Leveraging process mining for knowledge extraction and process optimization

Le Toan Duong

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *26/06/2023* **par :**

Le Toan DUONG

Leveraging process mining for knowledge extraction and process optimization

---

JURY

M. Teresa GÓMEZ-LÓPEZ	Catedrática de Universidad	Rapporteure
Christophe BIERNACKI	Professeur d'Université	Rapporteur
François PÉRÈS	Professeur d'Université	Président du jury
Marie COTTRELL	Professeure émérite d'Université	Examinatrice
Louise TRAVÉ-MASSUYÈS	Directrice de recherche	Directrice de Thèse
Audine SUBIAS	Professeure d'Université	Co-directrice de Thèse
Christophe MERLE	Ingénieur	Invité

École doctorale et spécialité :

EDSYS : Informatique 4200018

Unité de Recherche :

Laboratoire d'Analyse et d'Architecture des systèmes (UMR 8001)

Directeur(s) de Thèse :

Louise TRAVÉ-MASSUYÈS et *Audine SUBIAS*

Rapporteurs :

María Teresa GÓMEZ-LÓPEZ et *Christophe BIERNACKI*

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep gratitude to everyone who has supported and contributed to the successful completion of my doctoral research. This journey has been both challenging and rewarding, and I am incredibly grateful to the following individuals and organizations for their valuable help and encouragement.

First and foremost, I would like to express my sincere thanks to my supervisors, Research Director **Louise Travé-Massuyès** and Prof. **Audine Subias**, for their constant guidance, expertise, and motivation throughout my research. Their insightful feedback, constructive criticism, and commitment have been crucial in pushing me to do my best.

I would also like to extend my heartfelt appreciation to **Christophe Merle**, my manager at Vitesco Technologies, for his support and cooperation during my doctoral research. His understanding of the importance of academic pursuits and his willingness to accommodate my professional development within the company have been remarkable.

I am grateful to the members of my doctoral committee, Prof. **François Pérès**, and Prof. **Béatrice Laurent-Bonneau**, for their helpful suggestions and valuable inputs during the different stages of my research.

I would like to sincerely thank the reviewers of my manuscript, Prof. **Maria Teresa Gómez-López**, and Prof. **Christophe Biernacki**, for their constructive comments and suggestions, which have played a crucial role in improving the overall quality of my work. Additionally, I extend my gratitude to Prof. **François Pérès** and Prof. **Marie Cottrell** for agreeing to be members of my jury and for their time, expertise, and valuable feedback during my thesis defense.

I extend my appreciation to **Vitesco Technologies** for their support and collaboration during my PhD research. The company's resources, facilities, and access to real-world data have been instrumental in conducting comprehensive research and validating the practical applications of my work. I would also like to extend my thanks to **Udo Kreissing** for the opportunities provided to engage in meaningful industry partnerships and the invaluable insights gained through this collaboration.

I would like to acknowledge **LAAS-CNRS** and **ANITI**, the institutes where I conducted my research. I am thankful for the collaborative environment and the valuable

interactions with colleagues and researchers, which have enriched my academic and scientific journey.

I am grateful to **INSA Toulouse** and **Edsys** for providing me with the necessary resources, and a stimulating academic environment. The support from the direction and administrative staff has been exceptional, and I would like to thank each and every one of them for their assistance and cooperation.

Lastly, I want to extend my deepest gratitude to **my family**. Their unwavering love, encouragement, and understanding have been the foundation of my journey. Their constant support and belief in my abilities have given me the strength and resilience to overcome challenges and pursue my academic goals.

To all those who have contributed, directly or indirectly, to my doctoral journey, whether mentioned explicitly or not, please accept my heartfelt appreciation. Your guidance, encouragement, and support have played a significant role in shaping my research and personal growth.

Completing this PhD is the result of collective efforts, and I feel honored and humbled to have had the privilege of working with such exceptional individuals and institutions. The knowledge and experiences gained during this journey will undoubtedly shape my future.

Thank you all sincerely for your contributions.

Toulouse, France, 26 June 2023
Le Toan DUONG

Table of Contents

List of Figures

List of Tables

Notations and acronyms

This section provides the mathematical notations and the abbreviations/acronyms used throughout the thesis.

For definitions and problem formulation, the following notations were used.

A	The adjacency matrix.
AN	The set of attribute names.
$\#_n(e)$	The value of attribute $n \in AN$ for event $e \in \mathcal{E}$.
$\mathbf{B}$	The set of batches.
$\mathcal{D}$	The dissimilarity matrix.
$\mathcal{E}$	The event universe.
$\mathcal{E}^*$	The set of all finite subsets of $\mathcal{E}$.
$\mathbb{E}[.]$	The expectation of a variable or a function.
E	The set of edges.
$\mathcal{G}$	The graph.
$K(.)$	The kernel function.
L	The set of labels associated with edges in a graph.
$\mathcal{L}$	The event log.
$\mathcal{P}^*$	The nominal path.
Φ	The set of production phases.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}^p$	The p-dimensional Euclidian space.
$\mathcal{T}$	The set of traces.
V	The set vertices.

Let's also give the following abbreviations.

AOI	Automated Optical Inspection.
ARIMA	AutoRegressive Integrated Moving Average.
AWS	Amazon Web Services.
BE	Back End.
BPMN	Business Process Model and Notation.
CNN(s)	Convolutional Neural Network(s).
DFG	Directly Follows Graph.
DPM	Descriptive Process Mining.
FC	Fully Connected.
FE	Front End.
FE1	Front End side 1.
FE2	Front End side 2.
GAT(s)	Graph Attention Network(s).
GCN(s)	Graph Convolutional Network(s).
GNN(s)	Graph Neural Network(s).
GGNN	Gated Graph Neural Network.
GRNN(s)	Graph Recurrent Neural Network(s).
GRU(s)	Gated Recurrent Unit(s).
ICT	In Circuit Test.
JS	Jensen-Shannon.
KDE	Kernel Density Estimation.
KL	Kullback-Leibler.
LCT	Life Cycle Time.
LOO	Leave One Out.
LSTM	Long Short-Term Memory.
MAE	Mean Absolute Error.
MAPE	Mean Absolute Percentage Error.
MES	Manufacturing Execution Systems.
MISE	Mean Integrated Squared Error.
MLP	Multi-Layer Perceptron.
MSE	Mean Squared Error.
MSL	Moisture Sensitivity Level.
MPNN(s)	Message Passing Neural Network(s).

P&P	Pick and Place.
PCB(s)	Printed Circuit Board(s).
PPM	Predictive Process Mining.
R-GCN(s)	Recurrent Graph Convolutional Network(s).
RCT	Remaining Cycle Time.
RNN(s)	Recurrent Neural Network(s).
S3	Simple Storage Service.
SMD	Surface Mounted Devices.
SMT	Surface Mount Technology.
SPI	Solder Paste Inspection.
SPP	Solder Paste Printing.
SSE	Sum of Squared Errors.
t-PM	Timed Process Model.
VGAE	Variational Graph Auto-Encoders.
WIP	Work-in-progress.
XGBoost	eXtreme Gradient Boosting.

Résumé

Extraction de connaissances pour l'optimisation des processus de production par process mining

La recherche présentée dans cette thèse s'intéresse aux méthodes utilisées dans le domaine du *process mining*, qui est un domaine en pleine expansion ces dernières années. Le *process mining* est un sous-domaine de la science des données qui comprend des techniques permettant d'analyser et d'optimiser des processus en utilisant des données de journaux, ou logs d'événements. Dans ce travail, nous nous concentrons spécifiquement sur les logs d'événements enregistrés dans les processus de fabrication. Les travaux effectués se divisent en deux axes principaux du *process mining* : le *process mining* descriptif et le *process mining* prédictif. Le *process mining* descriptif consiste à analyser les logs d'événements pour extraire des connaissances sur le processus et identifier les points à améliorer, tandis que le *process mining* prédictif concerne la construction de modèles d'apprentissage pour prédire le comportement futur du processus.

Concernant le *process mining* descriptif, la première étape consiste à extraire et prétraiter les données brutes. Cette étape est cruciale pour mieux comprendre le processus et bien préparer les données. Ensuite, des techniques liées à la découverte automatique de processus, à la vérification de la conformité et à l'analyse de la performance sont mises en oeuvre. La découverte de processus consiste à construire le modèle de processus à partir des logs d'événements. Le chemin nominal, avec des contraintes temporelles de transition entre les opérations, a été établi comme référence pour les produits normaux. En comparant le chemin d'écoulement des produits avec le chemin nominal temporalisé, un indice de qualité a été calculé pour caractériser la qualité des produits. Ce calcul est basé sur différents critères tels que la présence ou l'absence d'opérations nominales, les défaillances pendant la production, le temps de transition et le lot de produits, etc. De plus, une analyse de la performance de production entre les équipes de travail a été menée en

utilisant l'indice de qualité obtenu. Au niveau applicatif, nous avons développé une application de suivi des encours en temps réel pour aider le fabricant à connaître l'état actuel du processus et à mieux gérer la production. Les résultats obtenus par les techniques de *process mining* descriptif ont fourni des connaissances approfondies sur le processus de production et les points d'amélioration. De plus, ils ont mis en évidence le grand potentiel de l'utilisation de la technique de *process mining* dans l'industrie manufacturière.

En ce qui concerne le *process mining* prédictif, nous développons des modèles d'apprentissage basés sur des logs d'évènements pour prédire le comportement futur du processus. Plusieurs problèmes de prédiction ont été proposés et testés ; parmi ceux-ci, deux problèmes ont été présentés dans ce manuscrit : la prédiction des encours et la prédiction du temps restant des produits en encours. Pour la prédiction des encours, nous avons évalué et comparé différents modèles, allant des méthodes classiques comme la régression linéaire aux modèles plus avancés et complexes tels que des réseaux de neurones récurrents LSTM. Les résultats révèlent que la régression linéaire donne les meilleurs résultats en termes d'erreur de prédiction, de temps de calcul et d'interprétabilité. Cependant, le modèle rencontre encore des difficultés avec le changement de tendance et la prédiction à plusieurs étapes. Les résultats suggèrent d'incorporer des informations supplémentaires telles que la planification de la production et les données logistiques.

Concernant la prédiction du temps restant, nous proposons d'utiliser les réseaux de neurones pour le graphe (GNNs) pour résoudre ce problème. GNN est un domaine de recherche relativement nouveau qui se concentre sur les réseaux neuronaux conçus pour traiter des données présentées sous forme de graphes. Nous avons choisi les GNNs car les modèles de processus sont généralement représentés par des graphes avec des nœuds représentant des activités, des événements ou des états, et des arcs représentant des dépendances ou des transitions entre eux. En exploitant la structure inhérente des modèles de processus, les réseaux de neurones à graphes peuvent capturer efficacement les dépendances complexes entre différentes activités. De plus, nous sommes les premiers à utiliser les GNNs pour prédire le temps restant dans les processus. Nous avons comparé la performance de ce modèle avec le *benchmark*, qui est le modèle LSTM. Les résultats indiquent que notre modèle GNN surpasse le LSTM de référence pour les processus longs et complexes. Ces résultats mettent en évidence le potentiel d'utilisation des GNNs dans les applications de *process mining*.

Le travail de thèse présenté dans ce manuscrit est appliqué au développement de l'usine du futur pour toutes les usines de Vitesco Technologies, avec un accent particulier sur la

production de cartes électroniques pour les véhicules électriques et hybrides à Foix, en France.

L'industrie automobile est en constante évolution, avec des changements technologiques, économiques et sociaux. Les dernières évolutions incluent l'automatisation et la transformation numérique. Le processus de fabrication dans cette industrie est complexe et requiert une grande précision ainsi qu'une coordination entre les fournisseurs, les fabricants et les usines d'assemblage. Comprendre ces processus en profondeur à l'aide de méthodes traditionnelles est de plus en plus difficile. C'est pourquoi des techniques avancées intégrant des informations provenant de différentes sources et appliquant une analyse à grande échelle des données sont nécessaires pour obtenir des informations utiles. Les entreprises qui adoptent des technologies avancées, telles que la robotique, l'internet des objets et l'intelligence artificielle, peuvent améliorer leur efficacité, réduire leurs coûts et accélérer la prise de décision.

Les données que nous avons traitées proviennent de l'usine de Foix de Vitesco Technologies dans laquelle sont conçus et fabriqués des produits électroniques à grande échelle. La production fonctionne 24/7 et produit des dizaines de millions de produits chaque année. La production comprend quatre phases principales : front end, ICT, back end et l'emballage. Les données sont collectées en temps réel tout au long du processus, depuis les PCB vides jusqu'aux cartes électroniques assemblées. Ces données sont stockées sous forme de log d'événements. Afin d'assurer la qualité des produits livrés et éviter les non-conformités, l'entreprise doit surveiller la performance des machines et du personnel, ainsi que les coûts et les délais de production. La surveillance se base sur une compréhension détaillée du processus et de l'état du système de production. Dans cette optique, cette thèse se focalise sur l'extraction de connaissances à partir de logs d'évènements, de signaux et d'indicateurs pour améliorer le processus de production. Cette analyse permettra de prendre des décisions qui conduiront à l'optimisation du processus, de la planification et de la maintenance.

Cette thèse CIFRE s'inscrit dans le cadre de la chaire *"Collaborative AI : Synergistic transformations in model based and data-based diagnosis"* à l'Institut Interdisciplinaire d'Intelligence Artificielle de Toulouse (ANITI). Les travaux ont été menés dans le cadre d'une collaboration entre le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) et l'entreprise Vitesco Technologies, située à Toulouse, France.

Mots clés: Process Mining, Journaux d'événements, Analyse Prédictive, Graph Neural Networks, Industrie 4.0

Abstract

Leveraging process mining for knowledge extraction and process optimization

The research presented in this thesis centers around process mining, a rapidly expanding field in recent years. Process mining is a subfield of data science that involves utilizing event logs to analyze and optimize processes. The focus of this work is specifically on event logs derived from manufacturing processes. The research is divided into two main areas of process mining: descriptive process mining and predictive process mining. Descriptive process mining involves analyzing event logs to extract insights about the process and identify areas that can be improved. On the other hand, predictive process mining involves constructing learning models to forecast the future behavior of the process.

Regarding descriptive process mining, the initial step involves extracting and preprocessing raw data, which is crucial for gaining a comprehensive understanding of the process and ensuring proper data preparation. Subsequently, techniques related to automated process discovery, compliance checking, and performance analysis are implemented. Process discovery specifically involves constructing a process model based on event logs. Then, a nominal path is established as a reference for normal products. By comparing the production path of products with the nominal path, a quality index is calculated to characterize product quality. This calculation takes into account various criteria, including the presence or absence of nominal operations, production failures, transition time, product batches, etc. Furthermore, an analysis of production performance is conducted by comparing time slots using the obtained quality index. Additionally, an application for real-time work-in-progress tracking has been developed to aid manufacturers in understanding the current state of the process and enhancing production management. The results obtained through descriptive process mining techniques offer profound insights into the production process and identify areas for improvement, thereby showcasing the immense potential of process mining in the manufacturing industry.

In terms of predictive process mining, we have developed learning models based on

event logs to predict the future behavior of the process. We have explored several prediction problems, two of which are presented in this manuscript: work-in-progress prediction and remaining cycle time prediction. For work-in-progress prediction, we have evaluated and compared different models, including classical methods like linear regression and more advanced and complex models like recurrent neural networks LSTM. The results indicate that linear regression performs the best in terms of prediction error, computation time, and interpretability. However, the model still faces challenges when it comes to handling trend reversals and making multi-step predictions. Therefore, the results suggest the need to incorporate additional information, such as production planning and logistics data, to enhance the model's performance.

In order to predict the remaining cycle time, we propose using Graph Neural Networks (GNNs) to tackle this problem. GNNs belong to a relatively new research field that focuses on neural networks designed specifically for processing data presented in the form of graphs. We chose GNNs because process models are generally represented by graphs with nodes representing activities, events, or states and edges representing dependencies or transitions between them. By leveraging the inherent structure of process models, graph neural networks can effectively capture the intricate dependencies among various activities. Moreover, we are the first to utilize GNNs for predicting the remaining cycle time in processes. We conducted a performance comparison of our GNN model against the benchmark LSTM model. The results demonstrate that our GNN model outperforms the reference LSTM for long and complex processes. These findings highlight the potential use of GNNs in process mining applications.

The research work presented in this manuscript is applied to the development of the smart factory for all Vitesco Technologies plants, with a particular focus on the production of electronic boards for electric and hybrid vehicles in Foix, France.

The automotive industry is constantly evolving, with technological, economic, and social changes. Recent developments include automation and digital transformation. The manufacturing process in this industry is complex and requires high precision and coordination between suppliers, manufacturers, and assembly plants. Understanding these processes in-depth using traditional methods is increasingly challenging. Therefore, advanced techniques that integrate information from different sources and apply large-scale data analysis are necessary to obtain meaningful insights. Companies that adopt advanced technologies such as robotics, the Internet of Things, and artificial intelligence can improve efficiency, reduce costs, and accelerate decision-making.

The data we processed come from the Foix plant of Vitesco Technologies, where electronic products are designed and manufactured on a large scale. The production operates 24/7 and produces tens of millions of products annually. The production includes four main phases: front end, ICT, back end, and packaging. Data are collected in real time throughout the process, from empty PCBs to assembled electronic boards. These data are stored in the form of event logs. To ensure the quality of delivered products and avoid non-conformities, the company needs to monitor the performance of machines and personnel, as well as production costs and cycle times. Monitoring is based on a detailed understanding of the process and the state of the production system. In this context, this thesis focuses on extracting knowledge from event logs, signals, and indicators to improve the production process. This analysis will enable decisions that lead to process optimization, planning, and maintenance.

This thesis is part of CIFRE program under the *"Collaborative AI : Synergistic transformations in model based and data-based diagnosis"* chair at ANITI. The research has been conducted through a collaboration between the Laboratory of Analysis and Architecture of Systems (LAAS) and Vitesco Technologies, situated in Toulouse, France.

Key words: Process Mining, Event logs, Predictive Analysis, Graph Neural Networks, Industry 4.0

INTRODUCTION

Manufacturing processes are crucial for organizations that produce goods and products for their customers. Generally, these processes are complex due to several reasons. Firstly, they involve multiple stages, resources, and dependencies between raw materials, machinery, and labor. Secondly, manufacturing processes often require a high level of customization and variability depending on the specific product being manufactured and the customer's requirements. Additionally, the use of technology, such as sensors, software, and automation systems, can improve production management, but it can also contribute to the process's complexity. As a result, understanding these processes thoroughly with traditional methods is increasingly challenging. To overcome this issue, advanced techniques that integrate information from various sources and apply analysis to a large amount of data must be used to obtain valuable insights.

One approach that has gained significant attention in recent years due to its potential to improve organizational efficiency and effectiveness is process mining. It is a subfield of data science that focuses on the analysis and improvement of processes using data, as shown in Figure 1. The field includes techniques from data mining, machine learning, and related areas to extract knowledge from data generated during process execution. There is an overlap between these subfields. The boundaries are not always clear-cut and may change over time. However, process mining is distinct in that it specifically aims to optimize processes. As a result, expertise in both data analysis and process modeling, as well as expertise in the specific domain, is necessary to effectively apply process mining techniques.

In recent years, process mining has gained popularity due to the increasing availability of data from processes and the need for businesses to optimize their processes for efficiency and cost savings. In addition, the growth of machine learning techniques and artificial intelligence has also contributed to the field's growth. Consequently, several research studies are being conducted in the field, and a comprehensive literature review is presented in Section 1.1. Simultaneously, many process mining tools have been developed, which provide users with functionalities to perform various types of process mining analysis on event data. Figure 2 provides an overview of these tools for both academic and

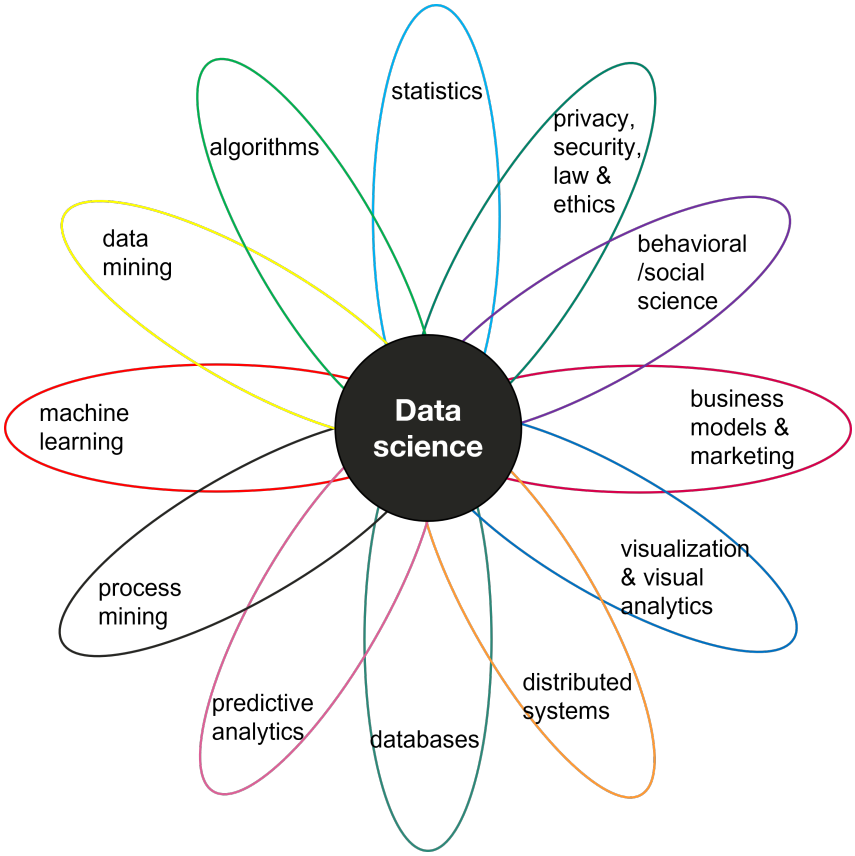


Figure 1 – The main ingredients contributing to data science from [106, p. 12].

commercial purposes. The availability of a variety of tools indicates a significant interest in the field, and many organizations have recognized the value of process mining for improving their processes and gaining a competitive edge.

In this thesis, we present our work in the framework of process mining with application to a real manufacturing process. The data used for process mining are stored in the form of an event log, which captures specific activities, their timestamps, and additional details such as resource or activity duration during process execution. Our work primarily focuses on two families of process mining techniques: descriptive and predictive process mining. Descriptive process mining involves analyzing event logs to extract knowledge from the process and identify areas for improvement, while predictive process mining concerns constructing learning models to predict the future behavior of the process.

During the descriptive process mining phase, we first extract raw data and perform necessary preprocessing. We conduct a comprehensive analysis to gain a global understanding of the production process and data structure. Then we use techniques related

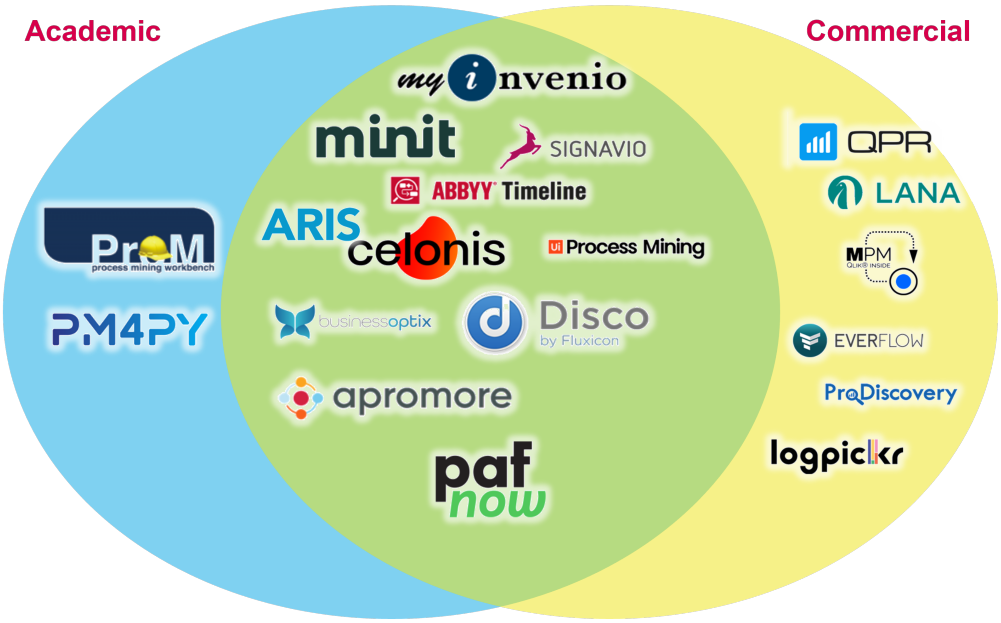


Figure 2 – Overview of Academic and Commercial Process Mining Tools (Source: `https://www.processmining-software.com/tools/`).

to process discovery, conformance checking, and performance analysis to gain insights into the process. The inefficient points are identified by analyzing the number of failures generated during each operation, product life cycle time, and transition time between operations. Based on this analysis, we propose a process mining framework to characterize product quality by comparing their production path with the nominal process model. Additionally, we conduct an analysis of production performance between work shifts using the quality index obtained from the previous shift.

In the predictive process mining phase, we develop learning models using event logs to predict the future behavior of the process for several prediction problems. We evaluate and compare the accuracy of the models ranging from simple regression to more advanced and complex models such as Long Short-Term Memory (LSTM) and Graph Neural Networks (GNNs). While LSTM is a well-known Recurrent Neural Network (RNN) that can model and predict sequential data, GNNs are a relatively new area of research that focuses on neural networks designed to work with graph-structured data. We choose GNNs because process models are commonly represented by graphs with nodes representing activities, events, or states, and edges representing dependencies or transitions between them. By leveraging GNNs, process mining can benefit from their ability to learn and generalize across graphs of different sizes and structures, particularly for analyzing complex process

models. Moreover, from the literature review presented in section 1.1.2, we find that there is limited research on GNNs in manufacturing and no GNNs application in our studied prediction problem regarding remaining cycle time prediction. Hence, our work presents a new method to apply to this problem and provides a significant perspective on using GNNs in manufacturing processes.

Our case study examines the manufacturing process of Vitesco Technologies, an automotive company that provides a wide range of products and solutions for internal combustion engines, hybrid vehicles, and electrified vehicles. The company operates in over 50 locations across 14 countries in Asia, Europe, and America, making it a global leader in the automotive industry, known for producing high-quality products for customers worldwide (as shown in Figure 3).

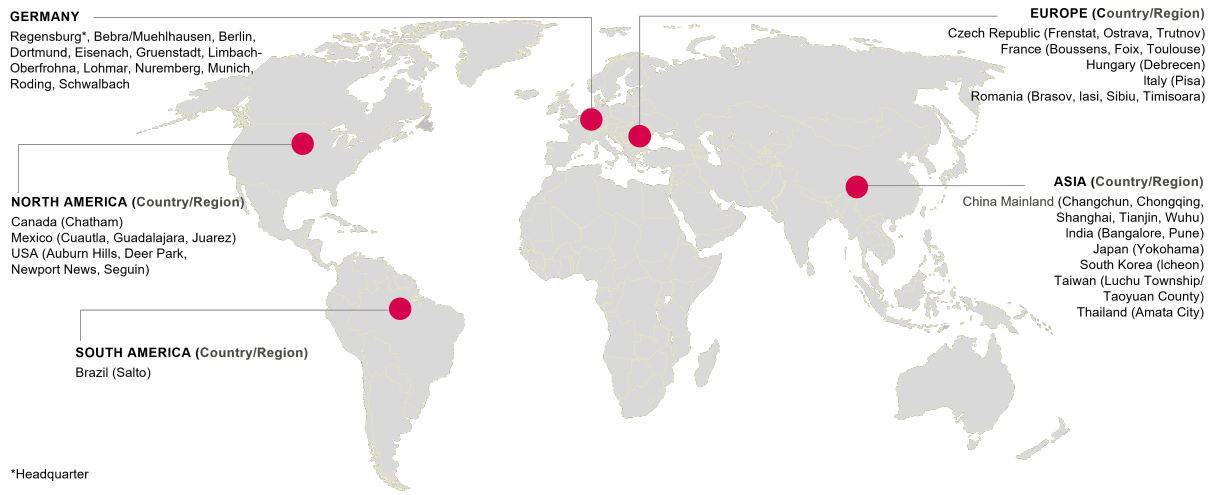


Figure 3 – Vitesco Technologies global footprint.

Specifically, we focus on the electronic board assembly process at the Foix plant in France, which we systematically describe in Chapter 2. Our findings aim to improve the manufacturing process by identifying areas for enhancement and predicting potential issues before they arise. By doing so, we can potentially save costs, improve efficiency, and enhance product quality. Additionally, our objective is to develop a method that is generalizable and can be applied to other plants within Vitesco Technologies and other similar production processes.

In summary, this thesis focuses on process mining techniques that are adapted and leveraged to be applied to a real manufacturing process and highlights the benefits of both descriptive and predictive process mining approaches. The results of this work can be used

to guide future efforts to optimize manufacturing processes and improve performance. The contributions of this thesis are as follows:

— Proposing a novel end-to-end process mining framework for computing a quality index for products from their production paths.
— Developing a method for analyzing performance across work shifts using the obtained quality indexes.
— Creating a tool for tracking the state of products and computing the work-in-progress (WIP) in real-time.
— Evaluating different models for the work-in-progress prediction problem.
— Proposing the first use of the GNNs model to predict the remaining cycle time of ongoing products during the production phase.

All these contributions are presented in the following chapters of the manuscript and organized as follows:

Chapter 1 presents a literature review of the two research topics investigated during the thesis: *Process Mining* and *Graph Neural Networks*.

Chapter 2 introduces the application context of the thesis, the automotive industry, and manufacturing in particular. The chapter provides a detailed description of the printed circuit board assembly process and related datasets used for the research.

Chapter 3 presents different case studies and techniques used in descriptive process mining. This includes three main tasks: assessing product quality, performance analysis, and work-in-progress calculation.

Chapter 4 presents two prediction problems: work-in-progress prediction and remaining cycle time prediction. Different models, including Graph Neural Networks, are used to solve these problems.

Finally, the last chapter provides conclusions and perspectives on our work for future research and application.

Chapter 1

State of the art

Summary of Chapter 1

This chapter presents a literature review of the two research topics investigated during the thesis. The first one is Process Mining which refers to a family of techniques that process data in the form of event logs. There are two main axes in this area: Descriptive Process Mining (DPM) and Predictive Process Mining (PPM). The study of the state of the art is conducted for each axis. Then, a literature review of process mining approaches in manufacturing is presented. The second topic is Graph Neural Networks (GNNs). GNNs refer to a family of neural networks that are developed to process graph data and resolve prediction problems related to graphs. Recent research has explored the use of GNNs in process mining. Such techniques have the potential to improve the accuracy and efficiency of the process mining approach. The implementation of GNNs can enable organizations to obtain valuable insights into their business processes. Consequently, this can empower them to make data-driven decisions, ultimately leading to enhancements in their operations.

1.1 Process Mining

"Process mining can be seen as a means to bridge the gap between data science and process science. Data science approaches tend to be process agnostic whereas process science approaches tend to be model-driven without considering the evidence hidden in the data [106, p. 15]". Especially, process mining is a field of study that uses data mining techniques to analyze processes based on event logs. By providing a visual and quan-

titative analysis, process mining helps organizations to understand how their processes are being executed, identify bottlenecks, inefficiencies, and compliance issues, and suggest ways to improve the process. Process mining techniques can be classified into two main families: descriptive and predictive [114, p. 369]. Descriptive techniques involve analyzing and visualizing the data from an existing business process to reveal insights into how the process is currently being executed. On the other hand, predictive process mining aims to predict future outcomes or behaviors of a process using data mining and machine learning techniques.

1.1.1 Descriptive process mining

DPM consists of data analysis techniques to model the processes and generate visualizations. It aims to understand and improve the performance of processes by analyzing data generated from various processes-related sources. DPM includes process discovery, conformance checking, and process enhancement (i.e., model extension, repair, etc.) (see Figure 1.1). Process discovery and conformance checking are the most important activities.

Process discovery involves automatically extracting process models from event logs. The technique analyzes a set of sequences or traces extracted from event logs to represent the process in the form of a flow diagram or other graphical models such as Petri nets or Business Process Modeling Notation (BPMN) diagrams. Many process discovery algorithms have been proposed in the literature. The α-algorithm [1] is the first and simple discovery technique that constructs causal relationships observed between tasks. It is based on the concept of a "footprint matrix", which is a matrix that represents the frequency with which different activities are performed in sequence. The α-algorithm works by building the footprint matrix and then using it to create a flow diagram that represents the process. For this, the algorithm begins by creating an initial set of nodes in the flow diagram, one for each unique activity in the event log. It then adds edges between the nodes to represent the sequence of activities in the process. The algorithm uses the frequency of each activity pair in the event log to determine the weight of the edges in the flow diagram. The α algorithm is known for its simplicity and efficiency [99]. However, it does have some limitations, such as its inability to handle noise and incompleteness. Noise in process mining refers to rare and infrequent behavior that is not representative of the typical behavior of the process. Incompleteness means that the event log contains too few events to be able to discover some of the underlying structures [106]. The Heuris-

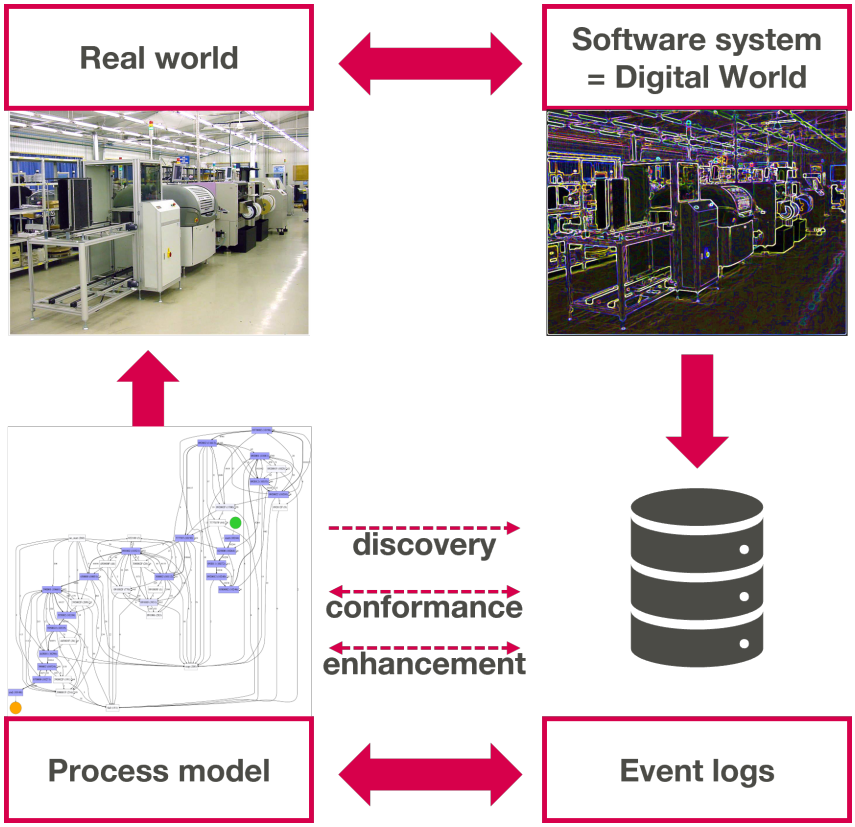


Figure 1.1 – Three main activities of descriptive process mining: *process discovery*, *conformance checking*, and *process enhancement*.

tic Miner algorithm is developed as an extension of the α-algorithm to address these limitations [118]. The algorithm uses causal nets [106] for process model representation. Moreover, the frequencies of events and sequences are taken into account when constructing a process model. Specifically, infrequent paths are filtered out by a user-predefined threshold, making it more robust than the α-algorithm. However, this algorithm is not the best choice to handle the event logs with a high degree of concurrency. Concurrency refers to the fact that multiple tasks can be performed at the same time in the process. This drawback is overcome by the Inductive Miner algorithm [54]. The algorithm works by repeatedly finding a split in the event log into smaller event logs. The procedure is repeated until a base case (sub-log with only one activity) is reached. The first Inductive Miner is presented in [54], then [55] proposed an adaptation of this algorithm to deal with incomplete logs, and [56] solved the problem of scalability with quality guarantees. These variations make it a powerful tool for process mining. However, like any algorithm,

it has some limitations. One limitation is that it may not be well suited for event logs that do not contain a high degree of concurrency, as it is specifically designed to handle such cases. Additionally, Inductive Miner may be computationally intensive, depending on the size of the event log, which can make it less practical for large-scale process mining projects. Another approach used in process discovery is the Genetic Miner [65]. In this case, a genetic algorithm is used to iteratively evolve process models. It generates a population of possible process models and uses evolutionary techniques to improve the process model over time until it finds an optimal model that fits the event log. Recently, neural networks have been used for process discovery [98]. However, these methods are usually time-consuming for the training process and are not always explainable.

Conformance checking refers to the analysis of the consistency between the behavior of a process described in a process model and event logs that have been recorded during the execution of the process [17]. This work brings important notions [84], like:

— *fitness* measures the extent to which log traces can be associated with execution paths specified in the process model: a model fits a log if all traces in the log can be replayed by the model. Fitness can be calculated using various measures, such as the fraction of events in the log that can occur according to the model or the fraction of traces in the log that can be fully replayed, etc.

— *appropriateness* provides the degree of accuracy and clarity in which the process model describes the observed behavior. This notion is composed of two parts: structural and behavioral appropriateness. Structural appropriateness refers to how minimal the model structure is to reflect the behavior clearly. At the same time, behavioral appropriateness analyzes the balance between overfitting and underfitting.

Conformance checking is useful in several tasks, among them performance analysis [5] and deviations analysis [6]. There are several techniques for conformance checking in process mining that depend on the assumptions made about the process model and its actual behavior. In particular, one may assume that the process model is correct and seek to verify whether the actual behavior matches the expected behavior. On the other hand, some may assume that the observed process log may not adhere to the exact sequence of activities specified in the process model or may demonstrate different behavioral patterns compared to the process model. Basically, two general approaches corresponding to these two assumptions are considered: log replay and trace alignment [17, 2].

— *Log replay* involves replaying the events recorded in an observed event log on a

given process model to check if the model can accurately reproduce the log. One example of this approach is the token-based log replay method described in [83]. Although log replay is time-efficient, it assumes the process model is accurate. Therefore, any events related to activities that are not included in the process model cannot be considered during the replay process.

— *Trace alignment* is the approach that tries to find the best alignment between traces from the event log with the process model. Once the alignment is computed, it can identify any deviations between the observed behavior and the expected behavior and suggest possible improvements to the process model.

In addition to the two presented approaches, there are conformance checking techniques based on rule checking presented in [17]. Furthermore, the authors in [26] demonstrate that current techniques focus solely on the process control flow and suggest that future research should consider additional perspectives such as time, resources, and others.

Conformance checking is an important aspect of process mining as it enables organizations to ensure that their processes are being executed according to the expected behavior and identify deviations that may indicate inefficiencies, compliance issues, or other problems. It is important to note that conformance checking is not only limited to comparing the event log with the process model, but it can also be done with other forms of reference models such as performance indicators, regulations, or policies.

Process enhancement is an essential aspect of process mining that involves using the insights obtained to extend or improve an existing process [106]. Process improvement consists of identifying bottlenecks and inefficiencies and suggesting ways to optimize the process. There are several approaches to improve processes, such as performance analysis, case duration analysis, or performance prediction [40]. Process extension refers to adding perspectives extracted from the event log to the control-flow model, such as organizational, time, or other perspectives. Furthermore, processes can be repaired or redesigned by removing unnecessary steps, consolidating activities, or introducing new activities.

Process enhancement is a critical aspect of process mining. Organizations can increase process efficiency, reduce costs, and improve overall performance by using the information gained from process mining to optimize processes. It is important to note that process enhancement is an ever-evolving process that requires constant monitoring and analysis to identify new improvement opportunities and to adapt to changing business conditions. In addition, it is beneficial to involve domain experts in process improvement activities to ensure that proposed solutions are aligned with business objectives and are achievable.

1.1.2 Predictive process mining

Predictive process mining in general

PPM is a discipline that uses advanced data analytics and machine learning algorithms to analyze process data and make predictions about future process behavior. By discovering patterns and trends in process data, PPM allows organizations to proactively identify potential problems and opportunities within their processes, making informed decisions and improving process performance. Several research have been conducted on the topic of PPM. These can be categorized into different classes depending on the criteria, i.e., PPM problems, prediction model classes, application domains, etc.

When it comes to prediction problems, there are various examples to consider. For example, we can predict the next activity or sequence of activities of an ongoing (uncompleted) process [22]. In [80] Rama-Maneiro et al. describe different prediction problems within the context of the PPM. Specifically, given a sequence of events for a running process instance, i.e., the prefix, the PPM aims to forecast various attributes of the next event or sequence of events, i.e., the suffix. The PPM also involves predicting the output of a running process and the remaining time, i.e., the time required to complete it.

Regarding the prediction models, the authors in [64] classify the approaches into two families, process-aware and non-process-aware methods. Process-aware methods consider the structure and behavior of processes when making predictions. These methods use process models and data to predict future process behavior. For example, process-aware methods may use process mining techniques to discover process models and then use these models to make predictions about future process outcomes. Non-process-aware methods, on the other hand, do not take into account the structure and behavior of processes when making predictions. Indeed, these methods only use data from the process to analyze and make predictions. For example, non-process-aware methods may use machine learning algorithms to analyze process data and predict future process outcomes.

Both process-aware and non-process-aware methods have their advantages and disadvantages. Process-aware methods are more accurate but may be more complex and time-consuming to implement. Non-process-aware methods are simpler and faster to implement, but they may be less accurate as they do not consider the process structure and behavior.

Di Francescomarin et al. [23] review existing methods for the PPM based on prediction type, input data required, tool support, the validity of the algorithm, and the family of

algorithms to help industries select the method that best suits their problem. In recent years, deep learning has been widely exploited in the PPM due to its promising results against classical methods [80]. The authors in [37] focus only on deep learning approaches based on Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Stacked Autoencoder. A more complete review of the state-of-the-art deep learning methods in PPM is presented in [70, 80]. These reviews are followed by a benchmark of the approaches for the next activity and suffix prediction. Pasquadibisceglie et al. [72] use a 2-D Convolutional Neural Network (CNN) based on Inception blocks for next-activity prediction. Recently, Williams Rizzi et al. [82] present a toolkit, namely Nirdizati, that offers a useful and flexible instrument for investigating and comparing PPM techniques.

Remaining cycle time prediction

In this thesis, we focus on predicting the Remaining Cycle Time (RCT), and we hence conduct a state-of-the-art about methods used to achieve this. The RCT prediction approaches in the PPM are classified based on input data, data preprocessing (encoding) method, process awareness, the family of algorithms, etc. According to input data, all studies handle the event log with at least a case id, an activity, and a timestamp. Furthermore, many methods use additional attributes of case and event to feed the prediction models [24, 78]. Others use contextual information to obtain more accurate predictions [92, 29]. Most process-aware methods discover process models from event logs because the model is not always known and may be different from the real behavior. A transition system is constructed and used to predict the remaining time in [107, 11, 78, 77]. Meanwhile, Verenich et al. [112] make a prediction based on the process tree obtained from historical traces. Non-process aware approaches usually apply machine learning algorithms to learn a model from labeled training data, i.e., supervised learning [111]. Several regression models can be used such as linear regression [4], random forest [101], XGBoost [92] and neural networks [28, 103, 66, 102, 69, 16, 48, 116].

The authors in [111] present a systematic review and a benchmark of the existing methods for the RCT prediction problem. The results show that LSTM-based networks achieve the best accuracy in terms of Mean Absolute Error (MAE). Over the last few years, GNNs have also been used to solve PPM problems. Philipp et al. [74] present the first use of GNNs to predict the amount of disbursement per area regarding a process of application request. They develop a model that contains two graph convolutional layers followed by one linear layer. Venugopal et al. [110] present a comparison of Graph Convolutional

Network (GCN) with the CNN and LSTM models along with a Multi-Layer Perceptron (MLP) for the next activity and timestamp prediction. In addition, a Gated Graph Neural Network (GGNN) is used in [119]. The authors in [81] build a process model in the form of a Petri net, then feed it to a model that integrates GCNs and RNNs. Whiorrini et al. [18] develop a GNN model that performs well in the event log with a high presence of parallelism. However, all of these works only focus on the problem of the next activity and timestamp prediction. There have been no studies so far about GNNs on the RCT prediction. As such, one of our studies aims to address this gap in the literature by focusing on the use of GNNs to predict RCT outcomes. By exploring the potential of this approach, we want to enhance the accuracy and efficiency of RCT prediction models, as well as contribute to the advancement of PPM in general.

1.2 Graph Neural Networks

Graphs are a powerful tool to represent data because of their capability to depict complex relationships and interactions between entities. The versatility of graphs has led to their widespread use. They have been applied to a wide range of sectors, including Social Network Analysis, Computer Vision, Natural Language Processing, Cheminformatics, Bioinformatics, and many more. The abundance and complexity of graph data, along with the increase in computation capacity, has led to a significant increase in the use of deep learning models for handling graph-structured data. These models are referred to as Graph Neural Networks (GNNs). GNNs are different from other neural networks in the way that operated data is not a regular grid. Graphs are discrete objects that can differ in size and structure. Furthermore, information about node identity and ordering across multiple samples is not always accessible. These characteristics pose challenges in terms of expressiveness and computational complexity for learning compared to vector-based data [8].

Back to the history of GNNs, Sperduti and Starita made the first proposal of neural networks for graphs in 1997 [100]. They present the use of recursive neural networks for processing directed acyclic graphs, which inspires early research on GNNs. The notion of GNNs is initially outlined in [33], which extends recursive neural networks to general graphs, including directed, undirected, labeled, and cyclic graphs. Then in 2009, two GNN models were proposed to tackle the problem of mutual dependencies between variables in the cyclic graphs [87, 67]. Since 2012, several breakthroughs in the performance and

accuracy of deep neural networks have attracted the attention of research communities. Similar to the progress of deep learning in general, a wide range of GNN architectures have been proposed, including GGNN and GCN, etc. These architectures can be categorized in multiple ways based on different criteria. In the following paragraph, we will describe the taxonomy of GNNs presented in Figure 1.2.

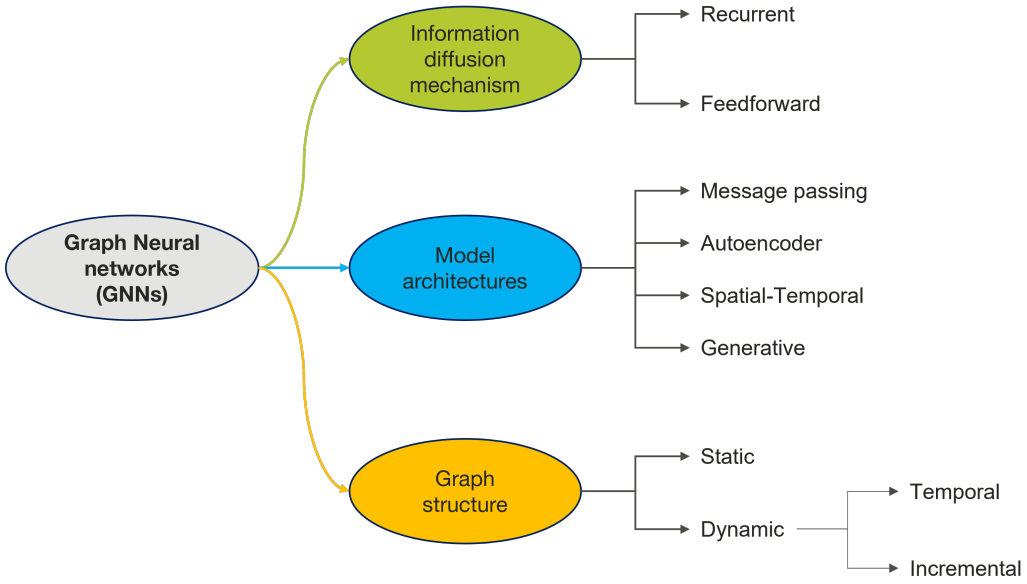


Figure 1.2 – A taxonomy of Graph Neural Networks.

The primary role of GNNs is to learn a relevant representation of graphs, including the graph's structure and the relationships between its nodes for the downstream tasks. These can be nodes, edges, or graphs-related tasks. However, most GNN models, regardless of the specific training objective, ultimately output node representations [8]. Indeed, a graph representation can be obtained by aggregating its node representations using pooling operations such as max-pooling, sum-pooling, or neural networks. Pooling is a common operation used in CNNs to reduce the dimension of feature tensors after convolutional layers. In GNNs, pooling is used to reduce the number of nodes and also to compute the graph embedding. For example, max-pooling involves selecting the maximum value within a group of nodes, or all nodes in the case of graph embedding, to represent the group or graph. The GNNs learn node representation by assigning each node an initial state representing its local information. Then, this information is spread out across the graph under specific diffusion mechanisms to update the state of its neighbors. In the meantime,

the state itself is updated by incorporating information from neighboring nodes. It is repeated multiple times to refine the representations of the nodes and edges in the graph. This process has been abstracted to a general framework called Message Passing Neural Networks (MPNNs) [32]. Two components that constitute the fundamental building blocks of MPNNs are the message function and the update function. The message function, when applied to a node, computes a representation of the node based on its interactions with its neighboring nodes. Subsequently, the update function utilizes this representation, in conjunction with the original node features, to update the feature representation of the node. Different architectures of GNNs are mostly based on the variant of the message function and the update function.

Regarding the information diffusion mechanism, there are two main approaches to define the message-passing process: *recurrent* and *feedforward*.

— The *recurrent* approach is based on the idea of recursively aggregating information from the neighborhood of a node.
One of the earliest recurrent GNN architectures is the Graph Recurrent Neural Network (GRNN) proposed by Scarselli et al. in 2009 [87], in which the authors propose a generalization of RNNs to graph-structured data. They introduce a message-passing mechanism, where the hidden state of a node at time step t is a function of the hidden state of the node at time step $t-1$ and the messages received from its neighboring nodes at time step t. Later, GGNN is introduced by Li et al. in 2015 [58]. In this architecture, the authors introduce a gating mechanism to control the flow of information between the nodes, which is similar to the gating mechanism used in Gated Recurrent Units (GRUs) and LSTM. Another recurrent GNN architecture is the Graph Attention Networks (GATs) proposed by Velickovic et al. in 2018 [109, 13], where the authors propose an attention mechanism to weigh the contribution of each neighbor to the new representation of the node. This architecture extends the traditional recurrent approach by incorporating an attention mechanism to weigh the contribution of each neighbor to the new representation of the node. Recently, Recurrent Graph Convolutional Networks (R-GCNs) have been proposed by Schlichtkrull et al. [88], where the authors propose a recurrent approach where the message-passing process is based on graph convolutions, which allows modeling the interactions between the nodes in a graph.
In conclusion, the recurrent approach in GNNs has been extensively studied in the literature, with several architectures proposed that generalize the traditional

recurrent neural network to graph-structured data. The recurrent approach allows GNNs to handle graphs with arbitrary topologies, and it can be used in various types of GNNs.

— The *feedforward* approach is based on the idea of applying a fixed number of layers of neural networks to the nodes and edges of the graph without recursion. In this approach, the representations of the nodes are updated in parallel, based on the representations of the nodes in the previous layer and the messages received from the neighboring nodes.

One of the earliest feedforward GNN architectures is the GCN proposed by Kipf and Welling [49]. In this architecture, the authors propose a convolutional operation on graphs, where the convolutional filters are applied to the nodes of the graph, and the convolution is defined as a symmetric normalization of the node features followed by a linear transformation. Another feedforward GNN architecture is the GraphSAGE proposed by Hamilton et al. [35], where the authors propose a generalization of the convolutional operation to graphs by sampling and aggregating the features of the node's neighborhood. The feedforward approach is simpler to implement and computationally more efficient than the recurrent approach, but it cannot handle graphs with arbitrary topologies, and it is typically used in cases where the graph structure is known and fixed.

Regarding the architecture and the complexity of the model, we can classify GNNs into four categories: *Message Passing GNNs*, *Graph Autoencoders*, *Spatial-Temporal GNNs* and *Generative GNNs*.

— *Message Passing GNNs* are approaches that utilize a message-passing mechanism to update the representations of the nodes in a graph. Examples of this category include GCNs [49], GATs [109], and GGNNs [58].

— *Graph Autoencoders* employ a graph autoencoder architecture to learn the latent representation of a graph. The graph is first encoded into a lower-dimensional space and then decoded back to the original space. The main objective of this architecture is to preserve the graph structure during the encoding-decoding process. Examples of this category include Variational Graph Auto-Encoders (VGAE) [50], and GraphVAE [97].

— *Spatial-Temporal GNNs* are designed to handle spatio-temporal data, such as videos or sensor data, by extending the graph representation to include the time dimension. These models are commonly used in tasks such as action recognition, anomaly

detection, and video prediction [14, 57, 34].

— *Generative GNNs* are used to generate new graph samples by learning the underlying probability distribution of graph data. These models can be applied to tasks such as graph generation, link prediction, and graph completion [126, 41].

An important aspect to consider in GNNs is handling changes in the graph structure over time. For this, they distinct into two categories *Static GNNs* and *Dynamic GNNs*. *Static GNNs* treat the graph as fixed and unchanging over time. They process the graph once and produce a fixed representation of the graph. These models are suitable for tasks that involve analyzing a single snapshot of a graph, such as node classification, graph classification, and graph generation. *Dynamic GNNs* handle changes in the graph structure over time by updating the graph representation as new edges and nodes are added or removed. These models are suitable for tasks that involve analyzing the evolution of a graph over time, such as node classification in evolving networks, link prediction, and node representation learning. Dynamic GNNs can be divided into two categories: temporal GNNs and incremental GNNs. Temporal GNNs process the entire sequence of graph snapshots and use RNNs or attention mechanisms to capture the evolution of the graph. Incremental GNNs update the graph representation incrementally as new edges and nodes are added, and they are more computationally efficient than temporal GNNs. The choice of which type of GNN to use depends on the specific task and the characteristics of the graph-structured data. Dynamic GNNs are more complex and computationally expensive than static GNNs, but they are necessary for tasks that require capturing the evolution of the graph over time.

1.3 Applications of Process Mining and Graph Neural Networks

As previously described, the field of process mining is concerned with extracting useful information about process execution by analyzing event logs [2]. The research area of process mining is large and leads to a wide range of applications. For example, process mining techniques have been applied to support the invoice handling process in a road construction and maintenance company [3], for organizational mining to discover social networks and to optimize the underlying processes [99], or in the healthcare environment to improve the quality of patient care [73].

Process mining was originally developed for Business Process Management (BPM),

so there have been many applications in real business processes. [75] processes event logs to configure process risk indicators (PRIs) that predict process delay. [45] presents an overview of existing discovery techniques for the construction of high-level Business Process Model and Notation (BPMN) models. There are also the applications of process mining in healthcare [63]. There have been fewer applications of process mining in the manufacturing industry. Valencia Parra et al. implement a prototype that extracts complex semi-structured data from an assembly-aircraft process and transforms it into an event log to utilize process mining tools [105]. Yang et al. propose a system architecture to use both structured and unstructured data to discover process models and analyze the performance [124]. Process mining techniques have also been applied in the analysis and prediction of manufacturing costs [39]. The heuristic miner algorithm is applied in [85] to obtain the process model, then used for maintenance inspection interval optimization.

GNNs are widely used in various fields due to their ability to model relationships between entities in a graph. In recommendation systems, GNNs are popularly used to model user-user and user-item interactions, with nodes representing users and items and edges representing interactions [123, 27, 125]. GNNs are also used in computer vision to model relationships between objects in an image and predict object interactions [43, 79]. In natural language processing, GNNs are useful for modeling relationships between words in a sentence or documents and making predictions based on those relationships [122, 59]. Moreover, GNNs show promising results in predicting the properties of molecules, which is essential in drug discovery in cheminformatics [120, 42]. In the context of manufacturing, Seito et al. use reinforcement learning with graph convolutional networks to create a dispatching rule independent of human expertise for production scheduling [91]. Another study in [46] presents a new method based on GNNs to improve the prediction of failure in missing value conditions from a home appliance manufacturer. In [61], the authors propose a new approach for online planner selection using GNNs, which are advantageous over other methods due to their invariance to node permutations and incorporation of node labels. They also propose a two-stage adaptive scheduling method to improve cost-optimal planning, and the experimental results show the effectiveness of the proposed method.

The primary objective of process mining and GNNs in the manufacturing industry is to improve production efficiency by identifying and eliminating bottlenecks and inefficiencies. Process mining, for instance, can uncover delays by analyzing production logs, which can then be rectified to increase overall production speed. Similarly, GNNs can

detect potential failures, enabling predictive maintenance. Another critical aspect is enhancing product quality. Through analysis of production logs, correlations between certain process parameters and product quality can be detected, providing insights that can be leveraged to improve the process and maintain consistent production of high-quality products. Additionally, process mining and GNNs can be integrated to optimize supply chain management in the manufacturing sector. Analyzing delivery times and quantities of raw materials and finished products can help to pinpoint areas for improvement in the supply chain [44, 53, 51].

1.4 Conclusion

In conclusion, process mining has been an active area of research and development in recent years. It has been widely applied in various domains and industries, including manufacturing, healthcare, finance, and IT. The literature review shows that process mining encompasses several tasks, including process discovery, conformance checking, process improvement, and process prediction. There are various process mining techniques available, ranging from process model-based to machine learning-based methods, and each technique has its strengths and weaknesses. Additionally, the literature review shows that process mining has been used with other related technologies, such as data mining and machine learning to achieve better results. Furthermore, process mining is shown to be a valuable tool for organizations looking to improve their processes and increase efficiency. The increasing demand for process mining and the continuous development of new techniques and applications highlight the importance and relevance of process mining in today's business world.

The analysis of the literature reveals that deep learning-based predictive process mining has made significant progress in recent years. The use of neural networks enables the identification of intricate relationships and dependencies among various process variables. Among several neural network architectures, GNNs have exhibited substantial potential owing to their ability to process input as a graph. Research in this domain is expanding at a rapid pace and has garnered significant attention in recent years. GNNs offer a potent tool for processing graph-structured data and have shown promise in numerous applications, such as node classification, graph classification, and graph generation. GNNs are still evolving, and new techniques and models are being developed rapidly. Several GNN models can be categorized in different ways based on various criteria, each with its own

advantages and disadvantages. The choice of which one to use depends on the specific task and the characteristics of the graph-structured data. There are also various challenges and open problems in GNNs, including scalability, interpretability, and generalization. However, progress in this area of research is ongoing, and new techniques and models will likely be developed to overcome these challenges.

Overall, process mining and GNNs are two emerging technologies that hold great promise for the manufacturing sector. The potential of their combination lies in their ability to offer valuable insights into production processes, minimize waste, and enhance overall efficiency. This thesis effectively demonstrates the practical application of process mining techniques and GNNs in a real-world scenario by exploring an electronic board manufacturing process at Vitesco Technologies. The findings of this study offer promising prospects for the future utilization of these technologies in the manufacturing industry.

Chapter 2

Context & Case study

Summary of Chapter 2

This chapter provides an introduction to the application context of the thesis, the automotive industry, and particularly on manufacturing. The chapter discusses how technological advances present both opportunities and challenges for the industry and highlights the importance of proactively adopting and leveraging these technologies to increase competitiveness in a constantly changing and challenging market. The chapter then introduces the case study of Vitesco Technologies, an automotive company, which is studied in the thesis. It provides a detailed description of the printed circuit board assembly process and related datasets, which are used for the research. In conclusion, the chapter explains that the production process and the vast amount of event log data have motivated the study of process mining and process-related methods as the core of the research in the thesis.

2.1 General context of the automotive industry

The automotive industry is a complex and dynamic sector that plays a crucial role in the global economy. The industry encompasses vehicle design, manufacturing, marketing, and sale. Over the years, the sector has undergone significant transformations, driven by factors such as technological advancements, shifting consumer preferences, and changing regulations. Indeed, the automotive industry is currently undergoing several evolutions, including electrified vehicles, autonomous and connected vehicles, zero-emission vehicles, and carbon-neutral manufacturing. All these trends are both opportunities and challenges

for the industry. In the highly competitive automotive market, traditional car manufacturers are now competing with tech companies and startups in electrified and autonomous vehicles. Governments worldwide heavily regulate the industry with stringent emission standards and safety requirements, making it crucial for companies to comply. The automotive industry also relies heavily on a global supply chain encompassing numerous suppliers and manufacturers, leaving it vulnerable to disruptions such as natural disasters or political conflicts, which can significantly impact production and costs. The recent Covid-19 pandemic, the blockage of the Evergreen ship in the Suez Canal, and the conflict between Ukraine and Russia have all been poignant examples of the difficulties companies face today. In conclusion, the automotive industry presents a unique set of opportunities and challenges. Companies must navigate this complex landscape by embracing technological advancements, complying with regulations, and staying ahead of the competition. By doing so, they can ensure continued growth and success in this exciting and dynamic sector.

Manufacturing is a critical aspect of the automotive industry and involves the production of vehicles and their components. The automotive manufacturing process is complex and requires a high level of precision and coordination between suppliers, manufacturers, and assembly plants. Companies that manage their manufacturing processes effectively will be better positioned to meet the market's demands and achieve long-term growth and success. The solution is to offer high-quality and customized products with optimized cost and short production time [15]. At the Hannover Fair in 2011 in Germany, the concept of "Industry 4.0" was introduced to describe the creation of a new industrial revolution that brings together virtual and physical manufacturing systems for increased flexibility [89]. The objective is to establish smart factories that are highly automated and data-driven, to increase efficiency, reduce costs, and improve decision-making. Many manufacturing industries are adopting advanced technologies such as robotics, 3D printing, and artificial intelligence to achieve this. Additionally, they are undergoing a digital transformation by utilizing digital twins, cloud computing, and other digital tools to enhance supply chain management and improve decision-making. Data have become a critical resource in this era, and companies are investing in infrastructure to collect, store, and analyze vast amounts of data to make informed decisions and improve processes. The release of ChatGPT, a conversational language model developed by OpenAI, has recently caused a stir in the community. With its in-depth knowledge of various fields and its ability to respond in a manner that resembles human speech, ChatGPT has marked a new milestone in the

development of machine learning and AI. Scientists and companies are considering the limits of this development and how they can leverage it to reduce costs, increase production capacity, and improve competitiveness. Soon after the release of ChatGPT, Google introduced its AI chatbot named Bard. The Chinese tech company Baidu announced its plan to launch an AI chatbot on Mars in 2023, and other tech giants such as Meta and Apple are racing to develop generative AI similar to ChatGPT to keep pace with the boom. In the era of rapid technological development, manufacturers that can effectively adopt and leverage these technologies will be better positioned to succeed in the era of Industry 4.0.

This thesis project is in the context of the development of the factory of the future for all Vitesco Technologies plants. The studied plant in Foix, France, focuses on producing electronic boards for electrified vehicles.

2.2 Case study

Electronic boards, also known as Printed Circuit Boards (PCBs), are crucial components of modern vehicles. They control various functions and systems in the vehicle, such as engine management, navigation, and safety systems. The manufacturing of electronic boards for the automotive sector is a specialized field that requires high levels of precision, reliability, and quality. Automakers must work closely with suppliers and manufacturers to ensure that the boards are of the highest quality and meet the required performance criteria. Companies that can effectively manage the manufacturing of electronic boards will be better positioned to succeed in the highly competitive automotive market. At Foix plant in France, Vitesco Technologies designs and manufactures high-volume electronic and mechanical products for the automotive industry. The production runs 24/7 and produces tens of millions of products annually. The quality of the delivered products, the rejection rate, the availability and performance of machines and personnel, and the time and cost dimensions are key performance factors of a production site. Non-conformities can arise from issues with the products, handling, equipment, and operations coordination over time. These require a detailed understanding of the process and state of the manufacturing system. The main objective of this thesis is to extract knowledge from the analysis of logs, signals, and indicators to build and update explicit knowledge. This explicit knowledge serves as a basis for making decisions that will lead to the optimization of the process, including control actions, replanning, and maintenance actions, as well

as the optimization of production through the sequencing of batches and products. The following paragraphs present the studied assembly process and the dataset.

2.2.1 The PCB assembly process

The use case of this study concerns the PCB assembly process given in Figure 2.1. This process is divided into four main phases:

— Front End assembly (FE): electronic components are placed and soldered onto the PCB.

— In Circuit Test (ICT): PCBs with mounted electronic components are tested to verify their electrical performance before they are fully assembled into the final product.

— Back End assembly (BE): connectors are added to the electronic boards, and the whole is covered by the housing.

— Packaging: final electronic boards are packaged into different boxes following the client's instructions for delivery.

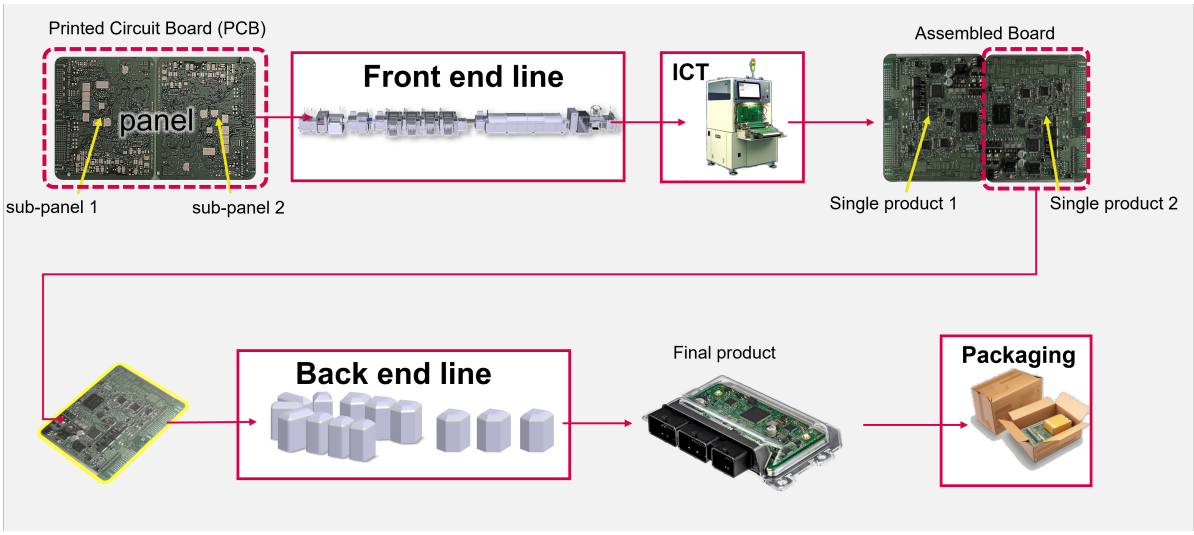


Figure 2.1 – A schematic view of PCB assembly process.

In the *FE*, the electronic boards are assembled using Surface Mount Technology (SMT). A schematic view of a SMT line is presented in Figure 2.2. SMT refers to a method of mounting electronic components in which components are mounted or placed directly onto the surface of the PCB. This method is more efficient in terms of production time and costs than the older through-hole technology, where components are inserted into

holes drilled into the PCB. SMT is a highly automated process, with many of the steps performed by machines, such as component placement, soldering, and inspection. This results in greater accuracy and repeatability. Hence, SMT allows for mounting miniaturized components, allowing a higher component density. Besides, through-hole technology is still preferred for high-reliability and high-power products, as the connection provided by this method may offer superior mechanical stability and better heat dissipation.

SMT lines typically consist of several steps. In the first process, Solder Paste Printing (SPP), solder paste is injected through a stencil mask to create a solder layer over the PCB that forms the primary interconnection basis between the components and connection pads. Next, the solder deposit quality is verified by the Solder Paste Inspection (SPI) machine. In this process, several properties of the pad, such as volume, area, height, and position, are measured and verified according to the standard limits. PCBs that are judged as good continue through the placement process. There, electronic components are mounted directly onto the surface of the PCB. At this point, the joints are still in a liquid form, so the components float in their position but are not firmly attached to the PCB. This happens in the next step, called reflow.

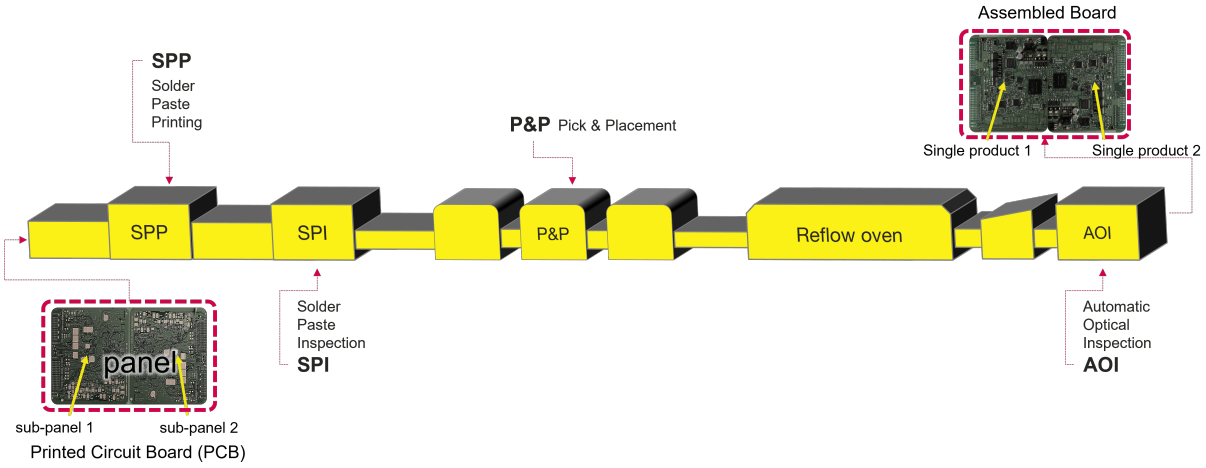


Figure 2.2 – Surface Mount Technology line.

The reflow oven is divided into several chambers allowing the PCBs to be conveyed in a specific temperature profile. Temperatures in each part are strictly controlled to guarantee the quality of the solder pads and the integrity of the placed components. The FE phase ends with an Automated Optical Inspection (AOI) that checks the components' final position and solder quality. The FE phase is then repeated on double-sided boards. In the rest of this paper, the front end process on the first side of PCB is denoted as $FE1$

and $FE2$ for the second side. It is important to notice that each PCB can contain one or more sub-panels depending on the product type. In the FE phase, the whole panel passes through processes and is modified or controlled at the same time.

The *ICT* phase involves using a designed test fixture to make contact with the board's test points. Then, a test program measures and verifies each component's operation to detect different types of defects, such as missing or incorrectly installed components, short circuits, and open circuits. Products that fill the required quality standards are passed through the BE phase.

The *BE* phase starts by cutting the PCB into single products. In this phase, the connectors that communicate with peripherals are fixed to the PCB, and the set is assembled in a housing for protection and thermal dissipation. The final product is ready to be packed after performing the functional tests verifying product operation. It is worth noting that, contrary to FE lines that have a standard configuration, BE lines are suited to the specific needs of each product and, in many cases, employ a combination of human operators and specialized robots [94].

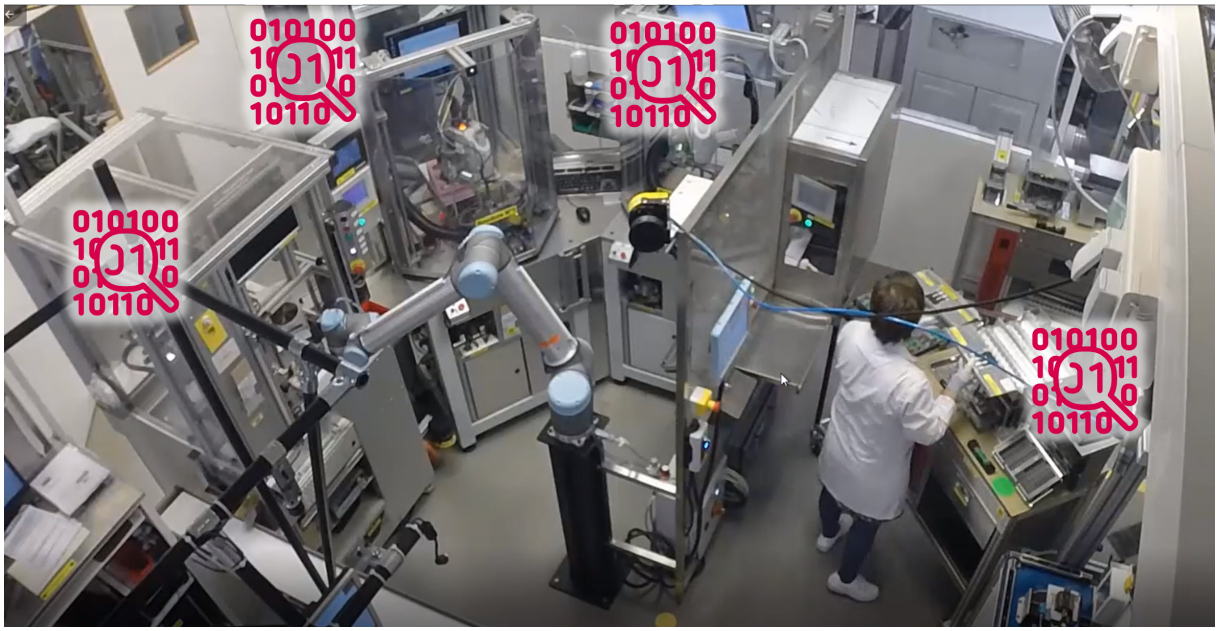


Figure 2.3 – Back end process.

The packaging phase is the last step before products are transported to the clients. This phase involves placing the final products into different boxes according to the orders. Both the boxes and the products inside them are tracked and stored in the system to keep

track in case of damage during the packaging or transportation process.

Although several tests check the quality of solder pads and the function of electronic components during the production process, i.e., SPI, AOI, ICT, etc., more is needed to evaluate the global quality of final products. When assembling PCBs, processing times are one of the most important factors affecting product quality. Indeed, throughout the assembly process, the PCBs are exposed to several factors affecting the reliability of electronics. Among the well-known factors are temperature, relative humidity, and dust. The impact of these factors on the reliability of electronics increases as the size of electronics reduces. Some moisture-sensitive components can be damaged during reflow when moisture trapped inside the component expands. These sensitive components are always sealed in air-tight packaging, including the Moisture Sensitivity Level (MSL) information, i.e., the time period in which a moisture sensitive device can be exposed to ambient room conditions. Most of the damages caused by moisture, e.g., delamination, die damage, internal cracks, etc., are not visible on the component surface and, therefore, not detected by visual inspection processes. In order to prevent moist-related damage, the MSL should always be respected, and therefore the assembly processing times should be under control.

At Vitesco Technologies, between process phases, production is organized in batches. *Batch production* involves the fabrication of a finite number of products within a time frame. A batch can undergo a series of steps in a large manufacturing process to make the desired product [52, 121]. Products within a batch are generally identical and belong to the same product family and reference, as in our case study. The batch concept aims to ensure that the products are assembled under nearly identical conditions and configurations, resulting in a homogeneous final quality. This also makes the process both time- and cost-efficient. There is a break between two consecutive batches for product recharging and reconfiguration. This time range varies depending on the batch size and the product type. In our case study, batches can be retrieved automatically from event logs by identifying inactive periods, as this information is unavailable in the collected data. An idle period is supposed as a time interval in which no product passes along the line. Besides, it is noteworthy that, even though products in a batch are produced under the same configuration, their flow path may vary due to operational failures.

2.2.2 Dataset

In the context of Industry 4.0, data play a central role and are generated at every stage of the production process. Since 2017, Vitesco Technologies has implemented a project to

collect data in a structured manner and store them in one location. Figure 2.4 presents the cloud architecture developed in this project. The data collection is performed directly from all machines or by the Manufacturing Execution Systems (*MES*). Then, the NiFi data broker collects and routes data from various sources, such as sensors or log files, to a centralized location. Vitesco Technologies uses a cloud storage service such as the one from Amazon to manage these datasets. The collected data, which are stored in Simple Storage Service (S3), are then passed to lambda functions which perform various tasks, such as data transformation and filtering. Finally, the processed data are extracted and executed for further analysis and downstream applications.

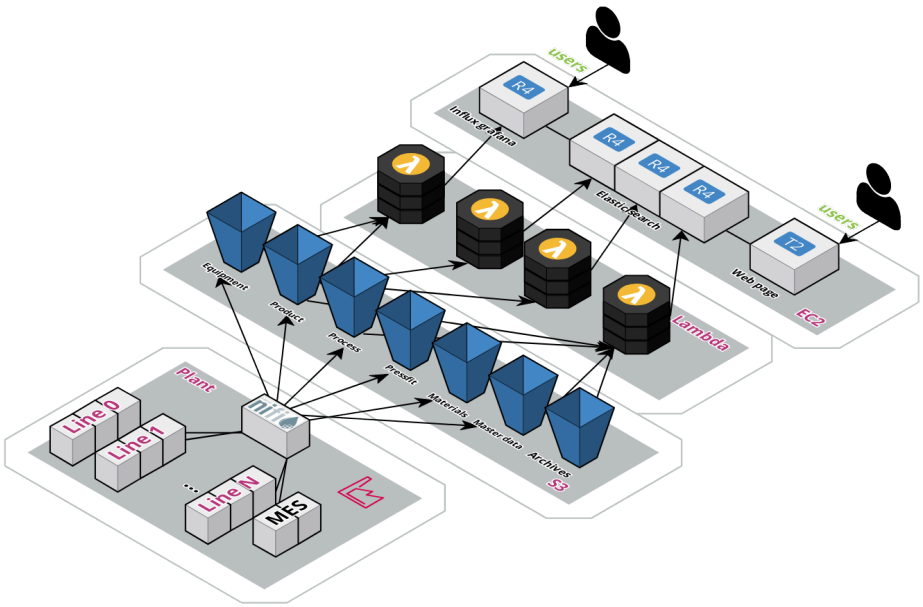


Figure 2.4 – Cloud architecture for the data collection at Foix plant.

At Foix plant, data are collected from the following sources:

— **Equipment**: contains resource usage metrics related to CPU, memory, and disk space in the SPP, SPI, and Oven.

— **Product**: contains results of tests performed by each testing equipment (SPI, AOI, ICT, etc.) and the *mailbox* storing messages generated by the *MES* to track the whole production process.

— **Pressfit**: contains data from the specific operation named *Pressfit*.

— **Process**: contains data about programs, placements (P&P), as well as other MES.
— **Materials**: contains information about the materials.
— **Master data**: contains metadata about products, programs, equipment, and materials.

The data used in this thesis relate to products. In particular, data are extracted from *mailbox*, which contains event logs about the whole production process. All the products at Vitesco Technologies plants are tracked by a unique identifier which is encoded in the form of a data matrix marked on the PCB. This matrix is read each time the PCB goes through an operation. Information from all processes is collected in real-time or near-real-time by the MES and recorded as logs. An example of decoded messages from the cloud storage service of Vitesco Technologies is presented in Figure 2.5. Each message contains product-related and machine-related information. The main features of the messages are given below:

— **Type of message (red)**: there are several types of messages, among which transitive messages and control messages are mainly used. While transitive messages notify that the PCB has entered or exited some operations, control messages give us information on whether or not the process passed as expected. Hence, these messages have a feature of sanction that could be Pass (P) or Fail (F). An example of a control message is the one generated from the AOI machine.
— **Machine hostname/Machine_ID (pink):** the identification of the machine or computer that performed an operation in the PCB.
— **Description of operation (purple):** description of the performed operation.
— **Family code (green):** product family that is being produced.
— **Serial Number/Board_ID (orange):** the identification number of the product.
— **Operation code/Operation_ID (blue):** the identification number of the operation being performed.
— **Sanction (brown)**: for control messages, the sanction given by the operation (F/P), F means the operation has failed; meanwhile, P letter means the operation has been performed successfully.
— **Timestamp (magenta):** the date and time an operation was performed.

In addition to the data extracted from the mailbox, master data are also used to retrieve additional information, such as the number of subpanels, product family, and operation description. Furthermore, master data are used to join tables to link the reference to the product family and the machine ID to the production line. These pieces of in-

```
P2|mid00002|Ope7_Line2|FamilyC128|Board_560| Operation_7|FACE2||006380|U55H4583|Line2-Config|2|1548866902|
PS|mid00003|Ope4_Line3|FamilyC128|Board_544|1548866935|
SF|mid00000|Ope2_Line0|FamilyC128|Board_566| Operation_2|002|P|1548858285|||||||||||||||||||||||||||||||
SS|mid00000|Ope15_Line0|FamilyC128|Board_730|Operation_15|P|1546860302|
```

Figure 2.5 – Snapshot of messages file from MES. Data are anonymized to preserve confidentiality.

formation provide additional knowledge for analyzing and understanding the production process.

In this thesis, we work mainly on data collected from 2019 to 2022. The dataset is massive and holds a significant amount of information. On average, approximately 4 million products are produced each year, with around 100 million related messages generated. Processing this vast amount of data presents significant challenges in terms of computational time and capacity. Various techniques, such as multiprocessing and GPU computing, must be employed to overcome these obstacles.

2.3 Conclusion

In conclusion, this chapter highlights the main objective of the thesis, which is to gain insights into the production process, with a particular focus on the assembly of electronic boards. The production process comprises four main phases: the FE, the ICT, the BE, and the packaging. Among these, the FE and the BE process are a series of operations performed on the PCBs conveyed through the production line. Real-time data are collected throughout the product lifecycle, from the raw PCBs and components to the completed electronic boards. These data, which are stored and managed by Amazon Web Services (AWS), are in the form of event logs. Given the industrial context and the data collected, the research presented in this thesis adopts a process mining approach to study, develop, and apply appropriate techniques for extracting knowledge about the process to provide optimization solutions. The following sections of the manuscript present different process mining approaches and results obtained during the thesis.

Chapter 3

Descriptive Process Mining

Summary of Chapter 3

Descriptive Process Mining (DPM) is a technique that involves using event logs to extract information about processes. This chapter presents an overview of the works and results, which focus on applying techniques in DPM to the PCB assembly process. The chapter is organized into four sections. The first section covers the basics of process mining, including definitions such as event, trace, event log, and prefix. The second section explores process discovery, which is the process of extracting process models from event logs. The section details the proposed discovery algorithms and the calculation of the nominal path. The third section discusses conformance checking, which is the process of identifying deviations between the actual execution of a process and its intended model. It presents a method to compute a penalty index to characterize product quality. The fourth and final section provides an overview of Work-In-Progress (WIP) calculation and tracking. Overall, the chapter provides a comprehensive overview of the works and results in descriptive process mining, highlighting the importance of event logs and the different techniques used in process discovery, conformance checking, and WIP calculation and tracking.

3.1 Preliminary concepts

This section provides necessary concepts and notations of process mining used in the manuscript. Let $\mathcal{E}$ be the event universe, and AN be the set of attribute names. For any

event $e \in \mathcal{E}$ and attribute name $n \in AN$, $\#_n(e)$ is the value of attribute n for event e.

Definition 1 (Event) *An event e is represented by a tuple of attributes* $(\#_c(e), \#_a(e), \#_t(e), \#_{d_1}(e), ..., \#_{d_j}(e), ..., \#_{d_m}(e))$*, where c is the case id, a is the activity during which the event occurs, t is the timestamp and* d_j *is the* j^{th} *additional attribute of the event.*

The additional attributes may be the resource (e.g., the person in charge of the task) or the associated cost, etc. Let $\mathcal{E}^*$ be the set of all finite subsets of $\mathcal{E}$.

Definition 2 (Case) *A case refers to an end-to-end process instance that includes multiple activities or tasks. Each case is assigned a unique identifier that serves to capture all events related to that instance.*

In the presented assembly process, a case corresponds to a specific product (PCB), and the case id is the serial number of that PCB.

Definition 3 (Trace) *A trace is a finite, non-empty sequence of events* $\sigma \in \mathcal{E}^* \setminus \{\emptyset\}$ *corresponding to some process instance,* $\sigma = \langle e_1, ..., e_{|\sigma|} \rangle$ *such that for* $1 \leq i < j \leq |\sigma|$: $\#_c(e_i) = \#_c(e_j)$.

Definition 4 (Prefix) *A prefix is the head of a trace* σ *with a length* $0 < k < |\sigma|$*. It is denoted as* $hd^k(\sigma) = \langle e_1, ..., e_k \rangle$.

As an illustration, consider a trace $\sigma = \langle a, b, c, d, e \rangle$, $hd^2(\sigma) = \langle a, b \rangle$ and $hd^3(\sigma) = \langle a, b, c \rangle$. The prefix notation is utilized in Section 4.2 for predicting the remaining cycle time.

Definition 5 (Event log) *An event log* $\mathcal{L}$ *is a set of complete traces, i.e., traces representing the execution from the beginning to the end of a case.*

An event log contains data related to a process, i.e., the assembly process of electronic boards. Each process gives rise to as many process instances as products. A process instance is characterized by a set of generated events that are gathered in a trace. The structure of event logs is presented in Figure 3.1.

Definition 6 (Graph) *A graph* $\mathcal{G}$ *is a tuple* (V, E)*, where* V *is the set of* $N = |V|$ *vertices and* E *represents the set of edges.*

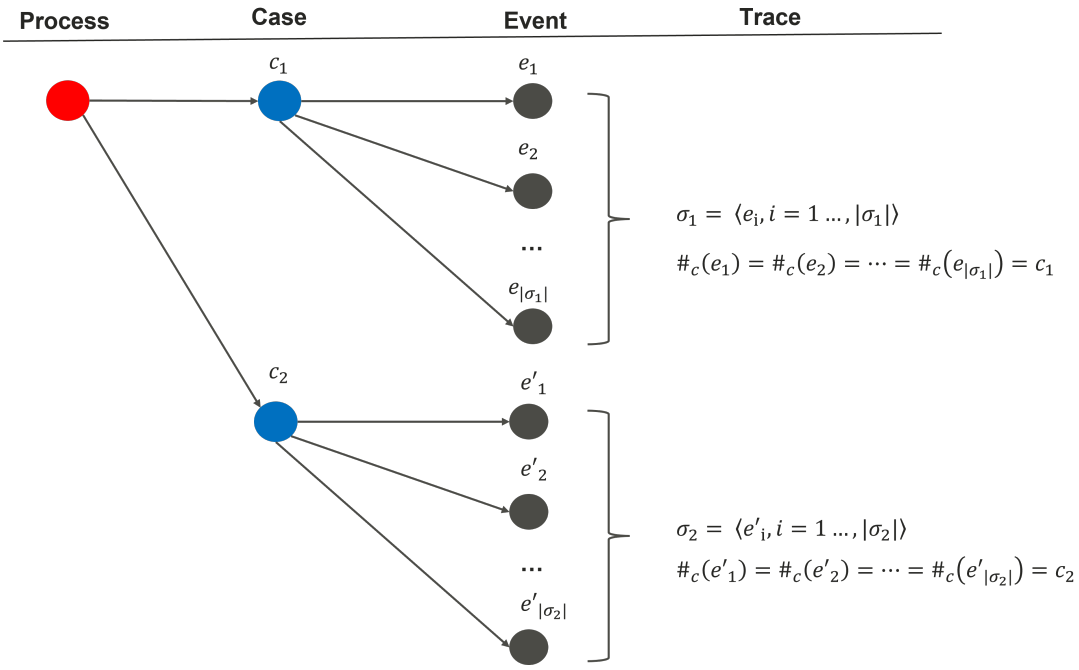


Figure 3.1 – Tree structure of an event log.

An edge $e_{ij} \in E$ indicates the transition between vertex v_i and vertex v_j. Nodes and edges can be assigned with features called node features and edge features.

Definition 7 (Adjacency matrix) *Given a graph $\mathcal{G} = (V, E)$, the adjacency matrix of $\mathcal{G}$ is denoted as $A \in \{0, 1\}^{N \times N}$. Each element $A_{i,j}$ of the matrix represents the connectivity between two corresponding nodes $v_i, v_j \in V$. This means that $A_{i,j} = 1$ if there exists an edge e_{ij} between nodes v_i and v_j, and $A_{i,j} = 0$ otherwise.*

3.2 Process discovery

3.2.1 Preprocessing and analysis of event logs

The first step in analyzing event logs is to extract and preprocess them. In the context of this thesis, the collected messages are stored in S3 buckets and accessed using the `boto3` package in Python. We employ parallel computing functions from the `multiprocessing` package to speed up processing. In our case study, we used an Amazon EC2 Instance with 32 CPUs that ran for over 35 hours to process one year of data, which would have been impractical with a classical machine.

Preprocessing is a crucial step that determines the quality of the analysis process. However, it can be tedious and challenging. The two major challenges we encountered were missing data and abnormal data. Due to the assembly line configuration, only one camera is installed at each operation, making it impossible to have both entry and exit information for all operations. Some operations have a camera at the entry point, while others only record the product state at the output of the operation. This missing data makes it difficult to analyze cycle time and transition time. Abnormal data includes abnormal serial numbers, duplicated messages, unrecognized operation IDs, and operations with the same timestamp. We worked with process experts to investigate these issues and developed functions to automatically identify and filter them out during processing. We found out that abnormal serial numbers refer to products used to test the operation function before running a batch. This is usually performed at the beginning of each shift. In addition, duplicated messages and the problem of operations with the same timestamp stem from an IT problem.

After the preprocessing, event logs are stored in a tabular format in which columns are different attributes and rows represent events. A fragment of the event log used in this study is shown in Table 3.1, where an event is associated with a product (a board), an operation, a timestamp, and the machine being operated.

Table 3.1 – A fragment of an event log generated from messages: an event per line.

Board_ID	*Operation_ID*	*Timestamp*	*Machine_ID*
Board_0	Operation_19	2019-01-21 11:20:49	Machine_0
Board_0	Operation_12	2019-01-21 11:20:50	Machine_26
Board_0	Operation_13	2019-02-07 23:52:37	Machine_19
Board_0	Operation_14	2019-02-08 00:24:21	Machine_11
Board_0	Operation_15	2019-02-08 00:31:33	Machine_12
Board_0	Operation_16	2019-02-08 00:31:35	Machine_12
Board_0	Operation_17	2019-02-08 00:39:39	Machine_20
Board_0	Operation_18	2019-02-08 00:42:49	Machine_14
Board_0	Operation_19	2019-02-08 00:43:00	Machine_14
Board_0	Operation_999	2019-02-08 00:43:01	Machine_0
Board_1	Operation_19	2019-01-21 11:20:33	Machine_0
Board_1	Operation_12	2019-01-21 11:20:34	Machine_26
Board_1	Operation_37	2019-01-30 23:55:46	Machine_10
Board_1	Operation_37	2019-01-30 23:57:54	Machine_10
Board_1	Operation_21	2019-01-30 23:57:55	Machine_0
...	...	...	...

Once the data is extracted and preprocessed, a general analysis is conducted. For example, Figure 3.2 displays the number of product families manufactured in 2019, revealing over 70 families. It is shown that the number of products varies by family, depending on the orders from clients.

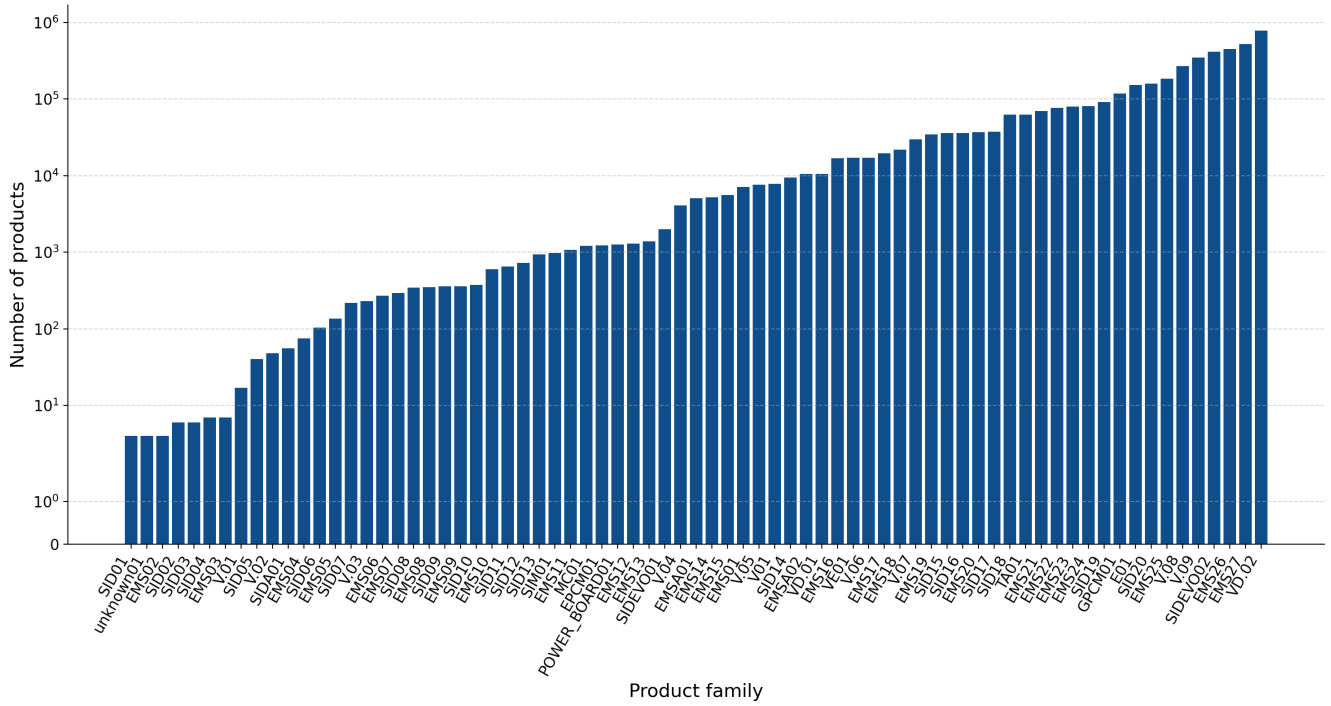


Figure 3.2 – Number of products by family in 2019.

Figure 3.3 demonstrates the number of products packaged each month during 2019. This graph shows a lower product delivery rate in the initial months of the year and during the summer vacation period in August.

In the analysis of cycle times, we encountered the challenge of missing data, making it impossible to measure the cycle time of products for each operation. However, we were able to calculate the Life Cycle Time (LCT) of products by determining the time between the first and the last event. As shown in Figure 3.4, we analyzed the LCT of products in the family **VD01** in 2019. The graph displays the number of products and their life cycle times, ranging from one day to a year. Process experts have defined these time intervals according to the product specifications. The graph indicates that most products (80%) are completed within two days, which is accurate since production is performed on demand. This means that products are produced only when needed and only in the requested quantities. Products that take two days to a month to finish are generally due

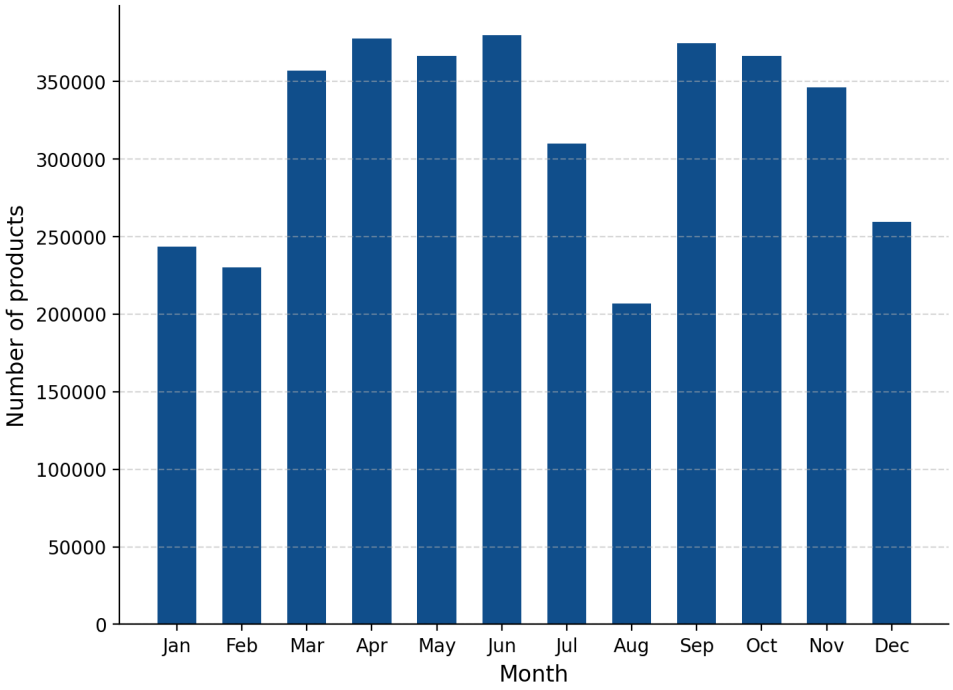


Figure 3.3 – Number of products by month in 2019.

to production failures or long waiting times between phases. However, products that have existed for over a month should be examined further because they can compromise the reliability of sensitive electronic components. This is a point that could be considered to improve the plant's production.

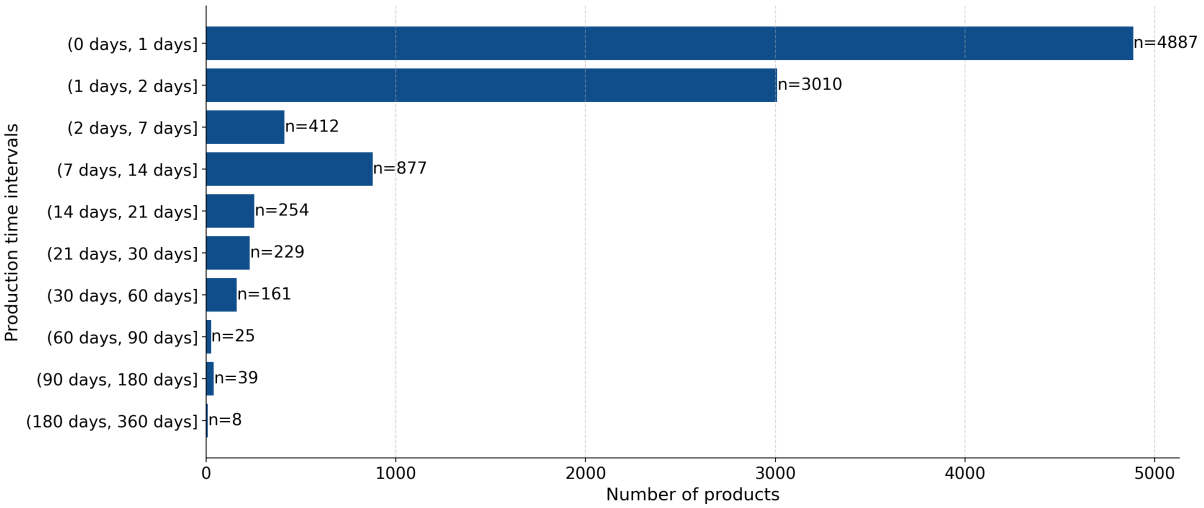
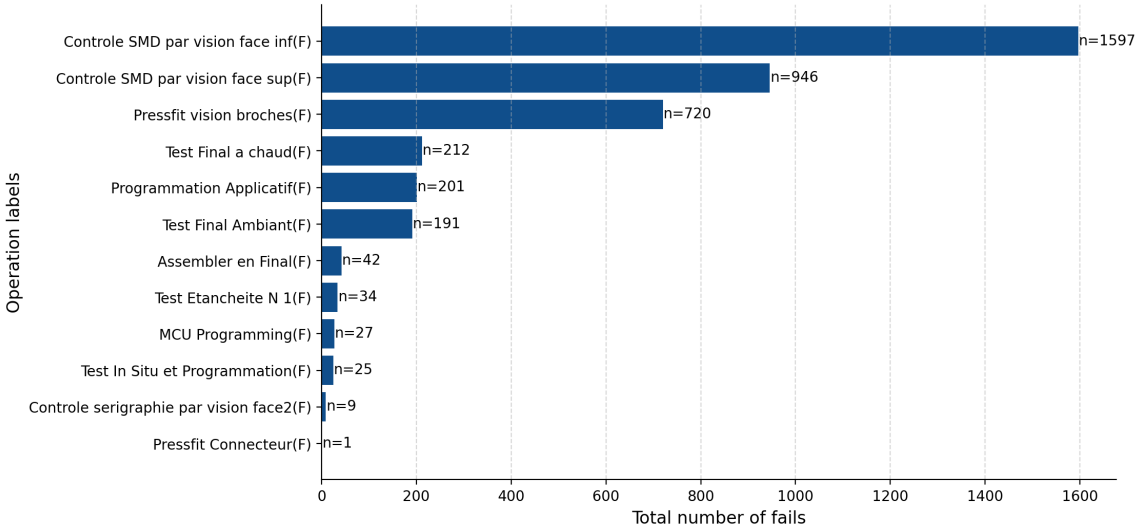
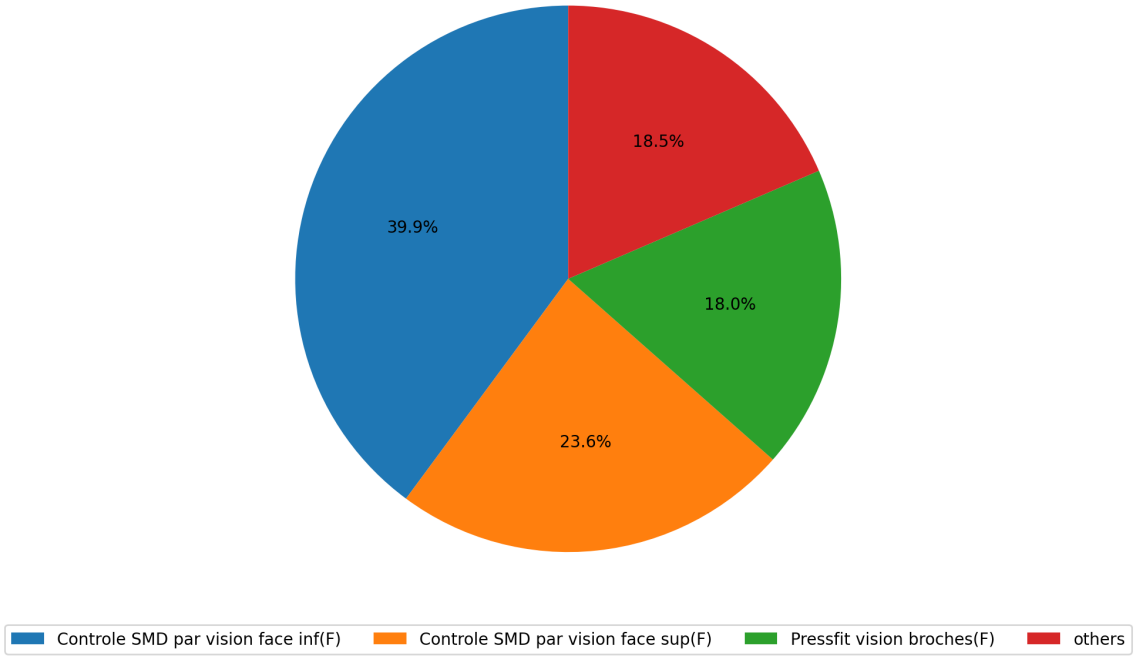


Figure 3.4 – Life cycle time of products VD01 in 2019.

Figure 3.5 presents the number of failures by operation. The barplot on the top (3.5(a)) presents the count of fail messages generated by each operation, while the pie chart on the bottom (3.5(b)) shows the percentage distribution. The figure reveals that the majority of the fail messages come from operations *Controle SMD par vision face inf* and *Controle SMD par vision face sup*, which are performed by the same machine. The only difference is that *Controle SMD par vision face inf* operates on the first side, and *Controle SMD par vision face sup* operates on the second side of the PCB. There are more failures on the first side, as it usually contains more components than the second. The third operation that produces the most failures is *Pressfit vision broches*, which is in the *BE* phase. However, further investigation revealed that the machine generated multiple fail messages due to a computer problem.



(a)

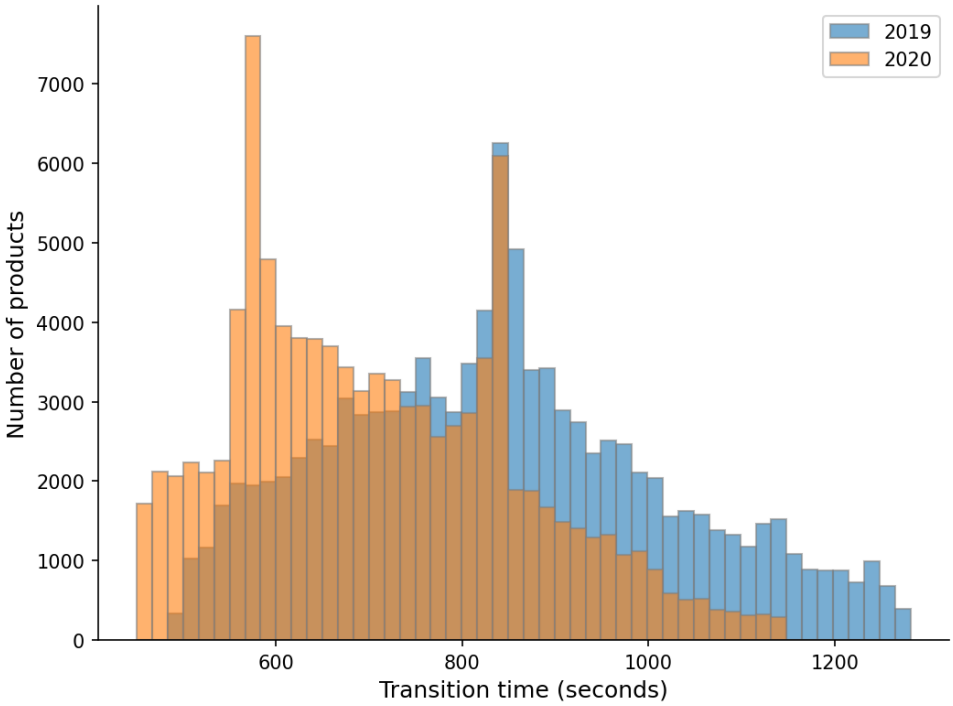


(b)

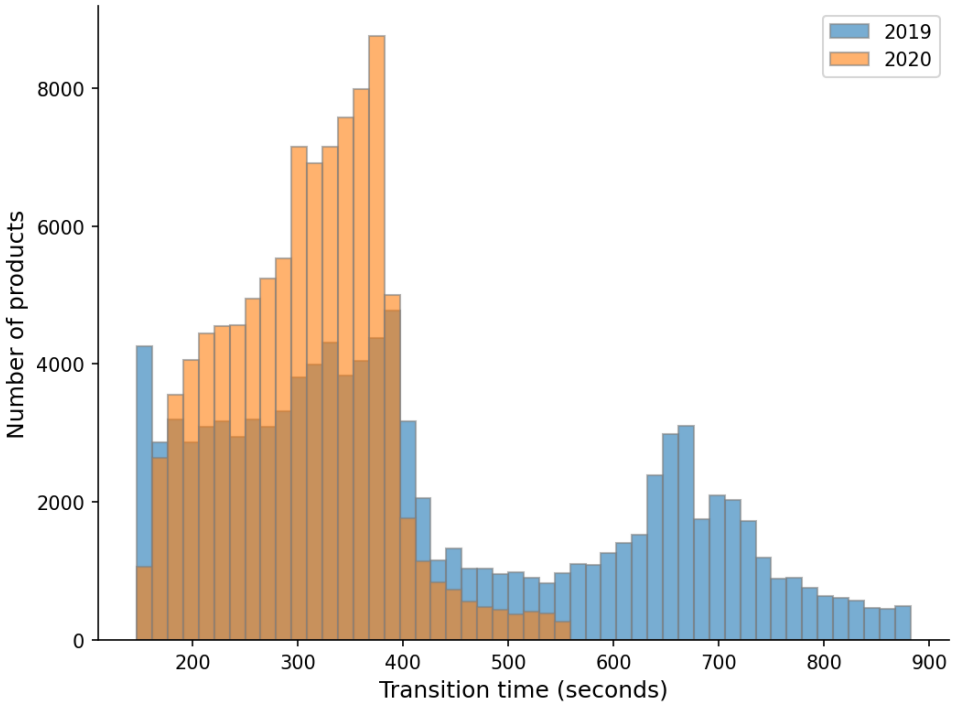
Figure 3.5 – Number of fails by operation in 2019.

We also analyzed the transition time between operations, i.e., the time difference between messages generated by two consecutive operations. Our analysis was performed by family and by year. Figure 3.6 displays the distribution of transition times between

Operation_03 and *Operation_04* performed on the first (3.6(a)) and second (3.6(b)) side of the PCB. We compared the transition time of the same product family between 2019 and 2020. The figure indicates an improvement in 2020, with a shorter transition time. The plant informed us that they had made a change in operation *Operation_03*. The results obtained confirm this information.



(a)



(b)

Figure 3.6 – Distribution of transition time between *Operation_03* and *Operation_04* performed on the first (a) and second (b) side of the PCB.

We have also conducted an analysis of the products of the family EMS01 returned by clients, specifically examining the transition times. Figure 3.7 displays the distribution of the transition time between *Operation_11* and *Operation_12* in the back end, comparing the faulty product's transition time in violet line with that of all other products in the same family and year. Additionally, the figure compares the distribution of the transition time of the batch containing the faulty product with the overall population. The results show that the transition time of the returned product is an outlier in the distribution, and the batch distribution is shifted to the right, indicating that the products in that batch require more time than others to pass through the two operations *Operation_11* and *Operation_12*. This finding provides operators with a clue to identify the root cause of product failure and prevent the issue from recurring in the future.

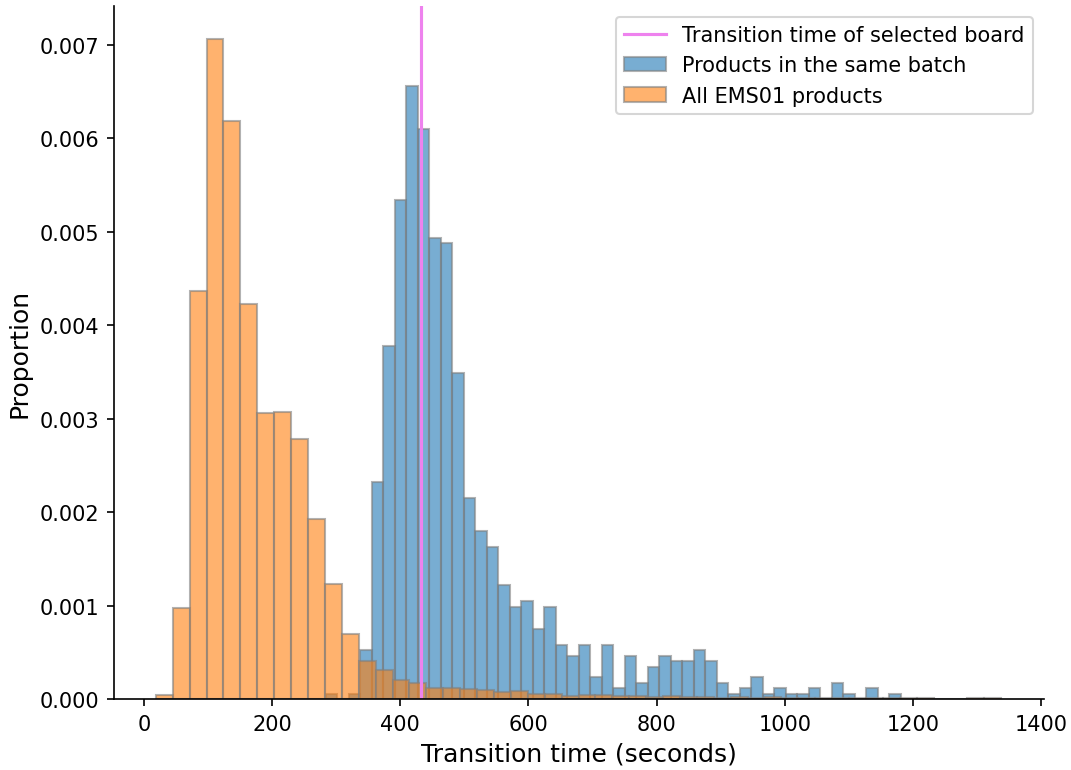


Figure 3.7 – Comparison of the transition time between *Operation_11* and *Operation_12* of returned product (violet line) with all other products (in orange) of the same family (EMS01) in the same year and with products in the same batch (in light blue).

3.2.2 Process discovery and further analysis

There are numerous process discovery algorithms presented in the literature with their own advantages and drawbacks (refer to Section 1.1). In this study, we develop a method to construct the Process Model (PM) from the event logs. Although the plant may have its own designed PM before launching a new product, deviations from this model often occur over time, creating a need for an automatic discovery process. Due to the dynamic nature of industrial processes, the aim of this work is to develop a data-driven solution to discover the PM automatically.

We proposed first to apply existing algorithms such as the α algorithm [1], Heuristic Miner [118], and Inductive Miner [54], but the process expert was not satisfied with the results. Indeed, the constructed process model abstracted the deviations into parallelism and concurrency, which did not reflect the actual production process. The process expert wants a visual depiction of all transitions during production. Indeed, a PCB passes normally through a sequence of operations. This sequence is called the nominal path. Any deviation from the nominal path is considered an anomaly. Therefore, we proposed the use of a Directly Follows Graph (DFG) (Definition 8) to capture all transitions of products recorded in the event log, meeting the process expert's requirements.

Definition 8 *(Directly Follows Graph) A directly follows graph, denoted as DFG is a pair $(\mathcal{G}, L)$ such that:*

- *$\mathcal{G}$ is a directed graph $\mathcal{G} = (V, E)$ with vertex set V and edge set E.*
- *L is a set of labels, where l_{ij} is associated with the edge between vertex v_i and vertex v_j.*

Definition 9 *(Process model) A process model, denoted as PM, is a DFG, where V is instantiated with the set of activities related to process operations, and the edges of E represent the precedence relation of operations according to the traces of the process. V includes two specific nodes, a source node v_{start} and a end node v_{end}. The edge labels in L can represent different information, such as transition counts, durations, etc.*

Figure 3.8 presents the process model of the product family $VD01$ constructed from the event logs in 2019. The visualization is produced by the process mining library developed in python PM4Py [9]. In this graph, the activities associated with nodes are numeric identifiers of operations and are anonymized for confidentiality concerns. The v_{start} and v_{end} nodes are in circle and square, respectively. The edges refer to product state transitions between operations. Additionally, the label l_{ij} on top of each edge represents the

number of products that have taken the corresponding transition. The number of products that have gone through operations is reflected by the more or less dark purple color of the nodes. This number is reported between brackets in each node.

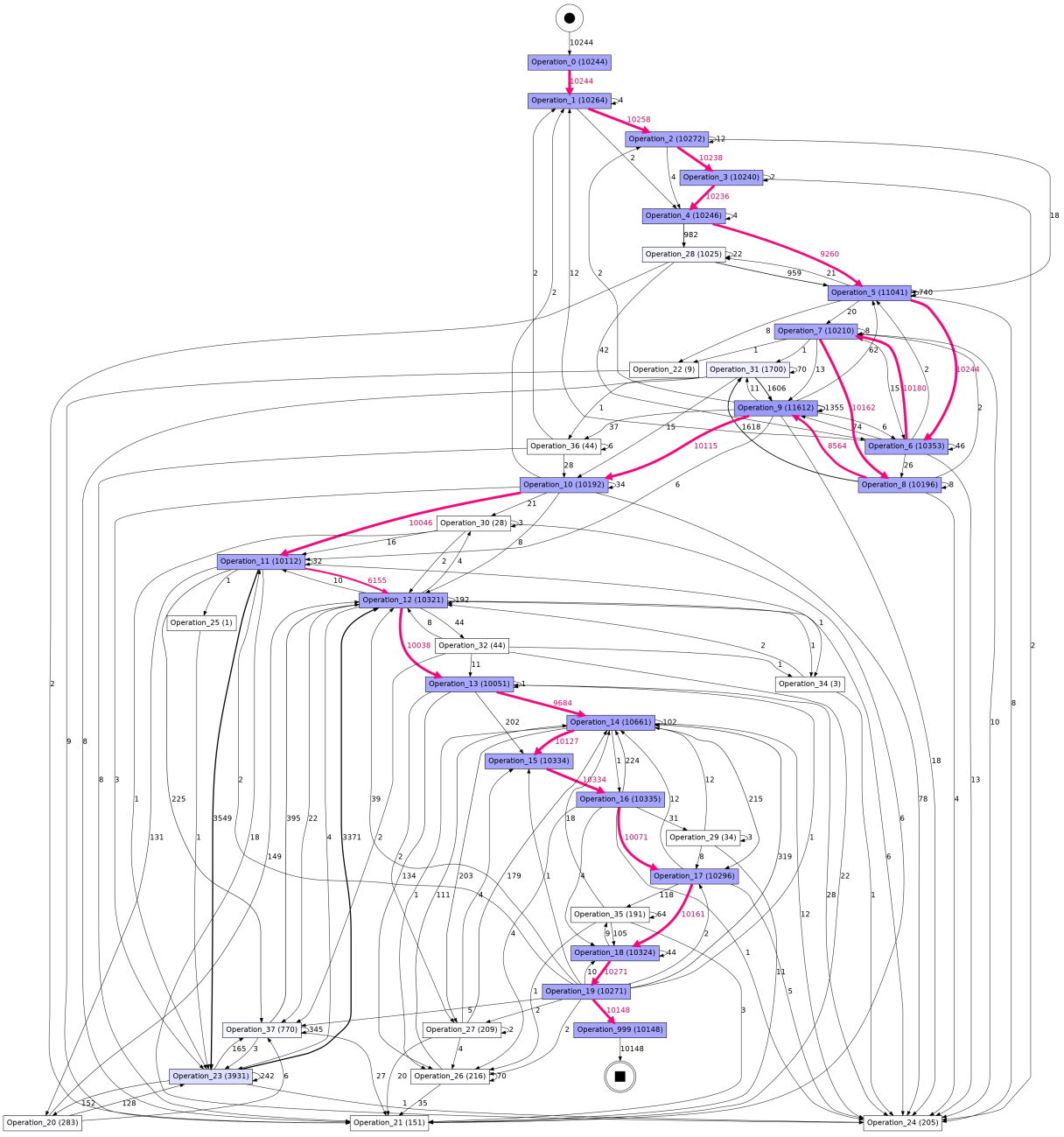


Figure 3.8 – Process model represented by a *DFG*.

For example, there are 10258 product transitions between the node marked with the activity Operation_1, i.e, the *laser marking* operation, and the node marked Operation_2

that corresponds to the *solder paste inspection side 1*. There are 10264 products that have gone through Operation_1 and 10272 that have gone through Operation_2 (see Figure 3.9).

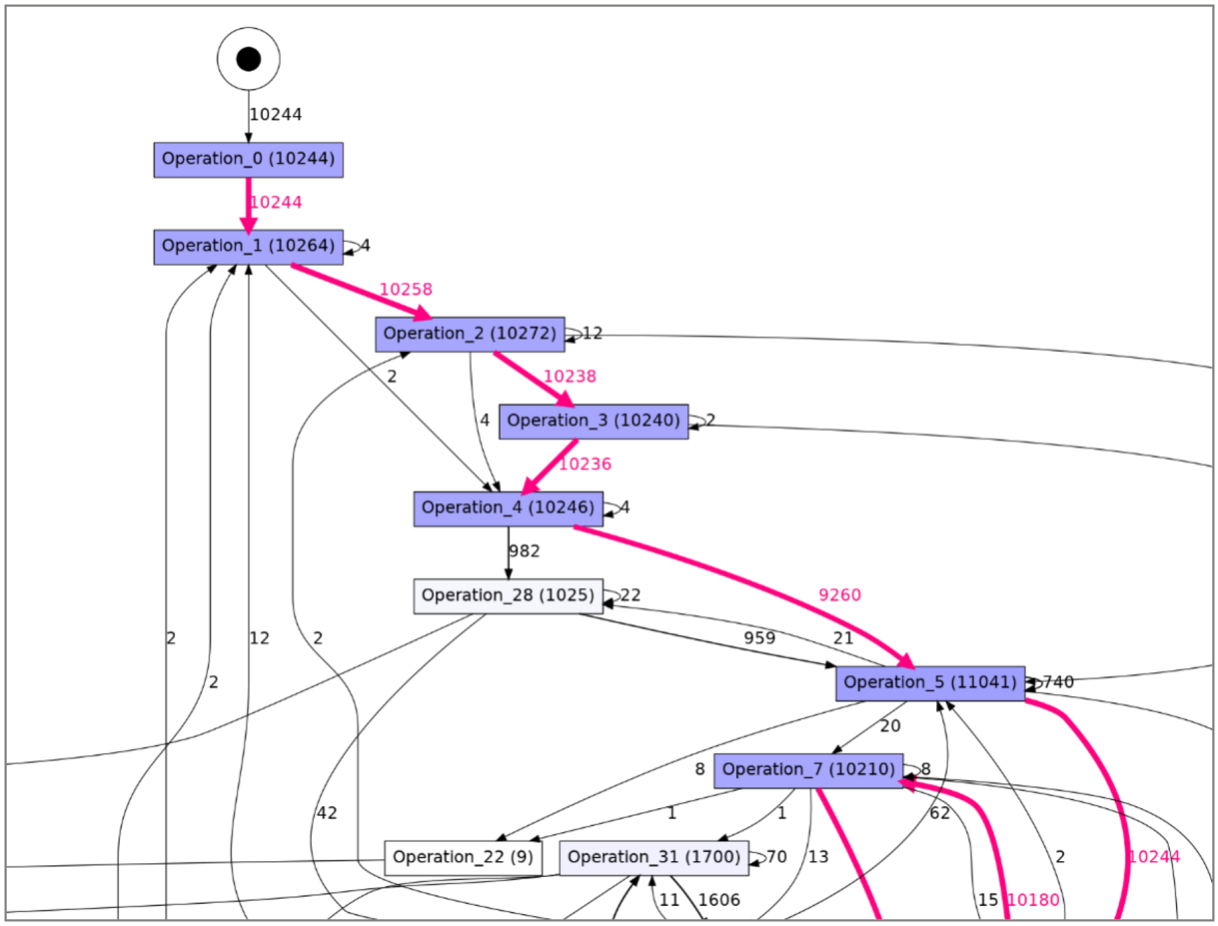


Figure 3.9 – Zoom of the *DFG* representing the process model on the front end phase.

Based on the frequency of each transition, only the transitions that most products follow are kept so that only the nominal path is captured. For this purpose, an algorithm has been developed to compute the *nominal path*, denoted as $\mathcal{P}^*$. Products of the same family follow the same path if nothing wrong happens along the production chain. The nominal path $\mathcal{P}^*$ is computed by the following formula:

$$\mathcal{P}^* = \underset{\mathcal{P} \in PM}{\operatorname{argmax}} \, freq(\mathcal{P}) \tag{3.1}$$

where $freq(\mathcal{P})$ returns the number of traces, i.e., process instances, that follow the path $\mathcal{P}$.

The proposed process for computing the nominal path is described in Algorithm 1. Firstly, the algorithm identifies the first and last operations, i.e., v_{start} and v_{end} (row 1). Subsequently, the DFG is computed from the event log $\mathcal{L}$ (row 2). In this case, the label l_{ij} of each edge e_{ij} represents the transition count, i.e., the number of times a product has passed from v_i to v_j. The algorithm then proceeds iteratively, searching for the next operation from the first operation where most of the products are directed until the last operation is reached (rows 3-9). This sequence of operations constitutes the nominal path. The nominal path can be obtained by following purple-colored operation identifiers in Figure 3.8.

Data: Event log $\mathcal{L}$
Result: Nominal path $\mathcal{P}^*$
1 Compute v_{start} and v_{end} from $\mathcal{L}$;
2 Compute DFG from $\mathcal{L}$;
3 $\mathcal{P}^* \leftarrow [v_{start}]$;
4 $v_{current} \leftarrow v_{start}$;
5 **while** $v_{current} \neq v_{end}$ **do**
6 $\quad E_i \leftarrow \{e_{ij} | v_i = v_{current}\ \&\ v_j \notin \mathcal{P}^*\}$;
7 $\quad v_{current} \leftarrow v_k$ where $e_{ik} \in E_i\ \&\ l_{ik} = \max_i\{l_{ij}\}$;
8 $\quad \mathcal{P}^*.append(v_{current})$
9 **end**

Algorithm 1: Nominal path discovery.

The algorithm is then applied to each product family to obtain the corresponding process model and nominal path. The goal of this step is twofold: to acquire global information about the actual production and to obtain a reference for the normal behavior as represented by the nominal path. The nominal path computed by the Algorithm 1 is shown as the pink path in Figure 3.8. The process model indicates that there are several deviations from the nominal path, which are considered anomalies because they represent variations from what is expected or designed. Such deviations can lead to faulty or unexpected quality outcomes. The graph highlights two obvious deviations: loops and reverse transitions. These issues arise when a product returns to an already executed operation, which is not typical in an assembly process. While a product may be retested in a check operation, this could mean that it malfunctioned before or there was another problem that required retesting. Therefore, it is considered an anomaly.

One solution to analyze each product's flow path (or a group of products) is to isolate and compare them with the nominal path. However, this solution requires significant

effort and resources, especially with many products and product families. To address this, a unified approach has been developed to check the conformity of the product path with the reference and calculate a quality index. We will present this approach in Section 3.3.1.

3.3 Conformance checking

3.3.1 Assessing product quality from event logs

After analyzing event logs and process models, different types of anomalies are identified in the production process. Several products deviate from the standard path, and some even fail certain check operations. Additionally, there is a significant variation in transition times between operations for different products and batches (see Section 3.2.1), which can impact the final product quality. Therefore, we propose a novel method to calculate a product quality index considering both the product path and production batches. This index is generated by categorizing the products based on their consistency to the nominal path and using advanced data analysis techniques enhanced by expert know-how. The resulting quality index can identify the risk of customer return, which is highly relevant information for after-sales service.

To achieve this, our method compares each product's production path to a reference standard known as the nominal path. Various criteria are established, such as the order of operations or the absence of certain nominal operations (operations that constitute the nominal path). Moreover, we use a time-based criterion to identify products that take a long time between processes, which may affect the final quality. To accomplish this, we compute time constraints that specify the standard transition time between nominal operations. This results in a timed process model that combines the nominal path and time constraint, where interval time labels represent the time ranges of all products traveling between two operations for a set of process instances. By integrating time aspects into the process model, we aim to determine the standard time constraints between process operations.

Definition 10 defines the timed process model as the process model labeled with time intervals representing the time constraints for process instances. This approach enables us to identify more precise deviations from the standard path and time constraints, which can help improve the quality of the final products.

Consider an event log $\mathcal{L}$ that gathers a set of traces $\mathcal{T}$ representing different instances

of the same process. Consider an edge between two adjacent vertices v_i and $v_j \in V$ corresponding to two events e_i and e_j. The subset $\mathcal{T}_k \subseteq \mathcal{T}$ gathers the traces that take a path through the consecutive operations represented by v_i and v_j. Let us also assume that in any trace $\sigma \in \mathcal{T}_k$, the timestamps of e_i and e_j are such that $\#_t(e_j) > \#_t(e_i)$, then the label l_{ij} is determined as follows:

$$l_{ij} = [\Delta t_{ij}^-, \Delta t_{ij}^+] \tag{3.2}$$

where:

$$\Delta t_{ij}^- = min_{e_i,e_j \in \mathcal{E}_\sigma, \sigma \in \mathcal{T}_k}(\#_t(e_j) - \#_t(e_i)),$$
$$\Delta t_{ij}^+ = max_{e_i,e_j \in \mathcal{E}_\sigma, \sigma \in \mathcal{T}_k}(\#_t(e_j) - \#_t(e_i)).$$

The time interval bounds defined by Equation (3.2) represent the *inf* and *sup* of the elapsed time between two consecutive product states in the process.

Definition 10 ***(Timed process model)***

A timed process model (t-PM) is a process model PM for which edges are labeled according to the time constraints given by Equation (3.2).

A process instance or a trace $\sigma = \langle e_1, e_2, ..., e_n \rangle$ satisfies the *timed process model* if:

1. The sequence of activities $\langle \#_a(e_1), \#_a(e_2), ..., \#_a(e_n) \rangle$ can be replayed in the graph $\mathcal{G}$ of *t-PM*.
2. All event pairs e_i and e_j satisfy the time constraint $[\Delta t_{ij}^-, \Delta t_{ij}^+]$, i.e., $\#_t(e_j) - \#_t(e_i) \in [\Delta t_{ij}^-, \Delta t_{ij}^+]$.

Once the *timed process model* is obtained, a categorization approach is proposed to sort products into populations. A population is the set of traces related to the product. Each population is then associated with a penalty index depending on the conformance of the paths with the obtained model.

Populations are defined by constructing a decision tree which is a technique for clustering [68]. The primary advantage of using a decision tree is that it is easy to follow and understand. Decision trees have four main parts (see Figure 3.10): a root node, internal nodes, leaf nodes, and branches. The root node is the starting point of the tree, and both

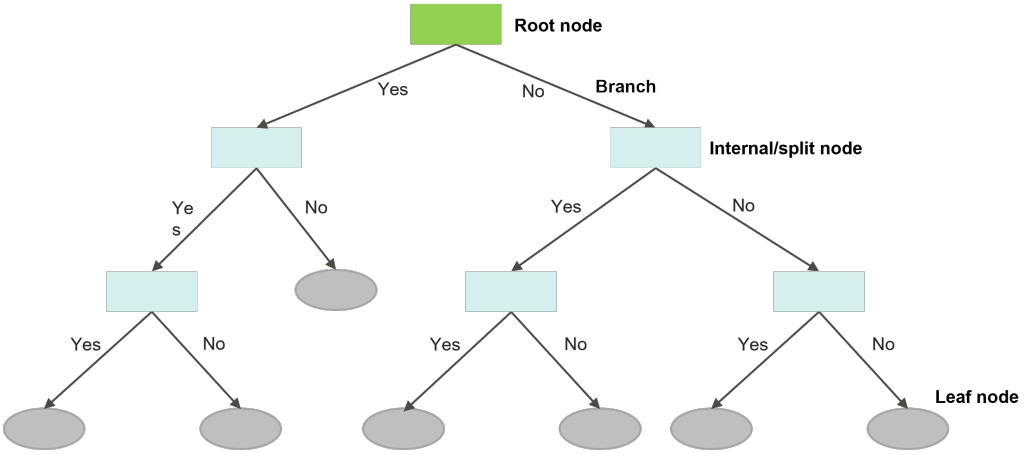


Figure 3.10 – Structure of the decision tree.

root and internal nodes contain a test based on an attribute. Each branch represents the answer to the test, and each leaf node represents a class label.

Let $\mathbf{Y} = \{y_\alpha, \alpha = 1, ..., m\}$ be a set of m class labels. The partition issued from the decision tree aims to gather the products that share the same path. Classes $(y_\alpha)_{\alpha=1...m}$ are then associated with individual penalty indexes $(\tilde{p}_\alpha)_{\alpha=1...m}$ provided by process experts to score process instances.

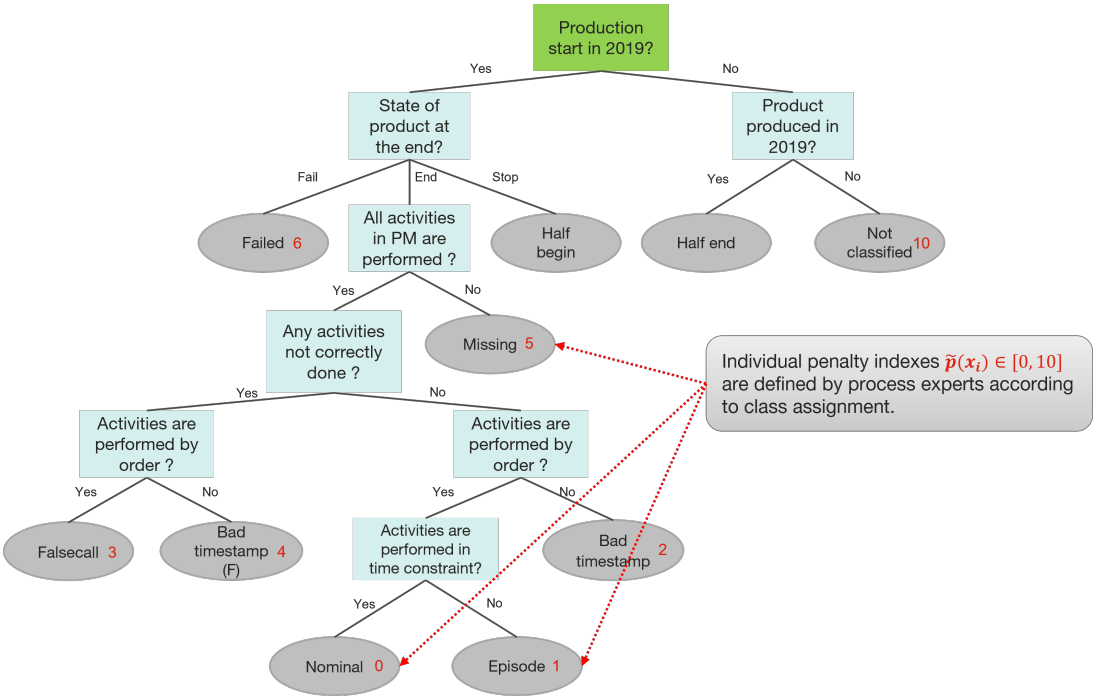


Figure 3.11 – Decision tree for product categorization.

Product categorizing consists of checking the conformance of the corresponding trace with the timed process model. Several criteria are used, particularly order of operations, presence or absence of nominal operations, and consistency with the time constraints. These criteria or categorizing rules are illustrated by the decision tree given in Figure 3.11. The building process of this decision tree involves several steps. First of all, products that have a path from the first operation to the end are selected. This means that those whose production path is incomplete are excluded. Then, among products that performed the first and last operation, the algorithm checks whether products missed at least one *nominal operation*. The next step is to check for the order of nominal operations and the time constraints for transitions between these operations. The light blue boxes in Figure 3.11 represent internal nodes that contain split rules. The gray boxes are leaf nodes with corresponding class labels. Each class is associated with a penalty index, a real number in $[0, 10]$, depicted in red. Moreover, a description of all class labels is presented in Table 3.2.

Table 3.2 – Description of class labels in the decision tree of Figure 3.11.

Class label	**Description**
Half begin	Products that lack information about the final state.
Half end	Products that lack start date information.
Not classified	Products that do not have a start state and end state information.
Failed	Products that have failed an operation and are then discarded.
Missing	Products whose production has been completed but some nominal operations have been missed.
Falsecall	Products have suffered some failures during production and have passed all nominal operations in order.
Bad timestamp (F)	Products that have suffered some failures during production and have not fulfilled the order of nominal operations.
Bad timestamp	Products that have completed production without failure but have not fulfilled the order of nominal operations.
Episode	Products that have completed production without failure and have passed all nominal operations in order, but the time constraints have not been respected.
Nominal	Products that have completed production without failure and have passed all nominal operations in order and within the time constraints.

Penalty index for products

As mentioned previously, each product is associated with a so-called *individual penalty index*, depending on the class it belongs to. This index aims at characterizing product quality. The quality of a product is evaluated based on its production path and production *batches*. Remember that a batch is composed of a set of products produced under the same configuration within a given time frame.

Generally, products go through several phases, from separate components to assembly and packaging (see Figure 2.1). Due to the heterogeneity of configurations in the different phases, the partition of products into batches in each phase is different. This means it is possible to have two products in the same batch in one phase and in different batches in the other. A given product hence belongs to several batches, one batch per phase. For a given phase, it is assigned a unique class indicated by its path along the operations of the phase: $y_\alpha, \alpha = 1, \ldots, m$.

Let $\mathbf{X} = \{x_i, i = 1, ..., n\}$ be the set of all products. Let $\Phi = \{\phi_k, k = 1, \ldots, |\Phi|\}$ be the set of phases and $\mathbf{Y}_k = \{y_{\alpha|k}, \alpha = 1, ..., m_k\}$ the set of class labels for phase ϕ_k. Assume a set of batches $\mathbf{B} = \{b_j, j = 1, ..., |B|\}$, partitioned in $\mathbf{B} = \bigcup_{k=1}^{|\Phi|} \mathbf{B}_k$ according to the phases, where $\mathbf{B}_k = \{b_{j|k}, j = 1, ..., |B_k|\}$. The batch of product x_i in phase ϕ_k is denoted by $b_{j_i|k}$.

— **Individual penalty index of product x_i in a given phase ϕ_k**
Let $\tilde{p}_k(x_i)$ denote the *individual penalty index* of product x_i in phase ϕ_k. This *penalty index* inherits the penalty index of the class it belongs to for the given phase. For example, a product x_i that belongs to class $y_{\alpha|k}$ for phase ϕ_k has individual penalty index $\tilde{p}_k(x_i) = \tilde{p}_{\alpha|k}$. As an example, the individual penalty index of products of the *nominal* class (see Figure 3.11) is equal to 0. Products in this class follow the nominal path without fail operations and respect the time constraints between operations.

— **Penalty index of a batch b_j of phase ϕ_k**
The *penalty index* of a batch is computed based on the percentage of products in different classes for that batch. Given a batch b_j containing $|b_j|$ products, the ratio of products in the class $y_{\alpha|k}$ with the corresponding penalty index $\tilde{p}_{\alpha|k}$ is denoted by $r^j_{\alpha|k}$. The penalty index of a batch b_j of phase ϕ_k is then given by:

$$p_k(b_j) = \sum_{\alpha=1}^{m_k} r^j_{\alpha|k} \times \tilde{p}_{\alpha|k} \tag{3.3}$$

As already explained, the quality of a product is influenced by the classes, i.e., the paths it follows along the production line for the different phases and the batches to which it belongs. The different phases and the corresponding batches may impact product quality differently.

— **Global penalty index of product** x_i The final penalty index of a product x_i is hence obtained by combining its individual penalty indexes for each phase and the penalty indexes of the batches it belongs to, weighted by their phases.

$$p_g(x_i) = \sum_{k=1}^{|\Phi|} \gamma_k \times \left(\lambda \times \tilde{p}_k(x_i) + (1-\lambda) \times p(b_{j_i|k})\right), \quad \lambda, \gamma_k \in [0,1], \quad \sum_{k=1}^{|\Phi|} \gamma_k = 1 \tag{3.4}$$

Note that the value of individual penalty indexes $\tilde{p}_k(x_i)$ is in $[0, 10]$, the penalty index of batches and of final products given by Equations (3.3),(3.4) are also in $[0, 10]$. The parameter λ determines the contribution of the individual penalty index and the penalty index of the batch to the overall penalty index in each phase, while the parameter γ_k defines the weight given to the phase Φ_k in the sum. These two parameters are determined by process experts in the first instance and can be tuned to find the optimal values.

Quality evaluation: A case study

The calculation of the penalty indexes has been performed on the case study of the PCB assembly process. In the first step, we built the timed process model. The building procedure has been modified as instead of using the min and max value to define the two bounds of the time constraints (see Equation (3.2)), statistical values, specifically the 5^{th} and 95^{th} percentile. In this way, outliers and extreme values have been excluded. Figure 3.12 shows the timed process model obtained for the process model of Figure 3.8. The model is relatively simple, with operations performed successively, one after the other, and no deviations or concurrent operations.

The next step involves categorizing products by comparing their production path with the timed process model. For this task, the decision tree presented in Figure 3.11 was used. The penalty index assigned to each class (indicated as a red number) was provided by the process expert, and their values range from 0 to 10, characterizing the conformance level.

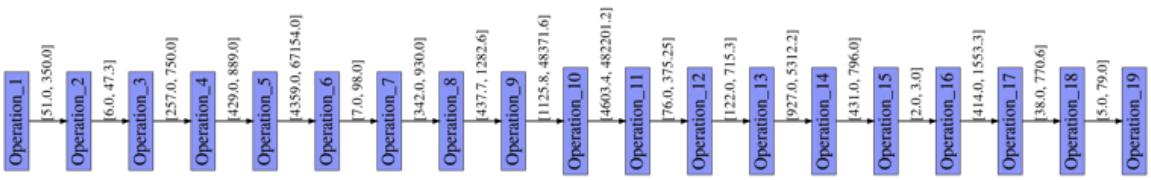


Figure 3.12 – The timed process model.

Note that the *half begin* and *half end* classes were excluded because the production paths of products in these classes are incomplete. Specifically, only data registered in 2019 were considered, and products that were put in the production line before 2019 and reappeared in 2019 were classified in the *half end* class. Similarly, boards that were not finished in production in 2019 were classified in the *half begin* class.

After categorizing products, the penalty index for batches and the global penalty index of products were computed using Equations (3.3) and (3.4). The computed results are presented below.

— **Penalty index obtained by batch:** Let us recall that a batch is a subset of products that are produced continuously and consecutively. Batches are extracted from event logs, and their penalty index is computed using Equation (3.3). Figure 3.13 shows the relation between the penalty index and batch size in the three phases of the production process, namely, $FE1$, $FE2$, and BE. Table 3.3 provides additional statistics, such as the standard deviation of batch size, the percentage of batches with penalty index < 1, and the percentage of perfect batches with a penalty index equal to 0. Over the three phases, more than half of the batches have a penalty index between 0 and 1 (see the third column of Table 3.3). Particularly in the $FE1$ phase, 84.21% of batches adhere to the timed process model. However, batches produced in the $FE2$ and BE phases are less consistent than those in the $FE1$ phase. The results also indicate that there were no perfect batches, which means that no batch had all its products pass through the production line without any failure or deviation within the accepted time intervals. Additionally, the number of products per batch varies with a standard deviation of around 200, and batches with a high penalty index have small sizes. Without an in-depth investigation, it is explainable that these batches are small due to process interruption or changes in the configuration after a sequence of products with bad behavior, which results in a high penalty index for them.

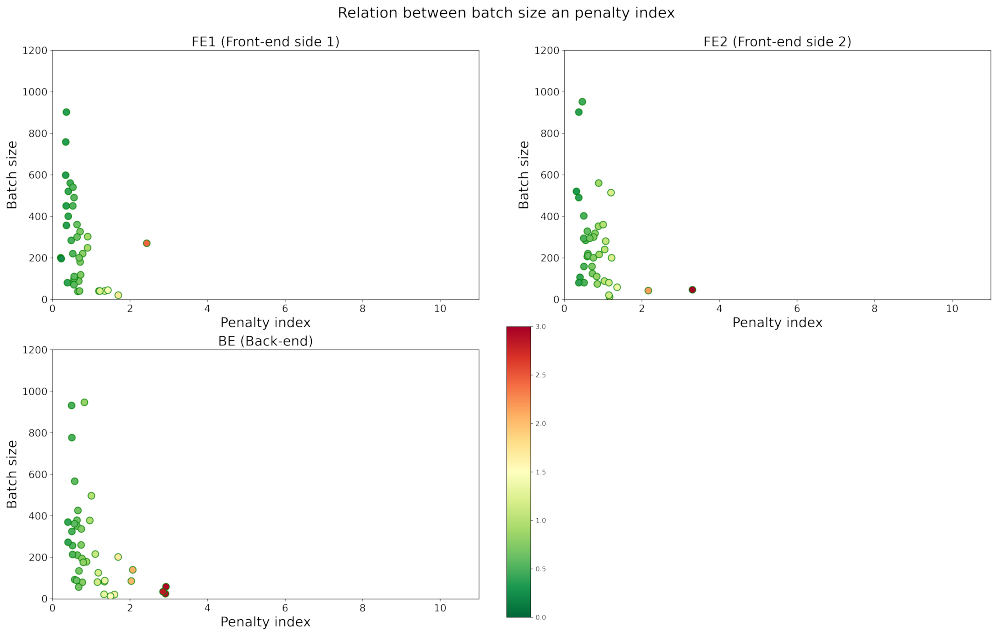


Figure 3.13 – Penalty indexes and batch sizes for each production phase.

Table 3.3 – Summary of batch quality.

Phase	Batch size		Penalty index of batches	
	Mean	**Standard deviation**	**% batch with penalty index < 1** (Good batches, only delays)	**% batch with penalty index $= 0$** (Compliant batches)
FE1	269.42	217.25	84.21	0
FE2	261.03	213.74	69.23	0
BE	241.55	229.25	50	0

— **Penalty index obtained for products in each phase:** The distributions of the penalty indexes for products in the three phases are shown in Figure 3.14. Table 3.4 presents statistics calculated on these distributions that allow for comparison of production between phases. As expected, more than 75% of products have penalty indexes between 0 and 1, which indicates good compliance with the process model. Among the three phases, $FE1$ is the most consistent, with a mean and median value of 0.54 and 0.4, respectively. Penalty indexes of products in the $FE1$ phase have less variation than in $FE2$ (as seen from their range and standard deviation). Note that $FE1$ and $FE2$ are the same operations performed on the first and second

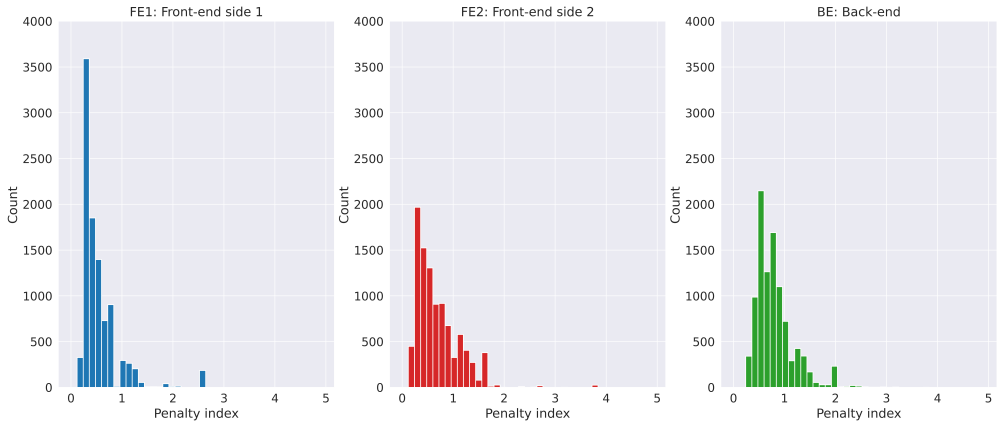


Figure 3.14 – Product quality for each production phase.

Table 3.4 – Summary of product quality.

Phase	**Mean**	**Median**	**Standard deviation**	**Range** (max − min)	**% products with penalty index < 1**
FE1	0.54	0.4	0.41	4.84	89.0
FE2	0.71	0.57	0.51	8.51	78.63
BE	0.8	0.75	0.38	3.52	76.38

sides of the PCB. These results are interesting and warrant further examination and improvement actions.

— **Global penalty index for products:** The global penalty index is computed once the product has undergone all phases. The distribution of this index for all products is shown in Figure 3.15. As expected from the previous results, most products (91.77%) have a small penalty index between 0 and 1. A detailed analysis should be performed on products with a high penalty index in association with process experts.

3.3.2 Analysis of product quality variation over work shifts

This section presents the analysis of production process performance through the obtained penalty indexes presented in Section 3.3.1. The overall performance of the assembly process is analyzed and compared across different time slots during one year (2019). The objective of this study is to verify whether there is a difference in performance between time slots and then to identify the factors that drive these differences.

The proposed approach uses a 2D Kernel Density Estimation (KDE) method to approximate the distribution of nominal products for each time slot and then compute their

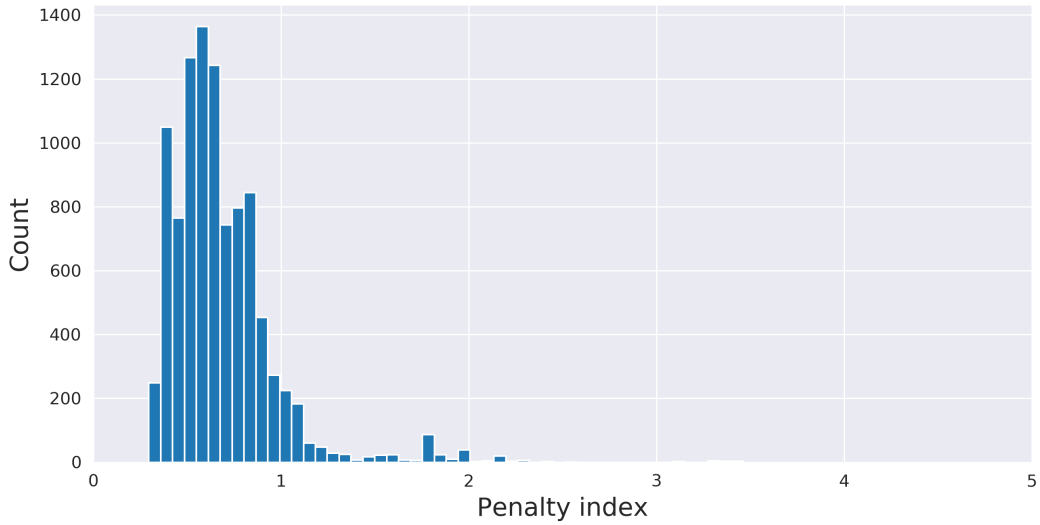


Figure 3.15 – Global penalty indexes for all products.

dissimilarity. The results are beneficial for the factory to evaluate the impact of the work shift and work time on production performance. From this point, the company can propose solutions to help improve the quality of work.

This study is performed on the Foix plant, which operates on a shift basis. This involves different groups of employees working the same job during agreed-upon periods throughout the day and night. The plant has three shifts on weekdays (morning, afternoon, and evening) and two shifts on weekends (morning and afternoon). We propose examining shift work's impact on the production process's performance. Many studies in the literature have been conducted on the matter of shift work. However, most of them evaluate the impact of this type of work organization on workers' health [60, 7, 47, 20, 10]. There is scarce research in the literature addressing the impact of shift work on production process performance and product quality [36]. Our objective is to fill this gap by presenting a quantitative analysis of this factor on product quality in a real electronic board manufacturing process.

Let us recall that in terms of production, the "nominal" class comprises all products that adhere to the nominal path and comply with all time constraints between operations. Therefore, it is considered that a high percentage of nominal products indicates that the production process operates correctly, with minimal interruptions, and that the product quality is guaranteed. Conversely, a low percentage indicates that a significant number

of products fail inspection or testing or do not meet time constraints. In such instances, identifying the root cause of the deviation is critical. The causes could be a malfunction of specific processes, a batch of raw material defects, or even a shift change. This study, however, focuses on the latter. An analysis of production performance during different work time slots is conducted to explore this issue.

We consider work slots of fixed duration: $\Delta t = 30$ minutes. The analysis is performed by comparing the performance of 48 production slots in a day, denoted as w_k, i.e., $k \in [1..48]$. Data are collected and computed for each slot and each day for a year to obtain the proportion and number of nominal products. Since the type and number of products produced per day vary based on customer demand, the analysis considers both the proportion and the number of nominal products. Given a work slot w_k, the study defines $x_k = (x_k^1, x_k^2) \in \mathbb{R}^2$ is a 2-dimensional vector, where x_k^1 and x_k^2 are the variables representing the proportion and the number of nominal products in the work slot w_k, respectively. Daily samples are indexed by j, i.e., $w_{k,j} = (x_{k,j}^1, x_{k,j}^2)$, where $j = 1, \ldots, N_k$, and the set of samples for every slot w_k is denoted by $W_k = \{(x_{k,j}^1, x_{k,j}^2)\}_{1 \leq j \leq N_k}$. It is important to note that the size N_k of W_k is not constant for all slots because there may be no production during specific time periods on some days. These breaks can be planned or unplanned. For instance, in the studied plant, there is no production from midnight to 5 a.m. every Monday, making these periods expected breaks. An example of an unexpected break time is a process breakdown or unplanned maintenance.

The difference in the size of the sets W_k, $k = 1, \ldots, 48$, raises a challenge in defining a dissimilarity metric. One straightforward approach is to extend the size of all sets to the same maximum size, but this results in a loss of information and may lead to inaccurate analysis. To overcome this issue, we propose to compare the two-dimensional distribution estimated from these data sets.

The workflow of our proposal to calculate the difference in performance between work slots for the case study is shown in Figure 3.16. To begin with, we extract data from the plant and apply the process mining framework presented in Section 3.3 to create a process model. We also classify products into different quality categories based on their deviation from the nominal path. This study takes into account the penalty index of all product families produced in a year. For performance analysis, we extract only the nominal quality class from the outputs of the process mining framework. Then we use a non-parametric method, namely KDE [71], and define dissimilarity using the L_1, L_2, and Jensen-Shannon distances [30] between obtained densities. The subsequent paragraphs provide more details

on the KDE approach used for estimating the nominal product distribution for each work slot and calculating their dissimilarity.

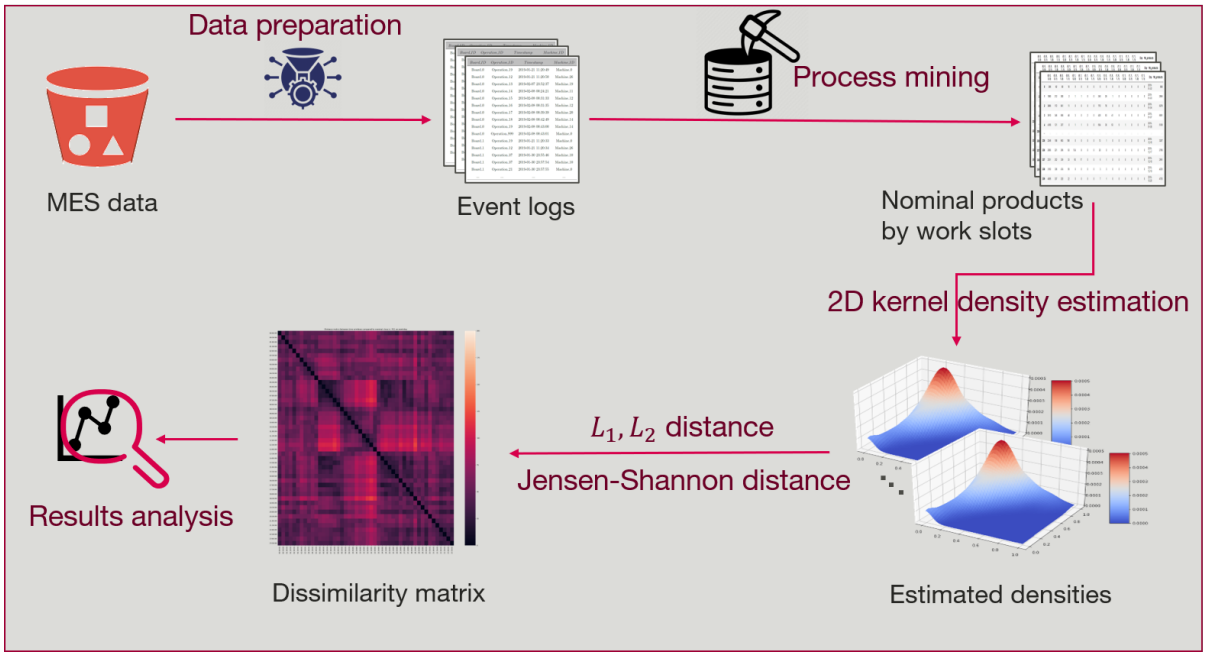


Figure 3.16 – Performance analysis workflow.

Kernel density estimation

In statistics, and particularly in statistical estimation, there are two categories: parametric and non-parametric statistics. In parametric statistics, the information about the distribution of a population is known and is associated with a finite set of parameters. Parametric methods are used to estimate these parameters, such as the mean, the variance, etc. On the contrary, non-parametric statistics are either distribution-free or use a specified distribution whose parameters are unspecified.

The KDE method is the most common non-parametric method to estimate the probability density function of continuous random variables. It is also known as the Parzen-Rosenblatt window method [71].

Definition 11 (Kernel density estimator of univariate distribution) *Consider* $\{X_i\}_{1 \leq i \leq n}$*, a random sample of size* n *drawn from an unknown probability density* f*. The*

kernel density estimator of f is:

$$\hat{f}_n^h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \qquad \forall x \in \mathbb{R} \tag{3.5}$$

where K is a kernel and h is the bandwidth or smoothing parameter.

Definition 12 (Kernel) *A* kernel *is a non-negative real-valued integrable function K such that*

- $\int_{-\infty}^{\infty} K(u)du = 1$
- *K is an even function, $K(-u) = K(u)$*

Table 3.5 shows some commonly used kernels.

Table 3.5 – Common kernels used in KDE.

Kernel	$K(u)$
Gaussian	$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$
Uniform (Tophat)	$\frac{1}{2}\mathbb{1}_{\lvert u\rvert \leq 1}(u)$
Epanechnikov	$\frac{3}{4}(1-u^2)\mathbb{1}_{\lvert u\rvert \leq 1}(u)$
Exponential	$\lambda \exp\left(-\lambda u\right)$
Linear	$(1-\lvert u\rvert)\mathbb{1}_{\lvert u\rvert \leq 1}(u)$
Cosine	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right)\mathbb{1}_{\lvert u\rvert \leq 1}(u)$

The smoothing parameter h controls the number of samples used to compute the probability for a new point. According to [71] and [104], the choice of h is much more important than the choice of K for the behavior of $\hat{f}_n^h(x)$. The quality of the approximation is controlled by the Mean Integrated Squared Error (MISE):

$$MISE(h) = \mathbb{E}_X\left[\|\hat{f}_n^h - f\|_{L_2}^2\right] = \int_{\mathbb{R}} \left(\hat{f}_n^h(x) - f(x)\right)^2 dx \tag{3.6}$$

$$= \int_{\mathbb{R}} \left(bias^2\, \hat{f}_n^h(x) + Var_X(\hat{f}_n^h(x))\right) dx \tag{3.7}$$

The error is decomposed into two terms. The first term is called bias, and the second term is variance. The quality of the estimation depends on the value of the bandwidth parameter h. We have the following properties:

— $Bias \to 0$ when $h \to 0$

— $Variance \to 0$ when $nh \to +\infty$

There is a trade-off between bias and variance. A large window of samples, i.e., a large value of h, may result in a very smooth density with a high bias. In contrast, a small window may have too much detail (high variance) and not be smooth or general enough to correctly cover new or unseen samples. Several approaches were developed to find the optimal value of h. Many of them are based on the assumption that data are sampled from a normal distribution, i.e., Silverman's rule [96], Scott's rule [90], Sheather and Jones method [95], etc. Cross-validation methods are another technique that does not use any assumptions about the data. These methods aim to fit the model to part of the data and then evaluate the remaining data.

The KDE can be extended to the multivariate case. The most general form is given by Definition 13 [117].

Definition 13 (Kernel density estimator of multivariate distribution) *Consider $\{Z_i\}_{1 \leq i \leq n}$, a p-variate random sample of size n drawn from an unknown probability density function $f : \mathbb{R}^p \to \mathbb{R}$. The kernel density estimator of f is:*

$$\hat{f}_n^H(x) = \frac{1}{n} \sum_{i=1}^{n} K_H (x - Z_i) \tag{3.8}$$

where $x = (x_1, x_2, ..., x_p)^T$ and $Z_i = (Z_i^1, Z_i^2, ..., Z_i^p)^T$. The bandwidth parameter H is a symmetric positive definite $p \times p$ matrix, and

$$K_H(x) = |H|^{-1/2} K(H^{-1/2} x) \tag{3.9}$$

The matrix H has $\frac{p(p+1)}{2}$ parameters. A simplified version of (3.8) can be obtained in (3.10) by choosing the matrix $H = diag(h_1^2, h_2^2, ..., h_p^2)$, allowing different amounts of smoothing in each of the coordinates.

$$\hat{f}_n^H(x_1, ..., x_p) = \frac{1}{n} \frac{1}{\prod_{l=1}^{p} h_l} \sum_{i=1}^{n} K \left(\frac{x_1 - Z_i^1}{h_1}, ..., \frac{x_p - Z_i^p}{h_p} \right) \tag{3.10}$$

Dissimilarity metric for performance analysis

To measure the dissimilarity between probability distributions, several metrics exist. Among those, both statistical and Euclidean distances are used. The aim is to learn whether the difference in the distances leads to different conclusions. Regarding statistical distance, Jensen-Shannon (JS), a commonly used distance, has been selected. The L_1 and

L_2 distances have been used for Euclidean distance. The L_1, L_2 distances between two density probability functions $g(x)$ and $g'(x)$ on a domain S of the Euclidean space are defined as:

$$d_{L_1}(g, g') = \|g - g'\|_{L_1} = \int_S |g - g'| dx \tag{3.11}$$

$$d_{L_2}(g, g') = \|g - g'\|_{L_2} = \sqrt{\int_S (g - g')^2 dx} \tag{3.12}$$

The JS distance is the root square of the Jensen-Shannon divergence, which is an extension of the Kullback-Leibler (KL) divergence [30]. KL divergence is a very common measure used in probability and statistics that computes a score indicating how much a probability distribution differs from another. Given g and g' the probability density functions of two continuous random variables, their Kullback-Leibler divergence is given by:

$$D_{KL}(g\|g') = \int_{-\infty}^{+\infty} g(x) \log\left(\frac{g(x)}{g'(x)}\right) dx \tag{3.13}$$

This measure has two problems. First, it is defined only if $\forall x, g'(x) = 0$ implies $g(x) = 0$. Second, the KL divergence score is not symmetrical, i.e., $D_{KL}(g\|g') \neq D_{KL}(g'\|g)$. The JS distance allows overcoming this later problem because it is symmetrical. The formula of JS distance is as follows:

$$D_{JS}(g\|g') = \sqrt{\frac{D_{KL}(g\|\bar{g}) + D_{KL}(g'\|\bar{g})}{2}} \tag{3.14}$$

where, $\bar{g} = \frac{g+g'}{2}$.

Application to the case study

The non-parametric density estimation method KDE is used because there is no assumption about the data distribution. Formula (3.10) is applied by choosing H as a diagonal matrix. As defined in the previous section, $W_k = \{(x^1_{k,j}, x^2_{k,j})\}_{1 \leq j \leq N_k}$ is the set of data representing the production in a 30-minutes work slot w_k.

According to (3.10), the 2D probability density function estimated by the KDE for each slot w_k is hence given by $\hat{f}_{k\,N_k}^{\,H_k}(x), k = 1..48$, simply denoted by $\hat{f}_k(x)$ for convenience,

as follows:

$$\hat{f}_k(x_1, x_2) = \frac{1}{N_k} \frac{1}{h_{k,1} \times h_{k,2}} \sum_{j=1}^{N_k} K\left(\frac{x_1 - x_{k,j}^1}{h_{k,1}}, \frac{x_2 - x_{k,j}^2}{h_{k,2}}\right) \tag{3.15}$$

where $H_k = diag(h_{k,1}^2, h_{k,2}^2)$.

The goal is to assess the dissimilarity of the estimated probability density functions over different time slots.

Computing the dissimilarity of probability density functions solves the problem of sets $(W_k)_{k=1..48}$ having different sizes. Indeed the density $\hat{f}_k(x)_{k=1..48}$ estimated respectively from $(W_k)_{k=1..48}$ can be compared as the dissimilarity between functions.

In this study, the Gaussian kernel is used as the basis function. The Leave One Out (LOO) cross-validation method is used for parameter optimization as the data are not normally distributed and the sample size is small [86]. This technique involves repeatedly fitting the model on a dataset with one observation removed and then using the fitted model to evaluate on the removed observation. The process is repeated for each observation in the dataset. The probability density function pairs (g, g') used in (3.11), (3.12), and (3.14) for the three dissimilarity metrics L_1, L_2, and the JS distances, respectively, are instantiated with all the possible pairs $(\hat{f}_\kappa, \hat{f}_\nu)$, $\kappa, \nu = 1..48$ and $\kappa > \nu$.

Results

The dataset used in this analysis consists of one year of production, including all process lines of several product families. The number and proportion of nominal products are computed for each time slot of 30 minutes. The study involves two main steps. In the first step, the distribution of nominal products in each time slot is estimated from a set of samples extracted during one year. In the second step, the dissimilarity matrix of the different time slots is calculated as the difference between the corresponding densities.

— **Kernel Density Estimation:** The first step in density estimation is to check if the data samples follow a normal distribution to determine if parametric estimation methods can be used. To do this, the Shapiro-Wilk test [93] is performed on all variables, as shown in Figure 3.17. The variables are separated into weekdays (a) and weekends (b), with each column representing the test result based on the p-value of 48 variables (denoted by $k = 1..48$) associated with either the proportion of nominal products x_k^1 or the number of products x_k^2 in a production phase (*FE1,*

FE2, BE).

	FE1		FE2		BE	
	x_k^1	x_k^2	x_k^1	x_k^2	x_k^1	x_k^2
Null hypothesis rejected (**p-value < 0.05**)	48	41	48	38	48	10
Null hypothesis accepted (**p-value ≥ 0.05**)	0	7	0	10	0	40
Total	48	48	48	48	48	48

(a) Weekday

	FE1		FE2		BE	
	x_k^1	x_k^2	x_k^1	x_k^2	x_k^1	x_k^2
Null hypothesis rejected (**p-value < 0.05**)	48	43	48	40	48	34
Null hypothesis accepted (**p-value ≥ 0.05**)	0	5	0	8	0	14
Total	48	48	48	48	48	48

(b) Weekend

Figure 3.17 – Results of Shapiro–Wilk test on the normality assumption of data on weekdays (a) and weekends (b)

The results show that most variables do not fit a normal distribution with a *p-value* < 0.05, indicating that parametric methods are not applicable in this case study. This test confirms our choice of using a non-parametric method, KDE.

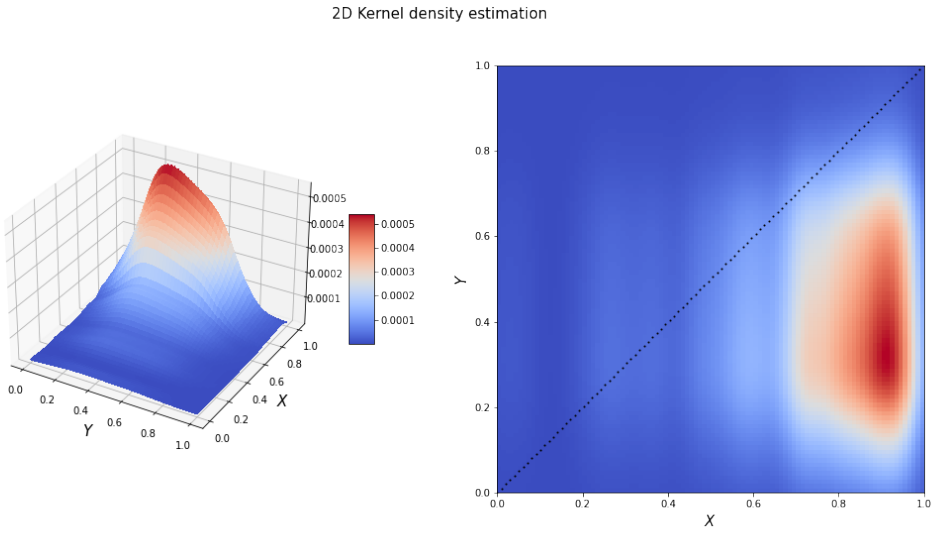
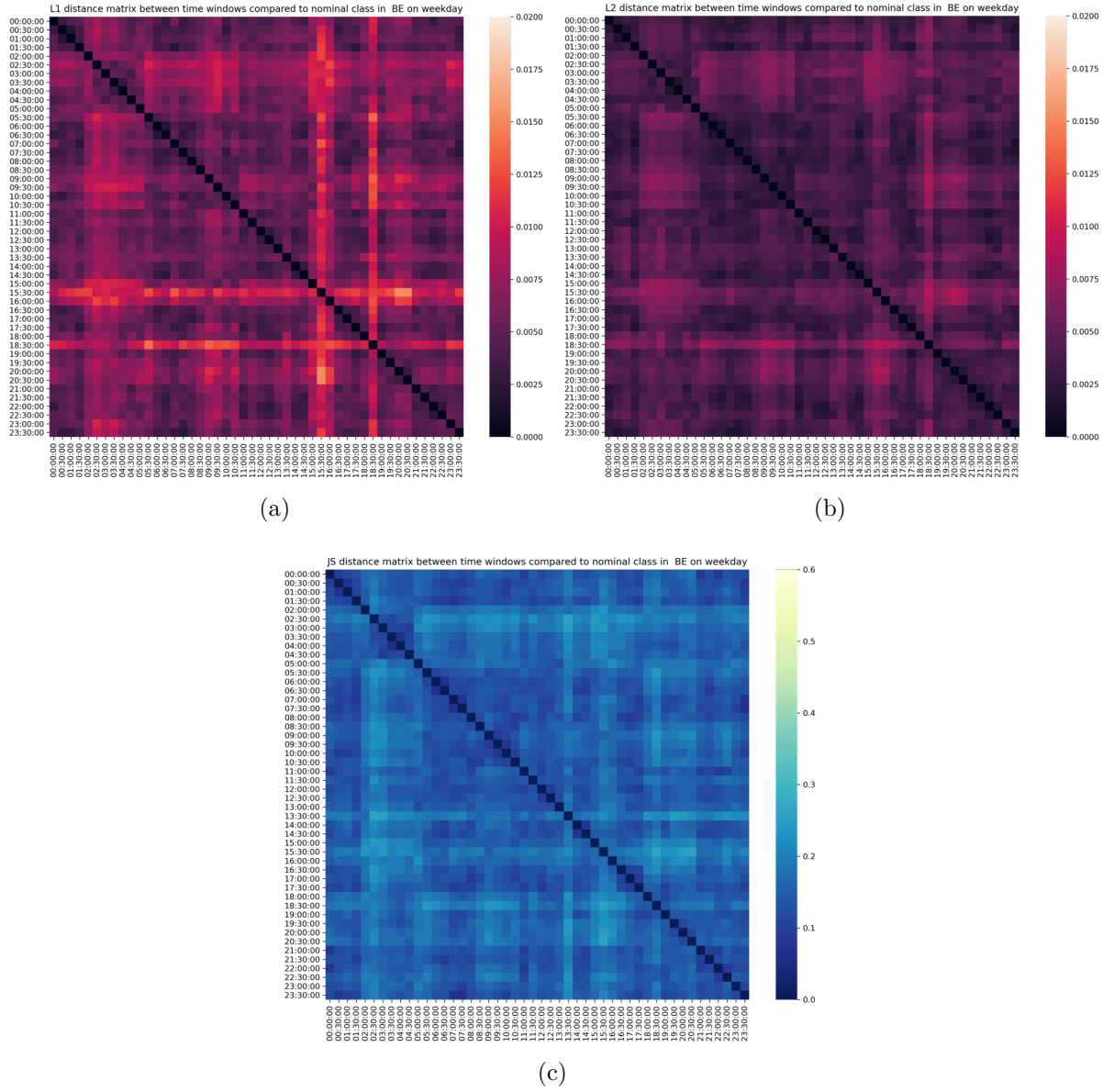


Figure 3.18 – Estimated density of nominal products in FE1 during time slot [12:00, 12:30] on weekdays. The density is visualized both in a 3D plot(**left**) and a 2D plot (**right**). The X-axis and the Y-axis represent the proportion of nominal products x_k^1 and the normalized number of nominal products $\tilde{x}_k^2$, with $k = 25$ corresponding to the time slot [12:00, 12:30], respectively.

As mentioned before, the choice of bandwidth parameters $h_{k,i|k=1..48,i=1,2}$ is much more important than the choice of kernel K. Hence, the study focuses on finding the optimal value of $h_{k,i}$. For this purpose, the well-known Gaussian kernel is used.

As data are not normally distributed, bandwidth selection techniques that rely on a reference distribution are not applicable. In this study, the cross-validation method is used. In particular, the Leave One Out (LOO) cross-validation is used as the sample size is small. The bandwidth parameter $h_{k,i}$ for each (k, i) is set among $\{0.03, 0.031, .., 0.1\}$ for the experimentation. The criterion used is the log-likelihood of the data under the estimated model to find the best parameter. Figure 3.18 shows an estimated 2D density obtained by the proposed method. This is the distribution of nominal products obtained in the $FE1$ phase during the time slot w_k = [12:00, 12:30] on weekdays. The X-axis represents the proportion of nominal products x_k^1, while the Y-axis represents the number of nominal products $\tilde{x}_k^2$ which is normalized by min-max normalization, i.e., $\tilde{x}_k^2 = \frac{x_k^2 - \min\limits_{k=1..48} x_k^2}{\max\limits_{k=1..48} x_k^2 - \min\limits_{k=1..48} x_k^2}$.

— **Dissimilarity matrices:** Given a production phase among FE1, FE2, and BE, a work slot w_k is represented by a 2D density $\hat{f}_k$. The dissimilarity matrix $\mathcal{D}$ is a symmetric matrix of size 48×48 where each row or column corresponds to a time slot. Each element $\mathcal{D}_{\kappa\nu}$, $\kappa, \nu = 1..48$, of the matrix is the dissimilarity measure of two corresponding time slots w_κ and w_ν, i.e., $\mathcal{D}_{\kappa\nu} = d(w_\kappa, w_\nu)$.
$\mathcal{D}$ is calculated as follows. First, the exact value of the two estimated densities $\hat{f}_\kappa$ and $\hat{f}_\nu$ are computed at each point of a grid G. The grid is defined in $[0, 1]^2$ with size of 100×100. $\mathcal{D}_{\kappa\nu}$, $\kappa, \nu = 1..48$, are then obtained with the three proposed metrics L_1, L_2, and JS distance, completing the calculation of $\mathcal{D}$. Figure 3.19 presents the dissimilarity matrices between time slots in the production phase BE on weekdays. The matrix in Figure 3.19(a) is computed using the L_1 metric, while Figure3.19(b) and 3.19(c) represent the dissimilarity matrices computed by L_2 and JS metric, respectively. In all three figures, darker colors represent lower dissimilarity, while lighter colors show higher dissimilarity. Since the JS metric has a different scale from the L_1 and L_2 metrics, a different color bar is used to make it more readable. The first remark that stands out is that L_1 distance presents higher dissimilarities. In contrast, the L_2 distance matrix is darker. This result is understandable because the difference at each point of two densities is smaller than 1, so this value becomes even smaller when squared. Secondly, it is unclear to recover work shifts from these matrices. Remind that the three work shifts on weekdays are [5:30, 13:30], [13:30, 21:30], [21:30, 5:30]. This result shows no effect of shift work on product quality.
However, the question arises as to what level of dissimilarity means that two dis-



(a) (b)

(c)

Figure 3.19 – BE production phase: dissimilarity matrices between 30 minutes time slots computed from 3 different metrics: L_1 (a), L_2 (b) and JS (c).

tributions are sufficiently different. The answer is given by experimentation. For that, the pair of slots with the largest dissimilarity in the L_1 distance matrix is chosen to draw the densities. Figure 3.20 presents the estimated densities of two selected slots. Figure 3.20a, 3.20b present the 2D densities, and Figure 3.20c, 3.20d present the 1D densities of each dimension. The 2D density associated with the

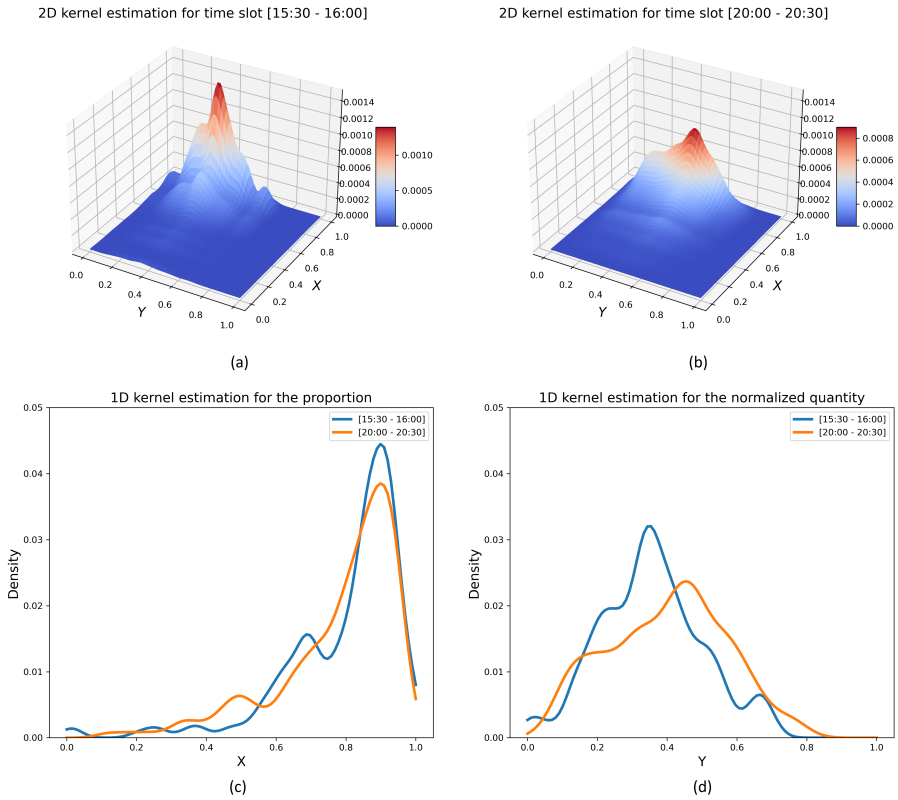


Figure 3.20 – Visualization of two estimated 2D densities for work slot [15:30-16:00] (**a**) and [20:00-20:30] (**b**) by the KDE method. The X-axis represents the proportion of nominal products. The Y-axis represents the normalized number of nominal products. The 1D densities for each axis X (**c**) and Y (**d**) are also plotted.

slot [20:00-20:30] is flatter, and more spread out than [15:30-16:00]. Regarding 1D density, there is not much difference between the two densities of the proportion of nominal products (X-axis). According to the normalized quantity of nominal products (Y-axis), the distribution that corresponds to the slot [20:00-20:30] is slightly shifted to the right compared to that of [15:00-16:00]. This means that it tends to take on larger values. Nevertheless, again, this difference is minimal. This result reinforces the assessment that there is a homogeneity of quality between slots and no shift work effect on the production performance. Even if the analysis doesn't show significant variability between shifts, it is still important to provide relevant

information to the quality service and management. This allows for a comprehensive understanding of the production process and can help identify potential areas for improvement. Additionally, the approach used in this analysis is interesting because it is generic and can be applied to other manufacturing processes. By using this approach, companies can gain insights into their production processes in all plants, which can lead to better decision-making and increased efficiency.

3.4 Monitoring of work-in-progress

Tracking the movement of products and the number of products being processed is a crucial task in production scheduling and monitoring. However, this is a challenging task that requires a significant amount of information and expertise in the field. Moreover, the calculation of work-in-progress (WIP) must integrate a vast amount of information, which makes the work time-consuming and prone to errors. To address these challenges, we have developed a tool that can automatically and accurately compute the number of WIPs.

In our case study, WIP is defined as partially finished goods that are waiting to move to the next process for completion. Calculating WIP requires integrating several pieces of information, such as product movement through phases, product status (pass/fail), packaging information, etc., which can be extracted from the event log database. Our developed program automatically extracts data and calculates WIP in real-time while also generating visualizations of WIP in different phases and for different product families, as well as the evolution of WIP over time.

To formulate the WIP calculation problem in each production phase, we define the WIP of phase ϕ_i at time t, denoted $E^m_{i,t}$, as the number of products that are waiting to move through phase ϕ_i, are not scrap, and were introduced into production less than m days ago.

To quantify the products waiting for a phase ϕ_i at time t, we search for products that finished the previous phase ϕ_{i-1} through events that occurred before time t. Let p be a product, and the state of a product p at time t be represented by a vector $[j_p, t^0_p, e_p, t^-_p]$, where

— j_p is the serial number of p, which is unique for each product.
— t^0_p is the date when the product was introduced into production.
— e_p is the last event associated with product p.

— t_p^- is the timestamp associated with event e_p (let $t_p^- \leq t$).

Then, the WIP of phase ϕ_i at time t is calculated as:

$$E_{i,t}^m = card(p|e_p \bowtie \phi_{i-1}; t_p^- < t; t_p^- - t_p^0 \leq m) \tag{3.16}$$

Where, $e_p \bowtie \phi_{i-1}$ means that event e_p corresponds to the last operation in phase ϕ_{i-1}.

The application we are using provides an interactive dashboard on PowerBI to track work-in-progress (WIP) at the plant, which is displayed in Figure 3.21. This dashboard computes WIP values, the number of packaged products (EMBAL), and products in rebut. It also allows users to visualize each quantity by product family (treemap on the top and barplot on the bottom left), production line, and product type (Pass/Fail). Furthermore, the time slide located in the top left of the dashboard allows users to filter products based on their duration in the assembly process, ranging from 0 to 200 days. The production managers find this dashboard to be an essential tool for monitoring and planning production. Moreover, it enables us to prepare a WIP dataset, which we can use to study and develop prediction models (Section 4.1).

Figure 3.21 – PowerBI dashboard for tracking of WIP for all processes in Foix plant, France.

The visualization depicted in Figure 3.21 is computed and updated every 30 minutes.

Additionally, a real-time version has been developed using Lambda functions on AWS. For this particular case, Grafana is employed for the visualization, as shown in Figure 3.22.

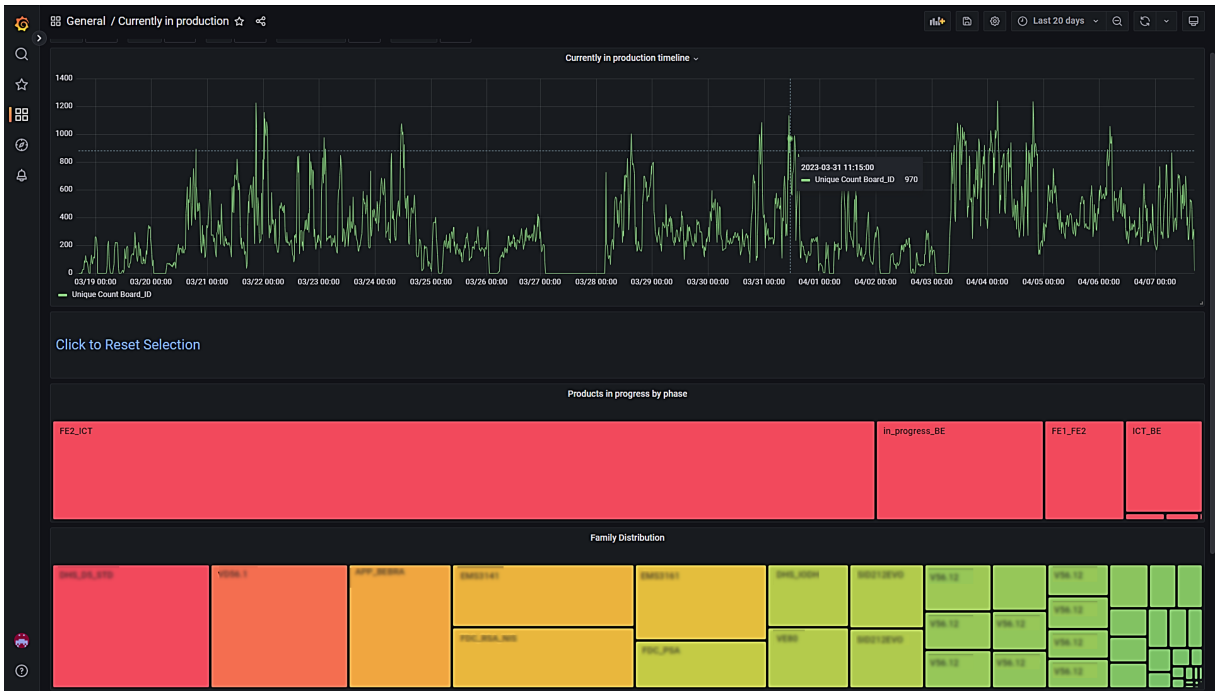


Figure 3.22 – Grafana dashboard for real-time tracking of WIP for all processes in Foix plant, France.

3.5 Conclusion

To conclude, this chapter has provided an overview of our contributions to the field of descriptive process mining, a crucial aspect of process mining. The basic concepts of process mining, such as event log, event, trace, prefix, etc., were introduced. The chapter has also focused on the three main tasks in descriptive process mining, namely process discovery, conformance checking, and work-in-progress calculation, at which level our contributions are placed. To compute the timed process model for process discovery, an ad-hoc process discovery method was proposed. The conformance checking task involved comparing the process model with the actual behavior of the system, and the work-in-progress calculation was used to monitor the production process by tracking the number of unfinished products. It is used daily in the Vitesco Technologies plant of Foix by the

operational teams.

The results of applying these techniques to real-world event logs were presented, which demonstrated their effectiveness in providing insights into process behavior and identifying process inefficiencies. Process discovery allowed identifying the most frequent paths in the process, while conformance checking revealed deviations between the process model and actual behavior. Work-in-progress calculation helped to identify bottlenecks and areas for improvement in the process.

Moving forward, the next chapter will focus on predictive process mining, which uses historical data to predict the future behavior of a process. Predictive process mining leverages advanced analytical techniques to provide valuable insights into process performance and enable organizations to make data-driven decisions.

Chapter 4

Predictive Process Mining

Summary of Chapter 4

This chapter explores two distinct prediction problems. The first problem focuses on predicting the Work-In-Progress (WIP) between production phases based on past variations. A range of prediction methods is used, from the classic linear regression model to more advanced LSTM models. The second problem involves predicting the Remaining Cycle Time (RCT) of ongoing products in the assembly line. This problem introduces the use of Graph Neural Network (GNN) models for the first time, and the results indicate that GNNs have the potential to be used in manufacturing processes. Overall, the chapter highlights the importance of predictive process mining in improving the efficiency and productivity of manufacturing processes.

Predictive Process Mining (PPM) is a subfield of process mining that encompasses a range of techniques aimed at predicting the outcome or future properties of an ongoing process case. To achieve this goal, input is derived from historical data of process executions that are stored in event logs. A predictive model is then constructed and trained based on the specific needs of the situation in order to predict the relevant information pertaining to the current process. Numerous PPM problems have been tackled and resolved, and this study focuses on two prediction problems. Section 4.1 presents the prediction of future WIP values between production phases based on their evolution in history. In Section 4.2, we explore the work done to predict the RCT of ongoing products in the production process.

4.1 Prediction of work-in-progress between production phases

In this section, we will explore the prediction of WIP between production phases. Accurate prediction of WIP is crucial for businesses to ensure that they meet production targets and avoid unnecessary delays. We have utilized various prediction methods, ranging from classic approaches such as linear regression and AutoRegressive Integrated Moving Average (ARIMA) models to more complex methods like Long Short-Term Memory (LSTM) models. These methods have been proven to be effective in forecasting WIP, and we will delve into the details of each method and its respective strengths and weaknesses. This section provides a comprehensive understanding of the different prediction techniques and gives the means to choose the most suitable approach for specific business needs.

4.1.1 Data preparation

Data preparation is essential for predicting the WIP between production phases in the assembly process. Hourly WIP values are calculated between successive phases $\phi_1 = FE1$, $\phi_2 = FE2$, $\phi_3 = ICT$, and $\phi_4 = BE$. In this case study, a training dataset of WIP values was created using historical data collected by Manufacturing Execution System (MES) from 2020 and 2021. The WIP variable is denoted as $E^m_{i,t}$. As explained in Section 3.4, this variable represents the number of products waiting to move through phase ϕ_i at time t and that entered production less than m days ago. In this context, the process experts set m to 200 days. Therefore, the WIP variable is referred to as $E_{i,t}$ for further analysis. The dataset is preprocessed by excluding periods of no production due to maintenance, holidays, and other reasons. The prediction of the WIP value focuses on the period between $FE1$ and $FE2$, which is denoted as $E_{2,t}$. Figure 4.1 shows the variation of $E_{2,t}$ over the two-year period. The dataset is divided into three parts: 70% for training data in green, 20% for validation data in violet, and 10% for test data in blue. The split is performed based on time to preserve the sequential nature of the data. It is important to ensure that the validation and test sets come after the training set in time. Standardization (Z-score normalization) was performed, which scales the data to have a mean of 0 and a standard deviation of 1. This was achieved by subtracting the mean of the time series from each value and dividing it by the standard deviation.

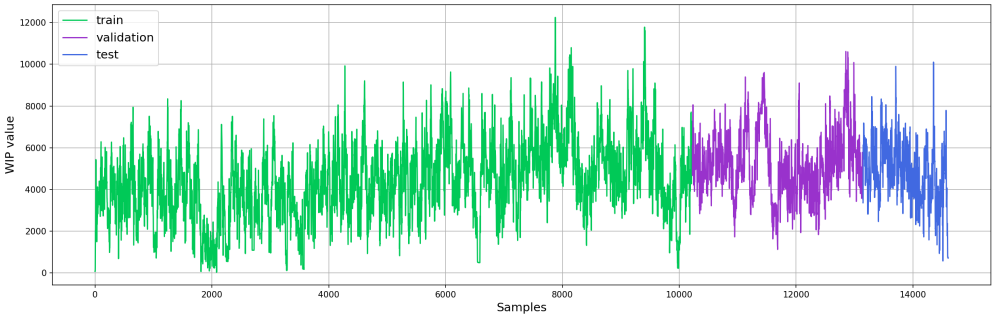


Figure 4.1 – WIP values variation in two years, 2020 and 2021, between $FE1$ and $FE2$.

4.1.2 Prediction models

ARIMA

ARIMA is a statistical model for time series forecasting using the past values [12]. It combines AR, MA, and differencing components. AR component captures the linear relationship between past observations and the current value, while the MA component captures the error term of the past observations. The differencing component helps in making the time series stationary, which means that the mean, variance, and autocorrelation structure of the series do not depend on time. The model is represented by the notation $ARIMA(p, d, q)$, where:

- p is the order of the autoregressive component, which represents the number of lagged observations used to predict future values,
- d is the degree of differentiation needed to make the time series stationary, which involves taking the difference between consecutive observations at least once or multiple times until the series becomes stationary,
- q is the order of the moving average component, which represents the number of lagged forecast errors used to predict future values.

The mathematical formula for the $ARIMA(p, d, q)$ model can be expressed as:

$$y(t) = c + \Phi(1)y(t-1) + ... + \Phi(p)y(t-p) + \epsilon(t) + \theta\epsilon(t-1) + ... + \theta(q)\epsilon(t-q) \quad (4.1)$$

where:

- $y(t)$ is the time-series,
- c is a constant,
- $\Phi(1), ..., \Phi(p)$ are autoregressive coefficients for lag 1 to p,

— $\epsilon(t)$ is the error term or innovation at time t,
— $\theta(1), ..., \theta(q)$ are the moving average coefficients for lag 1 to q,
— $\epsilon(t-1), ..., \epsilon(t-q)$ are the error terms for lag 1 to q.

Linear regression

Linear regression is a statistical technique used to model the relationship between a dependent variable Y and one or more independent variables X. The model assumes a linear relationship between the variables and tries to fit a straight line that best describes the relationship between them. There are two types of linear regression: simple linear regression and multiple linear regression. In simple linear regression, there is only one independent variable X, while in multiple linear regression, there are p independent variables X_1, X_2,..., X_p, etc.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{4.2}$$

where:

— Y is the dependent variable, which we want to predict or explain,
— $X_1, X_2, ..., X_p$ are the independent variables (also called the explanatory variables or predictors),
— $\beta_0, \beta_1, ..., \beta_p$ are the coefficients (also called the regression coefficients or parameters). β_0 is the intercept, which represents the value of Y when all the independent variables are equal to zero. $\beta_0, \beta_1, ..., \beta_p$ represent the change in Y for a unit change in $X_1, X_2, ..., X_p$ respectively,
— ϵ is the error term, which represents the variability in Y that is not explained by the model.

The goal of linear regression is to estimate the values of $\beta_0, \beta_1, ..., \beta_p$ that minimize the Sum of Squared Errors (SSE) between the predicted values and the actual values of Y. This is done using a method called Least Squares [31], which involves finding the line that minimizes the sum of the squared differences between the predicted and actual values of Y.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.3}$$

where:

— y_i is the actual value of the dependent variable for the i-th observation,
— $\hat{y}_i$ is the predicted value of the dependent variable for the i-th observation, calcu-

lated as Equation (4.2).

Once the regression coefficients are estimated, the linear regression model can be used to make predictions about the dependent variable based on the values of the independent variables.

XGBoost

XGBoost (eXtreme Gradient Boosting) is a popular machine learning algorithm used for both regression and classification [**friedman2001greedy**]. It is an ensemble learning method that combines multiple decision trees to create a model that can make accurate predictions. It is an extension of the gradient boosting algorithm that uses a more regularized model to prevent overfitting and achieve better performance.

The XGBoost algorithm is based on a series of decision trees that are trained sequentially, with each new tree attempting to correct the errors of the previous trees. The output of the model is the weighted sum of the predictions of each tree. The XGBoost algorithm combines gradient descent and Newton's method to optimize a loss function. The loss function is typically defined as the sum of a training error function and a regularization function:

$$Obj(\theta) = L(\theta) + \Omega(\theta) \tag{4.4}$$

where θ represents the model parameters, $L(\theta)$ is the training error function, and $\Omega(\theta)$ is the regularization function.

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) \tag{4.5}$$

where y_i is the actual value of the target variable for the i-th instance, $\hat{y}_i$ is the predicted value, and $l(y_i, \hat{y}_i)$ is the loss function (sum of squared errors for regression problems or cross-entropy loss for classification problems).

$$\Omega(\theta) = \lambda_1 \sum_{j=1}^{k} |\theta_j| + \lambda_2 \sum_{j=1}^{k} \theta_j^2 \tag{4.6}$$

where k is the number of parameters. The regularization term controls the complexity of the model that can prevent overfitting. This is typically a function of the model's parameters, and it is designed to encourage the model to have smaller parameter values. λ_1 and λ_2 are the regularization strength hyperparameters, which control the relative weight of the regularization term compared to the data fitting error term.

LSTM

LSTM is a type of Recurrent Neural Network (RNN) that is designed to handle the vanishing gradient problem of traditional RNNs [38]. LSTMs can learn long-term dependencies in sequential data and are widely used in tasks such as speech recognition, natural language processing, and time-series prediction.

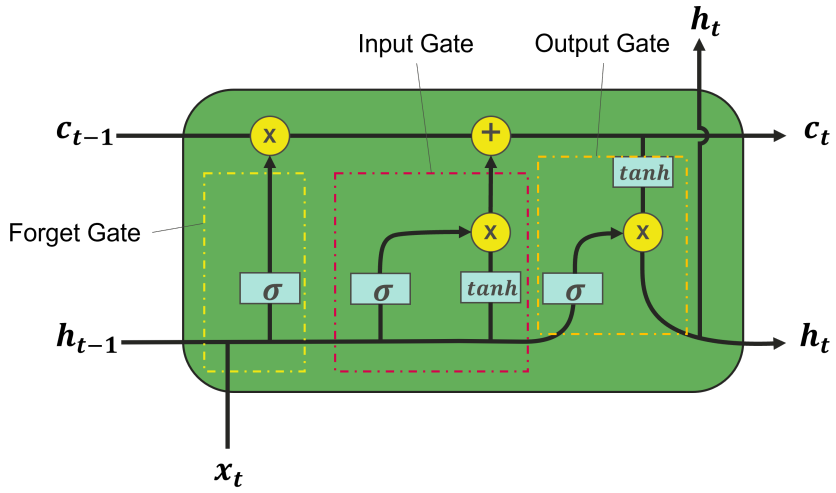


Figure 4.2 – The architecture of an LSTM cell.

The basic building block of an LSTM cell consists of three gates (Figure 4.2): the input gate i_t, the forget gate f_t, and the output gate o_t. Each gate is controlled by a sigmoid activation function, which produces output values between 0 and 1.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{4.7}$$

The input gate determines how much of the new input should be included in the cell state c_t. It takes the concatenation of the input x_t at time step t and the output h_{t-1} from the previous time step $t-1$ as its input (4.7). In these equations, W is the weights matrix, and b represents the bias term.

The forget gate f_t decides which information from the long-term memory stored in the cell state c_{t-1} should be kept or discarded. It takes the concatenation of the input x_t at time step t and the output h_{t-1} from the previous time step $t-1$ as its input (4.8). The forget gate is then multiplied with c_{t-1} to determine how much of the previous cell state should be retained (4.9).

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{4.8}$$

Then, the cell state is updated by combining the output of the input gate i_t and the output of the forget gate f_t and adding it to the previous cell state c_{t-1} (4.9). The updated cell state c_t is then passed through the output gate to obtain the output h_t for the current time step t.

$$c_t = f_t * c_{t-1} + i_t * tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4.9}$$

Finally, the output gate produces the short-term memory h_t from the current input and the current cell state c_t. It takes the concatenation of the input x_t and the output h_{t-1} from the previous time step, and also takes the current cell state c_t as its input (4.10, 4.11).

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{4.10}$$

$$h_t = o_t * tanh(c_t) \tag{4.11}$$

4.1.3 Experimentations and results

The previous section presents models used in this study to predict the WIP values from historical data. ARIMA and linear regression are simple models, while XGBoost is expected to provide more accuracy due to its ensemble-based approach. The LSTM model, on the other hand, can capture the long-term dependencies and nonlinear relationships and is expected to provide the best results among all the models. This section presents the implementation and prediction results obtained from these models.

Univariate and one-step prediction

In the first instance, the study focuses on the problem of one-step prediction using a single variable. This means that the model Ω takes as input the past values $E_{2,t}, E_{2,t-1}, ..., E_{2,t-k+1}$ depicted in the k most recent values (window size) to predict the future value in one time step, as shown in Equation (4.12).

$$\hat{E}_{2,t+1} = \Omega(E_{2,t}, E_{2,t-1}, ..., E_{2,t-k+1}) \tag{4.12}$$

Initially, we attempted to predict the WIP using the ARIMA model, but the results were unsatisfactory. The model failed completely in this case. Its poor prediction could be due to the fact that the data is not stationary and does not exhibit clear patterns, which the ARIMA model considers as random noise. We then compared three models: linear regression, XGboost, and LSTM. We evaluated the performance of these models in terms of computation time and prediction errors. Regarding prediction errors, we use three metrics commonly used in machine learning: MAE, MSE, and MAPE.

— Mean Absolute Error (MAE) measures the mean of the absolute differences between the actual and predicted values,
— Mean Squared Error (MSE) measures the mean of the squared differences between the actual and predicted values,
— Mean Absolute Percentage Error (MAPE) measures the average of the absolute percentage difference between the predicted and actual values.

In the case of linear regression and XGBoost models, the k most recent values are used as k predictors (explanatory variables) $X_1, X_2, ..., X_k$ and there is no temporal order between these variables. Note that in this case, these two models process k variables, but for time series forecasting, they still fall under the univariate category. For the XGBoost model, we defined a set of hyperparameters to be optimized, including `max_depth`, `learning_rate`, `subsample`, `colsample_bytree`, `min_child_weight`, and `n_estimators`.

➢ `max_depth`: `{3,4,...,10}`
➢ `learning_rate`: `{0.01, 0.05, 0.1, 0.2}`
➢ `subsample`: `{0.5,0.6,...,1}`
➢ `colsample_bytree`: `{0.5,0.6,...,1}`
➢ `min_child_weight`: `{0,1,...,5}`
➢ `n_estimators`: `{100, 300, 500}`

We used the `RandomizedSearchCV` method to vary these hyperparameters and chose the parameter setting that minimizes the mean square error. We then used these optimized parameter values to make predictions and evaluate the model performance.
As for the LSTM model, it takes as input the sequence of k values. Figure 4.3 presents the architecture of the 2-layers LSTM model used to predict the WIP in a one-time step. In particular, the WIP value at each time step is iteratively fed into the model to compute the hidden states at the output. These hidden states are stored and then passed as input to the next layers to compute the final hidden state h_k^2. This feature vector is then used

to predict the WIP value $\hat{E}_{2,t+1}$ by passing through a Fully Connected (FC) layer.

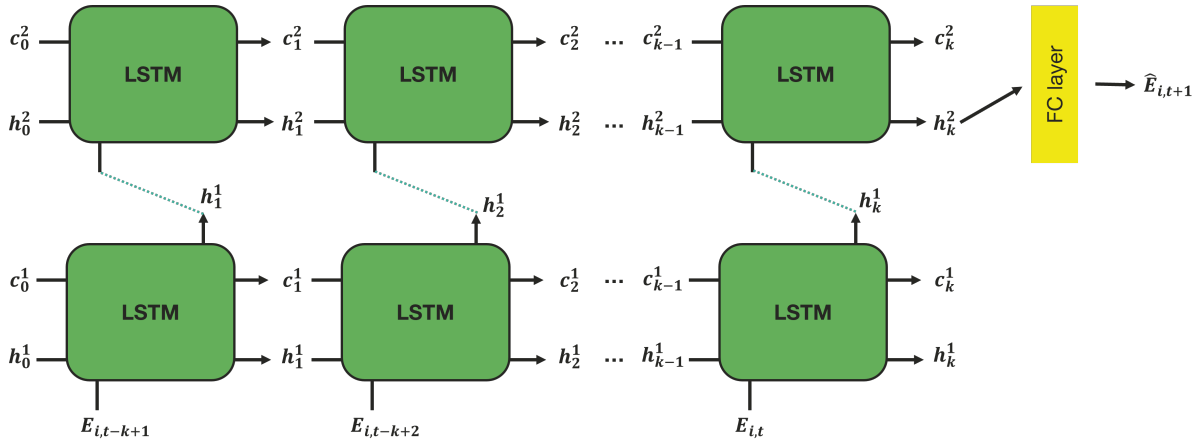


Figure 4.3 – The architecture of LSTM model for the WIP prediction.

For the implementation of the LSTM model, we used MSE as the metric to minimize and the optimizer Adam. The training was conducted for 20 epochs with a batch size of 32. We do not use more epochs as the model converges rapidly within the 20-epochs range. Additionally, since the problem is relatively straightforward, we selected a batch size of 32, which is commonly used in neural network training.

After fitting the models, they undergo evaluation on the test dataset (the blue part in Figure 4.1). For this study, we experimented with predicting from various window sizes k ranging from 2 to 24. As each time step represents an hour, the model uses past values from the preceding 2 hours to 24 hours to predict the WIP value for the subsequent hour. In Figure 4.4, we compare the performance of the three models: linear regression, XGBoost, and LSTM. In Figures 4.4(a), 4.4(b), and 4.4(c), we present the prediction errors (MAE, MSE, and MAPE) computed on the test set for each window size. Figure 4.4(d) depicts the computation time required for model training. The results indicate that XGBoost is the least effective among the three models, according to all metrics. Linear regression and LSTM have comparable performance in terms of prediction error, but linear regression is faster in computation.

Figure 4.5 presents the prediction results obtained from the linear regression model with window size $k = 8$. Figure 4.5(a) displays all test samples, whereas Figure 4.5(b) focuses on the first 100 samples. The model appears to perform well, with a MAPE of approximately 6.26% on the test dataset. However, upon closer inspection of the zoomed-in figure, it becomes apparent that the model struggles to predict trend reversal in WIP

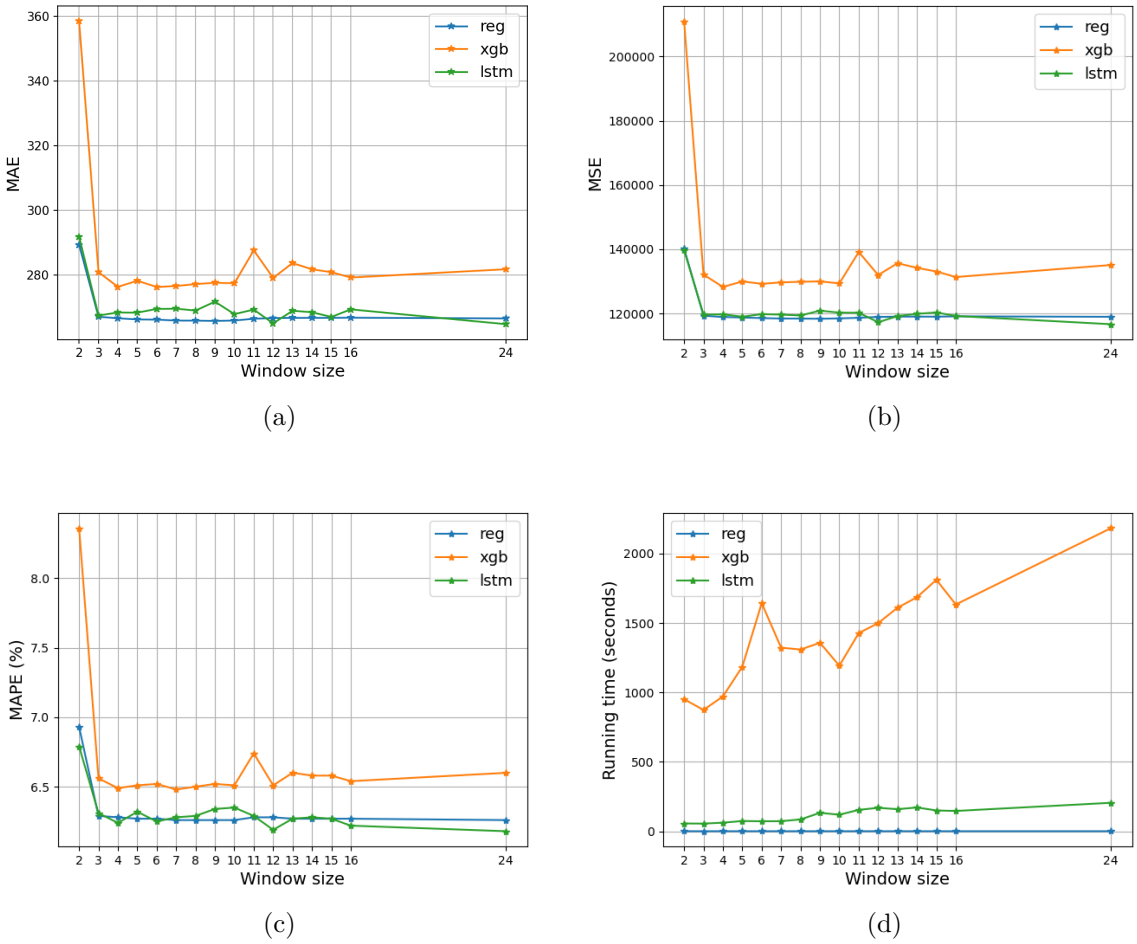


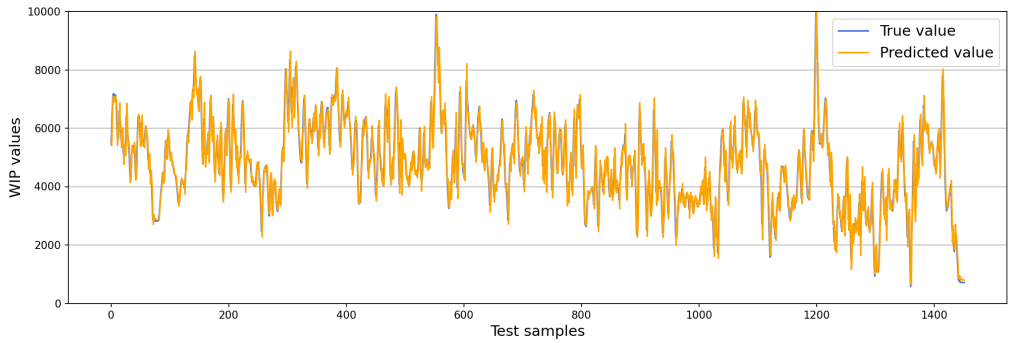
Figure 4.4 – Performance comparison of three models: Linear regression, XGBoost, and LSTM regarding the prediction errors MAE (a), MSE (b), MAPE (c) and the computation time (d).

(red circles). To improve prediction accuracy, we need to integrate more information. One potential solution is to incorporate WIP values from other phases in the process.

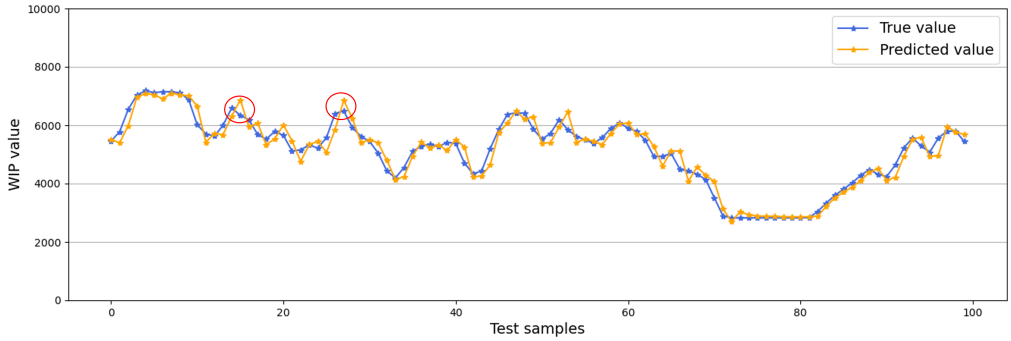
Univariate and multi-step prediction

This section describes the multi-step prediction of WIP values. Instead of predicting only the next WIP value, the model predicts a sequence of p future values.

$$(\hat{E}_{2,t+1}, \hat{E}_{2,t+2}, ..., \hat{E}_{2,t+p}) = \Omega(E_{2,t}, E_{2,t-1}, ..., E_{2,t-k+1}) \tag{4.13}$$



(a)



(b)

Figure 4.5 – Comparison of predicted (orange) and true (blue) values in test dataset: All data (a) and 100 samples (b). Trend reversal points are indicated by red circles.

In this study, we assessed two models that showed optimal performance for the one-step prediction task: linear regression and LSTM. We used a window size of $k = 8$ and a prediction horizon of $p = 4$. The linear regression model recursively predicts each value using the previously predicted values as input for the next prediction. On the other hand, the LSTM model directly predicts a 4-dimensional vector. Table 4.1 presents prediction errors of both models: Linear regression (a) and LSTM (b). The results show that they had similar performance for all steps. However, the models did not perform well in predicting WIP values at time steps $t + 2$, $t + 3$, and $t + 4$, indicating that they are not suitable for multi-step WIP prediction.

(a) Linear regression

Time step	MAE	MSE	MAPE (%)
t+1	265.87	118363.14	6.23
t+2	527.74	462386.46	12.25
t+3	711.86	847113.56	16.24
t+4	820.42	1109843.35	18.47

(b) LSTM

Time step	MAE	MSE	MAPE (%)
t+1	276.62	125306.22	6.34
t+2	530.68	465229.21	12.06
t+3	711.29	843660.46	16.00
t+4	821.85	1113086.44	18.38

Table 4.1 – Univariate prediction errors for multi-step.

Multivariate and one-step prediction

As stated in the previous sections, relying solely on information about past values of WIP in a single phase is insufficient for accurately predicting future values, particularly beyond t+2. Therefore, in this study, we incorporated WIP information from other phases to enhance the model. Specifically, in order to predict $E_{2,t}$ between $FE1$ and $FE2$, we used not only the historical variation of $E_{2,t}$, but also the WIP values between $FE2$ and ICT ($E_{3,t}$) and between ICT and BE ($E_{4,t}$). Consequently, the prediction problem, in this case, became multivariate, as shown below:

$$\hat{E}_{2,t+1} = \Omega\left(\begin{pmatrix} E_{2,t} \\ E_{3,t} \\ E_{4,t} \end{pmatrix}, \begin{pmatrix} E_{2,t-1} \\ E_{3,t-1} \\ E_{4,t-1} \end{pmatrix}, \ldots, \begin{pmatrix} E_{2,t-k+1} \\ E_{3,t-k+1} \\ E_{4,t-k+1} \end{pmatrix}\right) \tag{4.14}$$

Table 4.2 presents the prediction error obtained from multivariate models, as compared to the error obtained in the case of univariate models. The results indicate that incorporating WIP values from other phases into the model input does not yield significant improvement.

In summary, the results of the WIP prediction indicate that the past values of WIP can be used to predict the value of WIP in the next hour, and linear regression performs the best out of all the models tested. Additionally, the analysis shows that trend reversal points cannot be predicted, and including WIP information from other production phases does not improve prediction accuracy. These findings suggest that additional sources of information may need to be integrated to obtain more accurate predictions.

(a) Linear regression

Model type	MAE	MSE	MAPE (%)
uni-	265.87	118363.14	6.23
multi-	264.34	117647.54	6.21

(b) LSTM

Model type	MAE	MSE	MAPE (%)
uni-	276.62	125306.22	6.34
multi-	272.99	122738.89	6.26

Table 4.2 – Prediction errors from multivariate vs. univariate for one-step prediction of WIP.

4.2 Remaining cycle time prediction

The Remaining Cycle Time (RCT) is the amount of time needed for a process instance to complete from its current state. Accurately predicting the RCT of ongoing process instances is a crucial aspect of predictive process mining. It allows for the extraction of valuable insights into process performance, identification of bottlenecks, and proactive measures for enhancing process flow. In the literature, various approaches are employed to predict the RCT, which depend on input data, the prefix encoding method, process awareness, and the family of algorithms. Neural networks have shown promising results against classical methods in different benchmarks [111]. In this study, we focus on GNNs, a new type of network that has recently demonstrated good performance in PPM but has not yet been applied in RCT prediction. We present the first use of the GNN model and compare its performance with the LSTM model of [102], which serves as a baseline due to its highly promising results in the benchmark presented in [111].

4.2.1 Problem statement

This section provides the formalization of the RCT problem. Table 4.3 presents an example of an event log $\mathcal{L}$ that illustrates the data used for RCT prediction. Each row in the table corresponds to an event e.

Given the prefix of a process instance, which includes all the events that have occurred up to a certain point in time, and a prediction model, which is trained on historical process data, the goal is to estimate the remaining time until the completion of the process instance.

For example for the trace $\sigma_1 = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ associated with case ID number 1 in Table 4.3, some prefixes that can be extracted from σ_1 are $hd^2(\sigma_1) = \langle e_1, e_2 \rangle$ and

Table 4.3 – Example of an event log.

Event ID	Case ID	Activity	Timestamp
e_1	1	a	2022-01-30 08:20:07
e_2	1	b	2022-02-08 08:58:46
e_3	1	b	2022-02-08 09:59:05
e_4	1	c	2022-02-11 17:27:35
e_5	1	d	2022-02-15 09:45:20
e_6	2	a	2022-01-30 08:38:54
e_7	2	b	2022-02-07 09:30:07
e_8	2	c	2022-02-10 15:12:25
e_9	2	a	2022-02-10 17:15:08
e_{10}	2	d	2022-02-12 16:31:20

$hd^3(\sigma_1) = \langle e_1, e_2, e_3 \rangle$.

The RCT prediction takes as input the prefix of a running case and outputs the time remaining until the end of that case. An RCT prediction model is a function Ω such that:

$$\Omega(hd^k(\sigma)) = \#_t(e_{|\sigma|}) - \#_t(e_k), \quad k \in 1..(|\sigma| - 1).$$

There are two main approaches to predicting the RCT of a running process instance. The first approach, namely the recursive approach, involves recursively predicting the next activity and its timestamp until the end of the process, and the RCT is obtained with a function Ω' by summing the time between all intermediate activities (see Equation (4.15)). The second approach, namely the direct approach, involves directly predicting the RCT from the prefix (see Equation (4.16)). This latter approach trains a function Ω from the prefix space to $\mathbb{R}$ using historical data.

$$\Delta\hat{t} = \sum_{j=k}^{|\sigma_{max}|-1} \Omega'(hd^j(\sigma)) \tag{4.15}$$

$$\Delta\hat{t} = \Omega(hd^k(\sigma)) \tag{4.16}$$

A systematic review of state-of-the-art models is presented in Section 1.1.2. For the recursive approach, the model Ω' is trained to predict the next event and timestamp. Predicting the next activity and timestamp is a sub-task of the RCT prediction problem,

making it simpler and generally resulting in lower prediction errors. However, the recursive method requires many intermediate prediction steps to obtain the final result. As a result, the prediction is computationally expensive, and the prediction errors may accumulate and become significant for the final outcome, especially for long processes with many events and a lot of duplicated activities [102]. On the other hand, the direct approach builds its own model, resulting in faster and more direct predictions. However, for complex and long processes, this approach may be less accurate.

In our case study, we use a neural network as the function Ω. It is important to note that the prefix is represented as a subsequence of events, and each event is made up of an activity label, a timestamp, and other attributes. To utilize a neural network for processing prefixes, we must convert the prefix into numerical tensors, such as vectors or matrices, which can be used as inputs for the network. This conversion process involves two tasks: event encoding and sequence encoding. Event encoding is the process of transforming each event into a vector of features, such as the timestamp and the operation performed. Meanwhile, sequence encoding involves combining the event vectors into a matrix that represents the prefix as a sequence of events.

Event encoding

As defined in Definition 1, an event is a tuple of attributes with different types, and most of them are in text. Hence, event encoding involves extracting relevant attributes for the prediction problem and then computing their numerical representation. Activity and timestamp are essential attributes for predicting the RCT since they allow us to model the past behavior of the ongoing process instance over time. Other attributes, such as cost or resource, can be included to complement the prefix. The activity attribute is a categorical variable often represented as text or label. The set of all possible values or modalities is finite and predefined, making it countable. Common encoding techniques for categorical variables include *One-Hot Encoding*, *Integer Encoding*, and *Learned Embedding*.

— *One-Hot Encoding*, also known as dummy coding, creates a binary variable for each category of a nominal variable, i.e., a variable in which there is no order between categories. The variable takes a value of 1 if the category is present and 0 otherwise. This technique is popular for nominal variables.

— *Integer Encoding* is suitable for ordinal variables, where each category is assigned an integer value. This method is straightforward and computationally efficient, particularly for variables with a high number of categories. Nevertheless, integer

encoding can be arbitrary and can create unintended relationships among categories. Therefore, one must be cautious when choosing integer values.

— *Learned Embedding* involves the creation of low-dimensional vector embeddings for each category of variables. Typically, this is accomplished using a machine learning model, such as a neural network. A relevant loss function is established, and the embedding vectors are updated iteratively during training through backpropagation. While commonly used in natural language processing, this technique offers the advantage of capturing complex and implicit relationships among categories while reducing the dimensionality of the encoding vectors. However, it does require more data and resources for the training process, and the resulting embedding vectors may not be easily interpreted and can be sensitive to initialization and training parameters.

In our study, since the order or hierarchy between categories is not explicitly known, we consider the activity as a nominal variable. Then, the one-hot encoding technique is used to encode the activity.

Regarding the timestamp, we use the encoding technique presented in [102]. Especially four time-based features are computed for each event. These are the time from the previous event in the same case, the time from the start of its case, the time within the day, and the day within the week. In the end, each event is represented by a vector which is a concatenation of activity encoding and a vector of time-based features. The illustration shown in Figure 4.6 provides an example of how events can be encoded for the three events found in the prefix $hd^3(\sigma_1) = \langle e_1, e_2, e_3 \rangle$. In this particular example, it is assumed that there are four different activity labels, resulting in an activity encoding vector with a length of four. Furthermore, the time between events is computed in seconds, and the day of the week is encoded from 0 to 6, with Monday being assigned the value of 0 and Sunday the value of 6.

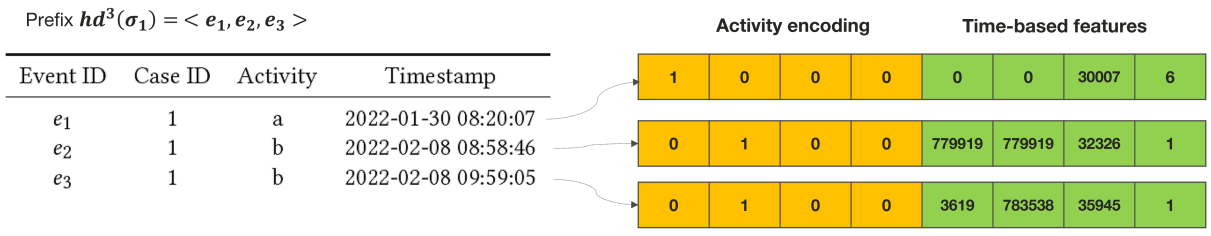


Figure 4.6 – Example of event encoding.

Sequence encoding

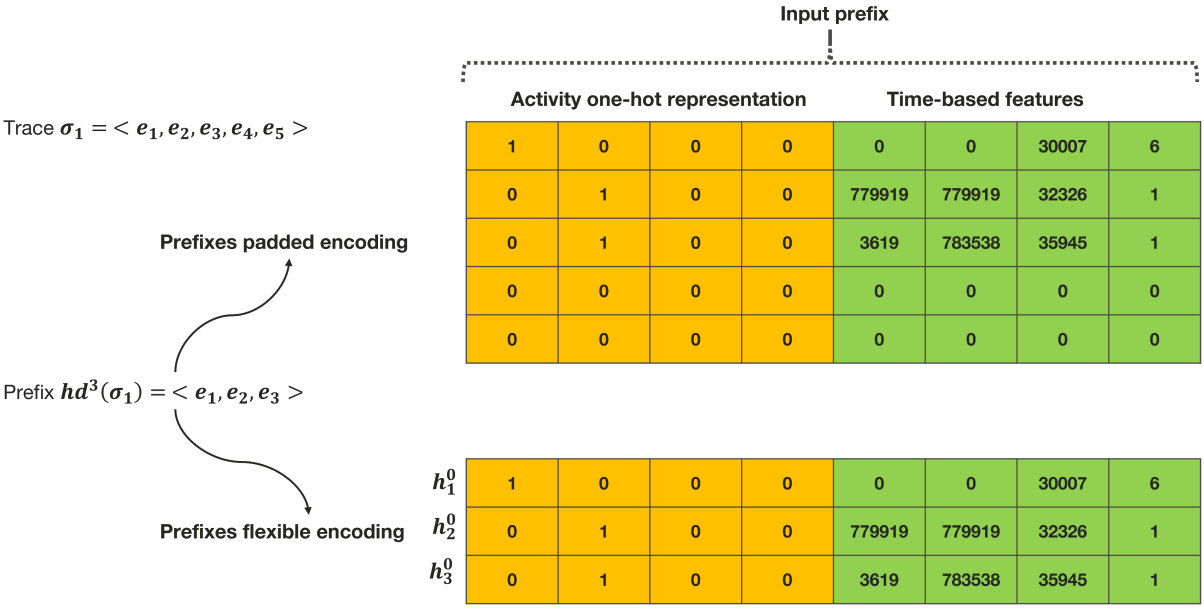
In the task of prediction of the RCT, inputs are prefixes with variable lengths. One solution is to train multiple predictors based on the prefix length. This approach involves training several predictors, each with a distinct prefix length of the input sequence. Each predictor only needs to be trained on a subset of the data, thus making the training process faster. However, the downside of this approach is that it requires numerous models and training sessions, which can be challenging when dealing with long traces. Additionally, the performance may deteriorate as the sequence length increases, as each predictor only has access to limited information.

The other solution is to combine all the prefix lengths and train a single model that takes the entire input sequence as input and predicts the RCT. By considering the entire input sequence, the model can capture complex dependencies between the input sequence and the remaining cycle time, resulting in more consistent and accurate predictions. In our case study, we utilized the latter approach. In this case, the model takes inputs with varying prefix lengths. After the event encoding, the prefix is represented by a sequence of vectors. Subsequently, a sequence encoding step is required to obtain relevant inputs for the model. This study utilizes two sequence encoding techniques:

— **Prefixes padded** [80]: Neural networks are, in general, designed to process data with a fixed dimensionality or shape, meaning that the input tensor must have consistent dimensions across all input examples. To accommodate this requirement, this encoding technique consists of setting a fixed maximum length for the input prefix and padding shorter prefixes with zeros. In this study, we take the length of the longest trace as the maximum prefix length. As illustrated in Figure 4.7 for our example trace σ_1 and prefix $hd^3(\sigma_1) = \langle e_1, e_2, e_3 \rangle$, the top table provides an example of prefix padding where the maximum length is assumed to be 5, the length of the trace. The prefix of length three is then padded with two rows of zeros.

— **Prefixes flexible**: This method encodes all events in the prefix without using padding. As a result, the length of the feature matrix matches the prefix length. It is specifically designed for models that can process inputs of varying sizes. An example of the flexible prefix technique is shown in the bottom table of Figure 4.7. We observe that events e_2 and e_3 correspond to the same activity, b, resulting in similar activity representations. However, their time-based features differ as they occur at different times. The number of lines in the matrix is determined by the

number of events in the prefix.



1	0	0	0	0	0	30007	6
0	1	0	0	779919	779919	32326	1
0	1	0	0	3619	783538	35945	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	0	0	0	0	0	30007	6
0	1	0	0	779919	779919	32326	1
0	1	0	0	3619	783538	35945	1

Figure 4.7 – Example of prefix encoding methods used in this study.

4.2.2 Neural network architectures for the RCT prediction

Graph Neural Networks

Graphs, which are defined in Definition 6 in Section 3.1, are essential tools for displaying relationships between entities and representing data. Their applications span various fields, including social science, linguistics, chemistry, biology, physics, and more. Social scientists use graphs to illustrate connections between individuals, while chemical compounds are often depicted using graphs with atoms as nodes and chemical bonds as edges. In linguistics, graphs are used to capture the syntax and structure of sentences. In this study, we utilize the power of graphs by applying GNNs to the assembly process. In our model, nodes represent different operations, and edges depict transitions between operations. GNNs belong to a class of neural networks designed to process graph data. This section introduces the fundamental concepts of GNNs and some common architectures.

In simple terms, GNNs can be viewed as a process of representation leaning on graph [62]. These representations are able to capture the connectivity and structure of the graph and can contain valuable information for tasks such as node classification or link prediction. There are two primary perspectives when it comes to GNNs: node embedding and

graph embedding. Node embedding is the process of representing each node in a graph as a low-dimensional vector. The aim of this process is to capture both the structural and semantic information of the node and its surrounding neighborhood. The goal is for similar nodes to be mapped to nearby points in the embedding space while dissimilar nodes are mapped to distant points. Graph embedding, on the other hand, involves mapping the entire graph into a low-dimensional vector representation. The aim of graph embedding is to capture the overall structure and meaning of the graph. Node embedding is useful for tasks like node classification, link prediction, and node clustering, while graph embedding is useful for tasks like graph classification, graph clustering, and graph visualization. The concise overview of graph representation learning is presented in Figure 4.8. In this process, GNNs take on the role of the function $f(.)$, which takes the graph G^0 as input. Typically, GNNs process both the graph structure, represented by the adjacency matrix $A \in \mathbb{R}^{N \times N}$ and the node feature matrix $H^0 \in \mathbb{R}^{N \times d_0}$, respectively. d_0 is the dimension of the node feature vector. The output graph G^L has the same structure as G^0, but the node features (or hidden states) are transformed to the latent space $\mathbb{R}^{d_L}$. The graph embedding vector is computed from the node embeddings through the pooling process.

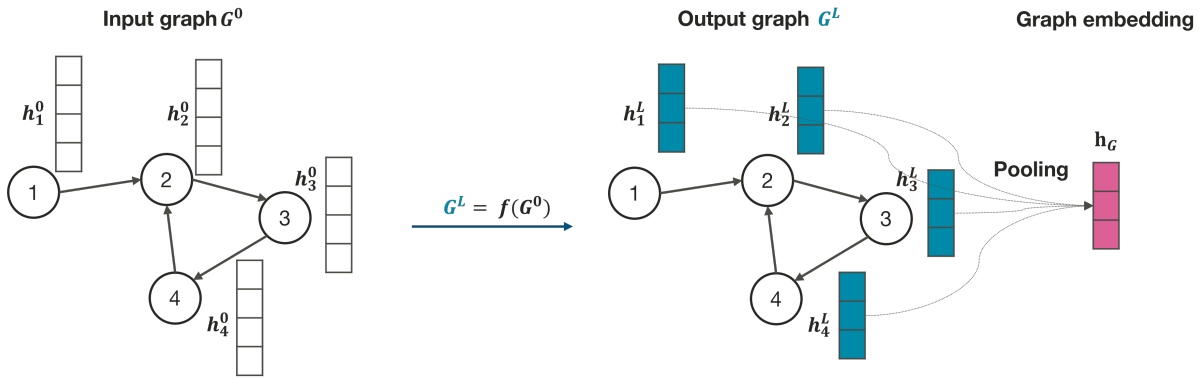


Figure 4.8 – Overview of graph representation learning.

Message Passing: This section presents the Message Passing framework, a general concept on which most GNN architectures are based [32]. It presents the process by which information is propagated through a graph. In this framework, each node in the graph aggregates information from its neighboring nodes and updates its own state. The process of message passing involves three phases:

1. **Initialization**
   Each node v_i in the graph is assigned an initial hidden state h_i^0 that describes its attributes or properties. In case no information is available, a random initialization can be performed.

2. **Message aggregation**
   In this phase, each node v_i with its hidden state aggregates information from its neighboring nodes $\mathcal{N}(v_i)$ (Equations (4.17), (4.18)).

$$m_i^{l+1} = \Psi(\{(h_j^l, e_{ji}) | v_j \in \mathcal{N}(v_i)\}) \tag{4.17}$$
$$= \Gamma(\sum_{v_j \in \mathcal{N}(v_i)} \psi(h_j^l, e_{ji})) \tag{4.18}$$

   where l is used to identify the iteration (layer), m_i^{l+1} is the message associated to node v_i at iteration $l+1$. The function Ψ accepts as input the hidden state h_j^l of all neighborhood node $\mathcal{N}(v_i)$ of the node v_i as well as the corresponding edge features e_{ji}. This function is invariant to permutation and produces an output that remains constant even if the order of input elements is changed. The expression for a permutation invariant function can be found in Equation (4.18), which follows Theorem 4.1 of [115]. The use of this type of function solves the issue of not having a consistent way to order nodes. Common examples of such functions include the sum and the mean.

3. **Update**
   This phase involves updating the state of each node based on the aggregated messages and its own previous state.

$$h_i^{l+1} = U(h_i^l, m_i^{l+1}) \tag{4.19}$$

The process of message aggregation and update is repeated for a fixed number of iterations L or until a convergence criterion is met. Once completed, a new hidden state h_i^L is obtained for each node v_i.

The selection of functions Ψ and U results in variations architecture of GNNs. Here are some specific examples of these architectures:

∗ **Graph Convolution Network (GCN)** [49]

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l\Theta) \tag{4.20}$$

where A is the binary adjacency matrix and $\tilde{A} = A + I$, I being the identity matrix. $\tilde{D}$ is the degree matrix associated with $\tilde{A}$, i.e., $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $H \in \mathbb{R}^{N \times F}$ is the matrix of node features. Θ is a trainable weight matrix, and $\sigma(.)$ denotes the activation function. The GCN model is a simplified version of the Cheby filter [21] with order $K = 1$ and an approximation of $\lambda_{max} = 2$. Equations (4.18) and (4.19) are implicitly included in Equation (4.20). The $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ matrix is non-zero only for nodes that are directly connected. Thus, the GCN model updates node features by considering directly connected neighbors [62].

* **GraphSAGE** [35]

$$\mathcal{N}_S(v_i) = SAMPLE(\mathcal{N}(v_i), S) \tag{4.21}$$

$$m_i^{l+1} = AGGREGATE(\{h_j^l | v_j \in \mathcal{N}_S(v_i)\}) \tag{4.22}$$

$$h_i^{l+1} = \sigma([h_i^l, m_i^{l+1}]\Theta) \tag{4.23}$$

The GraphSAGE model starts with a sampling process which samples randomly a fixed number S of nodes $\mathcal{N}_S(v_i)$ from the local neighborhood $\mathcal{N}(v_i)$ of a given node v_i (4.21). Then the information from these selected nodes is aggregated by the $AGGREGATE()$ function (4.22). Various aggregators have been introduced in [35] such as the mean aggregator, LSTM model or the pooling aggregator. Finally, the aggregated message and the old hidden state are combined to generate the new features as shown in Equation (4.23). In this phase, a fully connected layer is used as the update function $U()$, and $[.,.]$ is the concatenation operation. Θ is a trainable weight matrix, and $\sigma(.)$ denotes the activation function.

* **Graph Attention Network (GAT)** [108, 109] The GAT model and GCN both aggregate information from neighboring nodes to generate new features for each node. However, the GAT model differentiates the importance of neighbors during the aggregation process, while the GCN only considers the graph structure. To achieve this, the GAT model attends to all neighbors of a node and generates an importance score for each neighbor. These scores are used as linear coefficients during the aggregation process to generate new features for the node.
For each node v_i, the importance score of node $v_j \in \mathcal{N}(v_i) \cup \{v_i\}$ is calculated by passing a feedforward layer:

$$\alpha_{ij} = \frac{\exp\left(LeakyReLU(a^T[\Theta h_i, \Theta h_j])\right)}{\sum_{v_k \in \mathcal{N}(v_i) \cup \{v_i\}} \exp\left(LeakyReLU(a^T[\Theta h_i, \Theta h_k])\right)} \tag{4.24}$$

where Θ is the shared weight matrix, a is a parametrized vector, $[.,.]$ denotes the concatenation operation and $LeakyReLU$ is the nonlinear activation function.

Graph pooling: Graph pooling is a technique utilized for graph embedding. It involves computing the representation vector of the entire graph $\mathcal{G} = (V, E)$ by means of a readout function R. If there is no discernible order among the nodes of the graph, then the function R must be permutation invariant.

$$h_G = R(\{h_i^L | v_i \in V\}) \tag{4.25}$$

where h_G is the graph embedding.

There are several methods that are classified as either flat graph pooling or hierarchical graph pooling, according to [62]. Flat graph pooling computes the graph representation directly from node embeddings in a single step. Examples of this approach include max/min pooling, sum pooling, and average pooling. On the other hand, hierarchical graph pooling coarsens the graph step by step from the original graph until the graph embedding is achieved. This approach often involves multiple pooling layers. An example of this approach is sub-sampling, which involves selecting the most important nodes as the nodes for the coarsened graph.

Gated Graph Neural Networks

Like most GNNs, the GGNN's core operation is message passing, which consists of two phases: message aggregation and update (as shown in Figure 4.9).

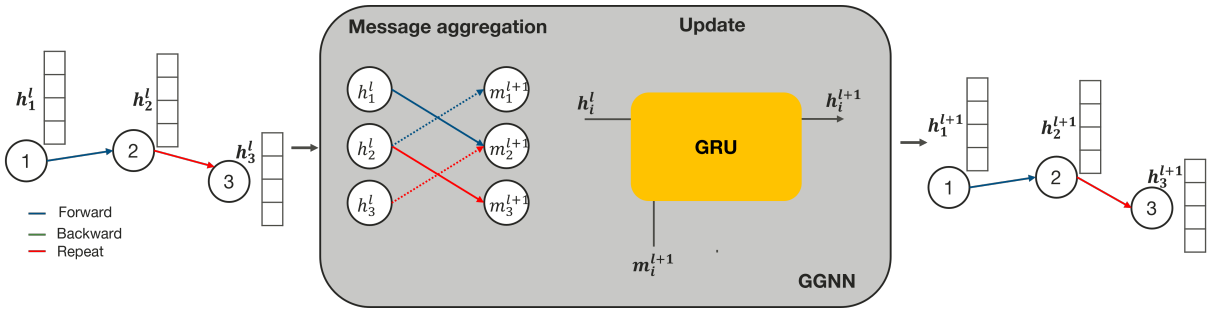


Figure 4.9 – Architecture of Gated Graph Neural Networks.

The input graph whose node features are initialized by the prefix encoding method presented in Figure 4.7 first goes through a message aggregation phase in which each node

v_i aggregates information from its neighboring nodes $\mathcal{N}(v_i)$.

$$m_i^{l+1} = \sum_{v_j \in \mathcal{N}(v_i)} e_{ji} \times \Theta \times h_j^l \tag{4.26}$$

Then, the hidden state h_i of node v_i is updated. In the case of GGNN, the function U is a Gated Recurrent Unit (GRU) cell [19].

$$h_i^{l+1} = GRU(m_i^{l+1}, h_i^l) \tag{4.27}$$

GGRU and LSTM are two popular types of recurrent neural networks. In comparison to the LSTM architecture discussed in 4.1.2, the GRU has a simpler structure with only two gates: the update gate and the reset gate. The message-passing process is repeated L times so that the information from one node is propagated to all other nodes in the graph. The hyper-parameter L is similar to the number of convolution layers in the CNN. The GGNN model is used in [119] to predict the next activity. In this study, we applied this model to solve the RCT prediction problem. The input prefix is considered as a graph whose nodes are events presented in the prefix, and the edges represent the precedence relations between the events. Furthermore, edges are distinguished into three types:

* *Forward*: edges from one to a new activity within a case, e.g., $\langle e_1, e_2 \rangle$ in case 1 of Table 4.3. This edge is shown in blue in Figure 4.9.
* *Backward*: edges from one to an activity that has been performed in a case, e.g., $\langle e_8, e_9 \rangle$ in case 2.
* *Repeat*: edges between two events associated with the same activity, e.g., $\langle e_2, e_3 \rangle$ in case 1. This edge is the red one in Figure 4.9.

At the output of the GGNN block, we obtain the node embeddings. They are then passed through a readout function to get the graph embedding. We use the *global mean pooling* function to do this. Finally, the graph embedding goes through fully connected layers with the activation function $ReLU$ to make a prediction.

RCT prediction model architectures

In this case study, two neural network architectures have been employed to solve the task at hand. The first architecture is LSTM, which was found to have the best performance in the recent benchmark [111]. The second architecture is the Gated Graph Neural Network (GGNN) which is proposed by Li et al. [58]. This model exhibits good

performance when compared to classical methods for predicting the next activity, as presented in [119]. Our study aims to compare these two models and determine whether the GGNN can outperform the current state-of-the-art model, LSTM.

Figure 4.10 presents the architecture of the two RCT prediction models. The LSTM model is shown in the top part of the figure, and it takes as input a sequence of vector features, with each vector corresponding to an event in the prefix. The hidden state from the output of the LSTM model is passed through fully connected layers to make the prediction. On the other hand, the GGNN model, shown in the bottom part of the figure, takes the input in the form of a graph, where each node represents an event. The graph is then passed through a certain number of layers of GGNN, resulting in a new graph with updated hidden states. The output graph is then subject to a global mean pooling operation to obtain the graph embedding, which is subsequently passed through fully connected layers to make the prediction.

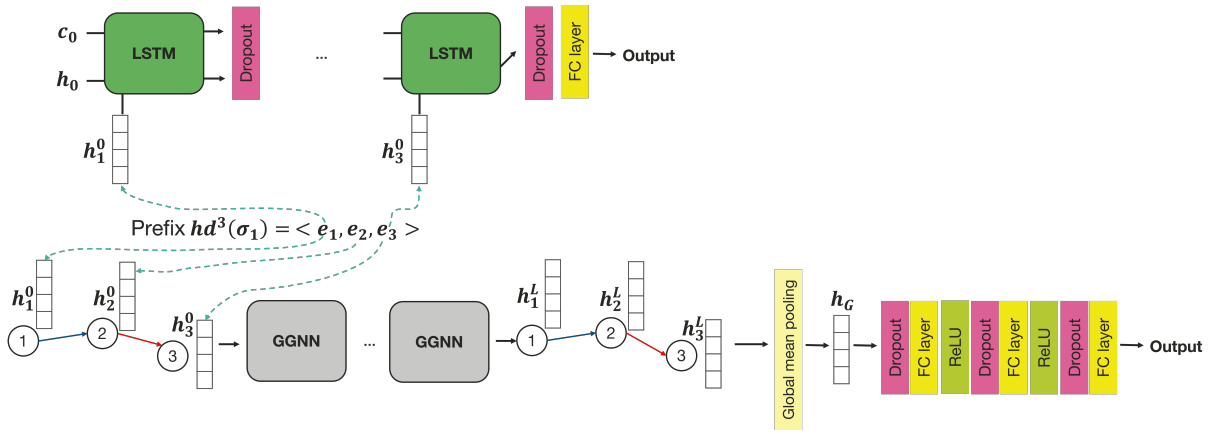


Figure 4.10 – Architecture of RCT prediction model.

Regarding the prefix encoding, the prefixes padded encoding is used in the LSTM model. The GGNN model uses the prefixes flexible encoding. The three first time-based features are computed in seconds. They are normalized using *min-max* normalization during the training phase.

4.2.3 Experimentation

Datasets

In this study, we use three event log datasets. The first two datasets, **Helpdesk** and **BPIC20**, are publicly available datasets commonly used in the field of process mining.

The third dataset is from the real manufacturing process of Vitesco Technologies.

— **Helpdesk** (Figure 4.11) contains event logs from a ticketing management process of the help desk of an Italian software company. The logs are available at [76].

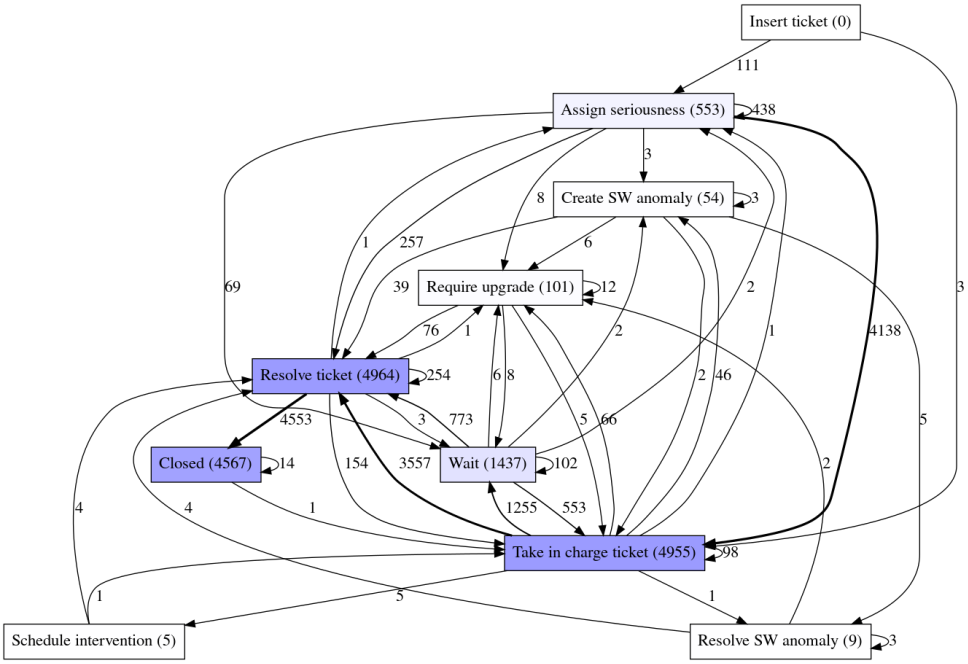


Figure 4.11 – Directly follows graph generated for the Helpdesk dataset.

— **BPIC20** (Figure 4.12) contains events pertaining to two years of travel expense claims [25].

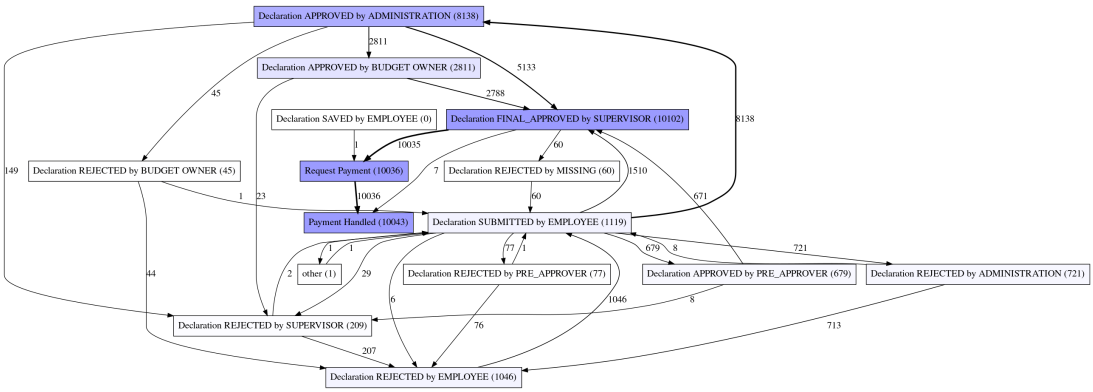


Figure 4.12 – Directly follows graph generated for the BPIC20 dataset.

— **EMS01** dataset (Figure 4.13) represents a real manufacturing process for assembling electronic boards of the family EMS01 in Vitesco Technologies.

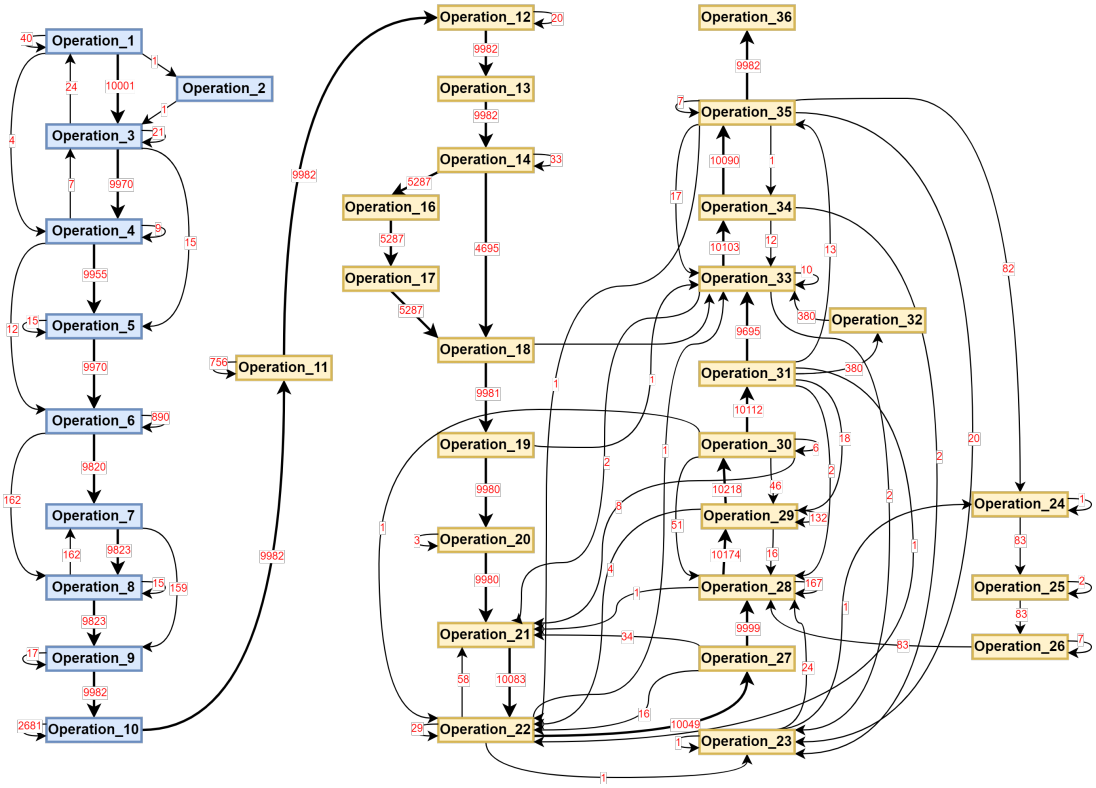


Figure 4.13 – Directly follows graph generated for the EMS01 logs. The FE phase is in blue, and the BE is in yellow. The label on each edge represents the number of products that have taken the corresponding transition. Operation IDs in nodes are anonymized for confidentiality reasons.

All of the datasets were preprocessed, and only complete traces were extracted. Table 4.4 displays the statistics of the processed event logs. The statistics reveal that the EMS01 dataset has the highest number of activities, the highest average events per case, as well as the highest number of variants. These findings suggest that the manufacturing process for EMS01 products is more intricate and detailed than the other two datasets. In contrast, the Helpdesk dataset has the lowest average number of events per case, which implies a relatively simple process. The BPIC20 dataset falls somewhere in between. Overall, the statistics suggest that the manufacturing process of EMS01 products is more complex than both the IT support ticketing process and the loan application process.

Each dataset is then split into two sets according to the start time of each case. The first 2/3 of the traces are used to train the model, and the last 1/3 of the traces constitute the test set. Regarding the input for the RCT prediction problem, only prefixes with a minimum length of two events are considered because the GGNN model requires a graph

input that contains at least two nodes.

Table 4.4 – Statistics of event logs used in the study.

Event log	Nb. activities	Avg. case length	Max. case length	Avg. case duration (days)	Max. case duration (days)	Min. case duration (days)	Variants
Helpdesk	10	4.66	15	40.85	59.99	30.64	207
BPIC20	15	5.49	24	11.62	368.19	1.06	64
EMS3141	35	28.76	44	7.01	80.87	0.69	296

Experimental setup

We conducted an experiment using Pytorch that involved two tasks. The first task was to compare the performance of the direct and recursive approaches in predicting the RCT. The second task was to evaluate the performance of the GGNN model against the LSTM baseline model. For the training step, we set aside 20% of the training set for validation and implemented dropout layers and early stopping techniques to prevent overfitting. We fine-tuned the dropout percentage as well as other hyperparameters, including batch size, learning rate, number of layers, and the dimension of hidden layers, using Bayesian optimization. Once we found the optimized hyperparameters, we retrained the model. Due to the randomness problem in neural networks arising from initialization, dropout process, and stochastic gradient descent, we repeated the training and reference process five times and computed the average and standard deviation for prediction outputs to ensure consistent results. We used the MAE metric to assess the model performance. For more information on hyperparameter tuning and the training process, please visit `https://github.com/duongtoan261196/RemainingCycleTimePrediction`.

4.2.4 Results and analysis

Direct vs recursive approaches for the RCT prediction

This section presents a comparison of the performance of recursive and direct approaches in predicting RCTs. To do this, we utilize the LSTM-based model shown in Figure 4.10 (top). The recursive approach was originally developed in [102], and we reused it for this study, as well as developing a direct approach using LSTM.

Table 4.5 presents the average prediction errors over all prefix lengths of the LSTM-recursive and LSTM-direct models on the two public datasets **Helpdesk** and **BPIC20**. The results demonstrate that LSTM-direct provides more accurate predictions for both datasets. The last column of the table reveals the percentage of error reduction achieved by using the direct approach rather than the recursive approach. Furthermore, our experiment shows that the recursive approach takes longer to predict than the direct approach because it must predict each subsequent event until reaching a stopping criterion, while the direct approach predicts everything at once. As a result, it was decided to only test the direct approach in future experiments.

Table 4.5 – Average MAE (days) over all prefix lengths of the prediction by the LSTM-recursive and LSTM-direct model.

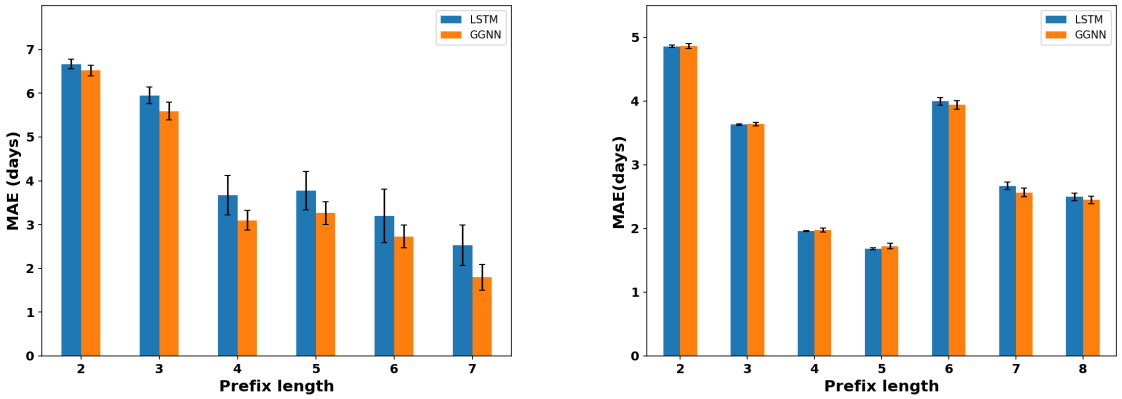
Dataset	LSTM recursive	LSTM-direct	Gain in %
Helpdesk	5.783	**5.666**	2.02
BPIC20	3.431	**3.269**	4.72

GGNN vs LSTM for the RCT prediction

The main contribution of this study is the use of graph neural networks for RCT prediction. This section compares the performance of the GGNN presented in Figure 4.10 (bottom) and the LSTM model in Figure 4.10 (top). Table 4.6 shows that the GGNN model outperforms the LSTM for the **Helpdesk** dataset. Concerning the **BPIC20** dataset, the two models perform nearly the same, with only 0.18% difference in the MAE. Figure 4.14 shows the prediction error for each prefix length. The figure does not show prefix lengths for which the number of samples is very small. It can be seen from Figure 4.14(a) that the prediction errors reduce when the prefix length increases. This result is reasonable because a long prefix gives more information to predict the RCT. However, this seems not to be the case for the **BPIC20** dataset (Figure 4.14(b)). Indeed, the error goes up for prefixes of length 6. Actually, in the **BPIC20** dataset, the number of samples for prefixes from the length of 6 is much lower than those for shorter prefixes. Hence, the model is trained less for these prefixes. Overall, these results indicate that the GGNN model may be applied to the RCT prediction problem to achieve better prediction.

Table 4.6 – Average MAE (days) over all prefix lengths of the prediction by the LSTM and GGNN model.

Dataset	LSTM	GGNN	Gain in %
Helpdesk	5.666	**5.345**	5.67
BPIC20	**3.269**	3.275	-0.18



(a) (b)

Figure 4.14 – MAE values (days) of different prefix lengths for two public event logs: **Helpdesk** (**a**), **BPIC20** (**b**). The error bars at the top of each bar represent the standard deviations of the metric.

Application of RCT prediction in a real manufacturing process

After comparing the two models on public datasets, we apply them to a real-life process of the automotive company Vitesco Technologies. The process is composed of different steps to assemble electronic boards from printed circuit boards (PCBs). Figure 4.13 presents the process related to the product family EMS01 by a directly-follows graph. The assembly process consists of two consecutive phases: Front-end (FE) in blue and Back-end (BE) in yellow. Compared to the two public datasets, the EMS01 dataset is more complex in terms of the number of activities, case length, and the number of variants (see Table 4.4). For the RCT prediction problem, we only consider products that have completed the FE phase and are currently in the BE phase. This is products that have completed the FE phase are stocked in an intermediate area while waiting to be processed in the BE phase, and the waiting time varies significantly depending on the production status and several other factors. This uncertainty can therefore degrade the performance

of predictive models.

Figure 4.15 presents the prediction error for each prefix length between the LSTM and the GGNN model. The results clearly show that the GGNN model outperforms the LSTM. The average error over all prefix lengths is 1.267 hours for LSTM and 0.339 hours for GGNN, respectively. That is 73.2% reduction on the prediction error of GGNN versus LSTM. This outperformance is clearly superior compared to the results obtained with **Helpdesk** and **BPIC20**. This also shows that the GGNN model works better with more complex processes.

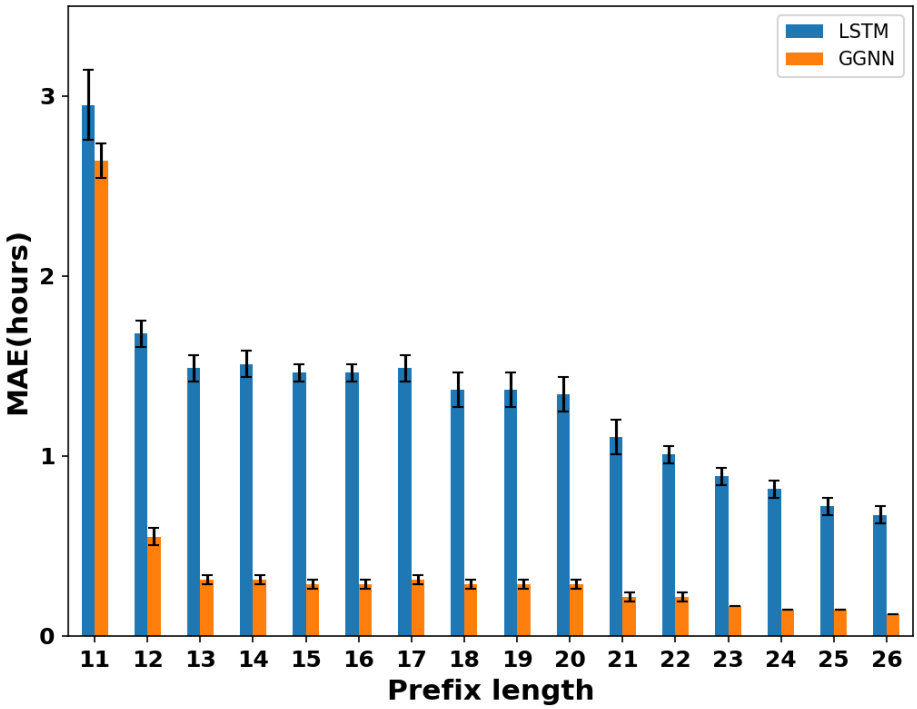


Figure 4.15 – MAE values (hours) of different prefix lengths for the EMS01 dataset. The error bars at the top of each bar represent the standard deviations of the metric.

4.3 Conclusion

In this chapter, we presented two prediction problems applied to the real manufacturing process from Vitesco Technologies: WIP prediction and RCT prediction. The objective of the WIP prediction is to help manufacturers better manage their production process

by predicting the amount of work in process at a given time. The RCT prediction aims to predict the remaining time required to complete a product, which can aid in planning and scheduling the production process.

Regarding the WIP prediction, our experiments show that linear regression performs the best with respect to prediction error, computational time, and interpretability. However, the model still struggles with trend reversal in WIP variations and multi-step predictions (t+2, t+3, etc.). Furthermore, historical WIP values alone are insufficient for accurate prediction. The complexity of manufacturing processes, which involve multiple stages and sources of variability, makes accurate prediction challenging, and additional information, such as planning and logistics data, should be incorporated.

For the RCT prediction, our results demonstrate that the GGNN model outperforms the baseline LSTM, particularly for long and complex processes such as the assembly process of our case study. This indicates that graphs can efficiently capture the complex relationships and dependencies between process variables, leading to more accurate predictions in complex processes.

In conclusion, predictive process mining is a promising approach for improving manufacturing process efficiency and productivity. However, accurately predicting process variables is challenging due to their complexity and variability. Future research should focus on developing more accurate and robust prediction models that can incorporate a broader range of data sources and handle the complexities of manufacturing processes.

Conclusion

In this thesis, we present an in-depth exploration of process mining techniques and Graph Neural Networks (GNNs) in the manufacturing industry. Initially, we conduct a comprehensive overview of state of the art in process mining, highlighting its importance and relevance in today's business world (Chapter 1). The literature review reveals that process mining has been widely used in various domains and industries, including healthcare, finance, and IT. We discuss various process mining techniques for process discovery, conformance checking, and process improvement, ranging from process model-based to machine learning-based methods. Moreover, we highlight the potential of process mining, combined with other related technologies, such as data mining and machine learning, to achieve better outcomes. Among various machine learning architectures, GNNs appear to be promising due to their capacity to process graph-structured data, such as process models generated from process mining techniques. Additionally, the underdeveloped status of process mining and GNNs in the manufacturing domain motivates us to apply GNNs to a real manufacturing process in the context of process mining.

Next, in Chapter 2, we introduce the general context of the thesis, which is the automotive industry and, specifically, the manufacturing process. We also introduce the case study of Vitesco Technologies, along with a detailed description of the Printed Circuit Board (PCB) assembly process and related datasets that are used for the research. The production process and the vast amount of event log data motivate the study of process mining and related methods as the core of the research in the thesis.

The main contribution of this thesis consists of two primary axes of process mining: Descriptive Process Mining (DPM) and Predictive Process Mining (PPM), presented in Chapters 3 and 4, respectively. Before that, we introduce fundamental process mining concepts such as event log, event, trace, and prefix. Concerning DPM, we discuss the two primary tasks in the field: process discovery and conformance checking. We have proposed an end-to-end process mining framework that computes the timed process model and compares it with the actual behavior of products to assess their quality and performance. The quality indexes provided by our framework serve as an indicator for manufacturers to evaluate how well their products performed during the production phases and to iden-

tify any potential issues in case of failure. One of the fascinating aspects of our work is that we have integrated domain knowledge from process experts alongside process mining techniques and event log data. This integration enhances our results' accuracy, enables their interpretation, and supports effective decision-making. Moreover, we have proposed a statistical methodology based on non-parametric kernel density estimation for performance analysis between work shifts based on the quality index. Although the analysis may not show significant variability between shifts, it provides relevant information to the quality service and management, allowing for a comprehensive understanding of the production process and identification of potential areas for improvement. Furthermore, the method is generic and can be applied to other manufacturing processes. Finally, we have developed a real-time Work-In-Progress (WIP) calculation and tracking program that facilitates production monitoring. In particular, the operational teams use it daily as an essential tool in the Vitesco Technologies plant of Foix.

Regarding PPM, we present two prediction problems applied to the PCB assembly process: WIP prediction and Remaining Cycle Time (RCT) prediction. The goal of WIP prediction is to assist manufacturers in better managing their production process by predicting the amount of WIP at a given time. On the other hand, the RCT prediction aims to predict the remaining time required to complete a product, which can be useful in planning and scheduling the production process. The experiments conducted on WIP prediction reveal that linear regression yields the best results in terms of prediction error, computational time, and interpretability. However, the model still struggles with trend reversals in WIP variations and multi-step predictions. These issues arise due to a lack of information. Past values of WIP alone are insufficient to accurately predict future values, particularly over long periods. The results suggest incorporating additional information, such as production planning and logistics data. Regarding RCT prediction, the results indicate that the Gated Graph Neural Network (GGNN) outperforms the baseline LSTM for long and complex processes. This is an exciting finding as it highlights the potential for using GNNs in process mining applications. In particular, the use of GNNs can improve the efficiency of handling graph-structured data, which is often encountered in manufacturing and other industrial processes. By leveraging the inherent structure of process models, GNNs can effectively capture the complex dependencies between different activities.

Despite several promising results achieved during this thesis, there are still various aspects that require further improvement and exploration in future works. For example,

in the evaluation of product quality using event logs (Section 3.3.1), a parameter learning process (λ, γ_k) could be conducted based on after-sales service data related to products returned by customers. Additionally, an additional dimension that affects product quality, which is the time slot during production, could also be considered.

Regarding performance analysis between work shifts (Section 3.3.2), other dissimilarity measures between distributions, such as the Wasserstein distance [113], could be tested. Furthermore, additional criteria, such as the number of rejected products, failed operations, or the average cycle time, could also be considered. Finally, with regards to the tool for WIP tracking, a unified and centralized application could be developed for all other plants of Vitesco Technologies.

As for the prediction models, the WIP prediction (Section 4.1) results suggest that more data should be collected to accurately infer future values of WIP. Once the prediction is considered good enough, it can be integrated into the WIP tracking application to provide both current and future information about the production process. For the RCT prediction (Section 4.2), we intend to develop and evaluate more GNN variations for the RCT prediction problem. Additionally, we will focus on the task of prefix encoding and event encoding to improve the prediction.

Overall, this thesis has made significant contributions to the field of process mining in the manufacturing industry. The case study demonstrates the practical application of process mining techniques and GNNs in a real-world scenario, and the results obtained have offered promising prospects for the future utilization of these technologies in the manufacturing industry. The descriptive process mining presented in this thesis has provided a basis for thoroughly understanding process behavior, while the predictive process mining techniques show promising results in predicting future process behavior. However, there are still various challenges and open problems in process mining and GNNs, including scalability, interpretability, and generalization, which require further research and development. Moving forward, future research could focus on the integration of GNNs and process mining techniques to develop more accurate and robust predictive models. Additionally, the development of more interpretable and scalable models for process mining is essential to enhance the transparency of these models and make them more accessible to a broader range of users in the manufacturing industry. Another promising direction for future research is to investigate the potential of incorporating other machine learning techniques, such as reinforcement learning and transformer, to further enhance the accuracy and efficiency of process mining. Finally, it is important to continue exploring

the applications of process mining in different industries and contexts to fully understand its potential benefits and limitations. With continued research and development, process mining has the potential to transform the way the manufacturing industry approaches process improvement and optimization.

BIBLIOGRAPHY

[1] W. van der Aalst, T. Weijters, and L. Maruster, « Workflow mining: discovering process models from event logs », *in*: *IEEE Transactions on Knowledge and Data Engineering* 16.*9* (2004), pp. 1128–1142, DOI: 10.1109/TKDE.2004.47.

[2] Wil van der Aalst, « Process Mining: Overview and Opportunities », *in*: *ACM Trans. Manage. Inf. Syst.* 3.*2* (July 2012), ISSN: 2158-656X, DOI: 10.1145/2229156.2229157, URL: https://doi.org/10.1145/2229156.2229157.

[3] Wil Aalst et al., « Business process mining: an industrial application. Inf. Syst. 32(5), 713-732 », *in*: *Information Systems* 32 (July 2007), pp. 713–732, DOI: 10.1016/j.is.2006.05.003.

[4] Ahmad Aburomman, Manuel Lama, and Alberto Bugarín, « A Vector-Based Classification Approach for Remaining Time Prediction in Business Processes », *in*: *IEEE Access* 7 (2019), pp. 128198–128212, DOI: 10.1109/ACCESS.2019.2939631.

[5] Arya Adriansyah and Joos CAM Buijs, « Mining process performance from event logs », *in*: *International Conference on Business Process Management*, Springer, 2012, pp. 217–218.

[6] Arya Adriansyah, Boudewijn F Van Dongen, and Nicola Zannone, « Controlling break-the-glass through alignment », *in*: *2013 International Conference on Social Computing*, IEEE, 2013, pp. 606–611.

[7] Torbjörn Åkerstedt and Kenneth P Wright, « Sleep loss and fatigue in shift work and shift work disorder », *in*: *Sleep medicine clinics* 4.*2* (2009), pp. 257–271, DOI: 10.1016/j.jsmc.2009.03.001.

[8] Davide Bacciu et al., « A gentle introduction to deep learning for graphs », *in*: *Neural Networks* 129 (2020), pp. 203–221, ISSN: 0893-6080, DOI: https://doi.org/10.1016/j.neunet.2020.06.006, URL: https://www.sciencedirect.com/science/article/pii/S0893608020302197.

[9] Alessandro Berti, Sebastiaan J van Zelst, and Wil van der Aalst, « Process mining for python (PM4PY): bridging the gap between process-and data science », *in*: *arXiv preprint arXiv:1905.06169* (2019).

[10] Diane B Boivin and Philippe Boudreau, « Impacts of shift work on sleep and circadian rhythms », *in*: *Pathologie Biologie* 62.*5* (2014), pp. 292–301, DOI: 10.1016/j.patbio.2014.08.001.

[11] Alfredo Bolt and Marcos Sepúlveda, « Process Remaining Time Prediction Using Query Catalogs », *in*: *Business Process Management Workshops*, ed. by Niels Lohmann, Minseok Song, and Petia Wohed, Cham: Springer International Publishing, 2014, pp. 54–65, ISBN: 978-3-319-06257-0.

[12] George EP Box et al., *Time series analysis: forecasting and control*, John Wiley & Sons, 2015.

[13] Shaked Brody, Uri Alon, and Eran Yahav, « How attentive are graph attention networks? », *in*: *arXiv preprint arXiv:2105.14491* (2021).

[14] Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi, « Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues », *in*: *Applied Intelligence* 52.*3* (2022), pp. 2763–2774.

[15] Juan Pablo Usuga Cadavid et al., « Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0 », *in*: *Journal of Intelligent Manufacturing* (2020), pp. 1–28, DOI: https://doi.org/10.1007/s10845-019-01531-7.

[16] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas, « Learning Accurate LSTM Models of Business Processes », *in*: *Business Process Management*, ed. by Thomas Hildebrandt et al., Cham: Springer International Publishing, 2019, pp. 286–302, ISBN: 978-3-030-26619-6.

[17] Josep Carmona et al., *Conformance Checking*, Springer, 2018.

[18] Andrea Chiorrini et al., « Exploiting Instance Graphs and Graph Neural Networks for Next Activity Prediction », *in*: *Process Mining Workshops*, ed. by Jorge Munoz-Gama and Xixi Lu, Cham: Springer International Publishing, 2022, pp. 115–126, ISBN: 978-3-030-98581-3.

[19] Kyunghyun Cho et al., « Learning phrase representations using RNN encoder-decoder for statistical machine translation », *in*: *arXiv preprint arXiv:1406.1078* (2014).

[20] Giovanni Costa, « Shift work and health: current problems and preventive actions », *in*: *Safety and health at Work* 1.*2* (2010), pp. 112–123, DOI: 10.5491/SHAW.2010.1.2.112.

[21] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, « Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering », *in*: *CoRR* abs/1606.09375 (2016), arXiv: 1606.09375, URL: http://arxiv.org/abs/1606.09375.

[22] Chiara Di Francescomarino and Chiara Ghidini, « Predictive process monitoring », *in*: *Process Mining Handbook. LNBIP* 448 (2022), pp. 320–346.

[23] Chiara Di Francescomarino et al., « Predictive Process Monitoring Methods: Which One Suits Me Best? », *in*: *Business Process Management*, ed. by Mathias Weske et al., Cham: Springer International Publishing, 2018, pp. 462–479, ISBN: 978-3-319-98648-7.

[24] B. F. van Dongen, R. A. Crooy, and W. M. P. van der Aalst, « Cycle Time Prediction: When Will This Case Finally Be Finished? », *in*: *On the Move to Meaningful Internet Systems: OTM 2008*, ed. by Robert Meersman and Zahir Tari, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 319–336, ISBN: 978-3-540-88871-0.

[25] Boudewijn van Dongen, « BPI Challenge 2020: Domestic Declarations 4TU ResearchData Dataset. », *in*: *BPI Challenge 2020* 1 (2020), DOI: https://doi.org/10.4121/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5.

[26] Sebastian Dunzer et al., « Conformance Checking: A State-of-the-Art Literature Review », *in*: *Proceedings of the 11th International Conference on Subject-Oriented Business Process Management*, S-BPM ONE '19, Seville, Spain: Association for Computing Machinery, 2019, ISBN: 9781450362504, DOI: 10.1145/3329007.3329014, URL: https://doi.org/10.1145/3329007.3329014.

[27] Wenqi Fan et al., « Graph Neural Networks for Social Recommendation », *in*: *The World Wide Web Conference*, WWW '19, San Francisco, CA, USA: Association for

Computing Machinery, 2019, pp. 417–426, ISBN: 9781450366748, DOI: 10.1145/3308558.3313488, URL: https://doi.org/10.1145/3308558.3313488.

[28] Weiguang Fang et al., « Big data driven jobs remaining time prediction in discrete manufacturing system: a deep learning-based approach », *in*: *International Journal of Production Research* 58.*9* (2020), pp. 2751–2766.

[29] Francesco Folino, Massimo Guarascio, and Luigi Pontieri, « Discovering Context-Aware Models for Predicting Business Process Performances », *in*: *On the Move to Meaningful Internet Systems: OTM 2012*, ed. by Robert Meersman et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 287–304, ISBN: 978-3-642-33606-5.

[30] Bent Fuglede and Flemming Topsoe, « Jensen-Shannon divergence and Hilbert space embedding », *in*: *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 2004, p. 31, DOI: 10.1109/ISIT.2004.1365067.

[31] Carl Friedrich Gauss and G. W. Stewart, *Theory of the Combination of Observations Least Subject to Errors, Part One, Part Two, Supplement*, Society for Industrial and Applied Mathematics, 1995, DOI: 10.1137/1.9781611971248, eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611971248, URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611971248.

[32] Justin Gilmer et al., « Neural Message Passing for Quantum Chemistry », *in*: *Proceedings of the 34th International Conference on Machine Learning*, ed. by Doina Precup and Yee Whye Teh, vol. 70, Proceedings of Machine Learning Research, PMLR, June 2017, pp. 1263–1272, URL: https://proceedings.mlr.press/v70/gilmer17a.html.

[33] Marco Gori, Gabriele Monfardini, and Franco Scarselli, « A new model for learning in graph domains », *in*: *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, 2005, 2005, pp. 729–734.

[34] Shengnan Guo et al., « Attention based spatial-temporal graph convolutional networks for traffic flow forecasting », *in*: *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 01, 2019, pp. 922–929.

[35] Will Hamilton, Zhitao Ying, and Jure Leskovec, « Inductive representation learning on large graphs », *in*: *Advances in neural information processing systems* 30 (2017).

[36] Awad S Hanna et al., « Impact of shift work on labor productivity for labor intensive contractor », *in*: *Journal of construction engineering and management* 134.*3* (2008), pp. 197–204, DOI: 10.1061/(ASCE)0733-9364(2008)134:3(197).

[37] Nitin Harane and Sheetal Rathi, « Comprehensive Survey on Deep Learning Approaches in Predictive Business Process Monitoring », *in*: *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough: Latest Trends in AI*, Cham: Springer International Publishing, 2020, pp. 115–128, ISBN: 978-3-030-38445-6, DOI: 10.1007/978-3-030-38445-6_9, URL: https://doi.org/10.1007/978-3-030-38445-6%5C_9.

[38] Sepp Hochreiter and Jürgen Schmidhuber, « Long short-term memory », *in*: *Neural computation* 9.*8* (1997), pp. 1735–1780.

[39] T. B. Hong Tu and M. Song, « Analysis and Prediction Cost of Manufacturing Process Based on Process Mining », *in*: *2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, 2016, pp. 1–5.

[40] Peter TG Hornix, « Performance analysis of business processes through process mining », *in*: *Master's Thesis, Eindhoven University of Technology* (2007).

[41] Ziniu Hu et al., « Gpt-gnn: Generative pre-training of graph neural networks », *in*: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.

[42] Dejun Jiang et al., « Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models », *in*: *Journal of cheminformatics* 13.*1* (2021), pp. 1–23.

[43] Licheng Jiao et al., « Graph representation learning meets computer vision: A survey », *in*: *IEEE Transactions on Artificial Intelligence* (2022).

[44] Bambang Jokonowo et al., « Process mining in supply chains: a systematic literature review », *in*: *International Journal of Electrical and Computer Engineering* 8.*6* (2018), pp. 4626–4636.

[45] Anna Kalenkova et al., « Discovering high-level BPMN process models from event data », *in*: *Business Process Management Journal* (2019).

[46] Seokho Kang, « Product failure prediction with missing data using graph neural networks », *in*: *Neural computing and applications* 33.*12* (2021), pp. 7225–7234.

[47] Göran Kecklund and John Axelsson, « Health consequences of shift work and insufficient sleep », *in*: *Bmj* 355 (2016), DOI: 10.1136/bmj.i5210.

[48] Muhammad Asjad Khan et al., « Memory-Augmented Neural Networks for Predictive Process Analytics », *in*: *CoRR* abs/1802.00938 (2018), arXiv: 1802.00938, URL: http://arxiv.org/abs/1802.00938.

[49] Thomas N Kipf and Max Welling, « Semi-supervised classification with graph convolutional networks », *in*: *arXiv preprint arXiv:1609.02907* (2016).

[50] Thomas N Kipf and Max Welling, « Variational graph auto-encoders », *in*: *arXiv preprint arXiv:1611.07308* (2016).

[51] Edward Elson Kosasih and Alexandra Brintrup, « A machine learning approach for predicting hidden links in supply chain with graph neural networks », *in*: *International Journal of Production Research* 60.*17* (2022), pp. 5380–5393, DOI: 10.1080/00207543.2021.1956697, eprint: https://doi.org/10.1080/00207543.2021.1956697, URL: https://doi.org/10.1080/00207543.2021.1956697.

[52] Eric R. Larson, « 2 - Why Use Plastic? », *in*: *Thermoplastic Material Selection*, ed. by Eric R. Larson, William Andrew Publishing, 2015, pp. 19–56, ISBN: 978-0-323-31299-8, DOI: https://doi.org/10.1016/B978-0-323-31299-8.00002-7, URL: https://www.sciencedirect.com/science/article/pii/B9780323312998000027.

[53] H.C.W. Lau et al., « Development of a process mining system for supporting knowledge discovery in a supply chain network », *in*: *International Journal of Production Economics* 122.*1* (2009), Transport Logistics and Physical Distribution Interlocking of Information Systems for International Supply and Demand Chains Management ICPR19, pp. 176–187, ISSN: 0925-5273, DOI: https://doi.org/10.1016/j.ijpe.2009.05.014, URL: https://www.sciencedirect.com/science/article/pii/S0925527309001741.

[54] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst, « Discovering block-structured process models from event logs containing infrequent behaviour », *in*: *International conference on business process management*, Springer, 2013, pp. 66–78.

[55] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst, « Discovering block-structured process models from incomplete event logs », *in*: *International Conference on Applications and Theory of Petri Nets and Concurrency*, Springer, 2014, pp. 91–110.

[56] Sander JJ Leemans, Dirk Fahland, and Wil MP van der Aalst, « Scalable process discovery with guarantees », *in*: *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2015, pp. 85–101.

[57] Mengzhang Li and Zhanxing Zhu, « Spatial-temporal fusion graph neural networks for traffic flow forecasting », *in*: *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 5, 2021, pp. 4189–4196.

[58] Yujia Li et al., « Gated graph sequence neural networks », *in*: *arXiv preprint arXiv:1511.05493* (2015).

[59] Wenxiong Liao et al., « Multi-level graph neural network for text sentiment analysis », *in*: *Computers & Electrical Engineering* 92 (2021), p. 107096.

[60] Arne Lowden et al., « Eating and shift work—effects on habits, metabolism, and performance », *in*: *Scandinavian journal of work, environment & health* (2010), pp. 150–162, DOI: 10.5271/sjweh.2898.

[61] Tengfei Ma et al., « Online planner selection with graph neural networks and adaptive scheduling », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 04, 2020, pp. 5077–5084.

[62] Yao Ma and Jiliang Tang, *Deep Learning on Graphs*, Cambridge University Press, 2021.

[63] Ronny S Mans, Wil MP Van der Aalst, and Rob JB Vanwersch, *Process mining in healthcare: evaluating and exploiting operational healthcare processes*, Springer, 2015.

[64] Alfonso Eduardo Márquez-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés, « Predictive Monitoring of Business Processes: A Survey », *in*: *IEEE Transactions on Services Computing* 11.*6* (2018), pp. 962–977, DOI: 10.1109/TSC.2017.2772256.

[65] Ana Karla A de Medeiros, Anton JMM Weijters, and Wil MP van der Aalst, « Genetic process mining: an experimental evaluation », *in*: *Data Mining and Knowledge Discovery* 14.*2* (2007), pp. 245–304.

[66] Andreas Metzger et al., « Comparing and Combining Predictive Business Process Monitoring Techniques », *in*: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.*2* (2015), pp. 276–290, DOI: 10.1109/TSMC.2014.2347265.

[67] Alessio Micheli, « Neural Network for Graphs: A Contextual Constructive Approach », *in*: *IEEE Transactions on Neural Networks* 20.*3* (2009), pp. 498–511, DOI: 10.1109/TNN.2008.2010350.

[68] Anthony J Myles et al., « An introduction to decision tree modeling », *in*: *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.*6* (2004), pp. 275–285.

[69] Nicolo Navarin et al., « LSTM networks for data-aware remaining time prediction of business process instances », *in*: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, USA: IEEE, 2017, pp. 1–7, DOI: 10.1109/SSCI.2017.8285184.

[70] Dominic A Neu, Johannes Lahann, and Peter Fettke, « A systematic literature review on state-of-the-art deep learning methods for process prediction », *in*: *Artificial Intelligence Review* 55.*2* (2022), pp. 801–827.

[71] Emanuel Parzen, « On estimation of a probability density function and mode », *in*: *The annals of mathematical statistics* 33.*3* (1962), pp. 1065–1076, DOI: 10.1214/aoms/1177704472.

[72] Vincenzo Pasquadibisceglie et al., « Predictive Process Mining Meets Computer Vision », *in*: *Business Process Management Forum*, ed. by Dirk Fahland et al., Cham: Springer International Publishing, 2020, pp. 176–192, ISBN: 978-3-030-58638-6.

[73] Gustavo Bernardi Pereira, Eduardo Alves Portela Santos, and Marcell Mariano Corrêa Maceno, « Process mining project methodology in healthcare: a case study in a tertiary hospital », *in*: *Network Modeling Analysis in Health Informatics and Bioinformatics* 9.*1* (2020), pp. 1–14.

[74] Patrick Philipp et al., « Analysis of Control Flow Graphs Using Graph Convolutional Neural Networks », *in*: *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, Johannesburg, South Africa: IEEE, 2019, pp. 73–77, DOI: 10.1109/ISCMI47871.2019.9004296.

[75] Anastasiia Pika et al., « Profiling event logs to configure risk indicators for process delays », *in*: *International Conference on Advanced Information Systems Engineering*, Springer, 2013, pp. 465–481.

[76] M Polato, « Dataset belonging to the help desk log of an Italian Company », *in*: *University of Padova, Padova* 1 (2017), DOI: 10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb.

[77] Mirko Polato et al., « Data-aware remaining time prediction of business process instances », *in*: *2014 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Beijing, China: IEEE, 2014, pp. 816–823.

[78] Mirko Polato et al., « Time and activity sequence prediction of business process instances », *in*: *Computing* 100.*9* (2018), pp. 1005–1031.

[79] P Pradhyumna, GP Shreya, et al., « Graph neural network (GNN) in image and video understanding using deep learning for computer vision applications », *in*: *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, IEEE, 2021, pp. 1183–1189.

[80] Efrén Rama-Maneiro, Juan Vidal, and Manuel Lama, « Deep learning for predictive business process monitoring: Review and benchmark », *in*: *IEEE Transactions on Services Computing* 1 (2021), pp. 1–1.

[81] Efrén Rama-Maneiro, Juan C Vidal, and Manuel Lama, « Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring », *in*: *arXiv preprint arXiv:2112.09641* 1 (2021).

[82] Williams Rizzi et al., « Nirdizati: an Advanced Predictive Process Monitoring Toolkit », *in*: *arXiv preprint arXiv:2210.09688* (2022).

[83] Anne Rozinat and Wil MP Van der Aalst, « Conformance checking of processes based on monitoring real behavior », *in*: *Information Systems* 33.*1* (2008), pp. 64–95.

[84] Anne Rozinat and Wil MP Van der Aalst, « Conformance testing: Measuring the fit and appropriateness of event logs and process models », *in*: *International Conference on Business Process Management*, Springer, 2005, pp. 163–176.

[85] Edson Ruschel, Eduardo Alves Portela Santos, and Eduardo de Freitas Rocha Loures, « Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing », *in*: *Journal of Intelligent Manufacturing* 31.*1* (2020), pp. 53–72.

[86] Claude Sammut and Geoffrey I Webb, « Encyclopedia of machine learning », *in*: Springer Science & Business Media, 2011, pp. 600–601, DOI: https://doi.org/10.1007/978-0-387-30164-8.

[87] Franco Scarselli et al., « The Graph Neural Network Model », *in*: *IEEE Transactions on Neural Networks* 20.*1* (2009), pp. 61–80, DOI: 10.1109/TNN.2008.2005605.

[88] Michael Schlichtkrull et al., « Modeling relational data with graph convolutional networks », *in*: *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, Springer, 2018, pp. 593–607.

[89] Klaus Schwab, *The fourth industrial revolution*, Currency, 2017.

[90] David W Scott, *Multivariate density estimation: theory, practice, and visualization*, John Wiley & Sons, 2015.

[91] Takanari Seito and Satoshi Munakata, « Production scheduling based on deep reinforcement learning using graph convolutional neural network. », *in*: *ICAART (2)*, 2020, pp. 766–772.

[92] Arik Senderovich et al., « Intra and Inter-case Features in Predictive Process Monitoring: A Tale of Two Dimensions », *in*: *Business Process Management*, ed. by Josep Carmona, Gregor Engels, and Akhil Kumar, Cham: Springer International Publishing, 2017, pp. 306–323, ISBN: 978-3-319-65000-5.

[93] Samuel Sanford Shapiro and Martin B Wilk, « An analysis of variance test for normality (complete samples) », *in*: *Biometrika* 52.*3/4* (1965), pp. 591–611, DOI: https://doi.org/10.2307/2333709.

[94] M.J. Shaw, *Information-Based Manufacturing: Technology, Strategy and Industrial Applications*, Springer US, 2012, ISBN: 9781461515999, URL: https://books.google.fr/books?id=F6wJCAAAQBAJ.

[95] Simon J Sheather and Michael C Jones, « A reliable data-based bandwidth selection method for kernel density estimation », *in*: *Journal of the Royal Statistical Society: Series B (Methodological)* 53.*3* (1991), pp. 683–690, DOI: 10.2307/2345597.

[96] Bernard W Silverman, *Density estimation for statistics and data analysis*, Routledge, 2018.

[97] Martin Simonovsky and Nikos Komodakis, « Graphvae: Towards generation of small graphs using variational autoencoders », *in*: *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, Springer, 2018, pp. 412–422.

[98] Dominique Sommers, Vlado Menkovski, and Dirk Fahland, « Process Discovery Using Graph Neural Networks », *in*: *2021 3rd International Conference on Process Mining (ICPM)*, 2021, pp. 40–47, DOI: 10.1109/ICPM53251.2021.9576849.

[99] Minseok Song and Wil M. P. van der Aalst, « Towards Comprehensive Support for Organizational Mining », *in*: *Decis. Support Syst.* 46.*1* (Dec. 2008), pp. 300–317, ISSN: 0167-9236, DOI: 10.1016/j.dss.2008.07.002, URL: https://doi.org/10.1016/j.dss.2008.07.002.

[100] A. Sperduti and A. Starita, « Supervised neural networks for the classification of structures », *in*: *IEEE Transactions on Neural Networks* 8.*3* (1997), pp. 714–735, DOI: 10.1109/72.572108.

[101] Sjoerd van der Spoel, Maurice van Keulen, and Chintan Amrit, « Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital », *in*: *Data-Driven Process Discovery and Analysis*, ed. by Philippe Cudre-Mauroux, Paolo Ceravolo, and Dragan Gašević, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 60–83, ISBN: 978-3-642-40919-6.

[102] Niek Tax et al., « Predictive Business Process Monitoring with LSTM Neural Networks », *in*: *Advanced Information Systems Engineering*, ed. by Eric Dubois and Klaus Pohl, Cham: Springer International Publishing, 2017, pp. 477–492, ISBN: 978-3-319-59536-8.

[103] Farbod Taymouri, Marcello La Rosa, and Sarah M. Erfani, « A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences », *in*: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*,

Virtual conference: Proceeding of SIAM, 2021, pp. 522–530, DOI: 10.1137/1.9781611976700.59, eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976700.59, URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976700.59.

[104] Berwin A Turlach, « Bandwidth selection in kernel density estimation: A review », *in*: *CORE and Institut de Statistique*, Citeseer, 1993.

[105] Álvaro Valencia Parra et al., « Enabling process mining in aircraft manufactures: extracting event logs and discovering processes from complex data », *in*: *BPM2019IF: 17th International Conference on Business Process Management 2019 Industry Forum (2019), pp. 166-177.* CEUR Workshop Proceedings (CEUR-WS. org), 2019.

[106] Wil Van Der Aalst, « Data science in action », *in*: *Process mining*, Springer, 2016, pp. 3–23.

[107] Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song, « Time prediction based on process mining », *in*: *Information systems* 36.*2* (2011), pp. 450–475.

[108] Ashish Vaswani et al., « Attention Is All You Need », *in*: *CoRR* abs/1706.03762 (2017), arXiv: 1706.03762, URL: http://arxiv.org/abs/1706.03762.

[109] Petar Veličković et al., « Graph attention networks », *in*: *arXiv preprint arXiv:1710.10903* (2017).

[110] Ishwar Venugopal et al., « A Comparison of Deep-Learning Methods for Analysing and Predicting Business Processes », *in*: *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Shenzhen, China: IEEE, 2021, pp. 1–8.

[111] Ilya Verenich et al., « Survey and Cross-Benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring », *in*: *ACM Trans. Intell. Syst. Technol.* 10.*4* (July 2019), ISSN: 2157-6904, DOI: 10.1145/3331449, URL: https://doi.org/10.1145/3331449.

[112] Ilya Verenich et al., « White-Box Prediction of Process Performance Indicators via Flow Analysis », *in*: *Proceedings of the 2017 International Conference on Software and System Process*, ICSSP 2017, Paris, France: Association for Computing Machinery, 2017, pp. 85–94, ISBN: 9781450352703, DOI: 10.1145/3084100.3084110, URL: https://doi.org/10.1145/3084100.3084110.

[113] Cédric Villani et al., *Optimal transport: old and new*, vol. 338, Springer, 2009.

[114] Mark Von Rosing, Henrik Von Scheel, and August-Wilhelm Scheer, *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume 1*, vol. 1, Morgan Kaufmann, 2014.

[115] Edward Wagstaff et al., « On the Limitations of Representing Functions on Sets », *in*: *Proceedings of the 36th International Conference on Machine Learning*, ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov, vol. 97, Proceedings of Machine Learning Research, PMLR, Sept. 2019, pp. 6487–6494, URL: https://proceedings.mlr.press/v97/wagstaff19a.html.

[116] Nur Ahmad Wahid et al., « Predictive business process monitoring–remaining time prediction using deep neural network with entity embedding », *in*: *Procedia Computer Science* 161 (2019), pp. 1080–1088.

[117] Matt P Wand and M Chris Jones, « Kernel smoothing », *in*: CRC press, 1994, pp. 90–92, DOI: https://doi.org/10.1201/b14876.

[118] A. Weijters, Wil Aalst, and Alves Medeiros, « Process Mining with the Heuristics Miner-algorithm », *in*: *Cirp Annals-manufacturing Technology* 166 (Jan. 2006).

[119] Sven Weinzierl, « Exploring gated graph sequence neural networks for predicting next process activities », *in*: *International Conference on Business Process Management*, Springer, Cham: Springer International Publishing, 2021, pp. 30–42.

[120] Oliver Wieder et al., « A compact review of molecular property prediction with graph neural networks », *in*: *Drug Discovery Today: Technologies* 37 (2020), pp. 1–12, ISSN: 1740-6749, DOI: https://doi.org/10.1016/j.ddtec.2020.11.009, URL: https://www.sciencedirect.com/science/article/pii/S1740674920300305.

[121] Wikipedia, Wikimedia Foundation, *Batch production*, [Online; accessed 15 Jan. 2023], URL: en.wikipedia.org/wiki/Batch_production..

[122] Lingfei Wu et al., « Graph neural networks for natural language processing: A survey », *in*: *Foundations and Trends® in Machine Learning* 16.*2* (2023), pp. 119–328.

[123] Shiwen Wu et al., « Graph Neural Networks in Recommender Systems: A Survey », *in*: *ACM Comput. Surv.* 55.*5* (Dec. 2022), ISSN: 0360-0300, DOI: 10.1145/3535101, URL: https://doi.org/10.1145/3535101.

[124] Hanna Yang et al., « A system architecture for manufacturing process analysis based on big data and process mining techniques », *in*: *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 1024–1029, DOI: 10.1109/BigData.2014.7004336.

[125] Rex Ying et al., « Graph convolutional neural networks for web-scale recommender systems », *in*: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[126] Jie Zhou et al., « Graph neural networks: A review of methods and applications », *in*: *AI open* 1 (2020), pp. 57–81.