A Machine Learning Approach to Predict Production Time in Industrialized Building Construction

by

Osama Mohsen

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

Abstract

Industrialized building construction is an effective approach for improving the performance and management of construction projects by offering higher quality products, minimized environmental impacts, and improved schedule predictability. The industrialized building construction approach integrates manufacturing principles and techniques into the construction industry where products (in this case, building components) are built in a controlled factory environment and then transported, in sequence, to the construction site for the final assembly.

With the marked growth of available data gathered as part of the daily operations in industrialized building construction facilities, one promising approach is to utilize machine learning techniques to identify valid, useful, and previously unknown patterns from historical data. These techniques are used to leverage the use of data to accurately predict the production cycle time. This thesis presents a framework to investigate potential improvements in estimating cycle time in industrialized building construction facilities where accurate prediction of cycle time can improve the quality of production planning and scheduling. The goal is to assist production/project managers to mitigate any delays and implement alternative actions should there be any unexpected delays due to factory operations as well as to manage the capacity and workload of the fabrication facility more efficiently.

The results of two case studies reveal that machine learning approaches can be successfully applied to accurately predict cycle time in different subsectors of industrialized building construction. The factors that significantly affect the prediction accuracy are (1) the physical characteristics and tracking information of products, (2) the engineered features that are generated to capture real-time loading conditions of the job shop, and (3) the lookback timeframe used for model training and validation. To examine the effect of shop loading features more thoroughly, a discrete-event simulation model is developed to investigate the various features that can be captured in the real system, but for practical reasons are not presently captured in the data, and to investigate their benefit in terms of improving the predictive accuracy. The major contribution of the research presented in this dissertation is that it provides practitioners in industrialized building construction with a roadmap to utilize their available production data for accurate estimates of delivery dates. The research results are also expected to be a point of reference for future studies in the academic field and for the industrialized building construction industry.

Preface

This thesis is an original work by Osama Mohsen. No part of this thesis has been previously published.

Dedication

To the soul of my father, Dr. Mohamed, to my kind and caring mother, Amal, to my loving wife, Mona, and to my beloved children Lyan, Raneem, and Mohamed, I dedicate this work.

"Foresight is not about predicting the future, it's about minimizing surprise."

Karl Schroeder

Acknowledgments

First of all, my utmost grace and gratitude goes to Almighty God for all his gifts and blessings upon me and for enlightening my path to complete this dissertation. My sincere appreciation goes to my supervisors Dr. Yasser Mohamed and Dr. Mohamed Al-Hussein for their continuous support, guidance, encouragement, and inspiration. Dr. Yasser Mohamed has been an exceptional academic mentor who never ceased to support me and always provide thoughtful and valuable advice in both my research and teaching journey. Dr. Maohmed Al-Hussein has always pushed me to do my best. They both have been great scholars to learn from in all aspects of life. I am also grateful to Dr. Ahmed Bouferguene for his valuable and insightful feedback given throughout my academic studies, and to the examining committee members, Dr. Ali Imanpour and Dr. Jeff Rankin for taking the time to review and critique my work and offer their feedback during my defense.

I would also like extend my profound thanks to my friends and colleagues in the Construction Engineering and Management group as well as the Department of Civil and Environmental Engineering at the University of Alberta. To Stephen Hague for his valuable help in developing the simulation model, to Jonathan Tomalty and Kristin Berg for their effort on editing and providing valuable suggestions on my work, and to the program service advisors for their responsiveness and help. Thanks also go to my peers with whom I have fruitful and thoughtful discussions, Emad Mohamed, Fatima Alsakka, Narges Sajadfar, Mohamad Darwish, AlaaaEldin Hebiba, Hadia Awad, and Aladdin Alwisy, from which I hope our discussions have helped them flourish as researchers and engineers as much as it helped me.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Description
AEC	Architecture, engineering, and construction
BIM	Building information modelling
BOM	Bill of material
BT	Buffer table (refer to chapter 4)
CC	Correlation coefficient
CCTV	Closed-circuit television
CNC	Computer numerical control
CRISP-DM	Cross-industry standard process model for data mining
CT	Production cycle time
CV	Cross-validation
DES	Discrete-event simulation
DI	Diameter inch
DM	Data mining
DSL	Diameter shop loading feature
DSR	Design science research
DT	Decision tree (model)
ERT	Extremely randomized tree (model)
ETL	Extract/transform/load
FE	Feature Engineering

Abbreviation	Description
FS	Framing station (refer to chapter 4)
GBDT	Gradient boosted decision trees (model)
IBC	Industrialized building construction
IFC	Industry Foundation Classes
ISL	Items shop loading feature
IT	Idle time
KDD	Knowledge discovery in databases
KNN	K-nearest neighbors (model)
LOOCV	Leave-one-out Cross-Validation
LR	Linear regression (model)
LT	Production lead time
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MaxAE	Maximum absolute error
MedAE	Median absolute error
MFB	Multifunction bridge (refer to chapter 4)
MIR	Material issue report date
ML	Machine learning
MSE	Mean square error
MTR	Material transfer report date
MWP	Multiwall panel

Abbreviation	Description
NDE	Non-destructive examination
NN	Neural networks (model)
OSC	Offsite construction
PC	Product composition properties
PT	Processing time
PWHT	Post-weld heat treatment
QC	Quality control
RAS	Required at site date
RDBMS	Rational database management system
RF	Random forests (model)
RFID	Radio frequency identification
RL	Reinforcement learning
SL	Shop loading
SS	Sheathing station (refer to chapter 4)
SSE	Sum of the squared errors
SSL	Surface area shop loading feature
SUR	Shop utilization ratio
TH	Throughput
TSCV	Time series cross-validation
WIP	Work in progress
WSL	Weight shop loading feature

Chapter 1 Introduction

1.1 Background

Industrialized building construction (IBC) approaches, including offsite construction, prefabrication, and modularization, are being used, in some form, by more than 80% of contractors in the U.S., underscoring the potential for future market growth [1]. In fact, it is projected that modular construction will reach $130 billion in market value in the U.S. and Europe by 2030 [2]. In Canada, modular construction accounted for a market value of more than 1 billion CAD in 2019 [3]. The growing level of interest in IBC by both industry and the academic community is due to the numerous benefits and advantages of IBC including shortened project durations, lower overall project costs, improved on-site safety, higher product quality, increased productivity, reduced construction wastes [4], and easier implementation of novel techniques and technologies.

Among the top motivation for increasing the adoption of prefabrication and modular construction is the improved project scheduling [5] accounting for an overall time savings of 20-50% compared to traditional onsite builds [2]. This is because building construction and site preparation activities can take place simultaneously, as shown in Figure 1-1, and there is less disruption due to extreme weather conditions. On the other hand, industrialized building construction is one of the more complex manufacturing environments due to numerous factors, namely, (i) the diversity of building components that leads, occasionally, to the full customization of building modules; (ii) the integration of many sub-systems (structural, electromechanical, finishing) that must be included in the building; (iii) the large and heavy machinery and components involved; and (iv) the numerous interrelated activities including production, assembly, finishing, delivery, and onsite installation [6]. All of this imposes challenges to accurately estimate activity durations at the initial stages of planning and scheduling.

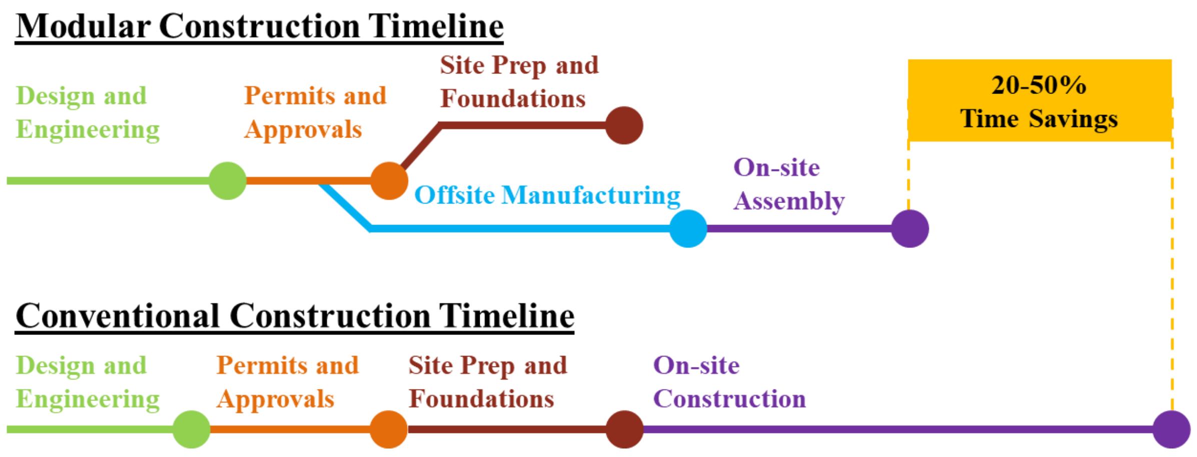


Figure 1-1. Time savings in modular construction

In any manufacturing process, as well as in the context of IBC, a job's cycle time (CT) is a key performance metric [7], which is defined as the time taken to complete the production process of one product, from start to finish. A closely related term is lead time (LT), which is defined as the time taken from order placement until the ordered item is delivered to the customer. Accurate forecasting of CT or LT for a product is crucial when determining customer delivery dates, when scheduling resources and actions, and when controlling and monitoring daily operations. In the literature, several methods have been developed to predict cycle time [8]. Simulation has been the most widely used approach in that it can realistically capture the complexities of manufacturing; however, practical disadvantages include the expense of model development and maintenance, and the emphasis on average performance rather than on the prediction of cycle time for an individual product. On the other hand, the most fundamental analytical approach to predict the average cycle time is Little's law [9], which states that the inventory between the start and end points of a product routing, which is referred to as work in progress (WIP), is equal to the product of average CT and throughput (TH), as shown in Equation 1-1:

$$WIP = CT \times TH \qquad 1\text{-}1$$

Such a static model does not predict well because it does not capture the stochastic nature of manufacturing; nevertheless, it can be used as a benchmark. Notwithstanding a large amount of data being collected in IBC factories, and the advances over the last decade to automatically gather and analyze production data, these data are not being fully utilized to improve the decision making in IBC. Therefore, the present study proposes a middle approach, between a simple statistical equation and more sophisticated simulation modelling, that uses machine learning (ML) techniques to predict cycle time, which is believed to provide many advantages to the industrialized building construction industry.

1.2 Research Motivation

Although industrialized building construction is an interdisciplinary field at the interface of manufacturing and construction, there have been relatively few studies that investigate the application of machine learning to predict cycle time in this domain. IBC typically employs a make-to-order production approach where products (i.e., building components or modules) are only built after a customer order has been confirmed. Products are also highly customized to satisfy customer requirements, and their production must be coordinated with site delivery; all the above factors make the prediction of cycle or lead time a complex task. This research study is conducted to fill this research gap in the body of knowledge, with practical application to the industry. Construction scheduling in IBC depends on the schedule of onsite activities, the factory production processes, and the expected delivery dates of raw material from suppliers. As such, production cycle time constitutes a critical component in this supply chain, from raw material to the end-product. The main goal of the present study is to devise a robust system for accurately predicting the cycle time, and, hence, to improve the predictability of that phase of the overall schedule. Having a more predictable production schedule will allow the production/project manager to mitigate any delays and implement alternative actions should there be any unexpected interruption to factory production. In addition, accurate prediction of cycle time enables effective management of the fabrication facility's capacity and workload. Figure 1-2 illustrates how the accurate prediction of cycle time improves factory scheduling by updating the baseline production schedule with real-time data in the context of the overall construction schedule.

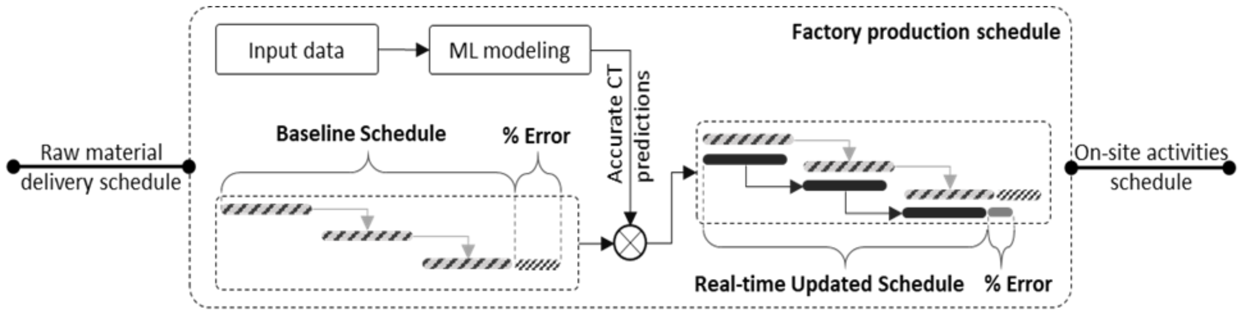

Input data
ML modeling
Factory production schedule
Accurate CT predictions
Baseline Schedule
% Error
Raw material delivery schedule
On-site activities schedule
Real-time Updated Schedule
% Error

Figure 1-2. Real-time update of production schedule by accurate prediction of CT

To validate the estimation of the production durations using machine learning approaches, case studies are investigated in two different IBC subsectors. One case is in the residential building sector where multiwall wood panels are produced in the factory and assembled onsite to construct single-family houses. The second case is in the industrial construction sector where prefabricated sections of steel pipes, called pipe spools, are manufactured offsite and delivered for onsite assembly. The two case studies are briefly described below.

1.2.1 Residential wall panels

In the residential housing industry, prefabricated panelized construction refers to a production system where a house is broken down into wall, floor, and roof elements. These elements are customized as per client requirements, manufactured in a factory, and then transported to the jobsite for assembly. The manufactured houses vary in size, shape, and style and have various customizable options for clients to choose from, adding to the uniqueness of each house [10]. In the wall production line, single-wall panels for each house are merged into multiwall panels to optimize workflow and reduce material waste, as shown in Figure 1-3.

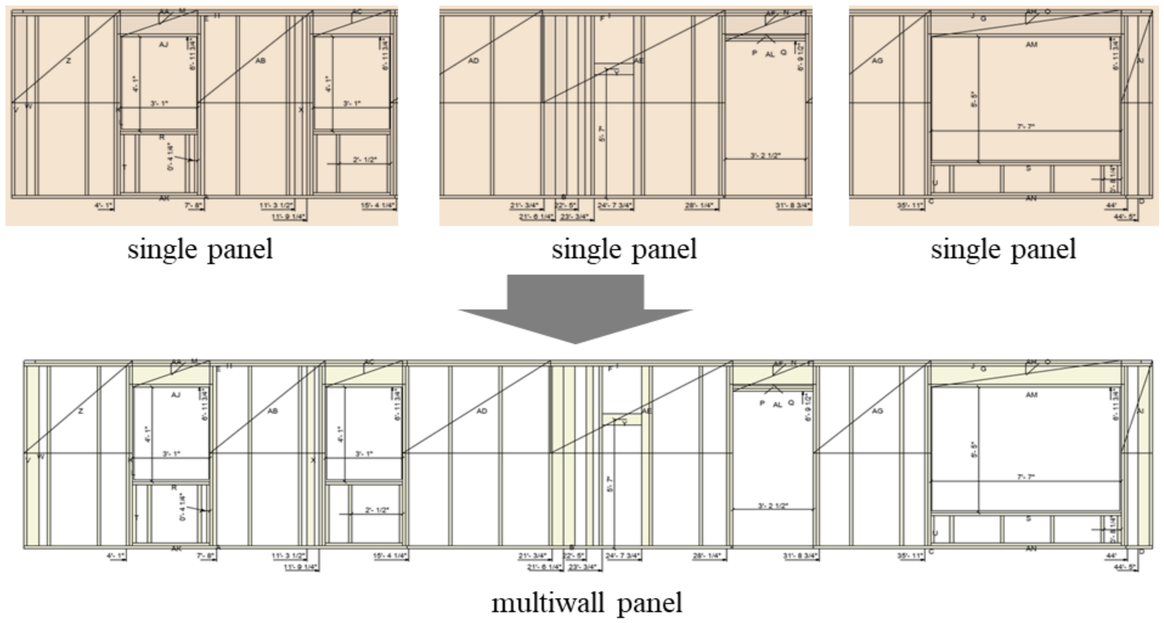


Figure 1-3. Merging of single-wall panels into a multiwall panel

Multiwall panels go through different processes and spend varying amounts of time at each workstation based on their physical properties (e.g., length, height, number and size of studs, window/door openings) as well as on the utilization of the production line. Also, the amount of work required to manufacture a wall panel can vary significantly based on the wall type. Interior walls require framing only, while exterior walls require framing, sheathing, nailing, and may also require window and door installation. Both types of walls are produced on the same production line causing variations that lead to line imbalances, which may cause large variations in waiting and processing times.

Predicting the cycle time of each wall panel has been based primarily on the experience of production managers, which is prone to human error and may not capture the variation that exits in the processing and waiting times at each individual workstation. Therefore, the proposed methodology in this thesis is applied to predict the cycle time using machine learning approaches which utilize the historical data about production of similar wall panels.

1.2.2 Industrial pipe spools

On industrial remote construction jobsites, massive process unit modules consisting of piping, electrical, and instrumentation equipment are prefabricated in locations closer to major cities, and then shipped to remote jobsites for assembly [11]. The prefabrication of pipe spools, representing the largest portion of modules, takes place in a fabrication yard or shop where heavy equipment is used to assemble sub-components (e.g., pipes, valves, elbows). Before the fabrication of modules, the associated spools need to be ready; however, a large inventory of pipe spools would increase the requirements for storage and double handling. Determining the right time to start the fabrication of a pipe spool can be challenging as there are many factors affecting the fabrication durations. Figure 1-4 summarizes the general processes involved in spool fabrication, each of which include several subtasks that are further described in Chapter 6.

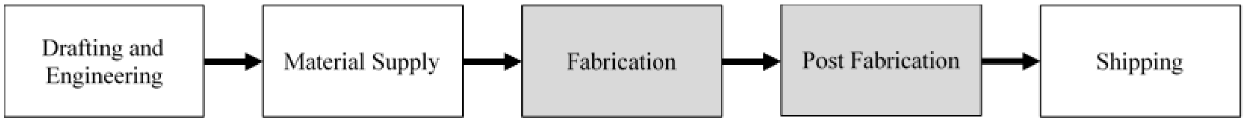


Figure 1-4. High-level workflow of the pipe spool fabrication process

In a pipe spool factory, fabrication durations are normally determined using heuristics based on the experience of production coordinators. This is traditionally done with little or no consideration to shop utilization and workforce availability, given there is a sufficient float allocated in the schedule [12]. Using heuristic rules for estimating fabrication durations, the lead time of the pipe spools cannot be accurately forecasted due to several factors. One factor is the difficulty in quantifying the impact of workforce availability on durations, which causes late delivery of some products, leading to low customer satisfaction. Meanwhile, other products are ready well in advance, leading to overloading on the storage and handling capacity of the plant. Another factor is the availability of raw materials. If there is shortage of materials, shop operations are disrupted resulting in delays that affect project completion; if there is an abundance of materials, the capacity and services of material storage and management are overloaded. Therefore, it is necessary to have a robust method to accurately predict the lead time so that the baseline schedule can be altered to reflect any possible delays caused by material and/or workforce availability. In this thesis, predicting lead time using historical data and machine learning approaches is proposed.

1.3 Research Objectives

One characteristic of industrialized building construction is the highly customizable products being manufactured. By using the latest advances in machine learning, the data that is accumulating continuously as part of the daily operations in an IBC facility can be leveraged to accurately predict cycle time, and, therefore, improve production planning and scheduling. To train data-driven models, we collect, integrate, and prepare data regarding the physical characteristics of building components, time-specific attributes capturing the planned production durations, and develop a set of engineered features that captures the real-time loading conditions of the job shop. This study explores the influence of the different product and process attributes on the predictability of cycle time. The proposed research is built upon the following hypothesis:

“In the context of industrialized building construction, historical data pertaining to the physical properties of building components and real-time tracking information, when augmented with engineered features (i.e., shop-loading attributes), will more accurately predict the cycle time required to manufacture new components, which will improve production planning and scheduling.”

Specifically, the present study tries to achieve the following four objectives:

1) To accurately predict the cycle time of producing wall panels using machine learning models and real-time RFID data in a residential manufacturing facility (Chapter 4).
2) To examine how prediction accuracy is improved by augmenting the dataset with domain-specific shop loading (SL) features, and to examine the effect of different grouping strategies used to combine several attributes of the dataset (Chapter 4).
3) To investigate which features should be collected, among those that are not practically collected, to improve the cycle time predictability using discrete-event simulation (Chapter 5).
4) To examine the generalization of the feature engineering and machine learning approaches by applying the methodology to an industrial pipe spool fabrication facility (Chapter 6).

1.4 Thesis Organization

This thesis consists of seven chapters. The current chapter provides a concise background of the current state of industrialized building construction, focusing on the use of machine learning approaches to tackle scheduling challenges, and briefly describing the two case studies investigated in this thesis. The research motivation and research objectives are also outlined.

Chapter 2 presents an overview of the literature that is relevant to the domain, problem, and methods used in the thesis. It includes sections on industrialized building construction, production scheduling, data mining concepts and techniques, and the industrial applications of knowledge discovery in databases in construction manufacturing settings.

In Chapter 3, the proposed framework to improve the predictability of production time in industrialized building construction facilities using machine learning approaches is presented. The specific research methods and steps are also described in detail.

Chapter 4 presents the implementation of the proposed framework in a case study of a residential wall panel fabrication facility. The knowledge discovery in databases approach is applied starting from data acquisition and concluding with model evaluation and lessons learned. The findings satisfy the first and second objectives of the research.

In Chapter 5, an experiment is developed using discrete-event simulation to investigate the effect of creating engineered shop-loading features on the accuracy of the predictive models. The simulation model is developed for the same case study presented in Chapter 4. The goal is to examine the features attributable to improving cycle time predictability, which aligns with the third objective.

Chapter 6 presents the second case study where the same framework is applied in an industrial pipe spool fabrication shop. The goal is to examine the generalization of the proposed approach to a different subsector of industrialized building construction.

The last chapter, Chapter 7, summarizes the work undertaken as described in the thesis and presents concluding remarks. Furthermore, future research work and academic and industrial contributions of the present research are presented.

Chapter 2 Literature Review

In this chapter, a review of the relevant literature is provided. Given the sophistication and complex nature of the research problem, four fields are covered in the literature review. The first section presents an overview of the industrialized building construction industry and the role that the Construction 4.0 paradigm plays in enabling more efficient design, planning, and delivery of construction products. The second section provides an overview of production planning focusing mainly on flow shop scheduling as it is more relevant to industrialized building construction. The third section provides an overview of data mining and machine learning techniques, focusing more on the methods used in the present research to predict production time. The fourth section reviews the industrial applications of knowledge discovery in databases in the construction industry, specifically the applications to predict production time in manufacturing facilities.

2.1 Industrialized Building Construction

2.1.1 Premise of industrialization in building construction

Various terms such as prefabrication, offsite fabrication, offsite construction (OSC), and modular construction, to name a few [13,14], might all refer to the same approach: a construction process that applies manufacturing principles and techniques to construction projects where products (in this case, building components) go through a particular life-cycle from concept, to design, to planning, to manufacturing, and then on-site assembly, as shown in Figure 2-1. The appropriate manufacturing model depends on the construction subsector (private housing, commercial building, industrial, major projects) being analyzed [15,16]. The term "industrialized building construction" (IBC) is used hereinafter to denote any of the terms above in which the various elements of residential, commercial, or industrial buildings are produced in a controlled factory environment. Building components (anything from bathrooms, prewired lighting fixtures, exterior walls, to complete building modules) are built in the factory and then transported, in sequence, to the construction site. At the site, a crane is used to lift components or modules off the transport truck and set them on a permanent foundation. Site crews then attach the components or modules to each other and to the foundation and seal the building. After setting the building, several finishing activities take place to complete the building [17–19]. Modular buildings comply with local building codes similar to those applicable to conventional construction, and are constructed using the same materials and details as conventional buildings, with minor modifications to accommodate shipping and installation [6]. The supply chain of industrialized building construction broadly includes eight main participants: customers (project owners), manufacturers, dealers, integrated companies (manufacturing and sale), material suppliers, design team (architects/engineers), general and specialty contractors, and local permit agencies [20].

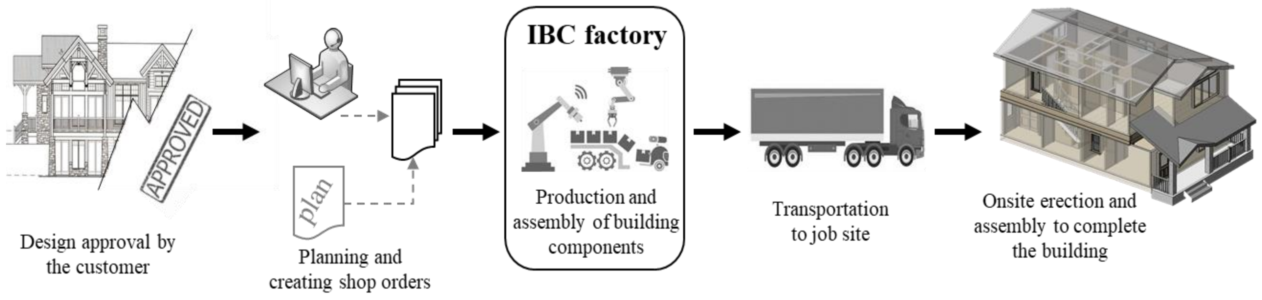

APPROVED
Design approval by the customer
plan
Planning and creating shop orders
IBC factory
Production and assembly of building components
Transportation to job site
Onsite erection and assembly to complete the building

Figure 2-1. Simplified high-level workflow in industrialized building construction

The burgeoning academic interest in the domain of modular and prefabricated construction has resulted in a large number of review articles related to industrialized building construction in the past decade. In the last three years, from 2018 to 2020, some articles applied topic modelling to examine trending topics and themes in IBC [21], some carried out critical reviews and bibliometric analysis to highlight key research words, identify research gaps, and suggest future research opportunities in industrialized building construction [22–24], while others focused on critical reviews of publications about emerging technologies and adoption of digitization in IBC [25–28]. Moreover, a few articles provided systematic reviews on specific topics such as policies and regulations related to IBC [29], constraints and barriers to the adoption of modular construction [30–34], construction risk management within the IBC industry [35], the environmental performance of OSC facilities [36], and supply chain management in IBC [37].

Project planning and design, including workflow optimization and project delivery processes, are among the more frequently studied research themes in IBC during the past decade [23]. Shortened project time and the enhanced predictability of activity durations are among the many benefits that are recognized and reported in the literature when using IBC approaches. As a considerable amount of data is being generated as part of the daily operations in the fabrication shop, the IBC environment enables the utilization of the latest technological advances brought by the Construction 4.0 paradigm which is described in the following section. Essentially, the development of machine learning models requires substantial amount of accurate data which can be provided by the manufacturing environment of IBC as opposed to the inferior and scattered data found in conventional construction. Acquiring, processing, and analyzing large amounts of data constitute one of the four technologies that are essential in Construction 4.0. Those four technologies are 3D printing, big data, virtual reality, and Internet of Things [38]. The research described in this thesis focuses on employing advanced data analytics techniques, including data mining and machine learning, to estimate the production cycle time of building components.

2.1.2 Industrialization of building construction and Construction 4.0

During the past few decades, the rate of adoption of industrialization techniques in the construction industry has been limited compared to other fields [39]. However, moving toward more digitized and automated processes offers enormous potential for productivity improvements. In recent years, the construction industry has realized the importance of using information technologies and data analytics approaches. The most recent evidence is the emerging paradigm of Construction 4.0, a term coined in the context of "Industry 4.0" which is the fourth industrial revolution involving automation, data exchange, and collaboration in manufacturing [40–42]. Muhuri et al. [43] provides a bibliometric analysis and an extensive overview of recent developments in the field of Industry 4.0. In essence, Construction 4.0 is the digitization and automation of the AEC industry. The ideas of Construction 4.0 can be based on two main concepts: digitization of the construction industry, and industrialization of construction processes [38,44]. Industrialized building construction promotes the enhancement of construction manufacturing processes by employing the concepts of mechanization, digitization, and automation in the prefab facility [28,45].

The Construction 4.0 framework provides a mechanism that facilitates the development of a digitized model of the built environment and/or its processes that supports the design, planning, monitoring and delivery of construction products. In the context of IBC, the driving technologies and concepts brought by Construction 4.0 that are relevant in this thesis are the following:

1. Building information modelling (BIM);
2. Automatic data acquisition such as radio frequency identification (RFID); and
3. Advanced data analytics, specifically data mining (DM) and machine learning (ML).

BIM, as one of the most promising developments, has been increasingly adopted in the architecture, engineering, and construction (AEC) fields. The National Institute of Building Sciences describes BIM as "a digital representation of physical and functional characteristics of a facility … a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition" [46].

BIM consists mainly of a computer-generated 3D model of a building that contains accurate and well-defined geometry data as well as an information database that AEC practitioners can use to design the building and simulate its construction [47]. To improve the performance of the construction industry and to realize the benefits of both the BIM technology and IBC approach, the joint application of BIM and IBC has seen increasing applications in both industry and academia [25]. According to the 2020 American Institute of Architects (AIA) Firm Survey Report, BIM tools are being used by 100% of large firms, 88% of medium size firms, and 37% of small firms. Moreover, the global BIM market is expected to grow from $4.5 billion in 2020 to $8.8 billion in 2024 [48]. BIM uses Industry Foundation Classes (IFC) for exchanging building information among different CAD packages [49]. In the present research, BIM is primarily used to obtain data about the physical characteristics of building components that are being manufactured in an IBC facility. This data will be combined with product tracking information obtained from automatic tracking systems, such as RFID, which will be used as the basis for developing data-driven ML models to assist in project planning and scheduling.

To enable the application of machine learning predictive models, the availability of large datasets is crucial. Automatic real-time data acquisition for construction project tracking includes, but is not limited to, enhanced IT, geospatial, 3D imaging, and augmented reality technologies [50]. One technology common to IBC is the radio frequency identification (RFID) tracking system. RFID is used as a sensing mechanism to locate product tags using the electromagnetic field of antennas that are connected to RFID readers populating a central database. Within any specific setup, each tag contains unique identification information. Figure 2-2 illustrates a typical RFID system setup consisting of three main components: a) scanning antennas, b) a transceiver that reads and interprets the data, and c) transponders/tags that have been programmed with information and are attached to the objects (e.g., wall panels) being tracked. In the literature, Valero et al. (2015) [51] and Valero and Adán (2016) [52] provide a comprehensive review of recent RFID applications in the construction industry.

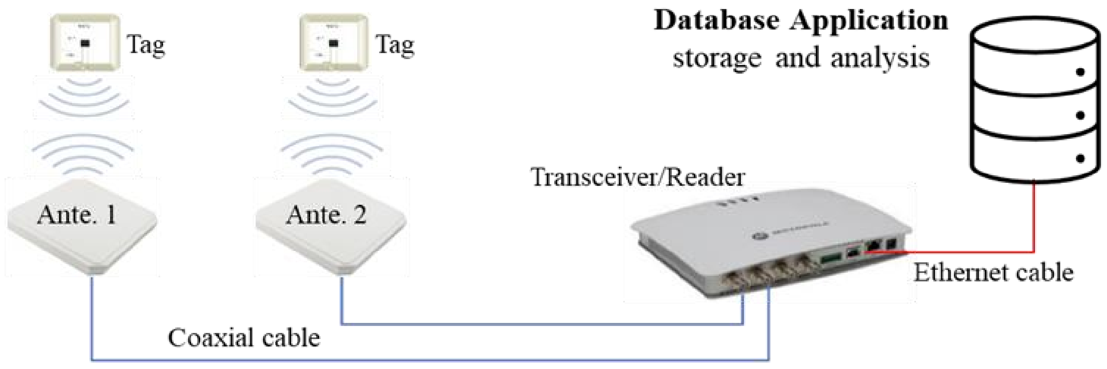


Figure 2-2. Example of a typical RFID system setup

Data mining and machine learning techniques are further discussed in Section 2.3

2.2 Production Scheduling in IBC

2.2.1 Project scheduling and control

Essential insights from the production management field have been and continue to be transferred to the field of project management [53,54]. Effective project management needs to be integrated with effective production management to achieve the highest operational improvements. For a given scope of work, the basic tasks of a project manager involve planning, scheduling, and control of project activities by using the available resources effectively and efficiently. In the planning phase, a listing of activities that must be performed to complete the project is developed including determining the different types of required resources and estimating the duration and costs of activities. In scheduling, the chronological ordering of the actual tasks is performed, the actual resources needed at each stage in the project are calculated, and the expected completion time of each activity is determined. In the control phase, the difference between the scheduled and actual performance once the project has started is closely monitored and analyzed [55].

Scheduling has been defined as “the determination of the timing and sequence of operations in the project and their assembly to give the overall completion time” [56] and the focus of the research presented in this thesis is on estimating activity durations. Per PMI’s Guide to the Project Management Body of Knowledge (PMBOK), estimating activity durations is a process of the Project Schedule Management knowledge area which encompass several techniques to determine duration estimates (e.g., expert judgement, analogous estimating, bottom-up estimating) [57]. Developing a schedule for an IBC manufacturing environment, such as panelized wall production or pipe-spool fabrication, poses unique challenges because of the stochastic nature of such environments due to the varying customer demands and the highly customizable products being produced. All planned projects carry an element of risk due to the uncertainties involved during the planning and scheduling phases. There has been ample research on production planning and control focusing on different types of manufacturing environments. A few studies were conducted to improve the production process at panelized wall fabrication facilities. For example, a lean production approach to improve process planning and control in the home building industry was developed by Yu [58]; a methodology to improve the panelized home production process through lean principles and simulation tools was proposed by Shafai [59]; and an integrated production planning framework for panelized residential buildings using simulation and RFID tracking system was presented by Altaf [60]. In addition, an integrated project control and monitoring framework in a steel prefabrication facility using RFID technology and discrete-event simulation was presented by Azimi et al. [61]. Although many studies and techniques have investigated production planning and control in industrialized building construction, the application of advanced data analytics to improve the planning practices and enhance the schedule predictability has not been fully examined in the construction industry.

In practice, activity durations in an IBC manufacturing facility are either estimated based on: (1) average times obtained from historical data and, for the most part, rely on the experience of project coordinators or trade foremen for specific job conditions [12], or (2) the unit quantity multiplied by the production rate (time/unit) for the activity scope of work in the case of deterministic scheduling [62]. Broadly speaking, there are two main approaches to develop schedules in IBC, and more generally in construction. The first approach is the traditional critical path method (CPM), a forward calculation method where the scheduling begins with a fixed start date and the durations of interdependent activities are estimated to provide an expected completion date. The second approach is the pull scheduling process, a backward calculation method where the schedule is developed starting at the end (i.e., based on an approved completion date) and the durations of each predecessor activity are identified providing a required start date for each product [63,64]. Both approaches are being used in IBC manufacturing; however, the pull-driven approach based on lean concepts is gaining more acceptance for operations within the production facility. Nevertheless, the adoption of any scheduling approach and the associated productivity rates is company-specific [62], and every facility adjusts its scheduling practice as to improve its overall operational performance.

In one case study, Tommelein, I. (1998) used process modelling to compare various scheduling approaches in a pipe-spool installation project and concluded that the lean-production pull-driven technique for scheduling improves the performance of construction processes [63]. Mosayebi et al. (2012) estimated the effect of different factors on the productivity and, therefore, the duration of pipe spool fabrication based on observations from previous projects and the expert opinion of shop superintendents. They concluded that one important factor is the configuration of spools, which affects the workflow in the shop and can result in an increase in the time required to fabricate a pipe spool by up to 50% [65]. Other case studies investigated different factors that affect the productivity and, hence, the development of the production schedule in industrialized building construction manufacturing facility. For example, these factors include using standardized as opposed to one-of-a-kind components [66], shop layouts and the associated lean production flow techniques [67], and changing the sequence of assembly activities [68]. In previous case studies, estimating activity durations appeared to largely depend on: (1) the experience of the production coordinators or shop foremen, (2) simulated data that were arbitrarily chosen or based on previous empirical research studies, or (3) averages of historical data that do not capture the uniqueness of the products or processes under study.

2.2.2 Flow shop scheduling problem

Since the early 1960s, many textbooks and manuscripts have been published focusing on sequencing and scheduling. More recently, some notable books that cover many aspects of scheduling, from the elementary to the more advanced topics, include: *Project Scheduling A Research Handbook* by Demeulemeester and Herroelen (2002) [55]; *Handbook of Scheduling Algorithms, Models, and Performance Analysis* edited by Leung (2004) [69]; *Scheduling Theory, Algorithms, and Systems* by Pinedo (2016) [70]; and *Principles of Sequencing and Scheduling* by Baker and Trietsch (2018) [71].

Scheduling models are broadly categorized as either deterministic or stochastic [70]. They basically differ in that "deterministic scheduling involves solving a scheduling problem … when the various parameters, viz., job processing times, due dates, release dates, and so on, are known with certainty. On the other hand, stochastic scheduling deals with problems when at least one of these parameters is not known with certainty" [72]. As the processing times and actual due dates are not known with certainty, scheduling in an IBC manufacturing environment is a prime example of a stochastic scheduling problem.

In addition to categorizing scheduling models as either deterministic or stochastic, the manufacturing environment in which these models are applied can be sectorized according to the layout of the machines, or workstations, and according to how the different job processing routes are specified. Basic manufacturing layouts include single machine, parallel machine, flow shop, job shop, open shop, and hybrid layouts [73], which are briefly described as follows.

a) Single machine layout: The shop layout is simply formed by a single machine where each job must be processed on that machine only once without rerouting. Each job is processed by the machine for a duration defined as p_j processing time units then the job leaves the machine once finished.

b) Parallel machine layout: As an extension of the single machine layout, a parallel machine layout has several parallel machines where each job must be processed one by one of *m* available machines. Parallel machine models are divided into three categories based on the speed by which the jobs are processed: identical parallel machines, uniform parallel machines, and unrelated parallel machines.

c) Flow shop layout: In a flow shop, there are *m* machines (i.e., workstations) organized in series, each of which performs a specific task. All the jobs must follow the same processing order, and each job must visit all machines. When the processing sequence is the same for all machines (i.e., a job cannot pass another while waiting in a queue), the flow shop is referred to as permutation flow job.

d) Job shop layout: Job shops are similar to the flow shops in that there are *m* machines organized in series. However, although each job might have a different route when visiting the machines, each job must visit all machines exactly once.

e) Open shop layout: The open shop is the more general and complex layout setting where there are *m* machines in series and *n* jobs to be processed by all machines. In this case, each job can be processed by the machines in any order making the routing of all jobs an arbitrary, but flexible sequencing task.

f) Hybrid flow layout: In a hybrid layout, there are *m* production stages, each stage with m_i parallel machines that may be either identical, uniform, or unrelated. The number of parallel machines at each stage might be different and all routing options are possible. Each job must be processed once at each stage by one of the available parallel machines.

In addition to the above basic layouts, some prototypical manufacturing environments include manufacturing cells, assembly shops, production lines, batch shops, and flexible manufacturing systems.

Most IBC manufacturing environments (e.g., panelized wall production or pipe-spool fabrication) resemble a mix of flow shops (i.e., a series of machines m_{i} each perform a specific operation on each job *j*), and assembly shops or production lines. In an assembly shop, operations for some jobs need to be finished in order for subsequent operations to begin, which will utilize the previous operations as raw materials. In a production line (also known as an assembly line), products move along via some means of transportation (e.g., a conveyor track) and operations are performed on the products with no buffer between operations. To be consistent with the scheduling literature, the term “job” refers to the products to be manufactured and in the case of an IBC manufacturing shop, the job is the building module, component, or part that needs to be fabricated.

According to the *complexity theory*, a branch of computer science, the relationship between scheduling problems and their solution methods is considered a combinatorial optimization problem. The complexity here refers to the “computing effort required by a solution algorithm” [71] which measures, roughly speaking, the number of computations as a function of the problem size *n*. Most scheduling problems, except for simple single-machine problems, are classified as non-deterministic polynomial-time (NP-hard) problems, meaning that an optimal solution to large scheduling problems is unlikely to be found in an acceptable polynomial time [73]. Therefore, heuristic solution procedures can offer near-optimal solutions that are practically acceptable with modest computational requirements. A heuristic method can be defined as “a logical sequence of steps, the execution of which yields a solution, which is not necessarily optimal. However, the solution is usually good enough to be used in practice for planning and control purposes” [55]. Most research efforts over the past few decades have been directed toward using heuristics to find feasible schedules in flow shops [74–76], job shops [77,78], and distributed permutation flow shops [79–82]. A comparative study that examined existing heuristics and metaheuristics methods for the permutation flow shop was carried out by Fernandez-Viagas et al. [83].

The current literature has focused on solving the flow shop scheduling problem either with fixed activity durations or probabilistic durations. The previous studies mainly examined the sequencing problem and how to find job sequencing that optimizes certain desired criteria (e.g., minimum overall production duration). However, none of the studies, as to my knowledge, has focused on developing a framework to accurately estimate individual job durations in a stochastic environment for industrialized building construction, using the latest advances in machine learning and data mining techniques. The present research described in this thesis investigates this application and is set to fill the research gap in the body of knowledge in the context of IBC.

2.3 Knowledge Discovery in Databases and Data Mining

2.3.1 Knowledge and data

The term knowledge discovery in databases (KDD) had been adopted by the end of the 1980s to replace all the terms referring to methods of finding patterns and similarities in raw data, including knowledge extraction, information discovery, information harvesting, data archeology, and data pattern processing. KDD is defined as the overall partially automated process of finding valid, useful, and previously unknown knowledge from data, and data mining (DM) refers to a particular step in this process [84–86]. The process of generating value from an organization's intangible assets is known as knowledge management [87]. Knowledge can be represented as a tree with data at its base, and with information, knowledge and wisdom at the top as shown in Figure 2-3 [88,89].

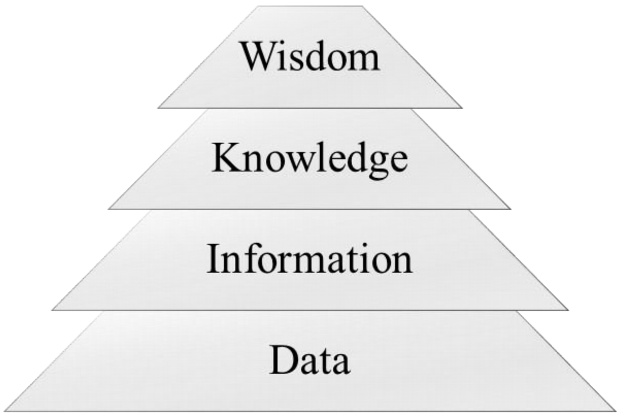


Figure 2-3. Knowledge Tree (from data to wisdom)

Data is unorganized information that is processed to make it meaningful, and it consists of facts, observations, perceptions, numbers, characters, symbols, and images that can be interpreted to derive meaning. In the construction industry, data represents raw elements such as timestamps used to track products along the production line or actual daily resource utilization figures during the execution of a project. Data is transformed to information when these raw elements are patterned in a certain way making them ready for analysis. Knowledge is then generated as actionable information when certain rules or heuristics are applied to the information, for the purpose of producing value-added benefits. Knowledge synthesizes and integrates the domain context, organizational culture, standards, human expertise, management initiatives, and lessons learned. Wisdom in the knowledge tree signifies the ability to make the right decisions at the right time using the acquired knowledge to maximize the tangible and intangible benefits. Wisdom is the value added to knowledge [90].

There are many data sources in construction projects. One of the ways in which raw data can be categorized is by its structure; it can be structured, semi-structured, or unstructured [91]:

a) Structured data: Structured data has a well-defined structure and can be stored in well-defined schemas such as databases and in many cases can be represented in a tabular format with rows and columns. Structured data is objective facts and numbers that can be collected, exported, stored, and organized in typical databases. Some of the sources of structured data include SQL databases, BIM models, spreadsheets such as Excel, and sensors such as RFID tags.

b) Semi-structured data: Semi-structured data is data that has some organizational properties but lacks a fixed or rigid schema. Semi-structured data cannot be stored in the form of rows and columns as in databases. It contains tags and elements, or metadata, which is used to group data and organize it in a hierarchy. Some of the sources of semi-structured data include E-mails, XML, JSON and other markup languages, and zipped files.

c) Unstructured data: Unstructured data is data that does not have an easily identifiable structure and, therefore, cannot be organized in a conventional relational database in the form of rows and columns. It does not follow any particular format, sequence, semantics, or rules. Unstructured data can deal with the heterogeneity of sources and has a variety of business intelligence and analytics applications. Some of the sources of unstructured data include web pages, social media feeds, images, video and audio files, and PDF files.

In the present research, the data used to predict production time is composed primarily of structured data. Here, the raw data represents the physical properties of the building components being manufactured, the time-related data used to track the products, as well as a set of engineered attributes (i.e., inferred information) generated to capture the system utilization.

2.3.2 Data mining and tidy data

Data mining may be defined as "the extraction of implicit, previously unknown, and potentially useful information from data" [90]. The purpose of DM is to develop computer programs that examine databases automatically, in search of uniformities or useful patterns to inform intelligent decision making based on data. In this context, machine learning algorithms provide the technical basis of data mining. Since the two terms (machine learning and data mining) are closely interrelated and the differences between them might not be very clear in some contexts, throughout this thesis the two terms may be used interchangeably. Machine learning algorithms build mathematical models from historical data in order to make predictions when supplied with new data without being explicitly programmed to perform this task. For an application of data mining to be effective, domain-specific knowledge is typically integrated within the ML algorithms to obtain representative results [92]. The application of a data mining process involves:

(i) Creating a dataset: The data mining process starts by creating a dataset that represents some aspects of the real world. A structured dataset consists of the following: (1) instances (also referred to as observations, records, or examples), which are objects in the real world that represent single observations; and (2) features (also referred to as variables or attributes), which are measurements of the different dimensions or aspects of the instances in the dataset. In essence, we need to present the data in a "tidy" form. Tidy data is a common term used in data science and has been formally defined by Wickham (2014) [93] as "a standard way of mapping the meaning of a dataset to its structure" in which each variable/feature forms a column, each observation/record forms a row, and each type of observational unit forms a table.

(ii) Tuning the algorithm: Many machine learning techniques have algorithm-dependent hyperparameters that are selected based on parametric studies, the type of problems, empirical studies, and even the researcher's personal experience. The performance (e.g., accuracy) of most ML algorithms depends strongly on finding an optimal set of hyperparameters. There exist various strategies to search for the best combination of hyperparameters including grid search and random search [94,95]. A combination of manual and grid/random search is used to search for the optimal combination of a predefined set of hyperparameters. This tuning allows the algorithm to make better predictions about the new data instances.

Data mining, and data analytics, can be categorized by the different levels of difficulty, value, and intelligence they entail [96]. Data analytics can be (i) descriptive to drive patterns that summarize the datasets; (ii) predictive to forecast the values of certain attributes based on historical performance; or (iii) prescriptive to derive actions of what should be done and why [97]. Another way to categorize data mining tasks is by their functionality, as described by Han et al. (2012), which leads to five categories: characterization and discrimination; association and correlation; classification and regression; clustering analysis; and outlier analysis [98].

Traditional data analysis techniques (e.g., stochastic models and time-series analysis) have major limitations in terms of extracting knowledge from datasets and are based on developing mathematical models representing the relationships between established variables. In many cases, neither the variables nor the relationships between them are easily established [89]. To overcome these limitations, data mining techniques are used. The data mining process is typically carried out as part of a larger KDD framework. The KDD used in the present research, within which the DM process is performed, is described in Chapter 3 as part of the research methodology.

2.3.3 Machine learning algorithms

2.3.3.1 Variables and terminology

As described in the previous section, a record in a dataset is characterized by a set of variables or features. The set of variables is called inputs, or predictors, when they are measured or preset and have some influence on one or more outputs, or responses. In classical statistical literature, inputs are called *independent variables*, and outputs are called *dependent variables* [99]. For each example in the dataset, the output is also known as the label, class, or target value. Following the same terminology in the literature, hereinafter, an input variable is denoted by X, while output variables are denoted by Y. If X is a vector, its elements can be accessed by subscripts X_j. Upper case letters are used to denote the generic aspects of a variable while lower case letters refer to the actual observed values. A matrix of n input vectors each of size m represents the $n \times m$ matrix having n examples each with m input variables (features) from the dataset. Because all vectors are assumed to represent column vectors, the ith row of the matrix X is x_i^T, the vector transpose of x_i. In summary, the learning algorithm is built using the value of an input vector X in order to make a good estimation of the output Y, which is typically denoted by $\hat{Y}$. If $Y \in \mathbb{R}$, then it is necessary that $\hat{Y} \in \mathbb{R}$.

When carrying out a KDD experiment, a dataset is typically divided into two subsets: one to build the predictive model (*training set*); and one to test the model performance (*testing set*). This is true in the case of supervised learning, which will be described in the next subsection. Building a predictive machine learning model is sometimes referred to as building a *learner* that will be used to predict the outcome of new unseen data [99]. Various performance metrics are used to measure how good a learner is; these evaluation metrics are discussed in Section 2.3.4.1.

2.3.3.2 Types of machine learning algorithms

Depending on the problem to be solved and the purpose of the learning, machine learning algorithms can be broadly divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning algorithms [100]. Figure 2-4 shows the broad categories and subcategories of machine learning algorithms, along with sample use cases.

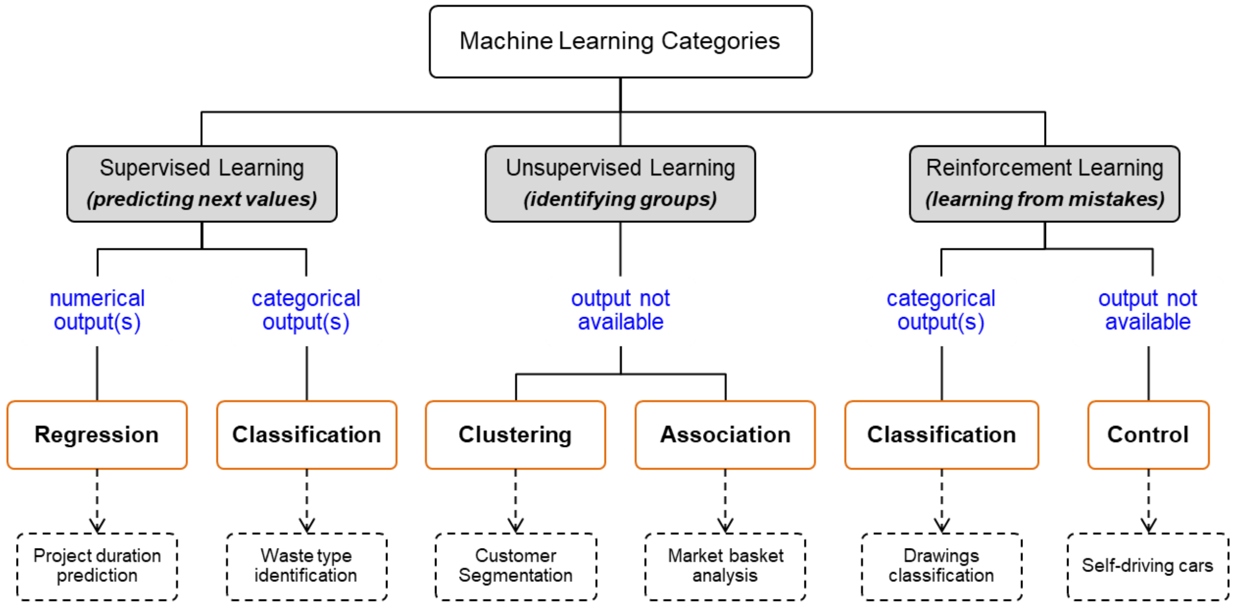


Figure 2-4. Broad categorization of machine learning algorithms

a) Supervised Learning:

In supervised learning, the development of a learner is being supervised by the labelled data (i.e., by the presence of the outcome variable, *Y*) to learn the relationships and dependencies between the input variables and the target output such that these relationships can be used to predict the output values of new data. Supervised learning algorithms are primarily used for predictive modelling and analysis. Based on the type of output variable, supervised learning algorithms are either (i) regression algorithms that predict quantitative output (continuous values); or (ii) classification algorithms that predict qualitative output (discrete values).

The main objective of this research is the prediction of production time in IBC, which is an example of a regression task. Regression learning is described in more detail in the next subsection. For a comprehensive overview of classification techniques, the interested reader can refer to some notable textbooks about data mining and machine learning such as *Data Mining: Practical Machine Learning Tools and Techniques* [90], which focuses on practical applications, and *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* [99], which provides more theoretical coverage.

b) Unsupervised Learning:

In unsupervised learning, there is no dependent random variable, output, or response (i.e., no outcome measure, *Y*) based upon which the algorithm can build relationships. The goal here is to analyze a set of input observations in order to describe how the data is organized or clustered. Therefore, unsupervised learning algorithms are primarily used for pattern detection and descriptive modelling. Unsupervised learning techniques include:

1) Clustering analysis: In clustering, also called data segmentation, objects that seem to fall naturally together are grouped. Data objects are grouped based on the concept of minimizing the distances between data points within the same cluster (i.e., objects are similar), and maximizing the distance between clusters (i.e., dissimilar objects) [98]. Distances can be calculated using the Minkowski generalized distance formula between two vectors $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}$ and $Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}$ as shown in Equation 2-1:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \qquad 2\text{-}1$$

where p=1 calculates the Manhattan distance and p=2 calculates the Euclidian distance.

Han et al. (2012) [98] grouped the major clustering methods into the following categories:

- Partitioning methods construct *k* clusters of the data and iteratively assign the *n* data points to each cluster. Most partitioning methods are distance-based algorithms such as *k-means* [101,102], *k-medoids* [103,104], and *k-modes* [105] algorithms.
- Hierarchy methods create hierarchical decompositions of the given data points. Hierarchical methods can be either agglomerative (bottom-up approach) or divisive (top-down approach), based on how the hierarchical decomposition is formed [103].
- Density-based methods create clusters based on the density of data points within their neighbourhood. For each data point within a cluster, the neighborhood of a given radius has to contain at least a minimum number of points [98] making them robust against outliers. These methods include DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [106], and DENCLUE (DENsity-based CLUstEring) [107].

- Grid-based methods create a finite number of cells of the data space forming a grid structure where all clustering operations are performed. An advantage of this approach is being independent of the number of data points making processing time fast. Such methods include STING (STatistical INformation Grid) [108], and WaveCluster [109].

2) Association analysis: Frequent patterns are patterns (e.g., itemset) that appear frequently in a dataset. Association rules are deduced for a set of items that often appear together in a large transactional or relational dataset. These items are the input variables that are said to be strongly associated with each other. A typical example of frequent item mining is *market basket analysis* in which the task is to analyze customer shopping habits by finding associations among the items that are bought together. Association rules are said to be strong if they satisfy both a minimum Support and Confidence levels. Support measures the percentage of transactions in the dataset that a given rule satisfies and confidence measures the degree of certainty of the detected association [98]. Let T be the total number of transactions in a dataset, A and B be two sets of items, and C is the number of transactions containing both A and B; then, the association rule is an implication of the form $A \Longrightarrow B$ and we can calculate the support and confidence as shown in Equations 2-2 and 2-3:

$$Support(A \Longrightarrow B) = P(A \cup B) = \frac{|\{A \cup B\}|}{T} \qquad 2\text{-}2$$

$$Confidence(A \Longrightarrow B) = P(B|A) = \frac{|\{A \cup B\}|}{C} \qquad 2\text{-}3$$

As the number of attribute values (i.e., itemsets) increases, the number of rules significantly increases, which is a major problem of this approach. Several algorithms are available to solve the problem such as Naïve, Apriori, and the Frequent Pattern Tree [89].

c) Reinforcement Learning:

Reinforcement learning (RL) is a very broad topic and is beyond the scope of techniques used in the present research; however, RL is briefly described here for the completeness of ML categorization. The goal in RL is to develop a system (agent) that learns continually from its environment by interacting with it. The agent receives a positive or negative reward based on the action it performs, as shown in Figure 2-5. Since the information about the current state of the environment typically includes what is known as reward signals, we can think of RL as an area related to supervised learning [110]. However, in reinforcement learning this feedback is not the actual (i.e., ground truth) measured value, but rather a measure of how well the action was evaluated by some reward function. An agent can then use RL, through its interaction with the environment, to learn the actions that maximize this reward by trial-and-error approaches or by careful and deliberate planning. For further description, interested readers can refer to the book by Sutton and Barto: *Reinforcement Learning: an Introduction* [111].

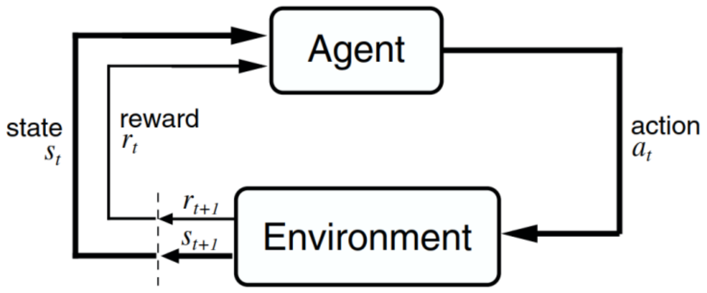


Figure 2-5. Interaction of agent and environment in reinforcement learning [111]

2.3.3.3 Supervised learning regression techniques

In a regression learning task, we are given a number of input variables (predictors) and a continuous output variable (target), and we try to find a relationship between those variables that allows us to predict the outcome of new input variables. There are many regression algorithms used in the literature to estimate task durations ranging from simple methods such as linear regression (LR) and k-nearest neighbors (kNN), to more advanced techniques and algorithms such as neural networks (NN) and Random Forests (RF). These methods, which are utilized in the present research, are briefly described below:

a) Linear Regression (LR)

A linear regression model "makes huge assumptions about structure and yields stable but possibly inaccurate predictions" [99]. In essence, the output (Y) can be calculated as a linear combination of input vectors $X^T = (X_1, X_2, \dots, X_n)$ as per Equation 2-4:

$$\hat{Y} = \beta_0 + \sum_{i=1}^{n} X_i \beta_i = X^T \beta \qquad \text{2-4}$$

where $\hat{Y}$ is a single output, β_0 is the intercept or bias of the model, and X^T denotes vector or matrix transpose (X_i is a column vector representing the features of a single example). The second part of the equation, $X^T\beta$, is a common way to represent the linear model as an inner product where the intercept β_0 is included in the vector of coefficients and the corresponding X_0 is set to the constant variable of 1. The most popular criterion to fit a LR model to a set of *n* training datapoints is the method of least squares which works by minimizing the sum of the squared errors (SSE) of predictions as shown in Equation 2-5:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \text{2-5}$$

LR is described in most standard statistical texts and in some textbooks [112–114]. There are a few other methods for fitting a multiple linear regression model including Lasso and Ridge regressions [115–117], which basically add a regularization term to Equation 2-5 in order to solve the problem of multicollinearity and to enhance the prediction accuracy and interpretability of the model. LR was employed to solve various construction management problems such as quantifying construction delays [118], predicting construction costs [119,120], and for cash flow forecasting [121].

b) k-Nearest Neighbors (kNN)

A nearest-neighbors method is an instance-based learning algorithm that "makes very mild structural assumptions: its predictions are often accurate but can be unstable" [99]. In kNN, the new instance x (the test datapoint for which we seek to compute its target value) is compared with existing instances using a distance measure, and the closest k instances to x are used to calculate its output value [90]. Euclidean distance, shown in Equation 2-6, is the most widely used metric for distance measure. The output value (Y) of the new instance is calculated by finding the k observations with x_i closest to x in input space and averaging their responses:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \qquad \text{2-6}$$

where $N_k(x)$ defines the neighborhood of x as the set of k closest datapoints x_i in the training set, and y_i is the target value for each of those neighbors. While the least-squares LR tend to have low variance and potentially high bias, the kNN method has high variance and low bias as it does not rely on any rigid assumptions about the underlying data, and can adapt to any situation [99]. kNN methods have been applied in several areas in construction such as construction project document management [122], prediction of construction cost index [123], and planning of construction technical specifications for deep foundations [124].

c) Neural Networks (NN)

Neural networks represent a large class of learning methods for regression or classification capable of modelling the nonlinear relationships between inputs and outputs, typically represented by a multilayer network diagram, as shown in Figure 2-6.

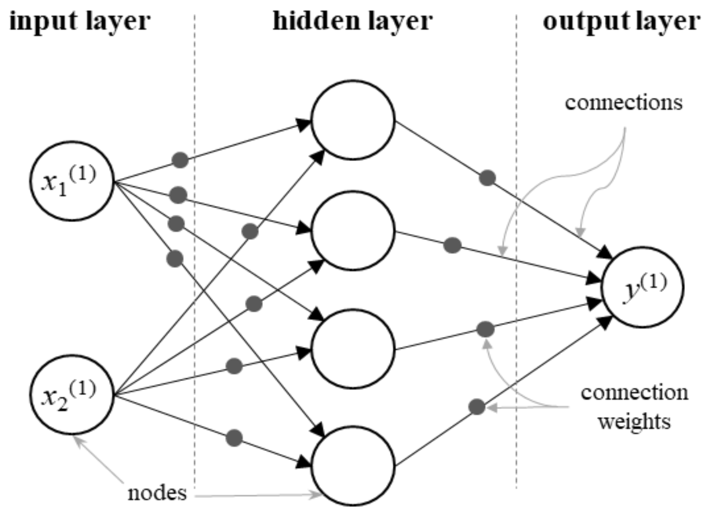


Figure 2-6. A neural network with 2 input features, 1 hidden layer, and 1 output

The derived features of the hidden layer are created from linear combinations of the inputs, and the output is created as a linear combination of these derived features. There are two aspects to develop a neural network: to learn the structure of the network, and to learn the connection weights [90]. Boussabaine [125] described the process of training a NN for regression as follows: (1) For each example in the training set, input features are fed to the NN and the specified target value is known. (2) Each hidden feature (a node in the hidden layer) is created as the weighted sum of all input features given the connection weights. (3) During this "feed-forward" step, the output node receives the values of each hidden node and calculate the weighted sum to generate a predicted result. (4) The difference between the predicted result and the target value represents the system output error. By comparing the obtained error with a specified acceptable error defined in advance, the system decides if further learning is required. If that is the case, a "back propagation" step is carried out by calculating the derivative of the squared error with respect to weights at the output node, and the results are fed back to all hidden nodes. (5) All hidden nodes calculate the weighted sum of the errors, and each, with the output node, change their weights to compensate for the corrections. Then, the "feed-forward" step is repeated again and again until an acceptable result is obtained.

Since the early 1990s, NN was investigated as an innovative management tool in all levels of construction management and engineering problems [126]. A few examples of NN applications in construction include predicting labor productivity [127], safe work behavior [128], and estimating schedule to completion [129] in construction projects.

d) Random Forests (RF)

Random forests are a significant variation of bagging, a technique for model averaging and improvement. In RF, a number of decision tree (DT) predictors are combined such that each tree is randomly and independently sampled, making them de-correlated, but with the same distribution for all trees in the forest [130].

Suppose we have a training set $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, we fit a model to it and obtain the prediction $\hat{y}$ at input x. In bagging (also known as bootstrap aggregation), this prediction is averaged over a collection of bootstrap samples, thus reducing the variance, as shown in Figure 2-7. The basic idea in bootstrap sampling is to draw datasets with replacement randomly from the training data with each sample having the same size as the original training set [99].

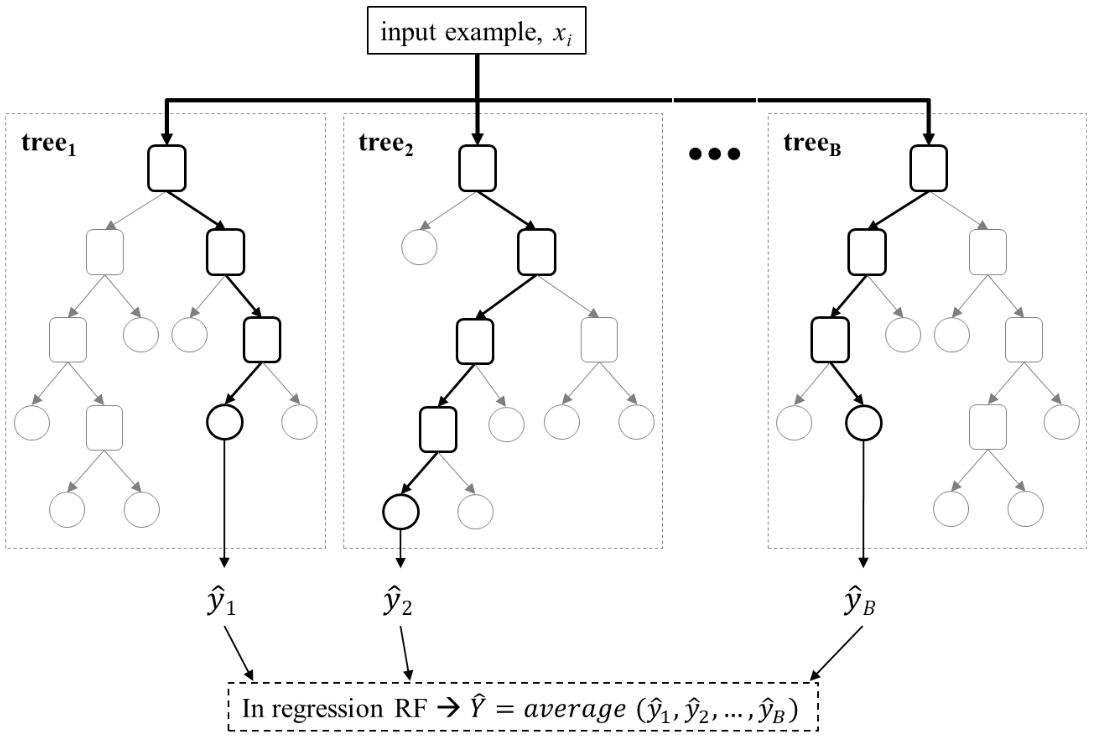

input example, x_i
tree$_1$
tree$_2$
tree$_B$
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_B$
In regression RF → $\hat{Y} = average\ (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_B)$

Figure 2-7. Schematic of constructing a random forest (RF)

A decision tree, the basic building block of RF, is based on a “divide-and-conquer” approach to the problem of learning in which a model predicts the target value through a set of rules that are arranged in a tree-like structure. Constructing a DT begins by selecting a feature to place at the root node where all observations are assigned. Observations are then split into subsets (i.e., sub-trees) at lower nodes based on satisfying a condition (e.g., $x_{ij} \leq c$). The splitting process is repeated recursively for each internal node, using only the observations that reached that node, until all observations reaching a node cannot be further split [90]. Decision nodes where the splitting processes end are called terminal nodes or leaves. In regression trees, the splitting decision at each internal node is based on calculating the SSE, Equation 2-5, between the assigned and actual values for each feature. The feature that results in the lowest SSE is selected to split the data at that internal node. DTs are easy to construct, use and interpret; however, they “have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy" [99]. Random forests can mitigate this problem by significantly improving the accuracy.

RF models have recently been gaining more interest from researchers in several areas in construction, for applications such as predicting ground surface settlement due to tunnel construction [131], improving the accuracy of BIM-enabled clash detection [132], and predicting the compressive strength of high-performance concrete [133].

It should be noted that the brief overview provided above of the regression methods investigated in the present research is presented at a high level. Each method has many variations and parameters to consider. In addition, other methods exist that are not investigated as they are either less common in practice or have some notable limitations. Moreover, when applying the above methods to the case studies described herein, further details are presented regarding the various parameters and variations of these methods.

2.3.4 Model evaluation and validation

2.3.4.1 Performance measures

In order to compare different machine learning models to each other, a systematic approach is employed to determine the performance of each model and its capability to reliably evaluate new data. There are many measures that are commonly used to evaluate the success of numeric predictions. The measures, presented here in Equations 2-7 to 2-11, are selected as per the literature analysis on performance metrics in machine learning regression algorithms by Botchkarev [134]. Given that a_i is the actual observed target value of example i, p_i is the predicted target value of example i, and n is the total number of examples in the dataset, these performance metrics include the following:

a) **Mean absolute error** (MAE) is the average of absolute individual errors. MAE is typically assessed based on the average production duration.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|a_i - p_i| \tag{2-7}$$

b) **Median absolute error** (MedAE) is the median absolute value of all errors. This measure is more robust than MAE as it is less influenced by outliers.

$$\text{MedAE} = median_{i=1,n}(|a_i - p_i|) \tag{2-8}$$

c) **Maximum absolute error** (MaxAE) is the maximum absolute error observed in the predictions of the model. It measures the extreme values and captures the extent of anomalous examples in the dataset.

$$\text{MaxAE} = max_{i=1,n}(|a_i - p_i|) \tag{2-9}$$

d) **Mean absolute percentage error** (MAPE) is typically and commonly used as a measure of quality for regression models because of its very intuitive interpretation in terms of relative error. It is also known as mean absolute percentage deviation (MAPD).

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{a_i - p_i}{a_i}\right| \tag{2-10}$$

e) **Correlation coefficient** (CC) measures the statistical correlation between the actual and predicted values. It ranges from 1 for perfectly correlated values, to 0 when there is no correlation, to -1 for perfectly adversely correlated values. CC differs from the other measures by being scale independent in that if all the predictions are multiplied by a constant factor and the actual values are left unchanged, this measure is unchanged.

$$\text{CC} = \frac{n\sum a_i p_i - \sum a_i \sum p_i}{\sqrt{n\sum a_i^2 - \left(\sum a_i\right)^2}\sqrt{n\sum p_i^2 - \left(\sum p_i\right)^2} \qquad 2\text{-}11$$

It should be noted that in addition to the correlation coefficient, the coefficient of determination (R^2) is a common accuracy measure for statistical models. However, in the present study CC is used rather than R^2 as the latter is more relevant when describing linear models. The coefficient of determination basically assesses the improvement in accuracy of the linear model over the mean value. CC is more relevant to assess the correlation of machine learning models in general as it considers the actual and predicted values of each record in the dataset. Moreover, CC has been widely used in many data mining textbooks and previous predictive ML publications as compared to using the R^2 as a measure to assess the model accuracy.

For MAE, MedAE, MaxAE, and MAPE, the lower the value is for each of these performance measures, the better the results. As for CC, higher positive values are more favorable.

2.3.4.2 Model validation

Model validation is an essential component when developing ML models, without which the model result cannot be deemed reliable for decision-making purposes. Most of the model validation techniques are based on analyzing the residuals for homogeneity, stationarity, independence, or normality [135]. In machine learning, model validation is the process where a trained model is evaluated with a testing dataset. The testing set is a portion of the whole dataset and is different and independent from the training set. The purpose here is to test the predictive ability of the model to generalize to new unseen examples [99]. It is assumed that both the training and the testing sets are representative samples of the underlying problem. There are many techniques for ML model evaluation including train-test split and k-fold cross validation [90]. In **train-test split**, the dataset is randomly split into two parts, one used for creating the model (i.e., a training set) and another for testing (i.e., a testing set) to understand what would happen if the model encounters data it has not seen before. The main shortfall of this method is what is typically referred to as a **sampling bias** where the two sets (training and testing) might, by random chance, not be representative samples of the underlying features of the data. Training sets typically represent 60–90% of the dataset and the remaining is used for testing. The dilemma with this technique is that in order to learn a good model, we want to use as much of the data for training as possible, but that would leave a testing set that may not be sufficient to provide a reliable estimate of prediction errors; this is the problem that cross-validation (CV) attempts to solve.

Typically, learning schemes involve two stages, one to train the model and the second to optimize its parameters (i.e., the hyperparameters as describe in Section 2.3.2), and separate sets of data may be needed in the two stages. The best approach to achieve this and to ensure that the test data is not used in any way in the model creation, is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to tune the model's parameters by minimizing prediction errors; and the test set is used to assess the generalization error of the final selected model [90,99].

In **k-fold cross-validation**, the dataset is split into **K** roughly equal-size parts. We fit the model using K-1 parts and calculate the prediction error of the fitted model using the remaining *k*th part. This process is repeated **K** times each time using a different part for testing. Then, the **K** estimates of the prediction error are combined and averaged. This technique is used to reduce the variability introduced by sampling training and testing subsets. The number of repetitions represents the number of folds of the method, and according to Witten et. al (2011), it has been shown that "10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up" [90]. To overcome some challenges related to small dataset, or overfitting, variations to k-fold CV are used including Leave-one-out Cross-Validation (LOOCV) and Nested-CV. A more sophisticated variation of CV and one that is more relevant to the present research is time series cross-validation (TSCV). In TSCV, there are a series of test sets, each consisting of a single example and the corresponding training set consists only of examples that occurred prior to the example that represents the test set [136]. It is important that all training data happens before test data as training should not contain information from the future. There is a temporal dependency between observations that must be preserved when selecting the training and testing sets. This represents a typical scenario when predicting the production time of a "future" product in a fabrication shop (e.g., wall panel in IBC) based on the historical data of only products that were manufactured prior to it.

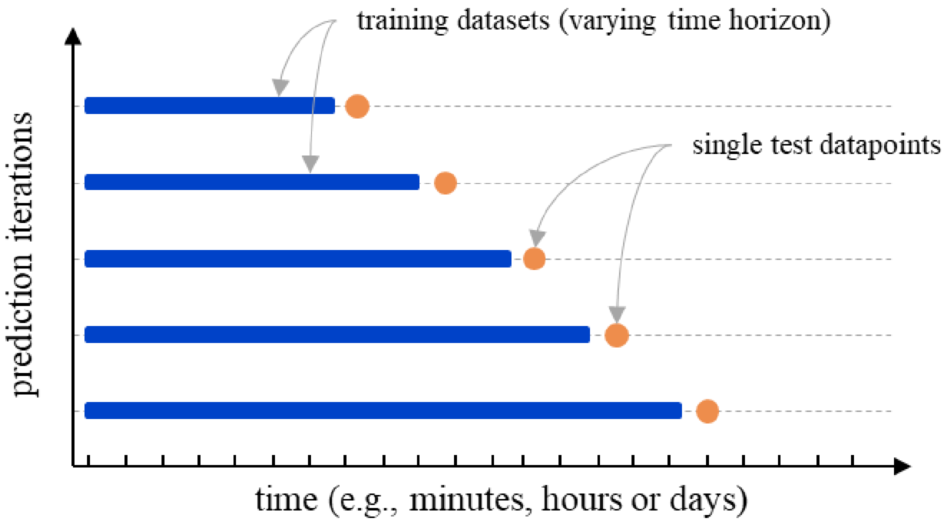

training datasets (varying time horizon)
single test datapoints
prediction iterations
time (e.g., minutes, hours or days)

Figure 2-8. Schematic of time series cross validation (TSCV)

Figure 2-8 shows a schematic of how TSCV works wherein the blue line represents a time period in the past from which the training examples are used, and the orange dot represents the single test datapoint. After the raw data are checked for uniformity and consistency as part of the data preprocessing phase, ML models are tested using actual production duration data using TSCV.

2.4 Industrial Applications of KDD

2.4.1 Knowledge discovery in databases in construction

Along with significant support through research and case studies, the application of data-driven decision making and the related data science techniques contribute substantially to supporting organizations to make better decisions, which results in a stronger, more feasible business [137,138]. Substantial improvements in strategies and policies, more informed decisions, minimized risks, and discovering hidden valuable insights are some of the potential gains from using data-driven approaches, especially in complex contexts such as the construction industry [139], and IBC in particular. In this regard, the use of automated data acquisition systems that integrate data from different sources and employ data analytics has been extensively researched in the construction literature for the purpose of project monitoring and control [140,141].

Several studies have investigated the application of KDD in the construction industry. For example, Soibelman and Kim [86] proposed a KDD process to identify the causes of delays in drainage pipeline installation, focusing largely on data preparation, while Kim et al. [142] used data mining to identify the key factors that significantly contribute to delays in an on-going construction project. Also, Hammad [89] proposed a KDD framework to estimate future resource requirements in multi-project industrial construction. A customized decision tree was developed to analyze labor productivity and estimate the labor requirements for masonry brickwork [143]. Moreover, a framework is proposed to forecast construction project time and cost using support vector regression methods [144].

A detailed literature review that highlighted the current applications and future potential of data analytics in construction affirmed that the adoption of these technologies in construction lags its wider application in other fields [39]. More recently, various machine learning algorithms are used to detect non-certified work on construction sites using deep learning algorithms [145], to classify workers encountering high fatality risk accidents at construction sites [146], and to verify the personal protective equipment compliance of construction workers [147,148]. A hybrid model of light and natural gradient boosting algorithms is proposed to predict the construction cost during the early stages of the project [149]. It is evident, from the literature, that the application of machine learning in many areas of the construction industry is gaining increasing acceptance and adoption; however, these techniques have seen limited applications in project planning and scheduling in industrialized building construction.

2.4.2 Manufacturing cycle time predictions using KDD

An overview of datamining applications in manufacturing was previously conducted and indicated that future research will be focused on the areas of design, shop floor control, supply chain management, as well as developing solutions that can be easily integrated with existing systems [150]. Similarly, Choudhary et al. [151] reviewed the literature about KDD and DM in manufacturing, focusing on categorizing the articles into the five major functions of DM, namely, description, association, classification, prediction, and clustering. In recent years, there have been a few studies focusing on predicting LT and CT using machine learning approaches in manufacturing environments other than IBC. For example, Öztürk et al. [152] used a regression tree to estimate the lead time in a make-to-order manufacturing facility, while Backus et al. [153] compared clustering, k-nearest neighbors, and regression trees to predict cycle time in a semiconductor factory and concluded that a hybrid method of clustering and regression tree provide superior results. Another study in the context of a wafer fabrication factory estimated a range for a job cycle time based on a fuzzy data mining approach using a fuzzy backpropagation network, a principal component analysis, and fuzzy c-means [154]. Another study used genetic programming to predict process cycle time based on system status information [155]. More recently, Gyulai et al. [156] compared analytical and machine learning techniques for lead time prediction in the optics industry. They concluded that random forests might be the most accurate model; however, linear methods provide acceptable accuracy and are simpler to interpret. Also, Lingitz et al. [157] compared the accuracy of several regression algorithms when predicting lead time in semiconductor manufacturing and suggested that random forests provide the highest accuracy in terms of lower normalized root-mean-square error.

The present study builds upon this previous research to achieve accurate prediction of cycle time in manufacturing environments using machine learning models; the focus of this study is on the application of ML in the industrialized building construction industry, which differs from other manufacturing environments as described above. Moreover, previous studies applying machine learning to predict cycle time have mainly used simulated data, whereas, in the present study, the data is collected from real-time data acquisition systems (e.g., RFID). As such, the data collected is more prone to noise compared to simulated data, and, hence, is more challenging to manipulate and prepare for modelling. Actual production data provides a more accurate and valid representation of the production system.

2.5 Summary and Research Gaps

Construction 4.0 is emerging as a technologically revolutionary paradigm for tackling the dynamic and uncertain nature of construction operations for improved system predictability. Industrialized building construction provides an adequate and enabling environment to implement machine learning modelling, one of the many aspects of this paradigm. Reviewing the relevant literature covers four essential topics: (1) industrialized building construction industry and the Construction 4.0; (2) production planning focusing mainly on flow shop scheduling; (3) data mining and machine learning techniques, focusing on the methods used in the present research; and (4) the applications of ML in the construction industry and the related prediction of production time in manufacturing environments. The literature review highlights several research gaps, some of which are investigated in the research described in this thesis. Specifically, the research gaps motivating the research presented in this thesis are described below:

- Although IBC provides an environment that is suitable for the application of the latest technological advances brought about by Construction 4.0, the adoption of such technologies lags other manufacturing sectors.
- The IBC industry can benefit from ML techniques to improve production planning and scheduling within the fabrication shop by accurately and dynamically estimating the cycle time of products. This approach has been applied in similar settings, but the particularities due to the highly customized nature of production in IBC has not been investigated.
- Many machine learning methods can be used for predicting the production time; however, it is clear from the literature that tree-based methods, specifically RF, provide satisfactory results in terms of a balance between accuracy and interpretability. RF models have not been yet applied in industrialized building construction manufacturing settings.

Chapter 3 Research Methodology

3.1 Proposed Framework

The goal of the proposed framework is to enable the integration of automatic data acquisition, with accurate depiction of loading conditions of the job shop, and production scheduling of building components (i.e., production units) in an industrialized building construction manufacturing environment. Based on the reviewed literature, existing frameworks and methods can be improved for such purposes by introducing novel principles of the Industry 4.0 revolution. Focusing on IBC manufacturing environments and introducing a novel approach to capture real-time shop utilization, a generic framework for the accurate prediction of production time is proposed to enable the inclusion of production managers' experience and knowledge, by way of historical production data, in the decision-making process. The proposed framework is expected to improve the production scheduling and cycle time prediction by accurately capturing the dynamicity of the job shop in industrialized building construction. The proposed framework is illustrated in Figure 3.1 and is developed based on frameworks that appear frequently in the literature on data mining. Such frameworks include the cross-industry standard process model for data mining (CRISP-DM) [158], Oracle's architecture development process [159], the work published by Fayyad et al. [84], Davenport et al. [160], Dietrich et al. [161], and the IBM foundational methodology for data science [162].

Product composition (i.e., physical characteristics of building components or modules) and the production workflow information of these products as they move through the job shop are the main inputs of the framework. Developed based on the specific manufacturing policies at a particular IBC facility, on domain knowledge, and on similar manufacturing environments from the literature, the proposed framework can be applied through five main phases, which are described as follows.

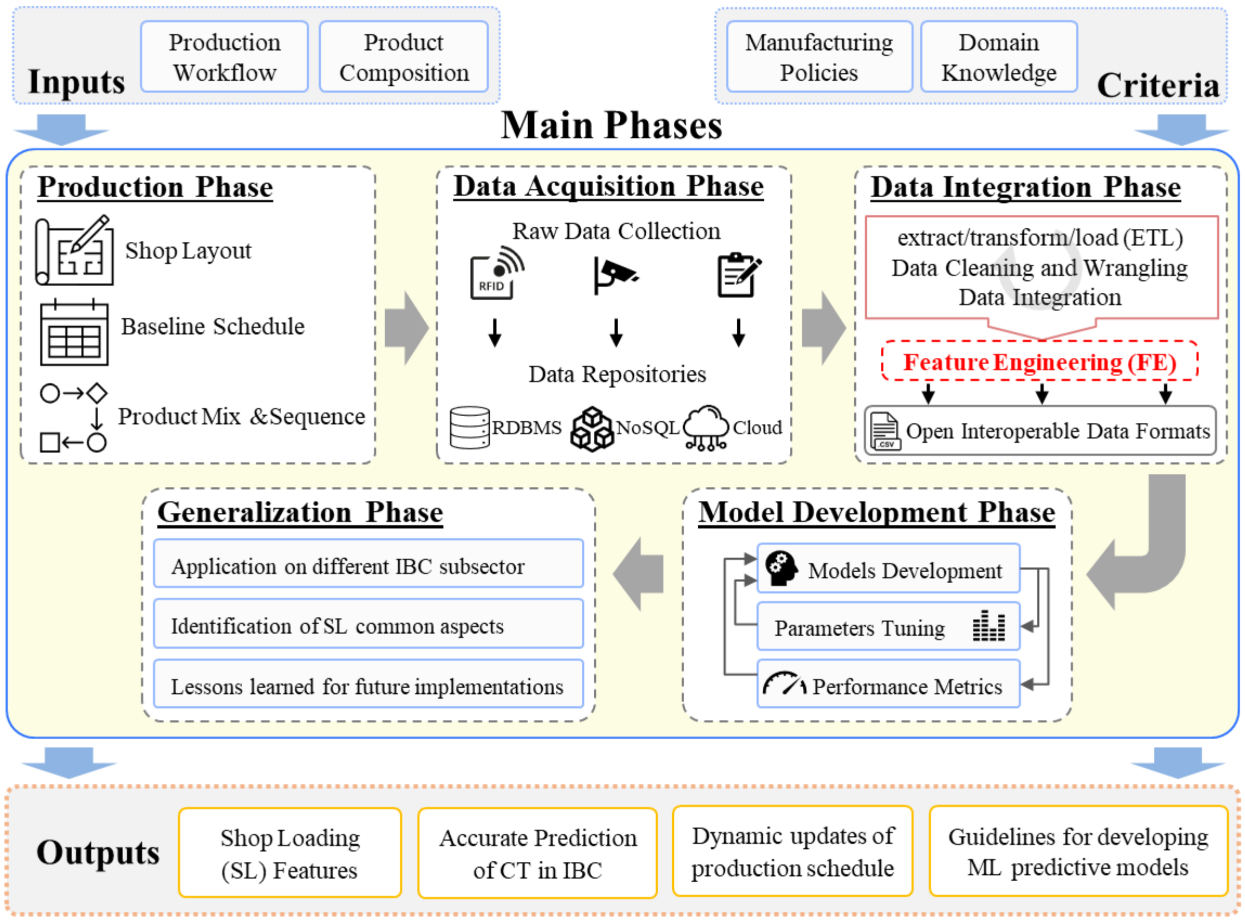


Figure 3-1. Proposed framework for production time prediction in IBC environment

In the **production phase**, the purpose is to develop an understanding of the actual production environment by investigating the layout of the job shop, product mixture and sequencing, and the baseline schedule. Accurate descriptions of the manufacturing environment result in an efficient collection of relevant data that will be used to build the predictive models. The **data acquisition phase** is composed of two parts: data collection techniques, and data storage media. Job workflow data can be collected using various methods ranging from manual data collection to automatic real-time systems such as RFID and online video streaming. The purpose of this phase is to systematically collect the relevant data about the flow of each building component along the production line. Such data are gathered as part of daily operations and can be 'big' in terms of size and speed of collection. The collected data can then be stored in a variety of data repositories either on-premises or online using structured or unstructured formats (e.g., RDBMS or NoSQL) or a mixture of both. The **data integration phase** involves all manipulation processes of raw data to transform it into interoperable formats that are suitable for data-driven modelling. This phase includes the extract/transform/load (ETL) process of transferring raw data from data sources to a data warehouse, data cleaning and wrangling, integration of raw data from different sources, and the process of feature engineering. These processes might be carried out in a different order and one can iterate between them until obtaining the tidy dataset. The feature engineering step is of particular importance as it enables the development of new features that capture real-time loading conditions of the job shop (i.e., current system utilization). In the **model development phase**, machine learning models are developed using the tidy dataset obtained from the data integration phase. The hyperparameters of the developed models are then tuned and the performance of the model is assessed using the various performance metrics. This process is iterative as the hyperparameters are further tuned and adjusted until certain performance criteria is achieved. If more than one model is developed, statistical significance testing is applied to examine the difference between these models. The outcome of this phase is a verified and validated model that can be implemented on the actual production system. The **generalization phase** includes three main processes. First, generalization is achieved via sharing the knowledge discovered by applying the framework processes to a different subsector of industrialized building construction that shares various aspects in terms of process sequencing, the high level of product customization, and the perceived dependency on capturing real-time loading conditions of the job shop. Second, conclusions can be drawn about the nature of shop loading features, their common characteristics, and how to derive them at the different IBC manufacturing environments. Lastly, lessons learned are documented highlighting the opportunities and challenges in acquiring, processing, and integrating data.

With respect to the developed framework, the research methods used in the present research are described in the following section as they apply to the two case studies, which are described briefly in Sections 1.2.1 and 1.2.2. The research methodology is first positioned within the construction engineering and management field as design science research.

3.2 Research Methods

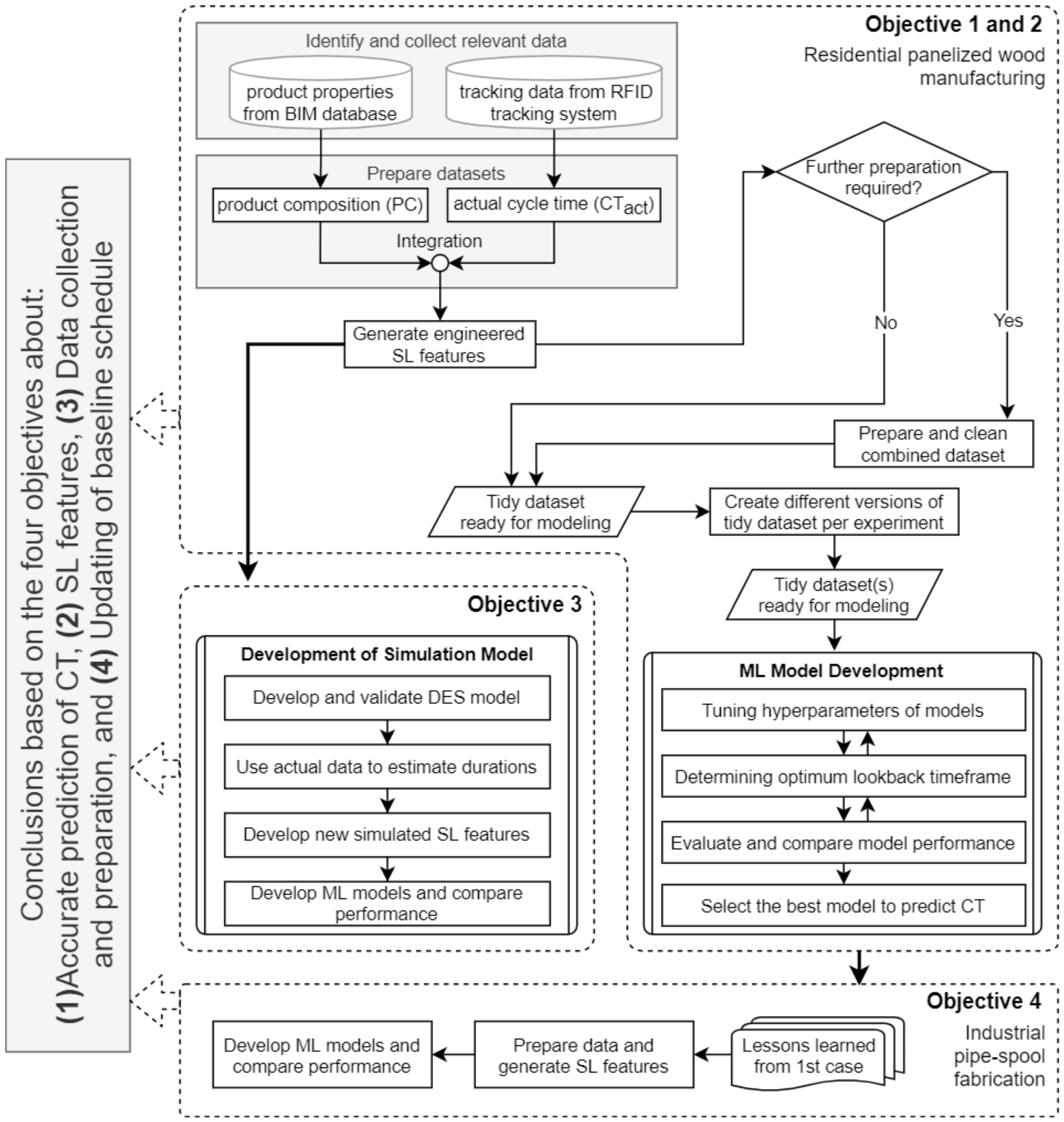
The research paradigms under which construction management is categorized has long been examined and argued [163]. More recently, however, construction management is strongly contended to be repositioned as a design science rather than as an explanatory science (i.e., social or natural sciences) [164]. Such an approach will better connect research and practice, and thus strengthen the relevance of academic construction management. Design science research (DSR), or constructive research, is the research methodological approach adopted in this research to develop the proposed framework. The main focus of DSR for solving problems is the development of a new artifact: a solution-oriented knowledge tool that professionals can use to design solutions for their field problems [165]. Design science research has been proven to be a suitable research approach in construction management, especially when developing artifacts to solve problems, implementing those solutions in the construction domain, and bridging the gap between academia and industry [166–168].

The rigorous procedure of developing an artifact starts by identifying gaps in the literature, then iteratively developing and evaluating the artifact in a reproducible manner, and lastly, communicating the outcomes or solutions in a clear and concise manner. Hevner et al. [169] described seven guidelines for a DSR as follows: 1) design a viable artifact in the form of a model, a method, or a construct; 2) develop solutions that are relevant to the problem; 3) rigorously evaluate the utility, quality, and efficacy of the design artifact; 4) provide clear and verifiable research contributions; 5) use rigorous methods to support both the construction and evaluation of the artifact; 6) provide the necessary means to reach the desired solutions while satisfying constraints of the problem environment; and 7) communicate the research effectively to both technology-oriented and management-oriented audiences.

For the research objectives to be achieved in this thesis, the DSR methodology is followed. The research methods employed to improve production planning and scheduling in IBC manufacturing environments is based on the developed framework and is represented by the workflow of activities shown in Figure 3-2.

Objectives 1 and 2:

The artifact developed to achieve the first and second objectives presented in this thesis consists of a data-driven predictive model containing the information extracted from a BIM model and an automated RFID production tracking system and attempts to mine the mapping relationship between features, which represent product properties and shop utilization, and the CT using historical data from a wall panel fabrication shop. After identifying and collecting data, the product composition properties (PC) obtained from the BIM models and the corresponding actual cycle time for each product (CT_{act}) are cleaned, prepared, and then combined. Essentially, in this phase (i.e., Data Integration phase in Figure 3-1), the following two questions: Is the collected data representative of the problem to be solved? What efficient cleaning steps should be followed so that the data is represented in a format suitable for prediction models?



Objective 1 and 2
Residential panelized wood manufacturing
Identify and collect relevant data
product properties from BIM database
tracking data from RFID tracking system
Prepare datasets
product composition (PC)
actual cycle time (CT_act)
Integration
Generate engineered SL features
Further preparation required?
No
Yes
Prepare and clean combined dataset
Tidy dataset ready for modeling
Create different versions of tidy dataset per experiment
Tidy dataset(s) ready for modeling
ML Model Development
Tuning hyperparameters of models
Determining optimum lookback timeframe
Evaluate and compare model performance
Select the best model to predict CT
Objective 3
Development of Simulation Model
Develop and validate DES model
Use actual data to estimate durations
Develop new simulated SL features
Develop ML models and compare performance
Objective 4
Industrial pipe-spool fabrication
Lessons learned from 1st case
Prepare data and generate SL features
Develop ML models and compare performance
Conclusions based on the four objectives about:
(1)Accurate prediction of CT, (2) SL features, (3) Data collection and preparation, and (4) Updating of baseline schedule

Figure 3-2. Workflow of the proposed research methods

Next, the development of engineered features is carried out using the RFID tracking data to reflect the real-time loading conditions of the job shop (i.e., shop loading (SL) features). These shop loading features are then combined with existing PC features and are further cleaned and prepared to eliminate anomalous records. The outcome of this iterative activity is the "tidy dataset". To determine the CT, the sets of features, PC and SL, are quantified, and therefore the predicted value of cycle time, CT_{pred}, can be formulated as shown in Equation 3-1:

$$CT_{pred} = f(PC, SL) \tag{3-1}$$

In the case of wall panels, different versions of the tidy dataset are developed for the purpose of experimenting with different aspects of data aggregation and integration. Each of these versions are then fed into the next phase, the ML model development, which involves three iterative processes: a) tuning the hyperparameters of the various machine learning models, b) determining the optimum lookback timeframe used to train the models, and c) evaluation and comparison of model performances. The best model to be used for implementation is then selected to predict the production cycle time. Model evaluation has two main phases: the diagnostic phase, which is used to ensure the model is working as intended; and the statistical testing phase, which is applied to ensure that the data is being properly handled and interpreted within the model. The output of this phase of the study satisfies the first and second research objectives and is used to support the third and fourth objectives.

Objective 3:

The artifact developed to achieve the third objective described in this thesis consists of the development of a discrete-event simulation model with the aim of examining the effect of generating engineered features that reflect the real-time loading conditions of the job shop (i.e., SL features). These features are similar to the features generated during ML model development. However, here we examine a few SL features that are not practically collected, and in most cases need to be custom-built. The goal of this objective is to investigate how SL features, that are proposed to improve the predictive accuracy of ML models, can be captured in practice. By using simulation modelling, the collection of such features in the job shop can be examined. Different sets of SL features, generated from the simulation model, augment the raw dataset to generate variations upon which ML models are developed. ML model performances are compared using the same measures as are applied in Objectives 1 and 2. In conclusion, the simulation model can be used to propose specific mechanisms for collecting certain types of data that are otherwise not accounted for during the normal daily operations of the factory. Proposed collection of data would enhance the development of ML predictive models and would make data preparation and feature generation more consistent and automated and less prone to subjectivity.

Objective 4:

The artifact developed to achieve the fourth objective in this thesis consists of a data-driven predictive model similar to the one developed for the first and second objectives. However, the application is conducted at an industrial pipe-spool fabrication shop with a different time horizon to be predicted, and different shop settings and manufacturing policies. The goal of this objective is to examine the generality of the results obtained during the first case study. IBC manufacturing environments share some similarities with a pipe-spool fabrication shop, namely, the highly customizable construction products. In any manufacturing facility, the cycle time is greatly influenced by shop utilization. Although the details of the engineered SL features will depend on the specific case study and one might need to custom-build these features, due to the similarity of industrialized building construction environments, these features should capture some aspect of shop loading to improve the predictability of the cycle time.

Chapter 4 Residential Wall Panel Fabrication Shop

This chapter describes the first case study where the proposed framework and methodology are applied to predict the production cycle time in a wall panel fabrication shop using several machine learning models. The chapter is divided into five main sections. First, the factory layout and the manufacturing process are described. Then, data acquisition and descriptive analysis is presented, followed by a detailed explanation of data cleaning and preparation tasks. The fourth section is dedicated to the development of engineered features that capture real-time loading conditions of the job shop. Lastly, the development, evaluation, and comparison of machine learning models are described in detail.

4.1 Factory Layout and Process Description

The production of prefabricated wall panels at an IBC facility located in Edmonton, Alberta, Canada is analyzed.

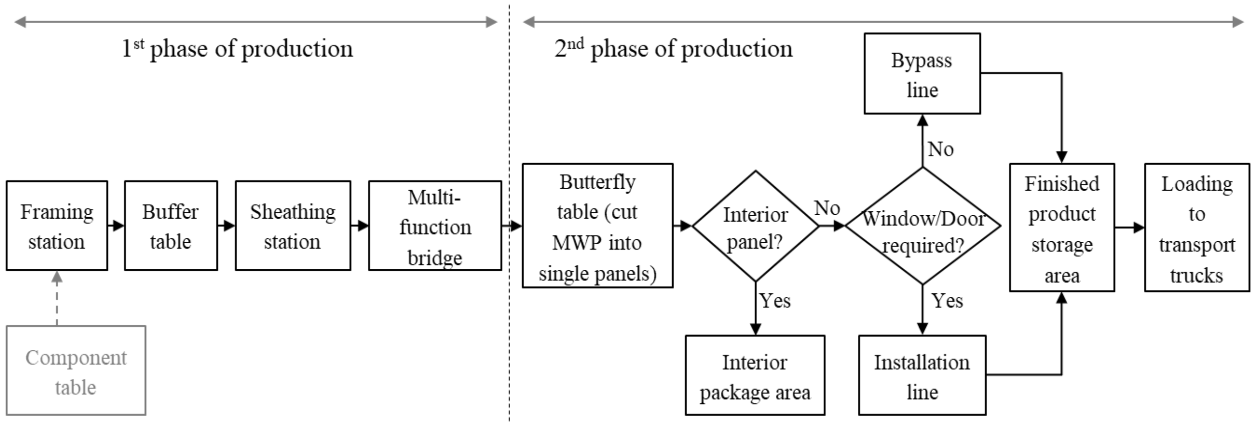


Figure 4-1. Workflow of wall production line

The facility produces wood-framed open-wall panels (i.e., with no electrical or plumbing fixtures), floor panels, and roof panels that are then transported to the construction sites for assembly. The facility is equipped with state-of-the-art computer numerical control (CNC) production lines capable of producing building framing components (i.e., wall, floors, roofs, and stairs) using different processes. The research effort described in this thesis focuses on the wall production process, which involves framing, sheathing, window and door installation, and loading operations as shown schematically in Figure 4-1.

The shop floor layout is shown in Figure 4-2, where the scope of the present study is restricted to the first phase of production (i.e., multiwall panel (MWP) production route), which consists of four consecutive workstations: framing station (FS), buffer table (BT), sheathing station (SS), and multifunction bridge (MFB). A wall panel goes through these stations in sequence, and no two panels are processed at one station at the same time. RFID antennas are located between consecutive stations where timestamps are recorded by reading RFID tags attached to wall panels as they pass by the antennas.

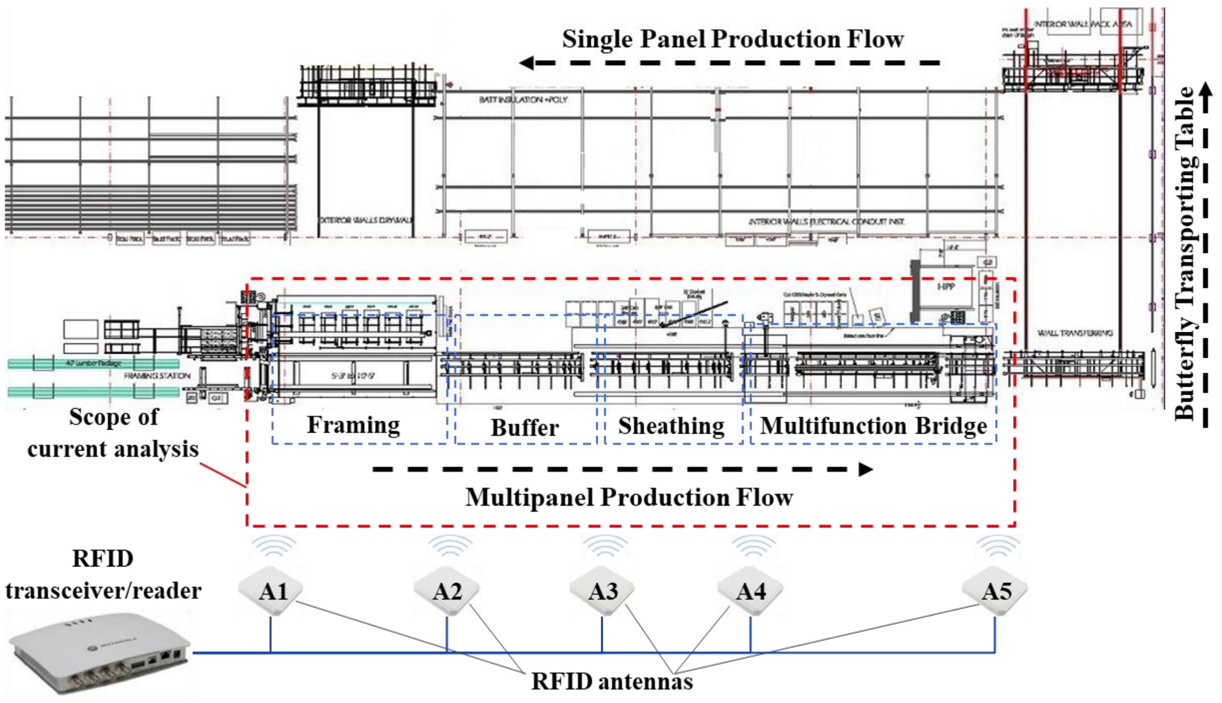

Single Panel Production Flow
Butterfly Transporting Table
Scope of current analysis
Framing
Buffer
Sheathing
Multifunction Bridge
Multipanel Production Flow
RFID transceiver/reader
A1
A2
A3
A4
A5
RFID antennas

Figure 4-2. Shop floor layout and location of RFID antennas between workstations

Wall panel production starts at the framing station, Figure 4-3, where workers load the studs (e.g., regular, double, or L-shaped) into the machine in sequence, after which the machine nails the studs to the top and bottom plates of the exterior and interior wall panels. Components such as windows and doors are built in the component table station and fed to the framing station. To maximize the utilization of the CNC machine, most MWPs are generated by combining several single-wall panels together up to 40 ft in length, which is the maximum length of the CNC table. In order to combine single-wall panels into a multiwall panel, the single-wall panels must all have identical heights and thicknesses. This is a mathematical optimization problem that was extensively described and investigated in previous research [60].

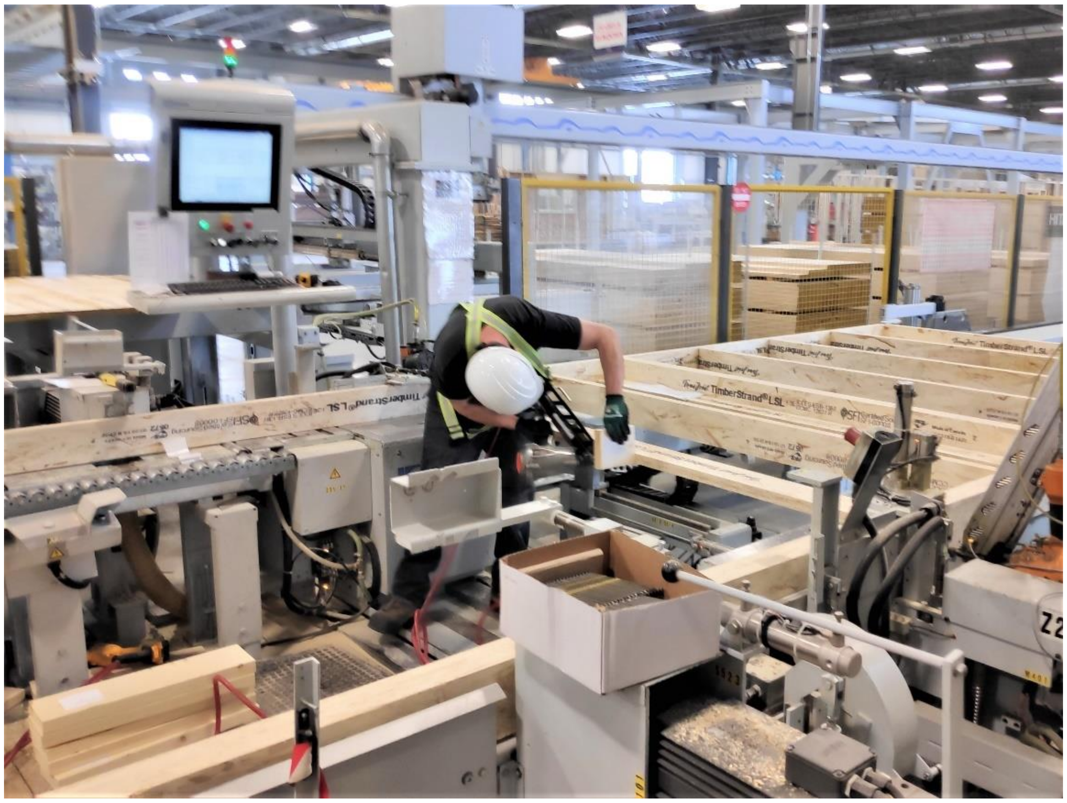


Figure 4-3. Framing station (installation of studs between top and bottom plates)

From the framing station, a MWP moves to the buffer table where one or more workers perform several manual tasks, including correcting errors that result from the framing machine, installing backing and other support material, and marking the wall panel name. The MWP then moves to the sheathing station, shown in Figure 4-4, where sheathing boards for exterior wall panels are placed, correctly positioned, and manually nailed. Then, the wall panel is moved to the multi-function bridge (MFB) for machine nailing. At the MFB, shown in Figure 4-5, the sheathing boards are automatically, and thoroughly fastened to the studs of the wall panel by a CNC machine using nail guns that are mounted on the moving bridge. The sheathing boards are only installed on exterior walls. The MFB process only requires one worker to (i) bring the panel in from the sheathing station, (ii) initiate the nailing process, and (iii) move the MWP out.

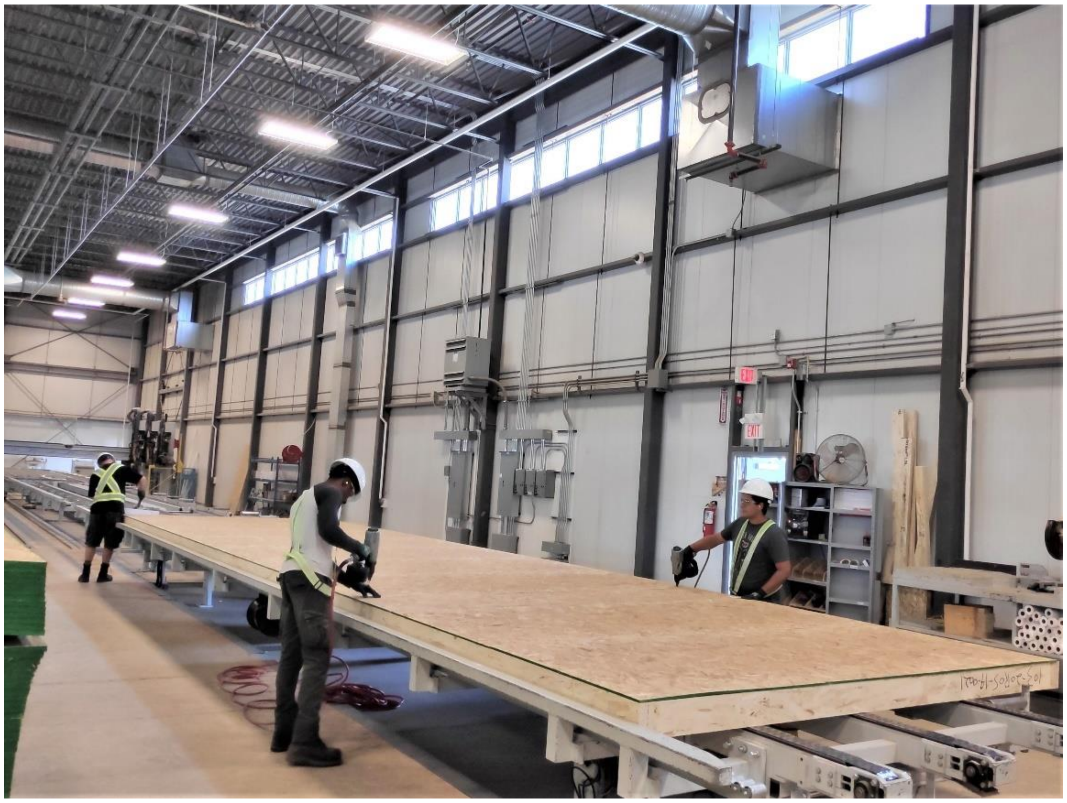


Figure 4-4. Sheathing station (manual nailing of sheets for exterior walls)

The floor layout is configured such that each MWP is framed, prepped, sheathed, and nailed between antenna locations A1 to A5. Then, each MWP is cut into several single panels and transferred to a butterfly table, where antenna A6 records when the single wall panel starts the next phase of production. In this study, the analysis focuses on the first phase of production: the MWP production starting at antenna A1 (located at the start of production at the FS) and ending at antenna A5 (located at the end of the MFB).

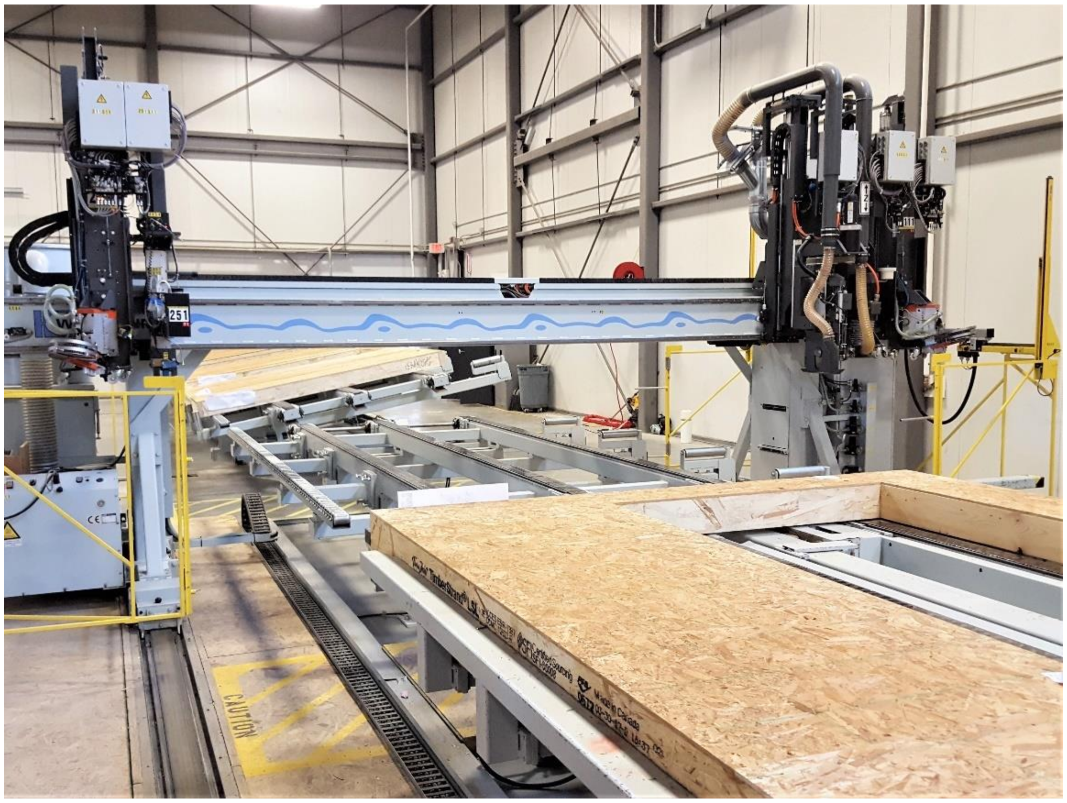


Figure 4-5. Multifunction bridge (at initial position when receiving a wall panel)

4.2 Data Acquisition and Description

The historical raw data analyzed in this study covers a period of 42 months between February 2015 and August 2018. The raw data is composed primarily of two datasets. The first dataset (DF1) is the “RFID Readings” dataset that contains timestamps for each MWP along the production route. It consists of 416,948 records and 10 attributes. The second dataset (DF2) is the “Multipanel” dataset that contains descriptive attributes of each MWP including the physical properties of the panels. It consists of 39,703 records and 37 attributes. Figure 4-6 shows a flowchart of the initial data identification and data cleaning steps. Initial data identification is performed to include only the information relevant to the present analysis. These steps depend largely on the domain knowledge and the specific application. For example, in DF1, the steps include a) removing records before September 11, 2015, since the factory layout and RFID tracking system setup was changed, and b) only keeping records corresponding to antennas A1 through A5 as these correspond to the production of MWPs. As for DF2, the steps include a) removing panels that are manually manufactured, and d) excluding any attribute where all data are missing or there is only one dominant value.

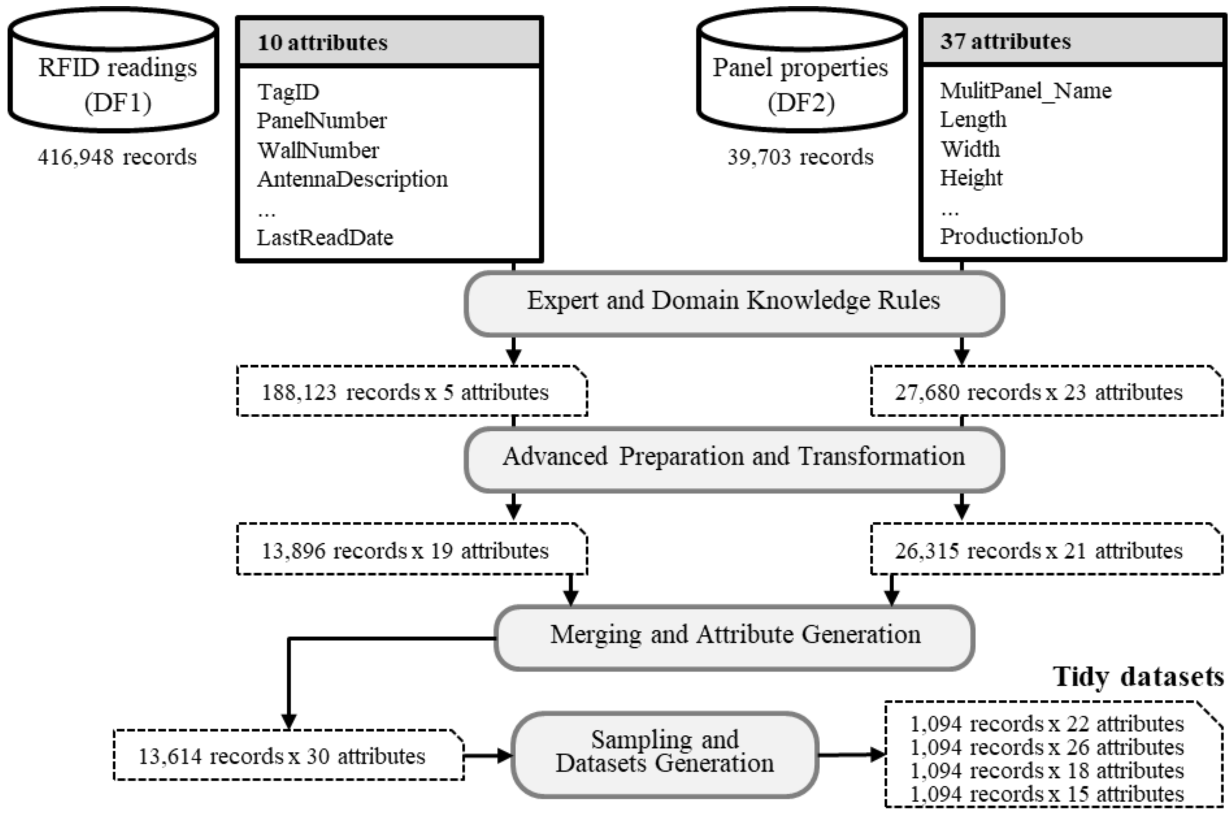

RFID readings (DF1)
416,948 records
10 attributes
TagID
PanelNumber
WallNumber
AntennaDescription
...
LastReadDate
Panel properties (DF2)
39,703 records
37 attributes
MulitPanel_Name
Length
Width
Height
...
ProductionJob
Expert and Domain Knowledge Rules
188,123 records x 5 attributes
27,680 records x 23 attributes
Advanced Preparation and Transformation
13,896 records x 19 attributes
26,315 records x 21 attributes
Merging and Attribute Generation
Tidy datasets
13,614 records x 30 attributes
Sampling and Datasets Generation
1,094 records x 22 attributes
1,094 records x 26 attributes
1,094 records x 18 attributes
1,094 records x 15 attributes

Figure 4-6. Flowchart of initial data identification and data preprocessing steps

In addition, as part of this phase of the methodology, descriptive statistics and data visualization are applied to the datasets to assess the content and quality, and to gain initial insights into the data. Figure 4-7 shows the total number of RFID readings per month, per week, and using a 14-

day rolling window. Periods during which there are a low number of readings, such as from June 2016 to October 2016, might indicate low productivity periods due to market recessions. Such periods represent abnormal operating conditions and are excluded from the analysis as they may skew the results.

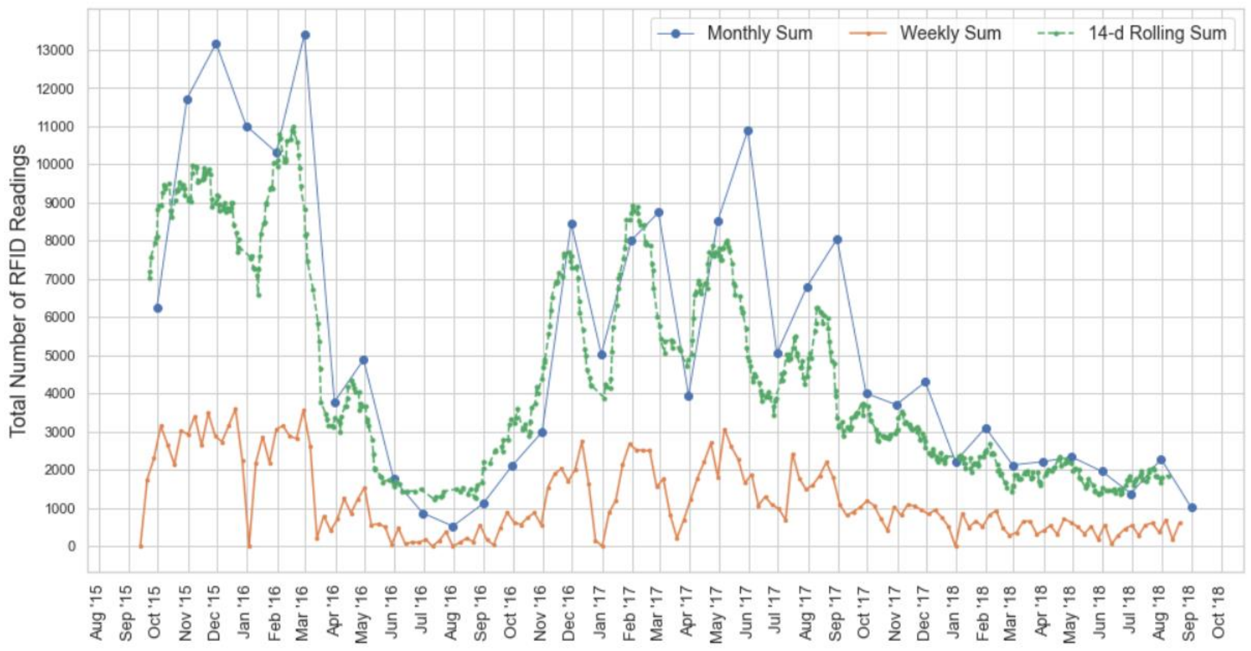


Figure 4-7. Number of RFID readings per month, per week, and per 14-days rolling window

However, a more consistent indicator of daily productivity is to count the number of panels produced each day within prescribed ranges of production over the study period. For example, there were approximately 1,400 panels manufactured when the daily production rate was between 35 and 40 panels/day, which is close to the average daily production, as shown in Figure 4-8. Working days are excluded if the total daily production is less than 5 or more than 75 panels/day as production beyond these limits represents abnormal operating conditions. Those out-of-bound working days cover approximately 3% of the total study period. The purpose of this exercise is to determine which production days represent anomalous operations and exclude the records during these days from further analysis. It is worth noting that most of the panels are produced when the daily production rate is between 30 to 55 panels/day, as shown in Figure 4-8.

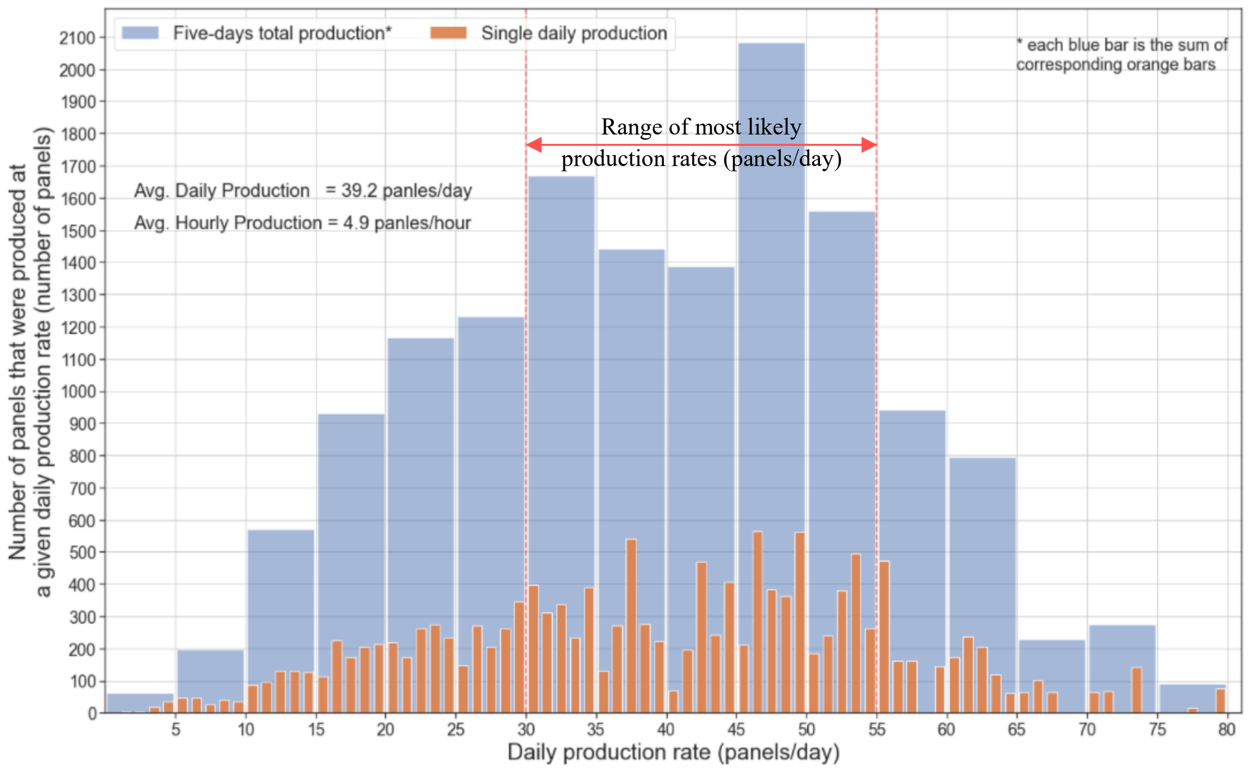


Figure 4-8. Total panel counts for each daily production rate

Furthermore, Table 4-1 shows a summary of the statistics for the attributes in DF2. Here the dimensions for MWP length, width, and height are in millimeters, while other numbers represent the count of the corresponding panel property (e.g., “Window” is the number of normal-sized windows that the MWP has). By examining the results in Table 4-1, a few observations can be noted to assist in better understanding the data and determining how different features might affect the predictability of the cycle time.

It can be noted that some attributes have extreme values compared to the mean and median. For example, “NailCount”, which represents the total number of nails used during the production of a single MWP, has an extreme maximum value of 2,466 which is considered an outlier. It can also be noted that the attribute ‘LargeDoors’ is expected to have minimal influence on the predictability of CT as most of the panels have no large doors installed. This can be inferred from the value of zero for the mean, standard deviation, and interquartile range.

Table 4-1. Statistics summary of attributes in DF2 dataset

	Description	mean	std	min	25%	50%	75%	max
Length	Panel length	8999	3315	1203	6087	10404	11878	12338
Width	Panel height	2525	158	1607	2467	2467	2505	3235
Height	Panel thickness	117	29	89	89	140	140	314
Window	No. of regular windows	0	1	0	0	0	0	9
LargeWindow	No. of large windows	0	1	0	0	0	0	5
Door	No. regular doors	1	1	0	0	0	1	10
LargeDoor	No. garage doors	0	0	0	0	0	0	3
Sheetfull	No. of full sheets for ext. panels	1	2	0	0	0	0	17
SheetPartial	No. of partial sheets for ext. panels	3	4	0	0	0	4	31
Cutzone	Cuts along MWP to single walls	2	2	0	0	1	3	17
Drillhole	No. of drills used to lift panels	4	3	0	2	4	6	26
Stud	No. of regular studs	15	7	0	9	15	21	215
DStud	No. of double studs	0	0	0	0	0	0	6
LStud	No. of L-shaped studs	2	2	0	0	1	3	18
MStud	No. of multi-studs	0	1	0	0	0	0	14
Block	No. of lumber blocks	1	3	0	0	0	0	42
Backing	No. of lumber backings	2	4	0	0	0	3	50
NailCount	No. of nails used to install sheets	327	401	0	0	190	566	2466
Nailline	Lines along which nails are installed	33	37	0	0	20	63	447

4.3 Data Preprocessing

4.3.1 Main preparation tasks

Data preprocessing is an essential step in the KDD framework to improve the quality and consistency of the data itself, and, consequently, the quality of the data mining results. Data preprocessing techniques include cleaning, integration, transformation, and reduction [170]. These techniques are applied to each dataset. Further preprocessing is applied to the merged dataset as depicted in Figure 3-2 and Figure 4-6. The main preprocessing steps are summarized in Table 4-2.

Table 4-2. Main preprocessing steps applied to DF1 and DF2

	Preprocessing Step	Reasoning
“RFID Readings” dataset (DF1)		
1	Keeping “InitialReadDateTime” and “LastReadDateTime”.	Initial and last timestamps used for time difference calculations.
2	Discarding “LocationSourceAntenna”, “LocationTagID”, “TagID”, “WallNumber”, and “LastReadDate”.	Contain either redundant or irrelevant information.
3	Transforming the dataset: each row stores timestamps for each MWP at each station; the original dataset stores each timestamp for each panel at each station in a separate row.	To prepare the data in a tidy format, and to facilitate CT calculations.
4	Replacing a missing value of timestamps at individual stations with the mean value provided that the timestamps at all other stations are not missing.	Replacing missing values so that the dataset is consistent and complete.
5	Records, where the production of a panel started on one day and finished on a subsequent day, are discarded.	Discarding outliers for consistent cycle time calculations.
“Multipanel” dataset (DF2)		
1	Keeping the dimension attributes (“Length”, “Width”, and “Height”) within a specific range. Panel lengths range between 1,203 and 12,338 mm; widths range between 1,607 and 3,235 mm; and heights range between 89 and 314 mm.	Only panel manufactured at the automatic production line are kept. Smaller panels, that are manually produced, are discarded.
2	Nominal values of the attribute ‘Type’ are converted to numbers using the “dummy coding” technique, commonly known as binarization. There are five unique values: ‘EXT’, ‘INT’, ‘STR’, ‘GAR’, and ‘MEC’.	For all unique values of the nominal attribute, a new attribute is created. The attribute corresponding to the value of the record gets a value of 1; other attributes are set to 0.
3	Discarding the attributes “Job”, “Component”, “wall”, “Siding”, “SidingLine”, “Model”, “Floor”, “Unit”, “GarageDoor”, “Sequence”, “Basementwall”, “position”, and “ProductionJob”.	These attributes are irrelevant as they contain redundant information, have one unique value, have few prevalent values, or have many missing values.
4	Missing values are either replaced using functions defined by fitting curves to the known values or discarded if they account for more than 90% of the total count of an attribute.	Replacing missing values so that the dataset is consistent and complete.

In general, preprocessing steps include discarding redundant attributes where the information is captured by other existing and more relevant attributes; transforming the structure of the dataset so it is a tidy format; replacing missing records or discarding them as applicable; and changing the type of data so it is more consistent and can be used with a wide variety of machine learning regression models. Moreover, it should be noted that the attributes "PanelNumber" in DF1 and "MultiPanel_Name" in DF2 are kept as unique identifiers for each record in the two datasets. These are common keys that are used, at a later step, to merge the datasets.

4.3.2 Cycle time calculation from raw RFID

As the multiwall panel passes over the RFID antenna, a timestamp is recorded in the attribute "InitialReadDateTime". When the panel leaves the RFID antenna range, the attribute "LastReadDateTime" records another timestamp. Ideally, when there is no interruption or waiting in the production line, the two timestamps should be similar indicating that the panel has not been sitting idle over the antenna. The idle time (IT) for panel *i* can be defined per Equation 4-1:

$$\mathrm{IT}_{panel} = t_{lastRead,i} - t_{initialRead,i} \tag{4-1}$$

The intermediate cycle time can be described as the processing time plus the idle time. Time differences between consecutive antenna locations indicate the time during which the panel is processed at each of the four intermediate stations. Total CT, referred to as CT_{act}, is calculated according to Equation 4-2 as the sum of these individual workstation times:

$$CT_{act} = \sum_{i=1}^{4} t_{initialRead,i+1} - t_{initialRead,i} \tag{4-2}$$

where $t_{initialRead,i+1}$, and $t_{initialRead,i}$ are the timestamps at the end and start antenna locations, respectively, for station i. It should be noted that CT_{act} as measured in the present study encompasses processing time, idle time, and the time spent on regular breaks during a typical working shift. Figure 4-9 shows the histogram and the kernel density smoothing curve of both the measured cycle time (ActualCT) and the cycle time without the break times (ActualCT-BT). In the case production facility, there are three break times during a typical working shift: two 15-minute coffee breaks and one 30-minute lunch break. If the production of the panel falls within these break times, the measured cycle time is reduced accordingly. In this manner, actual production times are calculated, and any adverse effect of break times on the accuracy of the cycle time predictions is mitigated. Hence, 'ActualCT-BT', rather than 'ActualCT', is used to develop and test the prediction models.

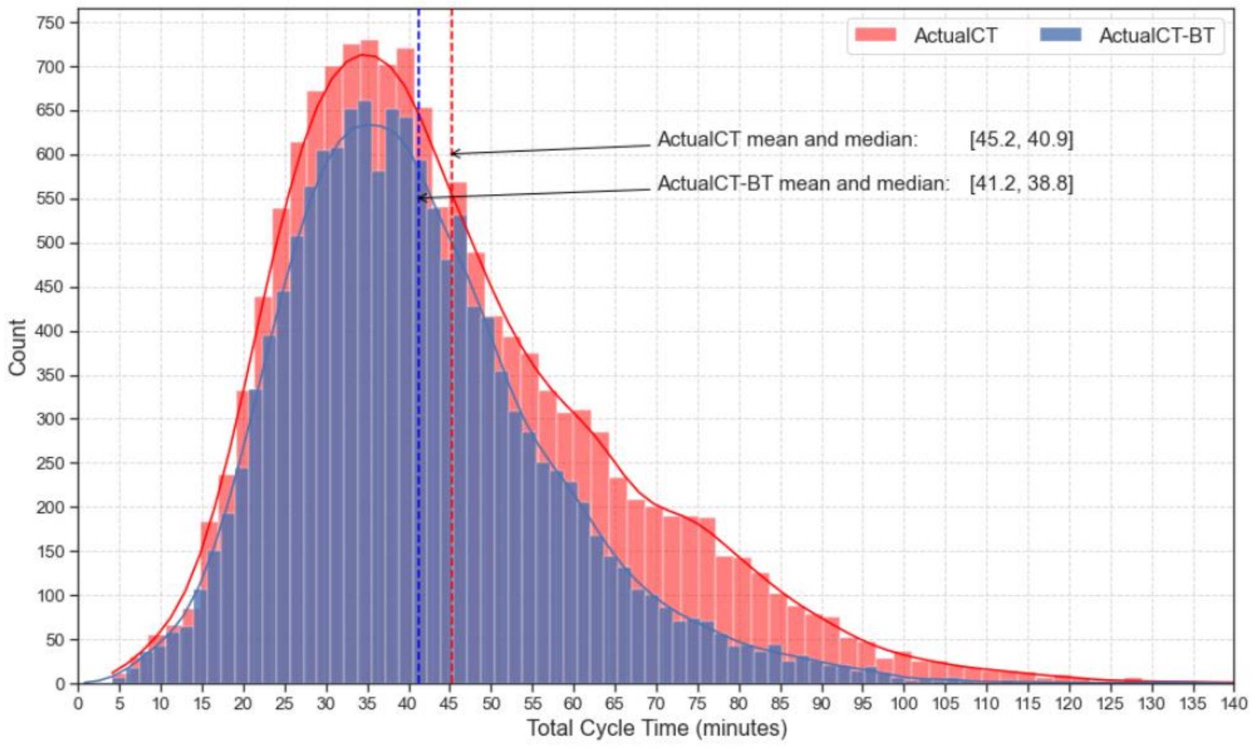


Figure 4-9. Total cycle time (between A1 and A5 in minutes)

4.4 Feature Engineering and Selection

Feature engineering (FE) has been widely used in data-driven modelling to improve the quality of feature sets. FE mainly encompasses three aspects: feature selection, feature extraction, and feature construction [171]. In the present research, each record in the tidy dataset stores the physical properties of a single multiwall panel, as well as the respective actual cycle time, which is the target value to be predicted. One of the main contributions of this research is to capture the system status by constructing new features that objectively measure the loading conditions of the job shop (i.e., SL features). In essence, these new features are constructed to augment the existing features of a product unit based on the units that immediately precede it along the production line. It is observed that CT in the IBC factory depends only partially on the physical features of the product and is governed largely by the SL features (i.e., a measure of how busy the fabrication floor is at the time production of a new wall panel begins). These shop loading features significantly enhance the model accuracy and hence the predictability of cycle time. This is mainly due to the fact that these features capture shop utilization, which otherwise (i.e., if only the physical features such as length, width, etc., are used) would be unaccounted for.

Let S_i represents the set of newly constructed features of product, i. Each feature, $s_i^k \in S_i$, is generated in such a manner as to capture attributes of panels that preceded the panel, i, in the production line but whose production is still underway (i.e., WIP). These WIP panels represent the current state of system utilization at the moment the production of the current panel, i, begins. If we let $X_{n\times m}$ be a matrix in which each row, n, represents one WIP panel preceding the current panel, and each column, m, represent an existing feature, x. Now, if we consider θ to be a binary vector of size m, where each element corresponds to one of the existing features, then a value of 1 corresponds to that feature being considered when calculating the new engineered feature of the current panel and other features are set to 0 and ignored. In this manner, each element of the set S_i is calculated as the sum of the elements of the matrix **M** as expressed in Equation 4-3:

$$s_i^k = \text{sum}(\mathrm{M}) \; where \; \mathrm{M} = \theta \cdot X^T = [\theta_1 \, \theta_2 \, \cdots \, \theta_m] \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ \vdots & & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{nm} \end{bmatrix} \qquad 4\text{-}3$$

Each element of **M** represents the contribution of each WIP when calculating the corresponding new engineered feature for panel *i*. This is a mathematical expression used to implement the algorithm for calculating each of the new engineered features for each panel in the dataset.

For the present study, SL features are developed to capture the system status based on the production during one work shift. The constructed features are "WP_i", "$WPLength_i$", and "$WPTime_i$", and they store the number, length, and processing time respectively, of the WIP panels. These features represent the set S_i where each element s_i^k is being generated by setting the appropriate values of the binary vector, θ. Given *i* = {1, 2, ..., n}, where *n* is the total number of WIP associated with the current panel, the attributes above are calculated based on Equations 4-4 through 4-7, as follows:

$$WP_i = |A| = cardinality \; of \; set \; A \qquad 4\text{-}4$$

$$A = \{(x, y) \colon x, y \in DF^2 \mid T_1(\mathrm{x}) > T_1(\mathrm{y}) \; and \; T_1(\mathrm{x}) < max(T_2(\mathrm{y}), T_3(\mathrm{y}), T_4(\mathrm{y}))\} \qquad 4\text{-}5$$

$$WPLength_i = \sum_{z \in A} L(z[1]) \; where \; z = (x, y) \qquad 4\text{-}6$$

$$WPTime_i = \sum_{z \in A} PT(z[1]) \; where \; z = (x, y) \qquad 4\text{-}7$$

where: *(x, y)* the pair of current panel, *x*, for which the new features are to be calculated, and panel *y* that represents any other panel which is concurrently being processed;

DF denotes the rows in the tidy dataset;

T_k is the timestamp at antenna location Ak where k ∈ {1, 2, 3, 4};

L is the length of panels that satisfy the condition; and

PT is the planned processing time of panels that satisfy the condition.

For the current dataset, a Python script is developed to automatically iterate over the records in DF and create the engineered features, which can be found in Appendix A. Figure 4-10 represents the process of calculating the length of the WIP panels, "$WPLength_i$", preceding panel i during a typical work shift. In this example, $t_{p,fs}$ is the timestamp associated with the panel when its production begins at the framing station, while $t_{p-1,ss}$ and $t_{p-2,mfb}$ are the timestamps for two panels that are ahead of the current panel but are still in production.

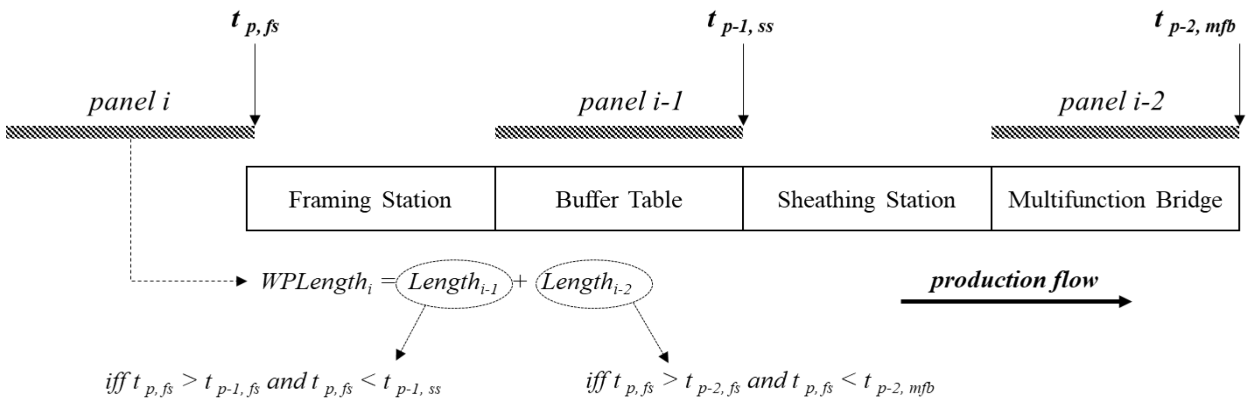

$t_{p,fs}$
$t_{p-1,ss}$
$t_{p-2,mfb}$
panel i
panel i-1
panel i-2
Framing Station
Buffer Table
Sheathing Station
Multifunction Bridge
$WPLength_i = Length_{i-1} + Length_{i-2}$
production flow
iff $t_{p,fs} > t_{p-1,fs}$ and $t_{p,fs} < t_{p-1,ss}$
iff $t_{p,fs} > t_{p-2,fs}$ and $t_{p,fs} < t_{p-2,mfb}$

Figure 4-10. Engineered feature calculations for Panel i at the time when it starts production

Moreover, based on the newly generated features, different versions of the dataset are generated each of which contains a different set of attributes to examine the effect of different attribute subsets on the prediction accuracy. These new attributes are created by combining existing physical characteristics in unified attributes in order to: (1) reduce the dataset dimensions, and (2) to examine the effect of features granularity on the accuracy of ML models. Table 4-3 summarizes the experiments conducted using different attribute aggregations. The resulting datasets are the ones depicted in Figure 4-6 as the tidy datasets.

Table 4-3. Subsets created by combining attributes of the tidy dataset

Name	Size (records × attributes)	Description
Raw data	1,094 x 22	Contains only physical properties of panels
Engineered	1,094 x 26	In addition to the physical properties of panels, it contains engineered features capturing the system status (WP, WPLength, WPTime, Panels/Day).
Engineered combined1	1,094 x 18	Combining similar properties of a panel into single attribute: **SheetFP** = Sheetfull + SheetPartial **TotalStuds** = Stud + 2*DStud + 2*LStud + 3*MStud **BlockBack** = Block + Backing **totalWD** = Window + LargeWindow + Door + LargeDoor
Engineered combined2	1,094 x 15	Combining all types of components (openings, stud, sheets, supportive elements) into one attribute: **Components** = totalWD + TotalStuds + SheetFP + BlockBack

4.5 Development of Machine Learning Models

4.5.1 Mining data using regression models

In the present case study, predicting production cycle time is a regression supervised learning task, and there exist numerous machine learning models to predict a continuous variable of this nature. Among these, four are selected for our study four models: linear regression (LR), k-nearest neighbor (KNN), neural network (NN), and random forest (RF). A successful model should provide reasonably accurate estimates of the cycle time to be used in factory planning and should be easy to interpret. Representative studies describing these methods include Hastie et al. (2009) [172] and Witten et al. (2011) [90]. Also, Raschka and Mirjalili (2017) [110] provided practical applications of machine learning libraries in Python. For a treatment of regression trees and random forests in particular, the interested reader may refer to Breiman et al. (1984) [173] and Breiman (2001) [130]. The four models mentioned above are selected from among the many available ML models available for regression based on the following considerations: (1) they range from simple to implement and easily interpretable (i.e., LR and KNN) to more sophisticated black-box models (i.e., RF and NN) so that the higher complexity of the models can be examined in terms of higher accuracy; (2) they have been widely used in the literature to investigate construction problems; and (3) they have been implemented in previous studies to predict production CT.

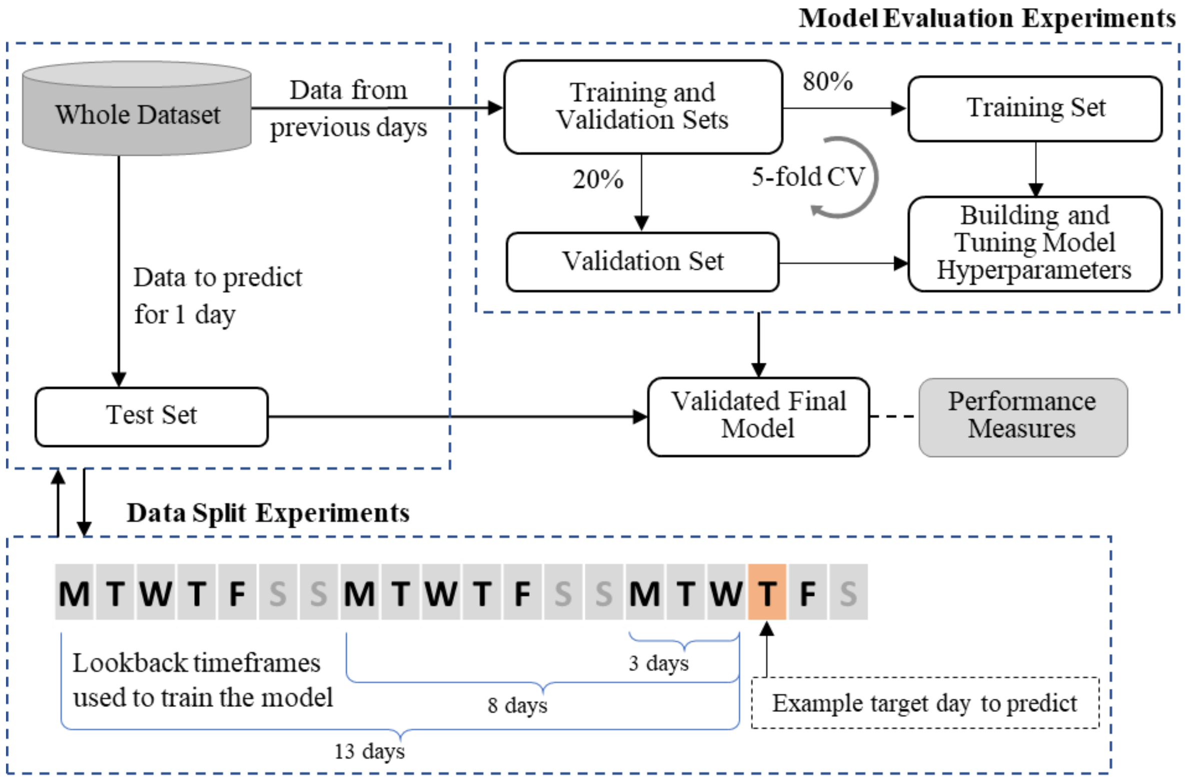

Model Evaluation Experiments
Whole Dataset
Data from previous days
Training and Validation Sets
80%
Training Set
20%
5-fold CV
Validation Set
Building and Tuning Model Hyperparameters
Data to predict for 1 day
Test Set
Validated Final Model
Performance Measures
Data Split Experiments
M T W T F S S M T W T F S S M T W T F S
Lookback timeframes used to train the model
3 days
8 days
13 days
Example target day to predict

Figure 4-11. Data splitting and model evaluation

The tidy dataset having been developed, the selected models are trained and tested. Per Witten et al. (2011), the dataset is divided into three subsets: a training set, a validation set, and a test set [90]. The test set contains the records from one day where cycle time is to be predicted. The training and validation sets contain the production data from previous days which are used for training and validation using a 5-fold cross-validation (CV) technique, as shown in Figure 4-11. In the current study, models are trained and tested using Scikit-Learn, a Python open-source library for machine learning [174].

4.5.2 Comparison of model performances

The performance metrics of the models on the testing sets are shown in Table 4-4 for the four datasets described in Table 4-3. Refer to Section 2.3.4 for descriptions of these metrics.

Table 4-4. Model performance measures for the datasets described in Table 4-3

	ML model	LR	KNN	RF	NN
	Hyperparameters	-	k = 11, p=2	400 trees, mae	(52, 52) 2000 itr.
Raw data	R^2 on training	0.27	0.91	0.80	0.24
	MAE	9.2	6.8	5.8	7.7
	MedAE	6.7	3.6	3.1	4.8
	MaxAE	47.3	42.1	21.7	40.2
	MAPE	23.6%	17.3%	17.0%	19.7%
	CC	-0.03	0.13	0.47	0.19
Engineered	R^2 on training	0.46	0.99	0.90	0.42
	MAE	7.1	6.1	5.3	6.7
	MedAE	5.6	3.4	3.5	4.4
	MaxAE	38.2	34.1	31.4	37.9
	MAPE	17.8%	14.9%	13.7%	16.4%
	CC	0.34	0.47	0.71	0.44
Engineered combined1	R^2 on training	0.43	0.99	0.90	0.40
	MAE	6.6	6.1	4.9	6.9
	MedAE	4.2	3.4	2.8	5.0
	MaxAE	38.1	34.2	30.6	38.4
	MAPE	16.4%	14.9%	12.3%	16.7%
	CC	0.40	0.47	0.73	0.39
Engineered combined2	R^2 on training	0.41	0.99	0.90	0.39
	MAE	6.1	6.0	4.4	6.5
	MedAE	4.3	3.3	3.2	4.3
	MaxAE	35.2	34.1	29.0	37.7
	MAPE	15.0%	14.9%	11.0%	15.6%
	CC	0.53	0.47	0.80	0.44

The hyperparameters of the four models have been tuned, using the training and validation dataset, to attain better performance and robust models. The model performance is measured on the testing dataset. The performance of machine learning algorithms depends strongly on finding an optimal set of hyperparameters. There exist strategies to search for the best combination of hyperparameters including grid search and random search [94]. In the present study, a combination of manual and brute force search using the GridSearchCV method available from the Scikit-Learn library is used to search for the optimal combination of a predefined set of values for each model's hyperparameters. The most important values are listed in Table 4-4 under the corresponding model's name.

4.5.3 Observations on model performances

4.5.3.1 General trends

Referring to Table 4-4, a distinct difference can be observed between the MAE and the MedAE values. The MedAE is smaller than the MAE for all models, a trend aligned with the distribution pattern of actual cycle time, which is positively skewed (moderately skewed to the right), as shown in Figure 4-9. This is attributable to the presence of a few outlier records that are significantly higher than the average values which affect the overall accuracy of the predictive models. Another observation is that achieving a higher value of R^2 on the training set does not necessarily translate into better performance on the testing set, as can be seen with respect to the performance of KNN, where the model is observed to have overfit the training data. As this example underscores, the model performance should not be evaluated based on the metrics of the training set.

Across all models and datasets, the MAPE range between 24% (worst) and 11% (best), and the CC range between 0.0 (no correlation) and 0.8 (strong positive correlation). Overall, the RF models have better performance metrics in terms of lower MAE, MedAE, MaxAE, MAPE, and higher correlation coefficients. It should be noted, however, that RF requires slightly more computational time. An advantage of RF over other methods is the ability to handle both categorical and continuous attributes, and attributes need not be scaled to common units. When testing several configurations of RF models, using approximately 400 trees and MAE as the evaluation criterion when splitting the nodes yielded the best performance.

4.5.3.2 Engineered features

The production system utilization status is valuable information to add to the ML models to improve the predictability of CT. The raw dataset, *'Raw data'* in Table 4-3, includes only the physical properties of the panels; the engineered features that represent the system utilization status are only included in the other three datasets. Based on the performance measures, using the raw data without the SL features leads to inferior model performance. The MAPE is found to be as high as 24%, while the CC is low for all models. Meanwhile, an improvement in MAPE of as much as 9% is achieved when augmenting the datasets with the proposed SL features. Moreover, the CC is found to have increased for all models, to as high as 0.80 in the case of the RF model.

Intuitively, augmenting the feature set with engineered features should result in better predictability of the developed ML models, since this provides in-depth information about how the production floor is being utilized at the point when production of a new product begins. Based on the historical data, this assumption is proven to be valid, performance improvements having been consistently achieved for all models.

4.5.3.3 Combining features

The performance of all models does slightly improve by reducing the granularity of the dataset. For example, the mean absolute percentage error of the LR, and RF models dropped from 17.8% and 13.7% in the case of the first version of the engineered dataset, *'Engineered'*, to 15.0% and 11.0% in the case of the reduced dataset, *'Engineered combined2'*, respectively. In the aggregated dataset *'Engineered combined2'*, all components of a wall panel (studs, windows, doors, backing, and sheathing sheets) are combined and represented by one feature, namely, "Components".

In addition to requiring less computational time, it can be concluded that reducing the dimensionality of the data by combining similar attributes does improve the performance in terms of lower prediction errors and higher CC. This is attributable to the fact that the combined attributes have a weak correlation with the target feature when computed individually, whereas, when these attributes are aggregated, the correlation with the target feature is improved and resulted in better predictability of cycle time. It should be noted, however, that this conclusion is specific to the current case study and is not necessarily true for all other cases. Further investigation is required to explore this aspect in the context of other cases.

4.5.3.4 Lookback timeframe for model training

A sample dataset is examined in reference to various lookback timeframes as part of the machine learning model development phase in order to determine the optimum timeframe used for testing. The goal here is to investigate the effect of the number of days used to train the model on the model performance and the predictability of cycle time. For the present case study, using the historical data of a lookback timeframe of 10 to 20 days for training and validation is found to result in better performance measure compared to shorter or longer lookback timeframes as shown in Figure 4-12. That is, to predict cycle time on a specific day, the production data from the previous 10 to 20 workdays is required to train and validate the machine learning models. Using the historical data in its entirety or using the data from a few previous days usually results in inferior performance. Figure 4-11 illustrates the manner in which splitting of the data is carried out. It is worth noting that further investigation is required to examine the full effect of varying lookback timeframes on different days of the week. This task will be undertaken as part of future work.

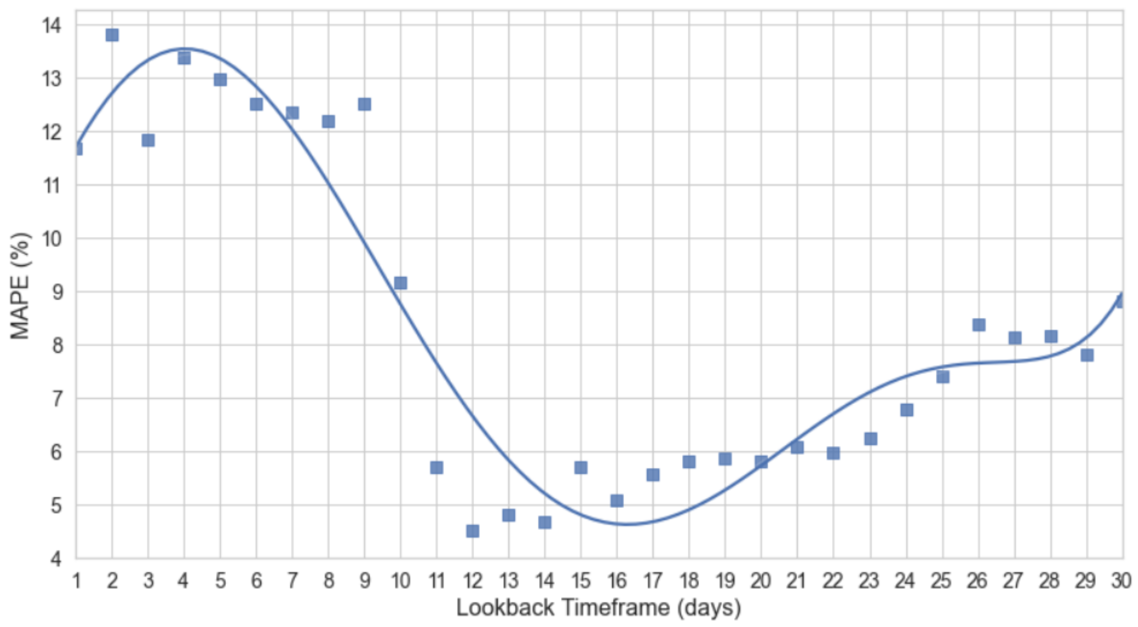


Figure 4-12. Minimum MAPE versus Lookback Timeframe for the dataset *'Engineered'*

4.5.4 Comparison with Little's law

The simplest and most widely used analytical method to predict CT is Little's law, which can be applied to dynamic systems by dividing the time into finite intervals. As described in Section 1.1, Little's law can be used as a benchmark to assess the performance of the proposed approach. Any of the average CT, throughput (TH), and average units in the system ($\overline{WIP}$) can be calculated by making observations about the other two quantities over a given time interval [9]. To compare these quantities, the time interval in the current study is set to be one workday (in this specific case, it is also one work shift). The average daily production, shown in Figure 4-8, is found to be 39.5 panels/day. Given a 7.5-hour work shift, this translates into a system throughput of about 5 panels/hour. When the observed panel enters the production route, the average number of panels that are being processed and that remain in the system is found to be 3.5 panels. The CT is then calculated, using Little's law, as 43.3 minutes, which is very close to the calculated average of 40.9 minutes as shown in Figure 4-9. Assuming that this average number is used as an estimate of CT for all panels, the performance metrics used to compare ML models are calculated for the given dataset as follows: MAE = 11 minutes, MedAE = 8 minutes, MaxAE = 36 minutes, and MAPE = 30%. These values are inferior to the values obtained from any of the ML models. Although this analytical method can be used to capture the average performance of the system by estimating CT based on the average daily production, error rates are high and the variability in the production cannot be efficiently captured which is the typical case of most IBC manufacturing environments.

4.5.5 Limitations and future work

The present case study focuses on using existing historical data obtained from an RFID tracking system in a residential IBC. This data contains considerable noise and requires extensive data cleaning and preparation. In addition, due to the existing RFID system setup, some data that might contain valuable information is not collected, especially data associated with panel waiting times throughout the production process. Therefore, this limitation is further investigated in Chapter 5 by developing a simulation model that can generate the missing data pertaining to waiting and idle times of panels while in production. Moreover, another limitation is the applicability of ML approaches to other subsectors of IBC. The goal is to examine the applicability and consistency of using machine learning models to predict CT in the context of mass customization production which is an important characteristic of the IBC industry. This limitation is investigated in Chapter 6 by examining a second case study in industrial IBC. It is worth noting that the details of the engineered feature development reflecting the loading conditions of the job shop will depend on the specific case study, and some custom development may be required. However, due to the similarities among IBC manufacturing environments, with some minor customization the presented features can give a representative picture of the current state of shop utilization in a given IBC facility and thereby improve the predictability of production CT. Investigating other case studies in IBC can lead to the generalizations of results obtained in this the present case study.

Chapter 5 Simulation Modelling

This chapter describes the development of a discrete-event simulation (DSE) model for the first case study presented in Chapter 4. The chapter is divided into four main sections. First, the simulation model environment and the logic based upon which the model is developed are described in detail. Then, formulas and data distributions used to estimate activity durations are presented for the various elements of the model. The third section discusses model verification, validation, and outputs. In the fourth section, machine learning models developed using the simulated datasets and the resulting improvement are discussed, followed by concluding remarks.

5.1 Description of the Simulation Model

Discrete-event simulation (DES) has been widely used over the past few decades in the construction and manufacturing industries as a versatile technique to model the various activities and processes of real-world systems. Many research studies have been carried out to explore simulation language development, simulation model design, model optimization, and combining statistical design-of-experiment techniques with simulation modelling [175]. An overview of advancements in simulation theory and its applications in the construction industry was conducted by AbouRizk [176]. In this section, the purpose and context of the DES model of this study is explained, followed by a detailed description of the model and its components.

5.1.1 Purpose and context

The DES model is developed to achieve the third objective of the present research that aims to examine the effect of generating engineered features that capture the real-time loading conditions of the job shop (i.e., SL features). Such simulated shop loading features are not practically collected, and in most cases need to be derived from existing data. The goal here is to investigate how certain SL features would improve the predictive accuracy of machine learning models if the relevant SL features could be captured in the real-world. Using simulation modelling, the collection of such features in the facility can be investigated. By experimenting with the DES model, we are searching for the right set of features that can help to improve the predictability of cycle time and thereby enhance the performance of data-driven models. The performance of the machine learning models that were created using the simulated datasets (i.e., with and without SL features) is examined and compared. Figure 5-1 illustrates the scope of the present DES experimentations and how they relate to the content in Chapter 4.

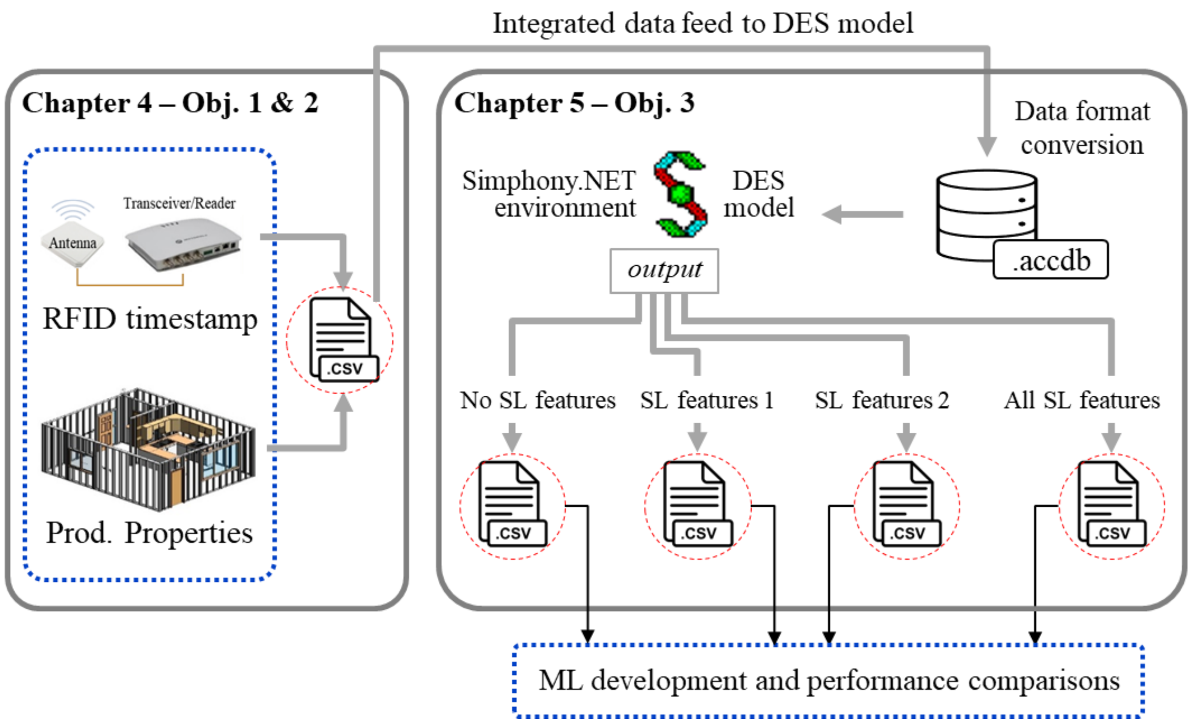


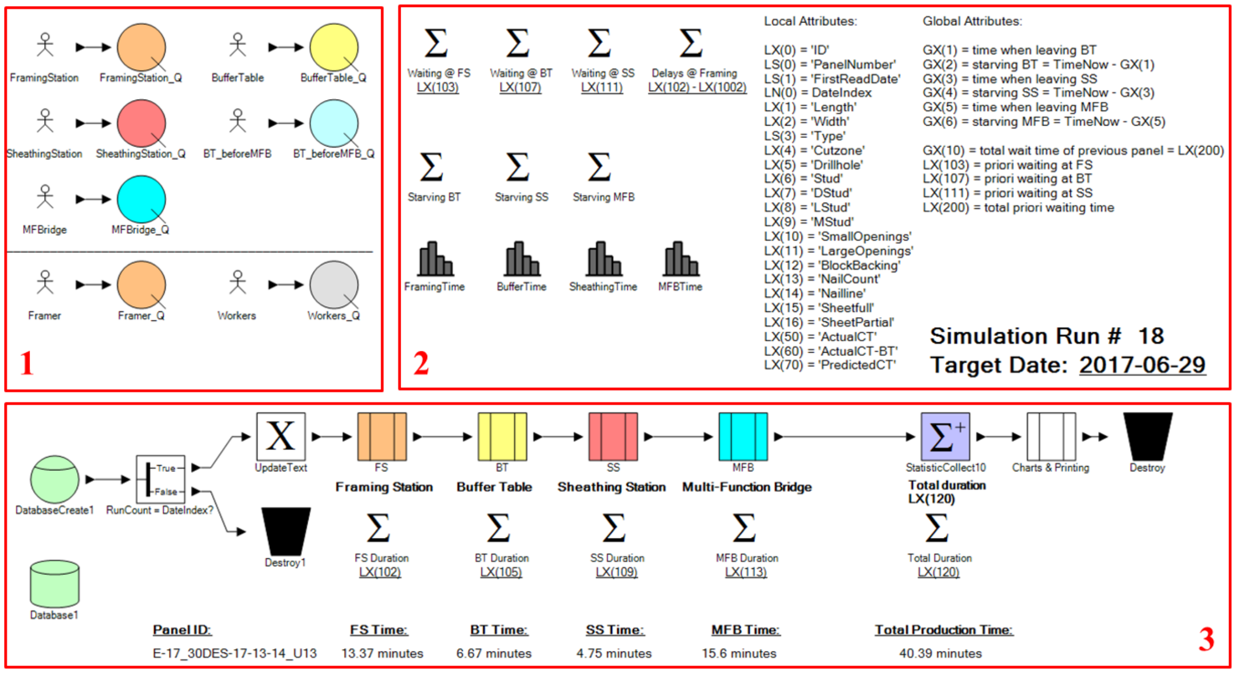
Figure 5-1. Simulation model produces four datasets

The integrated data (i.e., the dataset obtained by integrating the physical characteristics of wall panels and the RFID tracking data as described in Section 4.2) are converted to MS Access database format (.accdb file), which is the format accepted by the DES modelling tool. This database is used to generate the simulated entities in the simulation environment, each of which presents a single multiwall panel. The database records are fed into the DES model; the process for which is described in detail in the next section. The outcomes of the model are stored in four spreadsheets (.csv files) each of which is then used to build a machine learning model. Each of the four datasets contains a different set of features: a) the first dataset only contains the physical characteristics of wall panels, which is hereafter referred to as the 'raw dataset'; b) the second dataset adds to the raw dataset a new set of shop loading features that capture the waiting times of priori panels and starving times of workstations of priori production, and it is hereafter referred to as the 'SL wait dataset'; c) the third dataset adds to the raw dataset a set of shop loading features similar to the ones developed in Chapter 4 (i.e., features capturing the number, length, and processing times of priori panels), and it is hereafter referred to as the 'SL length dataset'; and d) the fourth dataset adds to the raw dataset all shop loading features developed earlier, and it is hereafter referred to as the 'SL full dataset'.

5.1.2 Discrete-event simulation model

The multiwall production process is described in Section 4.1 based upon which the detailed DES model in Figure 5-2 is developed using the General Template in Simphony.NET, a simulation environment developed by researchers at the University of Alberta [177,178].

The simulation model is colour -coded and subdivided into three main parts for consistent development, easier implementation, and streamlined maintenance and updating. Each graphical notation, or object, shown in Figure 5-2 represents an abstraction or summarization of an entity in the real-world production system. These objects are referred to as modelling elements and are used to resemble the actual production process. These modelling elements include *database, create, task, resource, capture, release, file, valve, set attribute, counter, execute, branch, composite,* *destroy,* and *statistics*. In general, the simulation entity, hereinafter the entity, represents a single MWP and it goes through various modelling elements to complete the simulation. The *DatabaseCreate* element is used to generate entities and set their local attributes by reading the database that is loaded by the *Database* element. The *resource* element represents machines and workers, and each entity is required to capture the associated resources in order to complete a task. The elements *capture, release,* and *file* are used to model the acquiring, releasing, and waiting of resources. When no resource is available, the entity waits in the *file* element and the waiting time is recorded. Actual work taking place in a workstation is represented by the *task* element where its time parameter simulates the actual processing time of the activity. After completing the associated activity, the entity releases the resources and moves forward to the next process.



Local Attributes:

LX(0) = 'ID'
LS(0) = 'PanelNumber'
LS(1) = 'FirstReadDate'
LN(0) = DateIndex
LX(1) = 'Length'
LX(2) = 'Width'
LS(3) = 'Type'
LX(4) = 'Cutzone'
LX(5) = 'Drillhole'
LX(6) = 'Stud'
LX(7) = 'DStud'
LX(8) = 'LStud'
LX(9) = 'MStud'
LX(10) = 'SmallOpenings'
LX(11) = 'LargeOpenings'
LX(12) = 'BlockBacking'
LX(13) = 'NailCount'
LX(14) = 'Nailline'
LX(15) = 'Sheetfull'
LX(16) = 'SheetPartial'
LX(50) = 'ActualCT'
LX(60) = 'ActualCT-BT'
LX(70) = 'PredictedCT'

Global Attributes:

GX(1) = time when leaving BT
GX(2) = starving BT = TimeNow - GX(1)
GX(3) = time when leaving SS
GX(4) = starving SS = TimeNow - GX(3)
GX(5) = time when leaving MFB
GX(6) = starving MFB = TimeNow - GX(5)

GX(10) = total wait time of previous panel = LX(200)
LX(103) = priori waiting at FS
LX(107) = priori waiting at BT
LX(111) = priori waiting at SS
LX(200) = total priori waiting time

Simulation Run # 18
Target Date: 2017-06-29

Panel ID:	FS Time:	BT Time:	SS Time:	MFB Time:	Total Production Time:
E-17_30DES-17-13-14_U13	13.37 minutes	6.67 minutes	4.75 minutes	15.6 minutes	40.39 minutes

Figure 5-2. DES environment in Simphony.NET using General Template

Part 1 of Figure 5-2 depicts the required resources and their waiting queues. For the MWP production line described in Section 4.1, the four workstations and the workers are represented by seven resources in the simulation model. For example, the framing station (FS) and its dedicated worker are represented by orange-colored *file* elements in the model and are used to simulate the framing station (FS) process as described later in this section. Part 2 of the simulation model includes static and dynamic information blocks as well as *statistics* elements that store and aggregate processing and waiting times of each entity. For example, "Local Attributes" are static information that shows the mapping of MWP data imported form the database to the local attributes of the entities. This is provided for easier interpretation and development of the model.

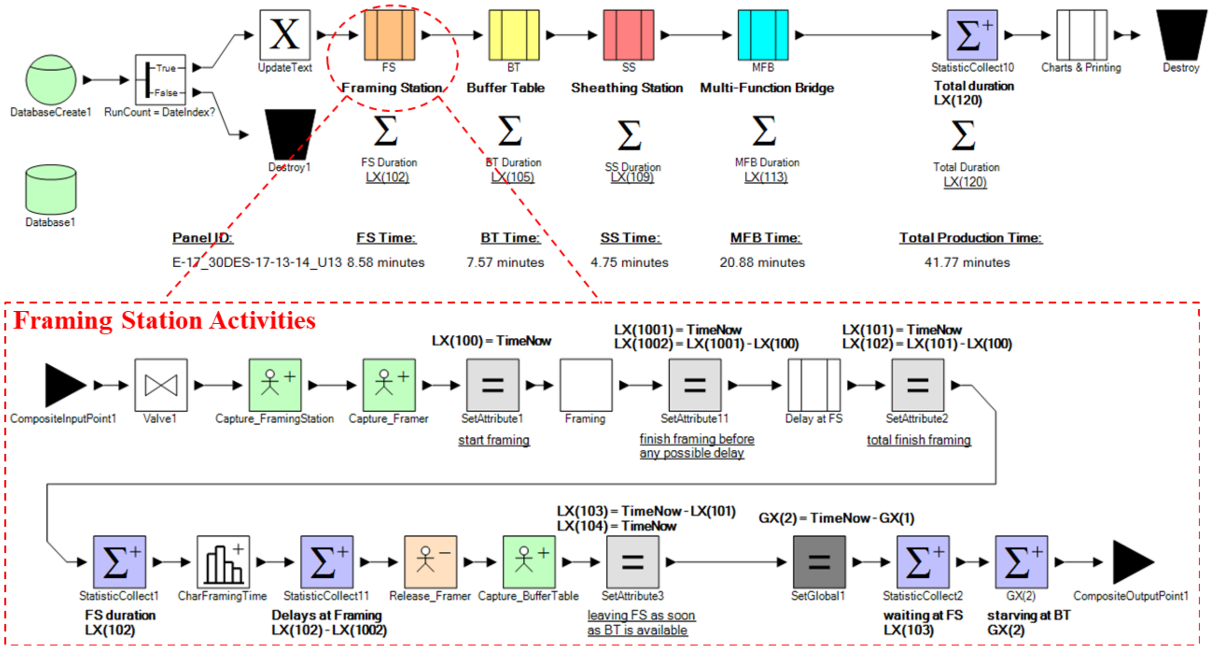


Figure 5-3. Detailed tasks, resources, and statistics of the framing station

Part 3 represents the simulated processes where each workstation is represented by a *composite* element that accepts an entity (i.e., a multiwall panel) from a previous process, route the entity through different modelling elements, calculate and store the processing and waiting times, and then outputs the entity to the next process. As an example, the details of the framing station *composite* element are shown in Figure 5-3. When an entity is routed to the FS *composite* element, it first captures two resources: a) a portion of the framing table equal to the length of the wall panel where the maximum allowable length is 40 feet, and b) the framer where there is only one available worker at the framing station. By restricting the resources availability in this fashion, once the worker has completed the work on one panel, s/he can begin to work on the next panel given that there is enough space on the framing table to accommodate the next panel. Once the required resources are captured, the entity is held for a specific length of time depending on the time it takes to frame the multiwall panel. This time depends on the physical characteristics of the panel and is coded in the corresponding *task* element. Estimating activity durations in the simulation model is known as "input modelling" and is discussed in Section 5.2.

In order to simulate possible delays at the framing station, a *composite* element is added after the main processing task with five observed delay incidents (i.e., checking plans, shortage of material, correcting errors, filling nail magazines, and workers idling) as shown in Figure 5-4. Each type of delay has a probability of occurrence and an average duration that are calculated based on the collected observations and further discussed in Section 5.2. It should be noted that only one type of delay might occur for an individual panel component. Before and after the framing *task* element and the delay *composite* element there are three *set attribute* elements that are used to set the value of local entity attributes to the current simulation time (e.g., **`LX(100) = TimeNow`** in Figure 5-4). By doing so, the processing and delay times at the framing station are calculated as the difference between the respective local attributes. Once an entity exits the delay *composite* element, it releases the framer resource and captures the next station (i.e., the buffer table (BT)) if it is not occupied by a preceding entity. If the buffer table is available, the entity exits the FS composite element which denotes the completion of the framing process.

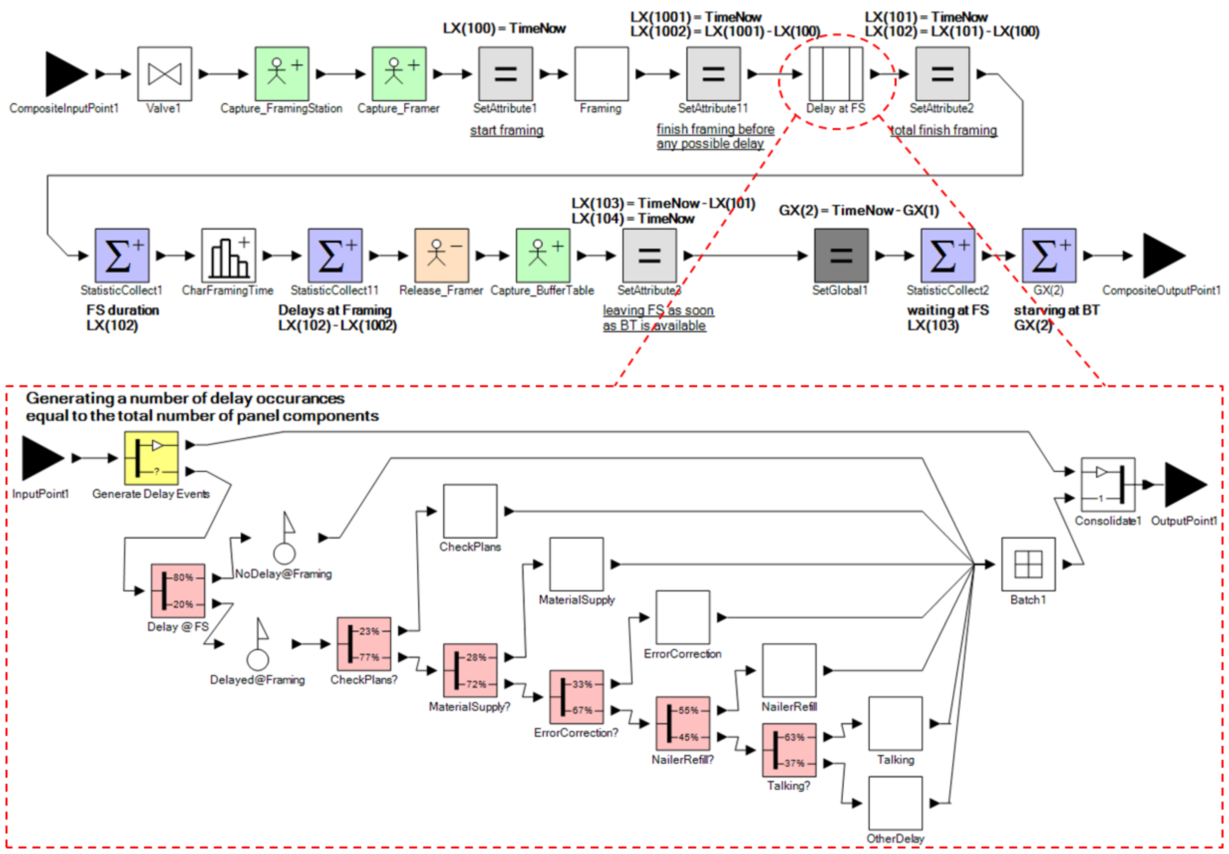


Figure 5-4. Simulating delay events at the framing station using the *branch* element

Following the same logic and similar approach, the entity moves sequentially forward to the buffer table, sheathing station, and multifunction bridge *composite* elements. At each workstation, the resources are captured and released between operations and the corresponding processing time is collected through *statistics* elements and stored in local attributes of the entity. Once an entity passes through all the workstations, all processing cycle and delay times are collected, and the entity is routed to the "Charts & Printing" *composite* element before being destroyed by a *destroy* element, which simulates the completion of this phase of production. Within the "Charts & Printing" element, an *execute* element is used to convert and store all the data and statistics of the passing entity (i.e., MWP physical properties and the simulated processing and waiting times) into a record in a spreadsheet (.csv file) for each day of production. Spreadsheets for all production days are then combined and used for later analysis and to develop the machine learning models.

The simulation model runs until it has executed all the entities that exist in the database. The DES model is setup to process wall panels on a daily basis (i.e., process and record statistics of the panels that have the same ‘date’ attribute from the database). The *counter* elements within the model are used to count and track the number of entities passing through a specific path. This is used for the verification process of the model, which is described in Section 5.3.1.

5.2 Estimating Activity Durations

In practice, if an activity is performed repeatedly, it takes a different duration every time it is carried out. This duration depends on a large number of factors including worker skill level, material availability, and machine performance, to a name a few. Activity durations in discrete-event simulation are traditionally modelled using probability distributions [176]. For each activity in the model, the task time required to complete the activity is estimated by using a) a closed-form formula that largely depends on the physical characteristics of the panel being processed, or b) a random variate sampled from a probability distribution that is fitted to actual data (i.e., actual data is either collected during a time study or extracted from RIFD timestamps).

Each MWP has different attributes (e.g., type, length, width, number of regular studs, number of windows/doors) which are required to estimate the processing time at different workstations and to route the entity as it travels through the DES model. For example, the processing time to complete the framing of single panel i, $\mathrm{T}_{frame,i}$, is the summation of installing individual components and is calculated using Equation 5-1 as follows:

$$\mathrm{T}_{frame,i} = T_{P,i} + (SR_i * T_{SR,i}) + (SDL_i * T_{SDL,i}) + (SM_i * T_{SM,i}) + (OS_i * T_{OS,i}) + (OL_i * T_{OL,i}) \quad 5\text{-}1$$

where: $T_{P,i}$ is the time required to load the top and bottom plates;

SR_i is the total number of regular studs (i.e., 2" × 4" or 2" × 6" standard stud);

T_{SR} is the time required to nail a single regular stud to the top and bottom plates;

SDL_i is the total number of double and L-shaped studs;

T_{SDL} is the time required to nail a single double or L-shaped stud;

SM_i is the total number of multi-studs (i.e., 3 or more studs installed together);

T_{SM} is the time required to nail a single multi-stud;

OS_i is the total number of small openings (window or door assembly);

T_{OS} is the time required to install a single small opening component;

OL_i is the total number of large openings (window or door assembly); and

T_{OL} is the time required to install a single large opening component.

As for the sheathing station (SS), the time required to install sheathing sheets for exterior panels, $\mathrm{T}_{sheathE,i}$, is the summation of the time required to place the sheets and then nail them to the frame by workers as per Equation 5-2. For interior panels, the entities bypass the installation path as interior panels do not have sheathing. This routing logic is built within the DES model.

$$\mathrm{T}_{sheathE,i} = (SSF_i + SSP_i) * T_{place} + (SSF_i + SSP_i) * T_{nail} \qquad 5\text{-}2$$

where: SSF_i is the total number of full sheets of sheathing for panel i;

SSP_i is the total number of partial sheets of sheathing for panel i;

T_{place} is the time required to place a single full or partial sheet; and

T_{nail} is the time required to nail a single full or partial sheet.

Work at the sheathing station is advanced primarily by manual workers, thus the process time for each task can be considered deterministic by splitting the entire process into sub-task groups. It should be noted that the time required to install single components as described in the above equations (i.e., time required nail a single stud or place a single full sheet) can either be deterministic or stochastic drawn from a probability distribution. Examples of activity durations, calculated in minutes, at each of the four workstations are shown in Table 5-1. Duration distributions are either developed based on 1) actual RFID timestamps for triangular and beta distributions, or 2) observations of the production floor operations for constant values. In the Duration Distribution column, the three numbers indicated as being sampled from a triangular distribution refer to the lowest, highest, and most likely values.

Table 5-1. Examples of activity duration distributions

Workstation	Activity	Duration Distribution (minutes)
Framing Station	Nailing single regular stud	`Triangular(0.10, 0.70, 0.15)`
	Nailing single double/L-shaped stud	`Triangular(0.20, 0.95, 0.35)`
	Nailing single multi-stud	`Triangular(0.30, 1.40, 0.45)`
	Installing small opening assembly	`Triangular(0.80, 2.00, 1.00)`
	Installing large opening assembly	`Triangular(1.00, 2.20, 1.20)`
	Checking plans – delay	`Triangular(0.05, 0.35, 0.16)`
	Correcting errors – delay	`Triangular(0.05, 1.00, 0.30)`
Buffer Table	Preparing cuts	`Constant(0.50 * #cutzones)`
	Correcting errors for interior panels	`Beta(2.02, 7.13, 0.03, 34.95)`
	Correcting errors for exterior panels	`Beta(2.13, 11.13, 0.33, 54.5)`
Sheathing Station	Placing exterior sheets	`Constant(0.30 * #sheets)`
	Nailing exterior sheets	`Constant(0.20 * #sheets)`
	Moving interior panels	`Constant(0.25)`
Multifunction Bridge	Refilling nail magazine	`Triangular(1.00, 2.00, 1.50)`
	Various delays – for exterior panels	`Triangular(1.50, 9.00, 4.50)`
	Removing wastes – for interior panels	`Constant(0.05 * panel length)`

To determine the delays that might occur at the framing station, the durations of individual delays are observed, and the distribution for delay time and the probability of delay occurrence are calculated. The likelihood of a delay occurrence happening is calculated as the number of observations where that type of delay occurs over the total number of observed delays.

5.3 Model Validation and Output

5.3.1 Model validation and verification

Simulation model verification is the process carried out to ensure the model has been correctly implemented, while model validation refers to confirming that the model emulates the real-world system to an acceptable level of accuracy. More formally, model verification is defined as "ensuring that the computer program of the computerized model and its implementation are correct", and model validation is defined as "the substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [179].

The developed simulation model can be verified in several ways. One way is by including a dynamic counter that changes as the entities pass through the model. This is implemented by the text block that changes the simulation run and the target date as shown in the lower right corner of Part 2 in Figure 5-2. As the entities flow through the model, the simulation run, and the target date (i.e., the date at which panel production took place) are updated as well as the processing times at each workstation which verify that the model is implemented correctly. Moreover, the total number of entities created matches the total number of entities destroyed. The model entities are traced along the simulation time using several *counter* elements to determine if the model logic and branch probabilities are correct. Also, the model integrity check provided by Simphony.NET is used prior to model execution where logical, data, and computer language errors that are identified.

The simulation model is validated at two levels. First, the simulated total cycle time is compared with the actual cycle time obtained from the RFID tracking system. This comparison is carried out by examining the mean value and variance of the two cycle times and checking the statistical significance of the difference using two-tail paired t-test statistics at a 95% confidence level (i.e., at a significance level $\alpha = 0.05$). The paired sample t-test is a statistical procedure used to determine whether the mean difference between two sets of observations is zero, and is performed by comparing the calculated p-value to the given significance level, α. Given the mean of simulation cycle times, μ_{sim}, and the mean of actual cycle times, μ_{act}, the null hypothesis, H_o, and alternative hypothesis, H_a, are set as shown in Equation 5-3.

$$H_o: \mu_{sim} - \mu_{act} = 0 \; and \; H_a: \mu_{sim} - \mu_{act} \neq 0 \qquad 5\text{-}3$$

The simulated data has been randomly sampled from all available data based on stratified sampling where each month represents one stratum. Given a sample of one month which contains a total of 503 records representing 18 working days, the mean values of simulated and actual cycle time data are $\mu_{sim} = 44.6$ minutes, and $\mu_{act} = 43.6$ minutes, respectively, as shown in Figure 5-5. The p-value is calculated as 0.182, which is greater than the selected level of significance of 0.05 which leads to the conclusion of failing to reject the null hypothesis. That is, from a statistical viewpoint, the performance of the simulation model is very comparable to the actual production system which validates the overall model. The second level of validation is performed at the workstation level. Each workstation time is validated by comparing the simulated CT and actual CT extracted from the RFID timestamps. Figure 5-6 shows the empirical density distribution of actual cycle times (in orange) along with the simulated cycle times (in blue) for each of the four workstations.

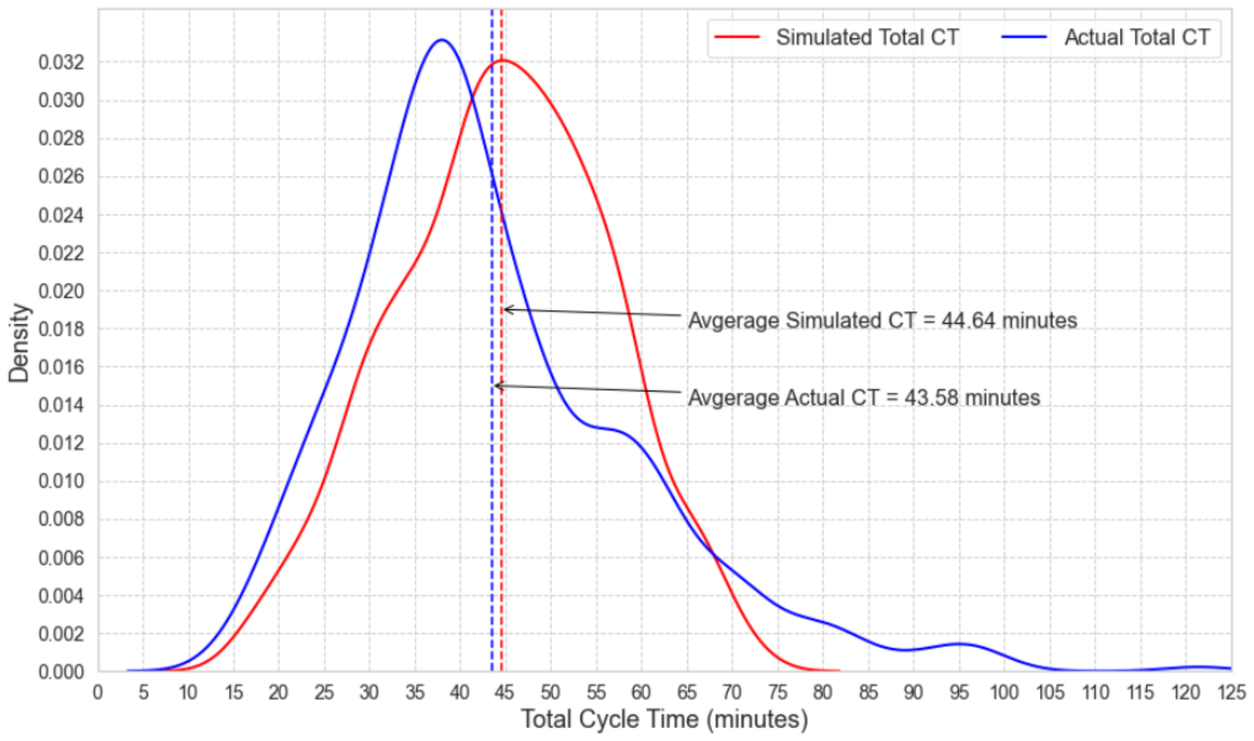


Figure 5-5. Probability distribution of simulated and actual total CT

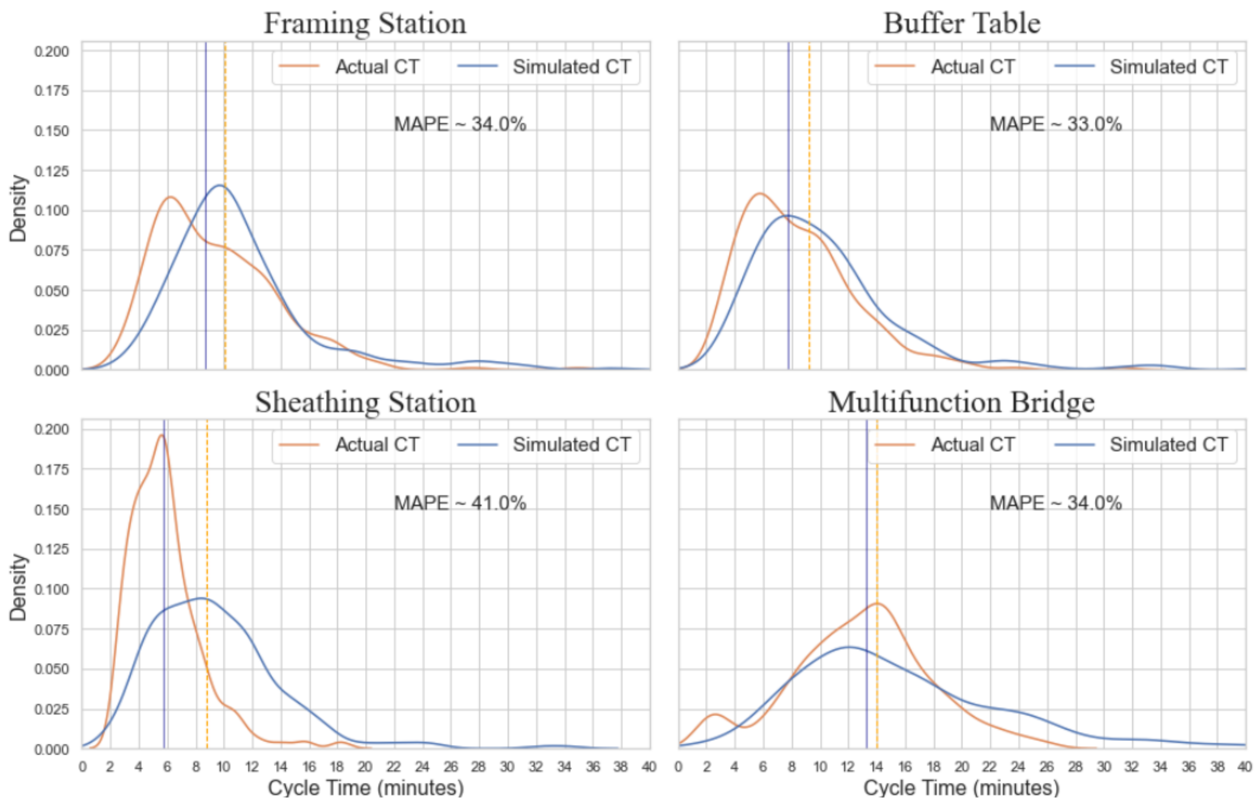


Figure 5-6. Probability distribution of actual and simulated CT for the four workstations

The vertical lines in Figure 5-6 indicate the median value of each cycle time. One observation is that the actual cycle time curve is more tortuous compared to the smoother simulated curve which is an indication of more irregularities during actual operations. Except for the sheathing station, the two curves (i.e., actual and simulated) seem to follow the same pattern and are very comparable suggesting that the simulation model is valid at the workstation level. As for the sheathing station, the apparent difference between the two curves might be due to the nature of operations at this station since they are more manual and less predictable, which indicates the simulation model does not accurately capture the variability of the operations.

Nevertheless, this behavior does not affect the validity of the simulation output as the purpose of the model is not to examine what-if scenarios in which case the model might need to be more accurate at an individual workstation level.

5.3.2 Simulation model output

Simulation model outputs can be very diverse, ranging from statistics reports to graphical representations of target variable distributions. However, the purpose of the present DES study, as described in Section 5.1.1, is to generate additional features capturing some aspect of the real-time loading conditions of the job shop. Those features are not present in the original dataset and might be impractical or impossible to extract from the raw data. Hence, the main output of the present simulation model is a spreadsheet (.csv file) that contains the original dataset augmented with additional engineered features. This dataset is used to generate four datasets, each of which contains a different set of features as described in Section 5.1.1 and are used to develop the ML models as illustrated in Figure 5-1 and further described in Section 5.4. For the second dataset (i.e., 'SL wait dataset'), the features generated from the DES model capture some aspect of the real-time loading conditions of the job shop that cannot be extracted from actual data, and they include the following:

a) Features that record the waiting time of the panel preceding the current active panel, i.e., "priori wait @FS", "priori wait @BT", and "priori wait @SS".

b) Features that record the starving time of workstations before the current active panel starts the production process, i.e., "starving @BT", "starving @SS", and "starving @MFB".

Those two sets of features are believed to capture important aspects of the system status at the time the current panel enters the production line. The effect of the newly created features is investigated when developing the ML models by examining the improvement of the predictive accuracy when those features are added to the dataset as compared to the raw dataset. As the simulation entity goes through the simulation model, those features are captured and stored in global attributes (i.e., *GX* attributes in Simphony.NET) and assigned to the next panel. Figure 5-7 illustrates how the process generates the new features and assigns them to entities. Before the entity exits the simulation at the *destroy* element, a new record capturing all the entity features (i.e., raw features and SL features) is added to the spreadsheet.

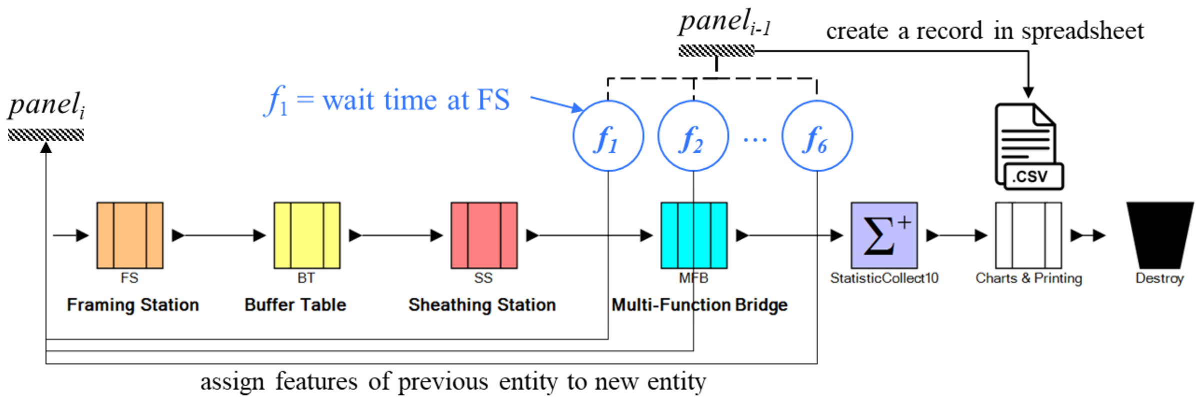


Figure 5-7. Process of creating and assigning new features within the simulation model

Moreover, the third dataset (i.e., 'SL length dataset') adds to the raw dataset a set of SL features that capture the number, length, and processing times of priori panels. These shop loading features are similar to the ones generated from the actual RFID timestamps, as described in Section 4.4.

5.4 Discussion and Comparison of ML Models

5.4.1 Experiment setup

Based on the sample data used to create the DES model and the resultant dataset (i.e., the dataset containing original features augmented with SL features generated using the simulation model), an experiment is conducted to examine the effect of those newly generated features on the predictive accuracy of ML models. An approach similar to the one used Chapter 4 in is used to prepare the data, develop the machine learning models, and compare their performance. The experiment comprises the development of an Extremely Randomized Tree (ERT) model [180] with optimized hyperparameters for the four datasets described in Table 5-2.

The target variable is set as the simulated total CT, which is now considered as the ground-truth against which the predictions obtained from the ML models are compared. Three performance comparisons are conducted: (1) to examine the improvement in CT predictability when adding 'SL wait features' to the dataset (i.e., using the second dataset), (2) to examine the improvement in CT predictability when adding 'SL length features' to the dataset (i.e., using the third dataset), and (3) to examine the improvement in CT predictability by adding all SL features to the dataset (i.e., using the fourth dataset). In each experiment, the dataset is divided into training and testing sets. The ERT model is developed using the training set and the performance is measured on the testing set using the measures described in Section 2.3.4. The testing set is always selected to be the records of one day (i.e., the one day where cycle times are to be predicted) for which the prediction of the model is tested while the training set is selected to be some previous lookback timeframe. As part of the model development exercise, it is necessary to determine the optimal lookback timeframe used for the training set for this type of time-sensitive prediction.

Table 5-2. Description of the datasets used in the experiment

Dataset Name	Description	Attributes
raw dataset	raw dataset that contains only the physical characteristics of wall panels – no SL features	'FirstReadDate', 'Length', 'Width', 'Type', 'Cutzone', 'Drillhole', 'Stud', 'DStud', 'LStud', 'MStud', 'SmallOpenings', 'LargeOpenings', 'BlockBacking', 'NailCount', 'NailLine', 'Sheetfull', 'SheetPartial'
SL wait dataset	raw dataset augmented with SL features capturing priori waiting time of WIP and priori starving time of workstations	'FirstReadDate', 'Length', 'Width', 'Type', 'Cutzone', 'Drillhole', 'Stud', 'DStud', 'LStud', 'MStud', 'SmallOpenings', 'LargeOpenings', 'BlockBacking', 'NailCount', 'NailLine', 'Sheetfull', 'SheetPartial', 'Wait@FS', 'Wait@BT', 'Wait@SS', 'starve@BT', 'starve@SS', 'starve@MFB'
SL length dataset	raw dataset augmented with SL features capturing the number, length, and processing time of priori panels (SL features of Chapter 4)	'FirstReadDate', 'Length', 'Width', 'Type', 'Cutzone', 'Drillhole', 'Stud', 'DStud', 'LStud', 'MStud', 'SmallOpenings', 'LargeOpenings', 'BlockBacking', 'NailCount', 'NailLine', 'Sheetfull', 'SheetPartial', 'WP', 'WPLength', 'WPTime'
SL full dataset	raw dataset augmented with all SL features described above	'FirstReadDate', 'Length', 'Width', 'Type', 'Cutzone', 'Drillhole', 'Stud', 'DStud', 'LStud', 'MStud', 'SmallOpenings', 'LargeOpenings', 'BlockBacking', 'NailCount', 'NailLine', 'Sheetfull', 'SheetPartial', 'Wait@FS', 'Wait@BT', 'Wait@SS', 'starve@BT', 'starve@SS', 'starve@MFB', 'WP', 'WPLength', 'WPTime'

The lookback timeframe refers to the number of days from the historical data used to train the model, as described in Section 4.5.1 and shown in Figure 2-8. This duration may have a substantial effect on the accuracy of the model, and it is largely dependent on the specific dataset used. After conducting several experiments, it is determined that a lookback duration of 5 days is optimal when developing the model. The effect of the lookback duration on the mean absolute percentage error (MAPE) is illustrated in Figure 5-8 for a few sample production days.

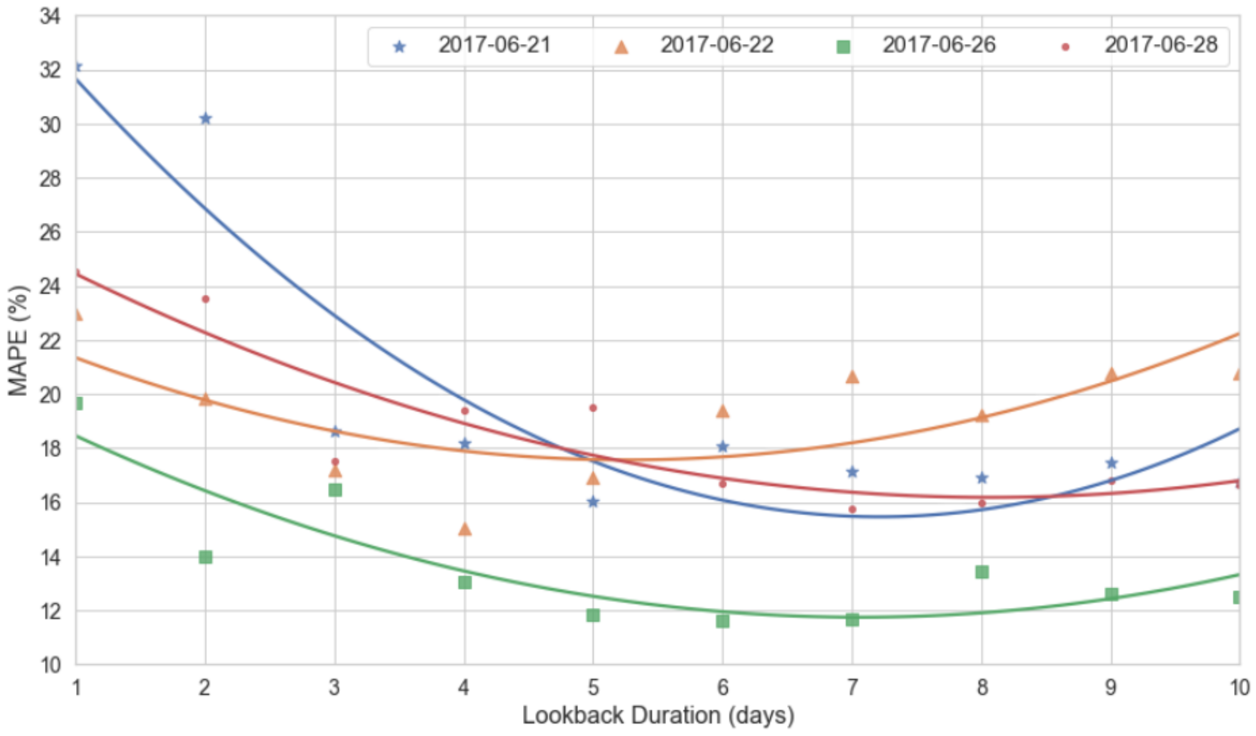


Figure 5-8. Effect of lookback duration on MAPE for sample production days

5.4.2 Results and discussion

Based on the above, ERT models are developed using the optimized parameters to predict the cycle time for the four datasets described in Table 5-2. The ERT optimized parameters include a lookback timeframe of five days, a number of trees to train of 400, and the number of features used to train the model is 90% of the total number of features for each dataset.

The results of the performance measures are summarized in Table 5-3 where the last row of the table shows the improvements achieved when SL features are added to the raw dataset. This improvement is measured as the reduction in MAPE when SL features are included. It is worth noting that the performance results represent the average values for running the experiments for ten randomly sampled days.

Table 5-3. Performance measures of ERT model on the four datasets

	Dataset 1 raw data	Dataset 2 'wait' SL	Dataset 3 'length' SL	Dataset 4 All SL features
Mean Absolute Error (MAE)	8.35	6.84	6.91	6.44
Median Absolute Error (MedAE)	6.95	5.77	6.01	5.18
Maximum Error (MaxAE)	21.46	19.44	18.16	17.66
Mean Absolute Percentage Error (MAPE)	21.0%	16.5%	17.3%	14.6%
Correlation Coefficient (CC)	0.53	0.66	0.67	0.70
Improvement on MAPE	n/a	4.5%	3.7%	6.4%

Examining the results of the performance measures in Table 5-3, it is observed that there is a consistent improvement in the model's predictive accuracy when shop loading features capturing some aspects of the real-time loading conditions of the job shop are added to the dataset. Dataset 1, the raw dataset with only the physical properties, shows inferior performance measures with an average MAPE value of 21%. In particular, the model performance improves by a reduction of 4.5% in MAPE score and an increase of 13% in CC when SL waiting time features (i.e., Dataset 2) are added; by a reduction of 3.7% in MAPE score and an increase of 14% in CC when SL length features (i.e., Dataset 3) are added, and when all SL features are added (i.e., Dataset 4) the reduction in MAPE is 6.4% and the increase in CC is 17%. All other performance measures exhibit similar improvement trends. The results of performance measures also show that all datasets exhibit similar trends by being positively skewed (i.e., the distribution of predicted CT has a long tail to the right). This can be inferred from the MAE scores being consistently greater than the MedAE scores for each of the four datasets. These results show consistency in performance improvement when shop loading features augment the raw dataset, which achieves the third objective in this thesis.

Moreover, it is observed that augmenting the raw dataset with different sets of SL features improves the performance of the machine learning models differently. Capturing the waiting time of prior panels and the starving time of workstations of priori production (i.e., as in the case of Dataset 2) is more beneficial than capturing the number, length, and processing time of priori panels (i.e., as in the case of Dataset 3). However, combining the two sets of features (i.e., as in the case of Dataset 4) offers the most significant improvement on the performance measures. Thus, the more information we can gather about panels immediately preceding the current panel in the production line, the better our ability to develop ML models with more accurate prediction power.

On another note, when using simulated datasets, the overall performance improvement (i.e., better predictability as observed by lower error measures and higher CC) is similar and comparable with the performance improvement observed using the actual data as described in Chapter 4. In both cases, the reduction in MAPE varies between 3% and 10% and the increase in CC varies between 13% and 56%. The difference in performance improvements between the simulated and actual data might be due to a few factors. One factor is that the development of the DES model may not accurately capture the actual nature of the production system because of the varying timeframe upon which the models were developed. The DES model was developed in 2019 while the available RFID data used to develop the ML models was made available for a different time period from 2015 to 2018. From this viewpoint, although the production system layout has not dramatically changed, numerous factors might have substantial effects on the system dynamics and how the collected data reflect the actual system performance. For example, the market supply and demand in the IBC industry are volatile as can be observed by the various cycles of high activity and recessions, as shown in Figure 4-7. Also, the skills of the available workers significantly affect the production. Skilled workers tend to perform manual tasks more efficiently than unskilled workers, a factor that was not implemented when developing the simulation model. Another factor is the nature of the generated SL features which might capture different aspects of loading conditions of the job shop. In essence, although capturing the waiting time of prior panels in the system and the starving time of workstations in the simulated environment seem intuitively informative, these generated features seem to capture the loading conditions differently than what SL features developed based on actual RFID data capture. Nevertheless, capturing real-time loading conditions of the job shop does indeed improve the prediction accuracy in both cases.

5.4.3 Conclusion

In conclusion, the simulation model can be used to propose specific mechanisms for collecting certain types of data that are otherwise not accounted for during the normal daily operations of the factory. The proposed collection of these specific features is suggested to enhance the development of ML predictive models and reduce the effort necessary for data preparation as well as rendering the generation process of relevant features more consistent and automated and less prone to subjectivity and customizations. In order to reap the full benefits of the simulated data, the simulation model needs to be developed based on actual data that reflect the current layout and operation of the production line. The simulation model, nevertheless, supports the hypothesis that real-time loading conditions of the job shop constitute a crucial element that should be included in the dataset in order to improve the predictive accuracy of any ML model. The loading conditions of the job shop can be represented by many forms and the right set of features that capture those conditions require a thorough understanding of the production process under investigation.

Chapter 6 Industrial Pipe Spool Fabrication Facility

This chapter describes the second case study where the proposed framework is applied to predict the fabrication duration in a pipe-spool fabrication shop. Following a brief introduction, the chapter is divided into five main sections. First, the manufacturing processes and work sequence are described. Then, data identification and collection are presented, followed by a detailed explanation of data cleaning and preparation tasks. The fourth section is dedicated to the development of engineered features that capture real-time loading conditions of the job shop. Lastly, the development, evaluation, and comparison of regression machine learning models are described.

Industrial construction projects such as petroleum refineries and oil and gas production facilities require intensive piping that connects a variety of equipment that is used to transport processed fluids and gases [181]. Because of time and space limitations, these industrial construction projects rely heavily on offsite prefabrication and preassembly. The prefabricated components of a piping system are called pipe spools and include pipes, fittings, elbows, flanges, and other components that differ widely in terms of material, shape, finish, and other properties. The number of pipe spools can be in the order of thousands in a typical-sized (i.e., \$200–\$300 million) industrial project [182]. These components are preassembled into modules before being shipped to remote sites for final assembly. The fabrication process takes place in a fabrication yard or shop and typically includes cutting, fitting, welding, quality control checking, stress relief, hydro testing, painting, and other surface finishing [65,182]. Before the fabrication of modules, the associated spools need to be ready; however, a large inventory of pipe spools would increase the requirements for storage and result in double handling. Determining the right time to start the fabrication of a pipe spool can be challenging as many factors affect the fabrication duration such as product specifications, workforce performance, shop loading and its capacity. Such conditions may lead to unbalanced production and impose challenges when predicting the completion time.

6.1 Process Description and Work Sequence

The case study analyzed in this chapter is an industrial pipe fabrication shop located in Alberta, Canada. Pipe spool fabrication involves several sequential phases as illustrated in Figure 6-1.

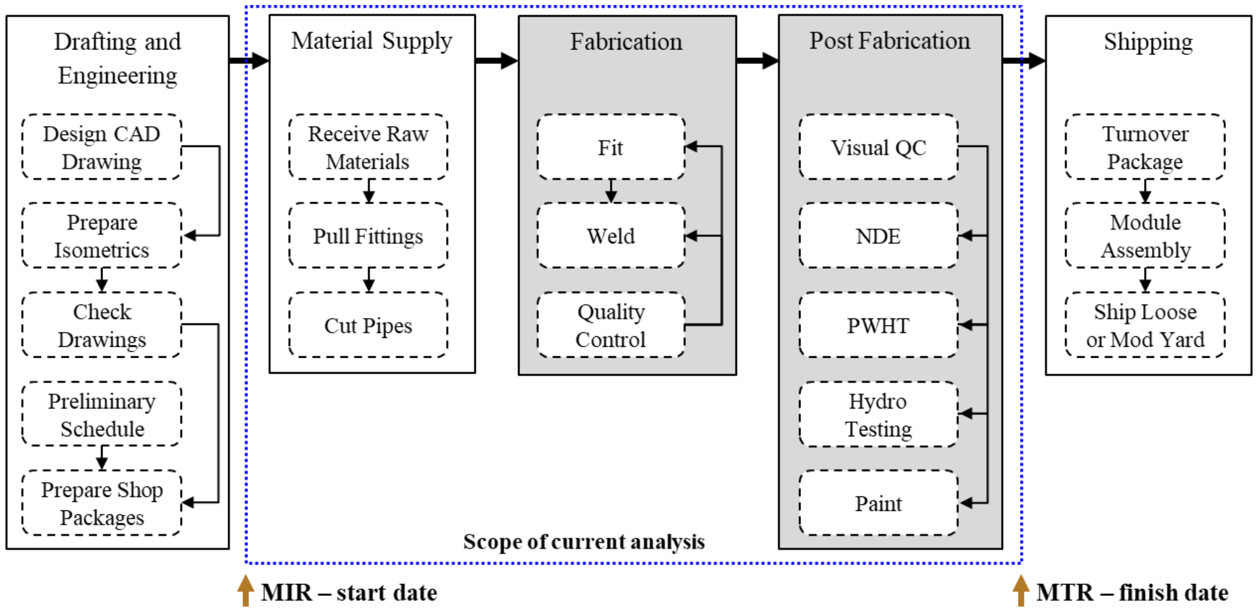


Figure 6-1. Main phases of pipe spool fabrication process

6.1.1 Drafting and engineering phase

This phase involves breaking the whole project into smaller work packages, each of which consists of the fabrication of a manageable pipe spool segment. The CAD drawings received from the designer are transformed into isometric drawings and typically contain a bill of materials (BOM) table. The isometrics drawings are detailed and then checked for consistency and errors. A preliminary construction schedule is prepared based on the fabrication packages. The BOM, shop drawings, and relevant information about the fabrication process are put together in what is known as the shop fabrication package. The material issue report (MIR) date records the date at which the raw materials for each package are required, and it represents the starting of the fabrication process of the pipe spool.

6.1.2 Material supply phase

Once the shop fabrication package is ready, the material supply phase starts with the "Receive Raw Materials" activity, which refers to the process of verifying, storing, and recording all the bulk material shipments of each project. The fabrication package cannot be issued until all the raw materials required for each package has a status of "available" in the material management database. A work crew picks up the required material based on the BOM by a) assembling all the non-pipe components (i.e., fittings such as elbows, valves, supports) on carts or pallets, and b) identifying and cutting the required pipes to the specified lengths using the cut summary sheet. Once all the fittings are placed on carts and the pipes have been cut and marked with a spool identifier, all materials are transported to the fabrication shop. Then, the remaining documentation including the isometric drawings and the package schedule is attached and delivered to the fabrication foreman in order to start the fabrication phase.

6.1.3 Fabrication phase

The primary value-added activities are carried out during this phase. The fabrication phase primarily consists of two sequential processes: fitting and welding. The fitting process requires assembling the components according to the drawings and specifications, and it typically involves pipe edge preparation (i.e., beveling and grinding), alignment and levelling of pipe assemblies, and tack welding of components to secure them in place. The welding process completes the welding of the fitted and partially finished subassemblies using various welding techniques. The entire pipe spooling process can be classified as either roll fitting and welding (RFW) or position fitting and welding (PFW). In RFW, the main pipe is rotated using a rolling machine, and the worker (i.e., welder) does not need to change their position to perform the task. In PFW, portions of the main pipe are too large to be turned by rolling machines in which case the subcomponents need to be aligned using pipe stands where the worker moves around to fit and weld the pipes. The process of PFW is generally more time-consuming than that of RFW. Hence, as an objective of pipe spool fabrication sequencing, considerable efforts are typically taken to reduce the required number of PFW [183]. The quality control (QC) procedures are typically carried out during both fitting and welding processes to check the alignment of the spools and the quality of the welding to ensure conformance with project specifications.

6.1.4 Post-fabrication phase

Once the fabrication phase is completed and before the spools can be delivered to the customer, post-fabrication processes need to be performed. It should be noted that post-fabrication tasks are performed either on all the welds or a random sample depending on project requirements. The material transfer report (MTR) date records when all fabrication and post-fabrication processes are complete, and it represents the finishing of the fabrication process of the pipe spool.

The post-fabrication phase typically starts with a detailed visual inspection of the pipe spool to check its dimensions, alignment, weld quality, and conformance with drawings. Then, nondestructive examination (NDE), also known as nondestructive inspection or testing, is performed. NDE methods rely upon the use of electromagnetic radiation, sound, and other types of signals to examine the pipe spools for integrity, composition, and conditions without affecting the serviceability of the spools. Common nondestructive examination methods include radiographic and ultrasonic testing [184], which are used to analyze the quality of the weld throughout its cross-section. Other NDE methods are described on the American Society for Nondestructive Testing website [185]. Post-weld heat treatment (PWHT) might be required depending on the type of material and pressures to which the pipe spool will be subjected during operation. PWHT is the process of reducing or redistributing the residual stress caused by the high-temperature gradients during the welding process. It is performed so that the internal structures of the metal are harmonized by reheating the weld and surrounding area of metal to below the lower transformation temperature at a controlled rate, keeping at that temperature for a specific time, and then cooling at a controlled rate [186]. Another type of testing that may be required is the hydro test, which involves pressurizing the pipes to a factor greater than its operating design pressure. This pressure is set by code (i.e., ASME III and ASME B31.1) and is often $1.5 \times P_{design}$ to ensure the welds can withstand operating loads [187]. Although this test is required for all spools subjected to a pressurized environment, the testing can be performed either before or after its installation in a module or when installed on site. When spools are to be welded together after being placed in the module or shipped for site assembly, hydro testing is performed after the final onsite assembly is completed. Once all the required inspections and tests are completed, pipe spools may be routed to the paint shop for sandblasting and coating. Each spool has specialized testing and painting requirements based on project specifications.

6.1.5 Shipping phase

After the post-fabrication phase, individual or batches of spools are marked as ready for delivery. Batching of spools leads to further uncertainty being introduced to the scheduling. The shipping phase includes tagging each spool with a unique identifier and preparing the turnover package that contains copies of all the documentation utilized during fabrication and testing. Once the turnover package is prepared, the spools are either shipped to the construction site individually (i.e., ship loose) or placed in a piping module in the module yard (i.e., mod yard) to be shipped as a complete structure.

6.2 Data Identification and Collection

6.2.1 Collection of raw data

The raw dataset analyzed in this study covers a period of approximately 18 months from September 2013 to February 2015 (based on actual fabrication dates as opposed to planned dates). The initial dataset is available as an MS Excel binary spreadsheet (.xlsb) provided by the industrial partner for analysis. In a previous study, an author developed a system to integrate data collection systems in the fabrication shop with heuristic scheduling rules provided by the manufacturing facility experts [12]. Figure 6-2 illustrates a data flow diagram where the information is accessed through three different data sources managed by two databases: Drafting Database, and Material Management Database.

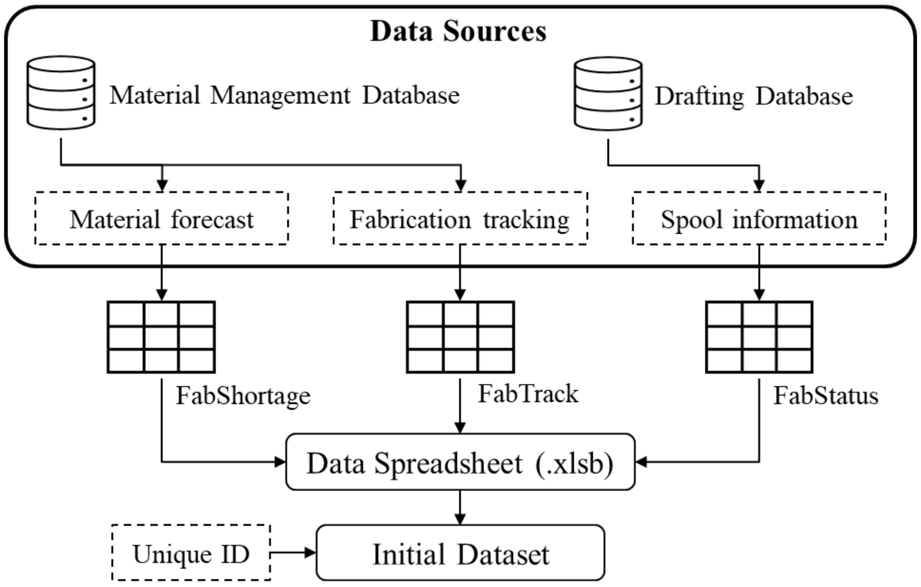


Figure 6-2. Data flow diagram from data sources to initial dataset

For the present analysis, the drafting database provides spool characteristics and some milestone tracking information that are exported to a spreadsheet called "FabStatus". The material management database provides a) material forecast and availability information that is exported to a spreadsheet called "FabShortage", and b) fabrication tracking and storing milestone tracking information that is exported to a spreadsheet called "FabTrack". These three data sources are combined together in a new data spreadsheet. A unique identifying number for each spool, called "Control Number", is created in the form of *xxx-yyyyyy* where *xxx* is the last three digits of the project number and *yyyyyy* is a serial number for a spool in that specific project. All redundant and repeated fields are removed at this stage and the material availability, number of items, and largest diameter of each spool is determined by consolidating information available at a component level. The output of this phase is the "Initial Dataset", where each row represents a single pipe spool along with all associated information.

6.2.2 Identification and description of collected data

The collected historical data is investigated with the goal of developing ML models that can predict the fabrication duration and estimate delivery date of future pipe spools. A pipe spool fabrication shop in Alberta has made data available for analysis that includes fabrication progress milestones that have been tracked. By looking at past fabrication durations, it is anticipated that useful relationships may be drawn to estimate the scheduling of future projects. The "Initial Dataset" spreadsheet has 68 attributes that have been obtained from the data collection tasks as described in the previous section. These attributes are described in Table 6-1 and organized into several categories. The seven fabrication processes (i.e., Fit, Weld, QC, NDE, PWHT, Hydrotest, and Paint) have similar attributes pertaining to the scheduling of these processes and are shown in full in order to present the complete composition of the dataset.

Table 6-1. Initial dataset attributes - column headers

Category	Attribute	Description
Spool Information	Control Number	Unique identifier of the spool
	Project	Project of which the spool is part
	FIWP	Field Installation Work Package of which the spool is part
	Priority	Assigned priority order of the FIWP
	Diameter Inches	Sum of the diameter inches to be welded
	Max Size	Diameter of the largest item in the spool
	Grade	Code for the material grade or type for the spool
	Weight	Total weight of the spool, computed at the drafting stage
	Surface Area	Surface area that a spool occupies
	No. of items	Total number of components making up the spool
Fabrication Details	Status	Fabrication status of a spool (i.e., On Hold or In Fab)
	Bay #	Number of the shop bay where the spool is fabricated
	ECN Column	Engineering Change Notice (contains subcontractor info)
	Hold Details	Any details regarding a client hold
	Drawing Check Date	Date the spool drawing has been checked
Issuing Schedule	Material Available?	Yes/No attribute derived using Material Management Data
	RAS Date	Required At Site Date - spool completion deadline
	Expected Ship Date	Date computed after a spool fitting has been completed
	MIR Number	Material Issue Report number
	Planned MIR Date	Scheduled Material Issue Report date of the spool, based on RAS date
	Actual MIR Date	Actual Material Issue Report Date – represents the start date of the spool
	Planned Matl Issue Date	Date by when material is required to be prepared
	Actual Matl Issue Date	Actual date when material preparation was complete
	Location	Spool fabrication location (either Bay# or Subcontractor)
Fitting Schedule	Scheduled Duration	Scheduling and tracking attributes for the Fitting process
	Scheduled Start Date	
	Actual Start Date	
	Scheduled Finish Date	
	Actual Finish Date	
	Actual Duration	
Welding Schedule	Scheduled Duration	Scheduling and tracking attributes for the Welding process
	Scheduled Start Date	
	Actual Start Date	
	Scheduled Finish Date	
	Actual Finish Date	
	Actual Duration	

Category	Attribute	Description
QC Schedule	Scheduled Duration Scheduled Start Date Actual Start Date Scheduled Finish Date Actual Finish Date Actual Duration	Scheduling and tracking attributes for the Visual Inspection process
PWHT Schedule	Scheduled Duration Scheduled Start Date Actual Start Date Scheduled Finish Date Actual Finish Date Actual Duration	Scheduling and tracking attributes for the Post-Weld Heat Treatment process
NDE Schedule	Scheduled Duration Scheduled Start Date Actual Start Date Scheduled Finish Date Actual Finish Date Actual Duration	Scheduling and tracking attributes for the Non-Destructive Examination process
Hydro Testing Schedule	Scheduled Duration Scheduled Start Date Actual Start Date Scheduled Finish Date Actual Finish Date Actual Duration	Scheduling and tracking attributes for the Hydro Testing process
Paint Schedule	Scheduled Duration Scheduled Start Date Actual Start Date Scheduled Finish Date Actual Finish Date Actual Duration	Scheduling and tracking attributes for the Painting process
Transfer Dates	MRR Date	Material Receiving Report - represents a transfer of the spool between business units
	MTR Date	Material Transfer Report - represents the completion date of the spool

The scheduled dates are computed from the preliminary schedule developed during the drafting phase (RAS Date), whereas the actual dates are obtained from the tracking databases. Table 6-2 and Table 6-3 show example records from the spool information and issuing schedule categories, respectively.

Table 6-2. Spool information example records

Control Number	Project	FIWP	Priority	Diameter Inches	Max Size	Material Grade	Weight	Surface Area	No. of items
130 - 010005	3215130	MD-02-52	10	12	20.00	G	1246.23	96.63	2.00
140 - 030047	3215140	W04-PM-1103-52	08	33.5	6.00	A	255.06	14.40	11.00
290 - 070007	3215290	D-PIP-13-3-SL	19	8	12.00	E	75.07	12.30	4.00
461 - 082082	3215461	200NLA201-15E-08	07	28	4.00	A	471.40	47.95	5.00

Table 6-3. Issuing schedule example records

Material Available?	RAS Date	Expected Ship Date	MIR Number	Planned MIR Date	Actual MIR Date	Planned Matl Issue Date	Actual Matl Issue Date	Location
Y	Jun 25 '14	May 22 '14	MIR-130019	May 14 '14	Apr 09 '14	May 22 '14	Apr 28 '14	Bay # 2
Y	Sep 27 '14	Oct 23 '14	MIR-140056	Aug 18 '14	Sep 17 '14	Aug 25 '14	Sep 19 '14	Academy Fabrication
Y	Jun 04 '14	Jan 21 '14	MIR-290074	Apr 29 '14	Nov 28 '13	May 06 '14	Jan 09 '14	Bay # 1
Y	Jul 15 '14	Jul 29 '14	MIR-461112	Jun 03 '14	Jun 11 '14	Jun 10 '14	Jul 03 '14	Bay # 4

It is crucial to identify unreliable or inconsistent data and carefully work around it. The attributes above have been carefully examined to determine which attributes are to be included in the current analysis. Attributes are discarded if they a) contain many missing data, b) contain redundant information that is available within other attributes, c) lack consistency in terms of content and value (i.e., data have been collected inconsistently over time between projects), or d) adds no value to the current analysis (i.e., heuristically they do not affect the prediction of fabrication duration). To avoid analyzing incomplete or incorrect data, heuristic approaches are followed to exclude certain columns and records; then, comprehensive data cleaning and preparation is carried out as described in the following section.

6.3 Data Preparation and Exploration

The raw dataset for this study is obtained in the form of a spreadsheet that contains 14,829 rows (i.e., records) and 68 columns (i.e., features or attributes). The dataset includes a mixture of categorical and numerical attributes. Although these attributes are grouped into several categories for the ease of referencing as described in Table 6-1, they can be classified primarily as being either spool identification and physical descriptions (e.g., control number, max size, weight), or tracking dates (e.g., planned and actual MIR and MTR dates).

6.3.1 Data transformation

Date attributes present difficulties for data processing and might not be meaningful for the purpose of predicting durations. The difference between two date attributes is more useful because it indicates the duration to be investigated. The first step is to calculate the actual fabrication duration, *FabDur*, by using actual dates of major milestones. Those milestones include (1) the actual MIR date ,which marks the start of the overall fabrication process; (2) the fit actual finish date, $Fit_{actualF}$, which indicates the end of the fitting process, prior to the start of welding; (3) the QC actual finish date, $QC_{actualF}$, which indicates the finish of quality control visual inspection prior to the start of post-fabrication processes; and (4) the actual MTR date, which marks the end of the fabrication process. Fabrication duration is then calculated as per Equation 6-1.

$$FabDur = MTR\ date - MIR\ date : MIR \leq Fit_{actualF} \leq QC_{actualF} \leq MTR \tag{6-1}$$

The condition imposed by the above equation is for consistency and practicality where the total fabrication duration needs to be valid and non-negative. Given missing date fields, which are denoted by the symbol '/' in the dataset, or inconsistent dates (e.g., $QC_{actualF} > Fit_{actualF}$), applying the above equation will produce data cells with an error or negative value, which are marked as invalid and are later removed from the dataset. Moreover, planned durations of the major milestones described above are calculated based on the planned finish dates. For example, the planned duration for the fitting process is calculated as the difference between the scheduled MIR and scheduled fitting process dates. It should be noted that determining the planned durations can be a challenging task since there exist many missing values. In such cases where one of the planned dates is missing, it was reasonable to assume the planned duration to be equal to the median value of actual durations. Four planned durations are calculated: MIR→Fit, Fit→QC, QC→MTR, and MIR→MTR. These planned durations will form one group of attributes that are used to experiment with ML model development.

6.3.2 Data cleaning

After calculating the total fabrication duration and the planned duration of major milestones, the following steps describe how the records and attributes are further prepared and cleaned for analysis (refer to Table 6-1 for attribute descriptions).

1) Records:
    a. Records with invalid total fabrication durations are removed.
    b. Records where the attribute "Diameter Inches" is missing or equal to zero are removed.
2) Attributes:
    a. Only "Control Number" is kept as a unique identifier for each pipe spool. Fabrication and other spool information are discarded as being irrelevant to the current analysis.
    b. Only actual and planned dates for MIR, fitting process, QC process, MTR, and RAS are kept for the purpose of calculating planned durations. These date attributes are discarded after completing the calculations and validation.

c. All data pertaining to the post-fabrication processes (e.g., PWHT and Paint) are replaced with a "Yes/No" attribute to indicate if the respective process has been completed or not for the specific pipe spool. Planned dates for post-fabrication processes are mostly missing and actual dates are not available for new products for which the duration is to be predicted.

6.3.3 Data descriptions

In this section, descriptive statistics are presented with respect to the physical numerical attributes of pipe spools and the planned and actual fabrication durations in order to gain a better understanding of the data and the process it represents. As can be observed in Table 6-4, most minimum values of physical attributes indicate unrealistic values, which might be due to inconsistency in collecting raw data (e.g., erroneous measurement, or wrong data entries). This is an indicator that these attributes may not be very informative for the purpose of predicting fabrication time. Intuitively, "Diameter Inch" is the attribute that would affect the fabrication duration the most as the time spent on fitting and welding is largely determined by how many inches need to be worked on. However, other physical attributes might affect the time of material storage and transportation as well as the type of welding material and techniques.

Table 6-4. Descriptive statistics of physical numerical attributes and fabrication durations

	mean	std	min	25%	50%	75%	max
Diameter Inches	19.93	25.42	0.50	6.00	12.00	24.00	552.00
Weight (lbs)	436	1171	0.36	40	103	282	26304
Surface Area	28.87	54.68	0.05	4.98	10.86	26.91	1020.39
No of items	4.25	2.83	1.00	2.00	3.00	5.00	24.00
Max Size	4.72	4.78	0.50	2.00	3.00	6.00	36.00
Planned MIR to Fit	14.34	1.55	8.0	13.0	14.0	15.0	21.0
Planned Fit to QC	2.96	0.33	-5.0	3.0	3.0	3.0	4.0
Planned QC to MTR	12.49	4.68	6.0	10.0	11.0	15.0	32.0
Planned MIR to MTR	29.88	4.65	19.0	27.0	29.0	32.0	53.0
Fab Duration	41.49	22.28	1.00	25.00	39.00	53.00	169.00

The values for actual fabrication durations average around 42 days with a median value of 39 days and an extreme maximum value of 169 days. These values indicate a positively skewed distribution of the actual fabrication durations (i.e., an empirical distribution plot with a long tail to the right). Figure 6-3 shows the distribution of actual fabrication durations (i.e., Fab Duration) where it can be observed that most durations fall in the range of 15 to 55 days. Also shown in the figure is the distribution of planned fabrication durations (i.e., Planned MIR to MTR) with a much smaller spread as compared to the actual durations. By comparing actual and planned durations, one can observe the over-optimistic approach followed when planning pipe spool fabrication as compared to the actual shop performance. Therefore, adopting a data-driven approach would provide more robust and realistic schedules. Such an approach would also adapt to future changes in the performance and allow for faster and smarter shop control.

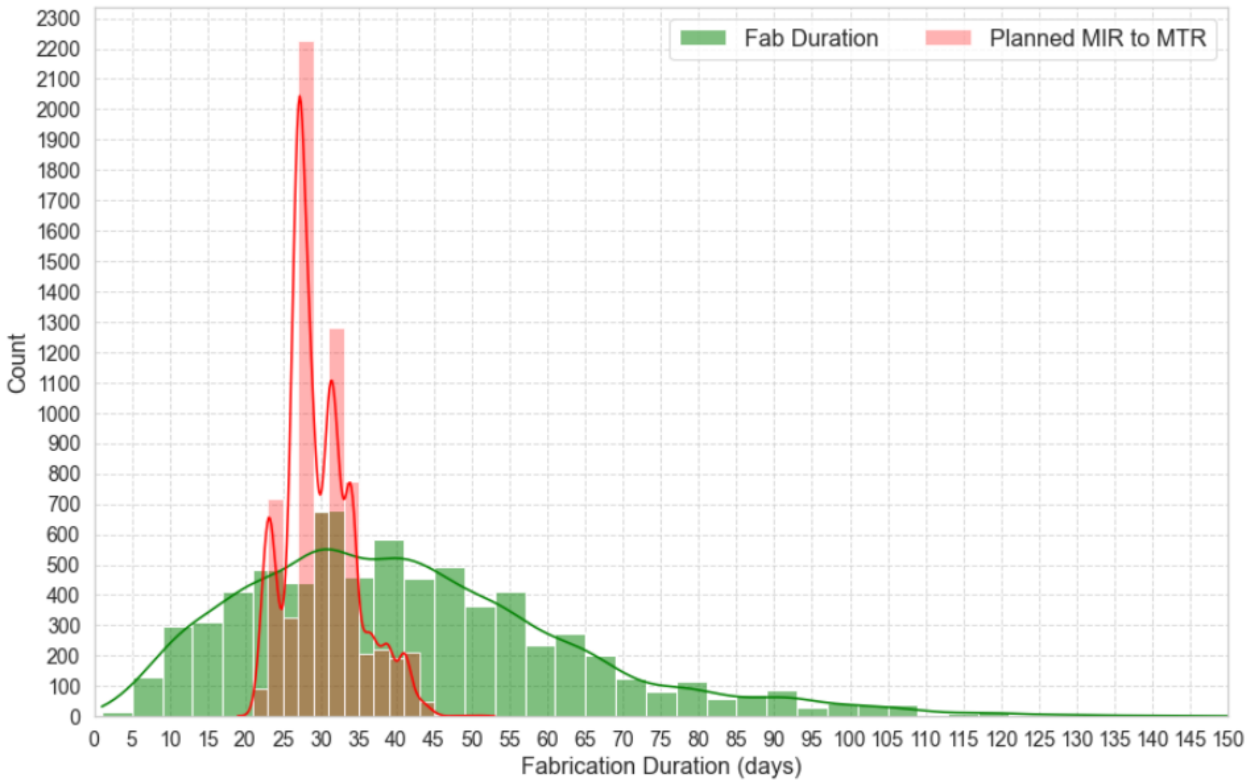


Figure 6-3. Actual and planned total fabrication durations (days)

As part of the data preparation phase, and in addition to the insights gained from the descriptive statistical analysis, the fabrication dates have been examined for logical sequencing of events. Due to inconsistency in data collection and recording, dates for completing some activities are recorded as having occurred after some successor activities. For example, the fitting process can only start after the raw material has been delivered to the fabrication shop, which in turn only occurs after the MIR is issued. In a large number of instances, the MIR date has a later date than the start of the fitting process, which cannot practically happen. Similar observations are noted for logical sequencing of post-fabrication processes. For this reason, several intermediate date attributes (i.e., between MIR and MTR) have been discarded as previously described. It should be noted that arriving at the tidy dataset as described above is a two-way approach where a thorough understanding of each activity and planning and control methods employed at each stage is required. Also, examining the raw data has added to a deeper understanding of the underlying processes and activities.

6.4 Feature Engineering

6.4.1 Shop utilization features

Similar to the first case study investigated in Chapter 4, information about the loading conditions of the job shop is believed to have a major impact on the overall fabrication duration. It is anticipated that shop loading at the time a new pipe spool starts production affects processing priorities, waiting time between fabrication phases, and the congestion levels at different workstations. To be able to quantify the shop loading conditions at the start date of fabrication, a measure of the shop utilization is developed based on the current volume of work being performed at the job shop. The four available physical characteristics of a pipe spool (i.e., Diameter Inches, Weight, Surface Area, and No. of Items) are used each in combination with fabrication tracking information to derive four shop loading (SL) parameters. Each parameter represents the shop utilization at the start date of fabrication based on one physical feature.

For the diameter inch (DI) shop loading parameter (DSL), it is computed using Equation 6-2 for a new pipe spool *k* by adding the DI of all the pipe spools that are still in fabrication at the time the new spool starts its production by the actual start date (MIR). The diameter inch count is commonly used as a measure of the amount of welding labor required. For example, the amount of labor required to weld a standard 12-inch carbon steel pipe is expressed as 12 DI. Diameter inch is the main factor used for the purpose of scheduling, cost estimating, and productivity measure [65,67].

$$DSL_k = \sum_{i=1}^{n} DI_i : MIR_k \geq MIR_i \ and \ MIR_k \leq MTR_i \qquad 6\text{-}2$$

where: DSL_k is the “Diameter Inches” shop loading of spool *k*;

MIR_k is the actual fabrication start date of spool *k*;

MIR_i is the fabrication start date of spool *i*;

MTR_i is the fabrication finish date of spool *i*;

DI_i is the Diameter Inches of spool *i* at the start date; and

n is the number of pipe spools *i* when spool *k* starts production.

Other shop loading features, including “Weight” shop loading (WSL), “Surface Area” shop loading (SSL), and “No of Items” shop loading (ISL), are calculated using a similar approach. The reason for deriving several shop loading features is to later examine which physical property has the largest effect on the predictability of fabrication duration. Before being added to the dataset, the four derived shop loading features (i.e., DSL, WSL, SSL, and ISL) are normalized so that the values lie between 0 and 1 [90]. Equation 6-3 shows how the DSL attribute is normalized:

$$Norm.DSL_k = \frac{DSL_k - DSL_{min}}{DSL_{max} - DSL_{min}} \tag{6-3}$$

In addition to the derived shop loading features described above, a shop utilization ratio (SUR) is calculated as the ratio between the actual DSL and the planned DSL as per Equation 6-4:

$$SUR_k = \frac{actual\ DSL_k}{palnned\ DSL_k} \tag{6-4}$$

The planned DSL_k is calculated when spool k starts production as per Equation 6-2 based on the planned MIR date and the planned finish date (i.e., RAS date). Essentially, this ratio reports how busy the job shop actually is in comparison to what has been planned, which is believed to capture the agreement/difference between the actual and planned scheduling of the work. A graphical representation of computing a shop loading feature is illustrated in Figure 6-4 showing three pipe spools that are in production at the fabrication start date of a new pipe spool.

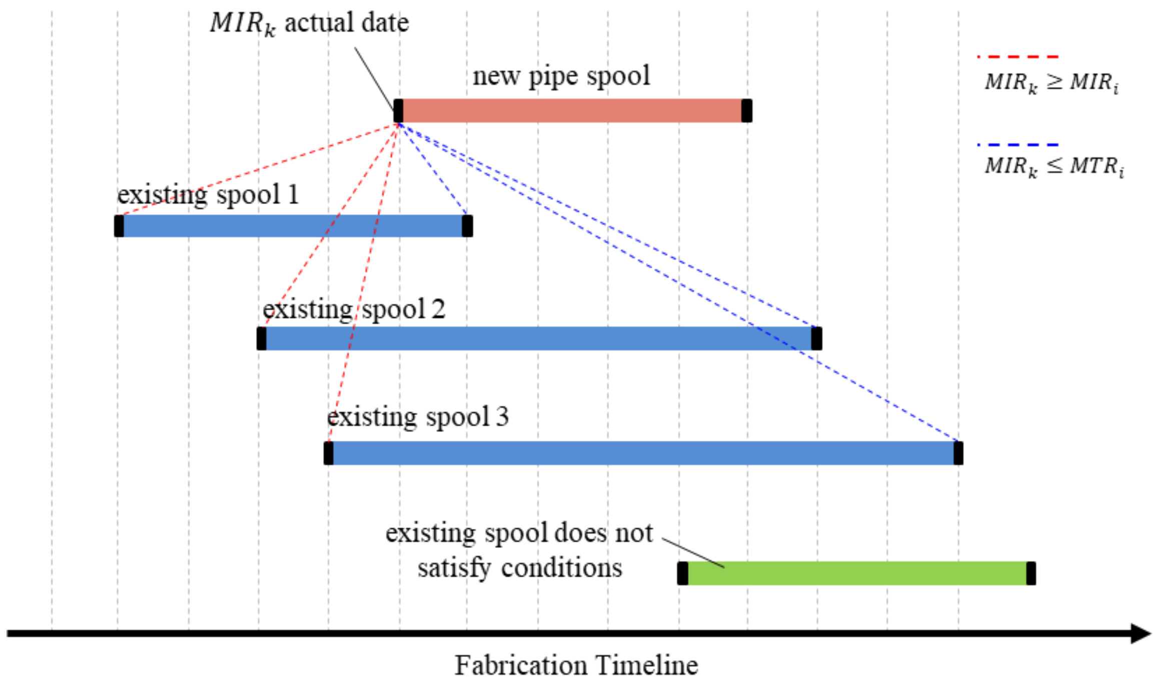


Figure 6-4. Computation of SL for a new pipe spool

6.4.2 Features categorization

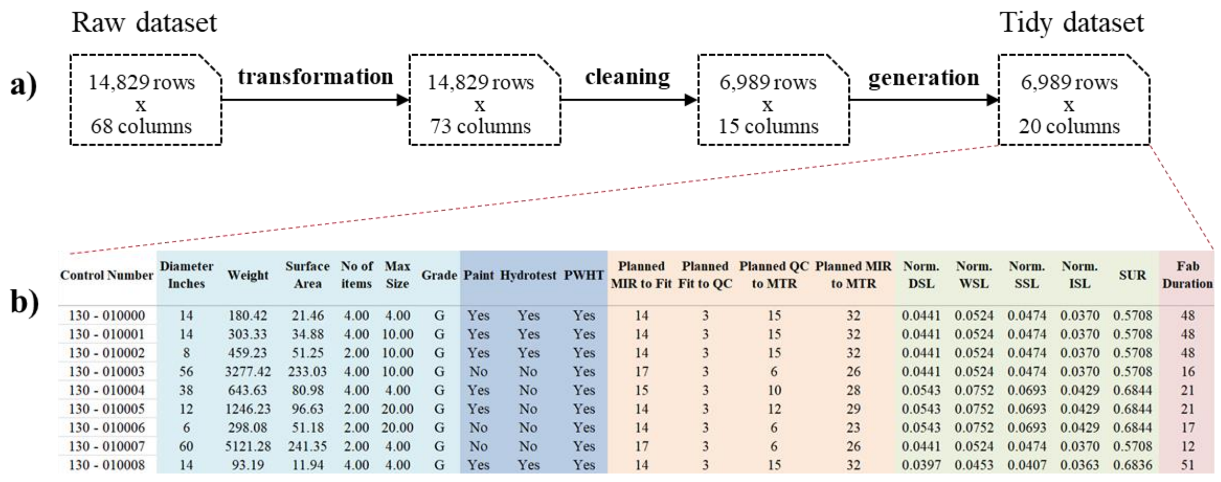
After applying the cleaning and preparation steps as previously described, five engineered features capturing shop utilization are derived as presented in Section 6.4.1. For consistent referencing, and to be able to examine the effect of different subsets of features on the prediction accuracy of fabrication durations, features are categorized into four groups:

a) Features pertaining to the physical properties that describe the dimensional and composition characteristics of the pipe spool. Those features include *"Diameter Inches"*, *"Weight"*, *"Surface Area"*, *"No of Items"*, *"Max Size"*, and *"Grade"*.

b) Features pertaining to post-fabrication phase indicating whether or not a post-fab process is applied to the spool. Those binary features include *"Paint"*, *"Hydrotest"*, and *"PWHT"*.

c) Features pertaining to the planned fabrication durations that computed the planned total duration *"MIR to MTR"* and its subprocesses *"MIR to Fit"*, *"Fit to QC"*, and *"QC to MTR"*.

d) Features pertaining to shop loading that capture the real-time loading conditions of the job shop (i.e., shop utilization) depending on different physical characteristics. Those include *"Norm. DSL"*, *"Norm. WSL"*, *"Norm. SSL"*, *"Norm. ISL"*, and *"SUR"*.



Control Number	Diameter Inches	Weight	Surface Area	No of items	Max Size	Grade	Paint	Hydrotest	PWHT	Planned MIR to Fit	Planned Fit to QC	Planned QC to MTR	Planned MIR to MTR	Norm. DSL	Norm. WSL	Norm. SSL	Norm. ISL	SUR	Fab Duration
130 - 010000	14	180.42	21.46	4.00	4.00	G	Yes	Yes	Yes	14	3	15	32	0.0441	0.0524	0.0474	0.0370	0.5708	48
130 - 010001	14	303.33	34.88	4.00	10.00	G	Yes	Yes	Yes	14	3	15	32	0.0441	0.0524	0.0474	0.0370	0.5708	48
130 - 010002	8	459.23	51.25	2.00	10.00	G	Yes	Yes	Yes	14	3	15	32	0.0441	0.0524	0.0474	0.0370	0.5708	48
130 - 010003	56	3277.42	233.03	4.00	10.00	G	No	No	Yes	17	3	6	26	0.0441	0.0524	0.0474	0.0370	0.5708	16
130 - 010004	38	643.63	80.98	4.00	4.00	G	Yes	No	Yes	15	3	10	28	0.0543	0.0752	0.0693	0.0429	0.6844	21
130 - 010005	12	1246.23	96.63	2.00	20.00	G	Yes	No	Yes	14	3	12	29	0.0543	0.0752	0.0693	0.0429	0.6844	21
130 - 010006	6	298.08	51.18	2.00	20.00	G	No	No	Yes	14	3	6	23	0.0543	0.0752	0.0693	0.0429	0.6844	17
130 - 010007	60	5121.28	241.35	2.00	4.00	G	No	No	Yes	17	3	6	26	0.0441	0.0524	0.0474	0.0370	0.5708	12
130 - 010008	14	93.19	11.94	4.00	4.00	G	Yes	Yes	Yes	14	3	15	32	0.0397	0.0453	0.0407	0.0363	0.6836	51

Figure 6-5. a) Preparation steps from Raw to Tidy dataset, b) Headers and example records of the tidy dataset

The tidy dataset consists of 6,989 records and 20 attributes. Figure 6-5 illustrates the change in dataset dimensions during the development of the tidy dataset. The bottom part of the figure shows the final complete dataset that is ready for analysis. As described above, the features are grouped into four sets with the target attribute conventionally located as the last column.

6.5 Machine Learning Models for Regression

Several experiments are conducted with the aim of developing an optimal ML regression model to predict the fabrication durations in the pipe spool shop. There are various available ML models that are suitable for regression analysis; however, due to recent advances, easier interpretability, and successful implementations in many fields, the focus in this case study will mainly be on tree-based ensemble algorithms including random forests (RF), extremely randomized tree or extra trees (ERT), and gradient boosted decision trees (GBDT). RF and ERT belong to a class of ensemble models called bagging, while GBDT is a boosting algorithm [188,189]. In addition, the prediction results of these models are compared to the use of simple averaging of actual historical durations to estimate future fabrication durations. Simple averaging (i.e., use of Little's Law) is used as a benchmark as it is used as a heuristic rule in practice.

Before examining the various ML models, it is necessary to carry out a set of experiments to evaluate the viability the performance of different subsets of attributes as described in the next section. Attribute subsets are constructed to examine the effect of inclusion and exclusion of various levels of information on the performance of the ML models. The most promising attribute sets are carried forward for further experimentation. Subsequently, the hyperparameters of the best performing model on the optimal attribute dataset are tuned to achieve the best performance.

6.5.1 Evaluation of attribute subsets

Presented in Table 6-5 are the different attribute subsets created for experimentation. The subsets present all possible combinations of the different attribute groups described in Section 6.4.2. Each dataset combination has been assigned an experiment number with a brief description.

Table 6-5. Attribute subset experiments

Subset No.	Attribute Description	Spool Physical Properties	Post-Fab Processes	Shop Loading Attributes	Planned Durations	Fabrication Duration
1	Dataset with all attributes including SL and planned durations	✓	✓	✓	✓	✓
2	Dataset with only spool information	✓				✓
3	Dataset with only spool information and post-fab processes	✓	✓			✓
4	Dataset with only spool information and SL	✓		✓		✓
5	Dataset with only spool information and planned durations	✓			✓	✓
6	Dataset with all attributes except planned durations	✓	✓	✓		✓
7	Dataset with all attributes except SL	✓	✓		✓	✓
8	Dataset with spool information, SL, and planned durations	✓		✓	✓	✓

The performance measures, described in Section 2.3.4.1, are used to compare the various combination of attributes defined as subsets in Table 6-5. The subsets are evaluated based on a default RF model with 100 trees, the mean square error (MSE) as the criterion to measure the quality of a split, and the maximum possible depth of the trees. In these preliminary experiments, each dataset has been split into 80% training set and 20% testing set. The model is developed using the training set and the performance is measured on the testing set. The results of attribute subset performance measures are summarized in Table 6-6.

Table 6-6. Performance measures of attribute subsets

Subset No.	1	2	3	4	5	6	7	8
Mean Abs. Error	9.05	14.89	13.84	10.02	13.72	9.36	13.51	9.15
Median Abs. Error	5.67	10.93	9.88	6.36	10.07	5.99	9.76	5.79
Maximum Error	108.17	92.25	103.05	103.73	101.82	105.02	100.81	109.18
Mean Abs. Percentage Error	30.48%	59.8%	52.04%	36.11%	51.2%	31.69%	50.24%	30.94%
Correlation Coefficient	0.77	0.47	0.54	0.74	0.55	0.76	0.57	0.77

The results show a consistent behavior, except for the maximum error, for all performance measures across all datasets as these measures are correlated to each other. Therefore, any measure can be used for comparison purposes. One important result that can be observed from Table 6-6 is the improved performance when the shop loading attributes are included in the dataset. This can be noted by the higher correlation coefficients (i.e., around 0.76) and the lower error measures (e.g., MAPE is about 32%) for the Datasets 1, 4, 6, and 8 where the SL attributes are included. When SL attributes are not present in the dataset, the results are inferior (e.g., MAPE as high as 60%) from which we can conclude that SL attributes are crucial for the development of optimal ML models.

Out of the four subsets where the shop loading attributes are included, the two with the slightly better performance measures (i.e., higher CC and lower MAPE) are Subsets 1 and 8, which are then used to further evaluate the effect of each individual shop loading attribute on the model performance. By selecting relevant SL attributes and discarding less-informative ones, the model performance is improved, and the computational requirements to develop and maintain the model are decreased.

According to the binomial theory, the number of *k-combinations* for all k is the number of all possible subsets of a set of *n* numbers; therefore, the total number of all possible combinations of the five SL attributes is found to be 31 subsets as per Equation 6-5.

$$\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{k} = 2^n - 1 \qquad \text{6-5}$$

Experiments are carried out to find the optimal combination of SL attributes using the two candidate Subsets 1 and 8. The performance measures are calculated for each trial and compared in a similar manner as is shown in Table 6-6. Out of the 62 possible subsets (i.e., 31 possibilities for each of Subsets 1 and 8), it is observed that the change in performance measures is moderate with a CC ranging between 0.74 and 0.77, while MAPE ranges between 37% to 30%. When only one SL attribute is included, the results are inferior in general (worse results when using only SUR) indicating that more than one SL attribute should be included in the subset to arrive at a better performance. The effect on model performance when including post-fab processes (i.e., the difference between Subsets 1 and 8) is negligible; therefore, to reduce the dataset dimensionality, Subset 8, which excludes post-fab attributes, is used for further analysis since the post-fab attributes do not seem to be very informative in predicting fabrication durations.

After examining the different combinations of shop loading attributes, it is observed that discarding the WSL attribute does not jeopardize the overall performance. Therefore, it was decided to keep the other four attributes (i.e., DSL, SSL, ISL and SUR) in Subset 8 and carry the resulting subset forward to develop and compare the various ML models. The resulting subset is referred to hereinafter as Subset 8A, and the corresponding performance measures are calculated as 0.77 and 30.6% for the CC and MAPE, respectively.

6.5.2 Application and comparison of models

For the Subset 8A described in the previous section, three tree-based ensemble algorithms are applied to predict the fabrication durations (i.e., Random Forests, Extremely Randomized Trees, and Gradient Boosting Decision Trees). Experiments to tune the hyperparameters of each of those models are carried out to achieve an optimal result. The three algorithms are employed using Scikit-Learn, a Python open-source library for machine learning [174], and the variable names shown below in parentheses refer to the parameters used in the library.

6.5.2.1 Tuning of hyperparameters

In an RF model, each tree in the ensemble is constructed from sample records drawn from the training set with replacement (i.e., a bootstrap sample). Also, when splitting each node, the optimum split is determined either using all input features or a random subset of features. In an ERT model, unpruned trees are built using the whole training set as opposed to bootstrap samples as in the case of RF. Moreover, the nodes are split fully at random through which "the explicit randomization of the cut-point and attribute combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods" [180] that uses locally optimized cut-points. In an GBDT model, decision trees are added to the ensemble one at a time where a subsequent model is fit to correct the prediction errors made by a prior model. Therefore, the final predictor includes combining "weak" models sequentially where the loss gradient of a differentiable loss function is minimized at each step.

All algorithms have three main and common parameters to configure: 1) the number of trees of the ensemble (`n_estimators`), 2) the size of the random subset of features (`max_features`), and 3) the minimum number of samples required for splitting a node (`min_samples_split`). For each of these parameters, several experiments are carried out to obtain the optimal values by exhaustive search over the range of heuristically selected values while holding all other parameters constant. For the number of trees, experiments are conducted for values ranging from 100 (i.e., default value) to 3000 trees in order to obtain a value that is large enough to ensure convergence of the ensemble effect for the present dataset. For the maximum number of features randomly selected at each node, Breiman [130] suggested to use one-third of the features while Geurts et al. [180] argues that using all the features results in better models. Accordingly, an experiment is conducted to examine the range of values between the two suggested limits (i.e., between 6 and 17 features for the present dataset Subset 8A). As for the minimum number of samples required to split an internal node, values between 2 (i.e., default value) and 12 are examined. In general, larger values lead to smaller trees, higher bias, and smaller variance; however, the optimal value largely depends on the level of noise in the dataset. The percentage of training set used to fit each individual tree (i.e., `max_samples` for RF, and `subsample` for GBDT) are examined for values ranging from 70% to 100%. It is observed that values above 90% yield substantially better performance for both the RF and GBDT models. This parameter is not applicable for the ERT model as the whole training set is used to fit the model. The results of the hyperparameter tuning experiments are summarized in Table 6-7 where these optimum values are used to fit the final models and compare the results in the following section.

Table 6-7. Summary of Hyperparameter Experiments

	range of tested values	RF	ERT	GBDT
`n_estimators`	100 – 3000	200	400	3000
`max_features`	6 – 17	16	16	15
`min_samples_split`	2 – 12	2	2	4
`max_samples/subsample`	70% – 100%	97%	n/a	93%

As an illustrative example of one of the experiments, Figure 6-6 shows a grid of the MAPE results for each possible combination of the two parameters, (`max_features`) and (`min_samples_split`), for each of the three models. The size of the circle indicates the relative value of MAPE where smaller circles correspond to better performance measures. The circles highlighted in red represent the six minimum MAPE for different combinations of the two parameters for the Subset 8A dataset. It can be observed that the performance of both the RF and ERT models are more consistent compared to the GBDT model.

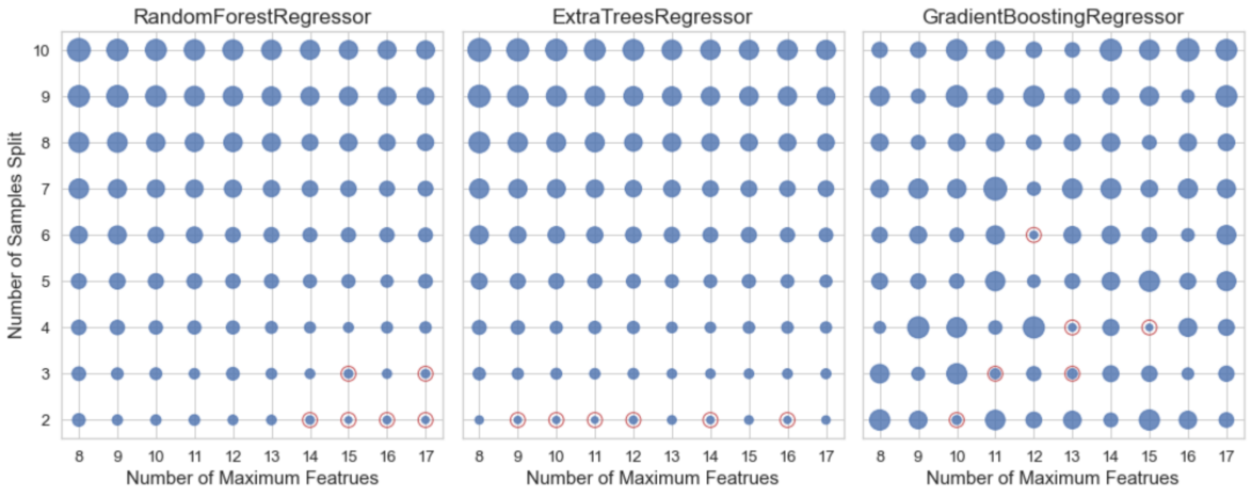


Figure 6-6. MAPE for combined number of max. features and min. samples to split

6.5.2.2 Comparison of model performances

Given the optimum hyperparameters obtained and described in the previous section, the three models (i.e., RF, ERT, and GBDT) are trained and tested using a 5-fold cross-validation technique. In addition, in order to compare the computational efficiency, the average training and testing times for each model are also calculated. For accurate, fair, and consistent comparison of the time requirements, the models are compared by setting the number of trees to 200. This is because the number of trees (i.e., `n_estimators` parameter) has the largest effect on the time required for training any tree-based model. Summarized in Table 6-8 are the performance measures, described in Section 2.3.4.1, for the three models along with the time required for training and testing. Also, the last column presents the error measures for using simple averaging (i.e., using the average of actual historical durations, which is equal to 41 days) as an estimator of fabrication duration for future pipe spools.

Table 6-8. Performance measures for the final fit models

	RF	ERT	GBDT	Simple Averaging
Mean Absolute Error (MAE)	9.50	8.93	10.38	16.35
Median Absolute Error (MedAE)	6.11	5.06	7.05	13.96
Maximum Error (MaxAE)	89.62	76.78	88.09	66.95
Mean Absolute Percentage Error (MAPE)	28.6%	26.6%	31.6%	60.3%
Correlation Coefficient (CC)	0.74	0.74	0.71	n/a
Training Time (seconds) @ 200 trees	6.84	3.62	2.11	n/a
Testing Time (seconds) @ 200 trees	0.085	0.098	0.006	n/a

6.5.3 Results and discussion

Examining the results of the performance measures in Table 6-8, one can clearly observe the substantial improved accuracy in predicting the fabrication duration when using any of the proposed ML models as compared to the heuristically simple estimate (i.e., using the average of actual historical durations). In general, the MAPE can be reduced by more than 30% when using the proposed predictive models and the MAE decreases by approximately 50%.

The results of performance measures also show that the ERT model outperforms RF and GBDT models in terms of lower error measures (e.g., MAPE of 26.6% compared to 28.6% and 31.6%, respectively). Both RF and ERT exhibit better performance than GBDT even though the latter is faster to train and test. The training time seems to be better for GBDT using the default settings (i.e., when setting the number of trees to 200); however, when using the optimized hyperparameters, the computational requirements, as summarized in Table 6-7, significantly increase for the GBDT while both the RF and ERT exhibits similar behaviors. The CCs are not significantly different among the three models indicating moderate-strong positive correlation between the actual and predicted fabrication durations.

Figure 6-7 shows the distribution of actual durations (in red) in comparison with the predicted durations using each of the models. It can be observed that ERT (in orange) is the closest to actual durations, which aligns with the performance measure in Table 6-8.

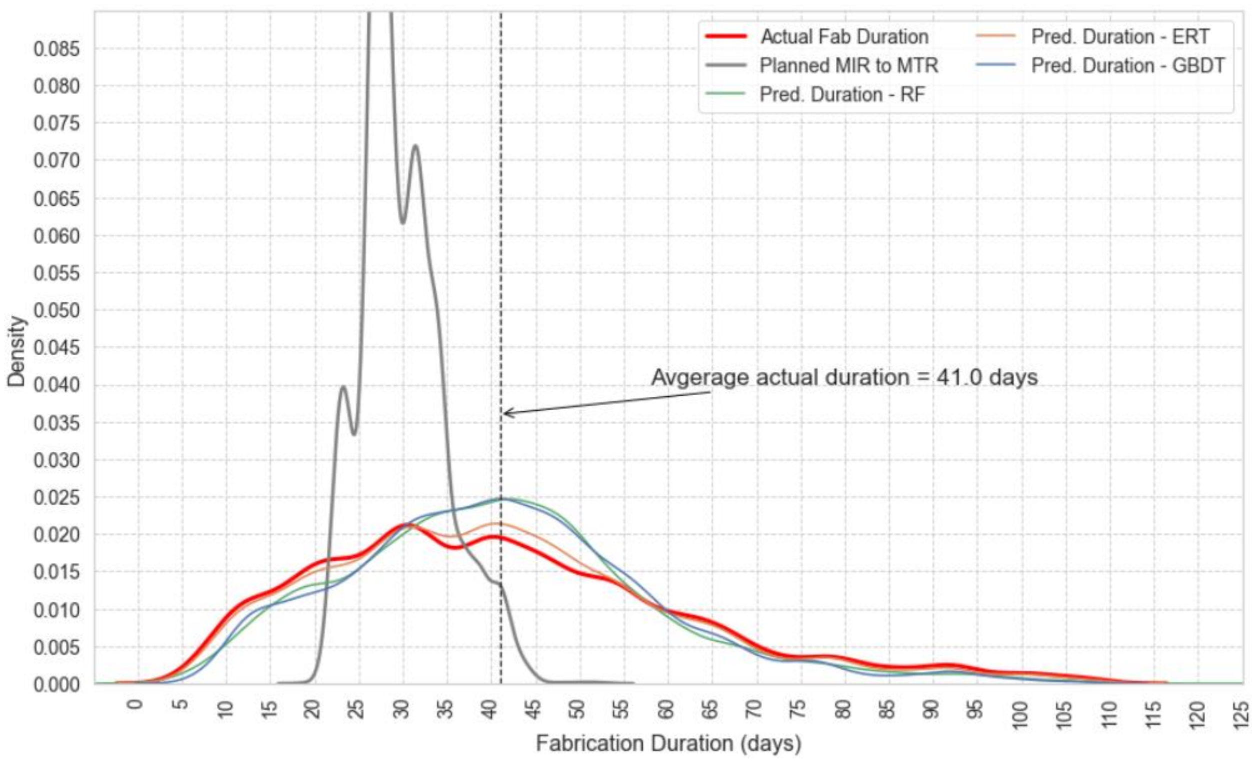


Figure 6-7. Kernel density estimate (probability) of planned, actual, and predicted durations

6.5.4 Conclusion

The exercise of selecting the relevant features, including features that capture real-time loading conditions of the job shop, is crucial as it improves the results significantly as described earlier. Also, experimenting with hyperparameters and tuning their values improves the performance of all models significantly. These two sets of experiments (i.e., feature engineering and hyperparameter tuning) are very important in the development of optimal ML models.

Tree-based models in general have a few common parameters that affect their performance including the number of trees in the ensemble, the size of the random subset of features, and the minimum number of samples required for splitting a node. Other parameters might affect the performance but to a lesser extent.

In summary, predicting the fabrication duration in a pipe spool shop is a representative example of a typical industrialized building construction factory. One distinguishing attribute is the highly customizable and variable type and volume of production. Examining various ML models proves their ability to accurately predict the production time. However, it is crucial to develop and include engineered features that capture the shop utilization in order to achieve acceptable predictions. Although the engineered features need to be developed in a customized fashion that incorporates case-specific knowledge, it is observed that those features nevertheless capture some aspect of the system utilization. In the pipe spool fabrication shop, the developed features capture the real-time shop loading conditions at the time a new pipe spools starts production. In this case, shop loading conditions capture the amount of variation in physical attributes of concurrently processed spools and assign the newly computed feature to the new spool. These generated features have been proven to add valuable information that assists in the predictability of the fabrication durations of the new pipe spools.

Chapter 7 Conclusions

7.1 Summary and Conclusions

This thesis presents the hypothesis that a data-driven framework using machine learning approaches can improve the planning and scheduling practice in industrialized building construction (IBC). The research presented in the thesis suggests that the production time, and consequently the delivery dates, can be accurately predicted in IBC fabrication shops by utilizing the product characteristics, fabrication tracking data, and a set of engineered features.

In essence, a data-driven framework, which is based on similar frameworks described in the literature, is proposed with the addition of and focus on generating features that capture some aspect of the real-time loading conditions of the job shop. Those features significantly improve the predictive accuracy of the machine learning models developed to estimate the fabrication duration. More accurate prediction of fabrication durations implies better estimation of delivery dates and hence improved production scheduling and customer satisfaction.

After an introduction, the thesis starts, in Chapter 2, by a broad literature review about IBC, production scheduling, and the application of machine learning predictive modelling in the construction industry. The chapter also covers an overview of the specific machine learning methods used in this thesis. The developed framework and the methodology followed in this thesis are described in Chapter 3, which also include a discussion on the positioning of construction management research under the design science umbrella rather than as an explanatory science. The implementation of the methodology covers two case studies: the first is at a residential wall panel fabrication factory, described in Chapter 4, and the second is at a pipe spool fabrication shop for industrial construction, discussed in Chapter 6. In Chapter 5, a simulation study is conducted to investigate more thoroughly the generation of features capturing some aspects of loading conditions of the job shop with the goal to improve the predictive accuracy of ML models. This experimentation is applied to the first case study described in Chapter 4.

In conclusion, the thesis proposes that machine learning methods can be used to accurately predict the production cycle time in industrialized building construction, and as a result the scheduling process can be improved. The accuracy of the developed ML models can be improved by augmenting the process dataset with engineered features capturing the real-time loading conditions of the job shop without which the performance of such models might be sub-optimal. Real-time loading conditions of the job shop constitute a crucial component of the dataset that needs to be included in order to improve the predictive accuracy of any machine learning model. The loading conditions of the job shop can be represented using various features and the development of the right set of such features requires a thorough understanding of the production process under investigation. This imposes a requirement that these features be custom-built and because of their dependency on the specific case study, a good understanding of the associated production process is mandatory. Nevertheless, those features are essential to the development of ML models and must capture the loading conditions of the job shop in one way or another. The use of machine learning models to predict production time results in an approximately 25% improvement in the accuracy as compared to using simple averaging estimates based on historical data. Augmenting the dataset with SL features increases the accuracy by an additional 10%.

Moreover, the lookback timeframe used to train the model has a significant effect on the prediction accuracy of the cycle time. Selecting the right timeframe is also case specific; however, using a short timeframe or using the entire dataset to train the model often results in inferior performance as compared to selecting the right timeframe.

7.2 Research Contributions

7.2.1 Academic contributions

The academic contributions of the research presented in this thesis can be summarized as follows:

- The proposed framework provides a comprehensive roadmap to analyze raw data in order to predict production cycle time, specific to industrialized building construction. It considers 1) the streamlining of factory operations from the design stage to the delivery and assembly on site, 2) the high level of customization of the building components, which makes this industry dissimilar to the mass production realized in other manufacturing environments, and 3) the seasonality of supply and demand in IBC which are inherently captured in the production data and can significantly affect predictability of production time.

- The methodology used in this thesis represents a practical approach to estimate the time required to produce building components based on the physical characteristics of the component and the holistic production operations. This means that new features are developed from existing properties in order to capture how the factory settings (e.g., shop capacity, dynamic utilization of the production line, storage and delivery to the site, number of available workers) influence the estimation of production time.

- In order to improve the predictive accuracy of the machine learning models used to estimate the production time, simulation modelling is used to investigate the different features that are attributable to improving the predictability of production time and thus improve the performance of data-driven models. Those features are not present in the original dataset and might be impractical or impossible to extract from the raw data or to collect in practice.

7.2.2 Industrial contributions

The industrial contributions of the research presented in this thesis can be summarized as follows:

- The proposed framework recognizes and identifies the common challenges faced by the industrialized construction industry in regard to data collection, storage, preprocessing, and analysis. Identifying these challenges enables decision makers to better understand the obstacles that hinder the wider adoption of data analytics in the industry, since a lack of adoption of data analytics results in missing the opportunity to use and benefit from this available commodity – the data.

- The proposed framework can provide an alternative method of predicting future uncertainties in terms of production time and resource utilization using existing obtainable data without the expertise required to develop more sophisticated simulation models.

- An integral part of the proposed framework is to develop a practical guideline to clean and prepare the raw data prior to actual modelling and prediction. This task, when done in a systematic and consistent manner, can save much of the time and effort used to develop a data-driven predictive model. Industry practitioners can use the developed procedures as an integral part of data collection and processing operations.

7.3 Limitations and Future Work

Based on the research conducted in this thesis, the following are some of the limitations of the present research and the proposed future research directions:

- The applicability of the research in this thesis has been proven to generalize to fabrication shops in the industrialized building construction industry by examining two cases studies that share some common aspects of construction manufacturing environments yet belong to different subsectors (i.e., residential construction and industrial construction). Although the generalization is illustrated by the two case studies, more research might be needed to further support the generalization of the results and draw conclusions based on the common aspects in IBC manufacturing facilities. Such common aspects include the high level of customization characterizing most building products, the variability in supply and demand of the industrial building construction market, and the inconsistency of labor availability and skill levels.
- The results of the accurate cycle time prediction (i.e., the output of machine learning models) can be leveraged by the automatic integration of ML algorithms with existing production scheduling and tracking systems. By doing so, it is possible to provide very accurate estimates of delivery dates for new products once the production is started. The integration with existing systems can be accomplished at varying levels of details including a dashboard that is updated in real-time to indicate if the new product will be completed as per the preliminary planned schedule, or whether additional time may be required based on shop utilization or material availability.

- One possible improvement to collecting and capturing real-time data in the fabrication shop could be to use a computerized vision-based system to capture real-time loading conditions of the job shop. One example is to use CCTV video streaming, when it is available, to capture shop activities and utilize advanced machine learning techniques to extract production and waiting times as well as the number of workers at each workstation from the multimedia data. Using video capturing techniques can improve the ease of data collection and can provide accurate calculations of production and waiting times compared to using RFID data.

- Simulation modelling requires regular and manual updates and revisions to ensure the simulation model represents the most recent production process. Along with each update, model validation and verification are required, which are tedious and monotonous tasks. A comprehensive automatic integration of the data collection systems (e.g., RFID or CCTV), simulation modelling, and ML predictive models can make this process more reliable, consistent, and error-free. The integration between simulation model and ML models can be carried out to automatically create simulated data that capture real-time shop loading conditions that may not otherwise be readily available by the data collection and tracking systems.

- The approach and methods used in the present thesis has focused on predicting the production cycle time to support short-term scheduling and planning in industrialized building construction facilities. The collected data, the developed machine learning models, and the constructed engineered features support time prediction for short-term scheduling because it is assumed that only the recent conditions of the production line (e.g., factory layout, processing sequencing, interruptions, labor skills) will affect the cycle time. Any changes that would have occurred longtime in the past will be suppressed by recent changes of production line conditions. On the other hand, therefore, mid-term and long-term planning, which spans from a few months to a few years, is not investigated. This is a limitation of the present study and is expected to be a promising area of future research.

References

[1] USG Corporation and U.S. Chamber of Commerce, Commercial Construction Index Q1 2018, 2018. https://www.uschamber.com/report/usg-us-chamber-commerce-commercial-construction-index-2018-q1 (accessed April 2, 2021).

[2] N. Bertram, S. Fuchs, J. Mischke, R. Palter, G. Strube, J. Woetzel, Modular construction: From projects to products, 2019. https://www.mckinsey.com/business-functions/operations/our-insights/modular-construction-from-projects-to-products (accessed April 2, 2021).

[3] Modular Building Institute, 2020 Canadian Commercial Modular Construction Annual Report, 2020. https://www.modular.org/HtmlPage.aspx?name=2020-MBI-annual-reports (accessed April 2, 2021).

[4] M. Kamali, K. Hewage, Life cycle performance of modular buildings: A critical review, Renew. Sustain. Energy Rev. 62 (2016) pp. 1171–1183. https://doi.org/10.1016/j.rser.2016.05.031.

[5] Dodge Data & Analytics, Prefabrication and Modular Construction 2020, 2020, (ISBN: 1800591446).

[6] M.A. Mullens, Factory Design for Modular Homebuilding: Equipping the Modular Factory for Success, Constructability Press, Winter Park, FL, 2011, (ISBN: 978-0983321200).

[7] M.E. Pfund, S.J. Mason, J.W. Fowler, Semiconductor Manufacturing Scheduling and

Dispatching, in: Handb. Prod. Sched., Kluwer Academic Publishers, Boston, 2006: pp. 213–241. https://doi.org/10.1007/0-387-33117-4_9.

[8] S.-H. Chung, H.-W. Huang, Cycle time estimation for wafer fab with engineering lots, IIE Trans. 34 (2002) pp. 105–118. https://doi.org/10.1080/07408170208928854.

[9] W.J. Hopp, M.L. Spearman, Basic Factory Dynamics, in: Fact. Phys. Found. Manuf. Manag., 2nd edition, McGraw-Hill/Irwin, Boston, MA, 2000: pp. 213–247 (ISBN: 9780256247954).

[10] M.S. Altaf, A. Bouferguene, H. Liu, M. Al-Hussein, H. Yu, Integrated production planning and control system for a panelized home prefabrication facility using simulation and RFID, Autom. Constr. 85 (2018) pp. 369–383. https://doi.org/10.1016/j.autcon.2017.09.009.

[11] O. Bedair, Engineering Challenges in the Design of Alberta's Oil Sands Projects, Pract. Period. Struct. Des. Constr. 18 (2013) pp. 247–260. https://doi.org/10.1061/(asce)sc.1943-5576.0000163.

[12] Cristian Petre, Instance-Based Model for Predicting Total Fabrication Duration of Industrial Pipe Spools, MSc Thesis, University of Alberta, 2016. https://doi.org/https://doi.org/10.7939/R39Z90P2N.

[13] C. Goodier, A. Gibb, Future opportunities for offsite in the UK, Constr. Manag. Econ. 25 (2007) pp. 585–595. https://doi.org/10.1080/01446190601071821.

[14] Z. Li, G.Q. Shen, X. Xue, Critical review of the research on the management of prefabricated construction, Habitat Int. 43 (2014) pp. 240–249.

https://doi.org/10.1016/j.habitatint.2014.04.001.

[15] J. Barlow, From Craft Production to Mass Customisation. Innovation Requirements for the UK Housebuilding Industry, Hous. Stud. 14 (1999) pp. 23–42. https://doi.org/10.1080/02673039982984.

[16] G. Winch, Models of Manufacturing and The Construction Process: The Genesis of Re-engineering Construction, Build. Res. Inf. 31 (2003) pp. 107–118. https://doi.org/10.1080/09613210301995.

[17] D. Carlson, Automated Builder: Dictionary Encyclopedia of Industrialized Housing, 3rd edition, Automated Builder Magazine, Publications Division, CMN Associates, Carpinteria, CA, 1995, (ISBN: 978-0960740826).

[18] M.A. Mullens, M.E. Kelley, Lean Homebuilding Using Modular Technology, Hous. Soc. 31 (2004) pp. 41–54. https://doi.org/10.1080/08882746.2004.11430497.

[19] A. Gianino, The Modular Home, Storey Publishing, LLC, 2005, (ISBN: 9781612122687).

[20] H. Said, A.R. Ali, M. Alshehri, Analysis of the Growth Dynamics and Structure of the Modular Building Construction Industry, in: Constr. Res. Congr. 2014, American Society of Civil Engineers, Reston, VA, 2014: pp. 1977–1986. https://doi.org/10.1061/9780784413517.202.

[21] G. Liu, J.H. Nzige, K. Li, Trending topics and themes in offsite construction(OSC) research: The application of topic modelling, Constr. Innov. 19 (2019) pp. 343–366. https://doi.org/10.1108/CI-03-2018-0013.

[22] M.R. Hosseini, I. Martek, E.K. Zavadskas, A.A. Aibinu, M. Arashpour, N. Chileshe,

Critical evaluation of off-site construction research: A Scientometric analysis, Autom. Constr. 87 (2018) pp. 235–247. https://doi.org/10.1016/j.autcon.2017.12.002.

[23] R. Jin, S. Gao, A. Cheshmehzangi, E. Aboagye-Nimo, A holistic review of off-site construction literature published between 2008 and 2018, J. Clean. Prod. 202 (2018) pp. 1202–1219. https://doi.org/10.1016/j.jclepro.2018.08.195.

[24] S. Abdelmageed, T. Zayed, A study of literature in modular integrated construction - Critical review and future directions, J. Clean. Prod. 277 (2020) p. 124044. https://doi.org/10.1016/j.jclepro.2020.124044.

[25] X. Yin, H. Liu, Y. Chen, M. Al-Hussein, Building information modelling for off-site construction: Review and future directions, Autom. Constr. 101 (2019) pp. 72–91. https://doi.org/10.1016/j.autcon.2019.01.010.

[26] X. Li, G.Q. Shen, P. Wu, T. Yue, Integrating Building Information Modeling and Prefabrication Housing Production, Autom. Constr. 100 (2019) pp. 46–60. https://doi.org/10.1016/j.autcon.2018.12.024.

[27] M. Wang, C.C. Wang, S. Sepasgozar, S. Zlatanova, A systematic review of digital technology adoption in off-site construction: Current status and future direction towards industry 4.0, Buildings. 10 (2020) pp. 1–29. https://doi.org/10.3390/buildings10110204.

[28] B. Qi, M. Razkenari, A. Costin, C. Kibert, M. Fu, A Systematic Review of Emerging Technologies in Industrialized Construction, J. Build. Eng. 39 (2021) p. 102265. https://doi.org/10.1016/j.jobe.2021.102265.

[29] T. Luo, X. Xue, Y. Wang, W. Xue, Y. Tan, A systematic overview of prefabricated

construction policies in China, J. Clean. Prod. 280 (2021) p. 124371. https://doi.org/10.1016/j.jclepro.2020.124371.

[30] B.G. Hwang, M. Shan, K.Y. Looi, Key constraints and mitigation strategies for prefabricated prefinished volumetric construction, J. Clean. Prod. 183 (2018) pp. 183–193. https://doi.org/10.1016/j.jclepro.2018.02.136.

[31] W. Zhang, M.W. Lee, L. Jaillon, C.S. Poon, The hindrance to using prefabrication in Hong Kong’s building industry, J. Clean. Prod. 204 (2018) pp. 70–81. https://doi.org/10.1016/j.jclepro.2018.08.190.

[32] I.Y. Wuni, G.Q. Shen, Barriers to the adoption of modular integrated construction: Systematic review and meta-analysis, integrated conceptual framework, and strategies, J. Clean. Prod. 249 (2020) p. 119347. https://doi.org/10.1016/j.jclepro.2019.119347.

[33] M. Razkenari, A. Fenner, A. Shojaei, H. Hakim, C. Kibert, Perceptions of offsite construction in the United States: An investigation of current practices, J. Build. Eng. 29 (2020) p. 101138. https://doi.org/10.1016/j.jobe.2019.101138.

[34] T. Salama, O. Moselhi, M. Al-Hussein, Overview of the Characteristics of the Modular Industry and Barriers to its Increased Market Share, Int. J. Ind. Constr. 2 (2021) pp. 30–53. https://doi.org/10.29173/ijic249.

[35] A. Darko, A.P.C. Chan, Y. Yang, M.O. Tetteh, Building information modeling (BIM)-based modular integrated construction risk management – Critical survey and future needs, Comput. Ind. 123 (2020) p. 103327. https://doi.org/10.1016/j.compind.2020.103327.

[36] R. Jin, J. Hong, J. Zuo, Environmental performance of off-site constructed facilities: A critical review, Energy Build. 207 (2020) p. 109567. https://doi.org/10.1016/j.enbuild.2019.109567.

[37] Z. Wang, H. Hu, J. Gong, X. Ma, W. Xiong, Precast supply chain management in off-site construction: A critical literature review, J. Clean. Prod. 232 (2019) pp. 1204–1217. https://doi.org/10.1016/j.jclepro.2019.05.229.

[38] E. Forcael, I. Ferrari, A. Opazo-vega, Construction 4 . 0 : A Literature Review, (2020).

[39] M. Bilal, L.O. Oyedele, J. Qadir, K. Munir, S.O. Ajayi, O.O. Akinade, H.A. Owolabi, H.A. Alaka, M. Pasha, Big Data in the Construction Industry: A review of Present Status, Opportunities, and Future Trends, Adv. Eng. Informatics. 30 (2016) pp. 500–521. https://doi.org/10.1016/j.aei.2016.07.001.

[40] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Bus. Inf. Syst. Eng. 6 (2014) pp. 239–242. https://doi.org/10.1007/s12599-014-0334-4.

[41] M. Hermann, T. Pentek, B. Otto, Design Principles for Industrie 4.0 Scenarios, in: 49th Hawaii Int. Conf. Syst. Sci., Koloa, HI, 2016: pp. 3928–3937. https://doi.org/10.1109/HICSS.2016.488.

[42] K. Schwab, The Fourth Industrial Revolution, Crown Business, New York, 2017, (ISBN: 9781524758868).

[43] P.K. Muhuri, A.K. Shukla, A. Abraham, Industry 4.0: A bibliometric analysis and detailed overview, Eng. Appl. Artif. Intell. 78 (2019) pp. 218–235. https://doi.org/10.1016/j.engappai.2018.11.007.

[44] F. Craveiro, J.P. Duarte, H. Bartolo, P.J. Bartolo, Additive manufacturing as an enabling technology for digital construction: A perspective on Construction 4.0, Autom. Constr. 103 (2019) pp. 251–267. https://doi.org/10.1016/j.autcon.2019.03.011.

[45] M. Spisakova, M. Kozlovska, Options of Customization in Industrialized Methods of Construction in Terms of Construction 4.0, in: Lect. Notes Civ. Eng., Springer International Publishing, 2020: pp. 444–451. https://doi.org/10.1007/978-3-030-27011-7_56.

[46] National BIM Standard - United States, Natl. Inst. Build. Sci. (2021). https://www.nationalbimstandard.org/faqs (accessed March 6, 2021).

[47] S. Kubba, Building Information Modeling, in: Handb. Green Build. Des. Constr., Elsevier, 2012: pp. 201–226. https://doi.org/10.1016/B978-0-12-385128-4.00005-6.

[48] K. Baker, J. Mentz, J. Riskus, M. Russo, Firm Survey Report, The American Institute of Architects, Washington, DC, 2020, (ISBN: 978-1-57165-016-0). https://www.taylorfrancis.com/books/9781000705829/chapters/10.4324/9780429347856-31.

[49] J. Sinopoli, Design, Construction, and Renovations, in: Smart Build. Syst. Archit. Owners Build., Elsevier, 2010: pp. 139–158. https://doi.org/10.1016/B978-1-85617-653-8.00013-2.

[50] T. Omar, M.L. Nehdi, Data acquisition technologies for construction progress tracking, Autom. Constr. 70 (2016) pp. 143–155. https://doi.org/10.1016/j.autcon.2016.06.016.

[51] E. Valero, A. Adán, C. Cerrada, Evolution of RFID Applications in Construction: A

Literature Review, Sensors. 15 (2015) pp. 15988–16008. https://doi.org/10.3390/s150715988.

[52] E. Valero, A. Adán, Integration of RFID with other technologies in construction, Measurement. 94 (2016) pp. 614–620. https://doi.org/10.1016/j.measurement.2016.08.037.

[53] E.M. Goldratt, Critical chain, North River Press, Great Barrington, Mass., 1997, (ISBN: 9780884271536).

[54] L.P. Leach, Critical Chain Project Management Improves Project Performance, Proj. Manag. J. 30 (1999) pp. 39–51. https://doi.org/10.1177/875697289903000207.

[55] E.L. Demeulemeester, W. Herroelen, Project Scheduling A Research Handbook, 1st edition, Springer US, Boston, MA, 2002. https://doi.org/10.1007/b101924.

[56] S. Mubarak, Introduction, in: Constr. Proj. Sched. Control, 3rd edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2015: pp. 1–14 (ISBN: 9781118846018).

[57] P.M. Institute, A guide to the project management body of knowledge (PMBOK guide), 6th edition, Project Management Institute, Newton Square, PA, 2017, (ISBN: 9781628253900).

[58] H. Yu, An integrated approach toward lean for production homebuilders, PhD Thesis, University of Alberta, 2010. https://doi.org/https://doi.org/10.7939/R39Q65.

[59] L. Shafai, Simulation Based Process Flow Improvement for Wood Framing Home Building Production Lines, MSc Thesis, University of Alberta, 2012. https://doi.org/https://doi.org/10.7939/R3SP65.

[60] M.S. Altaf, Integrated Production Planning and Control System for Prefabrication of Panelized Construction for Residential Building, PhD Thesis, University of Alberta, 2016. https://doi.org/https://doi.org/10.7939/R3QZ22N9G.

[61] R. Azimi, S. Lee, S.M. Abourizk, A. Alvanchi, A framework for an automated and integrated project monitoring and control system for steel fabrication projects, Autom. Constr. 20 (2011) pp. 88–97. https://doi.org/10.1016/j.autcon.2010.07.001.

[62] C. Carson, P. Oakander, C. Relyea, Schedule Development, in: CPM Sched. Constr. Best Pract. Guidel., Project Management Institute, Inc., Newtown Square, PA, 2014 (ISBN: 978-1-62825-037-4). https://learning.oreilly.com/library/view/cpm-scheduling-for/9781628250732/.

[63] I.D. Tommelein, Pull-Driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique, J. Constr. Eng. Manag. 124 (1998) pp. 279–288. https://doi.org/10.1061/(ASCE)0733-9364(1998)124:4(279).

[64] A. Benarroche, Construction Pull Planning Scheduling Backward Can Improve Efficiency, Constr. Paym. Blog. (2020). https://www.levelset.com/blog/construction-pull-planning/ (accessed July 24, 2021).

[65] S.P. Mosayebi, A.R. Fayek, L. Yakemchuk, S. Waters, Factors Affecting Productivity of Pipe Spool Fabrication, Int. J. Archit. Eng. Constr. 1 (2012) pp. 30–36. https://doi.org/10.7492/ijaec.2012.003.

[66] I.D. Tommelein, Process Benefits From Use of Standard Products – Simulation Experiments Using the Pipe Spool Model, in: 14th Annu. Conf. Int. Gr. Lean Constr.,

Santiago, Chile, 2006: pp. 177–189.

[67] P. Wang, Y. Mohamed, S.M. Abourizk, A.R.T. Rawa, Flow Production of Pipe Spool Fabrication: Simulation to Support Implementation of Lean Technique, J. Constr. Eng. Manag. 135 (2009) pp. 1027–1038. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000068.

[68] D. Hu, Y. Mohamed, Pipe Spool Fabrication Sequencing by Automated Planning, in: Constr. Res. Congr. 2012, American Society of Civil Engineers, West Lafayette, Indiana, 2012: pp. 495–504. https://doi.org/10.1061/9780784412329.050.

[69] J.Y.-T. Leung, ed., Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Chapman and Hall/CRC, New York, 2004. https://doi.org/10.1201/9780203489802.

[70] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 5th ed., Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-26580-3.

[71] K.R. Baker, D. Trietsch, Principles of Sequencing and Scheduling, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2018. https://doi.org/10.1002/9781119262602.

[72] S.C. Sarin, B. Nagarajan, L. Liao, Stochastic Scheduling, Cambridge University Press, Cambridge, 2010. https://doi.org/10.1017/CBO9780511778032.

[73] J.M. Framinan, R. Leisten, R. Ruiz García, Manufacturing Scheduling Systems An Integrated View on Models, Methods and Tools, Springer London, London, 2014. https://doi.org/10.1007/978-1-4471-6272-8.

[74] H. Aydilek, A. Allahverdi, Heuristics for no-wait flowshops with makespan subject to

mean completion time, Appl. Math. Comput. 219 (2012) pp. 351–359. https://doi.org/10.1016/j.amc.2012.06.024.

[75] C. Rajendran, H. Ziegler, An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs, Eur. J. Oper. Res. 103 (1997) pp. 129–138. https://doi.org/10.1016/S0377-2217(96)00273-1.

[76] C. Rajendran, Heuristic algorithm for scheduling in a flowshop to minimize total flowtime, Int. J. Prod. Econ. 29 (1993) pp. 65–73. https://doi.org/10.1016/0925-5273(93)90024-F.

[77] M.T. Pereira, M.C. Santoro, An integrative heuristic method for detailed operations scheduling in assembly job shop systems, Int. J. Prod. Res. 49 (2011) pp. 6089–6105. https://doi.org/10.1080/00207543.2010.527385.

[78] M. Yazdani, A. Aleti, S.M. Khalili, F. Jolai, Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem, Comput. Ind. Eng. 107 (2017) pp. 12–24. https://doi.org/10.1016/j.cie.2017.02.019.

[79] Q.K. Pan, L. Gao, L. Wang, J. Liang, X.Y. Li, Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, Expert Syst. Appl. 124 (2019) pp. 309–324. https://doi.org/10.1016/j.eswa.2019.01.062.

[80] V. Fernandez-Viagas, P. Perez-Gonzalez, J.M. Framinan, The distributed permutation flow shop to minimise the total flowtime, Comput. Ind. Eng. 118 (2018) pp. 464–477. https://doi.org/10.1016/j.cie.2018.03.014.

[81] B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop

scheduling problem, Eur. J. Oper. Res. 239 (2014) pp. 323–334. https://doi.org/10.1016/j.ejor.2014.05.024.

[82] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem, Int. J. Prod. Econ. 145 (2013) pp. 387–396. https://doi.org/10.1016/j.ijpe.2013.05.004.

[83] V. Fernandez-Viagas, R. Ruiz, J.M. Framinan, A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation, Eur. J. Oper. Res. 257 (2017) pp. 707–721. https://doi.org/10.1016/j.ejor.2016.09.055.

[84] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From Data Mining to Knowledge Discovery in Databases, AI Mag. 17 (1996) pp. 37–54. https://doi.org/10.1609/aimag.v17i3.1230.

[85] P. Cabena, R. Stadler, J. Verhees, A. Zanasi, P. Hadjnian, Discovering data mining from concept to implementation, Prentice Hall, New Jersey, 1998, (ISBN: 978-0137439805).

[86] L. Soibelman, H. Kim, Data Preparation Process for Construction Knowledge Generation through Knowledge Discovery in Databases, J. Comput. Civ. Eng. 16 (2002) pp. 39–48. https://doi.org/10.1061/(ASCE)0887-3801(2002)16:1(39).

[87] M. Alavi, D.E. Leidner, Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues, MIS Q. 25 (2001) p. 107. https://doi.org/10.2307/3250961.

[88] J. Liebowitz, I. Megbolugbe, A set of frameworks to aid the project manager in conceptualizing and implementing knowledge management initiatives, Int. J. Proj. Manag.

21 (2003) pp. 189–198. https://doi.org/10.1016/S0263-7863(02)00093-5.

[89] A.M. Hammad, An Integrated Framework for Managing Labour Resources Data in Industrial Construction Projects: A Knowledge Discovery in Data (KDD) Approach, PhD Thesis, University of Alberta, 2009. https://www.collectionscanada.gc.ca/obj/thesescanada/vol2/002/NR55814.PDF (accessed April 2, 2021).

[90] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition, Morgan Kaufmann, Burlington, MA, 2011. https://doi.org/10.1007/s00170-004-2497-5.

[91] L. Soibelman, J. Wu, C. Caldas, I. Brilakis, K.Y. Lin, Management and analysis of unstructured construction data types, Adv. Eng. Informatics. 22 (2008) pp. 15–27. https://doi.org/10.1016/j.aei.2007.08.011.

[92] M. Bilal, L.O. Oyedele, Guidelines for applied machine learning in construction industry—A case of profit margins estimation, Adv. Eng. Informatics. 43 (2020) p. 101013. https://doi.org/10.1016/j.aei.2019.101013.

[93] H. Wickham, Tidy Data, J. Stat. Softw. 59 (2014) pp. 1–23. https://doi.org/10.18637/jss.v059.i10.

[94] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, J. Mach. Learn. Res. 13 (2012) pp. 281–305. https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf (accessed April 2, 2021).

[95] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (2018) pp. 1–52.

[96] R. Akerkar, ed., Advanced Data Analytics for Business, in: Big Data Comput., Chapman and Hall/CRC, Boca Raton, FL, 2013: pp. 373–397. https://doi.org/10.1201/b16014.

[97] K. Lepenioti, A. Bousdekis, D. Apostolou, G. Mentzas, Prescriptive analytics: Literature review and research challenges, Int. J. Inf. Manage. 50 (2020) pp. 57–70. https://doi.org/10.1016/j.ijinfomgt.2019.04.003.

[98] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd edition, Elsevier, 2012. https://doi.org/10.1016/C2009-0-61819-5.

[99] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edition, Springer New York, New York, NY, 2009. https://doi.org/10.1007/978-0-387-84858-7.

[100] X.S. Yang, Introduction to Algorithms for Data Mining and Machine Learning, Elsevier, 2019. https://doi.org/10.1016/C2018-0-02034-4.

[101] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proc. Fifth Berkeley Symp. Math. Stat. Probab., University of California Press, 1967: pp. 281–297. http://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf.

[102] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, Proc. Eighteenth Annu. ACM-SIAM Symp. Discret. Algorithms. (2007) pp. 1027–1035.

https://www2.cs.duke.edu/courses/cps296.2/spring07/papers/kMeansPlusPlus.pdf.

[103] L. Kaufman, P.J. Rousseeuw., Finding Groups in Data, John Wiley & Sons, Inc., Hoboken, NJ, USA, 1990. https://doi.org/10.1002/9780470316801.

[104] H.S. Park, C.H. Jun, A simple and fast algorithm for K-medoids clustering, Expert Syst. Appl. 36 (2009) pp. 3336–3341. https://doi.org/10.1016/j.eswa.2008.01.039.

[105] A. Chaturvedi, K. Foods, P.E. Green, J.D. Carroll, K-modes clustering, J. Classif. 18 (2001) pp. 35–55. https://doi.org/10.1007/s00357-001-0004-3.

[106] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: Second Int. Conf. Knowl. Discov. Data Min., Portland, OR, 1996: pp. 226–231.

[107] A. Hinneburg, D.A. Keim, A General Approach to Clustering in Large Databases with Noise, Knowl. Inf. Syst. 5 (2003) pp. 387–415. https://doi.org/10.1007/s10115-003-0086-9.

[108] W. Wang, J. Yang, R. Muntz, STING : A statistical information grid approach to spatial data mining, Proc. 23rd Int. Conf. Very Large Databases, VLDB 1997. (1997) pp. 186–195.

[109] G. Sheikholeslami, S. Chatterjee, A. Zhang, Wavecluster: A multi-resolution clustering approach for very large spatial databases, in: Proc. 24th Int. Conf. Very Large Data Bases, New York, NY, 1998: pp. 428–439 (ISBN: 1558605665).

[110] S. Raschka, V. Mirjalili, Python Machine Learning - Second Edition, 2nd edition, Packt Publishing, Birmingham, 2017, (ISBN: 9781787125933).

[111] R.S. Sutton, A.G. Barto, Reinforcement Learning : an Introduction, 2nd edition, MIT Press, Cambridge, MA, 2018, (ISBN: 978-0262039246). https://mitpress.mit.edu/books/reinforcement-learning-second-edition.

[112] S. Weisberg, Applied Linear Regression, 4th edition, John Wiley & Sons, Hoboken, New Jersey, 2014, (ISBN: 9781118594858).

[113] J. Groß, Linear Regression, 1st edition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. https://doi.org/10.1007/978-3-642-55864-1.

[114] D.J. Olive, Linear Regression, Springer International Publishing, Cham, 2017. https://doi.org/10.1007/978-3-319-55252-1.

[115] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, Springer New York, New York, NY, 2013. https://doi.org/10.1007/978-1-4614-7138-7.

[116] R. Tibshirani, Regression Shrinkage and Selection via the Lasso, J. R. Stat. Soc. 58 (1996) pp. 267–288. https://www.jstor.org/stable/2346178.

[117] A.E. Hoerl, R.W. Kennard, Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics. 12 (1970) pp. 55–67. https://doi.org/10.1080/00401706.1970.10488634.

[118] A.H. Al-Momani, Construction delay: a quantitative analysis, Int. J. Proj. Manag. 18 (2000) pp. 51–59. https://doi.org/10.1016/S0263-7863(98)00060-X.

[119] D.J. Lowe, M.W. Emsley, A. Harding, Predicting Construction Cost Using Multiple Regression Techniques, J. Constr. Eng. Manag. 132 (2006) pp. 750–758. https://doi.org/10.1061/(ASCE)0733-9364(2006)132:7(750).

[120] T.M. Zayed, D.W. Halpin, Productivity and Cost Regression Models for Pile Construction, J. Constr. Eng. Manag. 131 (2005) pp. 779–789. https://doi.org/10.1061/(ASCE)0733-9364(2005)131:7(779).

[121] K. Blyth, A. Kaka, A novel multiple linear regression model for forecasting S-curves, Eng. Constr. Archit. Manag. 13 (2006) pp. 82–95. https://doi.org/10.1108/09699980610646511.

[122] M. Al Qady, A. Kandil, Automatic Classification of Project Documents on the Basis of Text Content, J. Comput. Civ. Eng. 29 (2015) p. 04014043. https://doi.org/10.1061/(asce)cp.1943-5487.0000338.

[123] J. Wang, B. Ashuri, Predicting ENR'S Construction Cost Index Using the Modified K Nearest Neighbors (KNN) Algorithm, in: Constr. Res. Congr. 2016, American Society of Civil Engineers, Reston, VA, 2016: pp. 2502–2509. https://doi.org/10.1061/9780784479827.249.

[124] Y. Zhang, L. Ding, P.E.D. Love, Planning of Deep Foundation Construction Technical Specifications Using Improved Case-Based Reasoning with Weighted k-Nearest Neighbors, J. Comput. Civ. Eng. 31 (2017) p. 04017029. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000682.

[125] A.H. Boussabaine, The use of artificial neural networks in construction management: A review, Constr. Manag. Econ. 14 (1996) pp. 427–436. https://doi.org/10.1080/014461996373296.

[126] O. Moselhi, T. Hegazy, P. Fazio, Neural Networks as Tools in Construction, J. Constr.

Eng. Manag. 117 (1991) pp. 606–625. https://doi.org/10.1061/(ASCE)0733-9364(1991)117:4(606).

[127] G. Heravi, E. Eslamdoost, Applying Artificial Neural Networks for Measuring and Predicting Construction-Labor Productivity, J. Constr. Eng. Manag. 141 (2015) p. 04015032. https://doi.org/10.1061/(asce)co.1943-7862.0001006.

[128] D.A. Patel, K.N. Jha, Neural Network Model for the Prediction of Safe Work Behavior in Construction Projects, J. Constr. Eng. Manag. 141 (2015) p. 04014066. https://doi.org/10.1061/(asce)co.1943-7862.0000922.

[129] M.-Y. Cheng, Y.-H. Chang, D. Korir, Novel Approach to Estimating Schedule to Completion in Construction Projects Using Sequence and Nonsequence Learning, J. Constr. Eng. Manag. 145 (2019) p. 04019072. https://doi.org/10.1061/(asce)co.1943-7862.0001697.

[130] L. Breiman, Random forests, Mach. Learn. 45 (2001) pp. 5–32. https://doi.org/10.1023/A:1010933404324.

[131] J. Zhou, X. Shi, K. Du, X. Qiu, X. Li, H.S. Mitri, Development of Ground Movements Due to a Shield Tunnelling Prediction Model Using Random Forests, in: Geo-China 2016, American Society of Civil Engineers, Reston, VA, 2016: pp. 108–115. https://doi.org/10.1061/9780784480106.014.

[132] Y. Hu, D. Castro-Lacouture, Clash Relevance Prediction Based on Machine Learning, J. Comput. Civ. Eng. 33 (2019) pp. 1–15. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000810.

[133] Q. Han, C. Gui, J. Xu, G. Lacidogna, A generalized method to predict the compressive strength of high-performance concrete by improved random forest algorithm, Constr. Build. Mater. 226 (2019) pp. 734–742. https://doi.org/10.1016/j.conbuildmat.2019.07.315.

[134] A. Botchkarev, A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms, Interdiscip. J. Information, Knowledge, Manag. 14 (2019) pp. 045–076. https://doi.org/10.28945/4184.

[135] H. Wang, H. Zheng, Model Validation, Machine Learning, in: Encycl. Syst. Biol., Springer New York, New York, NY, 2013: pp. 1406–1407. https://doi.org/10.1007/978-1-4419-9863-7_233.

[136] R.J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, (2021). https://otexts.com/fpp3/tscv.html (accessed March 31, 2021).

[137] F. Provost, T. Fawcett, Data Science and its Relationship to Big Data and Data-Driven Decision Making, Big Data. 1 (2013) pp. 51–59. https://doi.org/10.1089/big.2013.1508.

[138] J. Dean, Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners, John Wiley and Sons, Hoboken, NJ, 2014, (ISBN: 9781118920695).

[139] N. Naderpajouh, J. Choi, M. Hastak, Exploratory Framework for Application of Analytics in the Construction Industry, J. Manag. Eng. 32 (2016) p. 04015047. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000409.

[140] S. El-Omari, O. Moselhi, Integrating automated data acquisition technologies for progress reporting of construction projects, Autom. Constr. 20 (2011) pp. 699–705.

https://doi.org/10.1016/j.autcon.2010.12.001.

[141] E. Elbeltagi, M. Dawood, Integrated visualized time control system for repetitive construction projects, Autom. Constr. 20 (2011) pp. 940–953. https://doi.org/10.1016/j.autcon.2011.03.012.

[142] H. Kim, L. Soibelman, F. Grobler, Factor selection for delay analysis using Knowledge Discovery in Databases, Autom. Constr. 17 (2008) pp. 550–560. https://doi.org/10.1016/j.autcon.2007.10.001.

[143] V.S. Desai, S. Joshi, Application of Decision Tree Technique to Analyze Construction Project Data, in: S.K. Prasad, H.M. Vin, S. Sahni, M.P. Jaiswal, B. Thipakorn (Eds.), Inf. Syst. Technol. Manag., Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 304–313. https://doi.org/10.1007/978-3-642-12035-0_30.

[144] M. Wauters, M. Vanhoucke, Support Vector Machine Regression for project control forecasting, Autom. Constr. 47 (2014) pp. 92–106. https://doi.org/10.1016/j.autcon.2014.07.014.

[145] Q. Fang, H. Li, X. Luo, L. Ding, T.M. Rose, W. An, Y. Yu, A deep learning-based method for detecting non-certified work on construction sites, Adv. Eng. Informatics. 35 (2018) pp. 56–68. https://doi.org/10.1016/j.aei.2018.01.001.

[146] J. Choi, B. Gu, S. Chin, J.-S. Lee, Machine learning predictive model based on national data for fatal accidents of construction workers, Autom. Constr. 110 (2020) pp. 1–14. https://doi.org/10.1016/j.autcon.2019.102974.

[147] N.D. Nath, A.H. Behzadan, S.G. Paal, Deep learning for site safety: Real-time detection of

personal protective equipment, Autom. Constr. 112 (2020) pp. 1–20. https://doi.org/10.1016/j.autcon.2020.103085.

[148] J. Wu, N. Cai, W. Chen, H. Wang, G. Wang, Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset, Autom. Constr. 106 (2019) pp. 1–7. https://doi.org/10.1016/j.autcon.2019.102894.

[149] D. Chakraborty, H. Elhegazy, H. Elzarka, L. Gutierrez, A novel construction cost prediction model using hybrid natural and light gradient boosting, Adv. Eng. Informatics. 46 (2020) pp. 1–10. https://doi.org/10.1016/j.aei.2020.101201.

[150] J.A. Harding, M. Shahbaz, Srinivas, A. Kusiak, Data Mining in Manufacturing: A Review, J. Manuf. Sci. Eng. 128 (2006) pp. 969–976. https://doi.org/10.1115/1.2194554.

[151] A.K. Choudhary, J.A. Harding, M.K. Tiwari, Data mining in manufacturing: a review based on the kind of knowledge, J. Intell. Manuf. 20 (2009) pp. 501–521. https://doi.org/10.1007/s10845-008-0145-x.

[152] A. Öztürk, S. Kayalıgil, N.E. Özdemirel, Manufacturing lead time estimation using data mining, Eur. J. Oper. Res. 173 (2006) pp. 683–700. https://doi.org/10.1016/j.ejor.2005.03.015.

[153] P. Backus, M. Janakiram, S. Mowzoon, G.C. Runger, A. Bhargava, Factory Cycle-Time Prediction With a Data-Mining Approach, IEEE Trans. Semicond. Manuf. 19 (2006) pp. 252–258. https://doi.org/10.1109/TSM.2006.873400.

[154] T. Chen, R. Romanowski, Precise and Accurate Job Cycle Time Forecasting in a Wafer Fabrication Factory with a Fuzzy Data Mining Approach, Math. Probl. Eng. 2013 (2013)

pp. 1–14. https://doi.org/10.1155/2013/496826.

[155] B. Can, C. Heavey, A Demonstration of Machine Learning for Explicit Functions for Cycle Time Prediction Using MES Data, in: 2016 Winter Simul. Conf., Washington, DC, 2016: pp. 2500–2511. https://doi.org/10.1109/WSC.2016.7822289.

[156] D. Gyulai, A. Pfeiffer, G. Nick, V. Gallina, W. Sihn, L. Monostori, Lead time prediction in a flow-shop environment with analytical and machine learning approaches, IFAC-PapersOnLine. 51 (2018) pp. 1029–1034. https://doi.org/10.1016/j.ifacol.2018.08.472.

[157] L. Lingitz, V. Gallina, F. Ansari, D. Gyulai, A. Pfeiffer, W. Sihn, L. Monostori, Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer, Procedia CIRP. 72 (2018) pp. 1051–1056. https://doi.org/10.1016/j.procir.2018.03.148.

[158] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, R. Wirth, CRISP-DM 1.0 Step-by-Step Data Mining Guide, 2000. https://the-modeling-agency.com/crisp-dm.pdf (accessed April 2, 2021).

[159] P. Heller, D. Piziak, R. Stackowiak, A. Licht, T. Luckenbach, B. Cauthen, A. Misra, J. Wyant, J. Knudsen, An Enterprise Architect's Guide to Big Data — Reference Architecture Overview, 2016. http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf (accessed April 2, 2021).

[160] T.H. Davenport, J.G. Harris, R. Morison, Analytics at Work: Smarter Decisions, Better Results, Harvard Business Press, Boston, Massachusetts, 2010, (ISBN: 9781422177693).

[161] D. Dietrich, B. Heller, Y. Beibei, Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data, John Wiley & Sons, Indianapolis, IN, 2015, (ISBN: 9781118876138).

[162] J.B. Rollins, Foundational Methodology for Data Science, IBM Anal. (2015). https://tdwi.org/~/media/64511A895D86457E964174EDC5C4C7B1.PDF (accessed April 2, 2021).

[163] D. Seymour, J. Rooke, The culture of the industry and the culture of research, Constr. Manag. Econ. 13 (1995) pp. 511–523. https://doi.org/10.1080/01446199500000059.

[164] L. Koskela, Which Kind of Science Is Construction Management?, in: 16th Annu. Conf. Int. Gr. Lean Constr., Manchester, UK, 2008: pp. 51–60. https://iglc.net/Papers/Details/584.

[165] J.E. van Aken, Management Research as a Design Science: Articulating the Research Products of Mode 2 Knowledge Production in Management, Br. J. Manag. 16 (2005) pp. 19–36. https://doi.org/10.1111/j.1467-8551.2005.00437.x.

[166] C.G. Da Rocha, C.T. Formoso, P. Tzortzopoulos-Fazenda, L. Koskela, A. Tezel, Design Science Research in Lean Construction: Process and Outcomes, in: 20th Conf. Int. Gr. Lean Constr., San Diego, California, 2012. https://iglc.net/Papers/Details/770.

[167] S. Khan, P. Tzortzopoulos, Using Design Science Research and Action Research to Bridge the Gap Between Theory and Practice in Lean Construction Research, in: 26th Annu. Conf. Int. Gr. Lean Constr. Evol. Lean Constr. Towar. Matur. Prod. Manag. Across Cult. Front., Chennai, India, 2018: pp. 209–219. https://doi.org/10.24928/2018/0409.

[168] F.R. Hamzeh, G. El Samad, S. Emdanat, Advanced Metrics for Construction Planning, J. Constr. Eng. Manag. 145 (2019) pp. 1–16. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001702.

[169] A.R. Hevner, S.T. March, J. Park, S. Ram, Design Science in Information Systems Research, MIS Q. 28 (2004) pp. 75–105. https://doi.org/10.2307/25148625.

[170] S. Chakrabarti, E. Cox, E. Frank, R.H. Güting, J. Han, X. Jiang, M. Kamber, S.S. Lightstone, T.P. Nadeau, R.E. Neapolitan, D. Pyle, M. Refaat, M. Schneider, T.J. Teorey, I.H. Witten, Data Mining Know It All, Morgan Kaufmann, Burlington, MA, 2009, (ISBN: 9780123746290).

[171] Y. Li, C. Yang, Domain knowledge based explainable feature construction method and its application in ironmaking process, Eng. Appl. Artif. Intell. 100 (2021) p. 104197. https://doi.org/10.1016/j.engappai.2021.104197.

[172] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd edition, Springer New York, New York, NY, 2009. https://doi.org/10.1007/978-0-387-84858-7.

[173] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification And Regression Trees, 1st edition, Routledge, 1984. https://doi.org/10.1201/9781315139470.

[174] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, A. Mueller, Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) pp. 2825–2830. http://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf (accessed April 2, 2021).

[175] R.N. Callahan, K.M. Hubbard, N.M. Bacoski, The use of simulation modeling and

factorial analysis as a method for process flow improvement, Int. J. Adv. Manuf. Technol. 29 (2006) pp. 202–208. https://doi.org/10.1007/s00170-004-2497-5.

[176] S. AbouRizk, Role of Simulation in Construction Engineering and Management, J. Constr. Eng. Manag. 136 (2010) pp. 1140–1153. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000220.

[177] S. AbouRizk, Y. Mohamed, Simphony-an integrated environment for construction simulation, in: 2000 Winter Simul. Conf. Proc., IEEE, 2000: pp. 1907–1914. https://doi.org/10.1109/WSC.2000.899185.

[178] S. AbouRizk, S. Hague, R. Ekyalimpa, S. Newstead, Simphony: a next generation simulation modelling environment for the construction domain, J. Simul. 10 (2016) pp. 207–215. https://doi.org/10.1057/jos.2014.33.

[179] R.G. Sargent, Verification And Validation Of Simulation Models: An Advanced Tutorial, in: 2020 Winter Simul. Conf., IEEE, 2020: pp. 16–29. https://doi.org/10.1109/WSC48552.2020.9384052.

[180] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (2006) pp. 3–42. https://doi.org/10.1007/s10994-006-6226-1.

[181] M. Nahangi, J. Yeung, J. Amaral, F.N. Freitas, S. Walbridge, C.T. Haas, Automated Deviation Analysis for As-Built Status Assessment of Steel Assemblies and Pipe Spools, in: Comput. Civ. Build. Eng., American Society of Civil Engineers, Reston, VA, 2014: pp. 2063–2070. https://doi.org/10.1061/9780784413616.256.

[182] J. Song, C.T. Haas, C. Caldas, E. Ergen, B. Akinci, Automating the task of tracking the

delivery and receipt of fabricated pipe spools in industrial projects, Autom. Constr. 15 (2006) pp. 166–177. https://doi.org/10.1016/j.autcon.2005.03.001.

[183] D. Hu, Y. Mohamed, A dynamic programming solution to automate fabrication sequencing of industrial construction components, Autom. Constr. 40 (2014) pp. 9–20. https://doi.org/10.1016/j.autcon.2013.12.013.

[184] R.S. Sharpe, NDT handbook, second edition, volume 3: radiography and radiographic testing, 1985. https://doi.org/10.1016/0308-9126(85)90030-6.

[185] American Society for Nondestructive Testing, Introduciton to Nondestructive Testing, (2021). https://asnt.org/MajorSiteSections/About/Introduction_to_Nondestructive_Testing.aspx (accessed May 10, 2021).

[186] Welding Technology Institute of Austalia Guidance Note 6 - post weld heat treatment of welded structures, 2003. http://vibfem.com.au/resources/stress_relieving/post_weld_HT.pdf (accessed May 10, 2021).

[187] G. Antaki, R. Gilada, Design Basis Loads and Qualification, 2015. https://doi.org/10.1016/b978-0-12-417248-7.00002-3.

[188] H. Drucker, Improving Regressors using Boosting Techniques, in: 14th Int. Conf. Mach. Learn., Morgan Kaufmann, 1997: pp. 107–115. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.5683&rep=rep1&type=pdf.

[189] Y. Freund, R.E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and

an Application to Boosting, J. Comput. Syst. Sci. 55 (1997) pp. 119–139.

https://doi.org/https://doi.org/10.1006/jcss.1997.1504.

Appendix A: Python Script for ML Modelling – Case Study 1

Table of Contents (TOC)

1. Importing Libraries & Defining Functions

↑ back to TOC

```
# importing libraries
import pandas as pd
from pandas.plotting import register_matplotlib_converters
import numpy as np
import random
from datetime import *
```

```python
import os
from itertools import chain
from scipy import stats
from matplotlib import pyplot as plt; from matplotlib import cm
import matplotlib.dates as mdates; import matplotlib.ticker as mticker
import seaborn as sns

# importing Sci-kit Learn libraries
from sklearn import linear_model, ensemble, neural_network, tree, neighbors
from sklearn import metrics
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split, cross_val_predict
from sklearn.model_selection import cross_validate, KFold, GridSearchCV
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

# setting custom options
%matplotlib inline
pd.options.mode.chained_assignment = None
pd.set_option('display.max_columns', 50, 'display.max_rows',2000)
np.set_printoptions(precision=3, suppress=True)

# setting the folder from which data is read
folder = r"F:\Academic\Publications\Journals\04\Python"; os.chdir(folder)

# a function to calculate the time difference in minutes between two timestamps
def calculateTimeDiff(initialTime, lastTime):
    if lastTime.day == initialTime.day:
        timeDelta = lastTime - initialTime
        # returning the time difference in minutes
        return round(timeDelta.total_seconds()/60, 2)
    else:
        return -1

# a function to extract the values from the boxplot and save it as a dataframe
def getBoxplotData(box, labels):
    rows = []
    # looping over each box which represents one variable
    for x in range(len(labels)):
        dict1 = {}
        dict1['Label'] = labels[x]
        dict1['Lower Cap'] = box['caps'][x*2].get_ydata()[0]
        dict1['25th quartile'] = box['boxes'][x].get_ydata()[0]
        dict1['Median'] = box['medians'][x].get_ydata()[0]
        dict1['75th quartile'] = box['boxes'][x].get_ydata()[2]
        dict1['Upper Cap'] = box['caps'][(x*2)+1].get_ydata()[0]
        dict1['Outliers Count'] = len(box['fliers'][x].get_ydata())
        rows.append(dict1)
    return pd.DataFrame(rows)

# a function to train a model and calculate its accuracy on the 'training' set
def train_and_evaluate(clf, X_train, y_train, print_score=True):
    clf.fit(X_train, y_train)
    if print_score:
        score = clf.score(X_train, y_train)
        print("Coefficient of determination (R\u00b2)= {0:.4f}".format(score))
```

```
# a function to evaluate the performance of the model on the 'testing' set
def measure_performance(X, y, clf, show_accuracy=True,
                        show_classification_report=True,
                        show_confusion_matrix=True,
                        show_Reg_metrics=False):
    y_pred = clf.predict(X)
    if show_accuracy:
        print("Accuracy:{0:.3f}".format(metrics.accuracy_score(y, y_pred)), "\n")
    if show_classification_report:
        print("Classification report")
        print(metrics.classification_report(y, y_pred), "\n")
    if show_confusion_matrix:
        print("Confusion matrix")
        print(metrics.confusion_matrix(y, y_pred), "\n")
    if show_Reg_metrics:
        values = []
        values.append(round(metrics.mean_absolute_error(y, y_pred), 2))
        values.append(round(metrics.median_absolute_error(y, y_pred), 2))
        values.append(round(metrics.max_error(y, y_pred), 2))
        values.append(str(round(metrics.mean_absolute_percentage_error(y, y_pred), 2)
*100) +"%")
        values.append(round(metrics.r2_score(y, y_pred), 2))
        values.append(round(stats.pearsonr(y, y_pred)[0], 2))
        names = ['Mean Absolute Error (MAE)', 'Median Absolute Error (MedAE)', 'Maxim
um Error (MaxE)',
                 'Mean Absolute Percentage Error (MAPE)', 'Coefficient of determinati
on (R\u00b2)',
                 'Correlation Coefficient (CC)']
        return pd.DataFrame(values, index=names, columns=['Performance Metrics'])

print("Successful. All required libraries have been loaded.")
```

2. Loading raw data & Initial Preparation

↑ back to TOC

```
file_name = r"RFID_RawData.xlsx"

# Reading RFID, multipanel, and production data each into a dataframe
rfidreading = pd.read_excel(file_name, sheet_name="RFIDReadings")
multipanel = pd.read_excel(file_name, sheet_name="Multipanel")
production = pd.read_excel(file_name, sheet_name="ProductionVolume")

# RFID Readings dataframe (df1)
df1 = rfidreading.copy(deep=True)
print("The size of RFID Readings dataframe (df1) = {} rows by {} columns".format(df1.
shape[0], df1.shape[1]))
# deleting records where 'FirstReadDate' occured on or before 2015-09-10
# because factory setup has changed making data not relevant to the study
df1 = df1.loc[df1["FirstReadDate"] > datetime(2015, 9, 10)]
# keeping rows where 'AntennaDescription' = A1, A2, A3, A4, or A5
# these readings represent the Multiwall production phase
df1 = df1.loc[(df1["AntennaDescription"] == "A1") | (df1["AntennaDescription"] == "A2
") | (df1["AntennaDescription"] == "A3")
              | (df1["AntennaDescription"] == "A4") | (df1["AntennaDescription"] == "
A5")]
# keeping only relevant columns in df1
# "WallNumber" has redundant ID info that is already present in "PanelNumber"
# "LocationSourceAntenna" and "LocationTagID" are not required for the study
# "TagID" and "LastReadDate" contains redundant information
df1.drop(['TagID', 'WallNumber', 'LocationSourceAntenna', 'LocationTagID', 'LastReadD
ate'], axis=1, inplace=True)
print("Size after initial data filtering = {} rows by {} columns".format(df1.shape[0]
, df1.shape[1]))
print("---------------------------------------------------------------------")

# Multipanel dataframe (df2)
df2 = multipanel.copy(deep=True)
print("The size of Multipanel dataframe (df2) = {} rows by {} columns".format(df2.sha
pe[0], df2.shape[1]))
# keeping only rows where 'Panel Attribute' = 'Wall line'
# manually manufactured panels are execluded
df2 = df2.loc[df2["Panel Attribute"] == "Wall line"]
# keeping only relevant columns in df2
# description of deleted columns can be found in Chapter 4
df2.drop(['Job', 'Component', 'Wall', 'Panel Attribute', 'TypeX', 'Siding', 'SidingLi
ne', 'Model', 'Floor', 'Unit',
          'GarageDoor', 'Drywall', 'Sequence', 'Basementwall', 'position', 'Productio
nJob'], axis=1, inplace=True)
print("Size after initial data filtering = {} rows by {} columns".format(df2.shape[0]
, df2.shape[1]))
print("---------------------------------------------------------------------")

# ProductionVolume dataframe (df3)
df3 = production.copy(deep=True)
print("The size of ProductionVolume dataframe (df3) = {} rows by {} columns".format(d
f3.shape[0], df3.shape[1]))
# deleting records where 'Date1' <= 2015-09-10) to match the condition for df1
df3 = df3.loc[df3["Date1"] > datetime(2015, 9, 10)]
df3.reset_index(drop=True, inplace=True)
# changing the type of 'Date1' from datetime.datetime to datetime.date
# date values are later compared with other date fields
date1 = lambda row: row["Date1"].date()
df3["Date1"] = df3.apply(date1, axis=1)
print("Size after initial data filtering = {} rows by {} columns".format(df3.shape[0]
, df3.shape[1]))
```

3. Descriptive Statistics & Data Visualization

↑ back to TOC

3.1 Visualization of raw data in (df1)

```
n = df1.shape[0]; print("Total number of RFID readings =", n)
sList, nList = ['A1', 'A2', 'A3', 'A4', 'A5'], []

# a loop to count the number of RFID readings at each of the five stations
for i in range(len(sList)):
    x = df1.loc[df1['AntennaDescription'] == sList[i]]['AntennaDescription'].count()
    nList.append([x, str(round(x/n*100, 2))+'%'])
table = pd.DataFrame(nList, index=sList, columns=['Total RFID readings', '% of total
reading'])
display(table)

plt.figure(figsize=(12, 6))
plt.bar(x=sList, height=np.array(nList).T[0].astype(int), color='blue', alpha=0.6)

# a loop to display the % on top of each bar of the histogram
for i in range(len(sList)):
    y = int(np.array(nList).T[0][i])+20
    s = table.at[sList[i], '% of total reading']
    plt.text(x=sList[i], y=y, s=s, horizontalalignment='center', fontsize=13)

plt.ylim(36900, 38200)
plt.yticks(list(range(36900, 38200, 100)))
plt.title("Total Number of Readings at Each Antenna Location", fontsize=16)
plt.xlabel("Antenna Location", fontsize=14); plt.ylabel("RFID Readings Count", fontsi
ze=14)
plt.grid(); plt.show()

register_matplotlib_converters()
# counting how many readings each multiwall panel has every day
df1_grouped = df1.groupby(['FirstReadDate', 'PanelNumber']).count().reset_index().set
_index('FirstReadDate')
df1_daily = df1_grouped.groupby(['FirstReadDate']).sum()

# the function to aggregate with in resampling and moving window
aggFunction = 'sum'
# resampling to monthly, weekly frequencies, and using 14-day rolling window, aggrega
ting all by 'aggFunction'
df1_monthly = df1_daily.resample('M').agg(aggFunction)
df1_weekly = df1_daily.resample('W').agg(aggFunction)
df1_14d = df1_daily.rolling(window=14, center=True).agg(aggFunction)

# using seaborn default style and set the default figure size
sns.set(rc={'figure.figsize':(16, 8)}); fig, ax = plt.subplots()

# start and end of the date range to extract
start, end = '2015-09', '2018-08'

ax.plot(df1_monthly.loc[start:end, 'AntennaDescription'], marker='o', linewidth=0.9,
label='Monthly Sum')
ax.plot(df1_weekly.loc[start:end, 'AntennaDescription'], marker='.', markersize=5, la
bel='Weekly Sum')
ax.plot(df1_14d.loc[start:end, 'AntennaDescription'], marker='.', linestyle='--', lab
el='14-d Rolling Sum')
ax.legend(ncol=3, fontsize=14)

ax.set_ylabel('Total Number of RFID Readings', fontsize=16)
# setting x-axis major ticks to monthly interval
ax.xaxis.set_major_locator(mdates.MonthLocator())
tick_loc = ax.get_xticks().tolist()
ax.set_xticks(tick_loc)
# formatting x-tick labels as '3-letter month' and '2-digit year'
ax.set_xticklabels(labels=tick_loc, rotation=90, fontsize=13)
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b \'%y'))
plt.show()

# creating a dataframe to count the number of panles produced each day
dfPanelCount = df1.groupby(['FirstReadDate', 'PanelNumber']).count()
dfPanelperDay = dfPanelCount.reset_index().groupby('FirstReadDate').agg({'PanelNumber
':'count'})

fig, ax = plt.subplots()
ax.plot(dfPanelperDay.loc['2015-09':'2015-12'].resample('W').sum(), label='2015')
ax.plot(dfPanelperDay.loc['2016-01':'2016-12'].resample('W').sum(), label='2016')
ax.plot(dfPanelperDay.loc['2017-01':'2017-12'].resample('W').sum(), label='2017')
ax.plot(dfPanelperDay.loc['2018-01':'2018-08'].resample('W').sum(), label='2018')
ax.legend(ncol=4, fontsize=14)

ax.set_ylabel('Number of Panels Produced Each Week', fontsize=16)
# setting x-axis major ticks to monthly interval
ax.xaxis.set_major_locator(mdates.MonthLocator())
tick_loc = ax.get_xticks().tolist()
ax.set_xticks(tick_loc)
# formatting x-tick labels as '3-letter month' and '2-digit year'
ax.set_xticklabels(labels=tick_loc, rotation=90, fontsize=13)
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b \'%y'))
plt.show()
```

3.2 Visualization of raw data in (df2)

↑ back to Heading 3

```
# using seaborn default style and set the default figure size
sns.set(rc={'figure.figsize':(16, 12)})
# plotting the heatmap of correlation between attributes
correlation = df2.corr(method='pearson')
sns.heatmap(correlation, xticklabels=correlation.columns, yticklabels=correlation.col
umns,
            cmap='RdBu_r', annot=True, linewidth=0.5, square=True)
plt.show()

# displaying summary statistics of attributes in df2
df2stat = df2.describe().T.loc[:, ['mean', 'std', 'min', '25%', '50%', '75%', 'max']]
.round()
```

```
display(df2stat)

# counting the number of interior and exterior panel as % of the total number of pane
ls
numIntExt = df2.loc[(df2['Type']=='INT') | (df2['Type']=='EXT')]['Type'].count()
numType = df2['Type'].count()

print("Number of 'INT' and 'EXT' panel types is {}% of all panels".format(round(numIn
tExt/numType*100, 2)))
```

4. Advanced Data Preparing

↑ back to TOC

4.1 Data Preparation in (df1)

```
# transforming df1 so that each station is represented by two columns ('stationName_i
nitial' and 'stationName_last')
# such that, for each record: 'stationName_initial' = minimum value found in 'Initial
ReadDateTime' and
#                              'stationName_last' = maximum value found in 'LastReadDa
teTime'

df1Transformed = (df1.groupby(["PanelNumber", "AntennaDescription", "FirstReadDate"])
# These are the indexes we want group by
.agg({"InitialReadDateTime": 'min', "LastReadDateTime": 'min'}) # We want to aggregat
ge based on two values of Time
.unstack(level="AntennaDescription") # Move station to column section
.droplevel(0,axis=1) # Reduce outermost multi-index of agg column 'Time'
.reset_index()) # completely remove multi-index and make simple table

# renaming columns in df1Transformed
df1Transformed.columns = ['PanelNumber', 'FirstReadDate', 'A1_initial', 'A2_initial',
'A3_initial', 'A4_initial',
                          'A5_initial', 'A1_last', 'A2_last', 'A3_last', 'A4_last', '
A5_last']
display(df1Transformed.head(5))
```

4.1.1 Replace Missing Values, Remove 'NaT', and Add 'Panels/Day'

```
# replace missing values at station 'A1' provided that the values in all other statio
ns are not missing
# the time at 'A1' is updated by randomly subtracting 8 to 12 minutes from the time a
t station 'A2'
fillA1i = lambda row: row["A2_initial"]-timedelta(minutes=np.random.uniform(8, 12)) i
f pd.isnull(row["A1_initial"]) & pd.notnull(row["A2_initial"])\
& pd.notnull(row["A3_initial"]) & pd.notnull(row["A4_initial"]) & pd.notnull(row["A5_
initial"]) else row["A1_initial"]
df1Transformed["A1_initial"] = df1Transformed.apply(fillA1i, axis=1)

fillA1l = lambda row: row["A2_last"]-timedelta(minutes=np.random.uniform(8, 12)) if p
d.isnull(row["A1_last"]) & pd.notnull(row["A2_last"])\
& pd.notnull(row["A3_last"]) & pd.notnull(row["A4_last"]) & pd.notnull(row["A5_last"]
) else row["A1_last"]
df1Transformed["A1_last"] = df1Transformed.apply(fillA1l, axis=1)
##################################################################
# replace missing values at station 'A2' provided that the values in all other statio
ns are not missing
# the time at 'A2' is closer to 'A1' than to 'A3', therefore, the time at 'A2' is upd
ated by
# adding 1/4 of the time difference (A3-A1) to the time at station 'A1'
fillA2i = lambda row: row["A1_initial"]+(row["A3_initial"]-row["A1_initial"])/4 if pd
.isnull(row["A2_initial"]) & pd.notnull(row["A1_initial"])\
& pd.notnull(row["A3_initial"]) & pd.notnull(row["A4_initial"]) & pd.notnull(row["A5_
initial"]) else row["A2_initial"]
df1Transformed["A2_initial"] = df1Transformed.apply(fillA2i, axis=1)

fillA2l = lambda row: row["A1_last"]+(row["A3_last"]-row["A1_last"])/4 if pd.isnull(r
ow["A2_last"]) & pd.notnull(row["A1_last"])\
& pd.notnull(row["A3_last"]) & pd.notnull(row["A4_last"]) & pd.notnull(row["A5_last"]
) else row["A2_last"]
df1Transformed["A2_last"] = df1Transformed.apply(fillA2l, axis=1)
#################################################################
# replace missing values at station 'A3' provided that the values in all other statio
ns are not missing
# the time at 'A3' is assumed to fall in the middle between 'A2' and 'A4', therefore,
the time at 'A3' is updated by
# adding 1/2 of the time difference (A4-A2) to the time at station 'A2'
fillA3i = lambda row: row["A2_initial"]+(row["A4_initial"]-row["A2_initial"])/2 if pd
.isnull(row["A3_initial"]) & pd.notnull(row["A1_initial"])\
& pd.notnull(row["A2_initial"]) & pd.notnull(row["A4_initial"]) & pd.notnull(row["A5_
initial"]) else row["A3_initial"]
df1Transformed["A3_initial"] = df1Transformed.apply(fillA3i, axis=1)

fillA3l = lambda row: row["A2_last"]+(row["A4_last"]-row["A2_last"])/2 if pd.isnull(r
ow["A3_last"]) & pd.notnull(row["A1_last"])\
& pd.notnull(row["A2_last"]) & pd.notnull(row["A4_last"]) & pd.notnull(row["A5_last"]
) else row["A3_last"]
df1Transformed["A3_last"] = df1Transformed.apply(fillA3l, axis=1)
#################################################################
# replace missing values at station 'A4' provided that the values in all other statio
ns are not missing
# the time at 'A4' is closer to 'A5' than to 'A3', therefore, the time at 'A4' is upd
ated by
# adding 3/4 of the time difference (A5-A3) to the time at station 'A3'
fillA4i = lambda row: row["A3_initial"]+3*(row["A5_initial"]-row["A3_initial"])/4 if
pd.isnull(row["A4_initial"]) & pd.notnull(row["A1_initial"])\
& pd.notnull(row["A2_initial"]) & pd.notnull(row["A3_initial"]) & pd.notnull(row["A5_
initial"]) else row["A4_initial"]
df1Transformed["A4_initial"] = df1Transformed.apply(fillA4i, axis=1)

fillA4l = lambda row: row["A3_last"]+3*(row["A5_last"]-row["A3_last"])/4 if pd.isnull
(row["A4_last"]) & pd.notnull(row["A1_last"])\
& pd.notnull(row["A2_last"]) & pd.notnull(row["A3_last"]) & pd.notnull(row["A5_last"]
) else row["A4_last"]
df1Transformed["A4_last"] = df1Transformed.apply(fillA4l, axis=1)
#################################################################
# replace missing values at station 'A5' provided that the values in all other statio
ns are not missing
# the time at 'A5' is updated by randomly adding 8 to 12 minutes to the time at stati
on 'A4'
fillA5i = lambda row: row["A4_initial"]+timedelta(minutes=np.random.uniform(8, 12)) i
f pd.isnull(row["A5_initial"]) & pd.notnull(row["A1_initial"])\
& pd.notnull(row["A2_initial"]) & pd.notnull(row["A3_initial"]) & pd.notnull(row["A4_
initial"]) else row["A5_initial"]
df1Transformed["A5_initial"] = df1Transformed.apply(fillA5i, axis=1)

fillA5l = lambda row: row["A4_last"]+timedelta(minutes=np.random.uniform(8, 12)) if p
d.isnull(row["A5_last"]) & pd.notnull(row["A1_last"])\
& pd.notnull(row["A2_last"]) & pd.notnull(row["A3_last"]) & pd.notnull(row["A4_last"]
) else row["A5_last"]
df1Transformed["A5_last"] = df1Transformed.apply(fillA5l, axis=1)
################################################################

# creating a data frame to count the number of panels per day, before the removal of
'NaT' values
dfPanelCount = df1Transformed.groupby(['FirstReadDate', 'PanelNumber']).count()
dfPanelperDay = dfPanelCount.reset_index().groupby('FirstReadDate').agg({'PanelNumber
':'count'})
# Add a new attribute that stores the number of panels produced each day
df1Transformed = pd.merge(left = df1Transformed,
                          right = dfPanelperDay.reset_index(),
                          left_on='FirstReadDate',
                          right_on='FirstReadDate')
df1Transformed.rename(columns={'PanelNumber_y':'Panels/Day', 'PanelNumber_x':'PanelNu
mber'}, inplace=True)

# removing rows with any remaining 'NaT' values from the dataframe
df1Transformed.dropna(axis=0, how='any', inplace=True)

# keeping records where the total number of panels per day is more than 5 and less th
an 80
df1Transformed = df1Transformed.loc[(df1Transformed['Panels/Day']>5) & (df1Transforme
d['Panels/Day']<=75)]
df1Transformed.reset_index(drop=True, inplace=True)

# plotting the distribution/histogram of Panels/Day
sns.set_style("ticks")
dailyPro = df1Transformed['Panels/Day'].mean()
print("Average Daily Production = {} panles/day".format(round(dailyPro, 2)))
print("Average Hourly Production = {} panles/hour".format(round(dailyPro/8, 2)))

plt.figure(figsize=(18, 12))
plt.hist(df1Transformed['Panels/Day'], bins=range(5,76, 5), rwidth=0.95, alpha=0.50,
label='Total production of five days')
plt.hist(df1Transformed['Panels/Day'], bins=range(5,76, 1), rwidth=0.70, alpha=0.95,
label='Single daily production value')

plt.axvline(x=30, linestyle='--', linewidth=1.6, color='red', alpha=0.6)
plt.axvline(x=55, linestyle='--', linewidth=1.6, color='red', alpha=0.6)

plt.annotate("Avg. Daily Production   = {} panles/day".format(round(dailyPro, 1)),
             xy=(15, 470), xycoords='axes points', fontsize=18)
plt.annotate("Avg. Hourly Production = {} panles/hour".format(round(dailyPro/8, 1)),
             xy=(15, 445), xycoords='axes points', fontsize=18)

plt.xlabel("Total Daily Production (panels/day)", fontsize=22)
plt.ylabel("Total Panel Count at Production Level", fontsize=22)
plt.xlim((3, 77)); plt.xticks(range(5, 80, 5), fontsize=16)
plt.yticks(range(0, 2001, 100), fontsize=16)
plt.grid(True); plt.legend(ncol=2, fontsize=20, loc='upper left'); plt.show()
```

4.1.2 Calculation of Actual Cycle Time - 'ActualCT'

```
# adding columns to calculate the time differences between consecutive antenna locati
ons using:
# calculateTimeDiff(initialTime, lastTime); each time difference corresponds to the c
ycle time at a single workstation
# For example, 'TotalTime12' --> 'Framing Station' and 'NoIdle12' --> 'Framing Statio
n' without idle time

stationNames = df1Transformed.columns[2:12]
for x in range(4):
    condition1 = lambda row: calculateTimeDiff(row[stationNames[x]], row[stationNames
[x+1]])
    newColumn1 = "Time"+str(x+1)+str(x+2)
    df1Transformed[newColumn1] = df1Transformed.apply(condition1, axis=1)

# dropping panels where the production is not completed in the same day
df1Transformed = df1Transformed.loc[~((df1Transformed['A5_last']-df1Transformed['A1_i
nitial']) > timedelta(hours=12))]
# only keeping records with positive value of cycle time at each workstation
df1Transformed = df1Transformed.loc[(df1Transformed["Time12"]>0) & (df1Transformed["T
ime23"]>0) &
                                    (df1Transformed["Time34"]>0) & (df1Transformed["T
ime45"]>0)]
df1Transformed.sort_values(by='FirstReadDate', inplace=True)
df1Transformed.reset_index(drop=True, inplace=True)

# Time between A1 and A5 is calculated by summing up the cycle times of individual wo
rkstations
total15 = lambda row: (row["Time12"] + row["Time23"] + row["Time34"] + row["Time45"])
df1Transformed["ActualCT"] = df1Transformed.apply(total15, axis=1)

# adding a column to calculate 'ActualCT' - break time
df1Transformed['ActualCT-BT'] = df1Transformed['ActualCT']

# if the production of a wall panel overlaps with a break time, subtract that break t
ime from the total cycle time
# break times are a) 15 minutes from 9:30 AM to 9:45 AM
#                 b) 30 minutes from 12:0 PM to 12:30 PM
#                 c) 15 minutes from 2:30 PM to 2:45 PM
for index, row in df1Transformed.iterrows():
    if (row['A1_initial'].time() < time(9,30)) & (row['A5_last'].time() > time(9,45))
:
        if (row['ActualCT-BT'] - 15) <= 0:
            pass
        else:
            df1Transformed.at[index, 'ActualCT-BT'] = row['ActualCT-BT'] - 15
    if (row['A1_initial'].time() < time(12,0)) & (row['A5_last'].time() > time(12,30)
):
        if (row['ActualCT-BT'] - 30) <= 0:
            pass
        else:
            df1Transformed.at[index, 'ActualCT-BT'] = row['ActualCT-BT'] - 30
    if (row['A1_initial'].time() < time(14,30)) & (row['A5_last'].time() > time(14,45
)):
        if (row['ActualCT-BT'] - 15) <= 0:
            pass
        else:
            df1Transformed.at[index, 'ActualCT-BT'] = row['ActualCT-BT'] - 15

display(df1Transformed.iloc[:,12:].head(5))

# plotting boxplots for station times and total cycle time, to visualize the distribu
tion of outliers
plt.figure(figsize=(12, 8))
data = [df1Transformed['Time12'], df1Transformed['Time23'], df1Transformed['Time34'],
        df1Transformed['Time45'], df1Transformed['ActualCT']]
labels = [i.name for i in data]

box = plt.boxplot(x=data, labels=labels, vert=True, showmeans=True, sym='r+')
plt.ylabel("Production Time (minutes)", fontsize=16)
plt.yticks(range(-5, 400, 20), fontsize=13)
plt.grid(True); plt.show()

# adding a dataframe to summarize the boxplot values (upper and lower limits)
getBoxplotData(box, labels)

# plotting the time-difference between antenna locations A1 and A5 (total time)
plt.figure(figsize=(15, 9))
sns.histplot(df1Transformed['ActualCT'], label='ActualCT', stat='count', kde=True, co
lor='red', alpha=0.5)
sns.histplot(df1Transformed['ActualCT-BT'], label='ActualCT-BT', stat='count', kde=Tr
ue,  alpha=0.8)

avg1, med1 = round(df1Transformed['ActualCT'].mean(), 1), round(df1Transformed['Actua
lCT'].median(),1)
plt.axvline(x=avg1, linestyle='--', linewidth=1.6, color='red')
avg2, med2 = round(df1Transformed['ActualCT-BT'].mean(), 1), round(df1Transformed['Ac
tualCT-BT'].median(), 1)
plt.axvline(x=avg2, linestyle='--', linewidth=1.6, color='blue')

plt.xlabel("Total Cycle Time (in minutes)", fontsize=16); plt.ylabel("Count", fontsiz
e=16)
plt.xticks(range(0, 141, 5), fontsize=13); plt.yticks(range(0, 751, 50), fontsize=13)
plt.xlim((0, 140))

plt.annotate("ActualCT mean and median:", xycoords='data', xy=(avg1-0.5, 600), fontsi
ze=15,
             xytext=(70, 610), arrowprops={'arrowstyle':'->', 'color':'black'})
plt.annotate("[{}, {}]".format(avg1, med1), xycoords='data', xy=(108, 610), fontsize=
15)
plt.annotate("ActualCT-BT mean and median:", xycoords='data', xy=(avg2-0.5, 550), fon
tsize=15,
             xytext=(70, 560), arrowprops={'arrowstyle':'->', 'color':'black'})
plt.annotate("[{}, {}]".format(avg2, med2), xycoords='data', xy=(108, 560), fontsize=
15)
plt.legend(ncol=2, fontsize=15); plt.grid(True, alpha=0.7, linestyle='--'); plt.show(
)

df1stat = df1Transformed.describe().T.loc[:, ['mean', 'std', 'min', '25%', '50%', '75
%', 'max']].round(2)
display(df1stat)
```

```
var1 = df1Transformed['ActualCT']; var2 = df1Transformed['ActualCT-BT']
cc, p_value = stats.pearsonr(var1, var2)
print("Correlation Coefficient (CC) between 'ActualCT' and 'ActualCT-BT' ={}\np-value
for CC ={}".format(cc, p_value))

# categorizing the target variable 'ActualCT' ---> 'CT_binned'
# using custom ranges for bins = {'low', 'below-norm', 'normal', 'above-norm', 'high'
}
group_names = ['low', 'below-norm', 'normal', 'above-norm', 'high']
avg, std = df1Transformed['ActualCT'].mean(), df1Transformed['ActualCT'].std()

# creating bin limits based on the average value and std. deviation of 'ActualCT'
l0, lmax = 0, df1Transformed['ActualCT'].max()
l25, l75 = df1stat.loc['ActualCT', '25%'], df1stat.loc['ActualCT', '75%']
capL, capH = l25 - (l75-l25), l75 + (l75-l25)
bins = [l0, capL, l25, l75, capH, lmax]

df1Transformed['CT_binned'] = pd.cut(df1Transformed['ActualCT'], bins, labels=group_n
ames, )
```

4.2 Data Preparation in (df2)

↑ back to Heading 4

```
# skipping the first 3 columns in df2 ('MultiPanel_Name', 'Length', 'Width'), a count
er is used to track this order in the loop
counter = 0

# calculate the percentage of the two most-occuring values in each column
# if the sum of these two values is 90% or more, this column(attribute) is to be dele
ted from df2

percentage1, percentage2, columnNames = [], [], []
# creating a list to store column names where the total percentage is 90% or more
columnsToDel = []

for columnName in df2:
    counter += 1
    # skipping the first 3 columns
    if counter > 3:
        val = df2[columnName].value_counts()
        columnNames.append(columnName)
        pr1 = round(100*val.tolist()[0]/sum(val.tolist()), 2); percentage1.append(pr1
)
        pr2 = round(100*val.tolist()[1]/sum(val.tolist()), 2); percentage2.append(pr2
)
        if (pr1+pr2) >= 90:
            columnsToDel.append(columnName)

# displaying all columns with the percentages of the two most-occuring values and thi
er sum
total = [sum(x) for x in zip(percentage1, percentage2)]
templist = list(zip(columnNames, percentage1, percentage2, (total)))
percentageDF = pd.DataFrame(templist, columns=["Attr.", "Value 1 freq. (%)", "Value 2
freq. (%)", "Both Values freq. (%)"])
percentageDF.set_index("Attr.", inplace=True); display(percentageDF)

# replacing values of the column 'Type' to numeric 'EXT', 'GAR', 'MEC' --> 1, 'INT',
'STR', 'NaN' --> 0
df2['Type'].replace(to_replace=['EXT', 'GAR', 'MEC', 'INT', 'STR', np.nan], value=[1,
1, 1, 0, 0, 0], inplace=True)

print(df2.columns)

df22 = df2.copy(deep=True)
# deleting rows where the following dimension limits apply:
# a)'Length' < 1200mm, b) 'Width' > 3300mm or < 1400mm
df22 = df22.loc[(df22["Length"] >= 1200) & (df22["Width"] >= 1400) & (df22["Width"] <
= 3300)]
cols = ['MultiPanel_Name', 'Length', 'Width', 'Height', 'Type', 'Window', 'LargeWindo
w', 'Door', 'LargeDoor', 'Sheetfull',
        'SheetPartial', 'Cutzone', 'Drillhole', 'Stud', 'DStud', 'LStud', 'MStud', 'B
lock', 'Backing', 'NailCount', 'Nailline']
df22 = df22[cols]
# Scenario 0, df22, original data frame
print("Scenario 0: df22   shape is", df22.shape)

# Scenario 1, df2_v1, add new attributes, created by combining existing attributes
df22_v1 = df22.copy(deep=True)
df22_v1['totalWD']      = df22_v1['Window'] + df22_v1['LargeWindow'] + df22_v1['Door'
] + df22_v1['LargeDoor']
df22_v1['TotalStuds']   = df22_v1['Stud'] + 2*df22_v1['DStud'] + 2*df22_v1['LStud'] +
3*df22_v1['MStud']
df22_v1['SheetFP']      = df22_v1['Sheetfull'] + df22_v1['SheetPartial']
df22_v1['BlockBacking'] = df22_v1['Block'] + df22_v1['Backing']
df22_v1.drop(columns=['Window', 'LargeWindow', 'Door', 'LargeDoor', 'Sheetfull', 'She
etPartial',
                      'Stud', 'DStud', 'LStud', 'MStud', 'Block',  'Backing'], inplac
e=True)
df22_v1.reset_index(drop=True, inplace=True)
print("Scenario 1: df22_v1 shape is", df22_v1.shape)

# Scenario 2, df2_v2, add new attributes, created by combining existing attributes
df22_v2 = df22_v1.copy(deep=True)
df22_v2['Components']   = df22_v2['totalWD'] + df22_v2['TotalStuds'] + df22_v2['Sheet
FP'] + df22_v2['BlockBacking']
df22_v2.drop(columns=['totalWD', 'TotalStuds', 'SheetFP', 'BlockBacking'], inplace=Tr
ue)
df22_v2.reset_index(drop=True, inplace=True)
print("Scenario 2: df22_v2 shape is", df22_v2.shape)

# Scenario 3, df2_v3, add new attributes, created by combining existing attributes
# this scenario is developed for the simulation model (i.e., Chapter 5)
df22_v3 = df22.copy(deep=True)
df22_v3['SmallOpenings'] = df22_v3['Window'] + df22_v3['Door']
df22_v3['LargeOpenings'] = df22_v3['LargeWindow'] + df22_v3['LargeDoor']
df22_v3['BlockBacking']  = df22_v3['Block'] + df22_v3['Backing']
df22_v3.drop(columns=['Height', 'Window', 'Door', 'LargeWindow', 'LargeDoor', 'Block'
, 'Backing'], inplace=True)
```

```
cols = ['MultiPanel_Name', 'Length', 'Width', 'Type', 'Cutzone', 'Drillhole', 'Stud',
'DStud', 'LStud', 'MStud',
        'SmallOpenings', 'LargeOpenings', 'BlockBacking', 'NailCount', 'Nailline',  '
Sheetfull', 'SheetPartial']
df22_v3 = df22_v3[cols]
df22_v3.reset_index(drop=True, inplace=True)
print("Scenario 3: df22_v3 shape is", df22_v3.shape)
```

5. Data Processing and Merging

↑ back to TOC

```
n1 = df1Transformed['FirstReadDate'].nunique()
print("Total number of production days = {} working days".format(n1))
```

5.1 Merging Dataframes and Generating Features

```
# create an empty list to store the final dataframes
df_final_list = []

# creating a list of cleaned dataframes of panel properties
df22_list = [df22, df22_v1, df22_v2, df22_v3]

for dataframe in df22_list:
    # merging 'df1Transformed' with each of the dataframes in 'df22_list'
    df1_merged = pd.merge(left=df1Transformed, right=dataframe, left_on="PanelNumber"
, right_on="MultiPanel_Name")
    df1_merged['WP'] = 0
    df1_merged['WPLength'] = 0
    df1_merged['WPTime'] = 0.0

    # extracting a slice of the data frame between start and end dates
    startDate, endDate = datetime(2015, 9, 1), datetime(2018, 8, 31)
    tempDF0 = df1_merged.loc[(df1_merged["FirstReadDate"] >= startDate) & (df1_merged
["FirstReadDate"] <= endDate)]
    tempDF0.reset_index(drop=True, inplace=True)

    # creating a list of unique dates by ignoring duplicates in 'FirstReadDate'
    listOfDates = pd.to_datetime(tempDF0['FirstReadDate'].unique().tolist())
    listOfDates = listOfDates.sort_values()

    counter = 0
    for day in listOfDates:
        tempDF1 = tempDF0.loc[tempDF0["FirstReadDate"] == day]
        tempDF1.reset_index(drop=True, inplace=True)

        for index, row in tempDF1.iterrows():
            # extracting the start time of processing a panel from the timestamp in '
A1_initial'
            panel_start = row['A1_initial']
            num, length, time = 0, 0, 0
            # for all other panels on the same day, check how many panels have alread
y been in production
            # and are still in-process (i.e. WIP) when the current panel enters the p
roduction
            for index2, row2 in tempDF1.iterrows():
                a1_other = row2['A1_initial']
                a2_other = row2['A2_initial']
                a3_other = row2['A3_initial']
                a4_other = row2['A4_initial']
                a5_other = row2['A5_initial']
                if (panel_start > a1_other) and (
                    (panel_start < a2_other) or (panel_start < a3_other) or (panel_st
art < a4_other) or (panel_start < a5_other)):
                    num += 1
                    length += row2['Length']
                    time += row2['ActualCT']
            tempDF1.at[index, 'WP'], tempDF1.at[index, 'WPLength'], tempDF1.at[index,
'WPTime'] = num, length, time

        counter += 1
        if counter > 1:
            df_final = pd.concat([df_final, tempDF1], ignore_index=True)
        else:
            df_final = tempDF1
        ##############################################################################
###############################

    df_final.drop(columns=['MultiPanel_Name', 'A2_initial', 'A3_initial', 'A4_initial
', 'A5_initial',
                           'A1_last', 'A2_last', 'A3_last', 'A4_last', 'A5_last'], in
place=True)

    # rearranging columns so target attributes 'ActualCT', 'ActualCT-BT', and 'CT_bin
ned' are the last columns in the dataframe
    CT_index = df_final.columns.get_loc('ActualCT')
    cols = df_final.columns.tolist()
    cols = cols[:CT_index] + cols[CT_index+3:] + cols[CT_index:CT_index+3]
    df_final = df_final[cols]

    # rearranging columns so that 'Panels/Day' is placed before the target columns in
the dataframe
    PD_index = df_final.columns.get_loc('Panels/Day')
    cols = df_final.columns.tolist()
    cols = cols[:PD_index] + cols[PD_index+1:-3] + [cols[PD_index]] + cols[-3:]
    df_final = df_final[cols]

    # sorting records by 'FirstReadDate'
    df_final.sort_values(by='FirstReadDate', ascending=True, inplace=True, ignore_ind
ex=True)
    df_final['FirstReadDate'] = df_final['FirstReadDate'].dt.date
    df_final_list.append(df_final)

# adding the raw dataset (without the generated attributes) to the list of dataframes
df_raw = df_final_list[0].drop(['WP', 'WPLength', 'WPTime', 'Panels/Day'], axis=1)
df_final_list.insert(0, df_raw)
print("Merging of all dataframes has been completed")

# using seaborn default style and set the default figure size
sns.set(rc={'figure.figsize':(16, 12)})
lengthIndex = df_final_list[2].columns.get_loc('Length')
# plotting the heatmap of correlation between attributes
pc = df_final_list[2].iloc[:, lengthIndex:].corr(method='pearson')
sns.heatmap(pc, xticklabels=pc.columns, yticklabels=pc.columns,
            cmap='RdBu_r', annot=True, linewidth=0.5, square=True)
plt.show()
```

5.2 Saving Merged Dataframes to CSV

↑ back to Heading 5

```
# saving final data frames to CSV files
names = ['df_raw', 'df_final', 'df_final_v1', 'df_final_v2', 'df_final_v3']
i = 0
for dataframe in df_final_list:
    file_name = "{}_{}_{}.csv".format(i+1, str(date.today()), names[i])
    dataframe.to_csv("CSV\{}".format(file_name), index=False)
    i += 1
```

6. Cycle Time (CT) Prediction

↑ back to TOC

6.1 Experiment A (1 Target Dat, 1 Model and 1 Dataframe)

```
# the variable 'df_id' refers to one of the four dataframes in the list 'df_final_lis
t' = {0, 1, 2, 3}
# df_0 = raw dataset without SL features
# df_1 = dataset with SL features with no feature combination
# df_2 = dataset with SL features with physical features combined into 4 features
# df_3 = dataset with SL features with physical features combined into 1 feature
df_i = 2; dataframe = df_final_list[df_i].copy(deep=True)

# specify the start and end dates for the experiment
sDate, eDate = date(2017,6,13), date(2017,6,21)

dataframe = dataframe.loc[(dataframe["FirstReadDate"] >= sDate) & (dataframe["FirstRe
adDate"] <= eDate)]
dataframe.reset_index(drop=True, inplace=True)
listOfDates = pd.to_datetime(dataframe['FirstReadDate'].unique().tolist()).sort_value
s()
print(listOfDates)

# splitting the dataframe into two sets: a training set, and a testing set
df_training = dataframe.loc[dataframe['FirstReadDate'] != listOfDates[-1]]
df_training.reset_index(drop=True, inplace=True)
df_testing = dataframe.loc[dataframe['FirstReadDate'] == listOfDates[-1]]
df_testing.reset_index(drop=True, inplace=True)
# storing independent features in the variable 'X' and the target attribute 'ActualCT
-BT' in the variable 'y'
# '7' represents the location of 'Lenght' and '-3' exclude targe variables from the i
ndependent variables 'X'
X_train = df_training.values[:, 7:-3]
y_train = df_training['ActualCT-BT'].values
X_test = df_testing.values[:, 7:-3]
y_test = df_testing['ActualCT-BT'].values

test_dim, train_dim = df_testing.shape[0], df_training.shape[0]
print("Testing set represents {}% of the whole data\n".format(round(test_dim/(test_di
m+train_dim), 2)*100))

# experimenting with a Random Forest model
clf = ensemble.RandomForestRegressor(n_estimators=1000, criterion='mse', random_state
=66)

print("Training the model:")
train_and_evaluate(clf, X_train, y_train)
print("Performance on the testing set:")
measure_performance(X_test, y_test, clf, False, False, False, True)
```

6.2 Experiment B (Many Target Days, 1 Model and 1 Dataframe)

↑ back to Heading 6

```
# experimenting with an ETR model
clf = ensemble.ExtraTreesRegressor(n_estimators=500, criterion='mse', random_state=66
)
df_i = 2; dataframe = df_final_list[df_i]

# specify the range of dates to check 'startDate' and 'endDate'
# and the length of historical timeframe to check 'lookbackTime'
startDate, endDate = date(2017,1,1), date(2017,12,31)
lookbackTime = 5

dataframe = dataframe.loc[(dataframe['FirstReadDate'] >= startDate) & (dataframe['Fir
stReadDate'] <= endDate)]
dataframe = dataframe.loc[(dataframe['Panels/Day'] >= 10) & (dataframe['Panels/Day']
<= 70)]

# determining the unique dates between startDate and endDate (eliminating duplicates)
uniqDays = dataframe['FirstReadDate'].unique()
```

Looping over Target Date Range (startDate --> endDate)

```
# a list to store the results
results = []
# creating an empty dataframe to append results to it
newDF = pd.DataFrame(columns = dataframe.columns)

for day in uniqDays[5:]:
    # locating the starting day of the prediction lookback timeframe
    sDayLoc = np.asarray(np.array(uniqDays) == day).nonzero()[0][0] - lookbackTime
    sDay = uniqDays[sDayLoc]
    #print("Start Day: {}, Testing Day: {}".format(sDay, day))

    # splitting the dataframe into two sets: a training set, and a testing set
    df_training = dataframe.loc[(dataframe['FirstReadDate'] >= sDay) & (dataframe['Fi
rstReadDate'] < day)]
    df_training.reset_index(drop=True, inplace=True)
    df_testing = dataframe.loc[dataframe['FirstReadDate'] == day]
    df_testing.reset_index(drop=True, inplace=True)
    # storing independent features in the variable 'X' and the target attribute 'Actu
alCT-BT' in the variable 'y'
    # '7' represents the location of 'Lenght' and '-3' exclude targe variables from t
he independent variables 'X'
    x_train = df_training.values[:, 7:-3]
    y_train = df_training['ActualCT-BT'].values
    x_test = df_testing.values[:, 7:-3]
    y_test = df_testing['ActualCT-BT'].values

    # fitting the model on the training set, and predicting target values on the test
ing set
    train_and_evaluate(clf, x_train, y_train, False)
    y_pred = clf.predict(x_test)

    # storing the predicted value in the dataframe, and appending the results to newD
```

```
F one day for each loop
    dayDF = dataframe.loc[dataframe['FirstReadDate'] == day]
    dayDF['PredictedCT'] = y_pred
    newDF = newDF.append(dayDF)

    MAE = metrics.mean_absolute_error(y_test, y_pred)
    MedAE = metrics.median_absolute_error(y_test, y_pred)
    MaxAE = metrics.max_error(y_test, y_pred)
    MAPE = metrics.mean_absolute_percentage_error(y_test, y_pred)
    CC = stats.pearsonr(y_test, y_pred)

    # A dictionary to aggregate the results of each subset
    dict1 = {}
    dict1['Start Date'], dict1['Target Date'] = sDay, day
    dict1['MAE'] = round(MAE, 2)
    dict1['MedAE'] = round(MedAE, 2)
    dict1['MaxAE'] = round(MaxAE, 2)
    dict1['MAPE'] = round(MAPE*100, 2)
    dict1['CC'] = round(CC[0], 2)
    results.append(dict1)

# creating a dataframe to summarize the results
resultsDF = pd.DataFrame(results)
display(resultsDF.head(10))

plt.figure(figsize=(16, 10))

# plotting the actual cycle time and the predicted results of the four models
sns.histplot(newDF['ActualCT-BT'], label='ActualCT-BT', stat='probability', kde=True,
color='red')
sns.histplot(newDF['PredictedCT'], label='PredictedCT', stat='probability', kde=True,
color='blue', alpha=0.5)

# plotting vertical lines for average values
avgAct = round(newDF.loc[:, 'ActualCT-BT'].mean(), 1); plt.axvline(x=avgAct, linestyl
e='--', linewidth=1.6, color='red')
plt.annotate("Avgerage actual CT = {} minutes".format(avgAct), xycoords='data', xy=(a
vgAct-0.2, 0.064), fontsize=16,
            xytext=(60, 0.061), arrowprops={'arrowstyle':'->', 'color':'black'})
avgPre = round(newDF.loc[:, 'PredictedCT'].mean(), 1); plt.axvline(x=avgPre, linestyl
e='--', linewidth=1.6, color='blue')
plt.annotate("Avgerage predicted CT = {} minutes".format(avgPre), xycoords='data', xy
=(avgPre-0.2, 0.060), fontsize=16,
            xytext=(60, 0.057), arrowprops={'arrowstyle':'->', 'color':'black'})

plt.xlabel("Production Cycle Time (in minutes)", fontsize=16); plt.xticks(range(0, 12
1, 5), fontsize=13); plt.xlim((0, 121))
plt.ylabel("Probability", fontsize=16); plt.yticks(np.arange(0, 0.071, 0.01), fontsiz
e=13)

plt.legend(ncol=2, fontsize=14); plt.grid(True); plt.show()
```

Experimenting with Different Lookback Timeframes

```
df_i = 2; dataframe = df_final_list[df_i]
lookbackTime = range(1, 31)

# lists to store the results
results = [[] for _ in range(len(lookbackTime))]
# creating empty dataframes to append results to it
newDFList = [pd.DataFrame(columns = dataframe.columns) for _ in range(len(lookbackTim
e))]

for day in uniqDays[len(lookbackTime):]:
    #print("\nTesting Day: {}".format(day))

    for index, lb in enumerate(lookbackTime):
        print('.', end='')
        # locating the starting day of the prediction lookback time
        sDayLoc = np.asarray(np.array(uniqDays) == day).nonzero()[0][0] - lb
        sDay = uniqDays[sDayLoc]
        #print("Start Day: {}, Testing Day: {}".format(sDay, day))

        # splitting the dataframe into two sets: a training set, and a testing set
        df_training = dataframe.loc[(dataframe['FirstReadDate'] >= sDay) & (dataframe
['FirstReadDate'] < day)]
        df_training.reset_index(drop=True, inplace=True)
        df_testing = dataframe.loc[dataframe['FirstReadDate'] == day]
        df_testing.reset_index(drop=True, inplace=True)
        # storing independent features in the variable 'X' and the target attribute '
ActualCT-BT' in the variable 'y'
        # '7' represents the location of 'Lenght' and '-3' exclude targe variables fr
om the independent variables 'X'
        x_train = df_training.values[:, 7:-3]
        y_train = df_training['ActualCT-BT'].values
        x_test = df_testing.values[:, 7:-3]
        y_test = df_testing['ActualCT-BT'].values

        # Fitting the model on the training set, and predicting target values on the
testing set
        train_and_evaluate(clf, x_train, y_train, False)
        y_pred = clf.predict(x_test)

        # storing the predicted value in the dataframe, and appending the results to
newDF one day for each loop
        dayDF = dataframe.loc[dataframe['FirstReadDate'] == day]
        dayDF['PredictedCT'] = y_pred
        newDFList[index] = newDFList[index].append(dayDF)

        MAE = metrics.mean_absolute_error(y_test, y_pred)
        MedAE = metrics.median_absolute_error(y_test, y_pred)
        MaxAE = metrics.max_error(y_test, y_pred)
        MAPE = metrics.mean_absolute_percentage_error(y_test, y_pred)
        CC = stats.pearsonr(y_test, y_pred)

        # A dictionary to aggregate the results of each subset
        dict1 = {}
        dict1['Start Date'], dict1['Target Date'] = sDay, day
        dict1['Lookback Time'] = lb
        dict1['MAE'] = round(MAE, 2)
        dict1['MedAE'] = round(MedAE, 2)
        dict1['MaxAE'] = round(MaxAE, 2)
        dict1['MAPE'] = round(MAPE*100, 2)
        dict1['CC'] = round(CC[0], 2)
        results[index].append(dict1)

# aggregating the results of all lookback timeframes performance metrics into one dat
aframe
resultsDFALL = pd.DataFrame(columns=pd.DataFrame(results[0]).columns)

for index, table in enumerate(results):
    resultsDF = pd.DataFrame(table)
    resultsDFALL = resultsDFALL.append(resultsDF)
display(resultsDFALL.sort_values(by='MAPE').head(50))

resultsDFALL.reset_index(drop=True, inplace=True)
grp = resultsDFALL.groupby(by='Lookback Time')
groupedDF = grp.min().reset_index()

plt.figure(figsize=(14, 7))
x = groupedDF['Lookback Time'].tolist()
y = groupedDF['MAPE'].tolist()
sns.regplot(x=x, y=y, order=6, ci=0, scatter_kws={'s':60}, marker='s')

plt.xlabel("Lookback Timeframe (days)", size=16); plt.xticks(range(1,31), fontsize=14
, rotation=0)
plt.ylabel("MAPE (%)", size=16); plt.yticks(fontsize=14, rotation=0)
plt.grid(True); plt.show()
```

6.3 Full Experiment (1 target day, 4 models and 4 dataframes)

↑ back to Heading 6

```
# a list storing the names of the four dataframes, which will be used in the loop to
iterate over these dataframes
datasetNames = ['df_raw', 'df_SLfeatures', 'df_SLfeaturse_v1', 'df_SLfeatures_v2']
# specify the start and end dates for the experiment
sDate, eDate = date(2017,5,24), date(2017,6,12)

# a list to store the four dataframes with predicted values for each model
df_predicted = []

# looping over the four dataframes and for each one, train the four models (LR, KNN,
RF, and NN)
# and calculate five performance metrics; then display the results for each dataframe
in a table
for df_i in range(len(datasetNames)):
    print('----- Experiment on dataframe "{}" -----'.format(datasetNames[df_i]))
    # the variable 'df_id' refers to one of the four dataframes in the list 'df_final
_list' = {0, 1, 2, 3}
    dataframe = df_final_list[df_i].copy(deep=True)

    dataframe = dataframe.loc[(dataframe["FirstReadDate"] >= sDate) & (dataframe["Fir
stReadDate"] <= eDate)]
```

```
    listOfDates = pd.to_datetime(dataframe['FirstReadDate'].unique().tolist()).sort_v
alues()
    #print(listOfDates)

    # test set consists of records of one day - End Date 'eDate'
    # training set consists of recoreds in all working days from 'sDate' until one da
y before 'eDate'
    df_training = dataframe.loc[(dataframe['FirstReadDate'] != listOfDates[-1])]
    df_training.reset_index(drop=True, inplace=True)
    df_testing = dataframe.loc[(dataframe['FirstReadDate'] == listOfDates[-1])]
    df_testing.reset_index(drop=True, inplace=True)
    # storing independent features in the variable 'X' and the target attribute 'Actu
alCT-BT' in the variable 'y'
    # '7' represents the location of 'Lenght' and '-3' exclude targe variables from t
he independent variables 'X'
    X_train = df_training.values[:, 7:-3]
    y_train = df_training['ActualCT-BT'].values
    X_test = df_testing.values[:, 7:-3]
    y_test = df_testing['ActualCT-BT'].values

    reg_1 = linear_model.LinearRegression(normalize=True)
    reg_2 = neighbors.KNeighborsRegressor(n_neighbors=11, weights='distance', p=2)
    reg_3 = ensemble.RandomForestRegressor(n_estimators=1000, criterion='mae', random
_state=66)
    reg_4 = neural_network.MLPRegressor(hidden_layer_sizes=(52, 52), solver='lbfgs',
                                        activation='relu', max_iter=2000, random_stat
e=99)

    modelList, modelName = [reg_1, reg_2, reg_3, reg_4], ['LR', 'KNN', 'RF', 'NN']
    scores, counter = [], 0
    for index, model in enumerate(modelList):
        print("* Model: {}".format(modelName[index]))
        # Fitting the model on training set, and predicting the target on testing set
        train_and_evaluate(model, X_train, y_train, True)
        y_pred = model.predict(X_test)
        # store the predicted values in the dataframe
        col_name = 'pred_' + modelName[counter]
        counter += 1
        df_testing[col_name] = y_pred

        MAE = metrics.mean_absolute_error(y_test, y_pred)
        MedAE = metrics.median_absolute_error(y_test, y_pred)
        MaxAE = metrics.max_error(y_test, y_pred)
        MAPE = metrics.mean_absolute_percentage_error(y_test, y_pred)
        CC = stats.pearsonr(y_test, y_pred)
        # a dictionary to aggregate the results of each run
        dict1 = {}
        dict1['MAE'] = round(MAE, 2)
        dict1['MedAE'] = round(MedAE, 2)
        dict1['MaxAE'] = round(MaxAE, 2)
        dict1['MAPE'] = round(MAPE*100, 2)
        dict1['CC'] = round(CC[0], 2)
        scores.append(dict1)

    df_predicted.append(df_testing)
```

```
# create and display a dataframe of the results for each dataset
scoresDF = pd.DataFrame(scores, index=modelName)
display(scoresDF.T)
```