Performance Modeling of Human-Machine Interfaces using Machine Learning

by

Anjian Wu
B.S. Electrical Engineering
California Institute of Technology, 2014

Submitted to the MIT Sloan School of Management and the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of

Master of Business Administration
and
Master of Science in Electrical Engineering and Computer Science

In conjunction with the Leaders for Global Operations Program at the
Massachusetts Institute of Technology

June 2019

Signature redacted

Author ____________________
MIT Sloan School of Management
Department of Electrical Engineering and Computer Science
May 10, 2019

Signature redacted

Certified by ____________________
Randall Davis
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Certified by ____________________
Yanchong Zheng
Associate Professor of Operation Management
Thesis Supervisor

Signature redacted

Accepted by ____________________
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Signature redacted

Accepted by ____________________
Maura Herson
Assistant Dean, MBA Program
MIT Sloan School of Management

This page has been intentionally left blank.

Performance Modeling of Human-Machine Interfaces using Machine Learning

by

Anjian Wu

Submitted to the MIT Sloan School of Management and the Department of Electrical Engineering and Computer Science on May 10, 2019, in Partial Fulfillment of the Requirements for the Degrees of Master of Business Administration and Master of Science in Electrical Engineering and Computer Science

Abstract

As the popularity of online retail expands, world-class electronic commerce (e-commerce) businesses are increasingly adopting collaborative robotics and Internet of Things (IoT) technologies to enhance fulfillment efficiency and operational advantage. E-commerce giants like Alibaba and Amazon are known to have smart warehouses staffed by both machines and human operators.

The robotics systems specialize in transporting and maneuvering heavy shelves of goods to and from operators. Operators are left to higher-level cognitive tasks needed to process goods such as identification and complex manipulation of individual objects.

Achieving high system throughput in these systems require harmonized interaction between humans and machines. The robotics systems must minimize time that operators are waiting for new work (idle time) and operators need to minimize time processing items (takt time). Over time, these systems will naturally generate extensive amounts of data. Our research provides insights into both using this data to design a machine-learning (ML) model of takt time, as well as exploring methods of interpreting insights from such a model.

We start by presenting our iterative approach to developing a ML model that predicts the average takt of a group of operators at hourly intervals. Our final XGBoost model reached an out-of-sample performance of 4.01% mean absolute percent error (MAPE) using over 250,000 hours of historic data across multiple warehouses around the world.

Our research will share methods to cross-examine and interpret the relationships learned by the model for business value. This can allow organizations to effectively quantify system trade-offs as well as identify root-causes of takt performance deviations. Finally, we will discuss the implications of our empirical findings.

Thesis Advisor: Randall Davis, Thesis Supervisor
Title: Professor of Electrical Engineering and Computer Science

Thesis Advisor: Yanchong Zheng
Title: Associate Professor of Operation Management, MIT Sloan School of Management

This page has been intentionally left blank.

Acknowledgements

The opportunity to study at MIT has fulfilled a long-standing dream that began in my high school years, when my curiosity in science and engineering manifested in the form of Department of Energy science bowls, math and economics competitions, and FIRST Robotics. I want to acknowledge and share my gratitude to the following individuals who made my MIT experience possible.

First, I would like to share my appreciation to my advisors Professor Davis and Professor Zheng. Since the inception of my project, both took time each month to provide guidance. Their dedicated and consistent involvement throughout my work was instrumental to the project's successes.

I would like to thank Augusta Niles (LGO Class of 2014) and Aaron Small (LGO Class of 2017). Augusta offered me the opportunity to join her team at Amazon and supported my work throughout the internship. I am indebted to my company supervisor Aaron for providing technical resources and leadership guidance before, during, and after my internship.

I would like to thank the LGO program. I am grateful that they believed in my potential and offered me a once in a lifetime opportunity to study engineering and management at MIT. I also would like to thank the Noyce Foundation for originally endowing the fellowship which has generously supported my education.

I would like to thank my fiancé Alice for her love and support throughout the last two years.

Lastly, I am deeply grateful for my mother and father, Huiqing and Yifeng. My father's journey, from being the first in the family to finish college in China to his pursuit of the American dream in the United States, has always been an inspiration to me. I attribute many of my life achievements to the foundations he laid and his encouragement.

This page has been intentionally left blank.

List of Figures

List of Tables

1 Introduction

1.1 Background

Retail e-commerce has seen explosive growth in the last few years. In 2017, retail e-commerce sales worldwide amounted to 2.3 trillion US dollars (24% year-over-year growth). Looking ahead, e-retail revenues are projected to grow at an average yearly rate of 20% and reach 4.88 trillion US dollars in 2021 [1].

As the industry has grown, so too will consumer expectations. Customers increasingly seek innovations that make their shopping experience more convenient, cost-effective, and fast. In response to this, e-commerce companies have come under increasing strain to maintain low costs and short times of delivery.

1.2 Early Industry History (1994 – 2012)

The history of e-commerce is closely tied to the history of the internet. Following soon after the public introduction of the internet in 1991, people around the world began to adopt online shopping.

Amazon.com (or Amazon) was one of the first e-commerce sites in the US to start selling products online. Founded in 1994 by Jeff Bezos, Amazon started as an online retailer of books and grew quickly. By 1997, Amazon had reached 1 million customer accounts and expanded their offerings to products including music, video, consumer electronics, video games, software, home improvement items, and much more [2].

At that time, two major strategic moves emerged. One is that Amazon began holding inventory in privately owned warehouses which allowed for more strict quality and cost control. Second, Amazon further expanded product offerings by other 3rd party sellers onto the Amazon platform. Other websites were able to offer merchandise for sale and Amazon.com would fill the order and receive a commission.

Other entrances into the e-commerce industry were not so fortunate. In 1996, Webvan was founded as an e-commerce company promising delivery of groceries within 30 minutes of ordering by leveraging state-of-the-art order fulfillment centers advanced automation. Unlike Amazon, Webvan's focus was on chasing the razor-thin grocery, drugstore, and prepared meals market first, which was estimated to be worth $650 billion in 1998 [3]. Webvan was focused on building critical mass, order frequency, and economies of scale by leveraging automation.

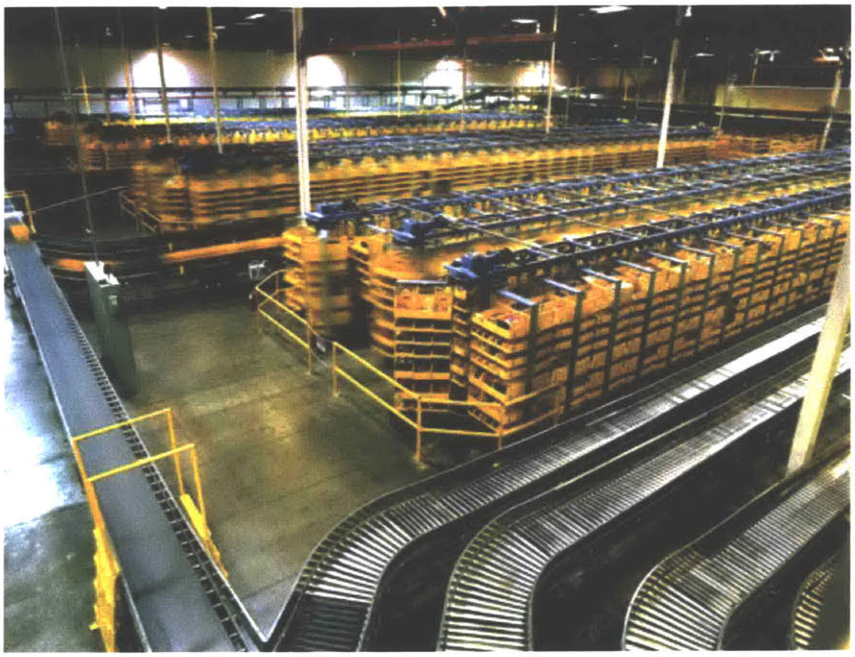


Figure 1: A look into Webvan's carousel automation technology

Investors at that time were flocking to fund Webvan. By 1999, Webvan received around $400 million in four rounds of venture capital financing, the most for an Internet company in 1999. Unfortunately, by 2001, the company had filed for bankruptcy. The internet was still in its infancy and Webvan had grossly overestimated market adoption. On top of that, Webvan's claim of massive cost savings by leveraging their carousel automation solution was unfounded in practice (see Figure 1) [4]. It turned out that many fresh and frozen goods, as well as heavy and fast-moving items, were not compatible with the automated technology available at that time.

On the other hand, Amazon survived the turn of the century and continued its mission to become the "Earth's most customer-centric company" by continually expanding greater product selection, faster delivery and lower prices. By 2006, Amazon launched Fulfillment by Amazon, which gave small businesses and 3rd party sellers the ability to use Amazon's own order fulfillment and customer service infrastructure.

Up until this point, however, Amazon relied largely on labor-intensive material handling at their fulfillment centers.

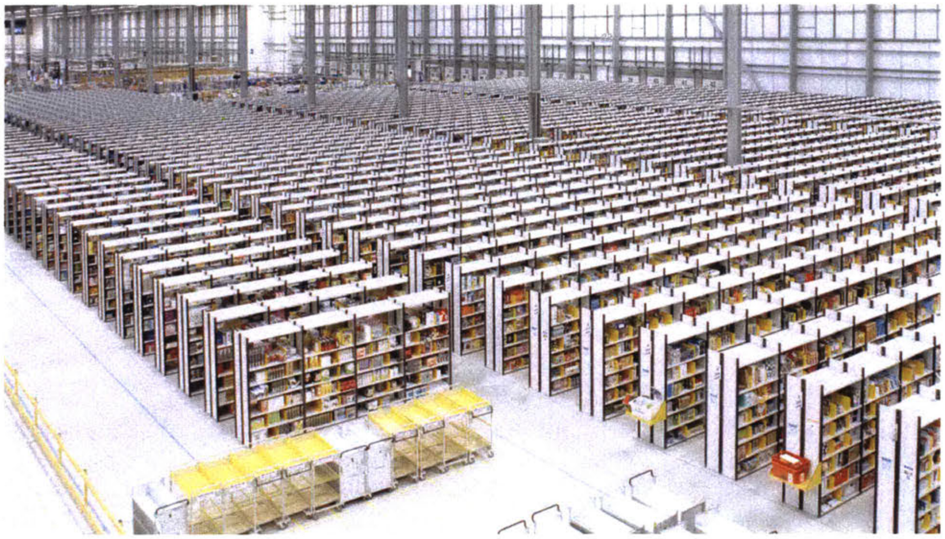


Figure 2: Amazon fulfillment center in the British Midlands in 2013

Operations were mainly comprised of human workers wandering these massive warehouses (see Figure 2), manually storing incoming products, pulling orders down from shelves, or packing them up for shipment [5].

1.3 Later Industry History (2012 - Now)

In 2012, Amazon acquired Kiva Systems, a robotics company that pioneered automated inventory fulfillment robotics. By 2014, Amazon begins integrating Kiva robots into their fulfillment centers [6]. Instead of humans walking around a vast warehouse, the robotics system will bring large containers of inventory to the humans.

This human-machine system frees humans from labor-intensive duties. Humans operators are left to focus on more cognitive-intensive tasks such as item identification, complex manipulation, and dealing with unstructured problems.

1.3.1 Modern Order Fulfillment

Today, Amazon has a network of fulfillment centers (see Figure 3) placed strategically across the world [7]. This allows them to service different regions where their customers make purchases online. We refer to these centers as "operational sites" for the remainder of our discussion.

Figure 3: Aerial view of Amazon fulfillment center

Each of these operational sites may contain multiple floors of robotics systems. An example of a robotics floor is shown in Figure 4 [8]. Each robotics floor contains the robots that move shelves of inventory. Humans are forbidden to enter into the field during normal operations.

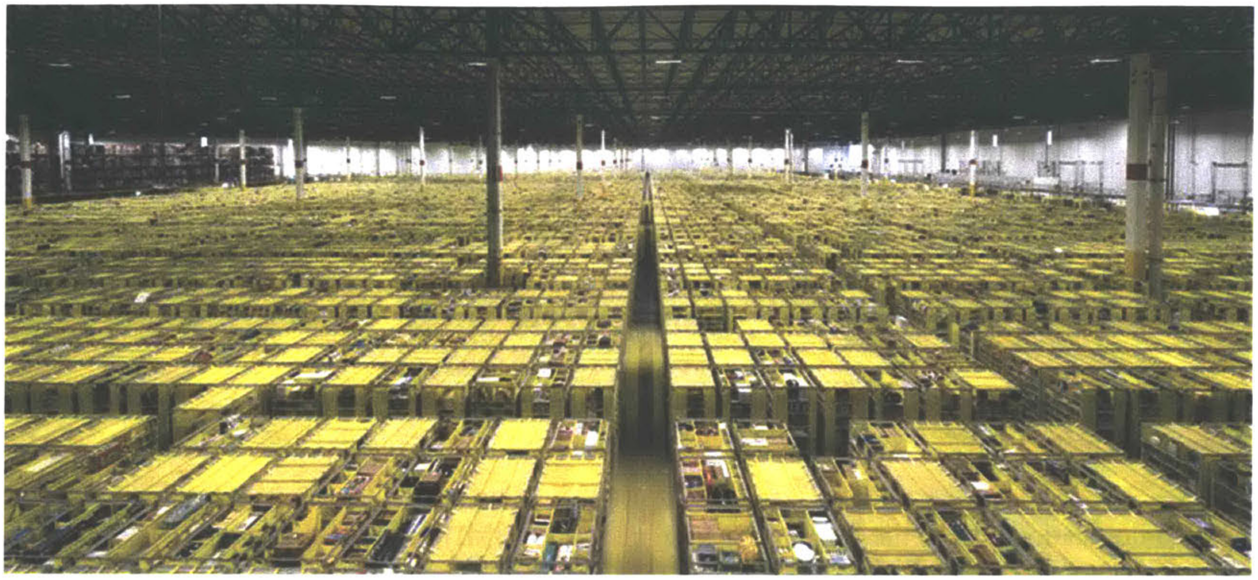


Figure 4: Inside glimpse into a single robotics floor in New Jersey

Operator stations are scattered along the outer perimeter of the robotics floor. These are the stations where the robots will eventually drop off a shelf of inventory for a human to process.

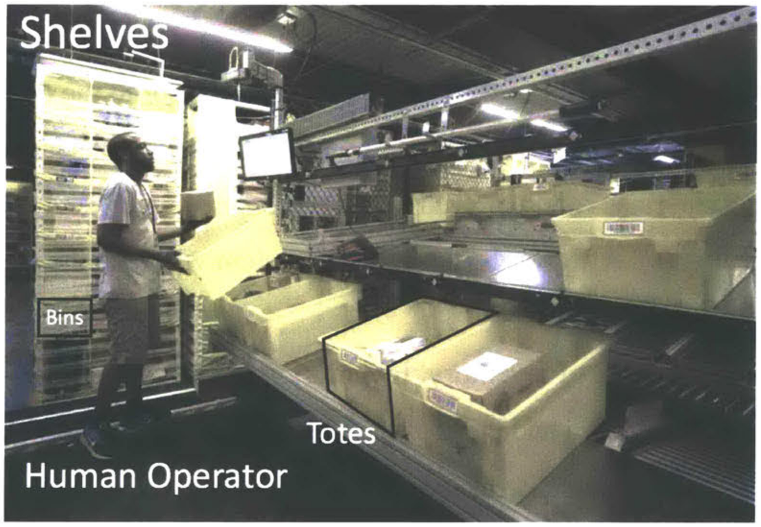


Figure 5: A typical operator station with key elements outlined

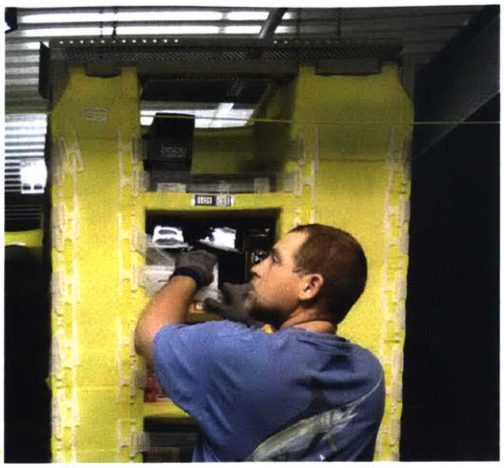


Figure 6: A closer view of an operator reaching for a target item in the target bin

The operator station acts as the human-machine interface (HMI) between the operator and the robotics system. Here are elements of an operator station key to our discussion (see Figure 5) [9].

- Operator – A warehouse associate tasked with processing inventory to-and-from shelves brought to them by the robotics system.
- Stations – Scattered in fixed locations along the outer perimeter of the robotics floor, stations are where the robots drop off shelves. Each active station is staffed with an operator.
- Shelves - The robotics system moves shelves across the floor to the operators. Each shelf has four faces. Each face contains an assortment of individual bins laid out in a grid-like manner. Once a shelf has safely arrived to the station, operator visits the shelf once to pull an item or item(s). Afterwards, the robotics system will take the shelf away and move a new shelf into position.
- Bins – Bins contain a mix of inventory. Inventory is removed manually by operators by simply reaching into the bin and pulling the item out as shown in Figure 6 [10].
- Totes – The containers where items are placed after being pulled from the shelves. As the Tote gets more items after multiple shelf visits, the user interface will notify the operator when each Tote is considered complete. When a Tote is complete, the operator is instructed to push the Tote forward where it will be moved to downstream processes. The operator then puts a new empty Tote to replace the prior Tote.
- Target item(s) – Item(s) that the operator must find from each visit to a new shelf. The user interface will then instruct which tote the target item(s) should be placed in.

- Target Bin – The bin containing the Target item(s).
- Target Tote – The Tote assigned for the Target item(s) to be placed in.
- Position of Tote – There are multiple fixed locations where the Totes are positioned. We define them as Position 1, 2, 3, 4, 5, and 6, where Position 1 is closest to the Shelf face. In general, the more Tote positions that are actively used, the more orders can be processed concurrently.

The human-machine operations relevant to our project is shown in Figure 7 from the perspective of the operator.

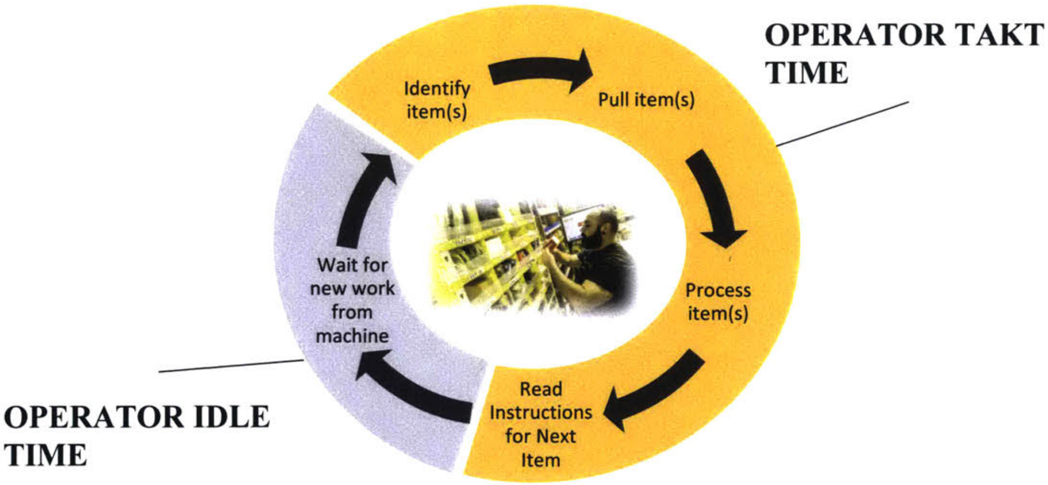


Figure 7: Process cycle of interest in our project from the operator perspective

The operator gets information from the system about the next target items that need to be processed. Then the operator waits as the robotics system brings the next shelf of inventory. After the new shelf arrives, the operator then identifies the target bin on the shelf. The target items are pulled out, processed, and placed into a tote. Afterwards, the robotics system will take the shelf away and move a new shelf into position. This process then repeats. We refer to this cycle as the process cycle.

We denote the portion of the process cycle where the operator is active as human takt time or takt time (see Figure 7). We denote the remaining time, where the operator is waiting for the robotics system to replenish new work, as idle time (see Figure 7).

This robotics system enhances overall system performance by isolating the tasks done by machines and humans in a manner that leverages their respective strengths. For example, machines bear the brunt of the physical labor by lifting and moving shelves of inventory across multiple miles daily [11}. On the other hand, the operators are left with higher-level object recognition and manipulation tasks.

2 Project Overview

2.1 Problem Statement

Achieving high system throughput in these systems require harmonized interaction between humans and machines. The robotics systems must minimize time that operators are waiting for new work (idle time) and operators need to minimize time spent processing items (human takt time or takt time).

One metric of this human-machine interface (HMI) performance is units processed per hour (UPH), which is a measurement of the average number of items one operator processes in an hour as shown in Equation 1.

$$\text{UPH} = \frac{\text{Total Units Processed}}{\text{Total Hours Worked}} \tag{1}$$

Diving deeper, we can breakdown the UPH metric into three multiplicative components: items per process cycle (IpC), operator takt time, and operator utilization as shown in Equation 2.

$$\text{UPH} = \frac{\text{units}}{\text{hour}} = 3600\left(\frac{\text{seconds}}{\text{hour}}\right) \times \frac{\text{Items per Cycle}\left(\frac{\text{units}}{\text{cycle}}\right)}{\text{Operator Takt}\left(\frac{\text{seconds}}{\text{cycle}}\right)} \times \%_{\text{operator utilization}} \tag{2}$$

A process cycle (or cycle) is comprised of idle time (operator is waiting for a new shelf) and operator takt (operator is actively processing a shelf). Operator utilization is the percentage of the process cycle where the operator is actively processing a shelf. It is defined in Equation 3.

$$\%_{\text{operator utilization}} = \frac{\text{Operator Takt}}{\text{Operator Takt} + \text{Idle Time}} \tag{3}$$

Items per process cycle (IpC) is the average number of items that an operator handles during a single process cycle. IpC is equal to or greater than 1 because an operator can handle 1 or more items per cycle. In our system, the only time an operator is instructed to pull more than 1 item in a process cycle is when the items being pulled together are identical.

Operator takt, as outlined in Figure 7, is measured in seconds per cycle and defined as the time during the cycle where the operator is actively handling items.

The value resulting from dividing IpC by operator takt measures the average number of units per second processed per operator when work is available. Multiplying this by the operator utilization gives the overall number of units per second per operator, which can then be converted into UPH.

Looking at this relationship, we can see that there are three ways to increase UPH. Namely, they can increase IpC, decrease operator takt time, or increase operator utilization.

This mathematical formulation shows that improving UPH requires both machines and human to improve together. However, modeling both sides of this problem poses different challenges. Machines can be consistently manufactured and programmed, allowing for straightforward simulation and modeling. Humans are quite the opposite. Unlike machines, humans have much more variability due to factors such as different prior experience, learning curves, physiology, and other hard to quantify factors. To address this challenge, our research focuses only on modeling operator takt time.

2.2 Goals

In this thesis, we outline an approach to designing and applying a machine-learning (ML) model of operator takt time in industry. We will present three areas of contribution in support of this goal. First, we will discuss an iterative approach to building a ML model of takt. The ML model attempts to model how various factors influence the average takt of a group of operators working at the same robotics floor. Our machine learning approach uses over 250,000 hours of historic data from different operational sites across North America. The final ML model achieves an out-of-sample performance of 4.01% mean absolute percent error (MAPE), compared to the MAPE of 9.21% when using a simple sample mean as the prediction, and the out-of-sample coefficient of correlation R value is 88.25%. Second, we will propose different methods to interrogate and interpret the relationships learned by the model. Lastly, we will share insights generated from these methods.

3 Literature Review

Several papers have discussed classification and regression approaches to modeling HMI metrics. Here, we present a few most relevant to our project.

Nicholas Paperno, et al. (2016) published a study to model a user's performance on completing specific tasks when operating an assistive robotic manipulator [12]. Leveraging prior research done on the identification of ten potential human factors, they build a model using data measuring dexterity (gross and fine), spatial abilities (orientation and visualization), visual acuity in each eye, visual perception, depth perception, reaction time, and working memory. They collect this data on 89 individuals who had to complete controlled tests of these human factors and then subsequently completed several tasks using a robotic manipulator designed to simulate find-and-fetch/pick-and- place tasks. They found that speed of information processing, spatial ability, dexterity, and working memory were all significant predictors of task performance. They used both linear and polynomial models with out-of-sample performance of 7.3% root mean percent error when predicting time per task. For number of moves per minute, they use a polynomial model which showed a 9.1% error.

Iwase and Murata (2001) published a new model extending Fitts's model by predicting movement time as a function of an index of difficulty (ID) of a three-dimensional pointing task [13]. Input to the model were target size, target distance, and approach direction to the target. Pointing time and coordinate of pointer finger were measured. It was shown that the trajectory length and mean velocity were greatly affected by the approach direction. Using this data, they defined the ID as a deterministic logarithmic function of those inputs and achieved an R^2 of 0.698.

Perez-D'Arpino and Shah (2015) published a data-driven approach that synthesizes anticipatory knowledge of both human motions and subsequent action steps in order to predict the intended target of a human performing a reaching motion when working with a collaborative robot agent [14]. They produce a library of motions from empirical human demonstrations, based on a statistical representation of the degrees of freedom of the human arm, using time series analysis, wherein each time step is encoded as a multivariate Gaussian distribution. They achieve 70% or higher correct classification on average for the first third of the trajectory (< 500msec). This model can then be used as an anticipatory signal for the robot agent to adjust its next action to avoid conflict between human and robot motions.

The majority of these related academic publications were done in controlled environments, using data captured on wearable sensors, video cameras, and 3D motion cameras. Unfortunately, this type of data is unlikely to be available at a significant scale in industry and the cost for adoption may be hard to justify. Instead, this project focuses on building an HMI predictive model focused on a widely adopted HMI system that naturally generates extensive amounts of data.

3 Desired Model Qualities

How practitioners approach the model building process depends on the model's intended use-case.

At one end of the spectrum, a practitioner might only be concerned about the model accuracy. For example, the problem of predicting next week's weather. Because humans cannot materially affect near-term weather, a satisfactory model is one that provides the highest accuracy, regardless of the model's complexity or interpretability. On the other end of the spectrum, practitioners also want to intervene to change the predicted outcome.

In our project, we designed our model to be practical for humans to glean actionable insights around drivers of operator takt, not solely to produce the most accurate operator takt model possible. Thus, the four desirable qualities that the model should strive towards are highlighted in Table 1.

Desired Quality	**Details**
Accuracy (output)	The model should achieve sufficient out-of-sample accuracy.
Human Interpretability (inputs)	The model should use a set of features (inputs) that humans can comfortably interpret. Models with numerous and convoluted features can become difficult for human interpretation.
Independence (inputs)	The model should use a set of features that are mutually-independent. Models with highly correlated inputs make isolating the individual contributions to the prediction difficult.
Direct Causality to Takt (inputs)	The model should use a set of features that have a logical and causal link to operator takt. The underlying model structures learned through ML algorithms may not always reflect the exact physical mechanisms or causal relationships linking the observed outcomes to the features. Thus, we must leverage domain-knowledge in selecting only features that have a logical and causal link to operator takt.

Table 1: The four desirable qualities of the model.

4 Model Development

4.1 Iterative Design Philosophy

Developing the right ML model given limited time constraints is inherently risky. In practice, there is often a gap between people who implement ML and people who have domain expertise. Also, model development might be hindered by the organization's level of data maturity such as availability of historic data and ability to collect new data. Lastly and most importantly, the experimental nature of ML almost always means that the first trained model rarely is the best solution, requiring multiple iterations before a sufficient solution is discovered.

Within the time-constraints of the project, we mitigate these risks by allocating sufficient time for several iterations of the iterative design process as shown in Figure 8.

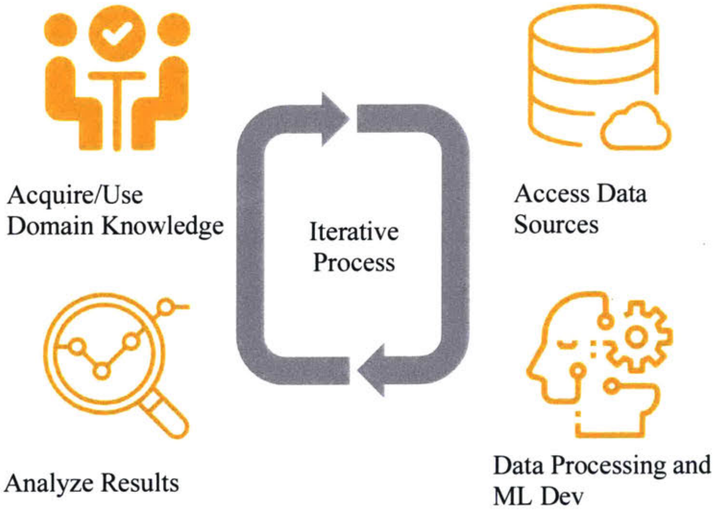


Figure 8: Iterative Design Approach

During early cycles of development, we focus on iterating as cheaply and quickly as possible. For example, we started the project by brainstorming a comprehensive list of features believed to be associated with Takt. Then, we worked to collect, preprocess, train, and analyze only a small subset of those features with sufficient number of observations that can be easily saved and processed by a single local machine.

After each iteration, we quickly learn more about where to further invest efforts. For example, we may expand the number of features, duration of historic observations, or types of ML algorithms explored.

In the later stage of model iteration, we move to cloud-based computing to scale model development with more historic observations across multiple facilities in different geographical locations. Data is aggregated and pre-processed using Hive on Spark running on Amazon Web Services (AWS). Apache Spark is an open-source distributed general-purpose cluster-computing framework, which enables data aggregation and computation on vast dataset.

The four parts of the cycle will be discussed in detail in the following sections. The main idea is that we keep iterating until the model sufficiently satisfies the four desirable qualities outlined earlier.

4.2 Domain Knowledge

Acquiring and leveraging domain knowledge is one of the most valuable activities when building and applying ML models. As a practitioner learns more about the domain, they will develop better intuitions around feature exploration, selection, and preprocessing. Considering that feature causality is an important quality in our application, domain knowledge is also a critical feedback mechanism when analyzing model performance.

The sources of domain knowledge that we pursued are subject matter experts, user observation, and firsthand experience. In the following sections, we will cover each method and briefly discuss their tradeoffs.

4.2.1 Subject Matter Experts

The Subject Matter Experts (SMEs) we interacted with can be divided into two groups. The first group were the engineers who design, maintain, and trouble-shoot the HMI system.

The advantage was that they typically have insights into the technical design of the system with years of experience debugging common issues from multiple operational site. The disadvantage was that they may have less visibility into how operators use the system day-to-day.

The second group was people who manage operations on site. The advantage was that these people have years of hands-on experience with floor-level workers. They also were largely in charge of designing the incentive structure and work schedule for the operators. The disadvantage was that they may have less technical understanding of the robotics system.

4.2.2 Naturalistic User Observation in the Wild

Our host company regularly sends teams to visit operational sites to better understand and improve the processes. We also took this opportunity to passively observed users “in the wild”. This approach largely comprised of us standing a comfortable distance away from the operators as they used the HMI system. As we observed, we recorded qualitative observations as we see fit.

The benefit to naturalistic user observation was that it can be more cost effective than controlled experimentation or user interviews. Considering also that users may not always effectively verbalize their perceptions or experiences, user observation provided actual user behavior.

On the other hand, naturalistic observation was less structured, more qualitative, and time consuming. Observations gathered may be hard to reproduce, not fully representative of the user population, or influenced by the observer’s biases.

4.2.3 Firsthand experience

Lastly, we immersed ourselves in the operator’s environment by spending time to operate the HMI system firsthand. While this is the least scalable approach, firsthand experience can be a fast way to build domain intuition and tacit knowledge.

4.3 Data Sources

The type of data we explored in the model building process can largely be categorized into shelf, station, and human related data.

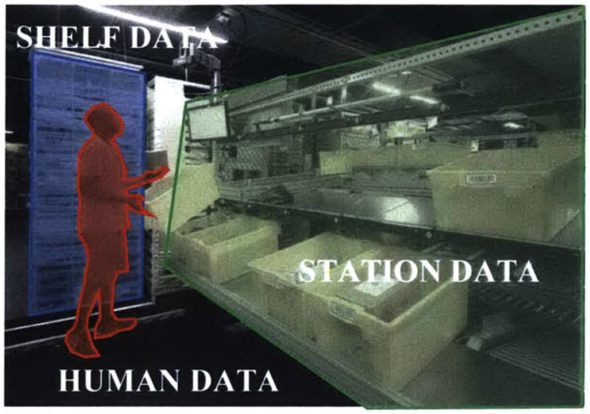


Figure 9: Categorizing Data Sources into Shelf, Human, and Station

4.3.1 Shelf Data

Each process cycle presents a new shelf face that has distinctive qualities. This category of data tries to encompass the variations in shelf, bin, and item characteristics. We hypothesize that these characteristics led to different cognitive and physical load onto operators, subsequently affecting takt time.

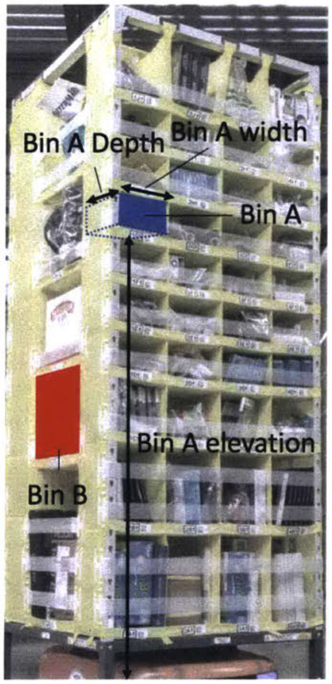


Figure 10: Shelf with example Bin A and its corresponding definitions of width, depth, and elevation. Also shows an example larger Bin B on a different shelf face for contrast.

Data chosen are presented in the following table:

Data Type	**Reasoning(s)**
Physical Dimensions of Target Bin such as width and depth (see Figure 10)	Size of bin may affect the difficulty in isolating and removing items. For example, operators may take longer to find where a smaller target bin (such as Bin A in Figure 10) is located on a shelf face compared to a larger target bin (such as Bin B in Figure 10) [15].
Target Bin Elevation from Ground (see Figure 10)	Height of bin may affect the difficulty in reaching items. For example, a target bin at the highest shelf level could be hard to reach.
Size of Items	Size of items may affect the difficulty in isolating and removing items. For example, a large sized item may take more effort to manipulate than that of a smaller sized item. On the other hand, large size items are more quickly identified.
Fullness of Shelves	The clutter of the bins may affect the difficulty in isolating and removing items. For example, it may take longer to retrieve an item from a highly cluttered bin.
Number of Items Processed per Cycle	The more items that an operator is expected to retrieve per process cycle, the longer the takt time is.
Items Unverifiable	Operators find the physical item, but the item is not recognized by computer system. Encountering these unverifiable items can lead to lost time in item handling.
Items Damaged	Operators find the physical item, but the item is damaged. Encountering damaged items can lead to lost time in item handling.
Items Missing	Operators cannot find the intended physical item. Encountering missing items can cause user confusion.

Revisits	Operators might encounter the same shelf or shelf face multiple times during a short period of time. This might lead to short-term familiarity that affects takt.

Table 2: Shelf data used with explanation

4.3.2 Station Data

The station designs and the utilization of totes can vary. This category accounts for the variations in station characteristics and dynamics.

Data Type	**Reasoning(s)**
Type of Station	There are multiple types of stations. Each type of stations may be more ergonomic or automated than others.
Positions of Totes	Operators have to traverse different distances per process cycle depending on the active number of totes. Usage of the totes furthest away from the shelf leads to a longer distance that an operator must cover.
Items in Complete Tote	Operators are instructed to place items in specific totes. At some point, totes will be considered completed and the system will instruct the operator to move the completed tote downstream. A tote's level of clutter may affect the speed of which an operator can properly place a new item into that same tote.

Table 3: Station data used with explanation

4.3.3 Human Data

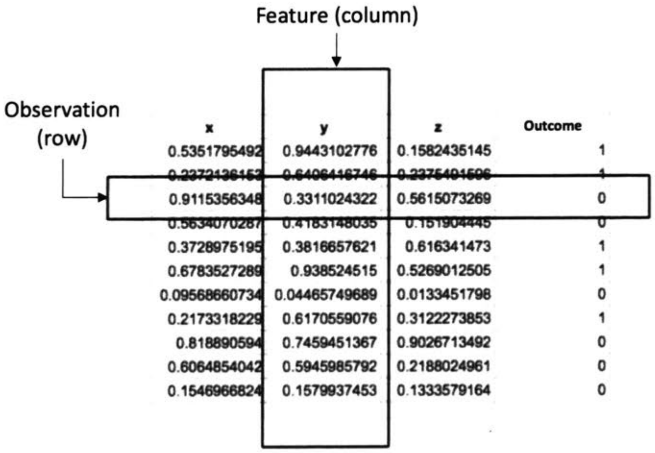
The final category is data around the operators. This category accounts for the variations in experience and work schedules.

Data Type	**Reasoning(s)**
Hour of Day, Day of the Week, Day of the Month	There may be cyclical patterns in human performance.
Number of Operators	People's performance may be affected by size of group.
Individual Takt Time Series	We use the time series to extract information such as historic performance, and duration of rest. For example, a group of operators with lower duration of rest may negatively impact takt.
Day vs Night Shift	There may be cyclical patterns in human performance relative to the type of shift.
Operation Site Name	Each operation sites may be operated by different people with different styles. The ability to tag historic data to their respective location of origin helps account for this variation.

Table 4: Human data used with explanation

4.4 Data Pre-processing and Feature Engineering

Supervised ML algorithms require using a feature dataset in a standardized and consistent tabular format (see Figure 11). Every row (observation) of the feature dataset must be associated with a corresponding outcome and shares the same number and set of features (columns).


Feature (column)
Observation (row)

x	y	z	Outcome
0.5351795492	0.9443102776	0.1582435145	1
0.2372136152	0.6406416746	0.2375401506	1
0.9115356348	0.3311024322	0.5615073269	0
0.5634070287	0.4183148035	0.151904445	0
0.3728975195	0.3816657621	0.616341473	1
0.6783527289	0.938524515	0.5269012505	1
0.09568660734	0.04465749689	0.0133451798	0
0.2173318229	0.6170559076	0.3122273853	1
0.818890594	0.7459451367	0.9026713492	0
0.6064854042	0.5945985792	0.2188024961	0
0.1546966824	0.1579937453	0.1333579164	0

Figure 11: Example of feature dataset amenable to ML

Our sources of raw data mentioned earlier occur with different frequencies. For example, an operations site may experience thousands of items processed in an hour. In that same hour, the type of stations staffed may remain unchanged. In order to build a feature dataset that is amenable to ML, we convert the raw data into hourly features of a group of operators working at each robotics floor.

The goal is to pose the ML problem as follows: given the characteristics of a group of operators at a robotics floor in the last hour, estimate in retrospect what the average takt of the group of operators in that same last hour (see Figure 12).

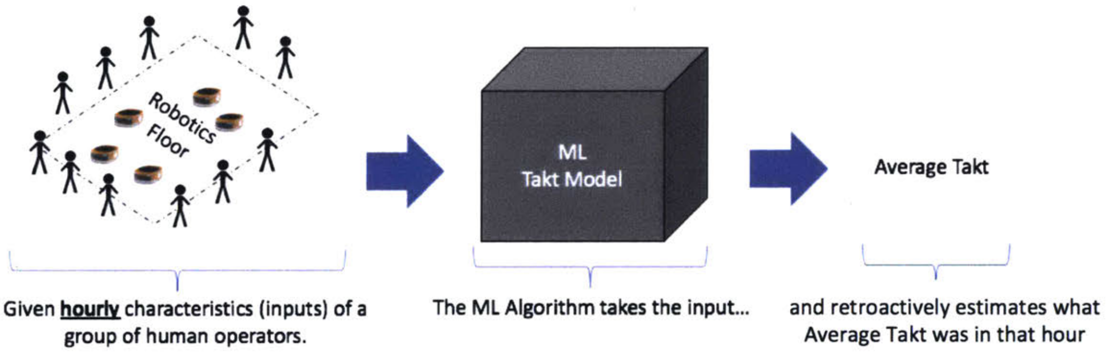

Robotics Floor
ML Takt Model
Average Takt
Given **hourly** characteristics (inputs) of a group of human operators.
The ML Algorithm takes the input...
and retroactively estimates what Average Takt was in that hour

Figure 12: Overview of problem posed for ML

We chose to model takt at the hourly level for the following reasons:

- The model performance at hourly level demonstrated promising accuracy.
- An hourly model enables flexibility because the output can be directly aggregated to produce shift, day, week, or month level outputs.

The feature set will look like the diagram of Figure 13.

	Feature 1 ($X_{m,0}$)	Feature 2 ($X_{m,1}$)	…	Feature n+1 ($X_{m,n}$)	Avg Takt (Y)
01-01-2018, 12am	$X_{0,0}$	$X_{0,1}$	…	$X_{0,n}$	Y_0
01-01-2018, 1am	$X_{1,0}$	$X_{1,1}$	…	$X_{1,n}$	Y_1
…	…	…	…	…	…
mth row	$X_{m,0}$	$X_{m,1}$	…	$X_{m,n}$	Y_m

Figure 13: Design of Final Feature Set

Each row of the feature set is associated with a timestamp indicating the hour block of operation. The features of the row are generated using data available in the past relative to the corresponding timestamp. In terms of time window size used, the features can be categorized by last hour and beyond last hour. In the following sections, we will explain how the features are created.

4.4.1 Continuous and Discrete Data Features (last hour)

The majority of the data are comprised of continuous and discrete values aggregated in the last hour. We convert each hourly set of data into hourly features by simply taking the average for easy interpretability. For example, in any given hour, we may observe multiple discrete bin depths values encountered by a group of operators. We then take the average of all the bin depths values as a feature called "Average Bin Depth".

4.4.2 Nominal Data Features (last hour)

Nominal data is a subset of categorical data where values are represented in discrete labels that have no quantitative value. For example, in any given hour, we may observe a group of operators assigned to either the Day Shift or Night Shift.

The first method is to convert the set of nominal data into a normalized frequency distribution across all possible unique values. For example, each individual operator in a given hour are either part of the day shift or night shift. Because shifts can overlap, a group of operators in a given hour can be a mix of both day and night shift operators. Thus, we can convert a set of shift values into two features called “Percentage from Day Shift” and “Percentage from Night Shift” every hour.

The second method is one hot encoding, which is primarily used to identify which Operational Site a row of data came from. One hot encoding involves converting a single nominal data type into multiple binary variable for each possible value of that nominal data type. For example, suppose we have data from Operational Sites X, Y, and Z. We can then create three binary features called “Is_X”, “Is_Y”, and “Is_Z”. If a row of data is from Operational Site X, then the corresponding feature of “Is_X”, “Is_Y”, and “Is_Z” would be 1, 0, and 0 respectively.

4.4.3 Ordinal Data Features (last hour)

Ordinal data is a subset of categorical data where values are represented in discrete labels and order matters. For example, in any given hour, we may observe a set of Tote Position values where items were ultimately placed after each process cycle. For reference, Tote Positions can take labels of 1, 2, 3, 4, 5, or 6, where the larger the value the further the operator must traverse to move the items. We convert each hourly set of Tote Positions data into integers and create hourly features by simply taking the average.

4.4.4 Time Series Data Features (beyond last hour)

We also created features based on operator work patterns beyond just the last hour. For each hour and robotics floor, we have information regarding which individual operators were present. Each operator has their own individual recent history of takt, which we defined as work pattern. As shown in Figure 14, historic work pattern is a time series signal showing how each individual operator performed across a historic window of time. For each operator, we extract metrics from their unique historic work pattern.

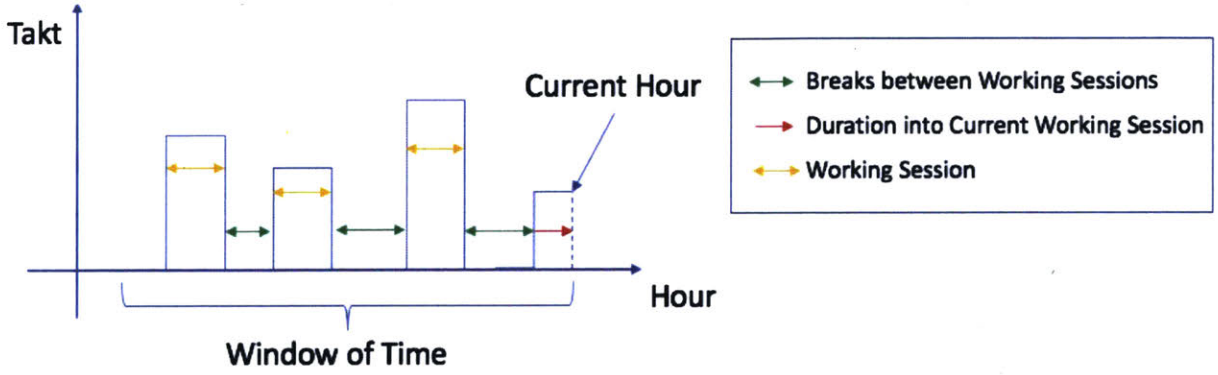

Takt
Current Hour
Hour
Window of Time
Breaks between Working Sessions
Duration into Current Working Session
Working Session

Figure 14: Example time series of an operator's work pattern during some snapshot in time

From each operator's historic work pattern, the following metrics are extracted:

- Working session: the consecutive hours where an operator is actively working prior to the current hour of interest. This is determined by measuring where individual hourly takt is non-zero.
- Breaks between working sessions: the consecutive hours where an operator is actively not working prior to the current hour of interest.
- Current working session: the most recent working session.

Thus, for each operator, we can then derive measurements such as typical breaks between working sessions, duration into current working session, total hours worked over the last week, and mean historic takt.

All these values per individual are then averaged to get hourly features per group.

4.4.5 Feature Cleaning

After creating the initial feature set, we clean the data through the following steps:

1. We remove hours of data with low number of man-hours logged and exceedingly large average takt. We do so by removing rows with man-hours logged in the lower 5th percentile of man-hours, and rows with average takt above 3x the population average takt

mean. The purpose is to avoid exposing our ML algorithms to hours of data with non-standard operations.

2. We impute missing values of a feature using the median value of that feature from the same robotics floor.

4.4.6 Feature Scaling

Lastly, features are normalized prior to algorithm training and testing. We do this by measuring the sample mean and sample standard deviation of just the training data set. Then we subsequently subtract the sample mean and divide by the sample standard deviation for both the training and test set. This scaling process allows the feature set to be compatible with more machine learning algorithms that assume standard normally distributed data.

4.5 Model Tuning and Evaluation

4.5.1 Extreme Gradient Boosting (XGBoost) Algorithm

We trained the ML regression model using gradient tree boosting with XGBoost. XGBoost is an open-source implementation of gradient boosting which originated from research by Chen and Guestrin, designed to be highly efficient, flexible and portable [16].

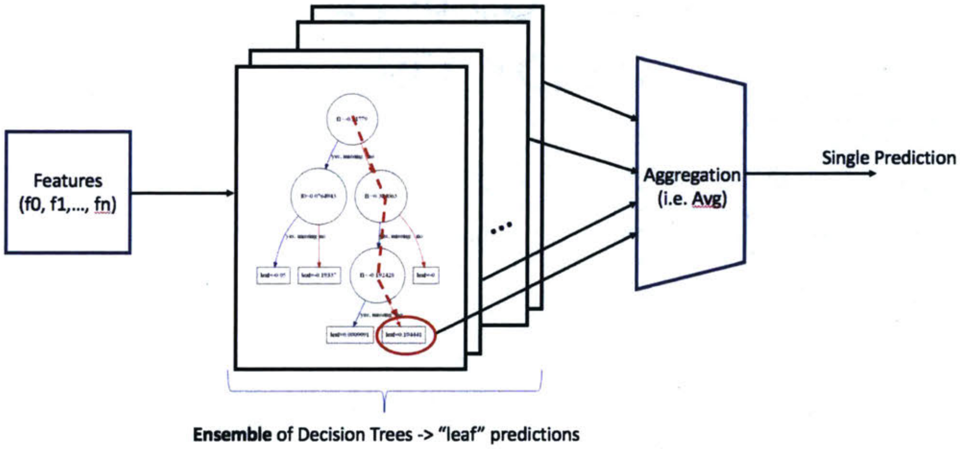


Figure 15: High-level diagram of Decision Tree Ensemble Regression Model

At a high level, tree boosting is an ensemble learning method where hundreds or thousands of weak decision trees are built in parallel using the training data (see Figure 15).

For each decision tree, a different subset of features is used to determine how to traverse down the tree structure. A 'leaf' value is reached at the end of each tree, which is the predicted value. While each individual tree prediction is generally inaccurate, the combination of their outputs tends to produce much more accurate predictions.

The "gradient boosting" aspect refers to the framework by which these ensemble of decision trees are built during the training process. At a conceptual level, gradient boosting involves building trees in stages, whereby each new stage of decision trees built addresses the prediction weaknesses of the previous stages. Fortunately, XGBoost's underlying algorithm is made available through open-source libraries.

4.5.2 Hyperparameters

Hyperparameters are a pre-determined set of ML algorithm parameters whose values are used to control the learning process. The set of values depend on the type of ML algorithm.

The following list outlines the main hyperparameters we tuned on XGBoost.

1. Learning Rate (from 0 to 1): Step size shrinkage value that effectively controls the level of boosting or learning. At each boosting step, the feature weights can be attenuated by the Learning Rate value.
    a. The lower the learning rate the more conservative the boosting process is.
2. Max Tree Depth (positive integer): Maximum depth of any tree built.
    a. The higher the value, the more complex the underlying tree structures, potentially leading to overfitting or diminishing returns.
3. N Estimators (positive integer): Maximum number of trees built.
    a. The higher the value, the more complex the underlying tree structures, potentially leading to overfitting or diminishing returns.

4. Subsample (between 0 and 1): Ratio of training data actually used for training for each tree.
    a. A value of 0.5 means that XGBoost will randomly sample 50% of the training data prior to growing each tree. This added randomness helps combat overfitting.
5. Column Sample by Tree (between 0 and 1): Ratio of columns (features) actually used for training for each tree.
    a. A value of 0.5 means that XGBoost randomly uses only 50% of the features in the training data prior to growing each tree. This added randomness helps to avoid overfitting.

4.5.3 Cross-Validation

We quantify performance using the holdout and K-fold cross-validation. Both techniques involve splitting the feature set into a training set and test set. The training set is use for training algorithm. The test set is used to evaluate the out-of-sample performance of the trained algorithm on unseen data. In this manner, we can better avoid the algorithm simply "memorizing" the data (a condition known as overfitting).

We use the holdout method for quantifying finalized model performance. In this process, a fixed portion of the whole data set is randomly selected for training and testing. We experiment with different proportions and our final model uses 66% of the data for training and the remaining 33% for testing. In addition, we also experiment with how the out-of-sample performance changes for different ratios of data allocated for training versus testing.

We use 3-fold cross validation for model hyper-parameter tuning. In this process, the data set is divided into 3 randomized subsets and the holdout method is performed 3 times. For each holdout, one subset is designated as the test set, and the remaining subsets are for training. More details of the hyper-parameter tuning process will be detailed later.

Our primary metrics of performance is Mean Absolute Percent Error (MAPE).

4.5.4 Algorithm Tuning

The optimal set of hyperparameters is determined empirically by sweeping through different combinations and seeing which results in the best performance.

First, we discretize a limited range of values for each hyperparameter, resulting in a discrete subspace of all possible hyperparameters called a hyperparameter grid. Hyperparameter grids with higher dimensions may increase the chance of discovering a better set of hyperparameters than those of lower dimensions. However, hyperparameter grids with more dimensions also require more computational resources due to a larger space of values that must be explored. Early in the project, we opted for wider ranges in values with larger step-sizes in hopes of locating a combination near a locally optimal solution. Later in the project, we opted for more focused ranges in values with smaller step-sizes. See Table 5 for the final hyperparameter grid used.

Hyperparameter Name	Range	Step-size	Total Steps
Learning Rate	10^{-2} to $10^{-0.8}$ (0.01 to 0.158)	10 evenly spaced on a log scale from -2 to -0.8	10
Max Tree Depth	3 to 8	1	5
N Estimators	700 to 1200	100	5
Subsample	0.8 to 0.96	0.025	7
Column Sample by Tree	0.6 to 1	0.05	8

Table 5: Hyperparameter Grid used for XGBoost tuning with 14000 possible combinations.

Lastly, we search the hyperparameter space using Randomized Grid Search (RGS). Instead of executing an exhaustive search, RGS samples only a fixed number of combinations (without replacement). Each sample is put through the 3-fold cross-validation process. RGS will return the combination that yields the best observed average MAPE.

While, RGS may yield solutions that are less optimal and consistent than those of exhaustive grid search, RGS had the advantage of returning near optimal solutions at a faction of the time. In fact, research by Bergstra et al. claims that random search is more efficient [17].

After arriving at a solution, with finalized hyperparameters and feature set, that sufficiently satisfies the four desirable qualities outlined, we move on to exploring techniques to apply the model.

4.5.5 Baseline for Comparison

In order to understand whether a certain MAPE resulting from an ML algorithm is respectable, we also determine a baseline MAPE value. We define the baseline MAPE as the MAPE achieved when, instead of exposing training data to a ML algorithm, we simply calculate the average takt value from the training data and use that as a constant prediction.

We also measure the performance using K-Nearest Neighbors algorithm (KNN), a simple, non-parametric, and lazy ML approach. It is non-parametric because KNN makes no assumptions of the underlying distribution of the data. It is lazy because it does not use the training data points to do any generalization. Instead when given new observations for a regression, KNN simply just returns a prediction based on averaging the observed values associated with the K nearest training samples, as measured by a distance function in the feature space, to the new observations. We set K to be 5 and use Euclidean distance function.

4.6 Feature Selection Methodology

During this iterative process, we use domain expertise and descriptive metrics to decide which features to add, modify, or remove. The two main descriptive metrics are correlation heatmap and feature importance of the trained ML model.

4.6.1 Pairwise Correlation

Building a heatmap can be an effective and simple visualization tool to discover and quantify the degree to which variables in the feature set are dependent upon each other. For every combination of two features, we calculate Pearson's R value as well as R-squared.

When we see R or R-squared significantly deviate from 0, we can hone in on the corresponding pair of features and use domain expertise and human judgement to decide on whether to modify the feature set.

4.6.2 Feature Importance

The structure of the trained ML model can also give insight into selecting features. For gradient tree boosting models like XGBoost, we mainly referred to feature importance scoring using the normalized weight metric.

The weight metric is calculated as follows. For every decision tree in the ensemble, we count how many times a particular feature is used for splitting. We then repeat this for every feature so that each one has their own corresponding weight value. Finally, we get the normalized weight metric by dividing each weight by the sum of all weights.

The advantage of the normalized weight metric is that it gives insight into which features seem to be important. The higher the weight, the more that the feature is embedded in the structures of the decision trees and used in the prediction generation process. This can be a helpful tool in conjunction with domain expertise in the feature selection process.

5 Model Insights

The business viability of implementing ML solutions ultimately relies on the insights derived from the model. Treating the ML model as a black-box, we propose the following two approaches to leveraging the model. Both have their own pros and cons, and targeted towards different use-cases.

5.1 Empirical Trade-off Analysis

Our goal in this approach is to quantify global trends in how different features affect the overall takt. We focus primarily on analyzing a stratified subset of the model's features which can be practically adjusted in the physical world either from system design or operational manner.

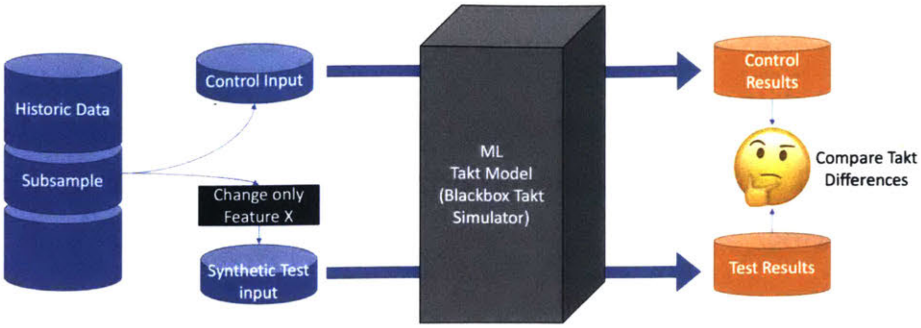


Figure 16: A/B testing approach to quantify takt impact due to feature X.

We can then pose many questions to the model and quantify how different features of interest contribute to takt by running many controlled A/B tests as shown on Figure 16. In the first step of each A/B test, a targeted subset of historic features is sampled and fed through the ML model, resulting in a set of takt estimations (which we refer to as the "Control Results").

Then, with the same original input subset, the values of a single feature (column) is randomly increased or decreased to create a slightly modified, synthetic input feature set. This synthetic input feature set is then fed through the same ML model to generate a set of takt estimations (which we refer to as the "Test Results").

Finally, we compare the Control Results and the Test Results. To quantify how takt is affected, we use a unitless measure called Elasticity, a concept inspired from the field of economics.

$$Elasticity\ of\ Feature\ X\ on\ Takt = \epsilon_X = \frac{\frac{\Delta Takt}{Takt}}{\frac{\Delta X}{X}} = \frac{\%\ Change\ in\ Takt}{\%\ Change\ in\ Feature\ X}$$

The benefit of this approach is that the ML model becomes a one-stop shop for providing analysis that can control for many factors at once and simulate takt outcomes under any set of starting assumptions. The macro-level trends discovered can be used to analyze system-level trade-offs in operations or design of the robotics system.

The limitation of this approach is that, from a diagnostic perspective, there is no direct way to isolate the exact contribution of each feature with any given prediction. Our next approach aims to address this issue.

5.2 Shapley Additive Explanation (SHAP) Values

Another method of model interpretation is by calculating SHAP (SHapley Additive exPlanation) values, which allows for interpretation of the factors at the level of individual observations. The SHAP values technique was proposed by Scott M. Lundberg and Dr. Su-In Lee [18].

SHAP values were inspired by Shapley values from game theory. Developed by Lloyd Shapley in 1953, the Shapley value is a solution to divide the rewards amongst each player in a collaborative game in a manner that fairly reflects each individual's contribution [19]. Similarly, SHAP value measures how much each feature in our ML models additively contributes to the overall predicted value.

Simply put, the SHAP algorithm, for any single observation (single row of features), calculates the contribution (SHAP value) of each feature by comparing what the ML model predicts with and without it. In the process of traversing the ensemble of decisions trees, any node reached that splits using a feature being actively withheld will simply return the average of all leaf values downstream of those nodes. Since the order in which an ensemble of decision trees withholds features can affect the overall prediction, the SHAP algorithm explores every possible order and returns the average contributions.

SHAP values have the following properties.

1. Efficiency: The sum of each feature's SHAP value and base value should equal the ML model's predicted value. The base value is a model's output if no features are used (essentially just the mean sample takt time).
2. Symmetry: Features that contribute the same value to the overall prediction should have the same SHAP value.

3. Null player: If a feature had no effect on the predicted value, the corresponding SHAP value should be 0.
4. Linearity: The contribution of any two features should equal the sum of each individual's SHAP value.

SHAP values with XGBoost have the following advantages.

- Unlike traditional feature which give feature importance metrics across the whole sample population, SHAP values offer sample-by-sample feature impact.
- SHAP values are consistent and interpretable.
- Due to the hierarchical nature of decision trees, the open-source SHAP library has optimizations that allow for SHAP values to be practically calculated without actually empirically executing every possible feature combination.

SHAP values with XGBoost have the following disadvantages.

- As with traditional feature importance metrics, SHAP values are sensitive to high correlations between features. Fortunately, this issue is accounted for in our project (see 3 Desired Model Qualities).
- SHAP values represent an additive approximation of the ML model at a specific sample. The SHAP values alone cannot estimate the impact of any intervention done on the system.

Thus, we leveraged SHAP values in our research in diagnostic use-cases. Namely, we want to prototype a diagnostic tool using SHAP values to help operations quickly identify potential root-causes of takt performance deviations.

6 Results and Discussion

6.1 Model Performance

6.1.1 Feature Set

We were provided with industry data that represents typical material handling metrics found within warehousing. The data was sanitized to eliminate any proprietary information and shared for educational purposes only. As an overview, the dataset is comprised of historic data from 81 different robotics floors across 18 operation sites around the globe. Our final feature set comprises of over 250,000 rows (hours) and 43 features. See Table 6 for details.

Data Source	Feature/Input Name	Details
Shelf	Bin Depth	Average Depth of bins that operators processed.
	Bin Elevation	Average height level of bins that operators processed.
	Inventory Item Size	Average size of items in inventory of the whole floor.
	Fullness of Shelves	Average percentage of available storage volume occupied by inventory of the whole floor. Equal to total inventory volume divided by the total available storage volume of all the shelves combined.
	Items Damaged per Cycle	Average # of items encountered that were damaged per cycle.
	Items per Cycle	Average # of items processed per cycle.
	Items Missing per Cycle	Average # of items encountered that were missing per cycle.
	Items Unverifiable per Cycle	Average # of items encountered that were unverifiable per cycle.
	Revisits	Average number of revisits per shelf.

Human	Hour of the Day	Hour of the day in 24-hour format
	Day of the Week	Integer value mapping to a day of the week
	Day of the Month	Day of the month
	Number of Operators	Total number of operators working
	Historic Performance	Individually calculate the median takt of last 2 months per person in the group, then create a single group average. We believe that the median takt across the last two months was an effective proxy to quantify operator skill level.
	Breaks Between Sessions	Individually calculate the average hours of break between sessions per person in the group over the last week, then create a single group average.
	Total Weekly Worked Hours	Individually calculate the total hours spent in shift per person over last 7 days in the group, then create a single group average.
	Duration into Current Session	Individually calculate the total hours spent in current shift per person in the group, then create a single group average.
	Operational Site Name	18 one-hot encode features, one for each of the 18 operational sites studied.
Station	% People from Day Shift	Percentage breakdown of operators assigned to night versus day shift. Max value of 100%
	% Station of Type X (covers 5 types)	Multiple features representing percentage breakdown of supported station types. All sum to 100%.
	Positions of Totes	Average position of totes where items were placed.
	Items in Complete Tote	Average number of items in completed tote.

Table 6: Detailed table of the 43 features used in ML Model

In support of our desirable model qualities, the final feature set also demonstrates low correlation amongst each other. R-squared, which is equal to the square of the correlation coefficient, is the metric we use for evaluating feature pair-wise association. R-squared is a unit-less metric that generally spans between 0 to 1 and represents the ratio of the variance explained, by a simple linear regression, to the total variance. The higher the pair-wise R-squared value, the higher two variables are linked. From Figure 17, we can see that the majority of features have R-squared equal to or less than 10%.

	Hour of Day	Day of Week	Day of Month	Bin Depth	Bin Elevation	Inventory Item Size	Fullness of Shelf	Breaks Between Session	Historic Performance	Weekly Worked Hous	Duration into Current Sessio	Num of Operator	Day Shift %	Ops Site 1	Ops Site 2	Ops Site 3	Ops Site 4	Ops Site 5	Ops Site 6	Ops Site 7	Ops Site 8	Ops Site 9	Ops Site 10	Ops Site 11	Ops Site 12	Ops Site 13	Ops Site 14	Ops Site 15	Ops Site 16	Ops Site 17	Ops Site 18	% Station Type 1	% Station Type 2	% Station Type 3	% Station Type 4	% Station Type 5	Position of Tote	Items Damaged per Cycle	Items per Cycle	Items Missing per Cycle	Items in Complete Tote	Items Unverifiable per Cycl	Shelve Revisits
Hour of Day	1	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Day of Week	0	1	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Day of Month	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bin Depth	0	0	0	1	0.1	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bin Elevation	0	0	0	0.1	1	0.3	0.1	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0.1	0	0	0	0
Inventory Item Size	0	0	0	0	0.3	1	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0.1	0	0.1	0	0	0	0.1	0.1	0.1	0	0.1	0	0	0	0	0	0	0	0.1	0.1	0	0	0
Fullness of Shelf	0	0	0	0.1	0.1	0	1	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0.1	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
Breaks Between Session	0	0	0	0	0	0	0	1	0	0.4	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Historic Performance	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Weekly Worked Hous	0	0	0	0	0	0	0	0.4	0	1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Duration into Current Session	0.1	0	0	0	0	0	0	0	0	0.1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Num of Operator	0	0.1	0	0	0	0	0	0.1	0	0	0	1	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Day Shift %	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0
Ops Site 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 3	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 5	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0	0	0	0	0	0	0
Ops Site 7	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0	0	0	0	0	0
Ops Site 8	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 10	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0
Ops Site 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 12	0	0	0	0	0.1	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 13	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0
Ops Site 14	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 15	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ops Site 16	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0
Ops Site 17	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.1
Ops Site 18	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
% Station Type 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	1	0.9	0	0	0	0	0	0	0	0	0	0
% Station Type 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0.9	1	0.1	0	0	0	0	0	0	0	0	0
% Station Type 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	1	0	0	0	0	0	0	0	0	0
% Station Type 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
% Station Type 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Position of Tote	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0.1	0	0	0.1	0	0	0	0	0	0.1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Items Damaged per Cycle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Items per Cycle	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.2
Items Missing per Cycle	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Items in Complete Tote	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Items Unverifiable per Cycle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Shelve Revisits	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0.2	0	0	0	1

Figure 17: Pairwise R^2 Heatmap of Features

The two minor exceptions are 30% for “Inventory Item Size” with “Bin Elevation” and 40% for “Breaks Between Session” and “Weekly Hours Worked”. Because Pod Faces often have the largest bins at low elevations, it is not surprising that Inventory Item Size shows some correlation with Bin Elevation. Similarly, there are a multitude of fixed work schedules which may account for why Breaks Between Session and Weekly Hours Worked are slightly correlated. That being said, we determined that these two exceptions still exhibited low enough correlation to be included.

The single big exception is 90% for “% Station Type 1” and “% Station Type 2”. Because the majority of stations are either of Type 1 or Type 2, the sum of “% Station Type 1” and “% Station Type 2” will typically come close to 100%, leading to a high correlation. We decided to leave them within the model because we were not planning to isolate individual contributions at the station level.

6.1.2 Baseline Performance for Reference

To put into context whether the MAPE performance resulting from our ML algorithm is adequate, we first calculated summary statistics of the baseline MAPE value. As a reminder, we define the baseline MAPE as the MAPE achieved when, instead of exposing training data to a ML algorithm, we simply calculate the average takt value from the training data and use that as a constant prediction. In other words, the baseline MAPE is the performance of a very simple algorithm.

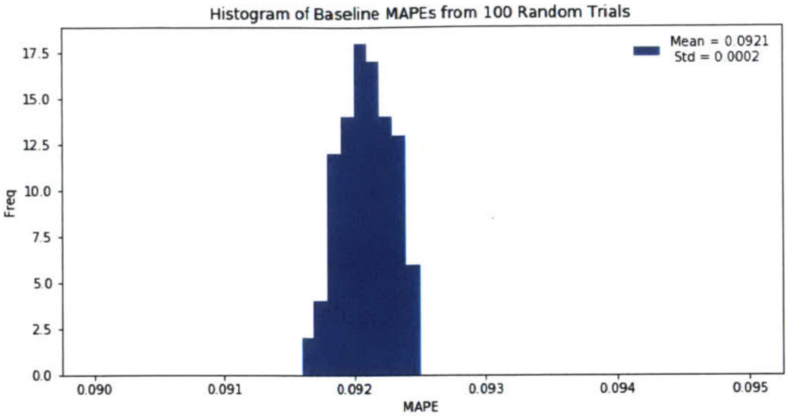


Figure 18: Histogram of Baseline MAPE after 100 trials

We ran 100 trials using the random holdout cross-validation method, with 66% of the set randomly chosen for getting the sample takt average and the remaining 33% for test. We can see from Figure 18 that the baseline performance is around 9.21% MAPE.

We also tried using a k-nearest neighbors algorithm approach, again with 66% of the data for training and the remaining 33% for testing. Setting k to 5 nearest neighbors, we observed performance of 5.84% MAPE.

6.1.3 Hyperparameters Chosen

The final hyperparameter values for XGBoost were determined using RGS of 5000 combinations (out of the possible 14000). The resulting hyperparameters are shown in Table 7.

Hyperparameter Name	**Value**
Learning Rate	0.034
Max Tree Depth	7
N Estimators	1100
Subsample	0.85
Column Sample by Tree	0.9

Table 7: Final hyperparameters used by XGBoost Model

6.1.4 Model Out-of-Sample Performance

We use the holdout cross-validation method for quantifying finalized model performance, with 66% of the data set randomly chosen for training and the remaining 33% for test. The final model's out-of-sample performance is shown in Figure 19, with axis labels removed for confidentiality.

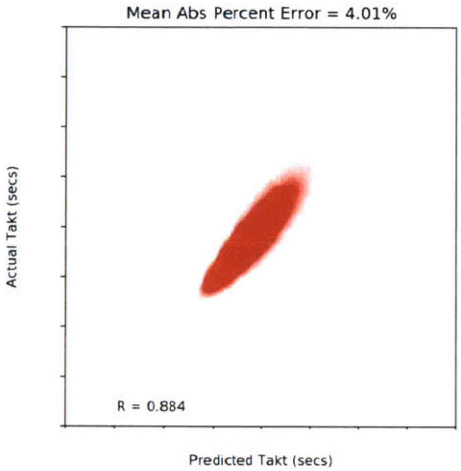


Figure 19: Out-of-sample Predicted Takt against Actual Takt.

Our final model achieves around 4.01% MAPE. For comparison, the baseline MAPE is determined to be 9.2% and our KNN approach achieved 5.84% MAPE. When we use a scatterplot of predicted versus actual takt, we calculated a correlation coefficient and R-squared of 88.4% and 78.14% respectively.

In Figure 20, Figure 21, and Figure 22, we show how the final model's estimated takt compare to those of the actual takt from a time series perspective across different robotics floors. To create these plots, we isolate all the data from a specific robotics floor as the test data set, and the remaining data as the training data. We re-train a new model using the training data and then we evaluate the model using the testing data set to see how well the model performs on a previously unseen robotics floor. This process is repeated for each different robotics floor of interest.

These plots give a different perspective of the model's accuracy as well as the variable nature of operator takt across time per robotics floor. Both the predicted and actual takt go through a 24-hour rolling mean to smooth the time series for easier viewing. For confidentiality reasons, we only present takt normalized by the total sample average takt.

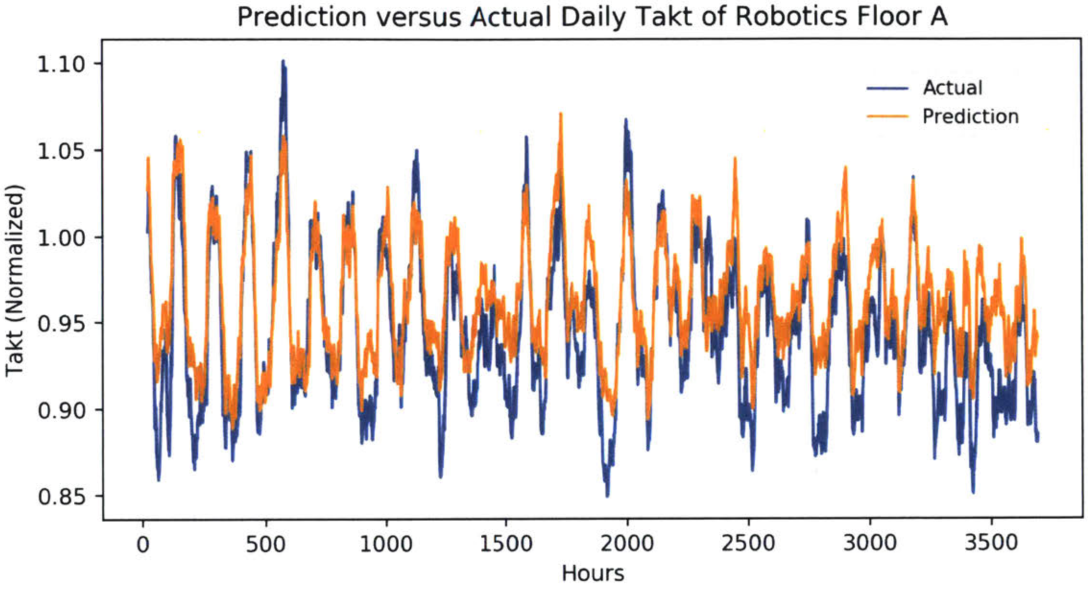


Figure 20: Final model predicted vs actual takt after 24-hour rolling mean

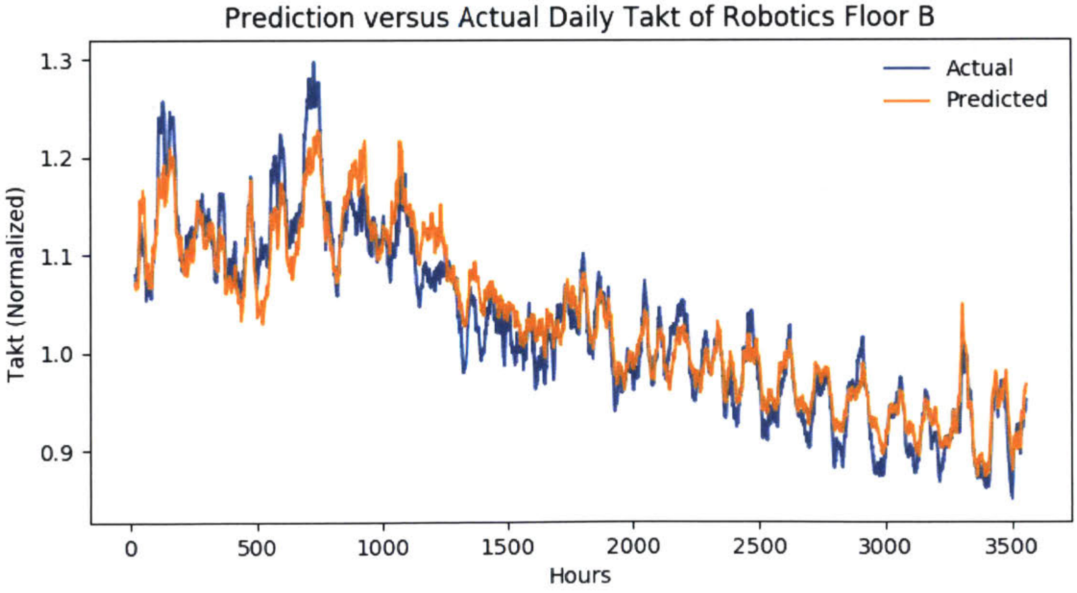


Figure 21: Final model predicted vs actual takt after 24-hour rolling mean

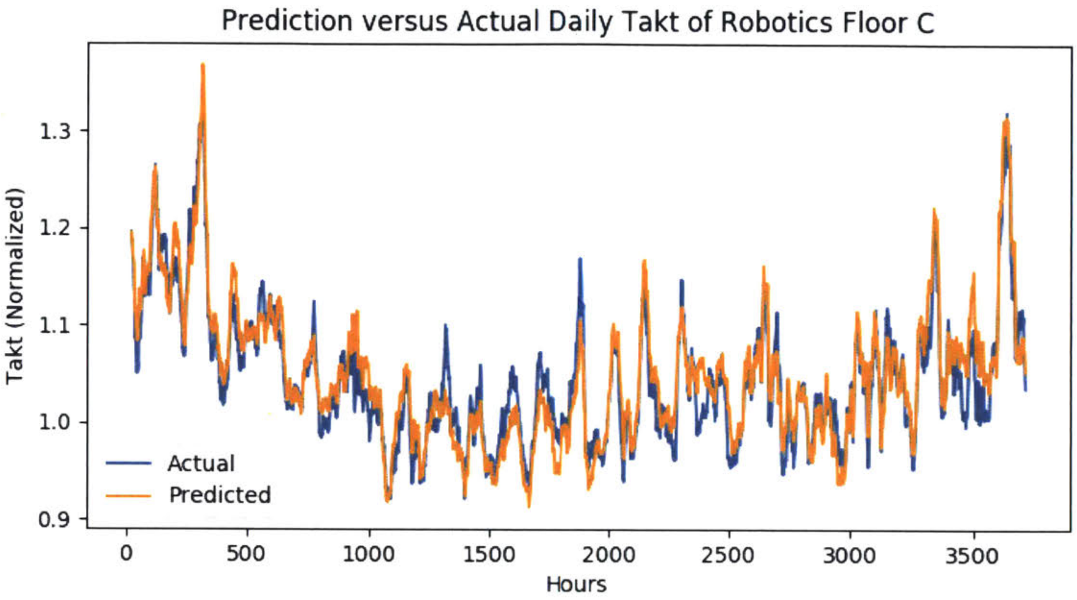


Figure 22: Final model predicted vs actual takt after 24-hour rolling mean

From a qualitative perspective, we can see from Figure 20, Figure 21, and Figure 22 that our model shows a credible ability to model previously unseen robotics floors.

In Figure 20, notice how the model's predictions are able to reflect the same large daily fluctuations from approximately Hour 0 to Hour 1000 on robotics floor A. In Figure 21, notice how the model's predictions are able to reflect the same downward trend of takt from approximately Hour 1000 to Hour 3500 on robotics floor B.

6.1.5 Feature Importance from Model

With the trained model, we also calculated the feature importance scoring using the normalized weight metric, shown in Figure 23.

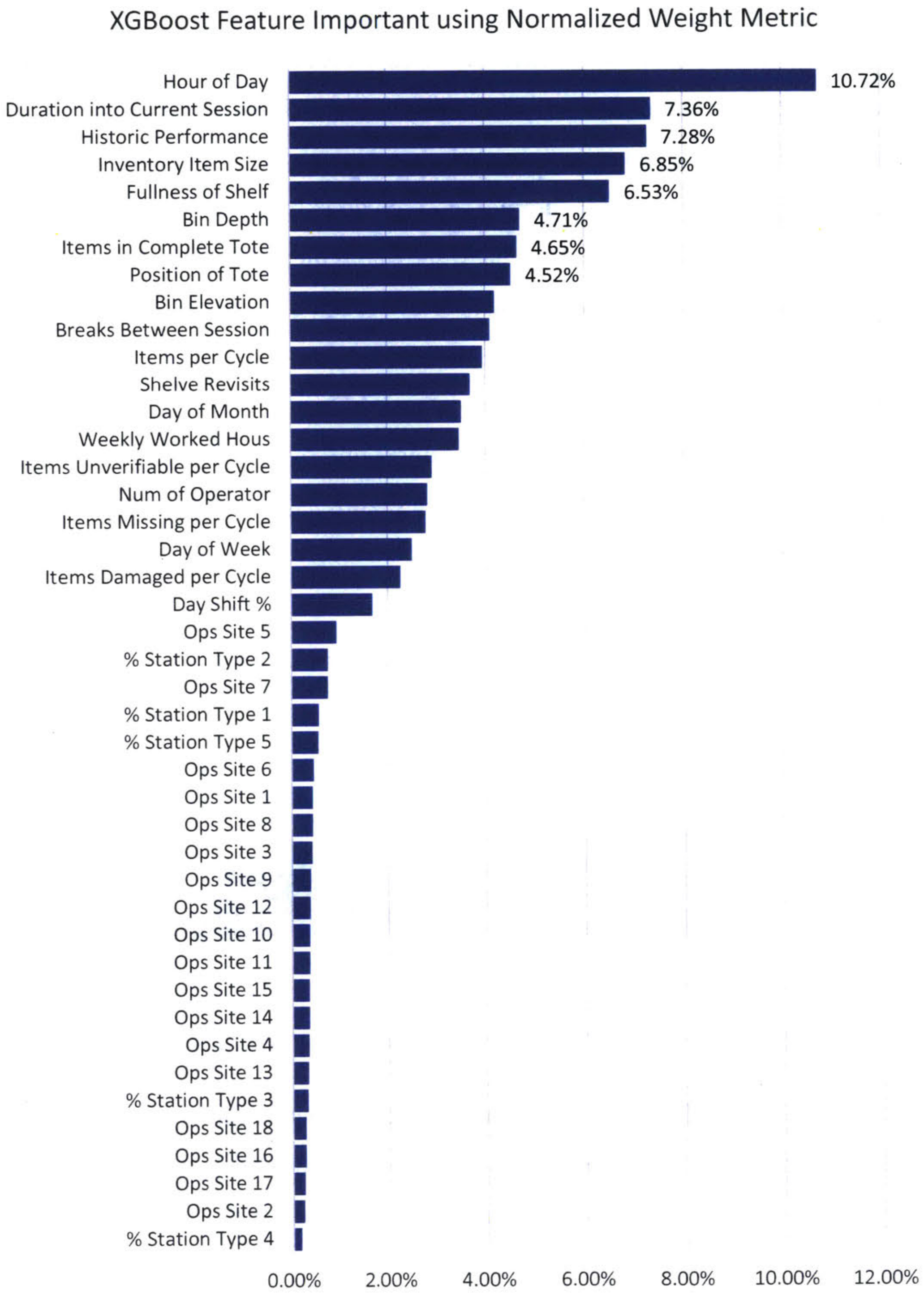


Figure 23: Feature Importance from XGBoost in Descending Order

Feature importance helps give insight into which features are used the most within the XGBoost tree structure.

The feature ranked with the highest importance by a noticeable margin is "Hour of the Day" with 10.72%, implying that the decision trees often rely upon the local hour of the day in making its prediction. We suspect that human performance in this cognitively intensive HMI task may exhibit regular and predictable changes across each day. This aligns with studies on circadian rhythms in cognitive performance. For example, research done by Centre for Chronobiology in Psychiatric University Clinics in Switzerland have shown predicable diurnal variations in highly directed tasks such as the speed of dealing cards, sorting cards, mirror drawing, multiplication, and code transcription [20].

Instead of addressing the importance of the other features one-by-one, we can also view the feature importance by grouping them by category (see Figure 24). For reference, we categorize the types of features used as shelf, station, or human related data.

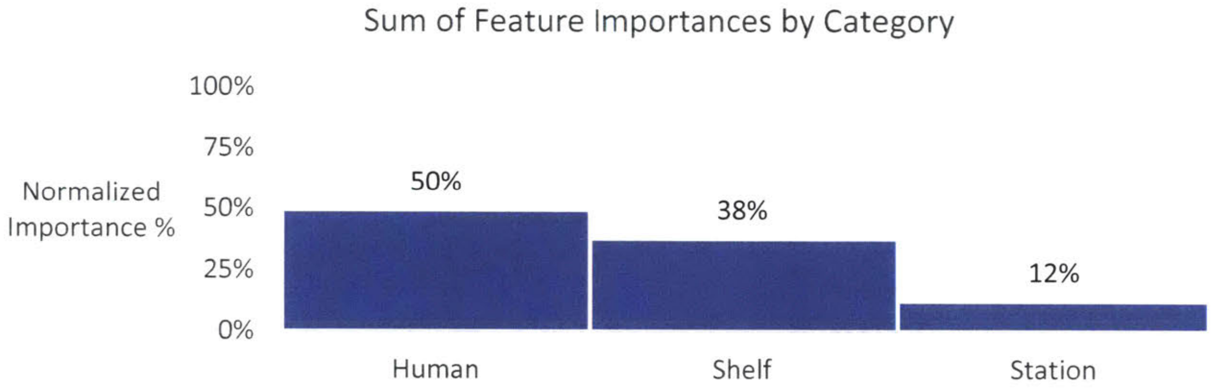


Figure 24: Normalized Feature Importance by Category

We can see that Human related factors leads with around 50% aggregate importance. This is a strong signal that factors such as time of the day, design of the work schedule, and skill development play important roles in takt outcomes. For example, we see that "Duration into Current Session", "Historic Performance" alone account for 7.36% and 7.28% respectively. Similar to Hour of the Day, Duration into Current Session also hints that human performance exhibit regular and predictable changes at the session level as well. The importance of Historic

Performance shows that past performance of operators is a strong indicator for future performance.

In second place are the shelf related factors, which account for a total of 38% of importance. The shelf category covers features related to physical characteristics of the bins and items that were presented to the operators. For example, we see that "Inventory Item Size" and "Fullness of Shelves" alone account for 6.85% and 6.54% respectively. Inventory Item Size and Fullness of Shelves have long been suspected by SMEs to directly impact process difficulty. The fact that they both have high feature importance confirms their significance in impacting takt.

Last are station related factors, which account for a total of 12% of importance. The biggest contributors in this category are "Items in Complete Tote" and "Position of Tote" which account for 4.65% and 4.53% respectively.

6.1.6 Size of Training Data and Model Performance

Holding the final hyperparameters constant, we experimented with how varying amounts of training data impacts the out-of-sample performance (see Figure 25 and Figure 26).

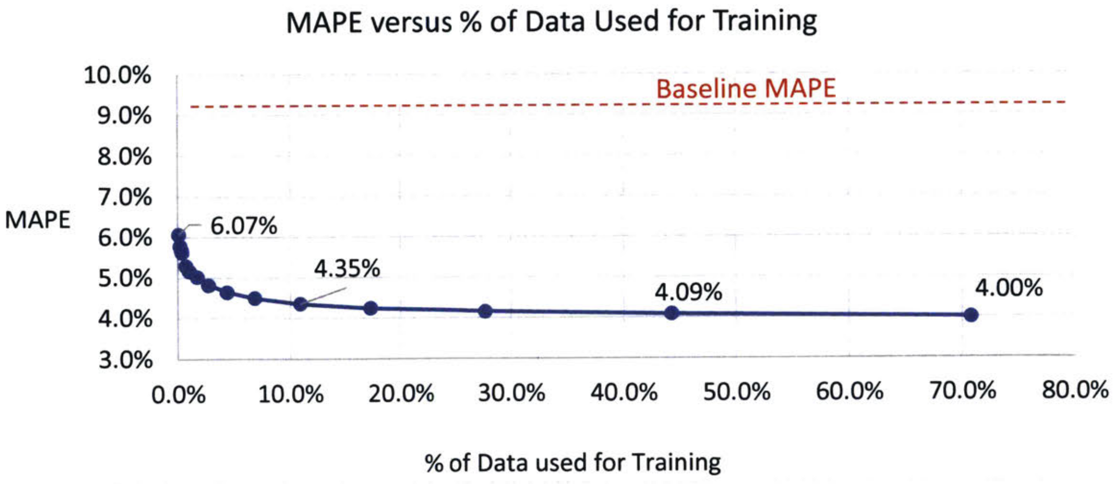


Figure 25: MAPE versus size of training data

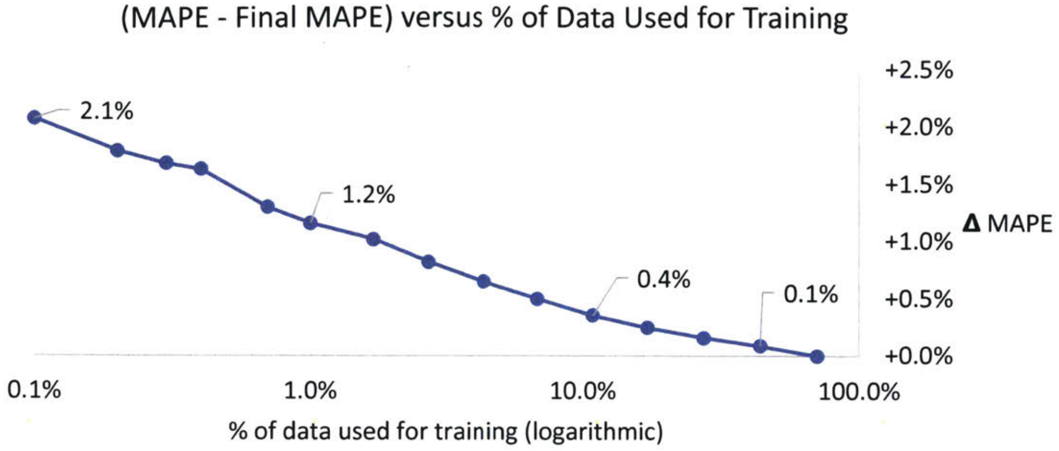


Figure 26: MAPE relative to final MAPE versus size of training Data

From Figure 25, we see that the model demonstrates diminishing returns in MAPE with increasing percentage of data used for training. From Figure 26, we can confirm that MAPE improves roughly at a logarithmic rate with % of data used. For example, we observed that MAPE reaches less than 0.5% MAPE away from our final MAPE with around 10% of the total dataset used for training. Increasing the training data 4-fold from around 10% to 40% only improved MAPE from 0.4% to 0.1% away from our final MAPE.

The characterization of this trade-off is useful when considering cost of implementation. Processing and storing larger amounts of historic data required for model training can result in higher data infrastructure costs. For example, popular cloud service providers such as Amazon Web Services, Microsoft Azure, and Google Cloud all charge customers monthly fees that are proportional to the size of data being transferred, stored, and processed [21]. If the target application of the model required only 5% MAPE (as opposed to the topline performance of 4% MAPE), the required amount of training data required would be around 1.7% as opposed to 70%, which is over 40 times less.

6.1.7 Desired Model Qualities Discussion

Prior to model development, we outlined four desired model qualities. In Table 8, we revisit each quality and argue why we believe our model sufficiently addresses it.

Desired Quality	Reasoning
Accuracy (output)	Our final model achieves 4.01% MAPE, compared to the baseline MAPE of 9.2% and KNN's MAPE of 5.84%.
Human Interpretability (inputs)	Our model uses 43 features. Excluding the 18 binary features, the remaining 26 features are common operations metrics.
Independence (inputs)	From a possible range of 0% to 100%, the vast majority of features used showed pairwise R-squared values from 0 to 10%.
Direct Causality to Takt (inputs)	Each feature used were reviewed with subject matter experts, naturalistic user observations, and firsthand experience.

Table 8: Outline of our progress in meeting the four desired qualities of the model.

6.2 Results of Empirical Trade-off Analysis

In the following figures, we present a select few trends observed by using the Elasticity values extracted from the Empirical Trade-off approach of model interrogation. These trends were chosen because they exhibited higher Elasticity values. For confidentiality reasons, we only present takt normalized by the total sample average takt.

6.2.1 Fullness of Shelves (FoS)

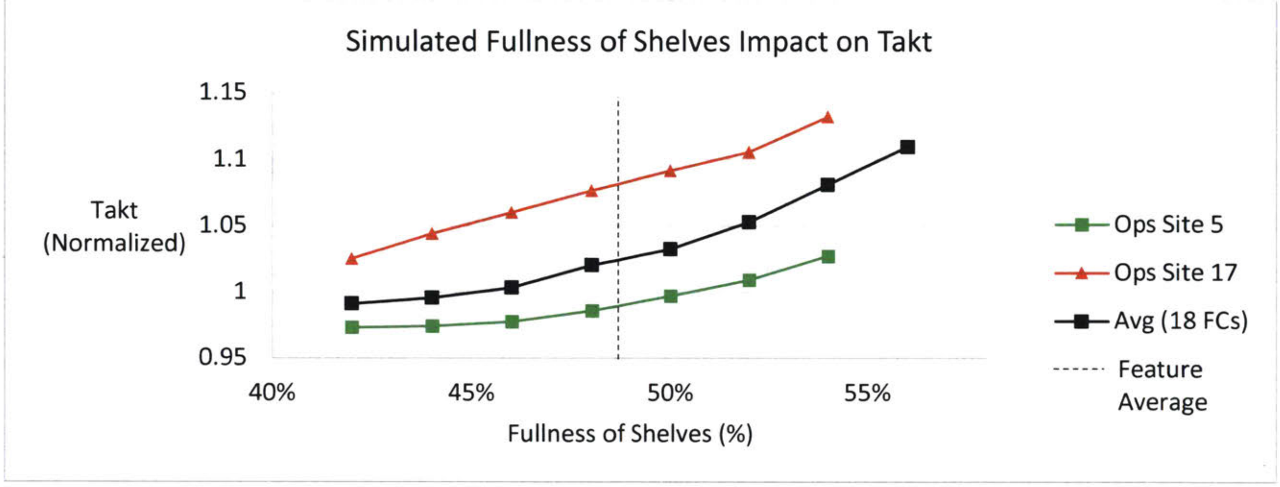


Figure 27: Fullness of Shelves Impact on Takt

The impact of Fullness of Shelves (FoS) on takt is shown in Figure 27. The analysis validates our intuition about FoS relationship, namely that an increase in FoS leads to an increase in takt. Also, trend is slightly convex with takt increasing at an increasing manner at higher FoS values. We can also investigate the model only at specific operational sites (see Site 17 and Site 5 in Figure 27).

This analysis could provide valuable insight for operations when setting optimal inventory levels. For example, businesses during peak times of the year may be tempted to drastically increase inventory levels in order to support wide product mixes and avoid stock-outs. This increase in inventory across a fixed number of shelves will lead to an increase in FoS. However, as shown in Figure 27, we see that higher FoS is associated with higher takt times, limiting the business's ability to quickly fulfill orders. This analysis is another tool by which management can better quantify this trade-off and make more informed decisions on inventory.

6.2.2 Weekly Worked Hours (WWH)

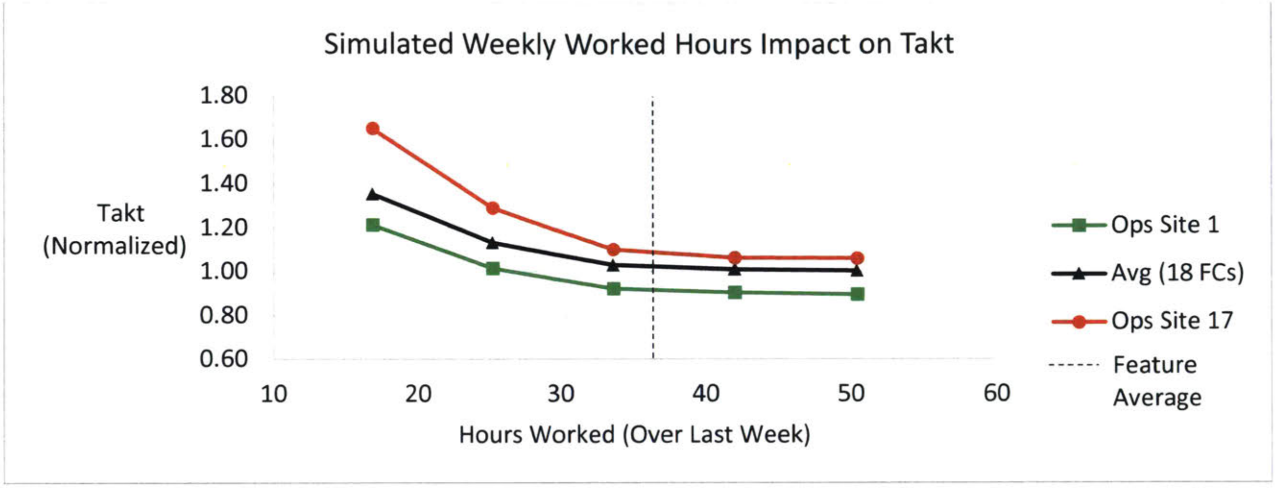


Figure 28: Weekly Worked Hours Impact on Takt

The impact of Weekly Worked Hours (WWH) on takt is shown on Figure 28. The model shows that the WWH starts to negatively impact takt when WWH goes below around 30 hours. We see this same effect when diving deeper into each individual operational site. For example, both Site 17 and Site 1 incur heavy takt increases if the operators have WWH below 30 hours.

One hypothesis is that the associated negative effect on productivity with less WWH is due to less opportunity for skill development or less worker motivation from underemployment. In general, humans experience a learning curve where the more a person performs a specific task, the more proficient they become at it. There may be a learning curve effect for this specific task at the weekly level where operators working over 30 hours per week accumulate enough experience to reach optimal performance. Also, there may be a psychological aspect as well. Previous studies have shown that underemployed individuals do not work as hard because they find their jobs demotivating, causing performance to suffer [22].

If proven true, this analysis can be useful when business design their workforce. For example, businesses might be tempted to increase the number of part-time workers for reduced cost of labor or flexibility in scheduling. However, a higher composition of part-time workers may lead to reduced WWH, which is associated with a negative impact on takt. We also recognize that there may be other confounding factors, and believe that future investigation into WWH is needed.

6.2.3 Items per Cycle (IpC)

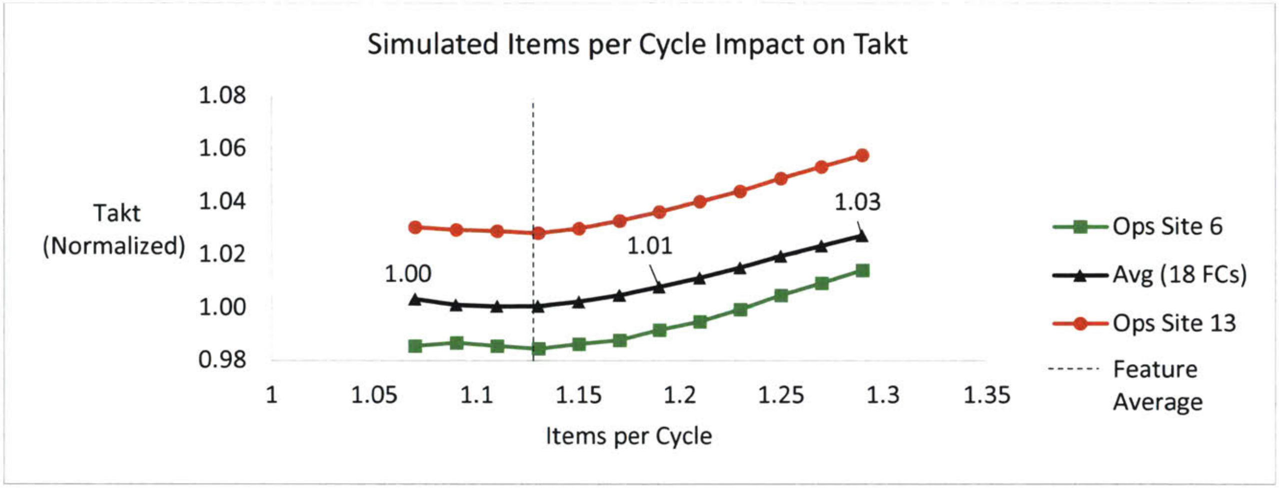


Figure 29: Items per Cycle Impact on Takt

The impact of Items per Cycle (IpC) on takt is shown on Figure 29. We see that IpC values between around 1.07 to 1.15 shows little change in takt. Once IpC exceeds 1.15, we see that higher IpC eventually has a negative impact on takt. This makes sense because in general the more items that an operator is expected to find at a time, the longer it takes to process. This is particularly relevant because of IpC's role in UPH (see Equation 4).

$$\text{UPH} = 3600 \times \frac{\text{IpC}}{\text{Takt}} \times \%_{\text{operator utilization}} \tag{4}$$

UPH is a function of both IpC in the numerator and denominator. When IpC increases, Takt will also increase, slightly hampering the overall impact on UPH. Our results show that increasing IpC still has a net positive impact on overall UPH across the observed range of IpC in our data.

This analysis could provide valuable insight for future design changes of this HMI system. For example, engineers may be able to alter the system in a manner that increases IpC in hopes of a proportional increase in UPH. Using Figure 29, they can then better estimate overall UPH improvement by accounting for the corresponding increase in takt.

6.2.4 Duration into Current (DCS)

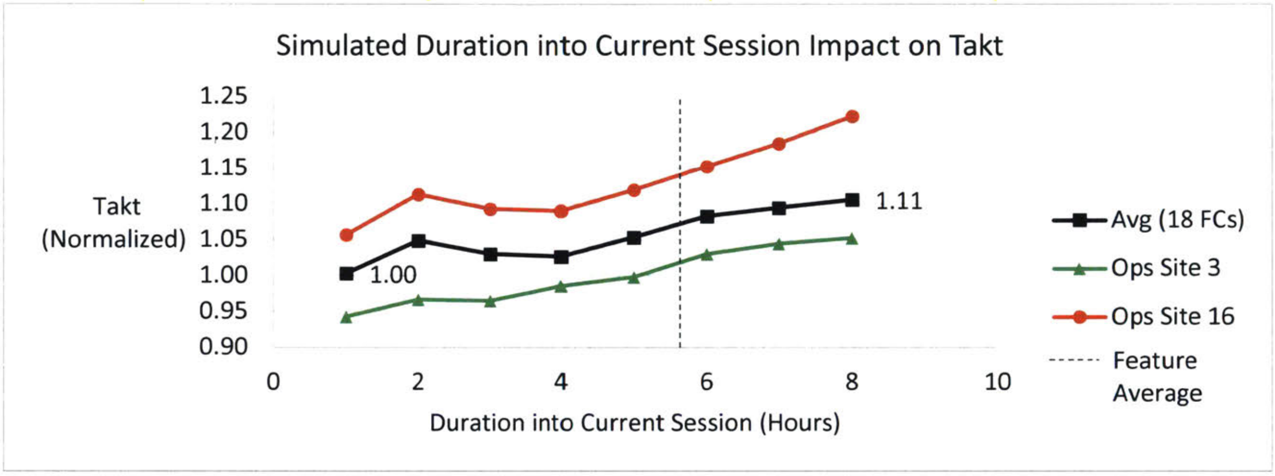


Figure 30: Duration into Current Session Impact on Takt

The impact of Duration into Current Session (DCS) on takt is shown in Figure 30. Between hours 1 and 4, we see that on average there is a slight hump in the 2nd hour. We currently do not have a good understanding why this occurs in some operational sites (see Ops Site 3) and not in others (see Ops Site 16). Perhaps there are slight adjustments or learnings that operators experience during the early, first few hours of every new session.

After the 4th hour, the model shows a clear and consistent trend where higher DCS of the operators negatively impact takt. We hypothesize that this effect is a result of workers getting tired and failing to maintain the same level of focus.

This analysis could provide valuable insight for operations when designing shift and break schedules. For example, businesses might be tempted to increase durations of work without a break to avoid fixed setup costs. However, the resulting higher DCS may lead to negative impacts on takt deep into a shift.

6.2.5 Position of Tote (PoT)

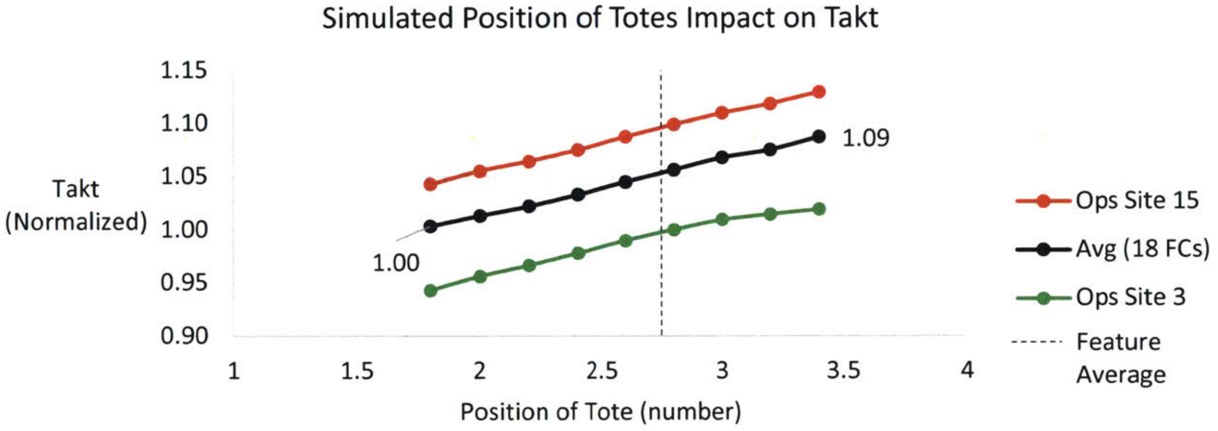


Figure 31: Position of Totes Impact on Takt

The impact of Position of Tote (PoT) on takt is shown on Figure 31. The model shows that the higher PoT the higher the resulting takt becomes. We see this same effect when diving deeper into each individual operational site. For example, both Site 15 and Site 3 incur takt costs as the PoT increases. This trend confirms our intuition. Operators processing items with target tote positions at a higher average number will have to transport items at further distances on average.

This analysis could provide valuable insight for operations when deciding station staffing. For example, suppose that operations managers notice that both takt is relatively high. One option to remedy this is to staff more unused stations and spread out the demand for totes across more stations, reducing the overall average position of totes and takt times. The downside is that this option also means higher labor costs. Managers can use this analysis to better weigh the benefits of staffing more unused stations against the added increase in labor costs.

6.3 Diagnostic Tool using SHAP Values

We also develop a proof-of-concept retrospective diagnostic tool using SHAP values. As a reminder, SHAP values measure the contributions (relative to the base value) of each feature to the takt prediction at the individual observations level. To see how this tool would be used, consider Figure 32.

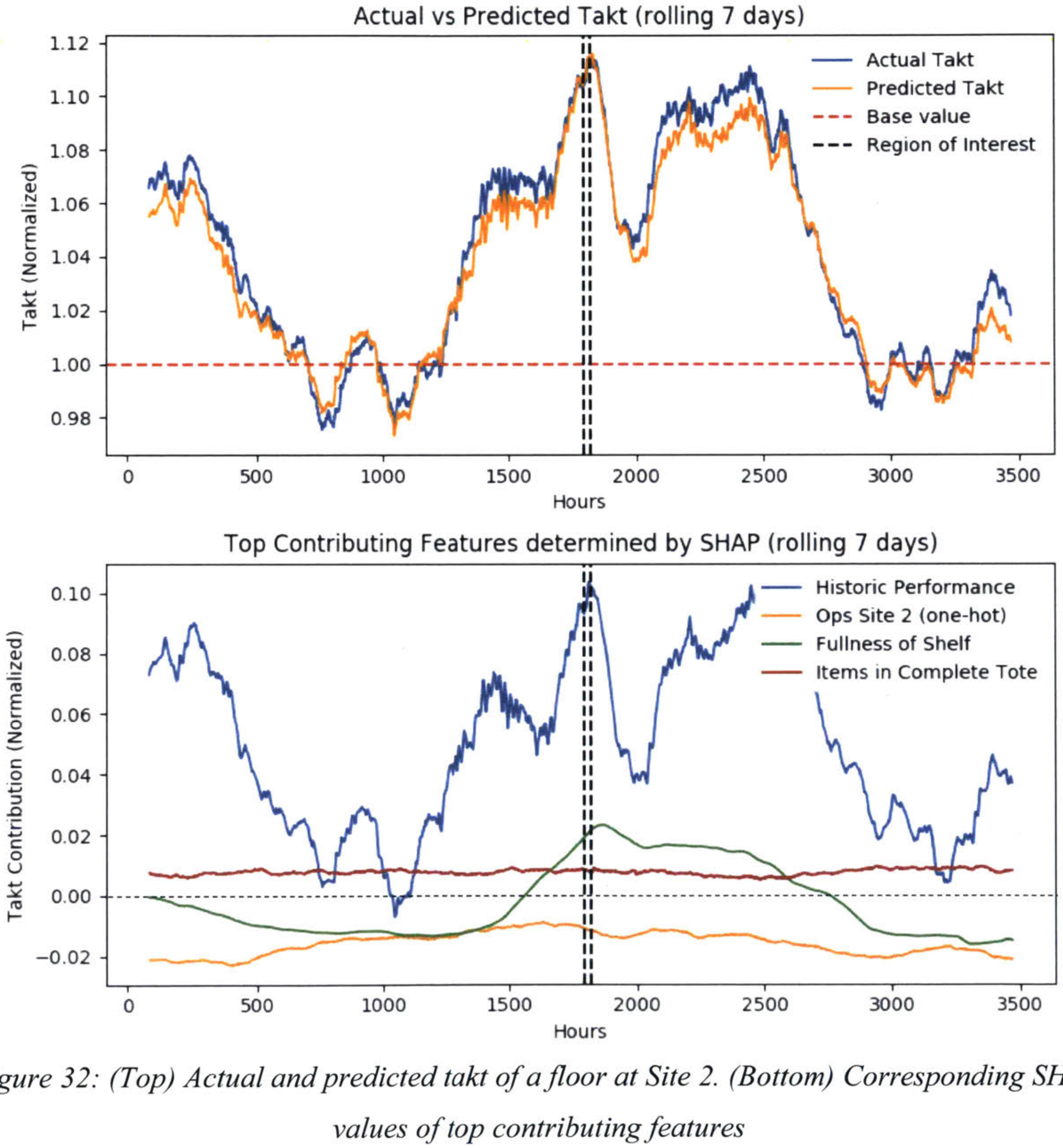


Figure 32: (Top) Actual and predicted takt of a floor at Site 2. (Bottom) Corresponding SHAP values of top contributing features

Figure 32 shows a time series of takt (normalized to the base value for confidentiality and smoothed using a 7-day averaging window for easier viewing) of a particular robotics floor at Site 2 across around 3500 hours of data. As a reminder, the base value is the ML model's output if no features are used (essentially sample mean of takt across all the whole data set). The top plot of Figure 32 shows agreement between the model and actual takt. The bottom plot of Figure 32 shows the top contributing features based on their SHAP values (also smoothed using a 7-day averaging window for easier viewing).

Visually in this example, we can see that the major contributor to fluctuation of takt is due to the Historic Performance feature. This indicates that the fluctuations are largely driven by the underlying aptitude of the operators staffed.

Other than the Historic Performance feature, we can see that the Fullness of Shelves plays a secondary role as well. From hours around 0 to 1500 (period 1), Fullness of Shelves shows a negative contribution, indicating that Fullness of Shelves assisted in reducing takt. Then from hours around 1500 to 2500 (period 2), Fullness of Shelves shows a positive contribution, indicating that Fullness of Shelves is exacerbating takt. Digging deeper into the data, we can confirm that there was a change in inventory levels where Fullness of Shelves increased from around 45% to 50% corresponding to the period 1 and period 2 respectively.

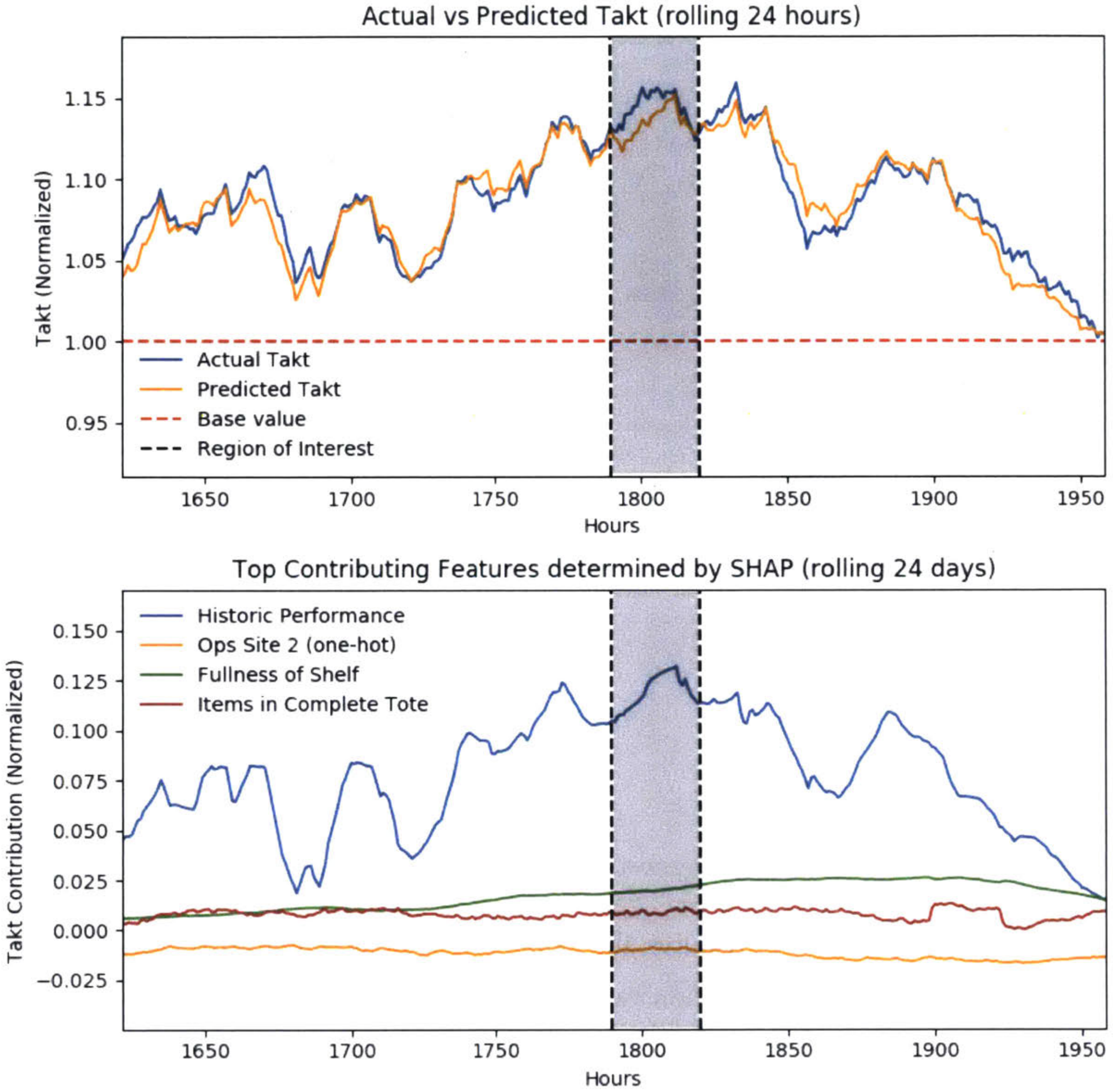


Figure 33: Closer view of the region of interest

Diagnosing root-causes of takt performance deviations at any time is as simple as taking a cross-section of the SHAP value waveforms. Suppose that a user is interested in understanding why takt was so high during a sliver of time between hours of 1790 and 1820 (see "region of interest" in Figure 33), which showed an average of +0.131 above the base value. We can zoom into the corresponding SHAP values associated with that region of time, take the average SHAP values by each feature, and generate a diagnostic waterfall plot as shown in Figure 34.

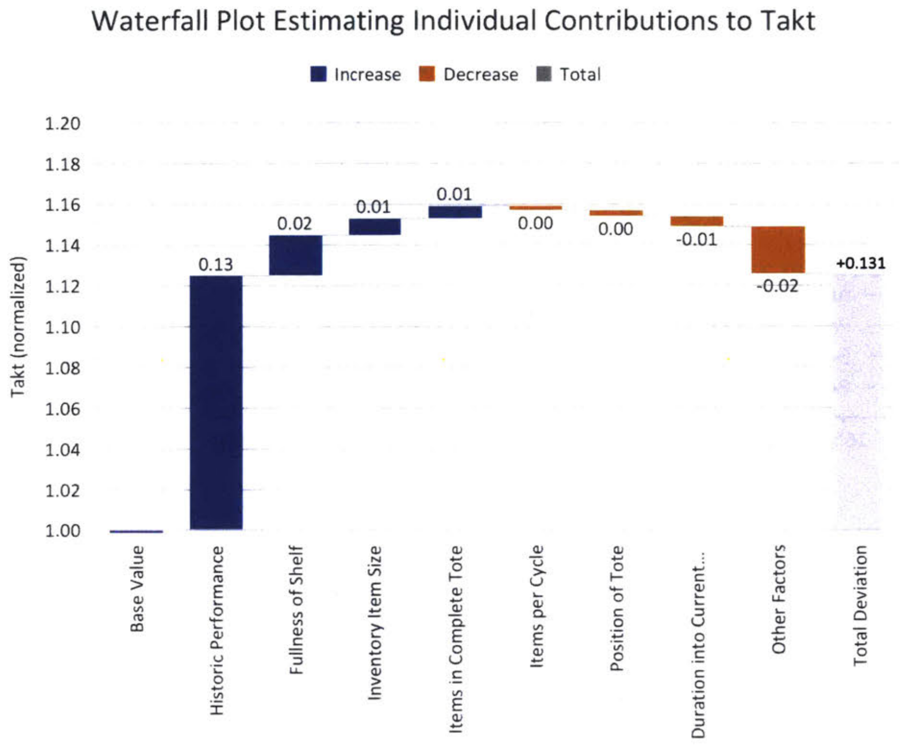


Figure 34: Diagnostic Tool using SHAP values

Using the average SHAP values of features, the tool is able to highlight how different features contributed to the overall +0.131 deviation away from the base value. For example, we can see that the biggest contributor was the "Historic Performance" feature, which may hint that the proficiency of the operators staffed during this period may need to be further investigated. Similarly, we also see that the "Fullness of Shelves" feature also increased takt, which may lead management to review current levels of inventory storage.

7 Conclusion

As the research problem we analyzed is a sub-system within a much larger and more complex fulfillment engine, any recommendations we provide may lead to a locally optimal solution. So instead, we will emphasis the following in our concluding thoughts.

7.1 Bridging Domain and Technical Experts

We believe that the positive outcome of our research project was a result of pursuing a collaborative and iterative approach to model development. Particularly for new entrants, developing the right ML model given limited time is inherently risky.

From a skill set perspective, the ideal practitioner needs to have the right technical abilities and domain expertise relevant to the specific problem at hand. This is often unrealistic, instead there are typically gaps between people who implement ML and people who have domain expertise. Businesses seeking to develop ML solutions should consider whether they have the right organizational culture and structural mechanisms needed to foster cross-functional collaboration.

From a technical feasibility perspective, the experimental nature of ML almost always means that the first iteration of the model rarely is the best solution. Depending on a business's stage of data maturity, the availability of historic data and ability to collect new data may also become barriers. Thus, we recommend that teams should anticipate and plan for several iterations within the time constraints of their project. An effective way to mitigate risk is to focus on iterating as cheaply and quickly as possible early into a project. After each iteration, the team will quickly understand where to invest future efforts.

7.2 Determining Model Sufficiency

The collaborative nature of building ML solutions can also lead to large influxes of recommendations and ideas. For example, stakeholders may constantly recommend new types of data to explore or approaches to data processing. Practitioners might be tempted to pursue multiple avenues and stretch the finite resources. Our recommendation is for practitioners to define criteria of success for a model based on the intended business problem early on, which can then be referenced often when vetting new modelling approaches. People often get fixated by model accuracy. However, depending on the application, model interpretability or complexity may also be relevant.

7.3 Diminishing Returns of Data

Google's Research Director Peter Norvig once claimed that "We don't have better algorithms. We just have more data" [23]. But as we have also seen through our project, having more data does not necessary guarantee significant improvement in model performance. With using just 10% of the data for training, we are able to achieve an MAPE of 4.35%, which uses over 6 times less training data than our topline performance of MAPE of 4.01%. We caution against the notion that quantity of data alone is a magic bullet. Equally, if not more, important aspects include data quality and feature selection.

7.4 Importance of Operator Training and Engagement

All evidence from our research seem to indicate that human factors such as historic performance, work schedule, and time of the day play a significant role in the takt model. From our final model's feature importance ranking, human related features accounted for around 50% of importance. From our Empirical Trade-off Analysis approach, we see how Weekly Worked Hours (WWH) and Duration into Current Session (DCS) are associated with negative impacts on takt when operators are underemployed or work for too long without a break. From the example Diagnostic Tool using SHAP Values, we observed how the major contribution to takt performance deviation is simply due to the staffing of high or low historic performers (a phenomenon we have anecdotally observed across other floors).

The strong benefits associated with investing into employee is not lost amongst leading e-commerce organizations. Amazon has implemented programs such as "Career Choice" and "Career Skills" which aim to motivate and retain employees [24]. Career Choice is available to Amazon hourly associates who have been employed for one continuous year and pre-pays 95% of tuition and fees for associates to earn certificates and associate degrees in high-demand occupations such as aircraft mechanics, computer-aided design, machine tool technologies, medical laboratory science, dental hygiene, solar technician and nursing. Career choice is a free development program available to all hourly Amazon employees beginning on day one. This program offers classes on skills such as resume building, interviewing skills, effective speaking, time management, Microsoft excel and more.

As organizations such as Amazon adopt new HMI technologies, we recommend that they invest more into initiatives to improved employee retention, work schedule, and skill coaching.

7.5 Future Work

7.5.1 Experimental Validation in the Field

Another step to building confidence in the model is to run controlled testing in collaboration with an operations site. In a series of pilot studies, we can run experiments where certain parameters of operations are changed and the corresponding takt value measured. In parallel, we could run the ML model to reveal what the algorithm believes the new takt value should be.

7.5.2 Diagnostic Tool User Interface Refinement

We had shown a proof-of-concept diagnostic tool using SHAP values. While this is a noteworthy first step, we believe that further refinement is needed. Depending on the exact target user, we may benefit from showing only a subset of metrics available. Each metric presented to the user should be evaluated carefully. Ultimately, we want to present the right information so that target users are empowered to make better decisions.

References

[1] Retail e-commerce sales worldwide from 2014 to 2021. [Online; accessed 1-April-2019]. URL: **https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/**

[2] Amazon.com, American Company. [Online; accessed 1-April-2019]. URL: **https://www.britannica.com/topic/Amazoncom**

[3] Deighton, John A., and Kayla Bakshi. "Webvan: Groceries on the Internet." Harvard Business School Case 500-052, November 1999. (Revised March 2003.)

[4] Webvan's Demise or When Technology Fails to Meet Operations, November 2016. [Online; accessed 1-April-2019]. URL: **https://rctom.hbs.org/submission/webvans-demise-or-when-technology-fails-to-meet-operations/**

[5] Think Your Office Is Soulless? Check Out This Amazon Fulfillment Center, July 2013. [Online; accessed 1-April-2019]. URL: **https://www.fastcompany.com/1672939/think-your-office-is-soulless-check-out-this-amazon-fulfillment-center**

[6] Meet Amazon's busiest employee -- the Kiva robot, November 2014. [Online; accessed 1-April-2019]. URL: **https://www.cnet.com/news/meet-amazons-busiest-employee-the-kiva-robot/**

[7] Amazon plans to tap solar energy at 50 fulfillment centers, March 2017. [Online; accessed 1-April-2019]. URL: **https://www.bizjournals.com/seattle/news/2017/03/02/amazon-tap-solar-energy-50-fulfillment-centers.html**

[8] Despite Decision, Amazon Has Huge NJ Presence, November 2018. [Online; accessed 1-April-2019]. URL: **https://njmonthly.com/articles/jersey-living/despite-decision-amazon-huge-nj-presence/**

[9] Amazon showcases new fulfillment center in Houston, September 2018. [Online; accessed 1-April-2019]. URL: **https://www.houstonchronicle.com/business/article/Amazon-showcases-new-fulfillment-center-in-Houston-13211073.php**

[10] How Amazon triggered a robot arms race, March 2016. [Online; accessed 1-April-2019]. URL: **https://www.chicagotribune.com/bluesky/technology/ct-amazon-distribution-center-robots-20160629-story.html**

[11] How Robots and Drones Will Change Retail Forever, October 2018. [Online; accessed 1-April-2019]. URL: **https://www.wsj.com/articles/how-robots-and-drones-will-change-retail-forever-1539604800**

[12] N. Paperno, M. Rupp, E. M. Maboudou-Tchao, J. A. Smither and A. Behal, "A Predictive Model for Use of an Assistive Robotic Manipulator: Human Factors Versus Performance in Pick-and-Place/Retrieval Tasks," in *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 6, pp. 846-858, Dec. 2016.

[13] H. Iwase and A. Murata, "Modelling of human's three-dimensional movement-extending Fitts' model to three-dimensional pointing task," *Proceedings 10th IEEE International Workshop on Robot and*

Human Interactive Communication. ROMAN 2001 (Cat. No.01TH8591), Bordeaux, Paris, France, 2001, pp. 594-599.

[14] C. Pérez-D'Arpino and J. A. Shah, "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 2015, pp. 6175-6182.

[15] 15,000 amazon kiva robots drive eighth generation fulfillment center, December 2014. [Online; accessed 1-April-2019]. URL: **https://www.designboom.com/technology/amazon-kiva-robots-generation-fulfillment-12-02-2014/**

[16] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 785-794. DOI: **https://doi.org/10.1145/2939672.2939785**

[17] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13 (February 2012), 281-305.

[18] Scott M. Lundberg, Su-In Lee. 2017. Consistent feature attribution for tree ensembles. ICML Workshop on Human Interpretability in Machine Learning (WHI 2017), Sydney, NSW, Australia. arXiv: **https://arxiv.org/abs/1802.03888v3**

[19] Informs Biographical Profiles: Lloyd S. Shapley. [Online; accessed 1-April-2019]. URL: **https://www.informs.org/Explore/History-of-O.R.-Excellence/Biographical-Profiles/Shapley-Lloyd-S**

[20] Katharina Blatter, Christian Cajochen, Circadian rhythms in cognitive performance: Methodological constraints, protocols, theoretical underpinnings, Physiology & Behavior, Volume 90, Issues 2–3, 2007, Pages 196-208, ISSN 0031-9384, DOI: **https://doi.org/10.1016/j.physbeh.2006.09.009**

[21] Deep dive on AWS vs. Azure vs. Google cloud storage options, 2017. [Online; accessed 27-April-2019]. URL: **https://www.networkworld.com/article/3191520/deep-dive-on-aws-vs-azure-vs-google-cloud-storage-options.html**

[22] Borgen, W. A., Amundson, N. E., & Harder, H. G. (1988). The experience of underemployment. *Journal of Employment Counseling, 25*(4), 149-159.

[23] Google's "Infringenovation" Secrets, 2011. [Online; accessed 1-April-2019]. URL: **https://www.forbes.com/sites/scottcleland/2011/10/03/googles-infringenovation-secrets/#716ceb4e30a6**

[24] Amazon Fulfillment Center Training. [Online; accessed 25-April-2019]. URL: **https://www.aboutamazon.com/amazon-fulfillment/our-fulfillment-centers/training**