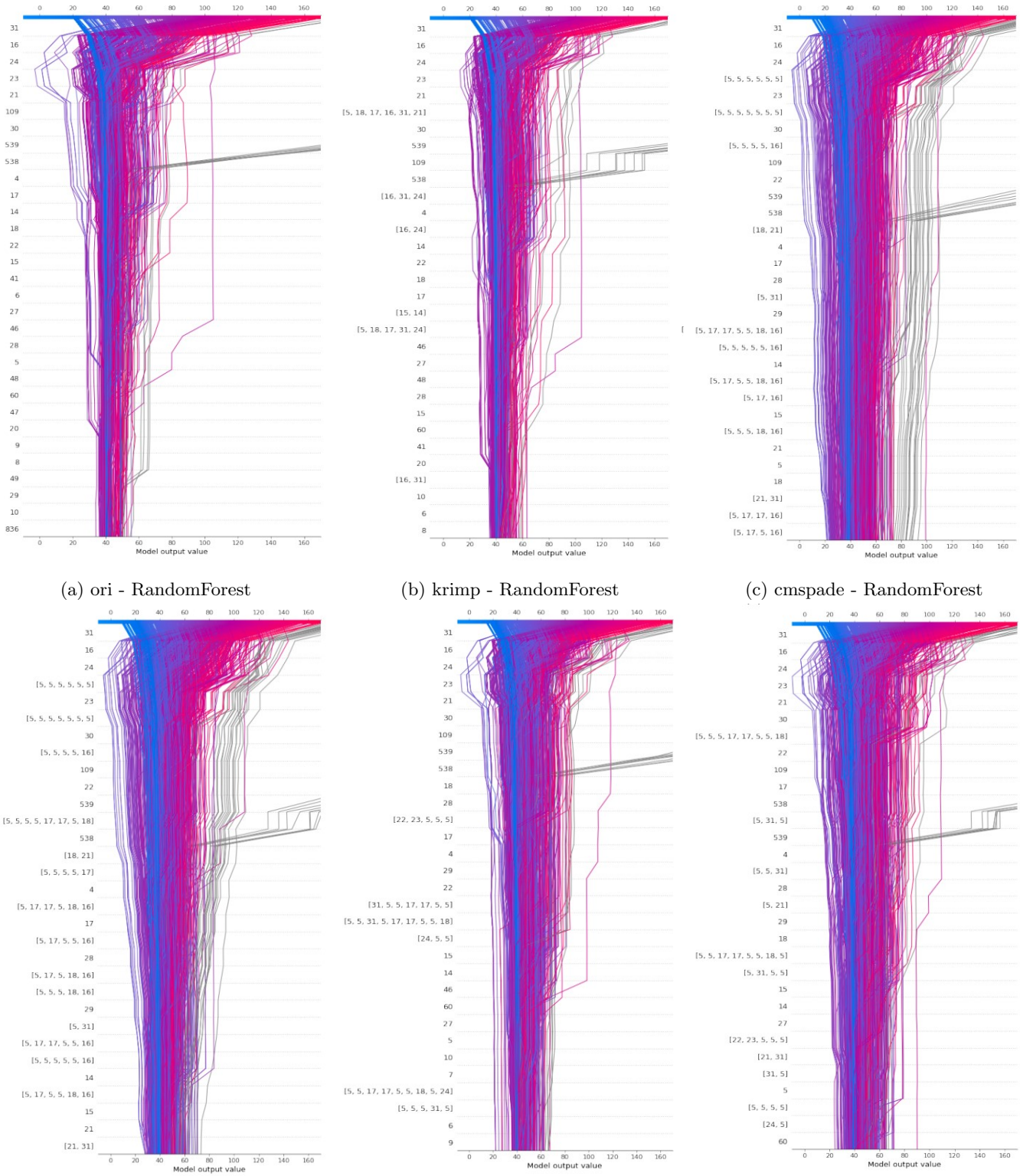


(a) ori - GradientBoosting (b) krimp - GradientBoosting (c) cmspade - GradientBoosting

(d) cmclasp - GradientBoosting (e) vmsp - GradientBoosting (f) vgen - GradientBoosting

Figure 8.4: Treeshap decision plot of 6 different feature matrices using Gradient Boosting model

Decision plot B: under random forest



(a) ori - RandomForest (b) krimp - RandomForest (c) cmspade - RandomForest

(d) cmclasp - RandomForest (e) vmsp - RandomForest (f) vgen - RandomForest

Figure 8.5: Treeshap decision plot of 6 different feature matrices using Random Forest model

8.3.3 Time influence summary plot

The summary plot can also show a large number of feature results densely and clearly like the decision plot. The difference is from the decision plot we can find the events' characteristics under a certain LOH value, while in the summary plot we lose connection with the LOH value but can inspect the effect of each event neater and extract tendency.

In our case, all the features(either event or pattern) are listed on the y-axis(listed from top to bottom in descending order with regard to their general meantime influence.) and the x-axis is the time influence value(Shapley Value) of certain event/pattern in a lot. The dots "pile up" along each feature row to show density [37]. If more features got the same Shapley Value, they will accumulate to form a vertical line in the graph. The color is designed to give information on the original value of a feature. One thing to notice is that the summary plot set the legend in the color bar type by default, which assumes the original feature value is continuous. In our case, the feature value in our created feature matrix can only be one or zero. Thus we only have(need) two colors in our figure, with red meaning the corresponding event/pattern is present in a certain lot, while blue meaning the event/pattern is absent.

The summary plot of the time influence results is in Figure 8.6 and 8.7, for the underlying prediction model Gradient Boosting and Random Forest respectively. We ordered them with regard to their general mean time influence.

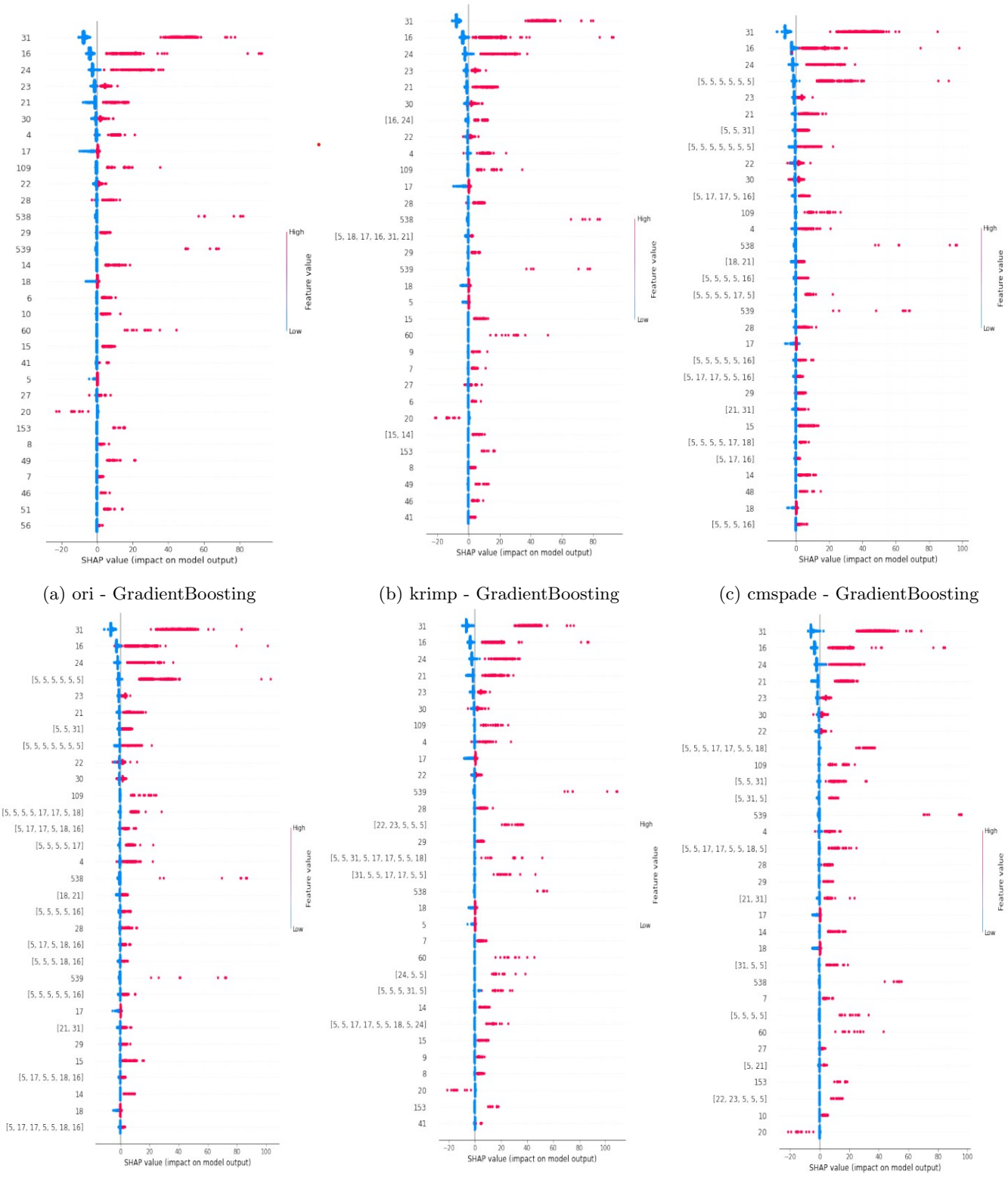
We can see that again events 538 and 539 tend to have a very huge positive effect. Event 31, 16, and pattern [5,5,5,5,5,5] also get some extreme time influence value, but their general time influence accumulated to appear at the lower x-axis.

The time influence of the event 31, 16, 24, 4,14,15 and pattern [5,5,5,5,5,5] tends to spread along the x-axis, their time influence tends to vary a lot in different situations. Thus they tend to have a strong interactive effect with other events or patterns.

Event 21 tends to have a similar absolute amount of time influence to the LOH when it appears or is absent from the lot, as the dots tend to spread symmetric on right and left sides.

Here we order the events in descending order with regard to their averaged time influence. One can easily change to set the order descending with regard to the max absolute time influence if needed. By doing so we can find extreme events in a certain lot(s).

Summary plot A: under gradient boosting

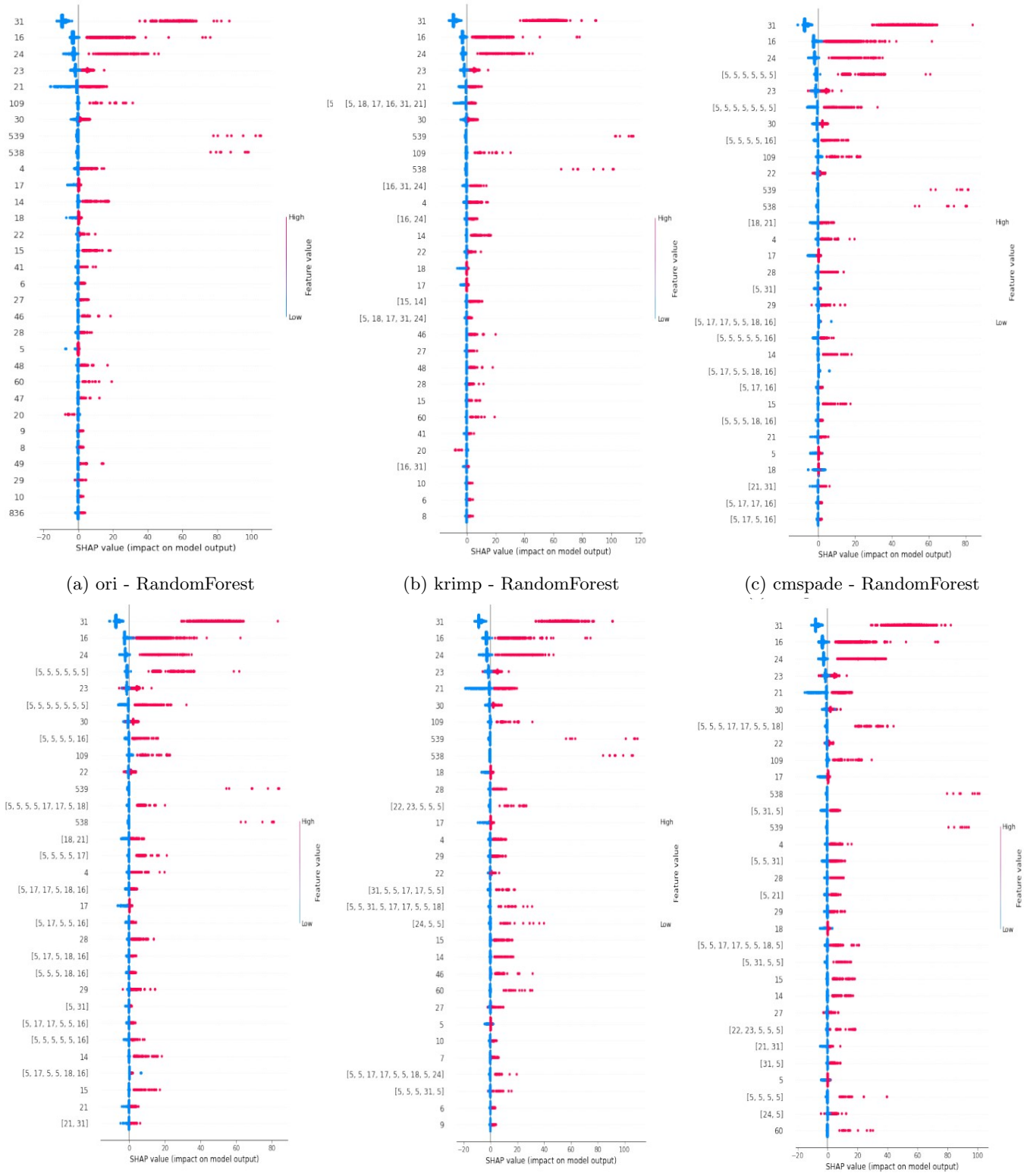


(a) ori - GradientBoosting (b) krimp - GradientBoosting (c) cmspade - GradientBoosting

(d) cmclasp - GradientBoosting (e) vmsp - GradientBoosting (f) vgen - GradientBoosting

Figure 8.6: Treeshap summary plot of 6 different feature matrices using Gradient Boosting model

Summary plot B: under random forest



(a) ori - RandomForest (b) krimp - RandomForest (c) cmspade - RandomForest

(d) cmclasp - RandomForest (e) vmsp - RandomForest (f) vgen - RandomForest

Figure 8.7: Treeshap summary plot of 6 different feature matrices using Random Forest model

8.3.4 Time influence barplot (global view)

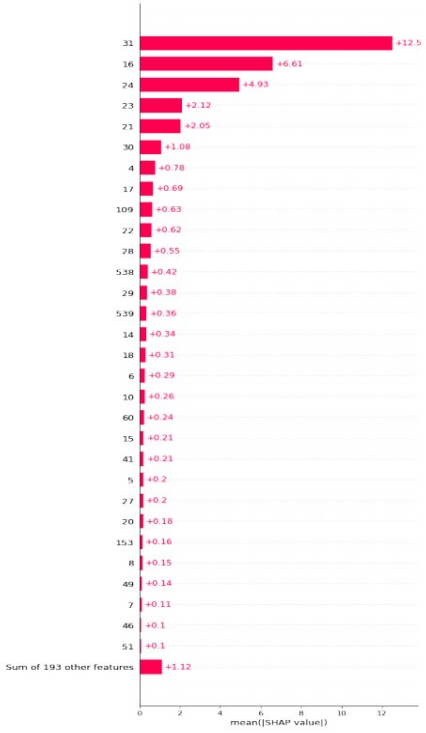
The bar plot of the time influence results are in Figure 8.8 and 8.9, for the underlying prediction model Gradient Boosting and Random Forest respectively. We ordered them with regard to their general mean time influence.

The value in the global bar plot is the mean of the absolute Shapley Value of a certain event or pattern. The calculated mean value is shown on the right side of each bar. The events/patterns are arranged along the y-axis ordered in descending order from top to bottom with regard to their time influence. In the global time influence bar plot, we can see besides the top 30 events or patterns, there is the 31st "event" located at the bottom of each figure. It is not really one "event" but the accumulated average time influence of all the events excluding the top30.

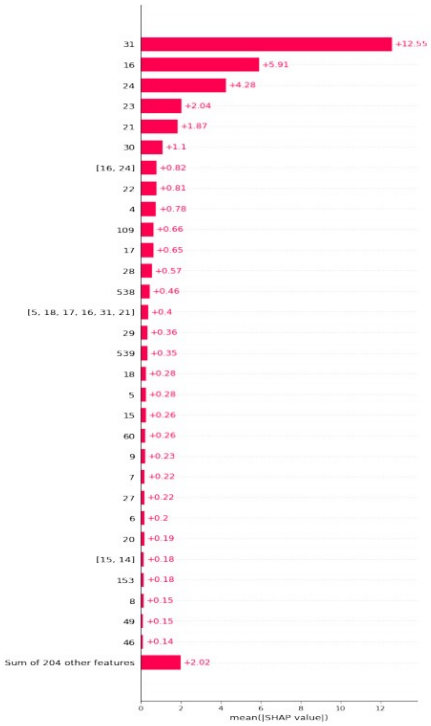
Bar plot(global) is the plot with the highest level view among the three, it tells us the feature's general time influence among the whole dataset. Practically in our case, we can easily see the general time influence of the event in a certain machine, within a certain period of time on a high level.

We can easily find events generally have a bigger time influence from top to bottom. Meanwhile, we notice that for events like 539, although it is known to take a long LOH time, among the whole dataset it will only push the predicted LOH value to the right for a small value. The most likely situation is this event does not happen that frequently for this machine within the specified period of time, thus on average, its influence is not that huge (but this event is still ranked among the top30).

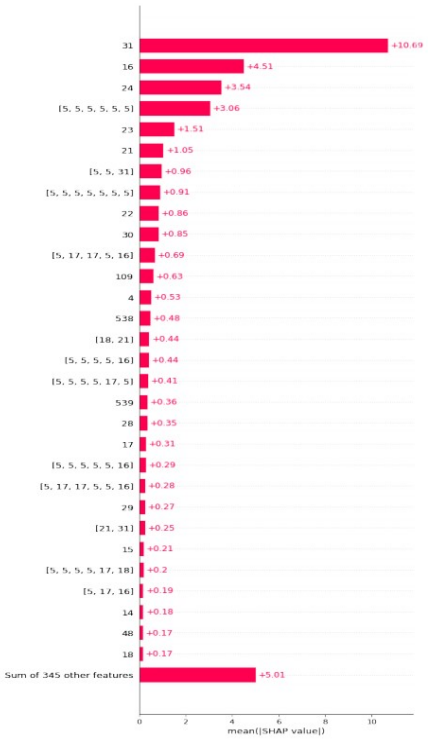
Bar plot A: under gradient boosting



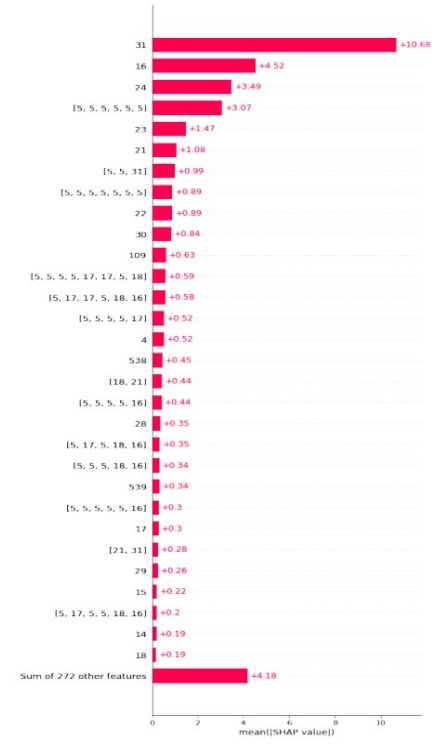
(a) ori - GradientBoosting



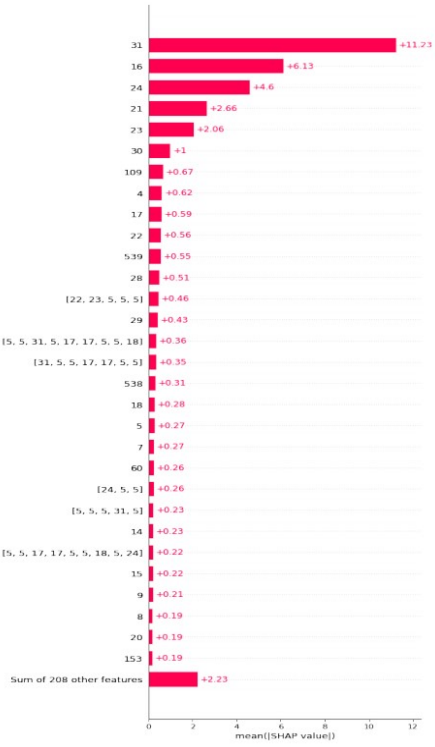
(b) krimp - GradientBoosting



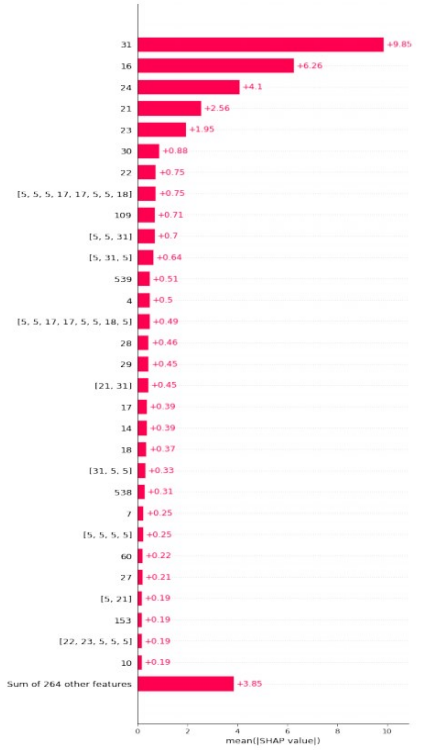
(c) cmspade - GradientBoosting



(d) cmclasp - GradientBoosting



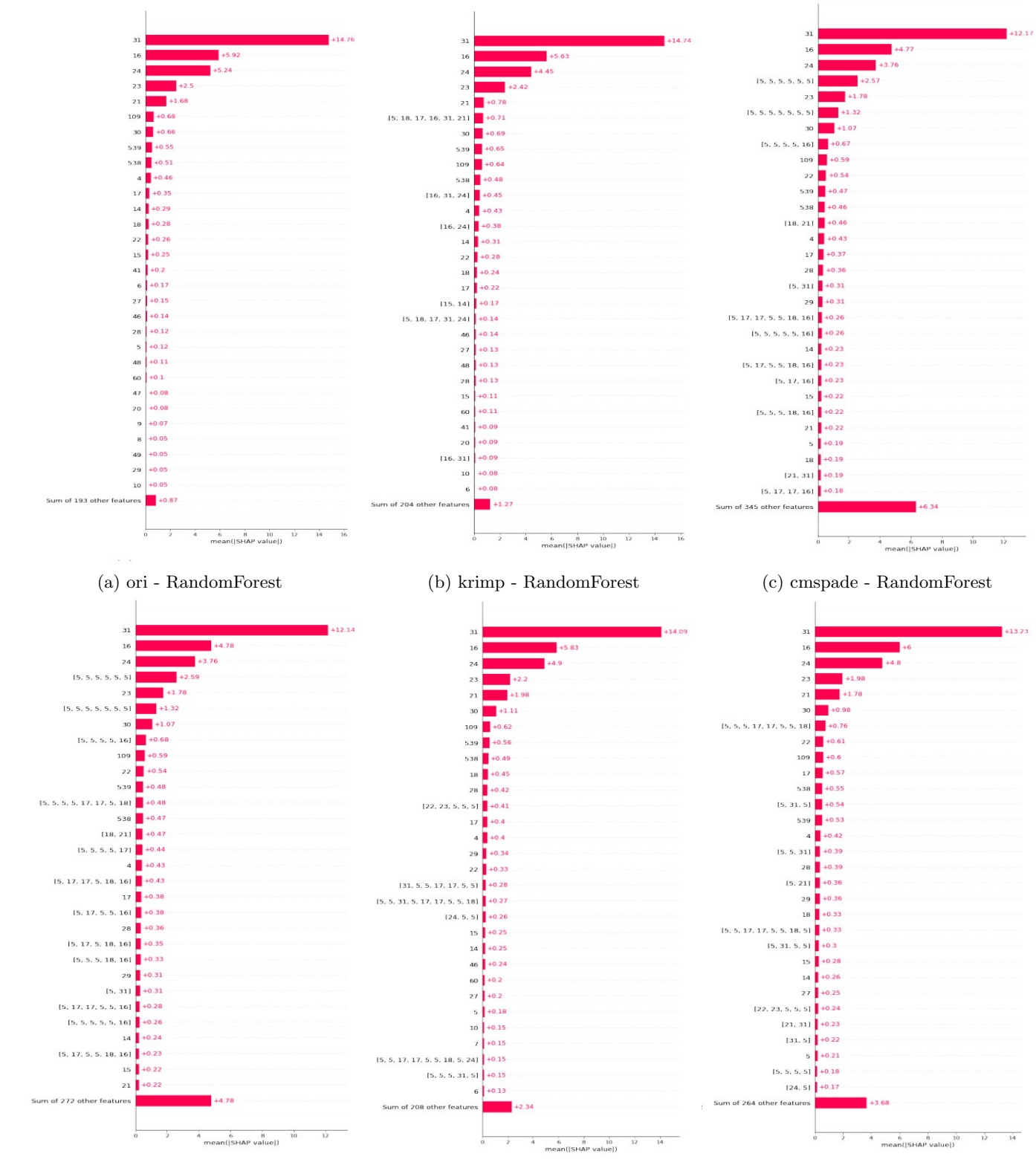
(e) vmsp - GradientBoosting



(f) vgen - GradientBoosting

Figure 8.8: Treeshap bar plot of 6 different feature matrices using Gradient Boosting model

Bar plot B: under random forest



(a) ori - RandomForest (b) krimp - RandomForest (c) cmspade - RandomForest

(d) cmclasp - RandomForest (e) vmsp - RandomForest (f) vgen - RandomForest

Figure 8.9: Treeshap bar plot of 6 different feature matrices using Random Forest model

8.3.5 Time influence heatmap

We also plot the Shapley Value result in the heatmap with the help of the hierarchical clustering [58] technique, shown in Figure 8.10 and 8.11. Applying hierarchical clustering allows us to order the lots based on their explanation similarity. This results in lots that have a similar output for a similar reason getting grouped together. The event/pattern features are listed along the y-axis ordered in descending order from top to bottom with regard to their mean time influence. The instances(lots) are spread along the x-axis grouped in order with regard to their similarity getting from hierarchical clustering. The Shapley Values are encoded on a color scale, The larger the value, the redder the color (with lower the bluer) [37]. The predicted LOH value for each lot is shown above the heatmap matrix (centered around the expected Shapley Value in Table 8.6), and the global importance of each event/pattern is shown as a bar plot on the right-hand side of the heatmap matrix plot [37]. We only plot the top 10 features in the heatmap for simplicity(with the $10th$ feature the aggregation of the rest of the features besides the top9 features). Another reason is for the rest of the features there is no strong trending appears. one can simply adjust the number of the displayed features according to the needs.

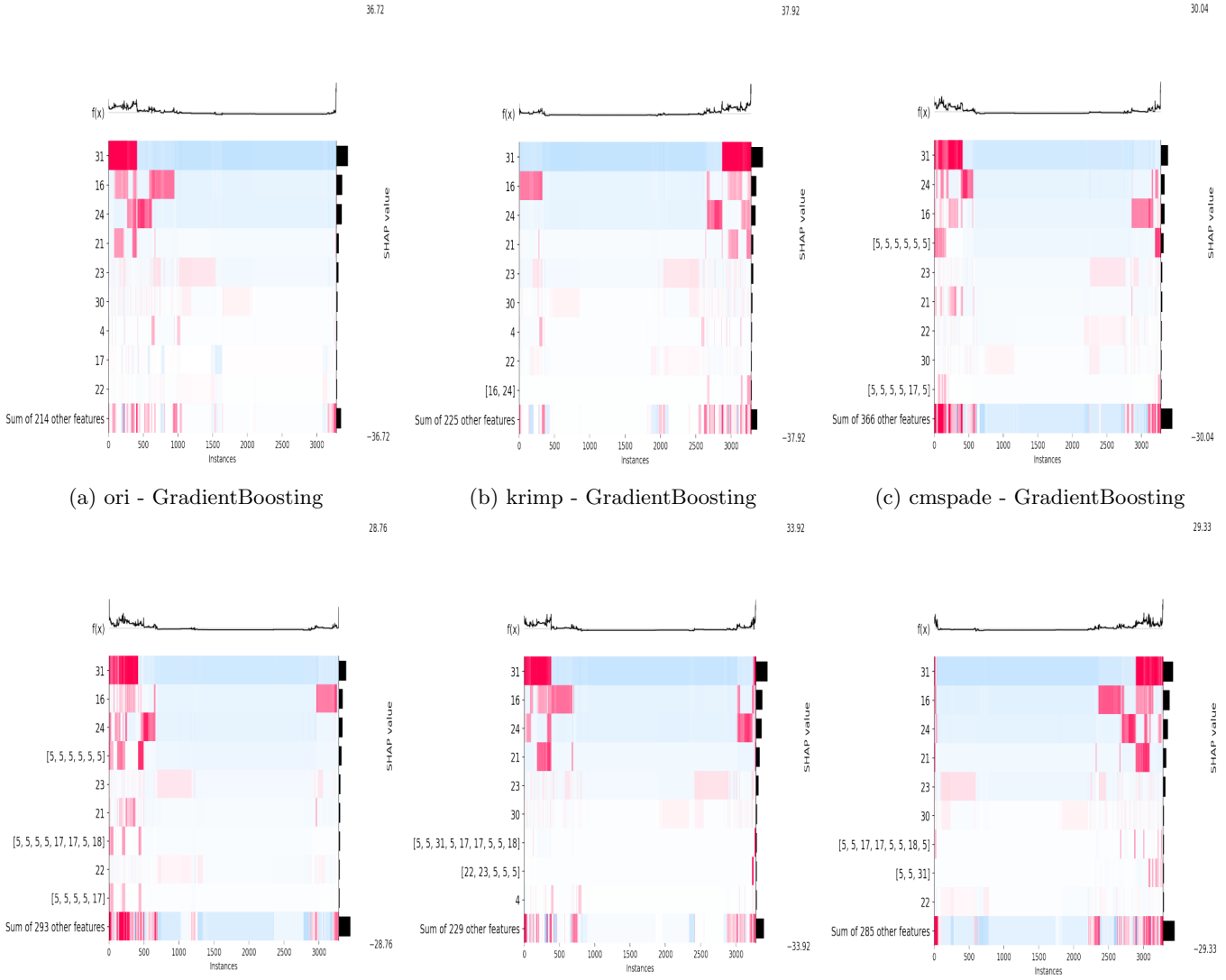
From heatmap we can mainly see the connection between the appearance of features(events/patterns) and the connection between features and the LOH time, while the quantitative details are not the thing we are most interested in (we can get quantification information from other graphs above).

In general, the lots with event 31, and 16 appears to have a bigger LOH value (the gathered red patches for event 31 and 16 and the peaks in f(x) value above the heatmap). From Figure 8.10a we can see that event 31 appears along has a bigger impact on the LOH than 16 appears alone, while the impact of the two events happening together tends to lead to a bigger impact than a simple summation of the two events happening alone(tends to lead to biggest peaks in LOH value).

From all six figures in Figure 8.11 we can see that event 16 and 24 tend to not appear together, while whenever they appear together the corresponding LOH value tends to be big from Figure 8.11a, 8.11d and 8.11e.

In Figure 8.11c we can see that whenever event 5 repetitively appears, there tends to be a large LOH.

Heatmap A: under gradient boosting

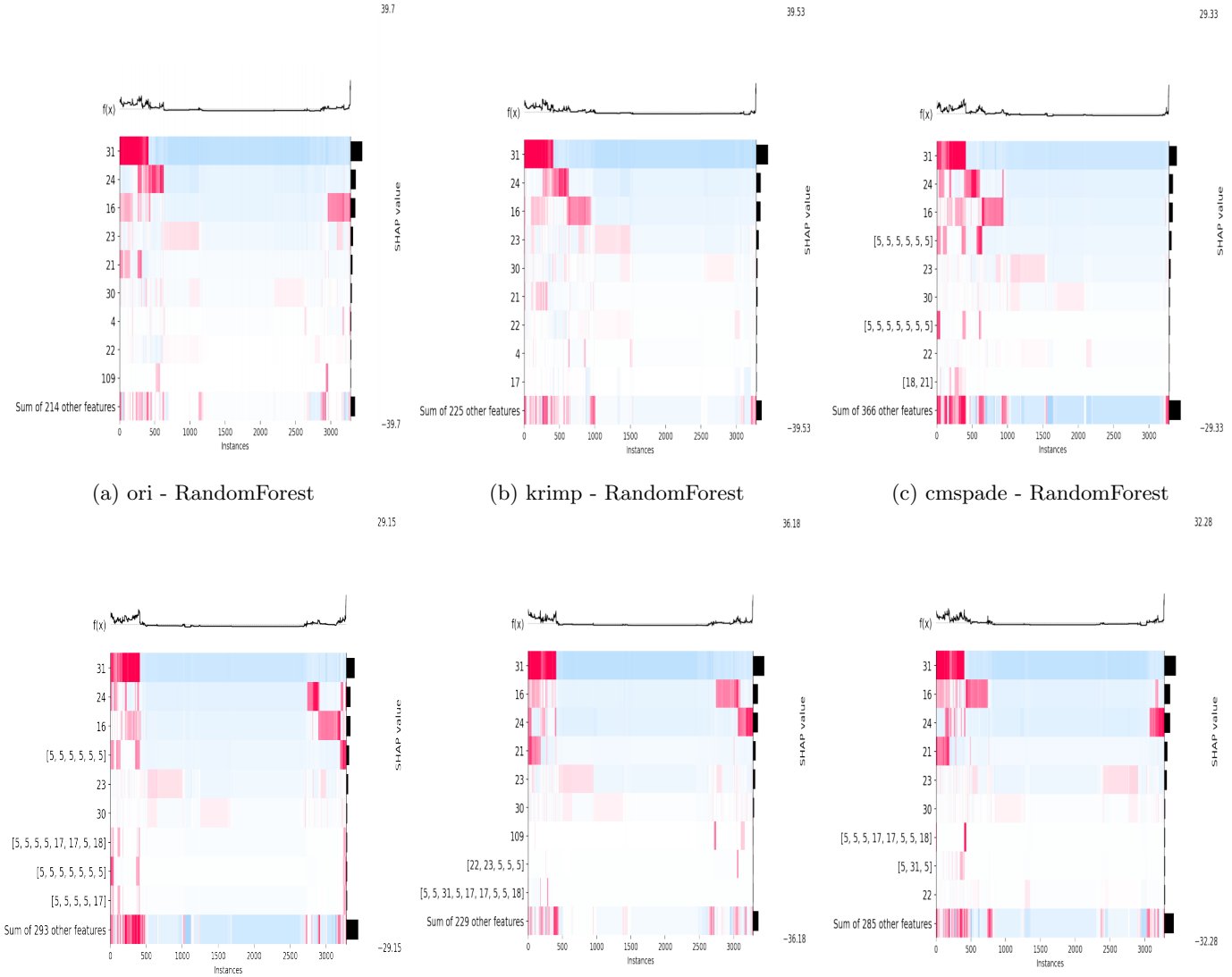


(a) ori - GradientBoosting (b) krimp - GradientBoosting (c) cmspade - GradientBoosting

(d) cmclasp - GradientBoosting (e) vmsp - GradientBoosting (f) vgen - GradientBoosting

Figure 8.10: Treeshap heatmap of 6 different feature matrices using Gradient Boosting model

Heatmap B: under random forest



(a) ori - RandomForest (b) krimp - RandomForest (c) cmspade - RandomForest

(d) cmclasp - RandomForest (e) vmsp - RandomForest (f) vgen - RandomForest

Figure 8.11: Treeshap heatmap of 6 different feature matrices using Random Forest model

8.3.6 Discussion on two prediction model result similarity

Due to the limited knowledge of event codes, we check the validity of our model by comparing different models' results.

	top30	top20	top10
ori	0.83	0.9	1
ori plus Krimp	0.8	0.95	0.9
ori plus CM-SPADE	0.7	0.65	0.7
ori plus CM-ClaSP	0.87	0.85	0.8
ori plus VMSP	0.9	0.9	0.9
ori plus VGEN	0.87	0.85	0.9

Table 8.8: Shapley Value results similarity under two models on top30, 20, 10 events and/or patterns.

We calculate the similarity of the results among the top30, top20 and top10 of the ranking list acquired from two underlying machine learning prediction models. The amount of similar events or patterns is shown in Table 8.8. The underlying feature matrix is named as "ori plus < *pattern mining method* >". For example "ori plus Krimp" is the original feature matrix adding itemset pattern candidates found by Krimp(after selection).

The similarity level is pretty high, which indicates the model is gaining some common insights and the model is valid with big potential, instead of returning random results.

We can see the similarity result drops a bit after adding selected Krimp pattern candidates compared with originally with only single events as features. Also, the similarity of the original feature matrix after adding selected Krimp pattern candidates is generally the lowest among the total six feature matrices we designed. The models tend to have different judgments on Krimp patterns, this potentially indicates the itemset patterns found by Krimp may not be representative enough as sequential patterns.

In general, the topN similarity is increasing from top30 to top20 to top10 (except for CM-ClaSP).

8.3.7 Discussion on final top pattern results under different settings

After the simple evaluation of model similarity, we check the results with the help of domain experts.

Result under feature matrix with only single events

Excluding the last aggregated "event", for the top30 events in the above figures, we can categorize them as 3 classes: *Yes*, *No* and *Unknown*. "Yes" means "known to have an impact". So the event is recognized by domain experts as the event that does have the potential to lead to long LOH. While "No" means "known to not have an impact" according to domain experts' past experience. The "Unknown" means "unknown about the existence of impact". The event codes do not recognize by domain experts so we are not sure about the real impact.

Take a look at the results from Figure 8.8a and 8.9a. We can see, among top30

results, for Shapley Value underlying gradient boosting model, we find 16 events are Yes, 12 of them are No, and 2 of them is unknown; for Shapley Value underlying random forest model, we find 18 events are Yes, 11 of them are No and 1 of them is unknown

Among the events recognized as *Yes*, there are indeed many event codes that have big potential to have big time contribution(or in other words, lead to longer LOH). For example, event 539 is the event that usually leads to a huge increase in LOH time. Also, event 31 is another important code that itself does not take any time, but a signal that the machine has been idle for at least 60 seconds. Event 15 is an indicator that the hiccup is happening in the system, this may lead to a delay. 24 again, is an event itself that does not have a big influence but could be an indicator of a 60-second delay in the system. 41 is also important as it already indicates there is a gap. The event that gets big time influence in the evaluation tends to start with certain letter combinations (in Appendix B). It looks reasonable as they are usually a lot and wafer handler-related events/actions.

For the events categorized as *Unknown*, more check and research needs to be done in the future for sure. While the event is recognized as *No*, we may also need to do some extra checks on the events that have a strong correlation with it to see if the No event occludes the other event that has an impact. This may be due to the Treeshap can handle part of the correlation problem but not all [57].

Result under feature matrix with selected itemset patterns

After analysing Figure 8.8b and 8.9b, we find both underlying prediction model recognize [5, 18, 17, 16, 31, 21] as an important pattern, ranked at 14 and 6 for Gradient Boosting and random forest respectively. As discussed before, this is indeed a pattern we are interested in. Event 16 tends to appear in every pattern while it does not have any influence on LOH, thus in the future, we can remove events like this in pre-processing (we can do this by simply add them to the "exclude code list"). The highest ranked candidates find by the 0.9 thresholds above tend to be long itemset patterns, while in the Figure we can see some smaller event set also get intense attention by the Shap-prediction model, although they were not the highest ranked candidates in terms of naive average time influence ϕ.

Result under feature matrix with selected sequential patterns

By comparing the results we present in the Figure8.8 and 8.9 above, we already know the similarity level(which is the comparison between two underlying machine learning models) is generally pretty high in all settings, but meanwhile, we can see that the events or patterns shown on the top30 generally have many overlaps(both the comparison between two models and comparison among 6 feature matrix, so 12 different settings here).

We notice that event 5 almost dominate all the sequential pattern found by Shapley. Especially in method CM-ClaSP and CM-SPADE, there are even event sequence patterns composed of pure event 5, with different lengths like [5, 5, 5,5,5,5] and [5, 5, 5, 5, 5, 5, 5]. These two patterns are correct by CM-SPADE and CM-ClaSP definitions but seem to not have lots of add-up value in terms of pattern variety. Meanwhile, this kind of pattern's dominating can be avoided by using VMSP. But does this mean VMSP is a better choice here?

Since the pattern composed by 5 is so strong, we especially checked this code. This code is related to recital movement, so it can tell us that the machine is restarting many times or taking many measurements of the reticle. Which indeed could have

big potential to have big time influence on LOH time according to domain experts. This is in line with our observations when analyzing Figure 8.2. Meanwhile, from the prediction result in Table 8.6 we can see that adding the VMSP pattern is the least helpful in prediction among all four types of sequential pattern mining patterns. Thus VMSP may not be a good choice in our case.

Also, in the pattern, we can see that event 5 is almost always followed by event 18, they are indeed a chain action that moves from library to robots to the reticle stage. So sequence mining is indeed useful to identify some meaningful chain actions, while these chain actions have the potential to have a large time influence from the domain perspective too.

8.3.8 Conclusion of experiment on Q3

Based on the above checking on top candidates and the final quantification result, we find the method is able to detect some of the most important patterns that are already known by the domain experts to lead to longer LOH. Either these patterns themselves are taking a long time or they are the reason leading to delay in LOH.

We also diagnose the root cause of the scanner we are analyzing not running smoothly or standard within a certain production time range is potentially due to a lot of too many not optimized reticle movements.

8.4 A bit more discussion

In the result checking phase, we have the chance to invite more domain experts to get involved in the project. We further discuss the data, methods, experiments, and results. Thus we come up with some notification points for the future project.

Firstly, not all events related to LOH are included in the Event dataset we have as discussed in section 4.1. To refine the analysis we can further involve the full events set into the analysis after consulting more domain experts. The incomplete data might be one of the reasons that lead to the difficulty in further improving machine learning model prediction performance in our experiment.

Secondly, the Event data logged in the system is in fact coming from different sources, each source has its own timer and these timers are not perfectly synced. Meanwhile, the lag between different timers is not fixed. This may potentially make the sequential pattern mining result not persuasive enough, as one event A could happen earlier than the other one B, while B is logged earlier in the Event dataset log than A. In the future, if the company can fix the sync problem, the sequence pattern should be more useful than itemset pattern, as it contains more information by taking the order into account. The sync problem does not affect our itemset pattern mining results, thus in the future, we can carry out more experiments on itemset pattern mining before the sync problem is solved.

Thirdly, in the thesis, we present the results that we carry out analysis on the whole dataset(more than 5k lots). Meanwhile, the domain experts prompt that we can also further subdivide the lots and analyze each subdivision part and conclude the root cause. There are two methods to subdivide the lots: 1) We can apply domain knowledge to divide the lots with known big factors, like if the certain lot has the reticle exchange or not. 2) We can take the advantage of Figure 7.3, to analyze the root cause of each peak, so that we can diagnose several major root causes under different typical LOH values. By doing this analysis accurately, we can save 24 weeks of the workforce, which is close to one FTE(Full Time Equivalent) in the team each year.

Chapter 9

Conclusion

In this project, we are dealing with data of lithography scanner machine data. We focus on the event data in a specific duration in the chip production, called lot overhead(LOH). It is a transition time between two consecutive batches/lots of wafers. From the perspective of productivity, we hope the machine runs smoothly and follow a standard procedure. However, some problems or changes do happen, resulting in a decrease in productivity. We want to know the reasons behind it.

Due to the large data volume, it is difficult to diagnose all the problems and observe patterns. We thus solve the problem by taking the advantage of data science knowledge and adopting methods from game theory, machine learning, and pattern mining. We carry out analysis on event codes logged in the machine and find out the events and patterns that have a big impact on the LOH.

To quantify the impact we adopt *Shapley Value*. To mine pattern (candidates) we design a two-stage pipeline by — firstly carry out experiments on mining four different types of sequential patterns(using *CM-SPADE*, *CM-CLASP*, *VMSP* and *VGEN*) and one itemset patterns(using *Krimp*), then let the patterns found in stage-one pass a selection mechanism. For each set of patterns mined, new feature matrices are built respectively. For the underlying machine learning prediction model for Shapley Value calculation, we choose *Random Forest* and *Gradient Boosting*.

Through the analysis of experiment results, we conclude that 1) the two-stage pattern mining method we design can help us to find high-quality pattern candidates. The patterns are meaningful in the industrial context and tend to have a big impact on LOH that either itself is taking a long time, or is the reason leading to delay in LOH. 2) by adding (appropriate)sequential patterns, we can improve the performance of the prediction model. Meanwhile, we think not all factors that influence LOH are included in our dataset, thus in the future, the prediction result can be improved. Considering both performance and time consumption, the best setup is using *Gradient Boosting* as the underlying prediction model and original feature matrix plus (selected)CM-SPADE sequential pattern candidates. 3) From the final quantification result, we find using Shapley Value together with the predictive model with feature matrices based on our pattern mining results is able to detect some of the most important patterns that are already known by the domain experts to have a big impact to LOH empirically. Either these patterns themselves are taking a long time or they are the reason leading to delay in LOH. From a data science perspective, we diagnose the main root cause that the current machine is not running smoothly or standardly within the time range of the data is potentially due to a lot of restarting and repeated measurements, and the diagnosis is recognized by domain experts.

The pipeline we design can be easily generated for the analysis of wafer exposing time(WET) and wafer swap time(WST).

Chapter 10

Future Work

Some improvements or variations could be made to current implementations. Firstly, Currently, we are using single events and patterns(itemset or sequence of events) as features to train the random forest. However, there are potential overlaps of patterns as single events. In the future, we can firstly design a cover function by integrating domain knowledge, to break the event sequence into small pieces without overlap. Then use these small pieces as new features to feed into the machine learning model.

Secondly, as we mentioned in the discussion of the experiment result part, we can design a distance measure, to measure the similarity of different sub-sequence. Thus we can use this to post-pruning the pattern candidates we get and reduce the redundancy in the candidates, with the potential to lead to better subsequent analysis.

Thirdly, the method proposed in this project can also be applied to other stages in the productivity context, like analyzing wafer swap time and wafer expose time.

Another possible research direction that can be built on the current project is anomaly detection. In the wafer production process, the procedures could be divided into normal and abnormal. One could define the normal/abnormal from the time perspective, like in this project we regard lots taking longer time as abnormal, and we want to know what are the reasons that lead to long LOH time. Another perspective is that we assume the normal industrial procedure is standardized, thus changes/deviations from this standard procedure are regarded as abnormal. However, currently, we don't have knowledge of what normal procedures are like (what is the standard event sequence). If we can assume, most of the time, the machine is running under the right condition, then the frequent event sequence could be regarded as "normal". In this situation, one can also use similar sequential pattern mining methods used in this project, to firstly find frequent event sequences as "normal", and then apply anomaly detection techniques to find "abnormal" behaviors. For example, we can find the fuzzy match of the "normal" (frequent) event sequence. Then by comparing the fuzzy match and the original frequent sequence, we can find the deviations, and those deviations are the abnormal events/patterns(anomaly) that we can notify domain experts to pay attention to when they are diagnosing the machine's problem in decreasing productivity.

One of the problems we face in this project is, that currently there is no previous work on mining and analyzing sequential patterns with a numerical target(consequently in a regression context). Thus new research direction could be established in mining sequential patterns with regard to numerical targets.

Bibliography

[1] Andreas Thoss. "*EUV lithography revisited.*" https://www.laserfocusworld.com/blogs/article/14039015/how-does-the-laser-technology-in-euv-lithography-work

[2] Jessica Timings. "TWINSCAN: 20 years of lithography innovation". Retrieved June 16, 2022 https://www.asml.com/en/news/stories/2021/twinscan-20-years-innovation

[3] Wikipedia contributors. (2022, June 7). "2020–present global chip shortage". In Wikipedia, The Free Encyclopedia. Retrieved June 16, 2022.

[4] ASML website. "The world's short on chips, the semiconductor industry is up for the challenge". Retrieved June 16, 2022.

[5] Mannila H, Toivonen H (1996) Multiple uses of frequent sets and condensed representations. In: Proceedings of KDD'96, pp 189–194.

[6] Fournier-Viger, Philippe, et al. "A survey of sequential pattern mining." Data Science and Pattern Recognition 1.1 (2017): 54-77.

[7] Wikipedia contributors. (2022, June 26). "K-means clustering". In Wikipedia, The Free Encyclopedia. Retrieved July 25, 2022.

[8] Arjun Sangitrao. "Patterns & correlations in scanner data for wafer output optimization". TUE master thesis(2020).

[9] Jason Brownlee. "How to Calculate Feature Importance With Python". Retrieved July 26, 2022. https://machinelearningmastery.com/calculate-feature-importance-with-python/.

[10] Wikipedia contributors. (2022, July 25). "Explainable artificial intelligence". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[11] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Model-agnostic interpretability of machine learning." arXiv preprint arXiv:1606.05386 (2016).

[12] Wikipedia contributors. (2020, September 23). "Combinatorial search". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[13] Dong, Guozhu, and Vahid Taslimitehrani. "Pattern-aided regression modeling and prediction model analysis." IEEE Transactions on Knowledge and Data Engineering 27.9 (2015): 2452-2465.

[14] Lesh, Neal, Mohammed J. Zaki, and Mitsunori Ogihara. "Mining features for sequence classification." Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999.

[15] Exarchos, Themis P., et al. "A two-stage methodology for sequence classification based on sequential pattern mining and optimization." Data & Knowledge Engineering 66.3 (2008): 467-487.

[16] Wikipedia contributors. (2022, July 26). Analysis of variance. In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[17] R. Agrawal, and R. Srikant, "Mining sequential patterns," The International Conference on Data Engineering, pp. 3–14, 1995.

[18] R. Srikant, and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," The International Conference on Extending Database Technology, pp. 1–17, 1996.

[19] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," Machine learning, vol.42(1-2), pp. 31–60, 2001.

[20] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.429–435, 2002.

[21] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information," The Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 40–52, 2014.

[22] Wikipedia contributors. (2022, June 2). "Breadth-first search". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[23] Wikipedia contributors. (2022, July 15). "Depth-first search". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[24] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng, "VMSP: Efficient vertical mining of maximal sequential patterns," The Canadian Conference on Artificial Intelligence, pp. 83–94, 2014.

[25] P. Fournier-Viger, A. Gomariz, M. Sebek, M. Hlosta, "VGEN: fast vertical mining of sequential generator patterns," The International Conference on Data Warehousing and Knowledge Discovery, pp. 476–488, 2014.

[26] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[27] Chaya Bakshi. "Random Forest Regression(Jun 8, 2020)". Retrieved April 20, 2022, from `https://levelup.gitconnected.com/random-forest-regression-209c0f354c84`

[28] Wikipedia contributors. (2022, May 31). Cooperative game theory. In Wikipedia, The Free Encyclopedia. Retrieved 01:44, June 7, 2022, from `https://en.wikipedia.org/w/index.php?title=Cooperative_game_theory&oldid=1090843550`

[29] Shapley, Lloyd S. "A value for n-person games." Contributions to the Theory of Games 2.28 (1953): 307-317

[30] Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In Advances in Neural Information Processing Systems 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Montreal: Curran Associates.

[31] Lundberg, Scott. 2019. "SHAP (SHapley Additive exPlanations)". https://github.com/slundberg/shap.

[32] Molnar, C. (2022). "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2nd ed.)". christophm.github.io/interpretable-ml-book/

[33] Wikipedia contributors. (2022, June 22). "Confounding". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[34] Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W. and Robins, J. (2018), Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21: C1-C68.

[35] Luke Merrick, Ankur Taly. "The Explanation Game Explaining ML models with Shapley Values". `https://course.ece.cmu.edu/~ece739/lectures/18739-2020-spring-lecture-14-shapley.pdf`

[36] Lundberg, Scott M., et al. "Explainable AI for trees: From local explanations to global understanding." arXiv preprint arXiv:1905.04610 (2019).

[37] Lundberg, Scott M., et al. "From local explanations to global understanding with explainable AI for trees." Nature machine intelligence 2.1 (2020): 56-67.

[38] Wikipedia contributors. (2022, June 20). "Random forest". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[39] Wikipedia contributors. (2022, July 21). "Gradient boosting". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[40] Wikipedia contributors. (2022, June 6). "Decision tree". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[41] Wikipedia contributors. (2022, May 3). "Decision tree pruning". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[42] Wikipedia contributors. (2022, June 8). "Overfitting". In Wikipedia, The Free Encyclopedia. Retrieved July 26, 2022.

[43] Wikipedia contributors. (2022, April 11). Minimum description length. In Wikipedia, The Free Encyclopedia. Retrieved 21:38, April 12, 2022, from `https://en.wikipedia.org/w/index.php?title=Minimum_description_length&oldid=1082210057`

[44] van Leeuwen, M., Vreeken, J. & Siebes, "A. Compression Picks the Item Sets that Matter". In Proc. ECML PKDD'06, pages 585-592, 2006

[45] Siebes, A., Vreeken, J. & van Leeuwen, M. "Item Sets that Compress". In Proc. SDM'06, pages 393-404, 2006.

[46] Pedregosa, F. et el. "Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research, page 2825-2830, 2011

[47] Chen, Tianqi and Guestrin, Carlos. "XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", KDD '16, (2016).

[48] Vishal Morde. "XGBoost Algorithm: Long May She Reign!" retrieved June 7, 2022. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[49] Fournier-Viger, P., Lin, C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H. T. (2016). "The SPMF Open-Source Data Mining Library Version 2". Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III, Springer LNCS 9853, pp. 36-40.

[50] Summer Hu. "Complete SHAP tutorial for model explanation Part 1". Shapley Value. retrieved 2022/5/18.https://summer-hu-92978.medium.com/complete-shap-tutorial-for-model-explanation-part-1-shapley-value-b67c7f19908c.

[51] Štrumbelj, Erik, and Igor Kononenko. 2010. "An Efficient Explanation of Individual Classifications Using Game Theory." Journal of Machine Learning Research 11 (March): 1–18. http://dl.acm.org/citation.cfm?id=1756006.1756007.

[52] Štrumbelj, Erik, and Igor Kononenko. 2014. "Explaining prediction models and individual predictions with feature contributions." Knowledge and Information Systems 41 (3): 647–65. https://doi.org/10.1007/s10115-013-0679-x.

[53] Reza Bagheri. "Introduction to SHAP Values and their Application in Machine Learning". retrieved 2022-5-18, https://towardsdatascience.com/introduction-to-shap-values-and-their-application-in-machine-learning-8003718e6827.

[54] James, Gareth, et al. "An introduction to statistical learning". Vol. 112. New York: springer, 2013.

[55] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling", IEEE Transactions on Information Theory, vol. 44(6), pp.2743-2760, 1998.

[56] Janzing, Dominik. "Causal regularization." Advances in Neural Information Processing Systems 32 (2019).

[57] Kjersti Aas, Martin Jullum, Anders Løland, "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values", Artificial Intelligence, Volume 298, 2021.

[58] Wikipedia contributors. (2022, June 7). Hierarchical clustering. In Wikipedia, The Free Encyclopedia. Retrieved June 29, 2022