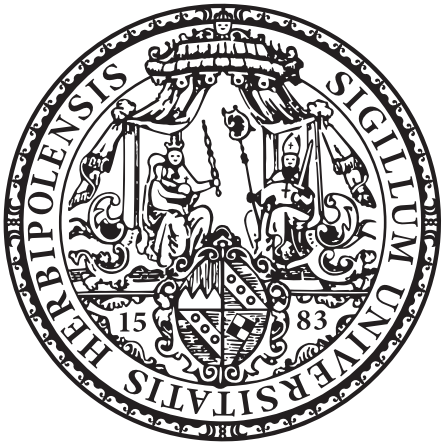
Julius-Maximilians-Universität Würzburg
Wirtschaftswissenschaftliche Fakultät



Data-driven Operations Management: From Predictive to Prescriptive Analytics

Inauguraldissertation
zur Erlangung des akademischen Grades
doctor rerum politicarum (Dr. rer. pol.)

vorgelegt von
M.Sc. Fabian Michael Taigel
geboren in Tübingen

Name und Anschrift:	Fabian Michael Taigel Erthalstr. 34 97074 Würzburg
Erstgutachter:	Prof. Dr. Richard Pibernik
Zweitgutachter:	Prof. Dr. Christoph Flath
Datum der Einreichung:	29. Oktober 2019

Acknowledgements

First, I thank my doctoral advisor, Prof. Dr. Richard Pibernik, for his persistent and inspiring support over the last five years. Our countless discussions provided invaluable guidance by sparking my excitement in Spanish wine and cuisine and led to the foundations of a (hopefully) successful start-up.

I also thank Prof. Dr. Christoph Flath for serving as my second advisor and for his support during the development of this thesis. His ability to understand and evaluate my research and to offer interesting new ideas was both helpful and astonishing.

I thank Prof. Gah-Yi Ban for the opportunity to spend a quarter at the London Business School and for providing inspiration for my work on prescriptive inventory management. Our discussions and the contacts that emerged from my stay have been central to the success of my research journey.

In addition, I thank all of my colleagues at the Chair of Logistics and Quantitative Methods in Business Administration for the interesting discussions, legendary Wednesday nights, and countless lunch breaks. Julian, thank you for your substantial support over the entire term of the PRACTICE project. Jan, you are the best combination of co-author, foosball partner, flat-mate and co-founder ever.

This dissertation was financially supported by the EU (FP7/ 2007-2013), grant agreement no. 609611 (PRACTICE), for which I am grateful.

Finally, I must express my deep gratitude to my family for their continuing support of my professional and personal development.

Contents

Deutschsprachige Zusammenfassung (Summary in German Language)

Diese Dissertation besteht aus fünf inhaltlich abgeschlossenen Teilen, die ein übergeordnetes Thema zur Grundlage haben: Wie können Daten genutzt werden, um bessere Bestands- und Kapazitätsplanung zu ermöglichen? Durch die zunehmende Digitalisierung stehen in verschiedensten Wirtschaftsbereichen mehr und mehr Daten zur Verfügung, die zur besseren Planung der Betriebsabläufe genutzt werden können. Historische Nachfragedaten, Sensordaten, Preisinformationen und Daten zu Werbemaßnahmen, sowie frei verfügbare Daten wie z.B. Wettervorhersagen, Daten zu Schulferien, regionalen Events, Daten aus den Sozialen Medien oder anderen Quellen enthalten potentiell relevante Informationen, werden aber häufig noch nicht zur Entscheidungsunterstützung genutzt.

Im ersten Artikel, "Privacy-preserving condition-based forecasting using machine learning", Kapitel 2 beziehungsweise Taigel et al. [2018], wird aufgezeigt, wie sensitive Zustandsdaten zur Nachfragevorhersage von Ersatzteilbedarfen nutzbar gemacht werden können. Es wird ein Modell entwickelt, das es erlaubt, Vorhersagen auf verschlüsselten Zustandsdaten zu erstellen. Dies ist z.B. in der Luftfahrt relevant, wo Dienstleister für die Wartung und Ersatzteilversorgung von Flugzeugen verschiedener Airlines zuständig sind. Da die Airlines befürchten, dass Wettbewerber an sensitive Echtzeitdaten gelangen können, werden diese Daten dem Wartungsdienstleister nicht im Klartext zur Verfügung gestellt. Die Ergebnisse des implementierten Prototypen zeigen,

dass eine schnelle Auswertung maschineller Lernverfahren auch auf großen Datenmengen, die verschlüsselt in einer SAP HANA Datenbank gespeichert sind, möglich ist.

Die Artikel zwei und drei behandeln innovative, datengetriebene Ansätze zur Bestandsplanung. Der Artikel "Machine learning for inventory management: Analyzing two concepts to get from data to decisions" in Kapitel 3 analysiert zwei Ansätze, die Konzepte des maschinellen Lernens nutzen um aus historischen Daten Bestandsentscheidungen zu lernen. Im dritten Artikel, "Machine learning for inventory management: Analyzing two concepts to get from data to decisions", wird in Kapitel 4 ein neues Modell zur integrierten Bestandsoptimierung entwickelt und mit einem Referenzmodell verglichen, bei dem die Schätzung eines Vorhersagemodells und die Optimierung der Bestandsentscheidung separiert sind. Der wesentliche Beitrag zur Forschung ist hierbei die Erkenntnis, dass unter bestimmten Bedingungen der integrierte Ansatz klar bessere Ergebnisse liefert und so Kosten durch Unter- bzw. Überbestände deutlich gesenkt werden können.

In den Artikeln vier und fünf werden neue datengetriebene Ansätze zur Kapazitätsplanung vorgestellt und umfassend analysiert. Im Artikel "Data-driven capacity management with machine learning: A new approach and a case-study for a public service office", Kapitel 5 beziehungsweise Taigel et al. [2019], wird ein datengetriebenes Verfahren zur Kapazitätsplanung eingeführt und auf das Planungsproblem im Bürgeramt Wilhelmshaven angewandt. Das Besondere hierbei ist, dass die spezifische Zielfunktion (maximal 20% der Kunden sollen länger als 20 Minuten warten müssen) direkt in ein maschinelles Lernverfahren integriert wird, womit dann ein Entscheidungsmodell aus historischen Daten gelernt werden kann. Hierbei wird gezeigt, dass mit dem integrierten Ansatz die Häufigkeit langer Wartezeiten bei gleichem Ressourceneinsatz deutlich reduziert werden kann. Im fünften Artikel, "Prescriptive call center staffing", Kapitel 6, wird ein Modell zur integrierten Kapazitätsoptimierung für ein Call Center entwickelt. Hier besteht die Innovation darin, dass die spezifische Kostenfunktion eines Call Centers in ein maschinelles Lernverfahren integriert wird. Die Ergebnisse für Daten von zwei Call Centern zeigen, dass mit dem neuentwickelten Verfahren, die Kosten im Vergleich

zu dem gängigen Referenzmodell aus der Literatur deutlich gesenkt werden können.

Den inhaltlichen Rahmen dieser Dissertation bilden die Einleitung im folgenden Kapitel sowie ein Ausblick in Kapitel 7. Im Hauptteil nicht dargestellte Beweise und Algorithmen werden in den Anhängen A bis D zusammengefasst.

1 Introduction

Autonomous cars and artificial intelligence that beats humans in *Jeopardy* or *Go* are glamorous examples of the so-called Second Machine Age that involves the automation of cognitive tasks [Brynjolfsson and McAfee, 2014].However, the larger impact in terms of increasing the efficiency of industry and the productivity of society might come from computers that improve or take over business decisions by using large amounts of available data. This impact may even exceed that of the First Machine Age, the industrial revolution that started with James Watt's invention of an efficient steam engine in the late eighteenth century. Indeed, the prevalent phrase that calls data "the new oil" indicates the growing awareness of data's importance. However, many companies, especially those in the manufacturing and traditional service industries, still struggle to increase productivity using the vast amounts of data [for Economic Co-operation and Development, 2018].

One reason for this struggle is that companies stick with a traditional way of using data for decision support in operations management that is not well suited to automated decision-making. In traditional inventory and capacity management, some data – typically just historical demand data – is used to estimate a model that makes predictions about uncertain planning parameters, such as customer demand. The planner then has two tasks: to adjust the prediction with respect to additional information that was not part of the data but still might influence demand and to take the remaining uncertainty into account and determine a safety buffer based on the underage and overage costs. In the best case, the planner determines the safety buffer based on an optimization model that takes the costs and the distribution of historical forecast errors into account; however, these decisions are usually based on a planner's experience and intuition, rather than on solid data analysis.

This two-step approach is referred to as *separated estimation and optimization (SEO)*. With SEO, using more data and better models for making the predictions would improve only the first step, which would still improve decisions but would not automize (and, hence, revolutionize) decision-making. Using SEO is like using a stronger horse to pull the plow: one still has to walk behind.

The real potential for increasing productivity lies in moving from predictive to prescriptive approaches, that is, from the two-step SEO approach, which uses predictive models in the estimation step, to a prescriptive approach, which integrates the optimization problem with the estimation of a model that then provides a direct functional relationship between the data and the decision. Following Akcay et al. [2011], we refer to this integrated approach as *joint estimation-optimization (JEO)*. JEO approaches prescribe decisions, so they can automate the decision-making process. Just as the steam engine replaced manual work, JEO approaches replace cognitive work.

In addition to this practical motivation, JEO approaches are at the forefront of current research. While the literature offers numerous applications of machine learning (ML) for making demand predictions in operations management problems [Choi and Varian, 2012, Asur and Huberman, 2010, Goel et al., 2010, Stein et al., 2018, e.g.,], research on prescriptive JEO approaches is just picking up [Ban and Rudin, 2018, Bertsimas and Kallus, 2019].

The overarching objective of this dissertation is to analyze, develop, and evaluate new ways for how data can be used in making planning decisions in operations management to unlock the potential for increasing productivity. In doing so, the thesis comprises five self-contained research articles that forge the bridge from predictive to prescriptive approaches. While the first article focuses on how sensitive data like condition data from machinery can be used to make predictions of spare-parts demand, the remaining articles introduce, analyze, and discuss prescriptive approaches to inventory and capacity management.

More specifically, the first article shows how massive amounts of sensitive condition-based data from distributed sources like sensor-equipped machinery run by various companies can be used to improve maintenance-demand

forecasting and spare parts and capacity planning. The article describes an innovative application for privacy-preserving forecasting of demand for spare parts based on distributed condition data that combines state-of-the art ML techniques – decision-tree classification in particular – with order-preserving encryption and uses encrypted data to forecast demand. The application's use is appropriate when one is planning for spare parts for the maintenance of condition-monitored machinery, and it is particularly suitable for cloud-based implementation since it can be applied efficiently to encrypted in-memory databases.

In the second article, we analyze the performance drivers of data-driven prescriptive inventory management in a Newsvendor setting with non-stationary demand. We study linear quantile regression and tree-based regression, two novel approaches that are based on ML – and that use historical demand observations and auxiliary data to prescribe optimal inventory quantities. We identify three major performance drivers: non-linearity, heteroscedasticity (i.e., feature-dependent uncertainty), and usability. We evaluate both models in an extensive simulation experiment that controls for various properties of the feature-demand relationship and on a complex real-world data set from a restaurant chain. From these experiments we conclude that, in situations in which the structure of the feature-demand relationships is not known to be predominantly linear – the typical case in practice – the tree-based approach is much more robust. The tree-based approach also provides better results in case of heteroscedasticity and, in the real-world setting, performs better with small data-sets because of its superior built-in feature selection, which is important in terms of usability in practice.

The third article considers an inventory problem with non-stationary demand, where variations in demand are driven by observable features. We use this setting to determine the difference between two fundamentally different concepts for turning data into decisions: the classical SEO concept, and the more recent prescriptive JEO concept. We add to the recent stream of research on JEO approaches in two ways: by introducing a novel JEO approach based on random forests, a powerful and flexible ML technique, and by providing a rigorous examination of what drives performance differences between

the JEO concept and the SEO concept. More specifically, we compare JEO's performance with that of its SEO counterpart – that is, an SEO approach that applies an ML approach to predicting demand that is similar to the approach JEO uses to make prescriptions. In the literature, these comparisons are either missing or show no significant differences. We provide analytical results for the comparison of SEO and JEO with an underlying linear model, whereas we use a controlled simulation setting and a real-world data set from a restaurant chain for the kernel-based and the random forest-based approaches. We find that, when there is feature-dependent uncertainty, JEO can lead to significantly better results than the SEO counterpart can, but results from a real-world data set suggest that the performance difference between the approaches is only marginal and that SEO's performance is surprisingly robust.

In the fourth article, we consider the case of a public service office in Germany that provides services like passport and ID card applications and notifications of changes of address. Their decision problem is to determine the staffing level for a specific staffing time slot (e.g., next Monday, 8am to 12.30pm). The required staffing level – that is, the service capacity – is driven by features like the day of the week and whether the day falls during school vacations. We present an innovative JEO approach to prescribing capacities that requires no assumptions about the underlying arrival process. In this approach, we integrate the service goals (e.g., "No more than 20% of the customers have to wait more than 20 minutes") into an ML algorithm to learn a functional relationship between features and the prescribed capacity from historical data. We analyze the performance of our JEO approach on a real-world dataset and compare it to an SEO approach that first uses out-of-the-box ML to predict arrival rates and then determines the capacity using queuing models. The JEO approach significantly outperformed the commonly used SEO benchmark, and both data-driven approaches significantly outperformed a naive benchmark.

The fifth article provides a JEO approach to staffing inbound call centers, where the main difference from the setting in the fourth article is that we can assign costs to waiting time and abandonments. Again, we need to determine the staffing level for a specific staffing time-slot (e.g., next Monday, 8am to

9am), and the required capacity is driven by observable features. We present an innovative JEO approach to prescribing capacities that can incorporate such features and that does not require assumptions about the underlying arrival process. We show how to integrate abandonment-cost functions into an ML algorithm to learn a functional relationship between features and optimal capacity from historical data. We find that our JEO approach significantly outperforms a state-of-the-art, data-driven benchmark in two real-world settings and that our approach is especially useful in cases with non-stationary arrival rates.

An overview of the scientific contribution of this dissertation is presented in Table 1.1.

ML and data is used in innovative ways in all five articles to improve current approaches to solving inventory or capacity management problems. The articles show that, by moving from predictive to prescriptive approaches, we can improve data-driven operations management in two ways: by making decisions more accurate and by automating decision-making. Thus, this dissertation provides examples of how digitization and the Second Machine Age can change decision-making in companies to increase efficiency and productivity.

	Methodological Contribution	*Implemented models or ML-techniques*	*Conceptual findings*
Privacy-preserving condition-based forecasting (Chapter 2)	• Condition-based forecasting of spare parts demand • Combination of ML with privacy-preserving computation	• Decision Tree Learning	• Enable usage of sensitive data from distributed sources e.g., for collaborative spare-parts forecasting • Implemented prototype achieves practicable computational performance
Prescriptive analytics for inventory management (Chapter 3)	• Sophisticated simulation framework to analyze drivers of performance of planning approaches • Evaluation of two state-of-the art approaches for data-driven inventory management	• Linear quantile regression • Tree-based regression (Random Forests)	• Identification of heteroscedasticity as key driver of performance • Tree-based regression is more robust against heteroscedasticity and non-linearity.
Data-driven inventory management (Chapter 4)	• Integrated approach for data-driven inventory management in newsvendor settings • Comparison with non-integrated benchmark	• Tree-based regression (CART and Random Forest)	• In settings with strong heteroscedasticity the integrated approach performs significantly better • On a real-world dataset, we find no dominance of either approach.
Data-driven capacity management for a public service office (Chapter 5)	• Integrated approach for data-driven capacity management with service level targets	• Tree-based regression (CART) • Erlang-C queuing model (as benchmark)	• New approach achieves significantly higher service levels with same capacities on a real-world dataset
Data-driven capacity management for call centers (Chapter 6)	• Integrated cost-based approach for data-driven capacity management	• Tree-based regression (CART) • Bassamboo's data driven method (as benchmark)	• Our approach significantly reduces the costs in two real-world applications • Our approach can deal better with variations in the arrival rates

Table 1.1: Overview of scientific contribution.

2 Privacy-preserving condition-based forecasting using machine learning

As machines get smarter, massive amounts of condition-based data from distributed sources become available. This data can be used to enhance maintenance management in several ways, such as by improving maintenance demand forecasting and spare parts and capacity planning. Regarding the former, machine learning techniques promise substantial benefits for forecasting the demand for spare parts over conventional techniques that are commonly used. While development and implementation of these techniques is difficult, practical applications pose another important challenge to providers of maintenance, repair, and overhaul services. Their customers are reluctant to provide access to sensitive real-time data because of privacy concerns, and even more so when their data is stored and processed in the cloud. In this paper we describe an application for privacy-preserving forecasting of demand for spare parts based on distributed condition data. It combines machine learning techniques – more specifically, decision-tree classification – with order-preserving encryption. The application is appropriate whenever planning for spare parts for the maintenance of condition-monitored machinery is needed, and it is particularly suitable for cloud-based implementation. [1]

[1]This paper was published in the *Journal of Business Economics* Taigel et al. [2018]. Is is co-authored by Anselme K. Tueno and Richard Pibernik.

2.1 Introduction

In many industries the availability of machines ("uptime") is critical to successful operations, so companies (or their service providers) perform tasks related to maintenance management, including forecasting future demand for service and spare parts, managing inventories of spare parts, planning maintenance capacity, and scheduling service operations. The condition of individual machines and components determines maintenance demand and is, as such, the main driver of any maintenance operation. Modern machinery is equipped with sensors and other devices that provide information about the machinery's current condition, to which we refer as "condition data". Condition data is data about the use of a component (e.g., machine hours since last overhaul) and information from sensors (e.g., oil temperature alerts, vibration levels). The effective use of condition data can significantly enhance maintenance operations because it improves forecast accuracy and supports the timely completion of maintenance tasks [Elwany and Gebraeel, 2008, Deshpande et al., 2006, Kurz, 2016].

Enhancing the operations of a maintenance, repair, and overhaul (MRO) service provider through the use of condition data poses two important challenges: First, it requires companies to devise and implement the right techniques and methods for generating good forecasts from condition data that is typically distributed across multiple systems belonging to multiple customers (i.e., machine users and owners). Meeting this challenge calls for a sophisticated machine learning and data mining solution in the cloud. The second challenge, which cannot be addressed in isolation, is ensuring the protection of customers' sensitive and private data. While machines and their condition data may belong to numerous customers, maintenance operations are usually carried out by a third party like an original equipment manufacturer (OEM) or a specialized MRO. Customers are often reluctant to make their condition data available to a third party or even a cloud-based system that runs outside their own IT infrastructures. Protecting the extensive and detailed data about current and future machine use and operating conditions from leaking to competitors and malicious intruders can be a major concern for

customers [Barkataki and Zeineddine, 2015, Zilli et al., 2015]. Therefore, ensuring data privacy can be a decisive factor in whether customers will adopt advanced maintenance management concepts based on condition data. The two challenges are interdependent: ensuring data privacy limits the choice to use machine learning and data-mining techniques and their implementation in cloud-based systems.

In this paper we outline an approach that combines machine learning and cryptographic techniques to enable privacy-preserving forecasting of maintenance demand based on distributed condition data. Our decentralized solution employs sophisticated machine learning techniques that are executed on data that can reside in a variety of cloud systems. It produces an aggregate forecast of maintenance demand for the MRO and corresponding information about the reliability of the forecast while neither the MRO nor any other party (including the cloud providers) gain access to individual customers' private condition data.

Our research is motivated by the problem of a MRO from the aerospace industry that provides maintenance services for commercial airlines' and air forces' jet engines. Modern generations of engines, such as those of Rolls-Royce, track and transmit data in real time from more than twenty parameters, including oil pressure, oil temperatures, and vibration levels. In combination with usage data like performed and planned flight hours and flight cycles, this data can be used for condition-based maintenance planning. This MRO caters to multiple customers (commercial airlines and air forces), each of which maintains its own set of condition data, so it is distributed across multiple proprietary systems and not readily available to the MRO. The customers are reluctant to provide real-time condition data openly to the MRO or a cloud-based system because they fear data leakage to competitors that could benefit from, for example, information about the state and the planned operations of their fleet. Moreover, air forces have to obey strict privacy regulations since leaked information about the state of their fleets is considered a national security concern [Zilli et al., 2015]. The MRO can use the condition data only if it can resolve the data-privacy issue. Figure 2.1 illustrates the current situation with distributed sources of condition data that the MRO

cannot currently use to plan its operations.

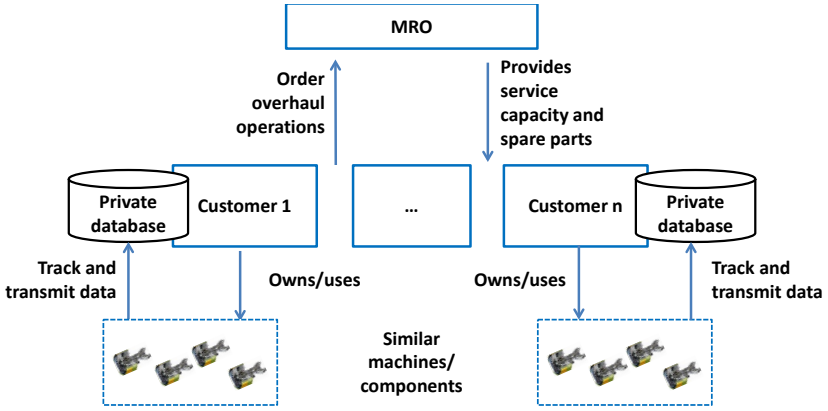


Figure 2.1: One MRO serving n customers with private condition data

The MRO's operational performance is strongly affected by its ability to forecast demand for maintenance services and spare parts. Currently, historical demand is the only source of information available to the MRO for forecasting. Because of the typically varying patterns of demand [Ghobbar and Friend, 2002, van Wingerden et al., 2014], forecast errors are high, leading to the need for high capacity and inventory buffers and, at the same time, frequent delays and contractual penalties. The purpose of the solution we propose in this paper is to improve forecast accuracy for the MRO by using machine learning techniques to generate forecasts based on the distributed condition data while also ensuring data privacy for the MRO's customers. Although we illustrate and formalize our approach for the task of forecasting spare parts demand, the approach can be modified for employment in other forecasting objects (e.g., capacity requirements).

The contribution of our research to practice is straightforward. Our solution can help to remove one of the major obstacles to the effective use of distributed condition data to improve maintenance management. Condition data's significant potential for maintenance management can be fully exploited only if multiple customers' data is combined and processed effectively, so companies who employ similar types of machinery have to make their data available to a third party. At first, this data may not appear critical: For example, the oil pressure of an engine at a certain point in time or the number of hours

an engine has been employed do not appear to be critical and sensitive pieces of information; however, one can obtain a blueprint of a company's operations by combining them over time. It is reasonable to assume that, even beyond our example of the airline industry, this potential will create reluctance and make the case for condition-based maintenance management less attractive unless data privacy can be guaranteed. The approach we present in this paper provides a solution to this problem by exploiting condition-based data while keeping this sensitive data private. Thereby, we make more information usable for the MRO's forecasts. This potentially leads to improved forecast accuracy which can lead to lower spare parts inventory costs for the MRO and shorter lead times for the customers.

From a theoretical and methodological perspective, we are the first to address data privacy issues in condition-based maintenance management. We do not seek to improve the performance of certain machine learning techniques but to show how they can be enhanced to meet customers' privacy requirements by providing a novel combination of machine learning and cryptographic techniques. We develop a framework with which to evaluate which machine learning techniques are suitable for a privacy-preserving implementation, based on which we describe how one particularly well-suited technique, decision trees, can be combined with order-preserving encryption techniques and privacy-preserving aggregation to produce forecasts that are based on condition data without revealing any of the sensitive condition data to any third party, including the MRO and the competitors' cloud providers. Although we focus on the domain of maintenance management, our approach may provide value in other domains where machine learning can be employed to generate valuable information from sensitive data that is distributed across multiple systems of customers (e.g., collaborative demand or supply management).

Our work is part of a large research initiative, the purpose of which is to build a platform for secure cloud computing.[2] The application for privacy-preserving spare parts forecasting with distributed sensitive data that we present in this paper builds on and will be part of this platform. Researchers

[2]Namely, the EU-funded PRACTICE project.

at SAP SE and the University of Würzburg, in close collaboration with an MRO in the aerospace industry, developed a prototype that implements the approach and concept provided in this paper. Preliminary performance tests show promising results, which is notable since runtime issues can hinder the implementation of privacy-preserving solutions in practice.

2.2 Related work

Our work is related to four research streams: 1) condition-based forecasting, 2) collaborative planning, forecasting and replenishment (CPFR), 3) machine learning, and 4) cryptography. The use of condition data for maintenance management in general and forecasting of maintenance demand in particular has received considerable attention. A vast amount of work focuses on detecting a fault in a single component or estimating its remaining useful lifetime. This work – see Jardine et al. [2006] and Peng et al. [2010] for extensive reviews – is particularly relevant to data preprocessing, which we assume is completed before we use the data for machine learning. In this context, data preprocessing means that raw sensor data is transformed into information like alerts on certain values (e.g., a temperature alert) or estimates for remaining useful lifetime. While this information is useful for ad hoc maintenance measures, the work in this field does not consider condition-based demand forecasting on an aggregate level, which is necessary for an external MRO that serves multiple customers. Two (more conceptual) contributions postulate an integration of condition-based information and aggregate maintenance planning: Yam et al. [2001] propose a concept for an intelligent predictive decision support system to predict the trend of equipment deterioration, pointing out that the results of their system can be used as input for subsequent maintenance management planning, such as maintenance scheduling and spare parts inventory planning. Hellingrath and Cordes [2014] identify a research gap with respect to integrating condition data into spare parts demand forecasting and describe an approach with which to overcome this gap. They use a binomial distribution to model spare parts demand and propose to determine its pa-

rameters using replacement probabilities for individual parts provided by an intelligent maintenance system. However, in contrast to our approach, they do not integrate information about the reliability of individual replacement probabilities into their overall concept and don't consider that condition data typically needs to be collected from multiple sources owned by customers that may be competitors, once again making privacy an issue for practical implementations.

Dekker et al. [2013] review the application of condition-based forecasting in practice and provide two case studies on maintenance operations in the aerospace and dredging industries, where information about the installed base is used for spare parts logistics. For the aerospace MRO case, they use the MRO's historical data to estimate the relationship between demand for spare parts and flight hours. The second case is based on the work of Veenstra et al. [2006], who investigate an OEM that manufactures dredging ships and is also responsible for the timely provision of spare parts to the ships used by a variety of dredging companies. The work explores the potential benefits of condition-based maintenance by simulating critical valves' deterioration process and comparing the overall maintenance costs with and without condition-based maintenance.

Veenstra et al. [2006] find significant potential for reducing excessive costs that, without condition-based maintenance, are caused by equipment downtime or unnecessary maintenance operations. But they also find that dredging companies are reluctant to provide the necessary condition-based data because a competitor could obtain information about the sand the dredging ship is producing in a certain location, which is highly sensitive information in this business. This is exactly the issue we tackle in the present paper.

While the aforementioned contributions are primarily of a conceptual nature, some formal quantitative work addresses the forecasting of maintenance demand. Lin et al. [2012], which is most closely related to our approach, consider multiple machines, each equipped with one similar component for which spare parts are kept in a central inventory, and use Markov chains to model the machinery's deterioration process. A Markov state is defined by the number of components that are in each of the predefined deterioration

states, and condition information is considered in terms of the probability that a component will deteriorate one state further (which would also change the state of the entire system). However, in contrast to our work, Lin et al. do not differentiate the probability of one component's further deterioration from that of another component in the same state, hence losing information that could be obtained from the condition data. In addition, they do not elaborate on how condition data can be used to derive probabilities for deterioration, and they neglect any privacy issues. The present paper provides guidance on how to develop a myopic heuristic for an inventory policy that can handle condition-based forecasts.

Our research is also related to the work in the area of collaborative planning, forecasting and replenishment (CPFR). CPFR was initially introduced in the retail sector to improve information sharing and coordination in supply chains, e.g., to rule out planning based on inconsistent forecasts, and to avoid that small variations in final customers' demands lead to large variations further upstream (i.e., to prevent the so called bullwhip effect) [Mertens et al., 2012]. Information sharing is an important element of CPFR [Seifert, 2006]. According to Mertens et al. [2012], one way to obtain common and consistent forecasts is to use an aggregated database. However, as Viswanathan et al. [2007] points out, if one of the potential collaborators perceives the required data as sensitive, collaborative planning will most likely not materialize. Surprisingly, the work by Deshpande et al. [2006] is the only paper that specifically addresses the issue of privacy in the context of CPFR. They propose a CPFR protocol that enables two collaborating parties to obtain a common forecast without the need to openly share potentially sensitive input data. Although our work is based on a similar rationale, it differs significantly from that of Deshpande et al. [2006]: First, we consider a setting where relevant data is distributed among many parties (an MRO and multiple customers), not only between one supplier and one retailer as in Deshpande et al. [2006]. Second, the demand model in Deshpande et al. [2006] is limited to a linear relationship between demand signals and demand realizations, while our tree-based approach can handle more complex feature-demand relationships. Third, we mainly rely on order-preserving encryption to enable privacy-preserving col-

laboration while Deshpande et al. [2006] use a protocol-based approach that is typically much slower. Nonetheless we think, that our approach could be used as part of a CPFR implementation if sensitive data requires privacy-preserving collaboration.

The third area of research that is related to our work is machine learning in the context of operations management. Letourneau et al. [1999] use decision-tree learning based on real-time condition data to develop a prediction model that should alert staff if an aircraft component needs to be replaced. Their work, which is that most closely related to ours in this literature stream, shows that the available data generated by aerospace equipment (even if it does not come from the latest generation of aircrafts) can be used to obtain useful information about a specific component's condition by using decision-tree learning. This seminal work focuses on the improvement of maintenance operations in the very short term (e.g., through real-time alerts), so it does not focus on issues like forecasting demand for spare parts or integrating their results into spare parts inventory planning. For the same reason, they do not address the issues, especially privacy issues, that arise when working with distributed condition data. Dahan et al. [2014] demonstrates the relevance of decision-tree learning by means of two real-world applications in operations management. In the first application they estimate the churn rate of customers of a wireless operator, given detailed data about the customers (seniority, monthly rate, usage, etc.). The second case predicts whether a potential customer of a security equipment company will respond positively or negatively if contacted by the sales team, given data about customer type (private or business), customer size, contact initiation (by company or by customer), sales person, and so on. In both cases the application of decision trees in these binary classification problems lead to highly valuable decision support. Although these applications are tailored towards different domains, they are helpful in explaining how decision-tree learning can be applied to improve operations management.

One of the unique features of our work is the integration of machine learning with cryptographic techniques to ensure that forecasting of maintenance demand can be carried out in a privacy-preserving way. Therefore, the work

in the field of cryptography is applicable to our interests. Our work faces two cryptographic problems: On one hand, we need to make distributed sensitive data accessible, such as via a cloud-based system. Since customers will provide their data only in encrypted form, in order to obtain the information that is required for the aggregated forecasting from the encrypted data, we need order-preserving encryption in order to perform comparison operations on data that is encrypted. On the other hand, we need to aggregate the results we obtain from evaluating customers' encrypted databases. Therefore, we apply a secret-sharing protocol, which is a concept from the broader field of secure multi-party computation. The first concept for order-preserving encryption was developed by Agrawal et al. [2004], but in order to avoid the trusted central party that is part of the original encryption scheme, we build our work mainly on Popa et al. [2011], who not only avoid the trusted central party but also develop an efficient framework for implementation of this advanced concept. In addition, we provide an innovative combination with a privacy-preserving aggregation protocol that is based on secure multi-party computation, which provides protocols that allow multiple parties to jointly compute commonly known functions without revealing any of the individual inputs. Four important contributions in the field of secure multi-party computation (SMC) are strongly related to our work. The work by Deshpande et al. [2006] uses SMC to enable collaborative forecasting in a retail setting where a supplier and a retailer each have private information about future customer demand. Their work indicates that various kind of data (e.g., on promotions) can be used to derive the additional information on future demand, but in contrast to our work, Deshpande et al. [2006] do not explain how they use this data – that is, how they obtain the single indicator per period that they use in their model.

Most closely related to our approach are Bost et al. [2015], Ducas and Micciancio [2015] and Wu et al. [2016], who also provide solutions for privacy-preserving classification using a decision tree, although in a somewhat different setting. In their scenarios, a server holds a private decision-tree model and a client holds a private input. The two parties use a SMC scheme to classify the client's input such that, at the end of the computation, only the client learns

the result and nothing else is revealed to the other parties. When the parties execute an oblivious comparison protocol, privacy for both parties is guaranteed, but interaction is needed in order to evaluate the comparison of two encrypted numbers, leading to low computational performance. Therefore, these approaches are impractical for large database. In addition, in contrast to the method presented in our work, these schemes cannot be retrofitted to existing database management systems. Finally, Lindell and Pinkas [2000] use SMC in a concept for privacy-preserving decision-tree learning. In contrast to our work, they assume that the learning data that is used to build the prediction model (the decision tree) is not centrally available. Such central availability would be helpful in a scenario in which the MRO cannot read the historical sensor data during overhaul operations. To the best of our knowledge, our work is the first to consider the situation of a MRO that serves multiple customers with distributed sensitive data and provides a solution that enables condition-based forecasting of demand for aggregated spare parts without the need to reveal the customers' private condition data.

2.3 Privacy-preserving supervised machine learning

In the context of spare parts forecasting, supervised machine learning is, in the most general terms, used to estimate a functional relationship $y = \boldsymbol{f}_{\boldsymbol{D}}(\boldsymbol{x})$ between the condition data $\boldsymbol{x}$ and the need for replacement y in a certain time period by analyzing a large set of learning data $\boldsymbol{D}$. Our goal is not to predict the exact time of replacement or any distribution of remaining useful lifetime but to determine whether the component needs to be replaced or can be reused after the engine that contains the component is overhauled within a certain time window. The set of learning data $\boldsymbol{D}$ includes values of the condition parameters and the information concerning whether this specific component had to be replaced during past overhaul events; that is, $\boldsymbol{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, N\}$, where y_i is the class label ($\boldsymbol{yes}$ or $\boldsymbol{no}$) and N is the number of available historic observations, that is, the number of rows in a

learning dataset, as depicted in Table 2.1. Each row represents one instance of a part that contains condition data that was measured before the overhaul and the information about whether the part was replaced or not. Table 2.1 contains all parts of a similar type from all of the machinery for which the MRO is responsible.

Table 2.1: Example of a learning dataset $\boldsymbol{D}$

Operating hours	# cycles	# temp_alerts	...	vibration level	**replace**
211	86	4	...	2	**yes**
125	66	1	...	3	**no**
⋮	⋮	⋮	⋮		⋮
170	88	5	...	2	**yes**

We assume that a row in the learning data set $\boldsymbol{D}$ can be collected during or immediately after the overhaul process for the specific part. At this point, the MRO can access the historical condition data directly from the component and can know whether a certain part has to be replaced. Therefore, this historical data is not classified as sensitive from the customers' viewpoint since it reveals no insights on current or future use of the fleet.

In the next phase, the relationship $\boldsymbol{f_D}(\boldsymbol{x})$ is used to classify instances for which the attribute values $\boldsymbol{x}$ are known, but the corresponding y is not. Therefore, we obtain condition data in real time and use our learned model to estimate whether a certain part needs replacement in a pre-defined time interval. Since customers perceive the real-time condition data of the machines that are currently in use as sensitive, privacy requirements are high in this prediction phase, and the data must be protected. Neither the MRO nor any other third party can be able to access this data. To this end, a privacy-preserving implementation of an appropriate machine learning technique is required. As we detail below, this requirement severely limits our choice of applicable machine learning techniques: Not only do we want to ensure good performance in terms of our forecasting results (i.e., the lowest possible forecast error), but we also have to be able to solve realistic problems in an

acceptable amount of time.

Even without privacy requirements, it is a priori not possible to determine which machine learning technique will perform best for a specific data set, since this depends heavily on the particular data and the chosen performance metric [Caruana and Niculescu-Mizil, 2006]. Caruana and Niculescu-Mizil [2006] provide an extensive empirical evaluation of the most common techniques based on eleven binary classification problems and various performance metrics. In their ranking, the top three techniques are *boosted decision trees (boosted-DT)*, *random forests (RF)* and *bagged decision trees (bagged-DT)*, followed by *support vector machines (SVM)*, *artificial neural networks (ANN)*, *K-nearest neighbor (KNN)*, *boosted stumps*, *decision tree learning (DT)*, *logistic regression (LOGIT)* and *Naive Bayes (NB)*. Caruana et al. [2008] extend the evaluation of Caruana and Niculescu-Mizil [2006] to higher dimensional datasets (i.e. with 750 to 650,000 attributes) and state that also for this kind of data, techniques based on *DT* are the top performers.

For our subsequent analysis, which focuses on the feasibility of privacy-preserving classifications using the various concepts, we can summarize this comprehensive list. *Boosted-DT*, *RF*, *bagged-DT* and *boosted stumps* are all based on the concept of *DT*. Classification via one of these tree-based methods basically means evaluating multiple decision trees [Murphy, 2012], so if we find a way to evaluate a single decision tree while satisfying the privacy requirements, we could expand this approach to enable the more sophisticated machine learning techniques. In addition, *NB* is not relevant for our work since it is limited to discrete-valued problems [Murphy, 2012]. In addition, we also consider basic linear regression (LINREG) which is the foundation for additional sophisticated regression techniques such as incremental regression analysis or extensions of the stepwise regression as for example described by Matt [2005]. Therefore, we have six machine learning techniques (listed in Table 2.2): *SVM*, *ANN*, *KNN*, *DT*, *LOGIT* and *LINREG*.

These techniques differ not only in terms of performance (e.g., forecast accuracy) but also in terms of the arithmetic operations (e.g., addition, multiplication, comparison), which is not an issue if classification is done on plaintext (unencrypted data). However, when privacy requirements are considered, the

sensitive real-time data can be made available and accessed only in encrypted databases. Hence, we must apply our learned model to encrypted data. Theoretically, there are so-called fully homomorphic encryption (*FHE*) techniques that allow us to perform any kind of efficiently computable operation (addition *and* multiplication) on encrypted data [Gentry, 2009], so while we can't read the encrypted data, we can still evaluate a function on it, and decrypting the return value yields the same result as performing the same function on plaintext. However, these techniques are not yet feasible for large problems. For example, a recent implementation by Gentry et al. [2012] takes around four minutes using 3GB RAM to evaluate an AES-128 encryption operation, a time span that renders operations on entire databases impractical. Operations on encrypted data become feasible if we only allow certain operations (e.g., addition *or* multiplication) on encrypted data. In this case, either partially homomorphic encryption or order-preserving encryption techniques can be employed [Popa et al., 2011, Agrawal et al., 2004]. Partially homomorphic encryption techniques allow only certain operations (addition *or* multiplication) to be performed on encrypted data, while order-preserving encryption allows comparison operations to be performed on encrypted data.

Table 2.2: Required encryption technique for privacy-preserving classification via different machine learning (ML) techniques

ML Technique	classification function [Murphy, 2012]	Required operations	Required Encryption
SVM	$w_0 + \boldsymbol{w}^T\boldsymbol{x}$	$\{+, *\}$	FHE
ANN	$sigm(\boldsymbol{w}^T\boldsymbol{z}(\boldsymbol{x}))$	$\{+, *, exp\}$	FHE
KNN	$dist(\boldsymbol{x}, \boldsymbol{d})$	$\{+, *, sqrt\}$	not available
DT	$\phi(\boldsymbol{x}, \boldsymbol{v}_l)$	$\{<, >, =\}$	order-preserving encryption
LOGIT	$sigm(\boldsymbol{w}^T\boldsymbol{x})$	$\{+, *, exp\}$	FHE
LINREG	$w_0 + \boldsymbol{w}^T\boldsymbol{x}$	$\{+, *\}$	FHE

In order to determine which type of encryption is required for each machine learning technique, we analyze their individual classification functions and assess which arithmetic operations need to be performed on encrypted data. We determine the required arithmetic operation and link each of the techniques with the required encryption technique for a privacy-preserving implementation of the classification function. Table 2.2 lists the classification function of each machine learning technique based on Murphy's 2012 notation. $\boldsymbol{w}^T\boldsymbol{x}$ denotes the scalar product of a transposed vector of weights $\boldsymbol{w}$ and a vector of attribute values $\boldsymbol{x}$. Computing a scalar product requires addition and multiplication since:

$$\boldsymbol{w}^T\boldsymbol{x} = \sum_{j=1}^{|\boldsymbol{x}|} w_j x_j \tag{2.1}$$

where $|\boldsymbol{x}|$ is the length of the vector $\boldsymbol{x}$. For privacy-preserving classification with *SVM* we need to compute the scalar product of encrypted vectors. Hence, classification via *SVM* is possible only on fully homomorphic encrypted data that enable addition and multiplication operations. In addition to this implementation issue, *SVM* has a drawback with respect to demand forecasting, as it cannot generate probabilistic output (i.e., a distribution over y), so it falls short in providing additional useful information about the reliability of an aggregated forecast from multiple classifications of individual components [Murphy, 2012].

LOGIT requires the computation of a scalar's *sigmoid* function:

$$sigm(\nu) := \frac{1}{1 + exp(-\nu)} \tag{2.2}$$

Since exponentiation can be rewritten as repeated multiplications, *LOGIT* requires both addition and multiplication. As in the case of *SVM*, a privacy-preserving implementation could only be realized on fully homomorphic encrypted data.

For classification via *ANN*, we first apply a vector-valued function $\boldsymbol{z}(\boldsymbol{x}) := g(\boldsymbol{V}\boldsymbol{x})$ on the attribute vector $\boldsymbol{x}$. The non-linear link function g is commonly the sigmoid function (also known as logistic or logit function), $g(u) = sigm(u)$.

Thus, classification via *ANN* also requires fully-homomorphic encryption of the data.

For classification via *KNN* we need to evaluate the distance between an instance $\boldsymbol{x}$ and the K nearest neighbors. Typically, the Euclidean distance is used, so we need to evaluate

$$dist(\boldsymbol{x}, \boldsymbol{d}) = \sqrt{\sum_{j=1}^{|\boldsymbol{x}|}(x_j - d_j)^2} \tag{2.3}$$

on encrypted data. Since there is not yet an efficient way of computing square roots on even fully homomorphic encrypted data, *KNN* is not an option for privacy-preserving classification.

For classification via *DT*, comparing the attribute values of $\boldsymbol{x}$ with split values given by $\boldsymbol{v}_l$ is sufficient. The indicator function $\phi(\boldsymbol{x}, \boldsymbol{v}_l)$ yields 1, if $\boldsymbol{x}$ belongs to leaf l and 0 otherwise. Hence, only comparison operations need to be executed on encrypted data, and *DT* requires only order-preserving encryption.

For privacy-preserving classification with approaches that build on *LINREG*, we need to compute the scalar product of encrypted vectors which requires addition AND multiplication of encrypted data. This however would require fully homomorphic encryption which is not practical in realistic settings.

Given the status quo in cryptography, only the classification with decision trees is feasible for use on encrypted data. Although this seems to be a very strong limitation, *(boosted-DT)*, *random forests (RF)* and *bagged decision trees (bagged-DT)* are based on the concept of *DT* and can be implemented using order-preserving encryption. Therefore, a number of techniques that have proven to be effective (e.g., Caruana and Niculescu-Mizil [2006]) lend themselves to a privacy-preserving implementation. None of the other techniques can currently be implemented in a privacy-preserving manner if we impose realistic practical requirements in terms of run-time performance. In the next section, we describe an implementation for privacy-preserving condition-based forecasting that combines *DT* learning with order-preserving

encrypted databases.

2.4 Privacy-preserving forecasting with decision trees

We now briefly describe the underlying idea of our approach, depicted in Figure 2.2, before we provide further details in the subsequent subsections. Decision trees can classify an instance by simple comparison operations. Order-preserving encryption allows comparison operations to be performed on encrypted data. Therefore, the sensitive real-time data can be stored safely in encrypted databases in one or more cloud systems. We use a separate database for each customer, each one with a distinct encryption key in order to fulfill privacy requirements, and only the customer has the key to access and decrypt his data. Nonetheless, we can perform the necessary computation on the encrypted data in order to classify each individual machine or component. Since the MRO does not need to know the result of classifying individual parts, we aggregate across multiple parts of the same type from different customers by applying a privacy-preserving aggregation protocol that returns the sum of the individual sensitive classification results from each customer without revealing any of the inputs to a third party. This approach adds an additional layer of security for the customers, as neither the individual machines' attribute values nor the forecasts for individual customers are revealed to a third party. Only aggregate classification results, which the MRO can't relate back to individual components, will be made available to the MRO. From these aggregate classification results we can provide the MRO with the necessary forecast information, that is, expected demand for the planning period and a measure of the forecast's reliability.

We can break our approach into six steps: In step 1 the MRO learns a probabilistic – meaning that the return value is not just the most likely class (replace: "yes" or "no") but also the estimated probability for replacement $\boldsymbol{f_D}(\boldsymbol{x}) = p(replace|\boldsymbol{x})$ given a certain condition vector $\boldsymbol{x}$ – decision tree model $\boldsymbol{f_D}(\boldsymbol{x})$ on a given learning dataset $\boldsymbol{D}$.

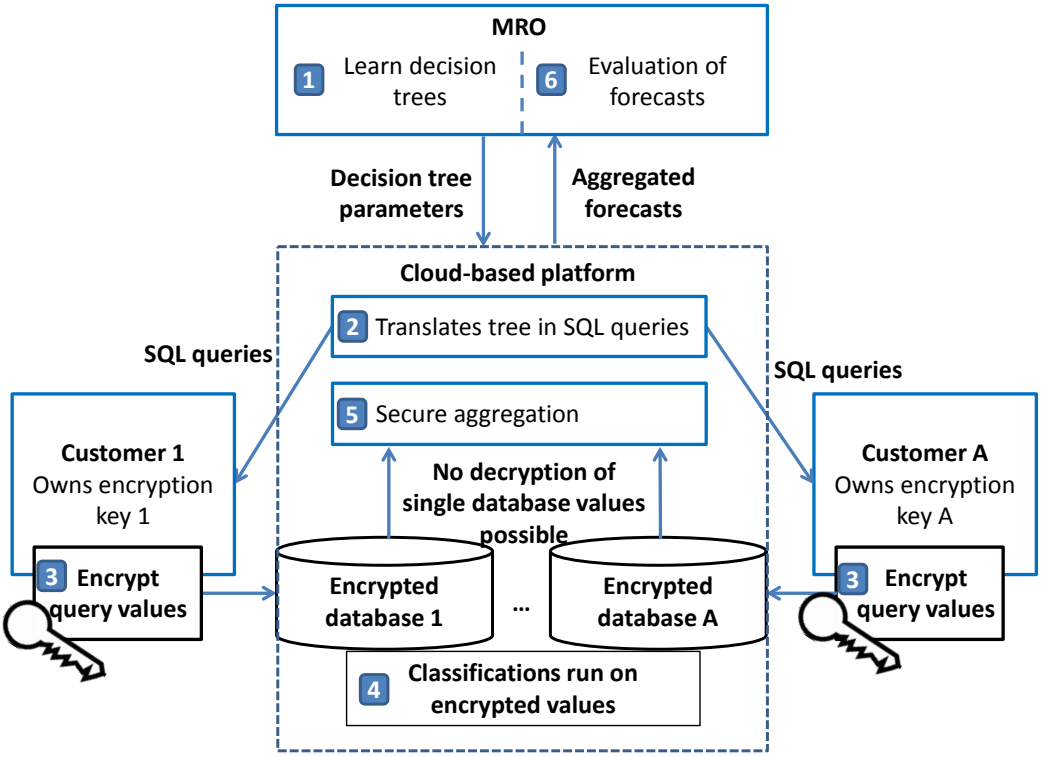


Figure 2.2: Concept for privacy-preserving condition-based forecasting with order-preserving encryption

In step 2 the decision tree is translated into basic SQL queries. In step 3 these SQL queries are encrypted so they can be executed on the encrypted databases that contain the various customers' sensitive real-time data. We refer to this encrypted classification function as $\boldsymbol{f}_D^{encr}(\boldsymbol{x})$. In step 4 we apply a part of the classification function on the encrypted data in order to obtain the number of instances of customer a that are classified in each leaf $l = 1, ...L$. In step 5 we sum these results over all A customers via a privacy-preserving aggregation protocol. The MRO (as well as any other potentially malicious party) cannot infer anything about the underlying condition data from these aggregated results. However, in step 6, the MRO obtains an estimate of the expected demand and a measure of the forecast's reliability. In the following sections we describe these six individual steps in detail and formalize our approach.

2.4.1 Probabilistic *DT* learning for bottom-up demand forecasting

We now turn to explaining the basic concept of probabilistic decision trees and how we use them for condition-based forecasting of aggregate demand (step 1 in Figure 2.2). In *DT* learning, the functional relationships between condition data and the need for replacement is represented in a tree-like structure, as depicted in Figure 2.3. Given such a tree we classify one instance by comparing its attribute values, which describe its current condition, with the values of the tree nodes following the matching branches from the top node – i.e., the root – to a leaf node. The probability that is attached to each leaf is the estimate for the probability that a component that is classified in this leaf needs to be replaced. More formally, we apply the following function as denoted by Murphy [2012]:

$$\boldsymbol{f_D}(\boldsymbol{x}) = p(replace|\boldsymbol{x}) = \sum_{l=1}^{L} \pi_l \phi(\boldsymbol{x}, \boldsymbol{v}_l) \tag{2.4}$$

where π_l is the replacement probability attached to leaf l, $\boldsymbol{v}_l$ is the matrix that defines the path from the root to leaf l and $\phi()$ is an indicator function that yields 1 if the attribute values of an instance $\boldsymbol{x}$ fulfill the criteria of the leaf described by $\boldsymbol{v}_l$ and 0 if not. Consider, for example, the leaf determined by the matrix $\boldsymbol{v}_2 = \begin{pmatrix} \#cycles & < & 72 \\ vibrationlevel & \geq & 2 \\ \#cycles & < & 66 \end{pmatrix}$ with the attached probability $\pi_2 = 0.2$.

Hence, $\phi(\boldsymbol{x}, \boldsymbol{v}_2) = 1$ if and only if the inequalities in $\boldsymbol{v}_2$ hold for the attribute values of $\boldsymbol{x}$ and 0 otherwise.

To clarify how the replacement probabilities π_l are computed, we must describe the learning phase of decision tree learning.[3]. What most learning algorithms have in common is that they use a greedy approach and build a

[3]For a more general introduction the reader is referred to [Quinlan, 1986, Rokach and Maimon, 2008, Murphy, 2012, Lomax and Vadera, 2013]

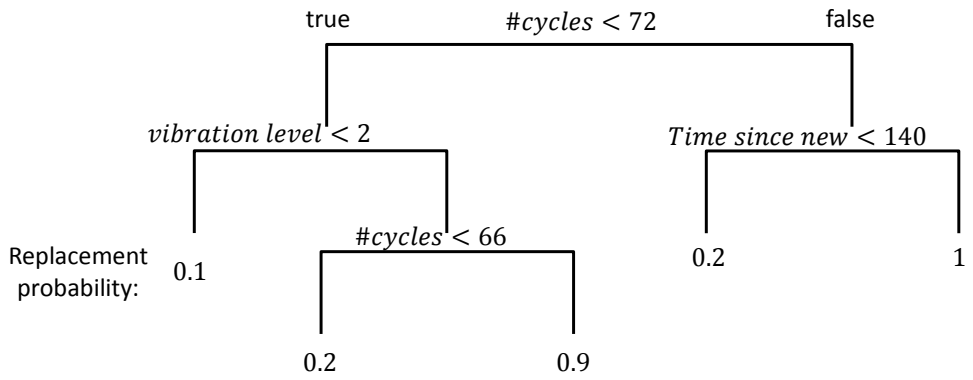


Figure 2.3: Example of a probabilistic binary decision tree of depth 3

tree by recursively partitioning the learning dataset – that is, they start with the full set and find the attribute, along with a split value that allows the best split of a dataset in two disjoint parts. The best split is the one that best assigns instances with similar class attributes to both subsets. Therefore, we compute the ratio of instances of one class in a (sub-)set $\boldsymbol{D}$ by:

$$\pi_{\boldsymbol{D}} = \frac{\sum_{(\boldsymbol{x},y)\in\boldsymbol{D}} \mathbb{1}(y = yes)}{|\boldsymbol{D}|} \tag{2.5}$$

Technically speaking, a split function determines the attribute, along with the best split value for this attribute, by minimizing the sum over the deviance in the separated parts [Murphy, 2012]. The deviance is a measure of the purity of the data in a node using $\pi_{\boldsymbol{D}}$ as an argument. It is 0 if all instances in this node belong to the same class and 1 if half the instances belong to each of the two possible classes. Finding the best split and splitting the learning data accordingly is continued recursively in both of the child nodes until a stopping criterion is reached. Typical stopping criteria are the depth of the tree, the number of instances that are available in a certain node, and the potential improvement achievable in the next split.

These stopping criteria are necessary to avoid overfitting of the tree to

the learning data; that is, the model would perform well on the learning data but much worse on new data even if it differs only slightly from the learning data. Another way to avoid overfitting is to use only a sample of the learning data for the learning phase and the remaining instances to test the model. We then obtain a realistic measure for the quality of the learned tree. This classification of test instances also gives us the probabilities we attach to each leaf of the tree. For each leaf, we use the ratio of instances that needed to be replaced. Let $\boldsymbol{T}_l$ be the set of instances from the test set classified in leaf l. Then the replacement probability π_l we attach to leaf l is given by $\pi_l := \pi_{\boldsymbol{T}_l}$.

If a new instance is then classified in a certain leaf l we interpret π_l as the estimated probability that this instance needs to be replaced. A perfect split is given if only instances of the same class are assigned to each subset of the split learning data. If, for example, a split along one attribute sorts all instances of the learning dataset that needed replacement into the left sub-node and no such instance to the right sub-node, then the estimated probability for an instance that is classified in the left leaf would be 1 (respectively 0 for the right leaf) [Murphy, 2012]. While we describe just the case of a binary decision problem with only two possible classes (replace: yes/no), we note that we could also apply decision tree learning to ordinal N-class classification problems, e.g. by dividing the problem in $N-1$ binary classification problems as proposed by Frank and Hall [2001].

The MRO is able to learn a tree, as depicted in Figure 2.3 for each type of spare part. The tree can be used to classify a new instance – that is, to estimate a probability for replacement given only the current attribute values that represent the condition of a component. However, the MRO wants to obtain an estimate for the overall demand for spare parts of a certain type. Therefore, we compute:

$$F_{spare} = \sum_{a=1}^{A} \sum_{\boldsymbol{x} \in S_a} \boldsymbol{f}_{\boldsymbol{D}}(\boldsymbol{x}) = \sum_{a=1}^{A} \sum_{\boldsymbol{x} \in S_a} \sum_{l=1}^{L} \pi_l \phi(\boldsymbol{x}, \boldsymbol{v}_l) \tag{2.6}$$

Thereby, we classify all instances that are currently in use by the different customers $a \in \{1, ..., A\}$ and for which the condition data is stored in the

respective databases, denoted by S_a. As an alternative that provides more information than just the aggregate forecast, we obtain the number of instances that are classified in a certain leaf l by:

$$\bar{\pi}_l = \sum_{a=1}^{A} \sum_{\boldsymbol{x} \in S_a} \phi(\boldsymbol{x}, \boldsymbol{v}_l) \tag{2.7}$$

Since the leaves differ in terms of purity (i.e. in the values of π_l), knowing how many instances are classified in each leaf gives us valuable information about the forecast's reliability. However, before we describe in Section 2.4.4 how we can use this information, we show how we apply DT learning on encrypted data.

2.4.2 Privacy-preserving classification of distributed data

This subsection details steps 2, 3 and 4 in Figure 2.2 and describes our approach for privacy-preserving computation of the demand forecasts from order-preserving encrypted real-time data. Subsection 2.4.2 provides results from performance tests with our prototype application.

A combination of order-preserving encryption and decision trees

In the classification phase privacy requirements are high since the real-time datasets of different (competing) customers are to be classified. In the aerospace example these datasets include current flight plans and real-time condition data. This is sensitive data, not only for passenger or cargo airlines, but more so for national air forces. In this scenario, these customers will only provide their data to a cloud-based system if encryption prohibits data leakage. Therefore, we store the customers' data in encrypted databases using an order-preserving encryption scheme (OPES). The concept of OPES was introduced by Agrawal et al. [2004] to allow the efficient processing of range queries on encrypted data.

For our application, data is encrypted locally by each customer and then sent to a cloud-based system. This system is referred to as a relaxed interface,

through which Popa et al. [2013] showed that ideal security can be achieved.

The main advantage of decision-tree classification comes into effect in combination with OPES. In the classification phase the computational requirements are rather low since only comparison operations $\{>; <; =\}$ are necessary to classify an instance. OPES allows us to perform exactly these comparison operations on encrypted data, satisfying:

$$encr_k(u) < encr_k(v) \Leftrightarrow u < v,$$

where $encr_k(u)$ represents the encryption of data u with encryption key k.

The main features of OPES are that (1) ordering and comparison operations can be directly applied to encrypted data, including equality and range queries, MAX, MIN, and COUNT queries; (2) mutability is given, which means that a small number of ciphertexts of already encrypted values can change as new plaintext values are encrypted [Popa et al., 2013]; and (3) time and space requirements are reasonable enough to allow deployment for large problem instances [Agrawal et al., 2004, Popa et al., 2011].

Based on the work of Popa et al. [2011], we present an architecture that avoids the need for a trusted database server, as the customers remain responsible for rewriting the queries, and no decryption of query results is necessary. This architecture is of great importance since the lack of trust in a third party (e.g., a cloud provider) has been identified as a major obstacle [Barratt, 2004, Barkataki and Zeineddine, 2015], and with our application the MRO and the customers can collaborate without necessarily trusting each other.

Comparison operations are sufficient to classify an instance by comparing its attribute values with all splits of a given decision tree. The COUNT operation then allows us to obtain the number of instances in each leaf and to compute the replacement probability. For illustration purposes, consider the sample database in Table 2.3, which contains encrypted real-time condition data for one type of part.

In order to classify this data, we need to translate the tree that the MRO sends to a cloud-based platform into SQL queries. For each leaf of the tree there is one query that counts how many machines or components fall into

this leaf. Consider one leaf as an example. The classification function that returns the number of instances that belong to the leaf from all A customers is given by:

$$\bar{\pi}_{2,a} = \sum_{\boldsymbol{x} \in S_a} \phi(\boldsymbol{x}, \boldsymbol{v}_2) \tag{2.8}$$

The corresponding SQL query for a plaintext database S_a could be written as:

```
SELECT COUNT (*)
FROM  S_a
WHERE #cycles ≤ 72
AND   vibration level < 2
```

This query returns the number of instances that are assigned to the leaf with an associated probability for necessary replacement of 0.1. However, this query does not work if it is sent to an encrypted database, as shown in Table 2.3, where not only the attribute values but also the attribute names are encrypted. Hence, even if the MRO or the cloud provider tried to access the encrypted database via SQL queries, no meaningful results could be retrieved.

Table 2.3: Example of an encrypted real-time dataset of customer a named S_a^{encr}

$encr_k(operating\,hours)$	...	$encr_k(vibration\,level)$	$encr_k(\textbf{replace})$
$encr_k(123)$	...	$encr_k(2)$	
$encr_k(239)$	...	$encr_k(2)$	
⋮		⋮	
$encr_k(171)$	...	$encr_k(1)$	

Table 2.3 contains encrypted real-time condition data for one type of part that was encrypted by one customer using its private key k. This data can also be classified with a corresponding tree like the one depicted in Figure 2.3. However, we first need to encrypt the corresponding SQL query in order to execute the query on the encrypted databases. In SQL notation this results in:

SELECT COUNT	$(*)$
FROM	S_a^{encr}
WHERE	$encr_a(\#\text{ cycles}) \leq encr_a(72)$
AND	$encr_a(vibration\,level) < encr_a(2)$

Similar to the unencrypted query, this query also returns the number of engines which are assigned to the leaf with an associated probability of necessary replacement of 0.1. All other leaves can also be evaluated through similar queries. However, we must run these kind of queries not only for each type of component but also on each of the customers' databases. For the MRO which instances in a certain leaf come from which customer is irrelevant. On the other hand, the customers consider the probability of replacement as sensitive information that must not be leaked to competitors. Subsequent to the following performance test, the next subsection describes how the aggregated numbers of instances in each leaf of each customer can be computed without revealing these numbers to any other party.

Performance test

We implemented the approach outlined in the previous sections in a prototype application. In this section we show that our prototype performs well on the most critical tasks and therefore avoids runtime issues that can hamper the feasibility of privacy-preserving solutions.

We first analyze two important indicators: 1) The time it takes to encrypt a dataset initially and 2) how our approach scales with the number of leaves. For this we use a dataset containing sixteen attributes that describe the condition of a propulsion plant [Coraddu et al., 2016]. The dataset contains $5,967$ lines, each representing one instance of a similar type of propulsion plant currently in use by one customer. We then compare our approach with the approach introduced in Wu et al. [2016], which is, so far, the best performing approach that can be applied to similar settings and tasks as our approach. To make our results more generalizable and robust, we use two additional real-world datasets that are also used by Wu et al. [2016].

For all datasets the test environment for the initial encryption was a laptop with 8GB RAM and an i5-3320M processor running on Windows 7. The classification of the order-preserving encrypted data was performed using SAP HANA infrastructure with 256GB RAM and a quadruple XEON E5-2670 processor running on Linux Suse. This process emulates a practical situation in which customers encrypt their data locally, while the privacy-preserving classification may run on more powerful infrastructure in the cloud.

For our first analysis the initial encryption of all $5,967$ instances took $545,216$ms (approximately 9 minutes). Time for encryption is proportional to the number of instances, so updating the encrypted database by inserting a single instance takes about 91ms. We consider this as more than acceptable for a practical implementation, especially because the initial encryption is a one-time task and 9 minutes for a realistically sized dataset is a reasonable runtime.

The main driver of computation time for our approach is the number of leaves that need to be evaluated. In Figure 2.4 we show the average time for classifying a single instance from a sample of 100 instances with different numbers of leaves. We see that the computation time increases approximately linearly with the number of leaves. We consider this as not being critical because the number of terminal nodes typically does not exceed 64, which is the maximum number of leaves of a tree with depth 6. We note that the number of instances and the depth of the tree is also not critical for our approach due to the efficient implementation of comparison operations. It takes just three times longer to classify all $5,967$ instances than to classify only 100 instances. Furthermore, we do not observe a correlation between the number of necessary comparison operations, i.e., the depth of the tree. The reason for this robustness against the number of instances and the depth of the tree is that we break down the evaluation to standard queries on the order-preserving encrypted database, which are highly optimized in the HANA infrastructure. This makes our approach especially well suited for settings where a large number of instances needs to be evaluated frequently by complex trees.

Wu et al. [2016] propose an alternative approach for secure evaluation

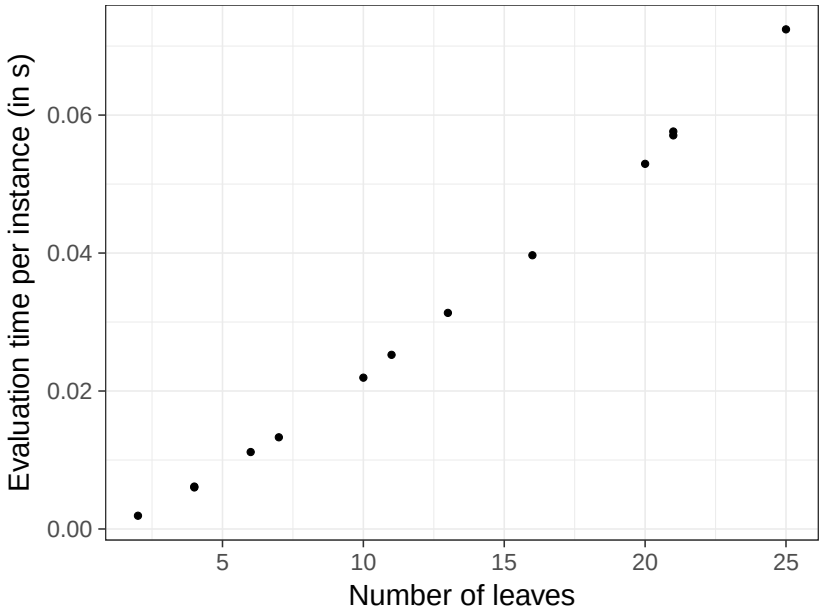


Figure 2.4: Evaluation time for different trees with different numbers of terminal nodes (leaves)

of decision trees. For comparing our new approach with Wu et al. [2016] we use two additional real-world datasets of different size and dimensionality that are also used in Wu et al. [2016] for performance testing. While these datasets are not related to condition-based maintenance, they still require basic classification tasks. In the first dataset (spambase) the task is to decide whether a mail is spam or not, given a set of 57 features. The second dataset has 9 features that are used to classify whether a patient has breast cancer.

We follow the approach of Wu et al. [2016] to compare the performance of our method with the method introduced in Wu et al. [2016] The results are reported in Table 4.1. We observe that our approach is about 50 times faster for both datasets compared to the approach of Wu et al. [2016]. The runtimes we report for our approach are average evaluation times for a sample of 100 instances. Since our approach is very robust to the number of instances we could achieve much lower average evaluation times per instance for larger samples. The computation time for the approach of Wu et al. [2016] increases linearly in the number of instances. Therefore, our approach is much faster when larger

data sets need to be evaluated. However, we note that the approach of Wu et al. [2016] has an advantage in terms of privacy since it does not only keep the feature values of the customers' data private, but also does not reveal the split values of the tree. Therefore, in some settings where the service provider considers the tree as sensitive information s/he can trade off the additional privacy of Wu et al. [2016] with the significantly better performance of our approach.

			Runtime (in s)	
Dataset	Features	Leaves	Our approach	Wu et al. [2016]
Breast cancer	9	12	**0.024**	**0.545**
Spambase	57	58	**0.38**	**16**

Table 2.4: Comparison of computational efficiency with Wu et al. [2016]

2.4.3 Privacy-preserving aggregation protocol

The classification based on a decision tree returns for each leaf $l \in \{1, \cdots, L\}$ and each customer $a \in \{1, \cdots, A\}$ the number $\bar{\pi}_{l,a}$ of engines that fall into this leaf. For the customers $\bar{\pi}_{l,a}$ is sensitive data because it reveals information about the condition of individual machines and components. Hence, this value should not be disclosed to other customers or to the MRO. The MRO does not need the individual data of each customer but the aggregated numbers to make decisions about the spare parts inventory. Hence, the MRO needs only the sum $\bar{\pi}_{l,1} + \cdots + \bar{\pi}_{l,A}$. This subsection implements step 5 of Figure 2.2 by showing how this aggregated result can be computed without disclosing the individual input data.

We solve the problem by using the concept of SMC. A SMC protocol allows mutually suspicious parties to compute a function on their private inputs without revealing anything but the function's output. We illustrate the basic idea of SMC with a simple example that is related to our specific setting [Schneier, 1995]: Say we have three customers CUS_1, CUS_2 and CUS_3 each has a number x_1, x_2 and x_3 as (private) input. They want to compute

$x = x_1 + x_2 + x_3$, but they do not want to disclose their private data to each other. CUS_1 conceals his number by adding a random number r and privately sends $r + x_1$ to CUS_2. CUS_2 adds his input and privately sends $r + x_1 + x_2$ to CUS_3. CUS_3 adds his input and sends $r + x_1 + x_2 + x_3$ back to CUS_1. CUS_1 recalls r, subtracts it from the received value $r + x_1 + x_2 + x_3$ and announces $x = x_1 + x_2 + x_3$. None of the involved parties is able to learn the input of the other parties. CUS_2 and CUS_3 are blinded by the random choice r of CUS_1, and CUS_1 can only learn the sum $x_2 + x_3$ but no individual input.

As in the preceding example, most existing SMC protocols assume secure and authenticated communication channels between each pair of parties. However, for our privacy-preserving aggregation problem, this assumption is not realistic because, in a real-world scenario, customer a does not necessarily know who customer $a + 1$ is, and the input values for all customers are not necessarily available at the same time. Therefore, we follow a central server approach, as Kerschbaum [2009] does, where the server computes the result of the function. The customers do not need the result of the computation and will be treated as input parties, which implies a producer-consumer model, as in Jawurek and Kerschbaum [2012], with the central server as aggregator in between. The MRO is the only party that needs to know the final result.

However, this central server does not have to be a singular cloud provider. Within our central cloud-based platform we can use the infrastructures of various cloud providers to enhance privacy and avoid single points of failure. Therefore, we propose a secure aggregation protocol based on a secret-sharing scheme (SSS). With SSS, the role of the central server is split among multiple cloud applications (APPs) that are hosted by various cloud providers. Each of these APPs receives only partial information from each customer. Then each APP aggregates the partial inputs, which are meaningless in themselves, and sends the aggregation to the MRO. Only the MRO can aggregate the parts and obtain the overall solution.

A secret sharing scheme consists of two algorithms: one specifies how to *share* the secret s in *shares*, and the other shows how to *reconstruct* s from the shares or a subset thereof. For our application an important property of SSS is *linearity* since *linearity* allows us to sum the shares of multiple

secrets. In a linear scheme, a secret s is viewed as an element of a finite field, and the shares are obtained by applying a linear mapping to the secret and several independent random field elements [Beimel, 2011]. An example is the so called $(t, n)-$*threshold* scheme, which assumes that, among the APPs that receive the secret shares, at most t parties can fail to deliver a result or cooperate to defeat the protocol, where $t < n$. If at least $t + 1$ valid shares are available, the secret can be reconstructed. Therefore, a subsets of APPs with cardinality at least $t + 1$ is referred to as authorized. Assume that s is a number (like our individual forecasts) and there are three parties APP_1, APP_2 and APP_3. Then a simple way to share s consists in randomly choosing two numbers r_1 and r_2 in $\mathbb{Z}_p$, with p prime, and setting $r_3 = s - r_1 - r_2 \mod p$. Now each party APP_i receives r_i and they just have to sum all their shares to reconstruct s. If two secrets s_1, s_2 have been shared in r_{1i}, r_{2i} then the parties can securely compute the sum of the secrets $s_1 + s_2$ by privately adding their shares and broadcasting the result. In our example, each APP_i sends the sum of the shares to the MRO, which cannot deduce anything about the customers' individual inputs but can sum the added shares from all APP_i and can, therefore, obtain the final result. In general, if $t < n/2$, then SMC with perfect security is possible [Cramer et al., 2015], which implies $n >= 3$. For more information on more complex linear SSS and how they are used in SMC the reader is referred to Beimel [2011], Cramer et al. [2015].

For our purposes, we propose a solution where each customer uses a linear $(t, n)-$threshold SSS to split its input among $n = 2t + 1$ APPs, which jointly compute the overall result. Only the MRO should be able to reconstruct the result, which requires a secure channel between each APP and the MRO. The privacy-preserving aggregation protocol is illustrated in Figure 2.5 for $n = 3$, where APP_1, APP_2 and APP_3 run, for example, on Amazon Web Services, Google Cloud Platform, and Microsoft's Azure. Customer a splits $\bar{\pi}_{l,a}$ in three shares $\bar{\pi}_{l,a,1}, \bar{\pi}_{l,a,2}, \bar{\pi}_{l,a,3}$ and sends $\bar{\pi}_{l,a,j}$ to APP_j for $j = 1, 2, 3$ through a secure channel. Since $\bar{\pi}_{l,a,j}$ is just one share, it does not reveal any information about $\bar{\pi}_{l,a}$. Then each APP adds the shares received from its customers, and after all customers have sent their inputs, the APPs send their individual results to the MRO, which can determine the final result. The properties of linear

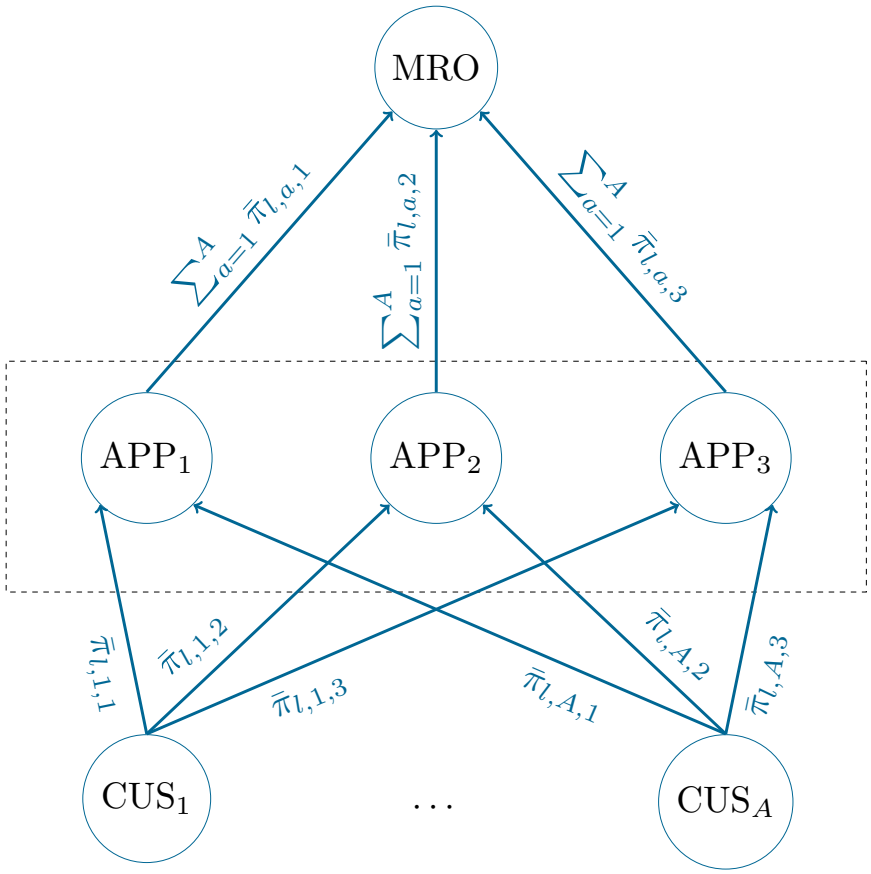


Figure 2.5: Overview of the privacy-preserving aggregation protocol

$(t, n)-$threshold SSS ensure that, even if at most t (in our example $t = 1$) APPs fail to send their result, the MRO will still be able to reconstruct the result of the aggregation. Assuming that the APPs behave semi-honestly[4] and that at most t APPs can fail or cooperate, our protocol is not only privacy-preserving but also fault-tolerant, because the result of at least $t + 1$ APPs suffice to reconstruct the aggregated result. Furthermore, a malicious MRO would have to corrupt more than t APPs to obtain the customers' private inputs. Protocol 1 is the core of our solution for a secure aggregation of the

[4]In SMC semi-honest (also called honest-but-curious) parties follow the protocol specification. In our setting this means that APPs won't willingly corrupt or block the transmission of shares.

numbers per leaf from the decentralized classification of multiple encrypted customer databases.

1: CUS_a splits $\bar{\pi}_{l,a}$ in n shares $\bar{\pi}_{l,a,1}, \cdots, \bar{\pi}_{l,a,n}$
2: CUS_a sends $\bar{\pi}_{l,a,j}$ to APP_j over a secure channel, $j = 1, \cdots, n$
3: APP_j computes $S_{a,j} = S_{a-1,j} + \bar{\pi}_{l,a,j}$,
where $S_{0,j} = 0$ and $S_{a-1,j} = \bar{\pi}_{l,1,j} + \cdots + \bar{\pi}_{l,a-1,j}$
4: After all customers have sent their inputs each APP_j sends
$S_{A,j} = \sum_{a=1}^{A} \bar{\pi}_{l,a,j}$ to the MRO
5: If at least $t + 1$ APPs send their result, MRO can reconstruct the overall result

Protocol 1: privacy-preserving aggregation using a secret sharing scheme

As depicted in Figure 2.2, the MRO gives its decision tree as input to the system and receives as output the result of the privacy-preserving aggregation, which is the number of engines that fall into a given leaf. With this aggregated number the MRO can determine the single forecast and an estimate of its reliability. The next section outlines how the latter can be obtained.

2.4.4 Estimation of forecast reliability

In the previous section we developed an approach that allows us to aggregate the numbers of instances per leaf from all customers without revealing any of the individual values. We now describe how the MRO can use this information to determine a single forecast value and an estimate of this forecast's reliability. This description refers to the final step, step 6, in Figure 2.2. Even though an estimate for the reliability may at first seem like a marginal detail, it is an important feature of our approach because based on the aggregated forecast, the MRO will devise inventory policies for different spare parts. Any state-of-the-art inventory policy requires not only a single valued (point) forecast, but also some distributional information, such as that to compute safety buffers. Therefore, this information is critical to subsequent planning tasks.

We build on the fact that our model performs differently in different regions. (That is, the leaves with attached probabilities that are close to 1 or 0 provide more reliable predictions than do leaves with probabilities that are closer to 0.5. We can leverage the information about how many instances fall

to each leaf to obtain an ex-post distribution of aggregated demand. This condition-based forecast uncertainty we differs from the classical approach, where a single static error distribution is obtained from historic data.

To obtain a measure for the forecast's reliability, we use the output from the preceding privacy-preserving aggregation step, where the MRO obtains the numbers of instances that are classified in each leaf, aggregated over all customers. We can write this information as a vector $\bar{\pi} = (\mathbf{f_D}(\mathbf{x}^1), ..., \mathbf{f_D}(\mathbf{x}^N))$ where $\bar{\pi}_i$ is the replacement probability of the leaf instance to which $\mathbf{x}^i$ was assigned. N is the number of instances – that is, the number of parts of this type that are currently in use.

As result of the privacy-preserving aggregation step, the MRO obtains the numbers of instances that are classified in each leaf, aggregated over all customers. Since the MRO knows the replacement probability for each leaf, computation of the weighted average over all leaves provides a point forecast of the instances to be replaced. In order to obtain a measure of this forecast's reliability, we derive a probability distribution for the number of instances that need to be replaced. Therefore, we start with a vector $\bar{\pi}$; the length of $\bar{\pi}$ corresponds to the number of all classified instances. Each entry of $\bar{\pi}$ represents one instance and contains the replacement probability of the leaf to which the instance was assigned. Algorithm 3 in Appendix A takes a vector with the individual replacement probabilities $\bar{\pi}$ and returns a distribution $\mathbf{p}$ with probabilities for 0, 1, ... N replacements.

This process provides us with valuable information for subsequent inventory decisions. We can obtain a measure for the reliability of the forecast by, for example, calculating the distribution's standard deviation. This measure changes depending on the current condition data. Measuring the reliability of the forecast is independent of the machine learning technique that was applied for classification. The only requirement is that it is probabilistic – that is, that it returns replacement probabilities.[5]

[5]Except for support vector machines, all techniques discussed in Section 2.3 satisfy this requirement

2.5 Conclusion and direction for further research

Although condition-based maintenance is a well-established concept for individual machines, research on integrating condition data that is created by multiple machines run by several customers of one MRO in order to forecast aggregate maintenance demand is scarce. In this context, the privacy concerns of customers that are not willing to reveal their sensitive real-time data become an issue that may hamper the use of data needed any collaborative forecasting approach. We overcome this issue and thereby enable collaborative condition-based forecasting in service supply chains. With our approach, sensitive data from different companies can be used to obtain combined condition-based forecasts without the need to reveal any of the input data. This may lead to substantial benefits such as more accurate spare parts demand forecasts for the MRO which subsequently would lead to reduced service times for the customers and lower inventory costs for the MRO. In addition to a point forecast, we show how to obtain information about its reliability, which allows the MRO to decide ex ante whether the subsequent inventory decisions should be based on the decision-tree forecast or on a previously used method that does not use real-time condition data.

Our work was motivated by the case of an MRO in the aerospace industry; one of the important and challenging tasks of this company is to determine the right inventory levels for different spare parts. The applicability of our approach, however, is not limited to inventory management and there may be applications of privacy-preserving condition-based forecasting with an even stronger monetary impact: Frequently, the machinery is not sent to the MRO as in the case of spare parts for aircrafts, but the the MRO employs expensive technicians to carry out on-site maintenance. Oftentimes, costs of the service technicians exceed the costs of the spare parts. Examples are the overhaul of complex medical machinery or wind turbines that are installed all over the world but maintained by SMEs with one or few sites. In these kind of settings, additional substantial costs can arise because service teams are frequently sent out either too early, that is, when overhaul is not yet necessary, or too late, after a break-down occurred. Also, the service team may take

significantly longer to examine the machinery without any prior information about its condition and/or the right spare parts in place. These excessive costs can be reduced if privacy-preserving approaches enable the use of sensitive information for condition-based forecasting. The information we generate with our approach can, for example, be used to better schedule maintenance teams, to provide them with better information and the right spare parts before arriving at the customer's site and to prioritize and plan maintenance jobs more accurately to prevent down-times and/or unnecessary maintenance operations.

Linking our condition-based forecast with a subsequent inventory policy could be a useful field for further research. Following the myopic heuristic Lin et al. [2012] introduce is one alternative since our approach provides the required probability distribution for the demand. Another alternative is provided by the dynamic re-order point policy Babaï et al. [2009] introduce. A dynamic policy is necessary because we cannot assume stationarity of demand, as demand depends on the condition of the machinery. Exploring how to best use even more advanced machine learning techniques, such as RFs, in a privacy-preserving manner would also be a useful area of inquiry. Ultimately, we want to develop a toolbox of methods from which the user can choose depending on the individual requirements and data structure.

In our work we considered a setting in which the time for a component's overhaul is given (typically by thresholds determined by the manufacturer) and the need for replacement is predicted using condition data. To schedule overhaul operations pro-actively based on sensitive condition data has significant potential for increasing the usefulness of machinery while also reducing the number of unexpected failures [Deloux et al., 2009]. However, this step from hard-time to condition-based overhaul requires high-quality data and a reliable performance of the prediction models, as the consequences of errors are more severe. Therefore, our solution gives an MRO a suitable test bed for condition-based maintenance that offers significant potential for improvements and that, if implemented successfully, can serve as a showcase to convince customers to delegate more responsibility for maintenance decisions to the MRO.

Acknowledgements

Special thanks goes to Florian Hahn, Product Security Research, SAP SE, for his support with the performance tests.

3 Prescriptive analytics for inventory management: a comparison of new approaches

We analyze the performance drivers for data-driven inventory management in a Newsvendor setting with nonstationary demand. For this, we study two novel approaches which are based on machine learning techniques (linear quantile regression and tree-based regression, respectively) and which use historical demand observations and auxiliary data to prescribe optimal inventory quantities. We identify three major performance drivers, that are non-linearity, heteroscedasticity and usability. We evaluate both models both in an extensive simulation experiment where we control different properties of the feature-demand relationship as well as on a complex real-world data set from a restaurant chain. From these experiments we conclude that in situations in which the structure of the feature-demand relationships is not known to be predominantly linear – which can be assumed to be the typical case in practice – we recommend the much more robust tree-based approach. Furthermore, the tree-based also provides comparatively better results for feature-dependent noise, i.e., heteroscedasticity. We also find that in the real-world setting, the tree-based approach performs better with very small data-sets due to the better built-in feature selection which is important in terms of usability in practice. [6]

[6]This paper is co-authored by Jan Meller and Richard Pibernik Meller et al. [2018].

3.1 Introduction

Throughout the last decades a large body of research in operations management addressed the question of how to determine optimal inventory levels when demand is uncertain. Traditionally, inventory models process input data such as demand forecasts based on historical demand/sales, information about the accuracy of the forecasts, and costs for overstocking and understocking to determine an inventory quantity. These models lead to acceptable results when forecasts are fairly accurate and forecast performance is stable across time. In many industries, however, these conditions are not satisfied. In the fashion, retail as well as many service industries (e.g., for restaurants, as we will see later), demand is oftentimes driven by a number of exogenous and endogenous factors (e.g., weather conditions, media attention, short and medium term seasonal factors, promotional activities, etc.) at the same time. As a result, decision makers frequently face highly nonstationary demand patterns that can hardly be predicted with sufficient accuracy when forecasts are based solely on historical demand. In these instances conventional methods for forecasting and inventory management may not only lead to low forecast accuracy, but, more importantly, excessive inventories and/or large unmet demand [Carrizosa et al., 2016].

During the last years, access to large amounts of data that can potentially explain demand (variations) has improved significantly, and, at the same time, the cost for obtaining, storing and processing this data has decreased substantially. Recent work in operations management has, for example, utilized data such as Google searches [Bertsimas and Kallus, 2019], clickstreams [Huang and van Mieghem, 2014], weather information [Arias and Bae, 2016] and condition data of sensor-equipped machinery [Elwany and Gebraeel, 2008]. How to leverage this auxiliary data by extracting suitable "features" – measurable exogenous variables that potentially have predictive power – and incorporating them into inventory models has recently attracted interest in the academic community.

The work by Ban and Rudin [2018] and Bertsimas and Kallus [2019] have recently proposed novel approaches that use historical demand *and* feature

data to directly prescribe inventory quantities. Ban and Rudin [2018] propose two distinct approaches for "The Big Data Newsvendor". In the first, they apply linear quantile regression (LQR) to derive the Newsvendor order quantity from historical demand observations and (potentially extensive) feature data.[7]. Their second approach is kernel-based optimization. They use kernel functions to derive weights for each observations. The actual decision is then a locally weighted average over the historical observations. Bertsimas and Kallus [2019] formulate five prescriptive models they consider as the most broadly and practically effective motivated by the following five well-known machine learning techniques: k-nearest-neighbors regression, kernel-based optimization (similar to the approach by Ban and Rudin [2018]), local linear regression and two approaches using tree-based regression (TBR), namely classification and regression trees (CART) and random forest (RF) which basically uses multiple CARTs.

Out of these models, we consider the ones based on LQR and TBR for our analysis since these approaches are representatives of fundamentally differing modeling assumptions. According to Hastie et al. [2013] a fundamental difference in the modeling assumption is, whether we try to find a global model such as LQR or a local model such as TBR. While the LQR approach globally fits the parameters of a (linear) model to the entire set of available data, TBR approaches first partition the feature space and then determine local decision quantities for each partition based on the observations that fall into this partition. Furthermore, sophisticated TBR approaches such as RF as introduced by Breiman [2001] are a powerful machine learning technique for regression-related tasks due to their accuracy and versatility in a wide range of settings as well as due to their internal feature selection mechanism. RFs are competitive in terms of prediction accuracy compared to other machine learning techniques in various settings [Caruana and Niculescu-Mizil, 2006] and [Caruana et al., 2008]. In addition, the two other local concepts Kernel-based optimization and k-nearest neighbors are not able to model interaction effects between features.

[7] A similar idea was presented by Beutel and Minner [2012].

In our work we analyze how the fundamentally different models – the global LQR and the local TBR – perform in a prescriptive model to derive inventory quantities. We show that they lead to different performances (that is, different total costs for overstocking and understocking) depending on the specific problem instance and dataset at hand. Obviously, a decision maker would like to know which approach to use under which conditions. Answering this question lies at the heart of the research presented in this paper. Our contribution is twofold: first, we analyze how well the two competing approaches can make use of feature data to prescribe inventory quantities. In particular, we focus on the question whether one approach strictly dominates the other, i.e, whether we can find conditions under which one approach consistently leads to better inventory decisions. Second, we explore the structural characteristics of the two approaches and examine which conditions would render one method preferable over the other from the perspective of a decision maker who considers applying them in a real-world setting.

For this purpose, we first introduce a demand model that allows us to formalize different properties of the feature-demand relationship and the demand uncertainty. Thereafter, we introduce and discuss the competing approaches and derive a set of conjectures regarding their performance and usability under different conditions. We then present the results of two studies, which we carried out to provide numerical and empirical evidence for our conjectures: The first study is a controlled simulation experiment that allows us to isolate individual effects of the feature-demand relationships as well as the structure of uncertainty on the performance of the two competing approaches. Studying the two approaches in a controlled environment helps us overcome a problem that one frequently encounters when evaluating the performance of machine learning methods based on real-world datasets: the performance is usually dependent on the specific problem instance and there are many confounding effects – it becomes virtually impossible to draw meaningful conclusions regarding the performance drivers of different approaches. In our second study (where the aforementioned confounding effects are present), we apply the two models to a real-world problem instance of a fast casual restaurant chain that has to take daily inventory decisions in the presence of highly nonstationary

demand. The nonstationary demand can, in part, be explained by features such as the day of the week, the time of the year as well as weather conditions. In this second study we focus less on structural effects of the two approaches, but concentrate on their usability. For example, we analyze how the two approaches behave when only a limited amount of data is available.

The results of our studies suggest that both approaches clearly outperform a realistic benchmark approach (based on sample average approximation - SAA) that ignores feature data. They both lead to very substantial performance improvements as long as the uncertainty in the demand observations is not extremely high. In both our studies the TBR approach appears to be very robust to different feature-demand relationships; moreover, it can effectively exploit feature-dependent uncertainty (systematic noise). Under most relevant conditions the TBR approach performs better than the (basic) LQR approach. The LQR approach can, however, be enhanced through additional feature engineering. In our second study we observe that the LQR approach with additional (engineered) features can outperform TBR. However, as we discuss in detail, feature engineering requires substantial effort and is a very difficult task, especially under big data regimes and without good a priori knowledge about feature-demand relationships. Thus, from a practical point of view, we recommend the use of tree-based approaches, especially because of their good performance, robustness and superior usability.

3.2 Feature-based Newsvendor problem

Determining inventory levels facing uncertain demand is a well-studied problem in operations management. However, as we outline in this chapter, the direct consideration of feature data for decision making is an area of active research. For single-period settings the Newsvendor model has attracted a lot of attention as it is simple, intuitive, and yet captures the most important characteristics of many underlying business problems. For extensive reviews, the interested reader is referred to Lee and Nahmias [1993] or Khouja [1999].

Since the Newsvendor logic constitutes an important building block of

our model, we briefly introduce it here: Based upon a known cumulative distribution function $F_D(\cdot)$ of demand D the objective is to minimize the expected sum of overage costs c_o and underage costs c_u:

$$\min_{q \geq 0} \mathbb{E}[C(q, D)] = \mathbb{E}[c_u(D - q)^+ + c_o(q - D)^+] \tag{3.1}$$

For this standard Newsvendor problem, it can be shown [c.f. Zipkin, 2000] that the optimal order quantity q^* is given by:

$$q^* = F_D^{-1}\left(\frac{c_u}{c_u + c_o}\right) \tag{3.2}$$

Traditionally, q is determined by first fitting a theoretical distribution to historical demand realizations to obtain F_D and then solving 3.2. This is referred to as *separate estimation and optimization* [Liyanage and Shanthikumar, 2005]. One drawback of this approach lies in the difficulties associated with estimating the "correct" demand distribution (e.g., Scarf et al. [1958] and Klabjan et al. [2013]), which is particularly relevant when demand is highly nonstationary. Another issue results from the fact that estimation and optimization are performed on different criteria that are not necessarily aligned: while the estimation of the demand distribution is usually based on the minimization of some symmetric measure of deviation (e.g., mean squared error), the subsequent optimization would consider potentially asymmetric (underage and overage) costs as exemplified by Equation (4.5). Recently, so-called data-driven approaches have been proposed to overcome these issues. They have in common that they directly relate the inventory decision to available historical data (that is, they follow a *joint estimation and optimization* logic. Sample Average Approximation (SAA) is the most basic of these approaches and has been shown to lead to asymptotically optimal results when demand can be assumed to be i.i.d. [Kleywegt et al., 2002]. For the Newsvendor problem, SAA can be applied as follows: Given a set of historical demand observations $\mathcal{D}_N = [d_1, d_2, \ldots, d_N]$, one determines the order quantity by minimizing the empirical risk, that is the average costs over the sample $\mathcal{D}_N$. The data-driven Newsvendor problem can be written as:

$$\min_{q \geq 0} \hat{R}(q, \mathcal{D}_N) = \frac{1}{N} \sum_{i=1}^{N} [c_u(d_i - q)^+ + c_o(q - d_i)^+] \tag{3.3}$$

where q is the order quantity, d_i the demand in period i, and $(.)^+$ is a function that returns 0 if its argument is negative, and else its argument. The solution to this sample quantile regression is

$$q^*_{SAA} = d_{<j>} \; with \; j = \left\lceil \frac{c_u}{c_u + c_o} N \right\rceil \tag{3.4}$$

which is the jth-largest demand observation [Bertsimas and Thiele, 2005]. An equivalent formulation of equation (3.4), which is slightly closer to the form used in quantile regression, as for example in Koenker [2005], is given by:

$$q^*_{SAA} = \inf_{d_j \in \mathcal{D}_N} \{d_j : |\{d_i \in \mathcal{D}_N : d_i \leq d_j\}| \geq \frac{c_u}{c_u + c_o} N\} \tag{3.5}$$

where $|\{\cdot\}|$ is the cardinality of a set. We refer to q^*_{SAA} as the Newsvendor quantity for a given set of demand realizations, based on SAA. Furthermore, we refer to the fraction $\frac{c_u}{c_u + c_o}$ as service level (SL) since this ratio between underage and overage costs determines how many demand instances of the sample could not have been satisfied when choosing q_{SAA}.

However, q^*_{SAA} is only (asymptotically) optimal if we assume that demand realizations are i.i.d. Clearly, this condition is not satisfied in many real-world situations where demand is oftentimes seasonal, cyclical, follows a trend and may be driven by a variety of other factors (e.g., weather).[8]

In the real-world data we use in our second study, for example, we observe highly nonstationary demand that can, in part, be explained by the day of the week, the time of the year as well as weather conditions. In such instances, SAA will most likely not provide satisfactory solutions to the Newsvendor problem. A common remedy to account for such nonstationarities is to consider potentially meaningful features to explain part of variations of demand.

In the remainder of this section we first describe how feature-dependent

[8] Bertsimas and Thiele [2005] adapt SAA to account for a trend by adjusting the sample size – however this does not account for other forms of nonstationary demand, e.g., seasonal or cyclical demand.

demand can be modeled. We extend traditional demand models to incorporate the influence of features (e.g. day of the week, weather conditions, etc.) and show how features can influence both the demand level and demand uncertainty. This forms an important basis for our further analyses. We then introduce two novel and distinct approaches that use feature data to determine "optimal" inventory decisions in a Newsvendor setting. We highlight the differences between the two approaches and conjecture under which conditions one approach will outperform the other. Finally, in Sections 3.3 and 4.4.3 we conduct two studies to explore whether our conjectures hold in a controlled environment (Section 3.3) and for a real-world dataset (Section 4.4.3). Next to a performance comparison of the competing approaches, we also discuss how suitable the different approaches are for practical implementation.

3.2.1 Demand model with feature-dependent demand

As it is common in the literature, we model demand consisting of two components, the demand level as well as an additional random component. We assume the demand level to be deterministic and dependent on the values of some features. The random component represents exogenous uncertainty, which may also be feature-dependent. More formally, we model demand D as follows:

$$\begin{aligned} D &= \mu + \epsilon \\ \text{with } \mu &= f(\mathbf{x}) \\ \text{and } \epsilon &\sim \mathcal{N}(0, \sigma(\mathbf{x})) \\ \mathbf{x} &\in \mathbb{R}^k \end{aligned} \tag{3.6}$$

where $f(\mathbf{x})$ is the function that describes the relationship between the demand level and the values of the features, represented by vector $\mathbf{x}$. The feature vector $\mathbf{x}$ can comprise, for example, the weekday, month, temperature, and other features that potentially influence the demand level. Moreover, ϵ is assumed to be a normally distributed random variable with mean zero and a

standard deviation depending on the feature vector $\mathbf{x}$. The demand model in (4.1) is very generic and provides us with the flexibility to model particular feature-demand relationships and to evaluate how different approaches for solving the feature-based Newsvendor problem perform under these scenarios. The different demand scenarios are determined by the particular choice of the functional relationship between the demand level and the feature vector, as well as the random component and the feature vector.

Consider first the relationship between the demand level μ and the feature vector $\mathbf{x}$, represented by $f(\mathbf{x})$. In the simplest case, the relationship is linear and $f(\mathbf{x})$ has a simple form such that $\mu = \beta_1 x_1 + \beta_2 x_2 + ...$. While this is a very desirable case from a modeling and prediction point of view, in reality we cannot expect to find linear relationships between (all) features and the demand level. A case in point is the relationship between demand and weather conditions. We cannot expect demand to be linearly increasing/decreasing in the temperature or precipitation.

For instance, in our second study (Section 4.4.3), demand for certain products like Calamari may at first increase as the temperature increases, but, beyond a certain high temperature, the overall demand may also decrease because fewer people eat at restaurants. In such cases, $f(\mathbf{x})$ should have some nonlinear functional form, consisting, for example, of polynomials involving the individual features and some scaling factors. This, however, may still be a too simplistic representation of the true demand generating process, because features may exhibit complex interactions in the way they affect demand. For example, weather conditions may affect demand differently on weekdays than on weekends. In this complex case the interaction of features may influence demand. In Figure 3.1 we illustrate how the demand level can be modeled as a function of the feature vector.

The representation of the uncertainty is the second important element of our demand model. We explicitly consider two relevant cases of how uncertainty can affect demand: The first one is the case of *homoscedasticity*, where the level of uncertainty is independent of the features, i.e., we can model it via a random variable ϵ with mean zero and a constant variance σ^2. In this case, the uncertainty is completely unsystematic. On the other hand, uncer-

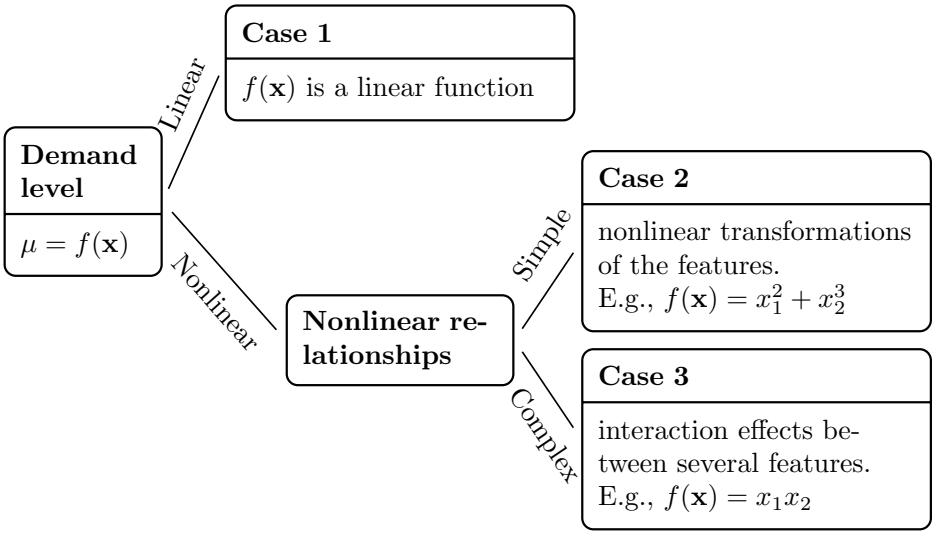


Figure 3.1: Potential characteristics of the demand generating model in terms of the demand level $\mu = f(\mathbf{x})$

tainty can be feature dependent. For example, uncertainty may be higher on weekends and public holidays than on regular weekdays.[9] In essence, this implies that there is some systematic structure within the uncertainty that can potentially be predicted. We term this as the *heteroscedastic* case. We model such behavior via the standard deviation of the random noise component $\sigma(\mathbf{x})$ which in the heteroscedastic setting depends on the feature vector $\mathbf{x}$. Based on these different manifestations of the demand model which cover a wide range of practically relevant scenarios we can evaluate the performance of feature-based inventory management models. In the following we describe two different approaches that incorporate the feature-demand relationship into the process of determining inventory decisions.

3.2.2 Competing models

The functional relationships between feature values and demand (uncertainty), as described in the previous section, are typically not known and/or not well-defined a priori. In the following we assume the decision maker has access to

[9] Meinshausen [2006] mentions a similar example in the context of predicting ozone levels.

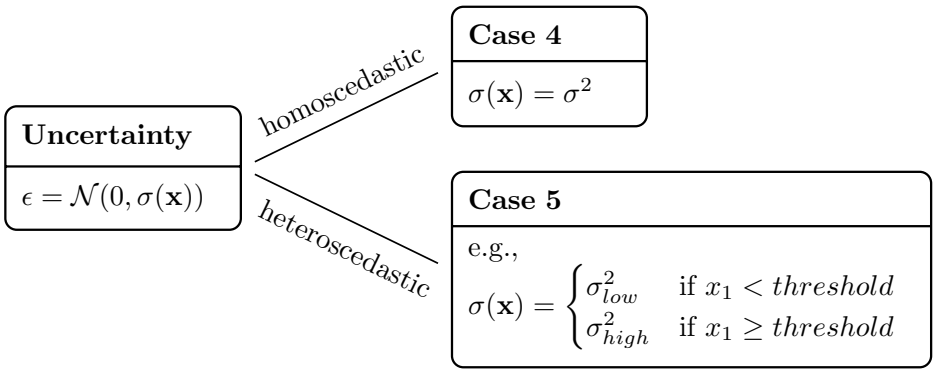


Figure 3.2: Potential characteristics of the demand generating model in terms of the uncertainty $\sigma(\mathbf{x})$

a set of historical demand observations and corresponding feature data and wants to employ a model that uses this data as input and returns the "optimal" inventory quantity. More formally, let $\mathcal{T} = \{(d_i, \mathbf{x}_i), i = 1, \ldots, N\}$ denote the available data, where d_i is the i-th demand realization and $\mathbf{x}_i$ the vector of feature values in period i. We assume that a relationship exists between the demand realizations d_i and $\mathbf{x}_i$ as described in the previous section, which, however, is not known a priori. A feature-based Newsvendor model prescribes the inventory quantity q based on a feature vector $\mathbf{x}$, more formally: $q : \mathcal{X} \to \mathbb{R} : q(\mathbf{x})$ where $\mathcal{X}$ denotes the feature space containing all feasible feature vectors.

Recently, a limited number of feature-based inventory models have been proposed. In our analysis we focus on two specific models: The first is based on linear quantile regression and was introduced by Ban and Rudin [2018] and Beutel and Minner [2012]. In the following we refer to this model as linear quantile regression Newsvendor (LQR-NV). The second approach applies tree-based regression and is a modification of the model proposed by Bertsimas and Kallus [2019]. In the following we refer to this model as tree-based regression Newsvendor (TBR-NV). As highlighted in Section 3.1 we choose these two models because they can be considered as representative for two distinct approaches to capture the relationship between features and demand in order to prescribe inventory quantities. In the sequel we explain the two

models in detail and provide a discussion of their distinct characteristics.

What both models have in common is that they calibrate, i.e., "learn" a model based on a given dataset $\mathcal{T}$, which is the fundamental concept of machine learning. In their contribution, Ban and Rudin [2018] learn a linear decision rule of the form $q : \mathcal{X} \to \mathbb{R} : q(\mathbf{x}) = \mathbf{q}^T\mathbf{x} = \sum_{j=1}^{k} q^j x^j$ to relate a feature vector $\mathbf{x}$ of length k directly to the Newsvendor quantity $q(\mathbf{x})$. They optimize the weights $q^j,\ j = 1, ..., k$ for each feature based on $\mathcal{T}$. In detail, the weights are determined by using the following objective function:

$$\min_{q:q(\mathbf{x})=\sum_{j=1}^{k} q^j x^j} \frac{1}{n}\sum_{i=1}^{n}[c_u(d_i - q(\mathbf{x}_i))^+ + c_o(q(\mathbf{x}_i) - d_i)^+] + \lambda \sum_{j=1}^{k} |q^j| \qquad (3.7)$$

where λ is a regularization parameter that controls the complexity of the resulting model and hence provides an internal feature selection mechanism. An important advantage of this approach is, that it allows for an efficient solution via standard linear programming techniques.

As an alternative to linear quantile regression, we consider tree-based regression for modeling the relationships between features and decisions. The basic idea of this class of approaches is to find partitions of the input feature space that contain observations which were made under similar conditions. Tree-based approaches achieve such partitions by recursively splitting the feature space along the feature dimensions. The presented model in this section is based on the contribution of Bertsimas and Kallus [2019] and is adapted to the basic Newsvendor setting. We start with a brief introduction to the fundamental concept of tree-based machine learning as an important preliminary for the subsequent sections. Tree-based algorithms partition the feature space by recursively finding the feature along with a split value that minimizes a loss function over a given set of historical "training data". This is recursively repeated until either an additional split does not lead to a substantial improvement or a minimum number of observations, which would be available for the next split, is reached. These are the stopping criteria of the algorithm. The mean squared error is the standard loss function for decision tree learning

as introduced by Breiman et al. [1984] and is also applied in Bertsimas and Kallus [2019]. Recursively splitting a set of learning data yields a partitioning of the feature space that can be represented in a tree-like structure as for example in Figure 6.2. We refer to the distinct partitions as regions or leaves.

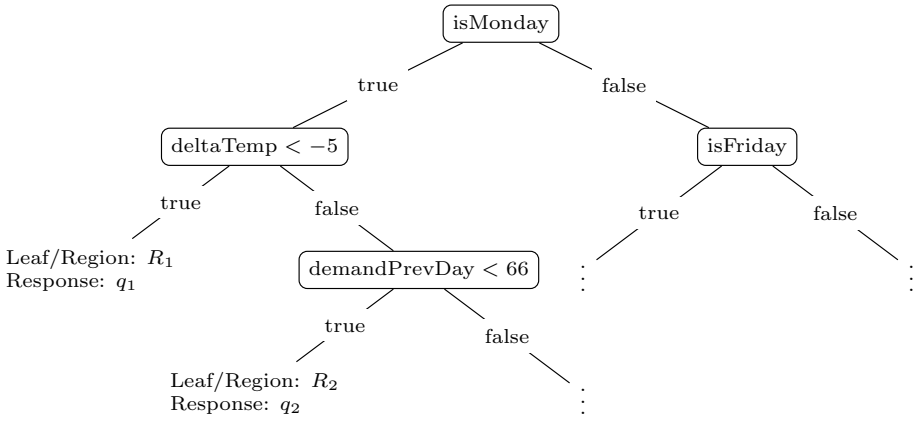


Figure 3.3: Example of a decision tree

After learning the regions, Bertsimas and Kallus [2019] propose to sort the learning data into the leaves according to their feature values and subsequently solve an optimization problem on that sample of demand realizations for each region. That is, for region R_l we find the response q_l by:

$$q_l = \arg\min_{q} \sum_{i:x_i \in R_l} c(q; d_i) \tag{3.8}$$

In a basic Newsvendor setting this is the same as finding the sample quantile, which is also referred to as the sample average approximation, for the subset in each region. That is, for each region R_r we obtain a constant response: $q^*_{SAA,r}$ from equation (3.4) or equivalently equation (3.5). Similar to the linear quantile regression approach in Ban and Rudin [2018], the tree-based approach yields a model that directly prescribes an inventory decision given a new, unseen instance of features. However, instead of using a linear model, the prescribed quantity is then the response of the region the new instance is sorted into by comparing its feature values with the nodes in the tree. For basic decision trees the function $q(\mathbf{x}')$, which returns the order quantity for

an unknown instance $\mathbf{x}'$ is given by:

$$q(\mathbf{x}') = \sum_{l=1}^{L} q_l \mathbb{1}(\mathbf{x}' \in R_l) \tag{3.9}$$

where the indicator function $\mathbb{1}(\mathbf{x}' \in R_l)$ returns 1 if instance $\mathbf{x}'$ belongs to region m and 0 otherwise and the responses q_l for all L regions are determined as described by equation (3.8).

Tree-based regression becomes more robust against overfitting if hundreds of trees are learned on random samples of learning data and then combined to so called random forests [Breiman, 2001]. In the following, we describe how Bertsimas and Kallus [2019] extend their concept for prescriptive analytics from single trees to random forests. The main idea is to find the decision quantity that minimizes the costs for the entire set of T trees which are learned on randomly drawn samples of the learning data. Therefore, for tree t we determine the average costs which would occur given a decision quantity q and the sample of demand realizations from the learning data, which would be sorted in the same region of this tree t as the unknown instance $\mathbf{x}'$, that is $\{d_i | x_i \in R^t(\mathbf{x}')\}$ where $R^t(\mathbf{x})$ assigns a region of tree t to a observation $\mathbf{x}$. More formally, this is given by:

$$\hat{q}^{RF}(\mathbf{x}') = \arg\min_{q} \sum_{t=1}^{T} \frac{1}{|R^t(\mathbf{x}')|} \sum_{i:\mathbf{x}_i \in R^t(\mathbf{x}')} c(q; d_i) \tag{3.10}$$

where $|R^t(\mathbf{x}')|$ is the number of instances from the learning data that are sorted into the same region as the unknown instance $\mathbf{x}'$.

3.2.3 Model comparison

To examine the performance of LQR-NV and TBR-NV, which we introduced in the previous section, we compare them along three dimensions: First, we consider different specifications of $f(\mathbf{x})$ and how the models account for these structural differences. Second, we analyze the implication of the noise structure on prescribed inventory quantities, i.e. different specifications of $\sigma(\mathbf{x})$ that control whether we are in a homoscedastic setting (for $\sigma(\mathbf{x}) = constant$)

respectively the degree of heteroscedasticity. Finally, we evaluate the models' usability, i.e. how practicable such a solution would be to roll out in a real-world setting. In this chapter we discuss the theoretical differences and derive specific conjectures before we present the results of our experiments in the subsequent chapters.

Comparing linear and tree-based regression, the most obvious distinction is the way features are related to the final decisions. LQR-NV explicitly models linear relationships between the input variables and the model output, imposing a global linearity assumption. In contrast, TBR-NV does not rely on a global structural assumption, but splits the parameter space into rectangular regions and finds a response for each of these regions. Despite its implicitly linear structure, LQR-NV is able to incorporate nonlinear feature-demand relationships by transforming the basic features and adding these "new" features to the training dataset [Ban and Rudin, 2018]. We will discuss the required feature engineering in Section 3.4.3

Without additional feature engineering, we expect performance differences between the two approaches to depend on the underlying structure of the feature-demand relationships. In settings where this relationship is predominantly linear, LQR-NV should outperform TBR-NV since the former is able to model this structure by its explicit internal representation. On the other hand, tree-based regression should provide better results when the feature-demand relationship is nonlinear and/or entails complex interactions. In cases that features and actual demand exhibit a nonlinear relationship, we expect that TBR-NV performs better than LQR-NV because it does not assume a specific feature-demand relationship. We summarize our first conjecture as follows:

Conjecture 1 (Nonlinearity)**.** *LQR-NV outperforms TBR-NV for settings where the feature-demand relationship follows a predominantly linear pattern. Compared to LQR-NV, TBR-NV provides better results the more the feature-demand relationship deviates from a linear relationship.*

Besides the expected demand level and its relationship with the features, a crucial property of any inventory policy is the remaining uncertainty around

these expected demands and how they are accounted for. In the way they use the learning data account for uncertainty, the two models under consideration reveal structural differences: LQR-NV minimizes the empirical risk for the entire learning dataset by optimizing weights for each feature. Hence, the inventory decision for an unseen instance is based on the global uncertainty of the entire learning data. On the other hand, TBR-NV performs a local optimization for each region based on the instances from the learning data that belong to this region. That is, the decision for an unseen new instance is based on demands that were observed under similar conditions.

In classical regression settings, we would usually assume $\sigma(\mathbf{x})$, i.e., the measure of how strongly demand varies around its mean, to be constant across all instances. Under this assumption each historical training instance would provide the same amount of information via its residuals and, hence, LQR-NV, as it takes into account the entire learning set, should be able to more accurately estimate this uncertainty. However, often this assumption of homoscedasticity does not hold in practice. In many cases, demand can be predicted much more accurately under some conditions than under other conditions. For example, the demand for a restaurant on a regular weekday during lunchtime may have significantly less variability than on a weekend with a special event taking place nearby. If we have feature-related uncertainty, i.e., if $\sigma(\mathbf{x})$ is not constant, the local optimization of TBR-NV can be advantageous since it implicitly accounts for feature dependent uncertainty.

As an example, assume that the demand observations in a certain region (which is characterized by splits of the feature space) exhibit higher uncertainty, measured, for example, in terms of the standard deviation. In this case, the decision quantity obtained for this region explicitly accounts for the uncertainty associated with the instances sorted into this region. Thus, uncertainty is feature-dependent (more generally: state-dependent) and the resulting decision is explicitly based on the feature-dependent uncertainty. The global optimization of LQR-NV cannot account for such feature-dependent uncertainty. This leads us to our second conjecture.

Conjecture 2 (Homo- vs. heteroscedasticity)**.** *In a homoscedastic setting,*

performance is driven by the feature-demand relationship. The performance of TBR-NV will increase relative to LQR-NV with increasing levels of heteroscedasticity, that is, the more $\sigma(X)$ depends on X.

Finally, ease of use and applicability to practically relevant settings is our third dimension for the comparison of the alternative models. In this context, we address two important aspects of usability. These are the influence of the amount of learning data on performance and the need for feature engineering.

Feature engineering is a crucial part of applying machine learning in practice since it turns raw data into potentially meaningful features. Feature engineering comprises the process of cleaning, summarizing and aggregating raw data that typically comes from distributed sources into a table of features that can be used as input by machine learning models [Zheng, 2017]. We expect that tree-based approaches such as TBR-NV require significantly less effort for this task compared to LQR-NV for two main reasons. For one, TBR can naturally handle categorical variables whereas linear regression requires those variables to be split in multiple binary features, one for each category. In addition, linear regression can only handle nonlinear feature-demand relationships and interaction effects by manually splitting the feature space into regions for which piecewise linear models can be fitted to the data. In practice, determining these splits correctly is very difficult because information about how to best model nonlinearity and complex interactions is typically not available ex ante.

Our second aspect of usability refers to the performance of both models with respect to the amount of available learning data. In practice it is important that a data-driven model requires as little learning data as possible to achieve a reliable performance since the amount of available learning data is usually limited. Learning data should be gained from past observations under similar conditions. Thus, its value is very limited when fundamental changes in the business processes occur or when the predictive power of past observations declines quickly over time. Because of this, a decision maker needs to know how many past observations a model requires until it provides reliable results and which model is preferable if only a limited amount of learning

data is available. We expect LQR-NV to perform relatively better if only limited learning data is available because it uses the entire dataset to derive the inventory decision while TBR-NV bases the decision only on subsets of the learning data with a lower number of observations. We expect TBR-NV to require more data to capture the particular feature-demand relationships and eventual heteroscedastic uncertainty. Based on the previous discussion we derive our third conjecture.

Conjecture 3 (Usability)**.** *We expect LQR-NV to require additional feature engineering in order to provide competitive performance (compared to TBR-NV) in realistic demand settings. Furthermore, we expect LQR-NV to perform relatively better (compared to TBR-NV) when only little learning data is available and TBR-NV to benefit more from additional learning data.*

Dimension	**LQR-NV**	**TBR-NV**
Nonlinearity (chapter 3.3.2)	Requires additional features from feature engineering to approximate nonlinear dependencies	Can be modeled by choosing partitions accordingly
Heteroscedasticity (chapter 4.4.2)	Can not make use of feature-dependent uncertainty due to global optimization	Local optimization can adapt to feature-dependent uncertainty
Usability (chapter 4.4.3)	Requires additional feature engineering to deal with nonlinearity, interaction effects and categorical variables; Requires additional testing to measure variable importance; no visualization	Can handle nonlinearity, interaction effects and categorical variables without additional feature engineering; Provides information about variable importance; Basic trees allow intuitive visualization

Table 3.1: Comparing key characteristics of linear regression and tree-based regression

Table 3.1 provides an overview of the differences between the two competing approaches along the dimensions we discussed previously. In the following chapter 3.3, we carry out a controlled simulation experiment to examine the first two conjectures regarding the effect of the (linear/nonlinear) feature-demand relationship and heteroscedasticity on the performance of each model. As it is difficult to evaluate aspects of usability in a self-designed simulation experiment, we also implemented both models in a real-world setting. In Section 4.4.3 we present the corresponding results of LQR-NV and TBR-NV, validate our findings from the controlled simulation experiment, and discuss the most relevant aspects of usability.

3.3 Study 1: Simulation analysis

Our first study is a controlled simulation experiment that allows us to quantify the individual effects related to the feature-demand relationships on the performance of LQR-NV and TBR-NV. More specifically, we want to shed light on how the two competing models perform when the relationship between level demand and features is linear/nonlinear and when demand uncertainty is independent/dependent on the feature values, that is when we have homoscedastic or heteroscedastic noise (see Figures 3.1 and 3.2 and our Conjectures 1 and 4).

In the next section we first describe our experimental setup. Most importantly, we explain how we induce the different types of feature-demand relationships through our particular choice of a demand model and its parameterization. Thereupon, we present the performance of LQR-NV and TBR-NV relative to the SAA benchmark (as defined in equation 3.4) under a variety of controlled scenarios, and discuss the impact of different types of demand-feature relationships. This will allow us to derive meaningful conclusions regarding performance differences between LQR-NV and TBR-NV under relevant conditions.

3.3.1 Experimental setup

For our simulation experiment we use an additive demand model as described in (4.1). We separately control the feature-demand relationship and the feature-dependent uncertainty. More formally, we determine demand D as follows:

$$\begin{aligned} D &= \mu_p + \epsilon_\gamma \\ \text{with } \mu_p &= f_p(\mathbf{x}) = f_p(x_1) + \cdots + f_p(x_k) = x_1^{p+1} + ... + x_k^{p+1} \\ \text{and } \epsilon_\gamma &= \mathcal{N}(0, \sigma_\gamma(\mathbf{x})) \\ \text{where } \sigma_\gamma(\mathbf{x}) &= \begin{cases} 2(1-\gamma)\sigma_{base} & \text{if } x_1 < median(X_1) \\ 2\sqrt{1-2(1-\gamma)^2}\sigma_{base} & \text{if } x_1 \geq median(X_1) \end{cases} \\ \text{with } \sigma_{base} &= \mathbb{E}\left[\mu_p\right] cv_{noise} \end{aligned} \tag{3.11}$$

where p is the parameter that determines linearity (for $p = 0$) and the level of nonlinearity (for $p = 0,\ 0.4, ..., 4$), respectively, and γ is the simulation parameter that determines whether we obtain homoscedastic demand (for $\gamma = 0.5$) or increasing levels of heteroscedasticity (for $\gamma = 0.55, 0.6, ..., 1$).

The intuitive approach to obtain a set of learning data would be to randomly generate features from a certain distribution and to transform them according to parameter p to obtain the level demand. The additional noise component would be drawn from a normal distribution based on parameter γ and added to the level demand, yielding the demand observations for our study. However, if we followed this procedure, changes of p or γ would not only affect the feature-demand relationship and the heteroscedasticity, but also the distribution of the demand realizations. Since all of our methods are data-driven, such changes of the demand distributions would also have an effect on the performance. Thus, we would not be able to isolate the effects of nonlinearity and heteroscedasticity.

For this reason, we follow a different approach: We start by drawing the sample of demand realizations and subtract the random noise components that are determined with respect to γ to obtain the level demands. We then

determine a random partition of each base demand into k features which are then transformed with respect to p using the inverse of the transformation function we apply in (4.23). Using this "backward" procedure, we can control the demand distribution and avoid that changes in the demand distribution obfuscate the effects that we actually want to analyze.

In detail, we draw N demand observations $d_i,\ i = 1, \ldots, N$ from a uniform distribution with range $[50; 150]$. Then, we partition each d_i randomly in k parts such that $d_i = x'_{i1} + \cdots + x'_{ik}$. k corresponds to the the number of features. We use the x'_{i1} to determine whether the error for an observation ϵ_γ is drawn from a distribution with high or low standard deviation according to the definition of $\sigma_\gamma(\mathbf{x})$ in (4.23). We then obtain the level demands $\{\mu_{p,i}\}_{i=1,\ldots,N}$ by subtracting $\epsilon_{\gamma,i}$ from d_i for all $i = 1, \ldots, N$. By subtracting the random error terms we may obtain negative base demands, which is not realistic and also causes problems when applying the inverse of our transformation function. Therefore, we shift these base demands by the minimum of their values to obtain only non-negative demand levels. Such a constant shift does not influence the performance of LQR-NV and TBR-NV. From the non-negative base demands we determine $x_{1i}^{p+1}, \ldots, x_{ki}^{p+1}$ by using the same ratios of the partition such that $x_{1i}^{p+1} + \cdots + x_{ki}^{p+1} = \mu_{p,i}$ and $\frac{x_{ji}^{p+1}}{x'_{ji}} = \frac{\mu_{p,i}}{d_i}$. Applying $f_p^{-1}(x)$ to each x_{ji}^{p+1} yields the final features x_{ji} we use with the respective d_i as learning data. Algorithm 2 summarizes our approach for data generation.

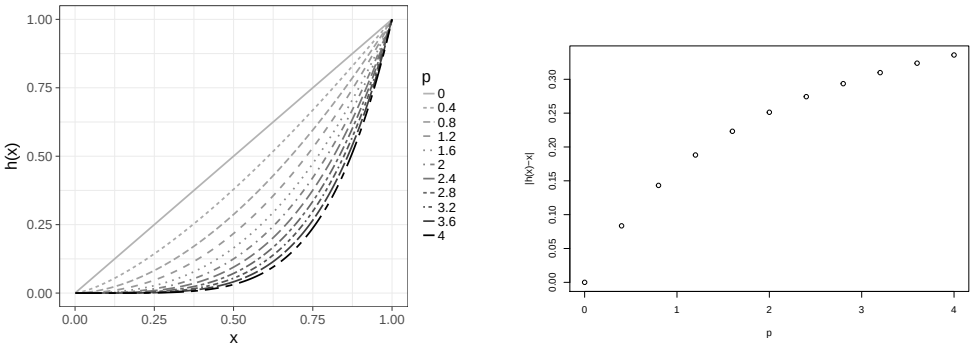
Parameter p determines how strong we deviate from a linear feature-demand relationship. In Figure 3.4 we show the effect of changes in p on a single feature x. Of course, transforming individual features with a function that maintains monotonicity is a relatively simple way to induce nonlinearity into the feature-demand relationship. We do not consider more complex structures that would arise from combinations of individual features or more complex transformation functions on individual features in order to keep the setting as tractable as possible.

We model heteroscedasticity by relating the standard deviation of the error term to the feature x_1. For values of x_1 below the median we obtain low standard deviations (depending on γ) and for values higher than the median,

Input : $I = (p, \gamma, cv_{noise}, N)$
Output: $R = \{d_i, \mathbf{x}_i\}_{i=1,\dots,N}$

begin
  for $i \leftarrow 1$ **to** N **do**
    $d_i \longleftarrow$ `Uniform(50, 150)` `// generate demand`
  end
  for $i \leftarrow 1$ **to** N **do** `// generate dummy features`
    Randomly choose $(x'_{1i}, \dots, x'_{ki}) | x'_{1i} + \dots + x'_{ki} = d_i$ and $x'_{ji} > 0$
  end
  $MedX1 \longleftarrow$ `Median(`$x'_{11}, \dots, x'_{1N}$`)` `// sample median`
  $\sigma_{base} \longleftarrow \mathbb{E}[D]\, cv_{noise}$ `// base noise level`
  for $i \leftarrow 1$ **to** N **do**
    if $x'_{1i} < MedX1$ **then** `// include heteroscedasticity`
      $\sigma_{\gamma i} \longleftarrow 2(1-\gamma)\bar{d}\, \sigma_{base}$
    else
      $\sigma_{\gamma i} \longleftarrow 2\sqrt{1 - 2(1-\gamma)^2}\bar{d}\, \sigma_{base}$
    end
    $\epsilon_i \leftarrow$ `Normal(0,` $\sigma^2_{\gamma i}$`)` `// error component`
    $\mu'_i \longleftarrow d_i - \epsilon_i$ `// determine level demands`
  end
  $\mu_{Min} \longleftarrow |min(\mu'_1, \dots, \mu'_N)|$ `// find smallest` μ
  for $i \leftarrow 1$ **to** N **do**
    $\mu_i \longleftarrow \mu'_i + \mu_{Min}$ `// shift all` μ_i `by constant`
    Determine $(x^p_{1i}, \dots, x^p_{ki}) | \frac{x^p_{ji}}{\mu_i} = \frac{x'_{ji}}{d_i}$
    for $j \leftarrow 1$ **to** k **do**
      $x_{ij} = x_{ji}^{p+1\frac{1}{p+1}}$ `// determine final features`
    end
  end
end

Algorithm 2: Simulation data generation



(a) Transformation of a single feature with respect to p to model nonlinearity.

(b) Mean absolute deviation from original feature after transformation with respect to p to model nonlinearity.

Figure 3.4: Effect of exponentiation of an individual feature to model nonlinearity

we set a higher standard deviation. The joint standard deviation is always approximately σ_{base} independent of γ, because the combined standard deviation of two samples is given by $\sigma_{12} = \sqrt{\frac{N_1\sigma_1^2+N_2\sigma_2^2}{N_1+N_2}}$, where N_1 and N_2 are the number of observations in each sample. In our sample from (4.23), $N_1 \approx N_2$ due to the sample median as criterion. Hence, inserting the expressions for low and high standard deviations in the formula for the combined standard deviation yields σ_{base}. Because it is easier to interpret, we use the coefficient of variation cv_{noise} instead of σ_{base} to describe the uncertainty in our simulation experiments. In reality x_1 could represent, for example, the temperature, assuming that lower values lead to lower, but also more steady demand compared to higher temperatures for which it is more uncertain whether people eat at a restaurant or choose other options.

From each choice of parameters p, γ, cv_{noise} and SL, we obtain a training dataset $\mathcal{T}_{N_{sim}} = \{(d_i, \mathbf{x}_i), i = 1, ..., N_{sim}\}$. We use the first $N_{sim} - 1$ instances to train each model and evaluate them for period N_{sim}. For our analysis we choose $N_{sim} = 1000$ and $k = 3$ as the number of features that determine the level demands. In order to achieve stable results, we run the entire simulation $S = 1000$ times for a certain set of parameters and each model. We implemented the different models as described in Section 3.2.2 in R. For the

implementation of TBR-NV we use the ranger-package [Wright and Ziegler, 2017] and for LQR-NV we employ the quantreg-package provided by Koenker [2016].

For each simulation run $s = 1, \ldots, S$, we compute the costs achieved by model $m \in \{\text{LQR-NV, TBR-NV, SAA}\}$ by:

$$C_{s,m} = c_u(d_s - q^*_{s,m})^+ + c_o(q^*_{s,m} - d_s)^+ \tag{3.12}$$

where $q^*_{s,m}$ is the inventory quantity of model m and c_u and c_o are determined by SL, because $SL = c_u/(c_u + c_o)$ and we assume normalized costs so that $c_u + c_o = 1$.

We evaluate the mean $\bar{c}_m = 1/S \sum_{s=1}^{S} C_{s,m}$ across all $S = 1000$ simulation runs for each model m. To make our results easier to interpret, we report, whenever appropriate, the percentage mean cost improvement of LQR-NV and TBR-NV over SAA, which is defined as

$$\delta_m = \frac{\bar{c}_m - \bar{c}_{\text{SAA}}}{\bar{c}_{\text{SAA}}} \times 100; \quad m \in \{\text{LQR-NV, TBR-NV}\}. \tag{3.13}$$

In Table 4.1 we provide the parameter values for our subsequent experiments.

Experiment	**Nonlinearity** (chapter 3.3.2)	**Heteroscedasticity** (chapter 4.4.2)
Parameters		
p	$\{0, 0.1, \ldots, 3\}$	0
γ	0.5	$\{0.5, 0.55, \ldots, 1\}$
Controls		
cv_{noise}	$\{0.05, 0.25, 0.5, 1\}$	$\{0.05, 0.25, 0.5, 1\}$
SL	$\{0.5, 0.67, 0.8, 0.95\}$	$\{0.5, 0.67, 0.8, 0.95\}$

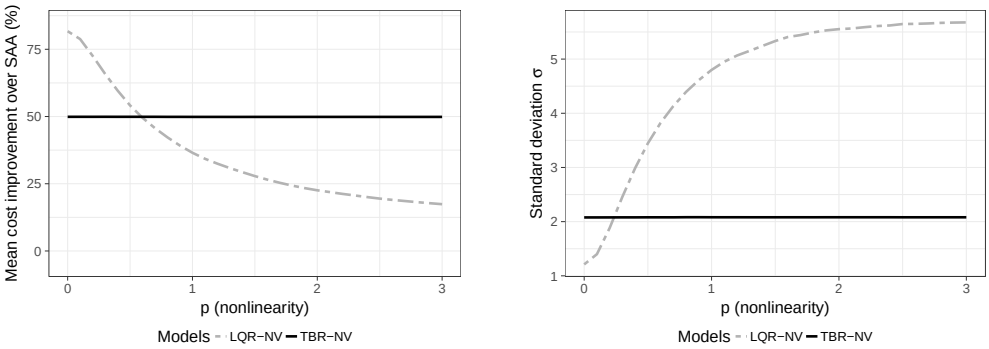
Table 3.2: Parameter settings for our experiments

In Section 3.3.2 we analyze the influence of the feature-demand relationship by varying p. Thereafter, in Section 4.4.2, we analyze the influence of

feature-dependent uncertainty by varying γ. In both analyses we use base noise (cv_{noise}) and different service levels (SL) as control variables.

3.3.2 Model performance depending on the feature-demand relationship

In our first set of analyses we examine the influence of the underlying relationships between the features and the final demand on the decision quality of the two compared approaches. In the panel on the left hand side of Figure 3.5 we report the δ_m for $m \in \{\text{LQR-NV, TBR-NV}\}$ as described in (4.26) under a 80% service level and a low level of noise ($cv_{noise} = 0.05$) for varying levels of nonlinearity (reflected by parameter p). The panel on the right hand side of Figure 3.5 displays the standard deviation of $C_{s,m}$ for $m \in \{\text{LQR-NV, TBR-NV}\}$.



(a) Percentage cost improvement (b) Standard deviation of mismatch costs

Figure 3.5: Effects of nonlinearity for $SL = 80\%$ and $cv_{noise} = 0.05$

We observe that for this "low noise"-setting, both models outperform SAA by up to 80%. Intuitively, these large overall improvements can be attributed to the relatively low amount of additional noise we introduce into our experiment: a large portion of the demand can be explained by the causal feature-demand relationship, which SAA is, by definition, not able to exploit. It is not surprising to see that in the genuinely linear case (for $p = 0$): $\delta_{\text{TBR-NV}} < \delta_{\text{LQR-NV}}$, i.e., the linear model is superior compared to TBR-NV. Linear feature-demand relationships and low levels of noise are obviously very

favorable conditions for LQR-NV. However, with increasing p, i.e., an increasing deviation from the purely linear feature-demand relationships, the performance gap between LQR-NV and TBR-NV diminishes, and for p-values above ≈ 0.6, TBR-NV leads to higher cost improvements than LQR-NV. It is interesting to see that TBR-NV's performance is very robust towards increasing levels of nonlinearity – independent of the value of p, TBR-NV leads to very stable cost improvements of $\delta_{\text{TBR-NV}} \approx 50\%$.

This behavior under increasing nonlinearity of the feature-demand relationship can be attributed to the specific structural characteristics of the two models. First of all, it seems natural that trying to fit a linear model to a nonlinear structure leads to worse results as the relationships become more nonlinear. This is also reflected in the standard deviations – as the nonlinearity increases, the model uncertainty, i.e., the inability of the model to approximate the (actual) nonlinear relationships with a linear model, increases as well. This can also be observed from the larger standard deviations of $C_{s,\text{LQR-NV}}$ as depicted in Figure 3.5b. On the other hand, the rather simple approach of TBR-NV to partition the feature space and find piece-wise constant decisions for each partition works well also for nonlinear feature-demand relationships.

As a result we conclude that TBR-NV is very robust to changes in the specific form of the feature-demand relationship. Furthermore, even in this very simplified experiment – without complex feature interactions in the data structure – LQR-NV proves to be rather sensitive to moderate deviations from the linear structure.

To evaluate how exogenous variables impact the models' performance, we carried out experiments under varying levels of our control parameters cv_{noise} and SL. Figure 3.6 compares the cost improvement of LQR-NV and TBR-NV over SAA for $cv_{noise} \in \{0.05,\ 0.25,\ 0.5,\ 1\}$. The results suggest that with additional noise, the influence of the direct feature-demand relationship diminishes; as a result, the cost improvements of both LQR-NV and TBR-NV decrease as cv_{noise} increases. At $cv_{noise} = 1$, LQR-NV and TBR-NV lead to almost the same costs as SAA because demand realizations are predominantly determined by its (purely) random component. At this level of noise

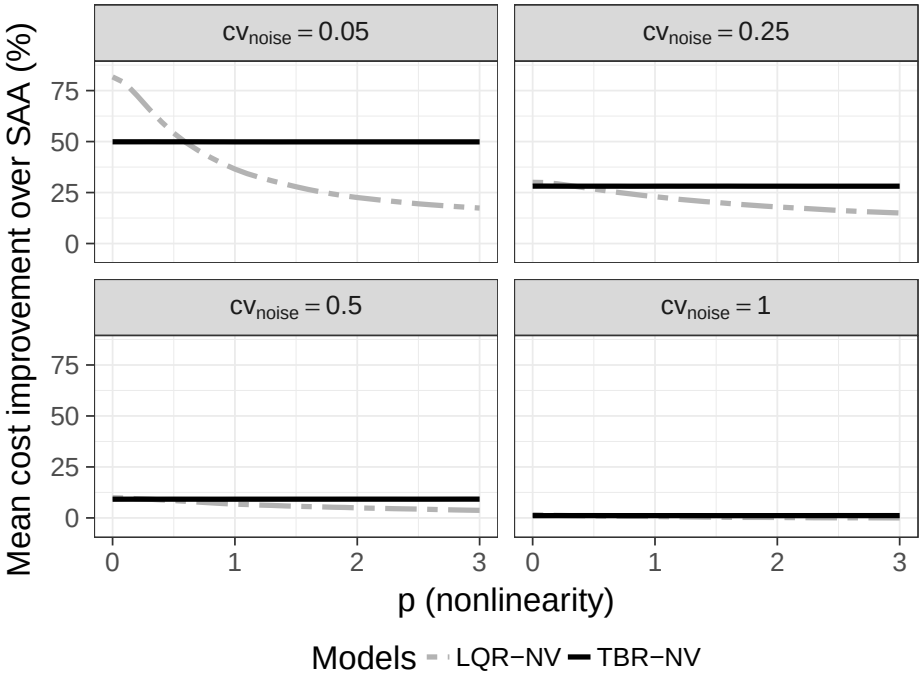


Figure 3.6: Percentage cost improvement depending on the level of nonlinearity for a 80% service level and different levels of noise (cv_{noise}).

the portion of demand that can be explained by the feature-demand relationship is so small that LQR-NV and TBR-NV only lead to marginal improvements compared to SAA (the approach that ignores features). For moderate ($cv_{noise} = 0.25$) and relatively high ($cv_{noise} = 0.5$) levels of noise, however, we obtain similar structural insights as for low levels of noise ($cv_{noise} = 0.05$):

1. LQR-NV performs better than TBR-NV up to a level of nonlinearity of $p \approx 0.5$, but is outperformed by TBR-NV when p exceeds this threshold level. Of course, for higher levels of noise, these effects are less pronounced.

2. TBR-NV's performance is again very robust with respect to changes in the nonlinearity, albeit on a lower overall level, as cv_{noise} increases.

As explained in Section 3.2.2, a specific property of LQR-NV and TBR-NV is that they directly prescribe inventory quantities based on the available

demand and feature data *as well as* the overage and underage costs from which the service level is determined as $SL = c_u/(c_u + c_o)$. So far, we compared the performance of our models for $SL = 0.8$, that is for a ratio of underage costs to overage costs of 4 to 1. In many practical instances, this ratio may be higher and decision makers will pursue higher service levels. Thus, the service level is a relevant (exogenous) parameter for our models and we want to explore the robustness of our results with respect to variations in SL.

To this end we compare the performance of LQR-NV and TBR-NV for varying service levels (SL = 0.5, 0.8, 0.95 and 0.99) at a fixed level of noise ($cv_{noise} = 0.25$). The corresponding results are reported in Figure 3.7.

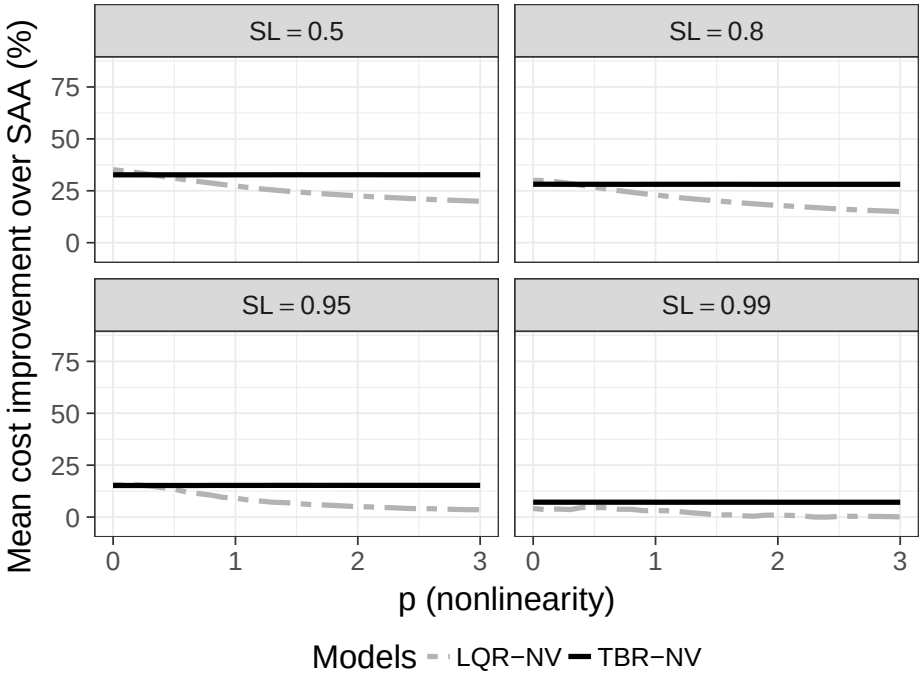


Figure 3.7: Percentage cost improvement depending on the level of nonlinearity for $cv_{noise} = 0.25$ and different service levels (SL).

The results presented in Figure 3.7 suggest that with increasing cost asymmetry, the performance of both LQR-NV and TBR-NV decrease relative to SAA. Moreover, the total mean costs decrease in SL for all of the models. This can be explained by our approach of normalizing underage

and overage costs. In appendix B.1 we show that the expected mismatch costs for SAA under a uniform demand distribution $D \sim \mathcal{U}(a, b)$ is given by $E[C_{SAA}(SL)] = SL(b-a)(1-SL)$ which (for $SL > 0.5$) decreases in SL. Beyond this shift in total mismatch costs, our structural findings are robust to changes of the cost asymmetry. However, we note a decrease in the performance gap between LQR-NV and TBR-NV for high service levels.

This can be explained by the internal structure of the specific models: The more asymmetric the cost structure, the more important potential outliers become in the partitioned leaf nodes of TBR-NV. Consider, for example, $SL = 0.99$. In this case the highest observation per leaf determines the inventory quantity independent of the distribution of the remaining instances assigned to a particular leaf. Hence, the inventory quantities are determined by the highest quantity per leaf, which may well be outliers compared to the remaining instances. Similarly, the (global) LQR-NV approach shifts the linear quantile regression line to the corresponding service level quantile—that is, for high service levels it is also mainly driven by high demand observations from the past. Thus, both approaches lead to similar inventory quantities that are predominantly driven by high demand realizations in the past. This explains the small differences in the performance of LQR-NV and TBR-NV for (very) high service levels.

In conclusion, the results of this first set of simulation experiments suggest that LQR-NV outperforms TBR-NV in settings with homoscedastic uncertainty and a linear feature-demand relationship, which is in line with our conjecture 1. With increasing nonlinearity, the performance of LQR-NV decreases. For moderate to high levels of nonlinearity TBR-NV leads to superior performance compared to LQR-NV (and, of course, SAA). All of our results provide strong evidence for the fact that TBR-NV is able to deal with nonlinearity in a very efficient way and that its results are very robust to varying degrees of nonlinearity in the feature-demand relationship.

3.3.3 Influence of feature-dependent uncertainty on model performance

In this section we examine the influence of feature-dependent uncertainty on the performance of LQR-NV and TBR-NV. More specifically, we introduce heteroscedastic noise into the learning data and control the level of heteroscedasticity through the parameter γ as explained in Section 4.4.2. Figure 4.2 displays the performance improvements of LQR-NV and TBR-NV compared to SAA for increasing levels of heteroscedasticity and varying levels of noise for a linear feature-demand relationship ($p = 0$) at $SL = 0.8$.

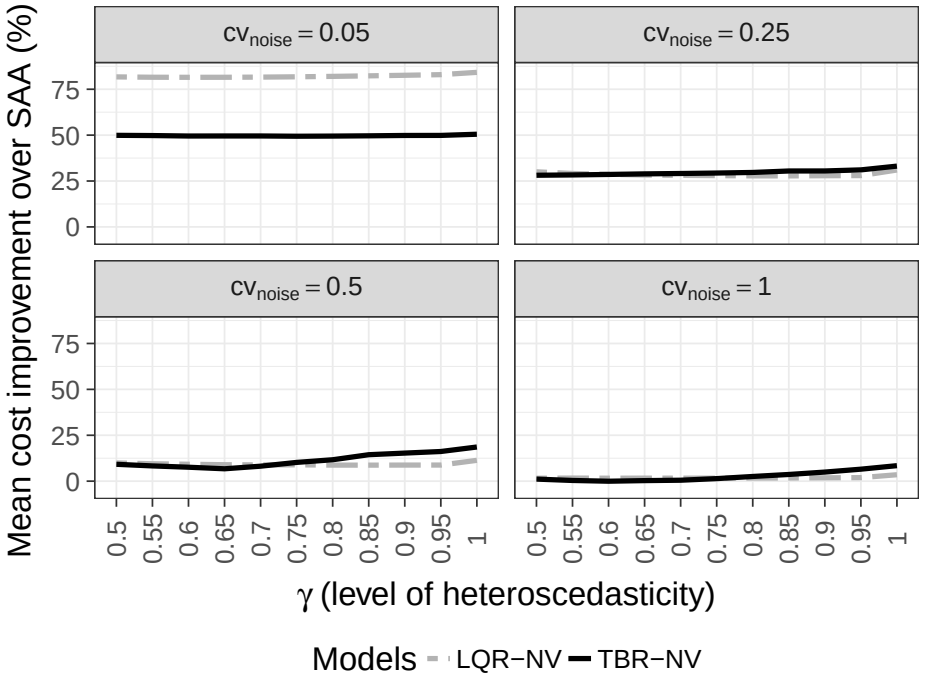


Figure 3.8: Percentage cost improvement depending on γ (level of heteroscedasticity) in a linear demand setting at 80% service level with different levels of base noise (cv_{noise}).

For low noise ($cv_{noise} = 0.05$), we see that heteroscedasticity – as we model it – has no visible influence on the relative performance of both models. This is rather intuitive because at this low level of noise it does not make much of a difference for the overall performance whether noise is homoscedastic or

heteroscedastic.

We observe that at higher levels of noise and for strong heteroscedasticity ($\gamma > 0.75$), TBR-NV outperforms LQR-NV, although the underlying feature-demand relationship is still linear. This supports our conjecture 2 where we presume that the performance of TBR-NV compared to LQR-NV improves with the level of heteroscedasticity. However, we also see that for $\gamma < 0.65$ the improvement of TBR-NV slightly decreases as γ increases. Hence, TBR-NV can only make use of this particular type of heteroscedasticity if it is high enough.

In line with the results presented in the previous section we see that a higher level of base noise (e.g. $cv_{noise} = 0.5$) leads to a lower difference in the performance of both LQR-NV and TBR-NV compared to SAA. As highlighted before, the reason for this is that the higher noise mainly obfuscates the feature-demand relationship, which does not matter for SAA, but reduces the performance of both feature-based approaches. This reduces the potential advantage of TBR-NV in settings with heteroscedastic uncertainty: for low uncertainty, the heteroscedasticity has no effect, while for high uncertainty the effect is drastically diminished due to lower overall performance.

Similar to the previous section, we also analyze the robustness of our results with respect to the choice of the service level. For this, we run the simulation for three additional service levels $SL = 0.5,\ 0.95$ and 0.99. Figure 3.9 shows the performance of LQR-NV and TBR-NV relative to SAA with respect to γ for $SL = 0.5,\ 0.95$ and 0.99. The structural effects are very similar to those we observed in the previous section: we see that in general a higher cost asymmetry leads to a lower difference in mean costs of both models compared to SAA. With respect to the influence of heteroscedasticity we do not see any meaningful structural impact (compared to the results presented in Figure 4.2) when varying the service levels.

We conclude that heteroscedasticity (at least as we model it) is a performance driver, but with a much lower impact than the nonlinearity of the feature-demand relationship. Nonetheless, feature-dependent heteroscedasticity slightly favors TBR-NV: except for perfectly linear feature-demand relationships with almost no noise, TBR-NV is able to obtain similar or better

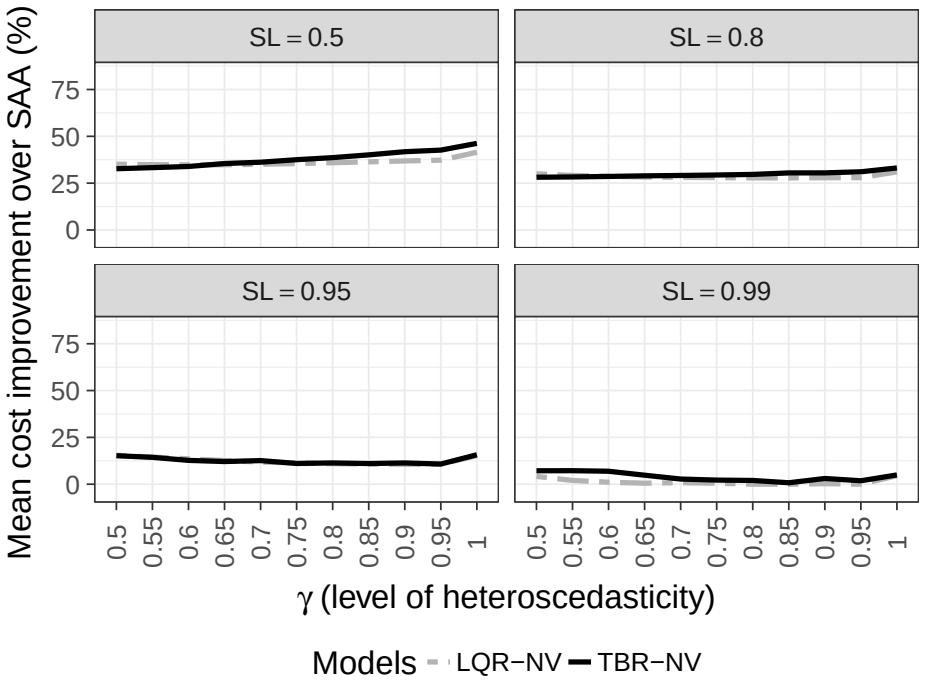


Figure 3.9: Percentage cost improvement depending on p (level of heteroscedasticity) for different service levels (SL) in a linear demand setting with a base noise level of $cv_{noise} = 0.25$.

results than LQR-NV.

Finally, we examine the influence of combined nonlinearity and feature-dependent uncertainty. We find that in our setting the effects of nonlinearity and heteroscedasticity are independent and additive. As we saw in Figure 3.6, the performance of TBR-NV is virtually independent of the level of nonlinearity. Hence, the effect of heteroscedasticity we deducted from Figure 4.2 is also unaffected by increasing nonlinearity. For LQR-NV these effects are reversed: heteroscedasticity has virtually no effect on the performance of LQR-NV, hence the strong effect of nonlinearity is unaffected by heteroscedasticity. In Figure B.1 in Appendix B.2 we visualize these results.

3.4 Study 2: Prescriptive analytics at Yaz

In the previous chapter we studied LQR-NV and TBR-NV in a controlled experiment to obtain structural insights on how the feature-demand relationships influence the performance of the two competing models. While this enabled us to study important individual effects (nonlinear feature-demand relationships; heteroscedasticity) in isolation, it also required major simplifications in comparison to most real-world settings. For example, we considered only three features with a known relationship with demand, while in reality decision makers will oftentimes be confronted with hundreds or even thousands of features which may or may not have predictive power. Also, we did not consider complex relationships between features, which are common in practice, and instead assumed specific functional forms to model the relationships between features, level demand and uncertainty (see our discussion in conjunction with Figures 3.1 and 3.2).

We now evaluate the performance of LQR-NV and TBR-NV for a real-world inventory management problem which exhibits (simultaneously) a number of practically relevant characteristics (many features, potentially complex nonlinear, but unknown, relationships) we deliberately did not consider in our controlled simulation experiment. On the one hand, we are interested in whether or not we find similar performance improvements relative to SAA and similar results when comparing the two models. On the other hand, we use this realistic setting to evaluate the two main aspects of usability which we discussed in Section 4.4: the impact of the amount of learning data and the need for, and importance of, feature engineering. Due to the overwhelming complexity in the required simulation design, such aspects of usability cannot effectively be evaluated in a controlled experiment, but require a real-world dataset to obtain valid insights. In this study, we analyze a dataset from a fast casual restaurant chain named Yaz. Yaz offers a limited range of main ingredients, but with a broad variety of mostly oriental-style preparations. Because the ingredients are perishable, Yaz has to decide upon their inventories on a daily basis. Thus, Yaz is facing Newsvendor problem.

In the following sections we first provide an overview of the data sources

we used and the features we derived from the available data. Thereafter, we describe the setup, i.e., the logic used to compare the different approaches. Finally, we present our results regarding the relative performance of both approaches and discuss aspects regarding the usability of these models in our real-world application.

3.4.1 Data

Yaz provided us with sales data from their main flagship restaurant in Stuttgart, Germany, for the period 2013/09/27 to 2015/11/09. The restaurant manager's observation (assumption) was that the weather has a strong influence on demand. Hence, we obtained weather data from databases of the German Meteorological Service and aggregated it to a daily level to reflect the same level of granularity of a potential weather forecast for the next day. Despite the fact that we used actual weather information that would not be available to the decision maker instead of a forecast, we believe that the features we derived from this data would be very similar to a weather forecast for the next day.

Based on this raw data, we derived 167 features for each product by extracting structural information about the underlying time series, e.g., the rolling mean demand for the same weekday. Table 6.4 provides an overview of the most important features. An overview of the entire set of features can be found in the appendix.

The demand structure of different products varies significantly in terms of the mean demand and the coefficient of variation. For this reason we report the model performance for three selected products that exhibit different demand structures (see Table 3.4).

As illustrated in Figure 4.4, the aggregated (and smoothed) demand is nonstationary over time, ruling out a basic Newsvendor solution to this inventory management problem that is based on a fixed demand distribution. Besides this nonstationarity over a longer period of time, we also observe nonstationarity in the short term. Figure 3.11 displays the mean demands and coefficients of variation per weekday for each product. It is straightforward to

Source	Feature	Description
Time Series	Total_M_2W	Average aggregate demand (for all products) on same weekday for the last two weeks
	Steak_M_3W	Average aggregate demand (for individual products) on the same weekday for the last three weeks
	Total_D1	Aggregate demand (for all products) the day before
	Is_Dec	Is December
	Is_Sat	Is Saturday
	Is_Outlier	Is special day (Event, Holiday, etc.)
Weather	Temp_D2	Air temperature two days ago
	Temp_M_4D	Average Air temperature over last four days
	Sun_M_5D	Average duration of sunshine over last five days

Table 3.3: Examples of relevant features for the product Steak

see that neither the mean demands nor the coefficients of variation are constant across different days of the week. Even though these variations might be partly explained by other endogenous features than the weekday, we can consider these differences as an indicator for the existence of heteroscedasticity, i.e. feature-dependent uncertainty.

In the past, the restaurant manager intended to have all products available at any time. Thus, inventory levels were very high and only on rare occasions Yaz faced stockouts. During the period we consider in our analysis Yaz experienced a stockout on 1.6% of the days, that is on 98.4% of the days all six main ingredients (Calamari, Steak, Lamb, etc.) were available. For the restaurant this high level of availability had severe negative consequences because it lead to high obsolescence. For our study, however, the high level of product availability has an advantage: sales closely represent actual demand

Statistic	Calamari	Steak	Lamb
Mean demand	4.3	22.5	31.6
Coefficient of variation	0.67	0.44	0.40

Table 3.4: Mean demand and coefficient of variation of different products

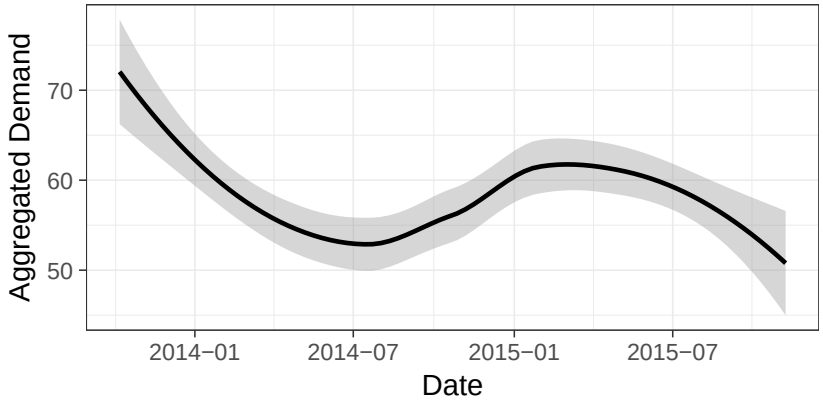
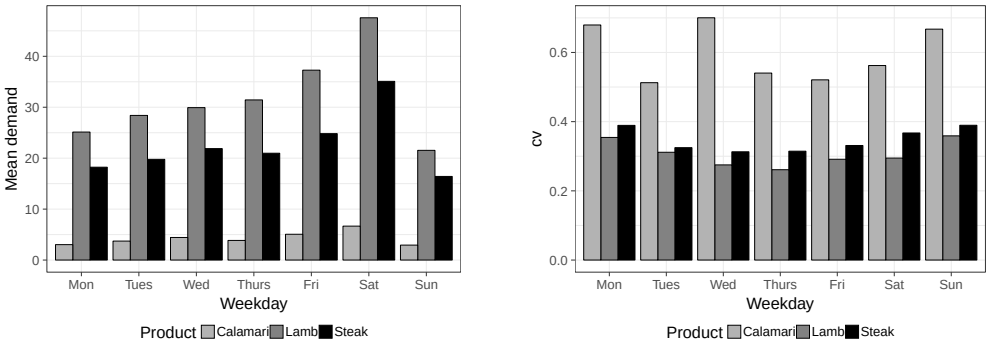


Figure 3.10: Evolution of smoothed aggregated demand over time

and we do not have to deal with issues related to censored demand.

3.4.2 Evaluation procedure

After cleaning and preprocessing the raw data as described in the previous section, we obtain a training dataset $\mathcal{T}_{N_{Yaz}} = \{(d_i, \mathbf{x}_i), i = 1, ..., N_{Yaz}\}$ with $N_{Yaz} = 760$ demand observations. We use these instances similarly to our approach in Section 4.4.2 but repeat the evaluation procedure for each single day within our evaluation period $t \in \{51, \ldots, 760\}$: Starting with day $t = 51$, we calibrate each model with the $t-1$ previous observations, hence by learning the model on the training data $\mathcal{T}_{t-1} = \{(d_i, \mathbf{x}_i), i = 1, ..., t-1\}$. Once the LQR-NV and TBR-NV models are calibrated, each of them is fed with the feature values $\mathbf{x}_t$ for day t in order to determine the prescribed quantity $q^*_{t,m}$ per model $m \in \{\text{LQR-NV, TBR-NV, SAA}\}$. The resulting cost $C_{t,m}$ is then calculated as described in (4.24). This procedure is repeated for all days in the evaluation period. The resulting costs are then averaged for the whole



(a) Mean demand

(b) Coefficient of variation

Figure 3.11: Demand characteristics per weekday

evaluation period to

$$\bar{c}_m = \frac{1}{(760 - 51 + 1)} \sum_{t=51}^{760} C_{t,m}$$

and reported as δ_m, the percentage improvement over SAA, as described in (4.26). We emphasize that the first 50 days are used exclusively for training the prediction models and are not considered in the same way for the performance evaluation. As both competing models are based on machine learning techniques, we argue that otherwise, without a minimum amount of training data, neither of the two methods would provide valid results.

3.4.3 Results

This chapter presents the main results of our application of LQR-NV and TBR-NV to the inventory management problem of Yaz. First, we compare the percentage improvement of the two approaches with respect to our SAA benchmark. Then, in the subsequent sections, we will address the main aspects of usability in more detail.

Performance of LQR-NV and TBR-NV

Figure 3.12 displays the percentage cost improvements of LQR-NV and TBR-NV relative to SAA for a service level of 0.8. We observe that both feature-

based approaches lead to considerable performance improvements. In line with our findings in Section 3.3 we observe lower performance improvements when uncertainty is higher. For Calamari – the product with the highest CV – we achieve substantially lower profit improvements than for Lamb or Steak, the two products with a much lower CV. This relationship also holds for Lamb vs. Steak: for Lamb, the product with the lowest CV, we observe the highest performance improvements. Of course, we have to be careful in drawing strong conclusions; in contrast to the results of our controlled experiment, we cannot rule out that the performance differences are caused by product-specific feature-demand relationships.

In essence, we face the typical problems that occur when trying to evaluate the performance of complex machine learning techniques that are applied to large datasets: it is virtually impossible to prove, in a rigorous way, which factors explain performance differences. This issue is also pertinent when trying to explain our next observation: TBR-NV outperforms LQR-NV across all products. In Section 3.3 we saw that TBR-NV performs better than LQR-NV when the feature-demand relationship is nonlinear. Thus, our results presented in Figure 3.12 may be an indication for nonlinear relationships between the demand for the different products (Calamari, Steak, Lamb) and the 167 features. We cannot, however, rule out that the superior performance may be caused by other factors, e.g., interaction effects between different features, which we did not consider in our controlled experiment. However, the results do present strong evidence for the fact that – without additional feature-engineering – a tree-based approach is more appropriate when nonlinear and/or complex interactions among features exist. We will further explore this issue in Section 3.4.3, where we perform additional feature engineering for LQR-NV.

Influence of the amount of available learning data

In our controlled experiment we evaluated the performance of LQR-NV and TBR-NV based on the premise that a large set of learning data is available. More specifically, we assumed that a training set with 1000 instances of de-

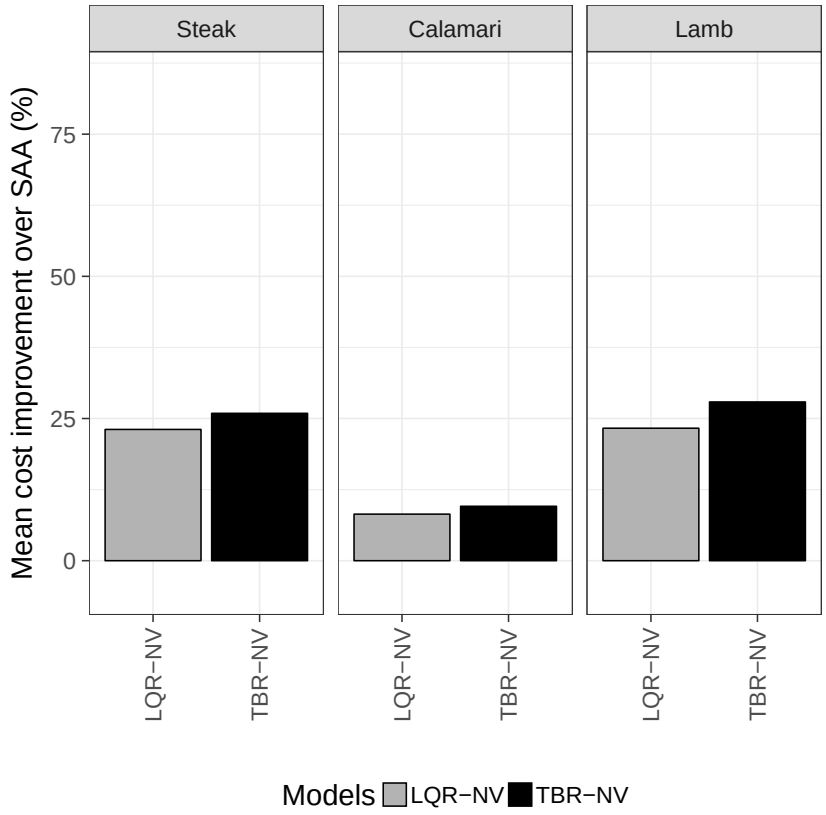


Figure 3.12: Percentage cost improvement relative to SAA for $SL = 0.8$

mand realizations and feature values is available. While this is convenient and appropriate for analyzing structural effects, it may not reflect practically relevant problem settings. In practice, companies may not have access to such a long history (approximately three years), or historical data may not be predictive over such a long time span. In the fashion industry, for example, demand (and feature) data from two or three years ago may have very little predictive power. Thus, for decision makers it becomes important to assess how different approaches perform when less (historical) data is available.

In this chapter we analyze the effect the available amount of learning data has on the performance of LQR-NV and TBR-NV. This is an important aspect of usability when prescriptive analytics is set into practice – simply speaking: we would like to know how long we need to collect data until we

can rely on the results of individual models.

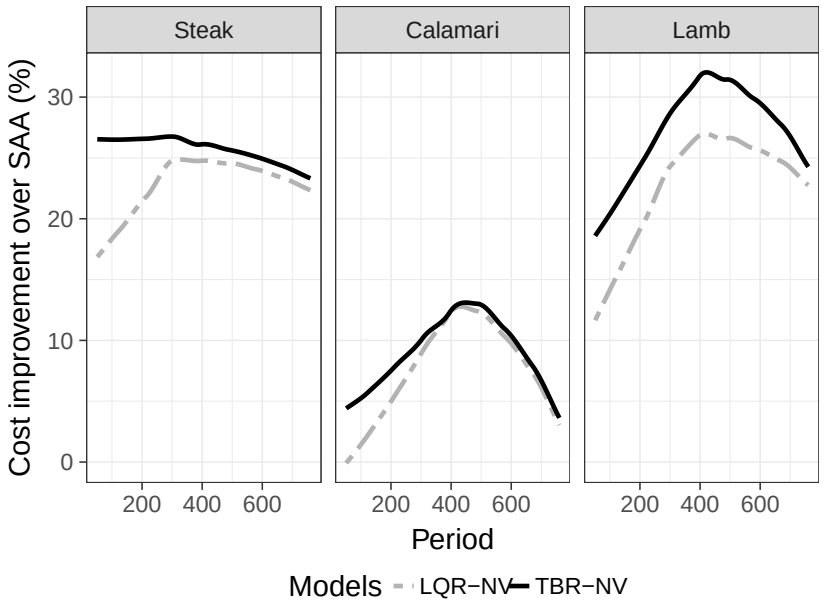


Figure 3.13: Smoothed percentage cost improvement over time for Calamari, Steak and Lamb at 80% service level

Figure 3.13 shows how the mean costs evolve over time as we evaluate period after period for our models starting with period 51. We used LOESS regression [Cleveland, 1979] to smooth the scatterplot of the mean costs in individual periods. Across all products both models can hold (TBR-NV for Steak) or significantly improve their performance compared to SAA within the first (additional) 300 periods (i.e, the first year in which we record the relevant data). Hence, both approaches benefit from additional data. In the initial phase with limited amount of learning data we see that TBR-NV seems to have an advantage over LQR-NV. This advantage diminishes with additional learning data. That is, in this particular setting and if only a limited amount of learning data is available, TBR-NV seems to be preferable over LQR-NV.

This finding contradicts our Conjecture 3 where we presumed that LQR-NV requires less learning data due to its global approach that bases the decision on the entire available set of learning data. The counterintuitive results

can be explained by the way TBR-NV selects the relevant features. For TBR-NV the most important features with only 50 observations of learning data are very similar to the most important features of the model when given the full set of data. For LQR-NV, with only 50 observations, the set of the most important features mainly consists of features that have a much smaller impact compared to the model learned on the full dataset. Hence, the feature selection of TBR-NV can adapt dynamically to the amount of given learning data and determine the most important features from a large set of features when given only a limited amount of learning data. In contrast, LQR-NV does not adapt dynamically to the amount of learning data, resulting in inferior model performance.

For the products Calamari and Lamb we see an increase in the cost improvements around period 500. This can mainly be explained by an increase in demand the restaurant was facing at this time as indicated in Figure 4.4. Hence, figure 3.13 shows that LQR-NV and TBR-NV can handle such non-stationarity in the underlying demand process better than SAA and similarly well compared to each other.

Influence of feature engineering

As discussed in Section 4.4.3, the reason for the superior performance of TBR-NV is most likely caused by nonlinear and/or complex relationships between features and demand, which LQR-NV in its basic (strictly linear) form cannot handle well. Ban and Rudin [2018] point out that LQR-NV can also deal with nonlinear and more complex relationships if the basic features are transformed, that is through feature engineering. A nonlinear function can, for example, be approximated locally by its Taylor expansion as long as the function is analytic. The parameters of the Taylor expansion can then be added as additional features to the linear objective function. These features may also include cross-product terms of the original features and can, thus, also capture interaction effects. This feature engineering, however, is rather challenging: A Taylor expansion, for example, only provides local approximation and the feature space needs to be split in order to obtain a piecewise linear

model. This leads to piecewise polynomial fitting, which can be accomplished by the standard linear least squares approach if we know where to split the feature space [Montgomery et al., 2007].

Unfortunately, we typically neither know where to split nor how many splits are required to achieve good performance. This is a major drawback because determining the variable along which to split in combination with the respective split value leads to a nonlinear regression problem [Montgomery et al., 2007]. Especially in "big data"-regimes where the number of features is high (thousands of attributes are not uncommon), manually integrating expert knowledge about individual features to model nonlinearity with a linear model as suggested in Montgomery et al. [2007] is likely to turn out infeasible from a practical point of view. Moreover, such manual pre-processing contradicts the fundamental idea of machine learning, that is, letting the algorithm find the relevant relationships from the data.

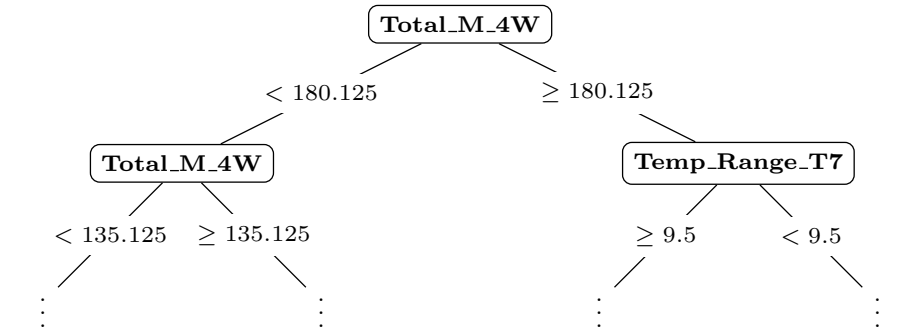
The problem is how to determine appropriate data preprocessing operations upfront and how to engineer suitable features that capture the most important feature-demand relationships that are unknown a priori. Even for our Yaz dataset, which "only" contains 167 features, it is extremely difficult to carry out successful manual feature engineering. Of course, some relationships and their structure are easy to detect without much effort, e.g., how the level demand differs across weekdays (see Figure 3.11a). However, determining, a priori, which weather-related features are relevant, identifying and modeling their presumably nonlinear relationship with the demand level as well as their interactions with other features (e.g. weekdays) and converting this into suitable features for a LQR-NV model is a daunting task. Despite these problems we want to explore if performance of LQR-NV can be improved through manual feature engineering. The main reason is that the results of such an analysis can potentially foster a better understanding of the factors leading to the performance differences between LQR-NV and TBR-NV we observed in Figure 3.12. The problems associated with such an analysis are that the results depend on how well we manually engineer the features – we have no means to identify the best set of manually engineered features and an exhaustive enumeration is infeasible – even for the limited number of features in our

dataset. To circumvent this problem, we use the information we obtain from LQR-NV to engineer the basic features in order to derive an additional set of features per product.

More specifically, we use a decision tree model to select relevant features, to find splits to build a partially linear model, and to determine interaction terms, i.e. combinations of individual features. Decision tree learning provides implicit feature selection since only the most meaningful features show up in the tree. Furthermore, decision trees partition the parameter space in groups of observations with similar demands. We replicate these splits by also splitting the corresponding features into separate features, one containing values above the split value and one containing the values below the split value. Figure 3.14 depicts this splitting procedure for the root node in the tree for one individual product (Steak).

Through this procedure, we are able to incorporate complex feature-demand relationships of the kind that a feature is positively correlated with demand in its lower range and negatively correlated in its upper range into an enhanced LQR-NV model. In order to capture interaction effects of individual features we construct interaction terms by multiplying the split variable in the root node with the split attributes in the first level of the subtrees. Hence, by resorting to the information from decision trees, we already consider three important aspects for our new, improved features for LQR-NV: Feature selection, piecewise relationships and interaction terms. Thus, we incorporate manually the main information that TBR-NV uses as part of its built-in logic.

We then repeat our evaluation logic for LQR-NV with these new features. Figure 3.15 depicts the results for our three products for a service level of 0.8. We see that with the optimized features LQR-NV can achieve considerably higher cost improvements. For two out of three products, LQR-NV with the optimized features now outperforms TBR-NV. This implies that indeed non-linear feature-demand relationships and complex interactions were responsible for the initial performance differences we observed in Figure 3.12. Through manual feature engineering we can enhance LQR-NV to an extent that this approach leads to (slightly) better results than TBR-NV (at least for two of our products). Even with feature engineering, the performance of LQR-NV



Total_M_4W
88
191
113
136
138
141
130

Total_M_4W_1	Total_M_4W_2
88	0
0	191
113	0
136	0
138	0
141	0
130	0

Figure 3.14: Piecewise splitting of features exemplified for the product Steak and the first split

for Calamari is lower than the performance of TBR-NV. For this product, the difference in the coefficient of variation between weekdays is largest (see Figure 3.11b). This is an indicator for more heteroscedasticity in the data of Calamari which may explain why TBR-NV is still better than LQR-NV, even after additional feature engineering.

In conclusion we find that additional feature engineering can indeed improve the performance of LQR-NV. Achieving this improvement does, however, require additional effort for the manual feature engineering. This effort can be very high, especially in settings where many features are relevant and the upfront advantage of TBR-NV approaches is larger, i.e. there is stronger or more complex nonlinearity or heteroscedasticity in the data. Hence, the decision maker has to trade-off the additional effort/cost for feature engineering with the (a priori unknown) benefits this may provide compared to TBR-NV that provides consistently good results and requires little effort.

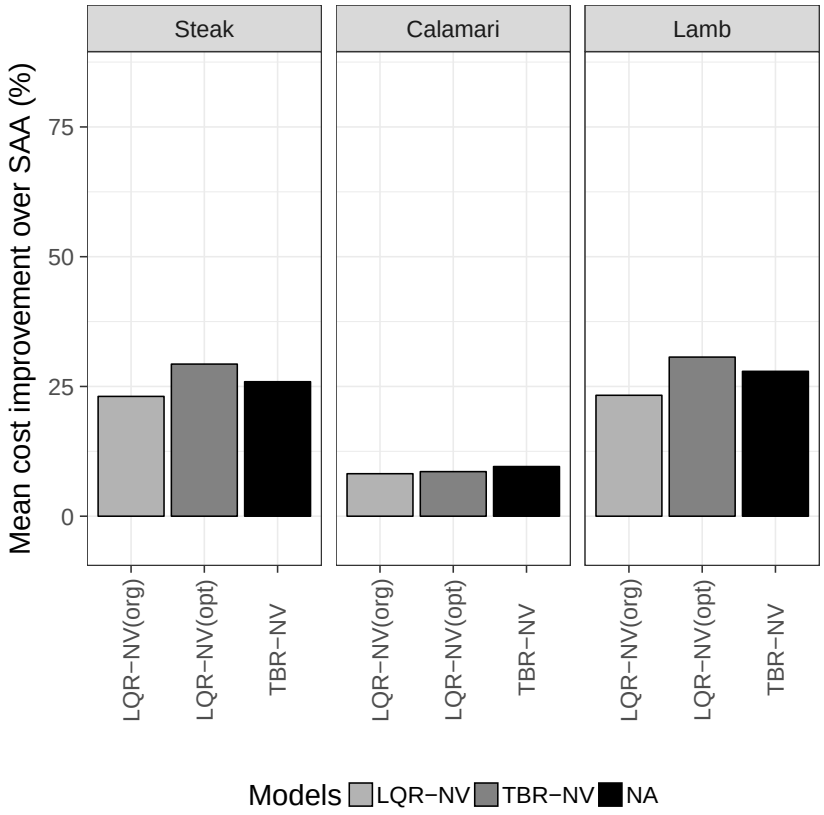


Figure 3.15: Percentage cost improvement for LQR-NV with standard feature data [LQR-NV (orig)] and and improved feature data [LQR-NV (opt)] versus the TBR-NV for $SL = 0.8$

3.5 Conclusion and future work

In this paper we examined the performance of two novel data-driven models (LQR-NV and TBR-NV) that directly prescribe inventory decisions based on historical demand observations and available feature data. In order to analyze the factors driving the performance and versatility of the two competing models, we conducted two studies. First we carried out a controlled simulation experiment to analyze the impact of two important properties of the feature-demand relationship, that is the degree of nonlinearity as well as the amount of systematic noise within the demand uncertainty. Our second experiment applied the two competing models to a real-world dataset. In this inventory

management problem for a fast casual restaurant chain, we also compared the performance of the two models, but focused mainly on usability aspects, i.e., the actual effort required to apply both models to realistic problem instances and to obtain reasonable results.

From our simulation experiment we found (not surprisingly) that LQR-NV performs well in settings where the feature-demand relationship is linear and the additional noise is not excessively high. However, as soon as the feature-demand relationship deviates from a linear pattern, the performance of LQR-NV decreases substantially while the performance of TBR-NV is robust to changes in the form of the relationship. Second, we found that systematic noise also favors TBR-NV but on a considerably lower level. From these results we conclude that in situations in which the structure of the feature-demand relationships is not known to be predominantly linear – which can be assumed to be the typical case in practice – we recommend the much more robust TBR-NV.

The results of our second study provide further support for this recommendation. We saw that TBR-NV outperforms LQR-NV for all of the products included in our study. In contrast to one of our conjectures we find that TBR-NV provides better results even if limited data to learn the models is available. Furthermore, we evaluated the effect of additional manual feature engineering. We saw that with additional structural information about the feature-demand relationships we can engineer features that improve the performance of LQR-NV. To do so, however, we used the structural information from the competing tree based approach, which is a rather artificial way to perform feature engineering, because this information would typically not be available upfront.

It is hardly possible to quantify the performance impact of additional feature engineering ex ante – even under very favorable conditions we observed very different effects across the different products included in our study. Thus, we conclude that TBR-NV leads to good and robust performance across a wide range of scenarios and requires much less effort for feature engineering. For this reason, a decision maker with no or little information about the actual feature-demand relationship should prefer the more robust TBR-NV

over LQR-NV. Of course, the decision maker should be aware of the fact that LQR-NV can lead to better results when the right data structure is present and/or good information that can be used for feature engineering is available. Additional feature engineering, however, requires substantial effort and ex ante it is not clear whether this additional effort will turn out to be worthwhile.

The study presented in this paper has a number of limitations. Despite the fact that we made a substantial effort to obtain generalizable results, we cannot guarantee that they hold true under all practically relevant settings. For our simulation experiment we made a number of simplifications in order to obtain tractable results and to identify and study causal effects. For example, we did not study the effect of complex interactions of features and did not include a larger number of features with less predictive power. Also, we cannot rule out that our particular experimental setup was more favorable for TBR-NV than for LQR-NV, e.g., in the way we modeled systematic noise. However, we believe that our experiments did bring out important results regarding the performance drivers of the different models that can be used as a basis for further and more extensive analyses. Moreover, our analyses focused only on the performance of two particular (albeit distinct) models in a Newsvendor setting. Of course, there may be alternative models that lead to better performance, and we do not know how the two models perform when being applied to more involved inventory management problems (e.g., when multiple periods have to be considered, or when demand is lumpy). We perceive our work as a starting point for a rigorous and more in-depth evaluation of new and very promising techniques that integrate feature data ("big data") into inventory management models.

4 Machine learning for inventory management: Analyzing two concepts to get from data to decisions

We analyze two fundamentally different concepts to considering data for planning decisions using the example of a newsvendor problem in which observable features drive variations in demand. Our work contributes to the extant literature in two ways. First, we develop a novel joint estimation-optimization (JEO) method that adapts the random forest machine learning algorithm to integrate the two steps of traditional separated estimation and optimization (SEO) methods: estimating a model to forecast demand and, given the uncertainty of the forecasting model, determining a safety buffer. Second, we provide an analysis of the factors that drive difference in the performance of the corresponding SEO and JEO implementations. We provide the analytical and empirical results of two studies, one in a controlled simulation setting and one on a real-world data set, for our performance evaluations. We find that JEO approaches can lead to significantly better results than their SEO counterparts can when feature-dependent uncertainty is present and when the cost structure of overage and underage costs is asymmetric. However, in the examined practical settings the magnitude of these performance differences is limited because of the overlay of opposing effects that entail the properties of the remaining uncertainty and the cost structure. [10]

[10]This paper is coauthored by Jan Meller.

4.1 Introduction

We analyze two fundamentally different concepts to consider data for inventory-management problems in which observable features drive variations in demand. In lockstep with the ever-increasing availability of data, research attention in the operations management community has shifted from approaches that rely on historical demand time-series to methods that can consider auxiliary data that may drive variations in demand [Feng and Shanthikumar, 2018]. Studies that use web traffic data to predict hotel demand [Yang et al., 2014], consider online clickstream data to forecast demand for a door manufacturer [Huang and van Mieghem, 2014], and derive daily demand from an analysis of social media data [Cui et al., 2018] are only a few examples of the use of such auxiliary data for planning decisions.

In the classical inventory-control literature, such demand forecasts are typically the first step in making inventory decisions. Then the decision-maker considers the forecast uncertainty (e.g., the empirical distribution of forecast errors) and the costs for underage and overage. More specifically, the decision-maker sets an inventory level to minimize the expected inventory-mismatch costs by balancing expected overage costs for leftover inventory with expected underage costs for stock-out situations. The literature refers to this concept as *separated estimation and optimization (SEO)* [cf. Ban and Rudin, 2018]. In contrast to sequentially estimating a demand prediction model and optimizing inventory decisions based on the former's inputs, another literature stream [e.g., Akcay et al., 2011, Beutel and Minner, 2012, Oroojlooyjadid et al., 2016, Ban and Rudin, 2018, Bertsimas and Kallus, 2019] promotes integrating these two steps. Their models have in common that the expected mismatch costs of the final inventory decision are already considered for estimating the model, resulting in a single optimization problem that learns cost-optimal decisions from historical data. In line with Akcay et al. [2011], we refer to this concept as *joint estimation-optimization (JEO)*.

A series of articles [Liyanage and Shanthikumar, 2005, Chu et al., 2008, Ramamurthy et al., 2012, Lu et al., 2015] has shown that a class of integrated approaches called operational statistics dominates SEO methods for newsven-

dor settings with parametric demand distributions. However, while intuitively attractive because of they do not lose information between the prediction and optimization stages, JEO approaches still lack proof of their superiority over SEO approaches in a data-rich environment with non-parametric, feature-driven demand. Most of the existing studies show that one JEO approach outperforms relatively simple benchmarks, such as sample average approximation, but to the best of our knowledge, a rigorous examination of SEO and JEO approaches using the same underlying machine-learning technique and the same raw data is lacking. Only in two studies do we find results that provide a fair comparison between the corresponding SEO and JEO approaches. In one, Ban and Rudin [2018], the linear SEO approach without regularization performs slightly better than the JEO counterpart, and in the other, Huber et al. [2019] find no significant performance difference between a JEO approach based on artificial neural networks and its SEO counterpart. For this reason, we see a research gap that calls for a rigorous examination of the performance differences between implementations of the JEO and the SEO concepts and a quantification of the performance gap in real-world application scenarios.

Our work contributes to the existing literature in two ways: First, we develop a novel JEO approach that is based on the random forest machine learning algorithm. Second, we provide a critical in-depth analysis of the structural differences and the factors that drive performance differences between corresponding SEO and JEO approaches for various underlying machine learning algorithms. We provide both the analytical insights and the empirical results of two studies, one in a controlled simulation setting and one on a real-world data set, for our performance evaluations.

After presenting the theoretical backgrounds of the SEO and JEO concepts in section 4.2, section 6.3 presents implementations with two underlying machine learning techniques: random forests, which includes our new tree-based JEO approach, and kernel optimization as a benchmark from the literature. Finally, the results of our analyses are presented in section 4.4.

4.2 Two concepts to get from data to inventory decisions

The problem of how to determine inventory targets when facing uncertain demand has been at the center of attention in operations management research for decades. In the classical stream of research, demand uncertainty is captured by parameterized probability distributions, which are often assumed to be known [e.g., Zipkin, 2000]. However, such a strong assumption is unrealistic for most practical settings, where the underlying demand distribution is usually unknown [Klabjan et al., 2013]. In many real-world situations, not only is the form of the distribution unknown, but demand is clearly not stationary, as it might be seasonal or cyclical, follow a trend, or be influenced by factors like weather, national holidays, and sales promotions. A common way to deal with such a situation is to cast information that may have predictive power into features, i.e., summarized representations of the auxiliary data. To illustrate the concept of feature-driven demand, assume an additive demand model that has two components: the demand level and an additional random component[11]. In this basic model, we assume that the demand level is deterministic and correlated to the data features, which we denote by the vector $\mathbf{x}$. The additional component ϵ internalizes all exogenous uncertainty which may also be feature-dependent. Hence, demand D can be modelled as:

$$\begin{aligned} D &= \mu(\mathbf{x}) + \varepsilon \\ \text{with } \mathbb{E}[\varepsilon] &= 0; \sigma_\varepsilon \sim \mathbf{x} \\ \mathbf{x} &\in \mathbb{R}^k \end{aligned} \tag{4.1}$$

where $\mu(\mathbf{x})$ is the function that describes the relationship between values of the features $\mathbf{x}$ and the the demand level $\mu(\mathbf{x}) = \mathbb{E}[D|X = \mathbf{x}]$. Exemplary data features that are subsumed in the vector $\mathbf{x}$ could include weekday, month, temperature, and representations of other attributes that could affect the expected demand level.

[11] This assumption is common in inventory management[cf., e.g., Nahmias, 2001].

While the function $\mu(\mathbf{x})$ is unknown in practice, we often have a data set of historical observations that consists of pairs of demand and feature values. We refer to such a set $\mathcal{T} = \{(d_i, \mathbf{x}_i), i = 1, \ldots, n\}$ as the training data set. Assuming an underlying demand model as in (4.1), we distinguish two generic concepts with which to consider the learning data $\mathcal{T}$ for making inventory decisions. We provide details about these two concepts in the subsections 4.2.1 and 4.2.2.

4.2.1 Separate estimation and optimization (SEO) with auxiliary data

SEO follows a two-step procedure: First, we estimate a demand-forecasting model to capture the relationship between the vector of data features $\mathbf{x}$ and the demand level $\mu(\mathbf{x})$. That is, we approximate the function $\mu(\mathbf{x})$ using an estimated function $\hat{\mu}(\mathbf{x})$. Since we cannot assume our model is perfect, we adjust the forecasts for uncertainty that is due to forecasting errors to obtain optimal stocking decisions [c.f. Brown, 1959, Nahmias, 2001]. For this reason, we evaluate the demand-forecasting model's prediction performance to produce a representation of the remaining uncertainty, that is, the distribution of the forecast errors[12]. The latter distribution then serves as an input to the inventory-optimization logic, which determines an additional safety stock that is calculated by trading off expected overage costs with expected underage costs. The final inventory decision then consists of both the prediction generated by the forecasting model and the safety stock.

More formally, the problem of interest is

$$q^*_{SEO}(\mathbf{x}) \in \mathcal{M}\times\mathbb{R} = \underset{\hat{\mu}(\cdot)\in\mathcal{M}}{\arg\min}\ \mathbb{E}\left[L(\hat{\mu}(\mathbf{x}), D)|X = \mathbf{x}\right] + \underset{z\in\mathbb{R}}{\arg\min}\ \mathbb{E}\left[C(z, D - \hat{\mu}(\mathbf{x}))\right] \tag{4.2}$$

where the prediction function $\hat{\mu} : \mathcal{X} \longrightarrow \mathbb{R}$ is selected from a function space $\mathcal{M}$ and maps from the set of all possible feature vectors $\mathcal{X}$ to real valued

[12]The forecast errors contain both the random component ε of the demand model and the model uncertainty when approximating $\mu(\mathbf{x})$ by $\hat{\mu}(\mathbf{x})$. For readability, we subsume both these components under ε in the following.

demands, and $L(\hat{\mu}(\mathbf{x}), D)$ and $C(z, D-\hat{\mu}(\mathbf{x}))$ are two unrelated loss functions. Typically, one would choose a symmetric loss function $L(\hat{\mu}(\mathbf{x}), D)$ like the mean squared error to generate unbiased predictions, whereas the second loss function $C(z, D-\hat{\mu}(\mathbf{x}))$ reflects specific (and presumably asymmetric) overage and underage costs as a consequence of a mismatch between the decision and actual demand.

4.2.2 Joint estimation-optimization (JEO) with auxiliary data

Despite its wide adoption in practice, the two-step SEO concept has a major drawback: By first fitting a prediction model for the demand and then optimizing the inventory decision, we have two separate optimization problems that are not necessarily congruent and so can lead to suboptimal decisions [Liyanage and Shanthikumar, 2005]. For this reason, another class of models has recently gained attention: JEO models that directly link the features with the final decision and so avoid the intermediate step of building a demand prediction model. Instead, the training of the demand prediction model and the inventory decision are combined into a single optimization problem. The underlying idea of combining statistical estimation and optimization goes back to Hayes [1969], who estimated policies from data by minimizing the *expected total operating cost*.

Bertsimas and Kallus [2019] propose a framework for JEO models and formalize the problem as:

$$q^*_{JEO}(\mathbf{x}) \in \mathcal{Q} = \underset{q(\cdot)\in\mathcal{Q}}{\arg\min}\ \mathbb{E}\left[C(q(\mathbf{x}), D)|\mathbf{x}\right], \tag{4.3}$$

where $q : \mathcal{X} \longrightarrow \mathbb{R}$ is a decision function from the function space $\mathcal{Q}$, which maps from the set of all possible feature vectors $\mathcal{X}$ to real valued decisions; and $C(q(\mathbf{x}), D)$ is the loss function that yields costs given a decision q and a realization of demand D. The main difference from SEO is that JEO is a single optimization problem whose solution is directly obtained with respect to the actual cost function $C(q(\mathbf{x}), D)$.

Several examples of JEO approaches in the literature differ primarily in the functional relationship $q^*(\mathbf{x})$ between decision and features. The contributions of Beutel and Minner [2012] and Ban and Rudin [2018] both employ linear functions $q : \mathcal{X} \to \mathbb{R} : q(\mathbf{x}) = \beta^T \mathbf{x}$ to relate a feature vector $\mathbf{x}$ of length k to the newsvendor quantity $q(\mathbf{x})$. They optimize the weights β^j for each feature from a set of learning data. Ban and Rudin [2018] also present a second JEO approach that uses kernel functions to derive weights for each observation. The decision is then a locally weighted average over the historical observations. We use the kernel approach in our analyses because it can be used for both SEO and JEO, a comparison that has not been reported before, and to contrast the results we get with our new, tree-based approach. In contrast to Ban and Rudin [2018], we focus on the difference between SEO and JEO and carve out the key performance drivers.

Oroojlooyjadid et al. [2016] combine deep-learning (a form of artificial neural networks) with a newsvendor-style loss function. They apply their new approach to a newsvendor problem with multiple items and compare their performance to several standard approaches. They show that their method works well in settings with sufficient training data and under unknown underlying demand distributions. However, they do not compare their JEO approach with an SEO version, where deep-learning would be used to predict demand. Therefore, how much of the cost improvement they achieve (compared to the benchmark approaches from the literature) is due to the integration of estimation and optimization and how much is due to the superior and more complex prediction method remain unclear.In addition, while deep learning algorithms are powerful and typically provide good results, they are black boxes in terms of interpretability and so are less adequate for use in an exploration of structural differences between SEO and JEO than are, for example, tree-based approaches.

Bertsimas and Kallus [2019] propose a tree-based approach that is a combination of SEO and JEO: Their model uses the standard mean-squared error loss function to determine the structure of the decision tree. In a second step, the authors determine the response for each leaf of the tree by solving a problem-specific instance of the optimization problem in (4.3), given the

sample of learning data that is sorted in each leaf. They extend this logic to random forests, which are an ensemble of decision trees that typically provides better results than single trees [Caruana et al., 2008].

While the tree-based approach in Bertsimas and Kallus [2019] is closest to our model in terms of the underlying machine learning method, the main drawback of SEO models, that is, the application of two independent optimization steps (the structural learning and then the actual cost "optimization"), is also present in their approach. In contrast to our approach, they do not consider the problem-specific costs of determining the structure of the decision tree. Only by integrating these costs can we obtain a truly JEO approach that is based on random forests. The next section provides a detailed description of our model for a newsvendor-style inventory problem.

4.3 Application to the newsvendor problem

Motivated by the problem in a real-world case at a restaurant chain, we consider a newsvendor setting to illustrate the structural performance differences between the SEO and JEO approaches[13]. In this case, the restaurant manager needs to determine the quantity q of a product to be prepared for the next day. Demand is not stationary but is driven by external effects, which we incorporate as k-dimensional feature vector $\mathbf{x}$. Unsold quantities must be disposed of at a cost of c_o per disposed unit, and the estimated cost of unmet demand is c_u per unit. As in (4.3), the goal is to minimize the total expected cost:

$$\min_{q(\mathbf{x})\in\mathcal{Q}} \mathbb{E}[C(q(\mathbf{x}), D)] \tag{4.4}$$

with the specific newsvendor cost function

$$C(q(\mathbf{x}), D) = c_u(D - q(\mathbf{x}))^+ + c_o(q(\mathbf{x}) - D)^+, \tag{4.5}$$

[13]The JEO concept can also be applied to other decision problems with more complex cost functions, such as in capacity management problems, as in Taigel et al. [2019].

where D is the random demand and $(.)^+$ is a function that returns 0 if its argument is negative, and else its argument.

To solve this optimization problem, we need to further specify the function $q(\mathbf{x})$. In the following, we present implementations with two underlying functions (i.e., machine learning techniques): the first is based on random forests and the second is based on kernel regression.

4.3.1 Implementation based on random forests

The random forests machine learning technique, first introduced by Breiman [2001], has been shown to have high prediction accuracy in various settings [Caruana and Niculescu-Mizil, 2006, Caruana et al., 2008]. For our analyses, tree-based approaches like random forests are particularly useful since we can use their final tree structures to measure heteroscedasticity, as described in subsection 4.4.3.

In general, a random forest consists of a number of trees T that partition the feature space into regions R that group instances whose features have similar values. The prediction of a new, unseen instance is obtained by grouping the instance into one of the regions based on the values of its features and assigning a demand estimate, such as their mean demand, based on the other instances in this region. The underlying rationale of this approach is that instances that are similar in known properties of the data (the features) can reasonably be assumed to be similar also in unknown properties (e.g., the realized demand). The regions are found by recursively applying axis-parallel splits on the training data set $\mathcal{T}$ to minimize a training loss function $L(\hat{f}(\mathbf{x}), D)$. Going forward, we call θ the parameter vector that determines how a tree is grown and $R(\mathbf{x}', \theta)$ the region of a single tree into which a new instance described by $\mathbf{x}'$ would be sorted. According to Athey et al. [2019], we can interpret such a region as a forest-based adaptive neighborhood of $\mathbf{x}'$ that is defined via the data-driven weights $w_i(\mathbf{x}')$ of each historical observation i.

The notion of providing a data-driven way to re-weight historical observations for predictions plays a key role when random forests are used in inventory decisions. In the following, we detail how the basic random forest mechanism

can be used via both the SEO approach and the JEO approach to derive such decisions. We note two differentiating properties of the two approaches: how regions are generated via the training algorithm and how the final decisions are derived given the specific neighborhoods.

SEO based on random forests As described in Section 4.2, the generic SEO approach separately estimates an expected demand level μ and accounts for the remaining uncertainty by calculating an additional safety stock, depending on the distribution of forecast errors. Following this methodology, the random forest algorithm is employed to predict the mean demand, conditional on the realization of the feature vector $\mathbf{x}'$. To receive the regions $R_{SEO}(\mathbf{x}', \theta)$ that are needed to predict the conditional mean, tree structures are learned by splitting the feature space to minimize the standard MSE loss function:

$$L(\hat{\mu}(\mathbf{x}), D) = L_{\text{MSE}}(\hat{\mu}(\mathbf{x}), D) = \frac{1}{n}\sum_{i=1}^{n}(d_i - \hat{\mu}(\mathbf{x}_i))^2. \tag{4.6}$$

Then, given regions $R_{SEO}(\mathbf{x}', \theta_t)$ from tree t into which a new instance $\mathbf{x}'$ is sorted, we can calculate weights $w_i(\mathbf{x}')$ for historical observations as:

$$w_i(\mathbf{x}') = \frac{1}{T}\sum_{t=1}^{T}\frac{\mathbb{1}_{(\mathbf{x}_i \in R_{SEO}(\mathbf{x}', \theta_t))}}{N(\mathbf{x}', \theta_t)}, \tag{4.7}$$

where $n(\mathbf{x}', \theta_t)$ defines the number of historical observations from the training set that fall into the same region as $\mathbf{x}'$. Given these weights, the prediction of the conditional mean is then a weighted sum over all observations d_i:

$$\hat{\mu}_{SEO}(\mathbf{x}') = \sum_{i=1}^{n} w_i(\mathbf{x}')d_i. \tag{4.8}$$

In the subsequent optimization step we find an additional safety stock that covers the decision-maker against forecasting errors by trading off the expected overage and underage costs. This problem corresponds to the solution of the simple data-driven newsvendor problem without features [Levi et al., 2015]. To solve this problem, we require an empirical distribution of the out-of-

sample prediction errors. Hence, after training the random forest on a subset of the training data, we evaluate the predictions on the remaining set that was not used for training. Then the out-of-sample prediction errors ϵ_i are calculated and the final inventory decision from SEO-RF is determined as:

$$\hat{q}_{SEO-RF}(\mathbf{x}') = \sum_{i=1}^{n} w_i(\mathbf{x}')d_i + \inf\{\varepsilon : \hat{F}_n(\varepsilon) \geq \frac{c_u}{c_u + c_o}\}, \tag{4.9}$$

where $c_u/(c_u + c_o)$ corresponds to the service level (SL) that determines the optimal fraction of demand shortages, and $\hat{F}_n^{-1}(\varepsilon)$ denotes the inverse of the empirical cumulative distribution of forecast errors. It can be shown that, if F is continuous, the second part of the sum becomes $\hat{\varepsilon}_n = \varepsilon_{\lceil n \cdot SL \rceil}$, the $\lceil n \cdot SL \rceil$th largest forecast error [Ban and Rudin, 2018].

JEO based on random forests The JEO method based on random forests (JEO-RF) has two major differences from the SEO random forest (SEO-RF) approach: First, the cost structure of overage versus underage quantities is already considered within the loss function of the training algorithm, generating tree structures that already reflect the second-stage optimization problem from the SEO approach. Second, given such tree structures, a different method of considering the neighboring observations is used to derive the final inventory decisions. Consider the following asymmetric loss function:

$$L(q(\mathbf{x}), D) = C(q(\mathbf{x}), D) = \sum_{i=1}^{N} c_o(q(\mathbf{x}) - d)^+ + c_u(d - q(\mathbf{x}))^+. \tag{4.10}$$

Here, excess quantity (i.e., if $(q(\mathbf{x}) - d) > 0$) is considered with c_o in the loss function, whereas missing quantities ($(q(\mathbf{x}) - d) < 0$) are weighted with c_u.

Having learned cost-aware tree structures, we apply the random forest kernel method developed in Scornet [2016] to define weight functions for the training instances as:

$$w_i(\mathbf{x}') = \sum_{t=1}^{T} \frac{\mathbb{1}_{(\mathbf{x}_i \in R_{JEO}(\mathbf{x}', \theta_t))}}{\sum_{t=1}^{T} N(\mathbf{x}', \theta_t)}. \tag{4.11}$$

According to Scornet, using this approach avoids rough estimates in regions

of the feature space where data is sparse. Similar to the SEO approach based on random forests, we can use these weights to define data-driven neighborhoods for a new instance $\mathbf{x}'$. Now, applying the framework of Bertsimas and Kallus [2019] and inserting our loss function (4.10), we can generate the final inventory decisions with JEO-RF by solving:

$$\hat{q}_{JEO-RF}(\mathbf{x}') = \arg\min_{q(\cdot)\in\mathcal{Q}} \sum_{i=1}^{N} C(q(\mathbf{x}'), d_i) = \inf\{d : \sum_{i=1}^{N} w_i(\mathbf{x}')\mathbb{1}_{(d_i \leq d)} \geq \frac{c_u}{c_u + c_o}\}. \tag{4.12}$$

The last equality follows from the fact that the resulting problem corresponds to a quantile regression problem [cf. Meinshausen, 2006].

4.3.2 Implementation based on kernel optimization

To validate the results we obtain with our new random forest-based approach, we also implement and evaluate the SEO and JEO concepts based on a kernel optimization (KO) method. The JEO-KO approach, introduced by [Ban and Rudin, 2018], provides the best results in a comparative study that uses a real-world data set.

The basic idea of kernel regression goes back to Nadaraya [1964] and Watson [1964]), who propose to estimate a dependent variable like demand using a locally weighted average of historic demands, where the weights are subject to how close the values of the historic observation's features are to those of the instance in question.

SEO with kernel regression For SEO-KO, the kernel-based SEO approach, we follow the SEO concept as described in Section 4.2 and use kernel regression to estimate a function $\hat{f}_{SEO-KO}$ that predicts demand D given a feature vector $\mathbf{x}'$. This function is referred to as the Nadaraya-Watson estimator and is given by:

$$\hat{\mu}_{SEO-KO}(\mathbf{x}') = \frac{\sum_{i=1}^{N} K_w(\mathbf{x}' - \mathbf{x}_i) d_i}{\sum_{i=1}^{N} K_w(\mathbf{x}' - \mathbf{x}_i)}, \tag{4.13}$$

where $K_w(\mathbf{u})$ is a kernel function with bandwidth w. Like Ban and Rudin [2018], we use the Gaussian kernel function:

$$K(\mathbf{u}) = \frac{1}{\sqrt{2}} exp^{-||u||_2^2/2}, \tag{4.14}$$

with $K_w(\mathbf{u}) = K(\mathbf{u}/w)/w$.

With the function $\hat{\mu}_{SEO-KO}$, we evaluate the predictions on the training data and obtain out-of-sample prediction errors $\varepsilon_i,\ i = 1, ..., N$. Similar to SEO-RF, we determine the final inventory decision as:

$$\hat{q}_{SEO-KO}(\mathbf{x}') = \hat{\mu}_{SEO-KO}(\mathbf{x}') + \inf\{\varepsilon : \hat{F}_n(\varepsilon) \geq \frac{c_u}{c_u + c_o}\}, \tag{4.15}$$

where $c_u/(c_u + c_o)$ corresponds to the service level (SL) that determines the optimal fraction of demand shortages based on overage and underage costs, and $\hat{F}_n^{-1}(\varepsilon)$ denotes the inverse of the empirical cumulative distribution of forecast errors.

JEO with kernel optimization The main difference between the kernel-based JEO approach (JEO-KO) and the SEO-KO approach is that, as introduced by Ban and Rudin [2018], the JEO-KO uses the Nadaraya-Watson estimator (as in (4.13)) to estimate the newsvendor cost instead of demand. The JEO-KO approach is then given by:

$$\min_{q \geq 0} \frac{\sum_{i=1}^N K_w(\mathbf{x}' - \mathbf{x}_i) C(q, d_i)}{\sum_{i=1}^N K_w(\mathbf{x}' - \mathbf{x}_i)}. \tag{4.16}$$

According to Ban and Rudin [2018], (4.16) is a one-dimensional piecewise linear optimization problem, and the solution is given by:

$$q_{JEO-KO}(\mathbf{x}') = \inf\{q : \frac{\sum_{i=1}^N \kappa_i \mathbb{I}(d_i \leq q)}{\sum_{i=1}^N \kappa_i} \geq \frac{c_u}{c_u + c_o}\}, \tag{4.17}$$

where $\kappa_i = K_w(\mathbf{x}' - \mathbf{x}_i)$. Therefore, $q_{JEO-KO}(\mathbf{x}')$ is the smallest value for which the inequality in (4.17) is just satisfied.

4.4 Comparison of SEO and JEO

In this section, we analyze the drivers of differences in the SEO and JEO approaches' performance. In the first subsection we compare SEO and JEO when the relationship between features and demand (SEO) and that between features and decision (JEO) are modeled as linear functions. In this linear setting we can show analytically that SEO leads to suboptimal decisions if the remaining forecast uncertainty follows a non-random pattern. In line with the econometrics literature, we refer to such feature-dependent uncertainty as heteroscedasticity [e.g., Asteriou and Hall, 2011].

Our findings from the analytical examination with linear models culminate in our hypothesis that heteroscedasticity is also the main driver of performance differences in the more complex JEO and SEO approaches. Since tree-based and kernel-based models do not allow for analytical treatments similar to those that linear models do, our following analyses are based on two studies: A simulation experiment in which we evaluate the impact of various specifications of the data structures on the models' performance while controlling for exogenous, confounding effects, and a test of our findings on a real-world data set, where we apply the two approaches to an inventory-planning problem from a restaurant chain.

4.4.1 Analytical examination

A common assumption in regression settings–that is when we want to model a relationship between a dependent variable and a set of independent variables– is the homoscedasticity of the error term. This assumption means that we can describe the variation of the dependent variable as the sum of a term explained by the model, $\mu(\mathbf{x})$, and a stochastic error component with constant variance across all instances. However, this homoscedasticity assumption often fails to hold in practice. Breiman and Friedman [1985] describe the problem of predicting the ozone levels for the subsequent day and show that these levels can be forecasted much more accurately on some days than on others. The same holds for demand predictions where, for example, the demand for a

restaurant on a typical weekday may vary significantly less than it does on a weekend. If $\sigma_\varepsilon(\mathbf{x})$ is not constant, the error term is heteroscedastic.

In this subsection, we compare the impact of heteroscedasticity on the cost performance of SEO and JEO when the relationship between features and demand (SEO) and that between features and decision (JEO) are modeled as linear functions. The linear SEO approach consists of a least squares estimate of the conditional mean function $\hat{\mu}(\mathbf{x})$ and a sample quantile of all residuals $\hat{q}_\varepsilon(SL) = \inf\{\varepsilon : \hat{F}_n(\varepsilon) \geq SL\}$ to account for the asymmetric cost structure. The decision is hence given by:

$$\hat{q}_{SEO-Lin}(\mathbf{x}) = \mathbf{x}\hat{\beta}_{LSE} + \hat{q}_\varepsilon(SL), \tag{4.18}$$

where $\hat{\beta}_{LSE} = (\xi'\xi)^{-1}\xi'\mathbf{d}$ is the parameter vector that is derived from the least squares regression with design matrix ξ containing all k-dimensional feature vectors and the according demand observations $\mathbf{d}$.

The linear JEO approach, as proposed by Beutel and Minner [2012] and Ban and Rudin [2018], is given by the conditional quantile:

$$\hat{q}_{JEO-Lin}(\mathbf{x}) = \mathbf{x}\hat{\beta}_{SL}, \tag{4.19}$$

where $\hat{\beta}_{SL} = arg\min_{\beta\in\mathbb{R}^k} \sum_{i=1}^{n}(C(\mathbf{x}_i\beta, d_i))$, with $C(q,d) = c_u(d-q)^+ + c_o(q-d)^+$ as the newsvendor cost function.

For a simple linear demand model with independent and identically distributed (iid) errors which do not depend on $\mathbf{x}$, Koenker [2005] points out that the quantile function as in Equation (4.19) is – similar to the linear SEO approach in (4.18) – just a vertical displacement by the sample quantile of the error distribution $\hat{q}_\varepsilon(SL)$. Hence, for an homoscedastic linear setting, both approaches lead to similar results.

However, if there is any form of feature-dependent uncertainty, the assumption of (iid) errors which is crucial for the linear SEO approach does not hold. We will analyze the impact of heteroscedasticity on both approaches in

simple univariate linear location-scale model:

$$D|(X = x) = \beta x + (\gamma x)u \tag{4.20}$$

with $u \sim F_u$ independent of the realizations x of the random feature X, with an (unknown) symmetrical densitiy function $f_u(.)$ with mean zero, $\gamma > 0$ a scale parameter for heteroscedasticity.

In this setting, the optimal newsvendor decision are given by [Koenker, 2005]:

$$q^*(x) = x(\beta + \gamma F_u^{-1}(SL)) \tag{4.21}$$

Proposition 4.1. *For a linear location scale model with heteroscedasticity as in (4.20), the following holds:*

$$\mathbb{E}_{X\times D}\left[C(q_{JEO-Lin}(x), D)\right] \leq \mathbb{E}_{X\times D}\left[C(q_{SEO-Lin}(x), D)\right] \tag{4.22}$$

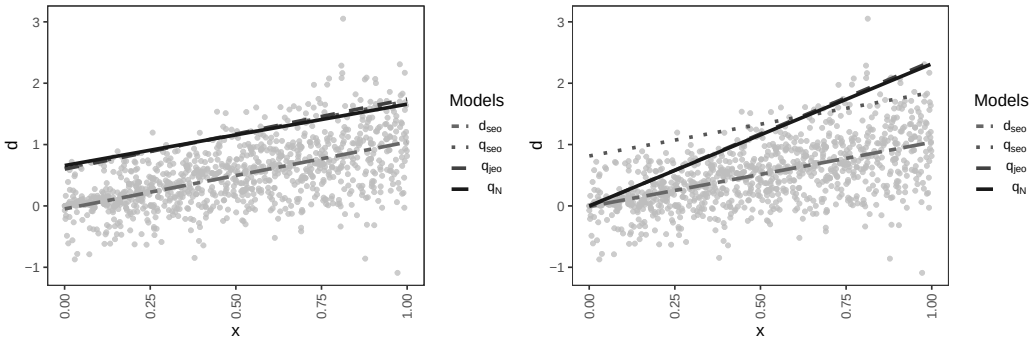
for $\gamma > 0$.

The proof of both propositions in this chapter can be found in the appendix.

Figure 4.1 illustrates this example for $X \sim unif(0, 1)$ by showing that, for the homoscedastic setting, both the SEO approach and the JEO approach perform well near the optimal decision quantile. However, for the heteroscedastic case, only JEO captures the structure of the noise appropriately by adjusting the slope of the regression line, while SEO results in inefficiently high or low ordering decisions since there is only a parallel shift of the regression line.

Figure 4.1 illustrates this example by showing that, for the homoscedastic setting, both the SEO approach and the JEO approach perform well near the optimal decision quantile. However, for the heteroscedastic case, only JEO captures the structure of the noise appropriately by adjusting the slope of the regression line, while SEO results in inefficiently high or low ordering decisions since there is only a parallel shift of the regression line.

Furthermore, the scale of the effect of heteroscedasticity depends on the service level, that is, the asymmetry of the cost structure:



(a) Homoscedasticity (b) Heteroscedasticity

Figure 4.1: Comparison of the linear SEO and JEO approaches under homoscedastic versus heteroscedastic settings

Proposition 4.2. *With $C(.)$ the newsvendor cost function from Equation (4.5), $0 \leq \gamma \leq 1$ and $X \sim unif(0,1)$ the following holds. For symmetric costs (i.e., $SL = 0.5$),*

$$\mathbb{E}_{X\times D}\left[C(q_{JEO-Lin}(x), D)\right] = \mathbb{E}_{X\times D}\left[C(q_{SEO-Lin}(x), D)\right].$$

For $SL > 0.5$, $\mathbb{E}_{X\times D}\left[C(q_{SEO-Lin}(x), D)\right] - \mathbb{E}_{X\times D}\left[C(q_{JEO-Lin}(x), D)\right]$ increases in SL.

From these findings for linear models, we derive two main conjectures, which we analyze with more complex underlying machine learning models in the following study:

Conjecture 4 (Homoscedasticity vs. heteroscedasticity)**.** *In a homoscedastic setting, JEO's performance is not better than that of SEO. JEO's performance will improve relative to SEO with increasing levels of heteroscedasticity – that is, the more $\sigma(\mathbf{x})$ changes subject to $\mathbf{x}$.*

Conjecture 5 (Effect of service level)**.** *For symmetric costs (i.e., a service level of 0.5) heteroscedasticity has no significant effect on the relative performance differences between SEO and JEO. The effect of heteroscedasticity increases with increasing asymmetry.*

In the following, we examine these structural differences between SEO and JEO for the more complex underlying machine learning models of random forests and kernel optimization. For this examination, we compare the models in a controlled simulation experiment and using a real-world dataset from a restaurant chain, since with these models, we cannot provide proofs of propositions as we did for the linear model.

4.4.2 Study 1: Simulation analysis

Our first numerical study is a controlled simulation experiment that allows us to quantify the effect of feature-dependent demand uncertainty when we have a homoscedastic or heteroscedastic uncertainty structure. In this controlled setting, we can isolate and examine single cause-effect relationships. We complement our simulation study with an analysis using a real-world data set, which does not allow similar insights, as many effects, such as nonlinearity, heteroscedasticity, and spurious correlations between predictors and prescriptions, overlay it. We posit that our simulation approach allows for the extraction of meaningful insights regarding the factors that drive performance differences and provides us with the possibility to underpin our findings statistically.

In this section we first describe our experimental setup. We explain how we control the feature-related uncertainty through our choice of a demand model and its parameterization and present the results first for the random forests approach and then for the kernel-based approach.

Experimental setup

We use an additive demand model that can control the feature-demand relationship and the feature-dependent uncertainty separately. More formally, we determine demand D as:

$$
\begin{aligned}
D &= \mu(\mathbf{x}) + \varepsilon_\gamma(\mathbf{x}) \\
\text{with } \mu(\mathbf{x}) &= x_1 + ... + x_k \\
\text{and } \varepsilon_\gamma(\mathbf{x}) &= \varepsilon_\gamma^0(1 - x_0) + \varepsilon_\gamma^1 x_0, \\
\text{where } \varepsilon_\gamma^0 &\sim \mathcal{N}\left(0, (1-\gamma)\sigma_{base}\right) \\
\text{and } \varepsilon_\gamma^1 &\sim \mathcal{N}\left(0, \sqrt{2 - (1-\gamma)^2}\sigma_{base}\right) \\
\text{with } x_0 &\in \{0, 1\}, \\
\sigma_{base} &= \mathbb{E}\left[\mu(\mathbf{x})\right] cv_{noise},
\end{aligned}
\tag{4.23}
$$

where γ is the simulation parameter that determines whether we obtain homoscedastic demand (for $\gamma = 0$) or discrete heteroscedastic demand with increasing levels of heteroscedasticity (for $\gamma = 0.1, 0.2, ..., 1$). The coefficient of variation cv_{noise} is the parameter that controls the level of noise. In our simulation, we control cv_{noise} since it is independent of the mean. We consider heteroscedasticity with a two-population model for the uncertainty component ϵ_γ and a feature x_0 that influences only the structure of the uncertainty and has no effect on the demand level. In reality, x_0 could represent, for example, whether we consider a typical weekday or a weekend day, assuming that the mean is similar but the uncertainty around our predictions is higher on weekends. Via this modeling approach, γ controls the level of heteroscedasticity by affecting the difference of the standard deviations of ϵ^0 and ϵ^1. As an example, $\gamma = 0.3$ results in an uncertainty model where the standard deviation of $\epsilon_\gamma^0 \sim \mathcal{N}\left(0, 0.7 * \sigma_{base}\right)$ is about 1.76 times higher than the standard deviation of $\epsilon_\gamma^1 \sim \mathcal{N}\left(0, 1.23 * \sigma_{base}\right)$.

In more detail, for each configuration of parameters γ, cv_{noise}, and σ_{base}, we draw N_{Sim} realizations from a uniform distribution with range $[0; 1]$ for each of the k demand features. The demand level is then given by the sum $x_1 + ... + x_k$. We also draw N_{Sim} realizations for $x_0 \sim Bernoulli(0.5)$, the feature that determines whether the uncertainty component for a particular observation should be drawn from ϵ_γ^0 or ϵ_γ^1.

Then, the final demand observation D is composed of the sum of demand

Experiment	**Simulation**	**Real-world application**
	Section 4.4.2	Section 4.4.3
Parameters		
γ	$\{0, 0.25, \ldots, 1\}$	–
SL	$\{0.5, 0.8, 0.95, 0.99\}$	$\{0.5, 0.8, 0.95\}$
Controls		
cv_{noise}	$\{0.25, 0.5, 0.75, 1\}$	–
Model configs		
n_{trees}	$\{100, 500\}$	$\{100, 500\}$
min_{node}	$\{5, 15, 30\}$	$\{5, 15, 30\}$

Table 4.1: Parameter settings for our experiments

level $\mu(\mathbf{x}) = x_1 + ... + x_k$ and the error term $\epsilon_\gamma(\mathbf{x})$ as described in (4.23). Following this approach, we obtain a training dataset $\mathcal{T}_{N_{sim}} = \{(d_i, \mathbf{x}_i), i = 1, ..., N_{sim}\}$. To measure the performance of each model, we use the first $N_{sim} - 1$ instances to train the model and then evaluate them for period N_{sim}. This procedure is repeated S times to achieve stable results. Mismatch costs incurred by model $m \in \{\text{JEO-X, SEO-X}\}$ with X either RF or KO are calculated for each simulation run $s = 1, \ldots, S$ via the cost function:

$$C(\hat{q}_m(x_s), d_s) = c_u(d_s - \hat{q}_m(x_s))^+ + c_o(\hat{q}_m(x_s) - d_s)^+, \tag{4.24}$$

where $\hat{q}_m^s$ is the inventory decision in simulation run s prescribed by model m. The cost parameters c_u and c_o are assumed to be normalized ($c_u + c_o = 1$), so they can be derived from SL since $SL = c_u / (c_u + c_o)$. Subsequently, we calculate the mean cost performance

$$\bar{c}_m = 1/S \sum_{s=1}^{S} C(d^s, \hat{q}_m^s) \tag{4.25}$$

per model m and report the relative cost improvement δ_{JEO} of the JEO ap-

proach compared to the SEO approach as follows:

$$\delta_{JEO} = \frac{\bar{c}_{\text{JEO-X}} - \bar{c}_{\text{SEO-X}}}{\bar{c}_{\text{SEO-X}}}; \tag{4.26}$$

To evaluate our conjectures, we run a series of simulation experiments under a wide range of parameter combinations, as shown in Table 4.1. We test for the influence of feature-dependent uncertainty on the relative performance of the JEO and SEO approaches while controlling for the overall uncertainty level and the asymmetries between overage costs and underage costs. More specifically, we vary the γ parameter for various combinations of service level SL and cv_{noise}. For all of our experiments, we choose $N_{sim} = 501$ observations and $k = 3$ as the number of features that determine the demand levels. Although the number of considered features in practical scenarios is usually much higher (e.g., for our yaz case study, we have $k = 168$), other studies [e.g., Bertsimas and Kallus, 2019] show that tree-based approaches like random forests are especially likely to perform robustly even with noisy features, (i.e., features without predictive power or with only minor predictive power). We fix the number of simulation runs to $S = 100$ for each parameter configuration and model.

We implemented the models we describe in Section 6.3 in the statistical programming language R. For the random forest models we extended the `ranger package` [Wright and Ziegler, 2017].

Results for random forest-based approaches

Figure 4.2 shows the relative performance improvement δ_{JEO} of JEO-RF over the SEO-RF approach for increasing levels of heteroscedasticity for various parameterizations of noise parameters and the service level parameters.

In settings with a low level of uncertainty ($cv_{noise} = 0.25$) there is no effect of increasing heteroscedasticity, and both approaches do equally well in recovering the underlying linear relationships. If the uncertainty is low, whether there is any structure in the remaining uncertainty that could be beneficial for JEO-RF seems to make no difference.

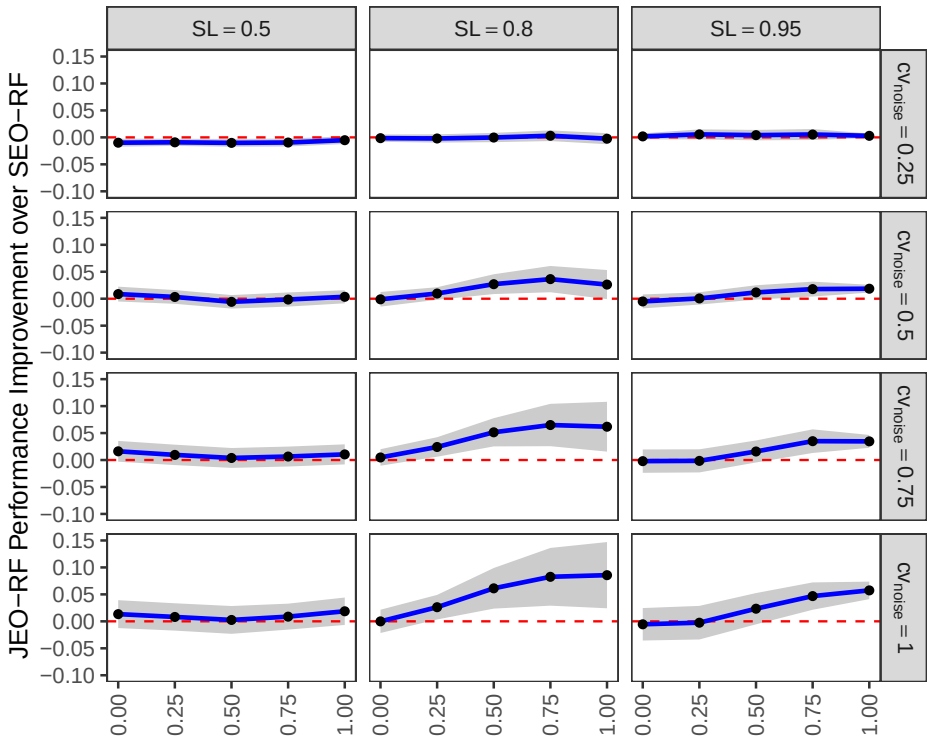


Figure 4.2: JEO-RF cost improvement over SEO-RF depending on γ (level of heteroscedasticity) in a linear demand setting for various service levels ($SL = 0.5,\ 0.8$ and 0.95 with different levels of base noise (cv_{noise}). The shaded area represents a 95% confidence interval around the mean improvement.

For higher levels of uncertainty, heteroscedasticity has a positive effect on the performance of JEO-RF compared to SEO-RF. In some settings (e.g., $cv_{noise} = 0.25$ and $SL = 0.95$), JEO-RF significantly outperforms SEO-RF, so Conjecture 4 holds if the uncertainty is high enough. However, for homoscedastic settings, JEO-RF can be inferior, especially in settings with high service levels. Given homoscedasticity, that SEO-RF uses all residuals in the optimizations step becomes an advantage since then the decision of SEO-RF is based on a larger sample compared to the JEO-RF. This larger sample for SEO-RF is especially important for high service levels since then the empirical quantiles come from the edges of the available samples which are even more sparse.

In line with Conjecture 5, we find that for symmetric costs (i.e., $SL = 0.5$), heteroscedasticity has no significant effect on the approaches' performance primarily because for symmetric costs, JEO-RF does not use the feature that drives the noise. Splitting along this feature would not make a difference in terms of costs since the distributions of the errors are both symmetric around zero and differ only in terms of variance. The minor difference stems from the fact that JEO-RF estimates the sample median, while SEO-RF with the MSE-loss estimates the sample mean. (See Appendix C.3 for additional details.)

We also see that the effect that heteroscedasticity has on the approaches' relative performance is more pronounced for higher service levels, a result that is again in line with Conjecture 5.

Results for kernel-based approaches

Figure 4.3 displays the relative performance improvements δ_{JEO} of JEO-KO over the SEO-KO approach for increasing levels of heteroscedasticity for different parameterizations of the noise and the service-level parameters. We use the same simulation setup as we used for our random forest approach. We find that the results with kernel optimization are mostly in line with the results for random forests, but the effects are less pronounced.

As is the case for random forests, for settings with low uncertainty there is no effect of increasing heteroscedasticity. For higher uncertainty levels heteroscedasticity has a positive effect on the performance of JEO-KO compared to that of SEO-KO, although the effect is somewhat less pronounced than it is for random forests. Still, in some settings (e.g., $cv_{noise} = 0.25$ and $SL = 0.95$), JEO-KO significantly outperforms SEO-KO. Hence, we state that Conjecture 4 holds if the uncertainty is high enough. However, for perfectly homoscedastic settings, JEO-KO can be inferior ($cv_{noise} = 0.75$ and $SL = 0.8$).

Also with regard to Conjecture 5, the results for the KO approaches are similar to those for random forests. We find no significant differences for symmetric costs (i.e., $SL = 0.5$). For higher service levels, , heteroscedasticity has a significant effect on the performance of KO-JEO compared to KO-SEO.

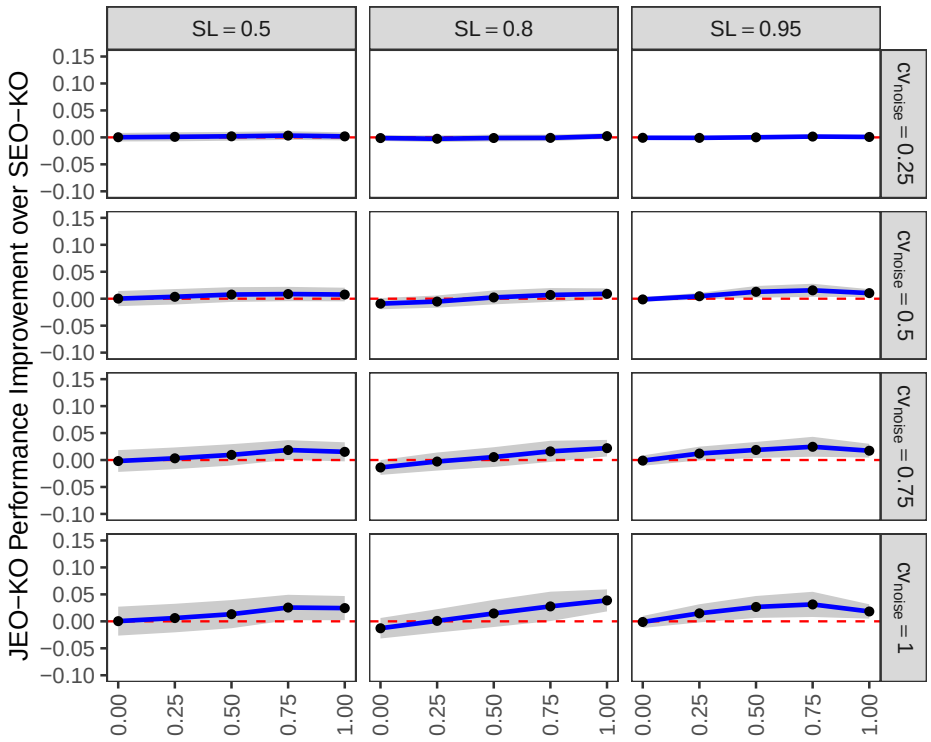


Figure 4.3: JEO-KO's cost improvement over SEO-KO depending on γ (level of heteroscedasticity) in a linear demand setting for various service levels ($SL = 0.5,\ 0.8$ and 0.95 with various levels of base noise (cv_{noise}). The shaded area represents a 95% confidence interval around the mean improvement.

The effect is more pronounced for higher noise levels.

We conclude that the key findings are related to fundamental differences between the JEO and SEO concepts and do not depend on the underlying ML technique.

4.4.3 Study 2: Prescriptive analytics at Yaz restaurant

In section 4.4.2, we examined the differences between the performance of SEO and that of JEO in a controlled experiment. While this approach allowed us to study the isolated effect of heteroscedasticity while controlling for the level of uncertainty and cost asymmetries, the overall setting was simpler than most real-world scenarios. In particular, our separating the feature-demand

relationship from the feature-uncertainty relationship is a strong assumption, as one would expect in scenarios where features drive the overall uncertainty of demand features also to influence the level of demand. Hence, the effect of heteroscedasticity cannot be traced as it can in a simulation experiment.

In this section, we compare the performance of JEO and SEO on a real-world inventory management problem that has many features with potentially complex nonlinear but unknown relationships to demand that are typically encountered in practical scenarios. We seek to confirm our simulation experiment's findings in terms of the relative performance between the two models. The data set stems from Yaz, a Germany-based fast-casual restaurant chain. Yaz offers meals with a limited range of main ingredients but with a broad variety of preparations. Because these main ingredients are perishable, Yaz has to decide how many of them to prepare each day. Hence, the problem structure (perishable items, per-unit overage, and underage costs) culminates in the well-known newsvendor problem described above.

The following sections first provide an overview of the data sources we used and the features we derived from the available data. Thereafter, we describe our evaluation setup – that is, the logic used to compare the two approaches. Finally, we present our results regarding the performance of both approaches in our real-world application.

Data

Yaz provided us with sales data from their flagship restaurant in Stuttgart, Germany, for the period from 2013/09/27 to 2015/11/09. The products' demand structure varies significantly in terms of the mean demand and the coefficient of variation. For this reason we report the model performance for three exemplary products (calamari, steak, lamb) whose demand structures differ. As illustrated in Figure 4.4, the smoothed demand is nonstationary over time, ruling out a basic newsvendor solution, which would require a stationary demand distribution to solve this inventory-management problem.

In the past, the restaurant manager wanted all products to be available at all times, so inventory levels were high and Yaz rarely faced stock-out

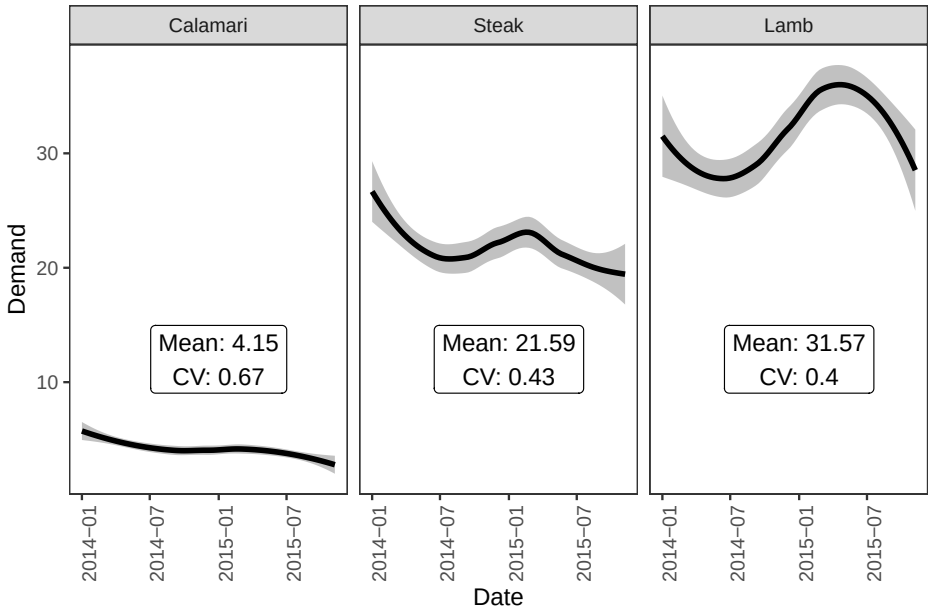


Figure 4.4: Evolution of the smoothed demand over time for different products.

situations. During the period under consideration, stock-out events occurred on only on 1.6% of the days, so all three ingredients were available on 98.4% of the time. Hence, we do not correct for censored demand data as Bertsimas and Kallus [2019]. We expect that this marginal rate of censored demand data will not have a significant effect on our comparison of JEO and SEO.

The restaurant manager's hypothesis was that the weather has a strong influence on demand, so we collected weather data from the databases of the German Meteorological Service and aggregated that data to a daily level to reflect the same level of granularity of a potential weather forecast for the next day. Since actual weather forecasts for the next day were not available, we used the actual weather information for the previous day as a proxy. Although this information would not be available at the time that a decision is made, the features we derived from this data are likely to be similar to a weather forecast for the next day.

Based on this raw data, we derived 168 features for each product by extracting structural information about the underlying time series (e.g., the rolling mean demand for the same weekday). Table 6.4 provides an overview

Source	Feature
Time Series	Average aggregate demand (for all products) on same weekday for the last two weeks
	Average aggregate demand (for individual products) on the same weekday over the last three weeks
	Aggregate demand (for all products) the day before
Calendar	Is December
	Is Saturday
	Is special day (Event, Holiday, etc.)
Weather	Air temperature two days ago
	Average Air temperature over last four days
	Average duration of sunshine over last five days

Table 4.2: Examples of relevant features for the product Steak

of the most important features.

Evaluation procedure

After cleaning and preprocessing the raw data, we obtain a data set $\mathcal{T}_{N_{Yaz}} = \{(d_i, \mathbf{x}_i), i = 1, ..., N_{Yaz}\}$ with $N_{Yaz} = 672$ demand observations. To evaluate our model performance on this data set, we use a five-fold cross-validation, splitting $\mathcal{T}_{N_{Yaz}}$ randomly into five roughly equal-sized subsets. Let

$$\phi : \{1, \ldots, N_{Yaz}\} \mapsto \{1, \ldots, 5\}$$

denote the indexing function that maps a particular observation to one of the five partitions. Then, $\hat{q}^{-\phi}(x)$) is the prescription function that is calibrated with the k-th part of the data removed. Thus, we calibrate our prescription model five times for different compositions of the training data set and evaluate it on the k-th part of the data. Subsequently, we compute the mismatch cost estimates as:

$$\bar{c}_m = \frac{1}{N_{Yaz}} \sum_{i=1}^{N_{Yaz}} C(d_i, \hat{q}_m^{-\phi(i)}(x_i))$$

Again, we report δ_m, the percentage cost improvement over the sample average approximation (SAA) benchmark per model m to improve our assessment of the models' performances.

In our simulation experiments described in section 4.4.2, we controlled for cost asymmetry in terms of the service level, uncertainty within the data, and heteroscedasticity, and now we quantify these drivers in our real-world experiment. For this, we calculate the out-of-sample mean squared error (*MSE*) of the predictions generated by the SEO approach as a measure of the remaining uncertainty (i.e., as a similar metric to the cv_{noise} parameter in our simulations):

$$\epsilon_{MSE} = \frac{1}{N_{Yaz}} \sum_{i=1}^{N_{Yaz}} (d_i - \hat{d}_{RF}(x_i))^2 \tag{4.27}$$

We also measure the heteroscedasticity in the residuals of the random forest predictions, so we calculate the state-dependent coefficient of variation over all historical observations $d_1, \ldots, d_{n_{lt}}$ sorted into a particular leaf l in a tree t of our SEO random forest:

$$cv_{lt} = \frac{\sqrt{(\sum_i (d_{ilt} - \frac{1}{n_{lt}} \sum_i d_{ilt})^2}}{\frac{1}{n_{lt}} \sum_l d_{ilt}} \tag{4.28}$$

Then we determine the standard deviation for each tree t separately:

$$sd_t = \sqrt{\sum_l \left(cv_{lt} - \frac{1}{L_t} \sum_l cv_{lt} \right)^2} \tag{4.29}$$

This standard deviation measures the heteroscedasticity in the residuals since it detects how much the coefficient of variation deviates depending on the actual state (i.e., the leaf into which an observation is sorted). Then we aggregate the sd_t to receive an indicator for heteroscedasticity γ_{RF}:

$$\gamma_{RF} = \frac{1}{T} \sum_t sd_t \tag{4.30}$$

Using this approach allows us to measure the state-dependent uncertainty for the SEO-RF method which serves as an approximation for the heteroscedas-

ticity in the residuals.

Results for random forest-based approaches

This section presents the main results for the application of JEO-RF and SEO-RF to Yaz's inventory management problem. Figure 4.5 shows the percentage cost improvements,

$$\delta_{m,SAA} = \frac{\bar{c}_m - \bar{c}_{SAA}}{\bar{c}_{SAA}} = \frac{\Delta_{m,SAA}}{\bar{c}_{SAA}},$$

of JEO-RF and SEO-RF relative to SAA for various service levels. As Figure 4.5 shows, both approaches considerably improve the mismatch costs compared to the SAA benchmark. We also find that the two methods perform similarly for the 0.5 service level, with slightly lower costs for SEO-RF. These results are in line with the outcome of our simulation, where neither approach outperformed the other for the 0.5 service level, as the resulting symmetric mismatch cost structure results in similar prescriptions.

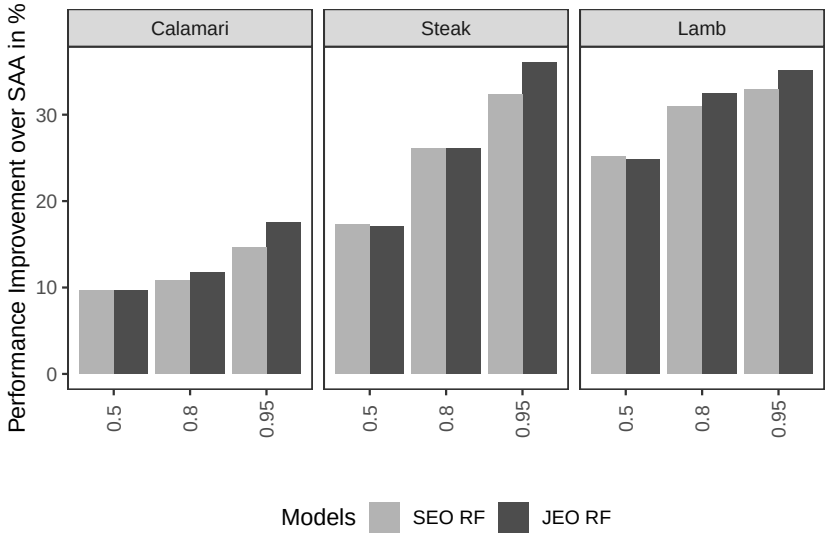


Figure 4.5: Percentage cost improvement $\delta_{m,SAA}$ over SAA for the SEO-RF and the JEO-RF models

For other service levels we find similar effects to those of our simulation ex-

periments: With increasing asymmetry of overage and underage costs, JEO's cost improvement over SEO increases. For example, for the 0.95 service level, we find considerable differences between JEO-RF and SEO-RF (e.g., for steak a cost improvement over SAA of 36% for JEO-RF and 32% for SEO-RF, and for calamari a cost improvement over SAA of 17% versus 15% for SEO-RF).

	Calamari	**Steak**	**Lamb**
MAE	2.00	5.53	7.17
MSE	6.72	54.35	89.00
γ_{RF}	0.35	0.18	0.16
Δ_{JEO}	0.02	0.07	0.02
$\delta_{JEO}(\%)$	6.31	7.32	1.58
p-value	0.008	0.086	0.599

Table 4.3: Measures for the forecast accuracy and heteroscedasticity of residuals of the SEO approach (upper part) and cost improvements of JEO over SEO for a 0.95 service level, and with the p-value results of a t-test (lower part).

In line with Conjecture 4, our simulation results in section 4.4.2 identified the level of heteroscedasticity in the residuals as a major driver of performance differences between the SEO-RF and JEO-RF approaches. To confirm these findings on the Yaz data set, we determined the structure of the remaining uncertainty by applying descriptive statistics to the residuals of the SEO, which are represented in Table 4.3. We find that calamari has the higher heteroscedasticity (measured by γ_{RF}) in the leaf nodes, followed by steak and lamb. Ceteris paribus, we expected JEO to have the highest cost improvement for calamari products, but as Table 4.3 shows, we achieve the highest relative improvement for steak (7.32%), followed by calamari (6.31%) and lamb (1.58%). We explain this outcome with an overlay of two opposite effects: Whereas we find heteroscedasticity is highest for calamari, we also see that the overall forecast accuracy for calamari is highest, i.e., remaining uncertainty for this product, which also affects the relative cost advantage of JEO over SEO, is lowest: The mean absolute error (MAE) of calamari is more

than 3.5 times lower than the forecasting error of lamb, considerably limiting the performance differences between the two approaches. In this case, the MAE (instead of a relative error) is the adequate measure since costs are related to the absolute deviations. For calamari, we find that the performance advantage of JEO over SEO is statistically highly significant, with a p-value of 0.008. Hence, we conclude that our results from the real-world case study are in line with the findings from the simulation study, providing additional support to our hypotheses that heteroscedasticity is an important driver of JEO's cost advantage over SEO.

Finally, we examined the stability of our results over a range of model parameter combinations. Table 4.4 presents the mean absolute cost improvements (Δ_{JEO}) and the scaled absolute cost improvements ($\frac{\Delta_{JEO}}{\bar{c}_{SEO}}$) for steak for various combinations of service levels and the model-specific tuning parameters n_{trees}, representing the number of trees, and min_{node}, the minimum number of observations in a node as an additional split. We find that, except for the 0.5 service level, all parameter configurations lead to lower mean costs for the JEO approach compared to SEO. However, in only four configurations do we find our cost improvement to be highly statistically significant.

Results for kernel-based approaches

In the following, we present the main results for the application of JEO-KO and SEO-KO to Yaz's inventory management problem. We use the same evaluation logic that we did in our random forest approach. Figure 4.6 displays the percentage cost improvements $\delta_{m,SAA} = \frac{\bar{c}_m - \bar{c}_{SAA}}{\bar{c}_{SAA}} = \frac{\Delta_{m,SAA}}{\bar{c}_{SAA}}$ $m \in \{\text{JEO-KO}, \text{SEO-KO}\}$ of JEO-KO and SEO-KO relative to SAA for different service levels.

We find that the results for KO are mostly in line with what we found for random forests, as SEO-KO and JEO-KO both improve the mismatch costs compared to the SAA benchmark. Other than for random forest, we find that for $SL = 0.5$, SEO-KO achieves a slightly higher cost improvement.

For high service levels, JEO-KO yields better results than its SEO counterpart. However, the kernel approach cannot measure the heteroscedasticity

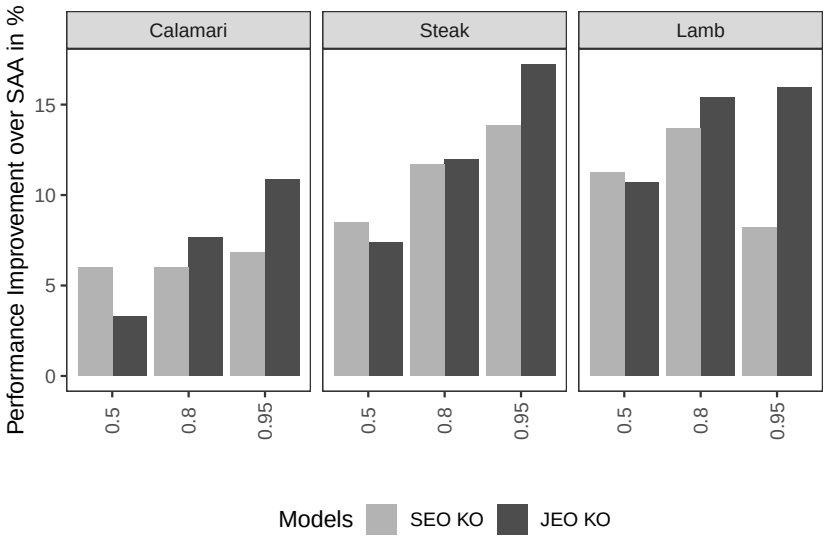


Figure 4.6: Percentage cost improvement $\delta_{m,SAA}$ over SAA for the SEO and the JEO kernel optimization models

as the random forest approach can. For this reason, we cannot draw conclusions concerning whether any differences in the performance of JEO-KO and SEO-KO using the kernel-based approach might be driven by heteroscedasticity in this practical setting.

While the main contribution of our paper is the comparison of JEO and SEO, we can also compare the results between JEO-RF and JEO-KO since we use the same benchmark and evaluation procedures. We see that JEO-RF's performance on this data set is considerably better than that of JEO-KO. For example, for steak with $SL = 0.95$, the mean performance improvement of JEO-RF is around 35 percent, while it is only around 17 percent for JEO-KO. However, these findings, reached using a specific data set, cannot be generalized.

4.5 Conclusion

We analyzed the performance of two fundamentally different concepts to consider data for a newsvendor-style inventory management problem, where vari-

ations in demand are driven by observable features. By comparing the respective implementations for SEO and JEO with two underlying machine learning algorithms, our in-depth analysis provides the first rigorous examination of performance differences between the two concepts. Moreover, the newly introduced JEO approach based on random forests is a novel method with which to determine optimal inventory quantities.

In a first analytical examination we showed that an SEO approach based on a linear regression model yields suboptimal results if heteroscedasticity in the residuals is present, whereas its JEO counterpart results in optimal inventory decisions. We saw a similar impact of heteroscedasticity for the two more complex nonlinear methods in a simulation study and in a study based on a real-world data set. The analysis of performance differences on our real-world data set suggests that both the random forest-based and the kernel optimization-based JEO approaches outperform their respective SEO counterparts in settings with high heteroscedasticity and high remaining uncertainty (i.e., low forecast accuracy), in combination with a highly asymmetric cost structure. Moreover, we find that our random forest-based JEO approach significantly outperforms the more established kernel-based JEO approach on our real-world data set. Furthermore, by exploiting its tree-based structure, we developed a measure to determine the amount of heteroscedasticity to derive further insights about the structure of the remaining uncertainty.

Hence, given settings with high service levels, low forecasting accuracy and presumed heteroscedasticity, using JEO models is appropriate because of their internal structure, which is geared to such settings. On the other hand, in situations in which forecast accuracy is high and mismatch costs are symmetric, SEO approaches perform well. In addition to the competitive performance of the established SEO approaches in such settings, they are flexible in terms of the underlying prediction model: While JEO approaches must be tailored to a specific setting, SEO approaches benefit directly from developments that lead to improved prediction models since they serve only as a building block while the subsequent optimization logic remains.

SL	n_{trees}	min_{node}	Δ_{JEO}	**LB 95% CI**	**UB 95% CI**	$\frac{\Delta_{JEO}}{\bar{c}_{SEO}}$
0.5	100	5	−0.01	−0.08	0.05	−0.00
0.5	100	15	0.00	−0.06	0.05	0.00
0.5	100	30	−0.03	−0.08	0.02	−0.01
0.5	500	5	−0.01	−0.06	0.05	−0.01
0.5	500	15	−0.02	−0.07	0.03	−0.02
0.5	500	30	−0.05	−0.09	0.00	−0.04
0.8	100	5	0.01	−0.05	0.07	0.00
0.8	100	15	0.03	−0.03	0.09	0.01
0.8	100	30	0.10	0.03	0.17	0.05
0.8	500	5	0.00	−0.05	0.05	0.00
0.8	500	15	0.02	−0.03	0.07	0.01
0.8	500	30	0.02	−0.03	0.07	0.01
0.95	100	5	0.03	−0.04	0.10	0.03
0.95	100	15	0.13	0.04	0.22	0.14
0.95	100	30	0.09	0.00	0.17	0.10
0.95	500	5	0.05	−0.02	0.12	0.06
0.95	500	15	0.07	−0.01	0.15	0.08
0.95	500	30	0.09	0.01	0.18	0.10

Table 4.4: Mean absolute performance differences between SEO and JEO for steak, depending on model configurations. The last column divides the absolute performance difference by the mean mismatch cost of the SEO model for the specific configuration to illustrate the magnitude of the improvements.

5 Data-driven capacity management with machine learning: A new approach and a case-study for a public service office

In this paper we consider the case of a public service office in Germany that provides services such as handling passports and ID card applications, notifications of change of addresses, etc. Their decision problem is to determine the staffing level for a specific staffing time-slot (e.g., next Monday, 8am to 12.30pm). Required capacity is driven by features such as the day of the week, whether the day is in school vacations, etc. We present an innovative data-driven approach to prescribe capacities that does not require any assumptions about the underlying arrival process. We show how to integrate specific service goals (e.g., "At most 20% of the customers should have to wait more than 20 minutes") into a machine learning (ML) algorithm to learn a functional relationship between features and prescribed capacity from historical data. We analyze the performance of our integrated approach on a real-world dataset and compare it to a sequential approach that first uses out-of-the-box ML to predict arrival rates and subsequently determines the according capacity using queuing models. We find that both data-driven approaches can significantly improve the performance compared to a naive benchmark and discuss benefits

and drawbacks of our approach. [14]

5.1 Introduction

In this paper we consider the problem of finding the right level of capacity for service operations. We address the case of a public service office in Germany that provides services such as the application andt issuance of passports or ID cards, notifications of change of addresses, etc. Their decision problem is to determine the staffing level for a specific time slot (e.g., next Monday, 8am to 12.30pm). Practitioners' intuition is that required capacity depends on the day of the week, whether this day falls on school vacations, etc. Our case is a typical example of over-the-counter service industries: multiple servers/stations that process customer orders in a first-come-first-served manner. Customer/order arrivals and the service time are uncertain and arrival rates are typically time-dependent. For-profit firms and governmental organizations face the same problem of determining the right capacity (i.e., number of servers) for different time intervals. Customers expect good service in terms of short waiting times and decision-makers want to avoid excessive costs for idle capacity.

Many well-established approaches in the literature determine capacity levels based on distributional assumptions for the inter-arrival and inter-departure times of the customers. Such an approach, however, ignores the uncertainty around an estimated distribution parameter and in many practical instances the approach lacks the suitability to be implemented.

We present a novel, data-driven approach to prescribe optimal capacity levels by directly modeling the functional relationship between capacity decision and features that potentially drive the required capacity. Our integrated approach does not require any assumptions about the underlying arrival process. Given a sufficiently large data set of historical observations of features and associated arrival processes, our approach derives a decision rule that

[14] This paper was published in the *Proceedings of the 2018 INFORMS International Conference on Service Science* Taigel et al. [2019]. Is is co-authored by Jan Meller and Alexander Rothkopf.

directly prescribes the minimal capacity to fulfill given service objectives. In this paper, we consider a single objective (e.g., at most 20% of the customer should have to wait more than 20 minutes), but we note that our approach can be extended to simultaneously incorporating additional service goals (e.g., at most x% abandonment rate or y minutes average waiting time).

5.2 Literature

Closely related to our approach is the work by Bassamboo and Zeevi [2009] who propose a data-driven approach to determine capacities in a call-center model with multiple customer classes and multiple server pools using historical call-arrival data. In their approach arrival rates of incoming calls are not assumed to be constant or known. Instead of making assumptions about the distribution of the arrivals, they use empirical estimates for the arrival rates which they derive from samples of historic call-arrival-epochs with similar characteristics. Based on these estimated distributions they can determine the expected penalty costs from abandonments with respect to a chosen capacity and hence minimize the sum of the expected penalty costs and the costs for capacity.

They can show that with an increasing amount of available data, their data-driven approach approximately achieves the same costs as one using a simulation-based approach with known arrival rates. However, their results also show that with a decreasing amount of observation, the average costs of their approach increase. We consider this as critical, since Bassamboo and Zeevi [2009] require samples of historic call-arrival-epochs with similar characteristics. Let for example a set of such similar epochs contain all Monday mornings without vacations, in the first week of a month, with no special weather event. This still rather broad specification limits the amount of similar observations to less than 10 given we have one year of data available. Hence, the choice of relevant characteristics and how we determine similar observations will influence the decision. In contrast to Bassamboo and Zeevi [2009], we integrate these considerations in our decision model. Our model

groups historical demand observations such that they allow for the best decision. Another methodical difference is that our approach does not require to estimate arrival rates, since we directly consider the capacity decision that would have been optimal given past arrivals.

Bertsimas and Doan [2010] also consider a call-center staffing problem and propose a data-driven approach that determines capacities for each unit period of the planning period (e.g., each hour of a day) by minimizing the mean cost over given historical arrival rates. They do not require explicit assumptions about the distribution of arrivals, however, they implicitly assume, that all historic observations from a specific time slot/unit period are similarly valuable for making the capacity decision for an upcoming period. Hence, they do not consider that external features could potentially explain parts of the variations in the historical data which is the main structural difference to the approach we present in this paper. Furthermore, their approach requires specific costs for waiting and abandonment which are not available in a setting like the public service office where specific service goals related to waiting time are more adequate.

5.3 Methodology

In this section we present a novel, data-driven approach to prescribe optimal capacity levels by directly modeling the functional relationship between capacity decision and features that potentially drive the required capacity. We first formulate the general model and show the flexibility of our approach. In the second subsection we describe an implementation based on the machine learning technique of decision tree learning.

5.3.1 Distribution-free approach for feature-based capacity decisions

In this section we introduce a novel approach to prescribe a capacity level $\mu(\mathbf{x})$ for a time-slot given a feature vector $\mathbf{x}$ that represents information characterizing this particular time-slot, e.g., day of the week, whether the time-slot

falls on a school holiday, etc. These prescribed capacities should fulfill certain service objectives G (e.g., ratio of staffing time-slots where at least 80% of the customers are served within a certain time). The actual capacity level $\mu(\mathbf{x})$ is then determined by minimizing the capacity level that is required to fulfill the service-level objectives $i = 1, ..., O$ for at least G_i^{target} of the observations:

$$\min_{\mu(\cdot)} \ \mu(\mathbf{x}) \tag{5.1}$$

$$s.t.\ G_i(\mu(x)) \geq G_i^{target} \quad \forall i \tag{5.2}$$

We can interpret Eq. (5.2) as second-level service goals that allow us to consider the trade-off between capacity and specific service-level objectives which are measured on a time-slot basis, e.g., the maximum waiting time or the average waiting time per customer. We note that Eq. (5.2) allows to control for multiple service goals independently which is a main difference compared to classical queuing approaches. Traditionally, decision makers have to focus on a single service goal. In the setting of our case study, the decision maker seeks to achieve that at most 20% of the customers within a certain time-slot should have to wait for more than 20 minutes. This is the only service goal, hence, $O = 1$. Such a constraint can be controlled and relaxed via Eq. (5.2). E.g., if $G_1^t arget = 0.95$, we allow the service goal to be missed in 5% of the cases. This makes the approach more robust against outliers.

Our data-driven approach learns the functional relationship $\hat{\mu}(x)$ from a set of historical data $\mathcal{T} = \{(\mu_n^{(*)}, \mathbf{x}_n)\}_{n=1,...,N}$ where each observation consists of an ex-post optimal decision $\mu_n^{(*)}$ and a feature vector $\mathbf{x}_n$ for each time-slot $n = 1, ..., N$.

In order to determine these ex-post optimal decisions $\mu_n^{(*)}$, we evaluate the historical arrival processes $\mathbf{y}_n$ representing the individual arrival times of each customer for each historical time-slot $n = 1, \ldots, N$. Hence, we solve the data-driven counterpart of Eqs. (5.1) - (5.2) for a given set of learning data $\mathcal{T}$:

$$\min_{\hat{\mu}(\cdot)} \sum_{n=1}^{N} \hat{\mu}(\mathbf{x}_n) \tag{5.3}$$

$$s.t.\ \hat{G}_i(\hat{\mu}(), \mathcal{T}) \geq G_i^{target} \tag{5.4}$$

Clearly, solving Eqs. (5.3)- (5.4) for a general function $\mu(\cdot)$ is infeasible due to too many degrees of freedom. For this reason, we need to specify a certain form of the functional relationship. For our approach we chose a tree-based model which we find highly suitable due to its high flexibility in modeling complex feature-demand relationships as well as its integrated feature selection mechanism. Besides these methodological properties, tree-based models have proven to perform well in various settings [see, e.g., Caruana and Niculescu-Mizil, 2006, Caruana et al., 2008].

5.3.2 Tree-based implementation

The general idea of tree-based machine learning algorithms is to partition the input feature space into disjunct "regions" by recursively finding the feature along with a split value that minimizes an objective function over a given set of historical "training data" $\mathcal{T}$. This procedure is recursively repeated until either an additional split would not lead to a substantial improvement or a minimum number of observations is reached. The interested reader is referred to the excellent presentation of tree-based models in Hastie et al. [2013] for further details.

The intuition behind this approach is that the decision we make for a specific time-slot is based on the decisions that would have been optimal in "similar" segments in the past. Our algorithm determines what is "similar" such that it allows for the best decisions (instead of mean predictions as with the standard tree-learning algorithm). Our solution encompasses the following four steps:

1. *Data pre-processing:* To make the algorithm computationally feasible, we build a NxM-dimensional look-up table W where N is the number

of available historical staffing segments and M is the maximum number of servers per time-slot that is available per time-slot. The entries in W are the ratios of waiting times violating the service target for the arrival process in a particular (historical) staffing time-slot given a specific capacity μ:

$$w_{n,\mu} = \frac{\sum_j \mathbb{1}(z_{nj}(\mu, \mathbf{y}_n) > t_{max})}{|\mathbf{y}_n|},\ n = 1, ..., N \text{ and } \mu = 1, ..., M$$

where $|\mathbf{y}_n|$ is the number of customers that arrived in time period n and $z_{nj}(\cdot)$ is the waiting time of arrival j in time period n and $\mathbb{1}(z > t_{max}) = 1$ if $z > t_{max}$ and 0 otherwise. The evaluation of the arrival process, i.e., computing $z_{nj}(\cdot)$ is the computationally expensive part. With the look-up table W, we have to do this only once for each capacity and historical time-slot. We can use W to obtain the ex-post optimal capacity decisions that we need as a training data set for our algorithm and to evaluate the resulting decisions. We note that for additional service goals we could compute additional look-up tables following the same logic.

2. *Ex-post optimization:* From W we can obtain the ex-post optimal capacity decision for each time slot $n = 1, ..., N$ by:

$$\mu_n^{(*)} = \inf_{\mu}\{w_{n,\mu} < (1 - \alpha)\} \tag{5.5}$$

where the service level α is the ratio of customers that are supposed to be served on time. We use these capacities in the learning data set $\mathcal{T} = \{(\mu_n^{(*)}, \mathbf{x}_n)\}_{n=1,...,N}$.

3. *Tree-learning:* We "learn" the structure of the tree by determining the partition of the parameter space that allows for the best capacity decisions. In detail, we recursively apply the following splitting step:

$$(x_p^*, s^*) = \underset{(x_p,s):p\in\{1,\dots,k\}\wedge s\in\mathcal{X}_p}{\operatorname{argmin}} \Big(\mathbb{L}\left\{\left(\mu^{(*)}, \mathbf{x}\right) \in \mathcal{S}_T | x_p \leq s\right\} + \mathbb{L}\left\{\left(\mu^{(*)}, \mathbf{x}\right) \in \mathcal{S}_T | x_p > s\right\}\Big) \tag{5.6}$$

where $\mathcal{X}_p$ is the set of all values of the p-th feature in the learning data and the loss function $\mathbb{L}(\mathcal{S}_T)$ for a set $\mathcal{S}_T \subset \mathcal{T}$ is the aggregated excessive capacity, defined as follows:

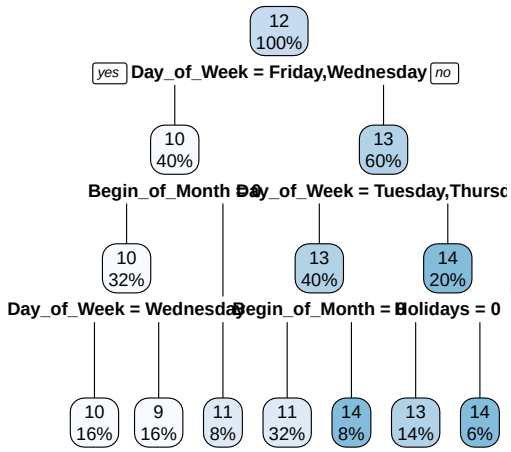
$$\mathbb{L}(\mathcal{S}_T) = \sum_{n:\left(\mu_n^{(*)}, \mathbf{x}_n\right)\in\mathcal{S}_T} (\mu_{\mathcal{S}_T} - \mu_n^{(*)})^+ \tag{5.7}$$

and

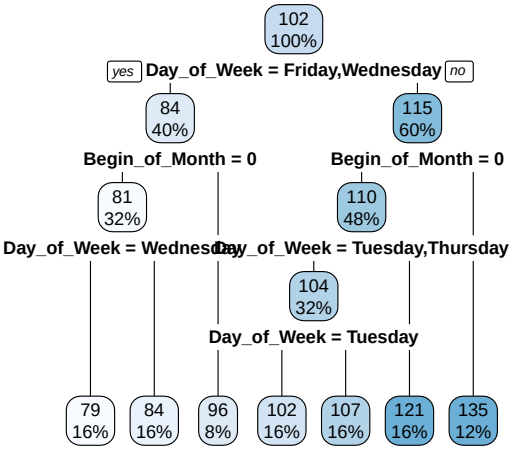
$$\mu_{\mathcal{S}_T} = \inf_{\mu}\{\mu \in 1, \dots, , M | \frac{1}{|\mathcal{T}|} \sum_{n:\left(\mu_n^{(*)}, \mathbf{x}_n\right)\in\mathcal{S}_T} \mathbb{1}(\mu_n^{(*)} \geq \mu) \geq G_{target}\} \tag{5.8}$$

where G_{target} is the ratio of time slots where the service level goal should be reached. Eq.(5.7) is the unutilized capacity if $\mu_{\mathcal{S}_T}$ is the capacity assigned to all historical observations in a set $\mathcal{S}_T$, which replaces the MSE as the basic loss function. Hence, Eq. (6.18) determines the split that allows for the best decision by grouping possibly similar decisions. Eq. (5.8) simply yields the sample quantile for the $\mu_n^{(*)}$ in a given subset, hence, if $G_{target} = 100\%$ then we get the maximum capacity in that subset. Essentially, the algorithm tries all possible splits (i.e., all combinations of x_p and s and finds the combination that minimizes the total loss resulting from the split.

4. *Apply staffing function:* Given the feature vector $\mathbf{x}'$ for a new, unseen, staffing time-slot, we now obtain the staffing decision by sorting $\mathbf{x}'$ into a region r by comparing the splits in the tree with the associated values



(a) Exemplaric tree from our approach; Numbers in the leaves are staffing decisions



(b) Prediction tree: Numbers in the leaves are predicted arrivals

Figure 5.1: Examples from case study: trees use similar variables but with a different structure.

of $\mathbf{x}'$. More formally,

$$\hat{\mu}(\mathbf{x}') = \sum_{r=1}^{R} \mu_r \mathbb{1}(\mathbf{x}' \in r) \tag{5.9}$$

where $r = 1, ..., R$ are the partitions of the feature space that were learned in the previous step. Figure 5.1a shows an example for a decision tree representation of the integrated learning approach. The obvious difference to a regression tree as depicted in Figure 5.1b are the leaf labels that are prescribed staffing levels of the integrated tree and predicted quantities for the classical regression tree.

Our main contribution is the integration of the specific optimization problem (minimizing capacity subject to certain service goals) into the estimation of a model that learns the functional relationship between features and output. We expect that this approach is especially useful if a) arrival rates are not stationary and if b) the non-stationarity is feature dependent. To clarify these two conjectures, we consider the following simple example. On average,

there are 100 customers per shift, but only 25 arrive in the first half of the shift whereas the second half of the shift sees on average 75 customers. Without any features, a separated approach bases the decision on the 100 estimated mean arrivals and typically misses the service goal due to the higher number of arrivals in the second half of the shift. Our integrated approach, would prescribe a capacity that would have achieved the service goal for past realizations of these arrival processes. Hence it would take the non-stationarity into account.

In order to clarify conjecture b), suppose we have a single binary feature, e.g., school holiday: yes/no that affects the arrival rates in the following way: During school holidays, arrival rates are constant throughout the shift with on average 100 arrivals. Without school holidays, we have non-stationarity as described above. In such a setting, a standard estimation model that aims at predicting the mean arrivals would not consider the school holidays feature, since it does not affect mean demand. Whereas our integrated approach would consider the feature if it improves the prescribed decisions, i.e., if it reduces the overall unutilized capacity, if different capacities are assigned to the subsets that are split by the school holiday feature. This is the main effect of the modified splitting function in Eq. (5) in step 3 of our procedure.

5.4 Case Study: Staffing Service Counters at a Public Services Office

In this section, we validate our approach from the previous section by applying it to the problem of finding optimal capacity levels for the staffing problem at a public services office in Germany. At this office citizens can apply and collect passports and ID cards, change their address, etc. We compare the results of our integrated approach with the more traditional separated approach that uses a standard decision tree model to estimate arrival rates and subsequently applies the Erlang-C formula to optimize capacities. While the separated approach based on Erlang-C may not be the most sophisticated solution available in the literature it is a relevant benchmark due to its preva-

lence in practice. For more details on the Erlang-C model, see for example Gans et al. [2003].

In our case study the labor laws and labor agreements force employers to assign employees to fixed shifts which is a time window, for example, from 8am to 12.30pm. Hence, we have one 4.5-hour staffing time-slot per day. We have one year (251 working days) of historical data including for each individual customer the time-stamp the customer arrived. These time-stamps are generated by an automated ticketing system: customers enter and have to draw a ticket and are called first-come-first-served once a server is free. Applying the service target to 'serve 80% of the customers in a waiting time below 20 minutes' to this historical data set, only for 35% of the staffing time-slots the service goal was reached. We denote this ratio by $G_{real} = 0.35$. That is, for more than 65% of the days more than 20% of the customers had to wait more than 20 minutes.

Labor laws in Germany prohibit employers to track the individual service times at service desks. However, we know that a typical service task takes around 20 minutes, the minimum service time is 5 minutes and maximum service time can be 'substantially longer than the typical time'. Hence, for each arrival we draw a service time from a triangular distribution with $min = 5$, $max = 60$ and $peak = 20$ minutes.

As features we use day of week, whether the day is a school holiday, in the first week of the month or a bridge day (i.e., a working day between weekend and a single holiday). The prediction model achieves an out-of-sample MAPE of 13.5% in predicting the number of arrivals per time-slot. Just using the mean as prediction would result in a MAPE of 20%.

To evaluate both approaches on the given real-world dataset we use leave-one-out cross validation. I.e., one-by-one we take one observation from the data set which we do not use for training the model, train the model and then evaluate the performance for the left-out observation. As performance measures, we consider the ratio of time-slots, where the service target was

achieved. $\hat{G}$ as defined in Eq. (5.10), i.e.,

$$\hat{G}(\mu(), \mathcal{L}) = \frac{1}{|T|} \sum_{\left(\mu_n^{(*)}, \mathbf{x}_n\right) \in T} \mathbb{1}\left(\mu_n^{(*)} \leq \hat{\mu}(\mathbf{x}_n)\right) \tag{5.10}$$

For $G_{target} = 1$ the integrated approach yields $\hat{G}(\hat{\mu}^{integrated}, T) = 96.0\%$ with a mean assigned capacity of 11.5. The benchmark approach with separate estimation yields $\hat{G}(\hat{\mu}^{separate}, T) = 82.4\%$ with a mean assigned capacity of 9.25. Considering the service target, the integrated approach is clearly better. However, it also requires higher capacity. Using the parameter G_{target} trade-off required capacity and achieved service target in a controlled way.

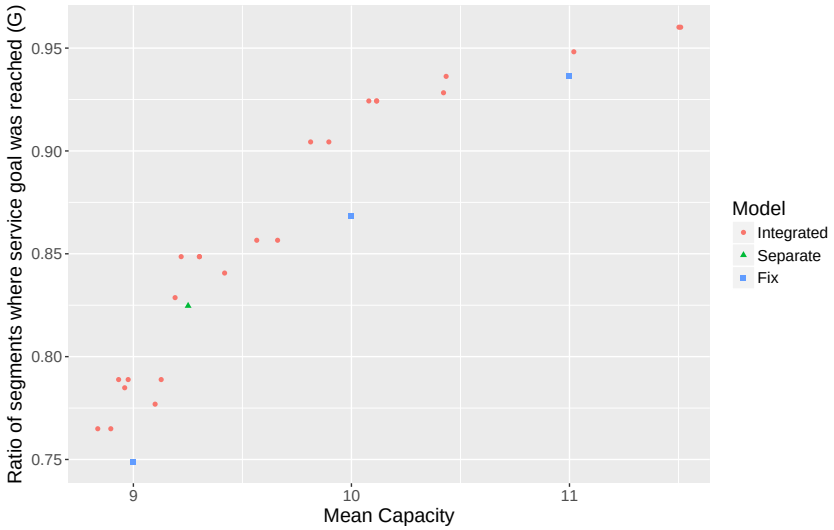


Figure 5.2: The integrated approach easily allows for trading-off the achieved service target and required capacity.

Figure 5.2 shows the ratio of achieved service target with respect to the required mean capacity. Compared to the naive approach where we assign a fixed capacity for all days, we can reduce the number of days where the service target is missed from 16 to 13 days (with mean capacity 11), from 33 to 24 days (mean capacity 10) and from 63 to 53 (mean capacity 9) using the integrated data-driven approach. The sequential approach achieves similar performance as the integrated approach ($\hat{G}(\hat{\mu}^{integrated}, T) = 82.8\%$ with a mean capacity

of 9.19). However, comparability (as well as applicability) is limited due to the lack of an adequate model parameter to evaluate different combinations of capacity and $\hat{G}(\hat{\mu}^{separate}, T)$.

In the following, we also evaluate integrated and separated approach for hourly time-slots and find that our integrated approach clearly outperforms the separated benchmark based on Erlang C. The following table shows the detailed results. We see that with the same capacity requirement our approach reduces the number of time slots where more than 20% of customers have to wait more than 20 minutes by 90 which is a 40% improvement in the performance criterion.

We suppose that the better relative performance of the integrated approach for hourly staffing segments compared to full shifts, where the performance is similar, is due to the following reason: For the longer staffing segments the fluctuations in the arrival processes average out. Since the typical pattern is an increasing arrival rate between 8 and 9am, a peak between 9 and 11am and a decline until 12.30, planning based on the average arrival rate provides acceptable results. For hourly planning, the separated approach leads to significantly worse results since it would assign the same capacity to time-slots with increasing and decreasing arrival rates, as long as the average rate is similar. A more detailed examination is part of our future research.

5.5 Conclusion and further research

In this paper we present a novel, data-driven approach to prescribe optimal capacity levels by directly modeling a functional relationship between features that potentially drive the required capacity and the actual capacity decision. Our main contribution is the integration of the specific optimization problem (minimizing capacity subject to certain service goals) into the estimation of a model that learns the functional relationship between features and decision. We expect that this approach is especially useful if a) arrival rates are not stationary and if b) the non-stationarity is feature dependent. For the staffing problem at a public services office we find that integrated approach signifi-

cantly outperforms the commonly used benchmark approach in the case of hourly planning time-slots.

Based on the basic model presented in this paper, our next steps for the case of the public service office will be to analyze the effect of the length of a planning segment on the relative performance of the integrated approach and the separated benchmark. We will also consider more complex service targets since, for example, from a customer's perspective, the mean waiting time is more relevant than the ratio of time-slots where an arbitrary service target is achieved. Our model allows to simultaneously take multiple service targets into account.

We will also extend our approach to other important capacity planning problems such as call-centers, where we can consider abandonments and multiple agent and customer classes. Call center typically track exact time-stamps for incoming, answering and ending calls. Hence, historical service times are given and we can avoid to work with generated service times. Since service times might as well be feature-dependent, we expect additional potential for integrated data-driven approaches like the one we present in this paper.

Furthermore, the comparison with more sophisticated benchmarks such as the data-driven approach introduced by Bassamboo and Zeevi [2009] is a topic of further research. We expect that given a clustering of similar historical observations the involved optimization procedure described by Bassamboo and Zeevi [2009] will lead to competitive results. However, in a complex practical setting, finding such a clustering might be challenging. We will investigate whether the clustering that comes as a byproduct of our approach can be used for the data-driven approach by Bassamboo and Zeevi [2009].

6 Prescriptive call center staffing

This paper addresses the call center staffing problem. We present a novel prescriptive staffing approach that minimizes the human labor cost and the cost for calls that were abandoned due to excessive waiting times. Our approach is novel in that it determines a prescriptive model based on the functional relationship between observable features such as call volumes in previous staffing segments, school holidays or other events and the optimal staffing decision. In order to abstain from strong assumptions about underlying data distributions, we learn the model from historical data by combining the staffing cost optimization problem with a machine learning algorithm. We analyze the performance of our approach on two real-world data sets and compare it to a state-of-the-art benchmark. Provided with the same information as the benchmark, our approach dominates on both data sets, resulting in a cost improvement of up to 8 percentage points and shows even greater cost improvements when provided with additional features. We can explain the cost advantage of our approach in part with its ability to consider non-random intra-slot patterns in the call arrival such as a trend. [15]

6.1 Introduction

In service systems such as call centers, employee staffing is one of the most important planning tasks. Facing uncertain demand for service capacity and limited available resources, a system manager has to trade off the costs for staffing additional employees against customers' expectations regarding the quality of service. At the same time, labor costs for call center agents con-

[15] This paper is co-authored by Jan Meller.

stitute by far the largest cost driver, accounting for $60 - 80\%$ of the overall operating cost in typical call centers [Aksin et al., 2007].

In this paper, we introduce a novel, prescriptive staffing approach for inbound call centers. The term "prescriptive" addresses two important characteristics of our method [cf. Bertsimas and Kallus, 2019]: First, our approach is based on a functional relationship that enables it to prescribe optimal staffing levels given observable feature values. We assume that such features, e.g., the call volumes on the same weekday in previous weeks, national holidays or further calendar events as well as weather or promotion campaigns, can have a considerable impact on the required staffing levels. Second, our approach is data-driven, that is, we do not make explicit assumptions about underlying distributions (e.g., distributions of inter-arrival or inter-departure times). Instead, we directly "learn" the functional relationship between observable features and staffing decision from a set of historical observations of arrival patterns and feature realisations.

Our approach departs from typical ways to address this kind of staffing problem where one would naturally use a queuing system to model the respective service environment. One central property of such queuing approaches is to impose a distribution on inter-arrival and inter-departure times which often are assumed to follow an exponential distribution. However, empirical studies show that the resulting homogeneous Poisson processes do not accurately reflect the observed call arrivals and service completions [Avramidis et al., 2004, Brown et al., 2005]. As a consequence, authors argue that in many situations, arrival rate uncertainty plays a more significant role for the performance of staffing methods than the inherent variability of the stochastic processes [e.g., Bassamboo et al., 2010]. Considering this type of uncertainty becomes even more important when arrival rates are forecasted and hence are prone to forecasting errors. For this reason, the same authors promote stochastic fluid models which are able to consider such arrival rate uncertainty. Unfortunately, these models implicitly assume that fluctuations of arrival rates are random within a planning segment, and for example, do not exhibit patterns such as a trend [cf. Harrison and Zeevi, 2005, Bassamboo and Zeevi, 2009]. In addition, none of these approaches considers external features that can be

relevant for staffing decisions.

Our contribution to the extant literature is two-fold: First, we present the - to the best of our knowledge - first prescriptive approach for call center staffing which integrates a cost-based staffing objective with a machine learning method (regression tree). Our approach "learns" optimal decisions from historical call arrival data and according feature observations. Second, we validate our approach at the hand of two real-world call center data sets and identify drivers of performance differences between our novel method and a state-of-the-art benchmark based on a stochastic fluid model. We show that our method results in considerably lower costs when using the same information as the benchmark approach. Moreover, we find that additional features further increase this cost advantage.

The remainder of this paper is structured as follows: In section 6.2, we review established staffing approaches from the literature with which we contrast our contribution. Section 6.3 then introduces our prescriptive staffing method and describes the specific implementation with a tree-based machine learning algorithm. Finally, we evaluate and benchmark the performance of this method on two real-world data sets for a setting with homogeneous customers and call center agents in section 6.4.

6.2 Literature review

A natural way to model call center operations is by using queuing systems [Gans et al., 2003]. These approaches have in common that the input stream of customer calls and the stream of service completions are modeled as independent stochastic processes. Due to their appealing internal mechanic, typical staffing goals that, for example, are based on the distribution of customer waiting times can be calculated by assuming some general properties of the system. In the past, one central set of assumptions in the literature were homogeneous Poisson arrival and departure processes in order to determine analytical results for the distributions of steady-state queue length, customer waiting times and the load factor of the servers.

However, empirical studies show that call arrivals in practice often depart from the theoretical assumptions in the literature in a number of ways: As an example, several authors find that call arrival times are often overdispersed, i.e., the standard deviation of their interarrival times is considerably higher than the mean [Jongbloed and Koole, 2001]; call arrival rates vary over the day [Brown et al., 2005, Kim and Whitt, 2014]; and are dependent on the arrival patterns from previous days [Avramidis et al., 2004] – all of which is contrary to the Poisson assumption.

As a result, authors have offered two main avenues to address these issues: First, a stream in the literature models incoming calls by nonhomogeneous Poisson processes [e.g., Liao et al., 2012, Kim and Whitt, 2014] instead of homogeneous Poisson processes. In order to keep the models solvable, the form of the nonstationary arrival patterns in this class of models is limited to rather simple patterns, e.g., by employing a linear function or a doubly stochastic Poisson process. A second stream in the literature provides more flexible modeling opportunities which allow to incorporate external factors that might drive staffing requirements. These approaches typically follow the *pointwise stationary approximation* paradigm [Green and Kolesar, 1991], according to which a day is split into shorter time intervals of equal length for which a constant arrival rate is assumed. Then, the arrival rates for these intervals can be separately forecasted, e.g., by applying time series methods [Taylor, 2012, Saccani, 2013, Ibrahim and L'Ecuyer, 2013, Ibrahim et al., 2016]. Applications of machine learning in the field of call center staffing are scarce. Li et al. [2019] apply machine learning to predict service levels given specific staffing decisions. However, they need to simulate call arrival data in order to have enough training which requires strong parametrical assumptions about the underlying arrival processes.

Forecasting arrival rates based on historical data entails the challenge of how the uncertainty from prediction errors can be considered within the staffing models. While many authors focus either on the forecasting task of arrival rates or on the cost and quality of service implications of stochastic scheduling methods, Gans et al. [2015] note that only few contributions combine sophisticated arrival rate forecasting with stochastic optimization models.

In the same paper, Gans et al. propose an auto-regressive time-series model to estimate the scaled arrival volumes for the staffing segments. In order to deal with the forecast uncertainty, they then generate scenarios for the arrival volumes and feed them into a stochastic programming model to find the staffing levels that minimize expected cost. While providing a way to combine arrival rate forecasting and stochastic call center staffing, their approach relies heavily on strong parametric assumptions concerning the stationarity of arrival rates (piece-wise Poisson) as well as the distribution of arrival rate uncertainty (normally distributed).

In a different stream of the literature [e.g., Harrison and Zeevi, 2005, Bassamboo et al., 2006, Bassamboo and Zeevi, 2009, Bassamboo et al., 2010], some authors argue that in many situations, the uncertainty that is induced by stochastic arrival rates dominates the uncertainty due to the stochastic nature of the interarrival times and hence they neglect the latter entirely. This argumentation provides the avenue and justification of fluid models where actual stochastic processes are replaced by (a distribution of) their rates [e.g., Harrison and Zeevi, 2005, Bertsimas and Doan, 2010]. At the same time, by ignoring the stochastic variability of the call arrival process in itself, the accuracy of the considered call arrival rates has an even larger impact on the quality of the final staffing decisions. Over the last decade, different approaches have been developed that build on the idea of fluid models for call center staffing and combine them with an approach to consider the uncertainty of arrival rates.

Bertsimas and Doan [2010] assume a risk-averse system manager and hence provide two formulations of the staffing problem that aim at protecting against worst-case realizations of the arrival rates. In their first formulation, the α-quantile of the total cost, consisting of staffing, waiting and abandonment penalties, is minimized. In their second model, which builds on robust optimization theory, uncertainty sets are defined that contain all potential realizations of the arrival rates. Then, the worst case outcome considering these potential realizations is minimized. An appealing characteristic of these approaches is the ability to guarantee a certain performance even under very unfavourable arrival rate realizations. On the contrary, these robust solu-

tions tend to be conservative, sacrificing much of the potential cost savings for robustness. Also, in their approach, Bertsimas and Doan do not consider external feature data to generate these uncertainty sets, ignoring potentially predictive information about the actual arrival rate realizations.

A second approach based on fluid models, proposed by Tulabandhula and Rudin [2013], promotes simultaneously solving the forecasting (of the arrival rates) task and the optimization (of the staffing decisions) task. To achieve this, they learn a prediction model where a regularization term that is proportional to the expected operating costs is added to the loss function. Hence, following this approach, the prediction model is already biased in favour of the subsequent optimization task. Then, they apply the simple square-root staffing rule onto the predictions generated with the biased model. We follow a similar idea with some important differences: First, instead of biasing the forecasting model, our approach fully integrates both tasks – that is, we obtain one single optimization model that learns optimal decisions from historical call arrival data. Second, Tulabandhula and Rudin [2013] use the square-root staffing rule which is based on the assumption of Poisson arrivals and departures. In our approach, we do not require this assumption in that we directly use the historical call arrival patterns and thereupon determine ex-post optimal decisions that are independent of any distributional assumptions of the underlying call arrival process.

Finally, Bassamboo and Zeevi [2009] introduce a method to derive data-driven staffing decisions based on historical call arrival data. Their approach builds upon a series of papers introducing stochastic fluid models for call center staffing problems where multiple customer classes and multiple server pools have to be considered [e.g., Harrison and Zeevi, 2005, Bassamboo et al., 2006]. In order to derive staffing decisions for a new staffing segment, empirical estimates for the arrival rates are calculated from samples of past call arrival epochs with similar characteristics. Then, these empirical distributions are fed into the staffing model which minimizes the sum of expected abandonment and capacity costs. While Bassamboo and Zeevi do consider the uncertainty of arrival rates their method is not able to consider structural changes of the mean arrival rate such as a trend. Also, their method requires data from

"similar" past staffing segments which necessitates a preprocessing in the form of a clustering approach. Hence, the choice of relevant characteristics and how to determine such similar observations largely influences the quality of the final staffing decisions. In contrast, our approach comes with integrated feature selection and can also factor in structural changes of the mean arrival rate such as a trend.

Our new prescriptive staffing approach that is presented in the next section adds to the literature in that we provide a novel way of deriving staffing decisions directly from data. Our approach considers both, external features driving staffing requirements, and the uncertainty that arises from estimating their impact on the final prescriptions.

6.3 Data-driven capacity management

In this section we formalize the call center staffing problem and introduce our modeling assumptions. Then, we present the competing approach based on a stochastic fluid model that we will use as benchmark in our analyses. Finally, we describe our data-driven approach to prescribe optimal call center staffing levels by directly modeling the functional relationship between staffing decision and features that potentially drive the required capacity.

6.3.1 Problem statement and modeling assumptions

We consider a call center setting where b identical servers are staffed within a staffing segment to handle arriving calls. We model the incoming calls as a doubly stochastic process $F(t) := (F(t) : 0 \leq t < \infty)$ where the arrival rate $\Lambda(t)$ is itself a random variable with unknown distribution. In the following, $F(t)$ represents the cumulative number of calls up to a time t. Each of these calls gets either directly answered by an idle server, or, if all servers are busy, is assigned to a buffer with infinite capacity. Once connected to a call center agent, the customers' service requirements are modeled as a second, independent random variable. Hence, the stochastic process $S(t) := (S(t) : 0 \leq t < \infty)$ describes the cumulative amount of service comple-

tions up to a time t where μ represents the service rate. Since each customer is endowed with an individual amount of patience, those whose calls could not directly be answered are willing to wait for a maximum of τ minutes until the call is abandoned. Their impatience is modeled as a third random variable and hence, the cumulative amount of abandoned calls up to a time t can be described via the stochastic process $A(t) := (A(t) : 0 \leq t < \infty)$ and abandonment rate γ.

Optimization problem A distinguishing characteristic of our prescriptive staffing approach is the explicit accounting for the call arrival structure within a slot and hence the resulting timing of arrivals, service completions and abandonments. To closely model the actual sequence of events, we adapt a model formulation that has originally been proposed for a more complex setting where single calls from different customer classes have to be routed to agents from different server pools who can potentially handle calls from one or more customer classes [e.g., Harrison and Zeevi, 2005, Bassamboo et al., 2006, Bassamboo and Zeevi, 2009].

We impose a cost-based staffing objective where the system manager aims to minimize the expected total costs resulting from her staffing decision b in a segment of length T. Assume the cost of an abandoned call to be p and the staffing cost per server being assigned to a staffing segment to be c. Then, the system manager's optimization problem can be formalized as:

$$\min_{b \in \mathbb{R}_+} \mathcal{V}(b) := c\, b + p\, \mathbb{E}\left[A\left(\int_0^T \gamma Q(s) ds\right)\right] \tag{6.1}$$

$$\text{s.t. } D(t) \leq b \tag{6.2}$$

$$Q(t) = Z(t) - D(t) \geq 0 \tag{6.3}$$

$$Z(t) = F(t) - S\left(\int_0^t \mu D(s) ds\right) - A\left(\int_0^t \gamma Q(s) ds\right), \tag{6.4}$$

where the *server process* $D(t)$ represents the number of servers engaged in customer calls at time t which we require to capture the timing and routing of single calls to agents. The first constraint (6.2) guarantees that the number of currently active servers $D(t)$ can not exceed the total number of available

servers. Constraint (6.3) links $D(t)$ with the *queue length process* $Q(t)$, which can be interpreted as the number of customers currently waiting in the buffer, and the *headcount process* $Z(t)$ that represents the number of customers in the system. Constraint (6.4) is the system dynamics constraint with $F(t)$ constituting the cumulative arrivals up to t, the second term being the cumulative service completions up to t and the third term being the cumulative abandonments up to time t. The three additional processes, $Z(t)$, $Q(t)$, $D(t)$, are defined over the time domain $[0, T]$ and take values in $\mathbb{R}_+$.

6.3.2 A stochastic fluid model-based benchmark

Given the stochasticity of the involved processes, the problem described by (6.1) is particularly hard to solve. For this reason, in recent years authors have proposed approximations by *fluid models* [e.g., Harrison and Zeevi, 2005, Bassamboo and Zeevi, 2009, Bassamboo et al., 2010] to this kind of staffing problem. Fluid models are based on additional assumptions with which the original problem is approximated. First, one assumes all stochastic processes to be Poisson flows. Then, one replaces these flows in the system with their rates, i.e.,

$$F\left(\int_0^t \Lambda(s)ds\right) \approx \int_{s=0}^t \Lambda(s)ds, \tag{6.5}$$

$$S\left(\int_0^t \mu D(s)ds\right) \approx \int_{s=0}^t \mu D(s)ds, \tag{6.6}$$

$$A\left(\int_0^t \gamma Q(s)ds\right) \approx \int_{s=0}^t \gamma Q(s)ds. \tag{6.7}$$

The main idea of this approximation is to treat stochastic variability of the customer arrivals, service requirements and abandonments as insignificant compared to variations in the rates themselves [Harrison and Zeevi, 2005]. Moreover, one assumes the system to instantaneously reach a steady-state equilibrium, i.e.,

$$\Lambda(t) = \mu D(t) + \gamma Q(t), \tag{6.8}$$

which constitutes a pointwise stationary approximation, replacing constraint (6.4). Given these approximations, the staffing objective (6.1) can be approx-

imated by $V(b)$ [cf. Bassamboo and Zeevi, 2009, Harrison and Zeevi, 2005]:

$$V(b) := cb + T \int p(\lambda - b\mu)^+ dG(\lambda), \tag{6.9}$$

where $G(\lambda)$ represents the cumulative distribution function of the arrival rates λ. Since we consider a very simple case – one customer class and one agent pool – the objective (6.9) can be further reduced to the well-known *newsvendor problem* [cf. Harrison and Zeevi, 2005] which gives us the following characterization of the optimal pool size:

$$G(b^*\mu) = 1 - \frac{c}{Tp\mu}. \tag{6.10}$$

Of course, in practice, it is not realistic to assume full knowledge of the distribution of arrival rates $\Lambda(t)$. Instead, a system manager would have access to records of historical call arrivals during past staffing segments. Let F_l be the record of realizations of the call arrival process $F_l(t)$, that is, the cumulated arrivals up to a time $t, t \in [0, T]$, during the historical staffing segment l. For now, let's assume, that all historical staffing segments are "similar". Then, the complete data set of n historical call arrival patterns is $\mathcal{R}_n = \bigcup_{l=1}^{n} F_l$. Since the actual arrival rates can not be observed, Bassamboo and Zeevi [2009] propose a method to calculate the linear approximations $\hat{\Lambda}_l(s)$ of the arrival rates on the window $w > 0$:

$$\hat{\Lambda}_l(s) = \frac{F_l(s+w) - F_l(s)}{w}. \tag{6.11}$$

Based on these estimates, the empirical cumulative distribution function of the arrival rates is calculated as follows:

$$\hat{G}_n(\lambda) = \frac{1}{T} \int_0^T \frac{1}{n} \sum_{l=1}^{n} \mathbb{1}_{\{\hat{\Lambda}_l(s) \leq \lambda\}} \, ds, \ \lambda \in \mathbb{R}_+. \tag{6.12}$$

Then, one can determine the optimal staffing decision $\hat{b}^*$ as:

$$\hat{b}^* := \frac{\hat{G}_n^{-1}\left(1 - \frac{c}{Tp\mu}\right)}{\mu} \tag{6.13}$$

In the following we refer to this approach as stochastic fluid model (SFM) approach and use it as benchmark for our evaluations in section 6.4.

6.3.3 A novel prescriptive analytics approach

The fluid model approximation approach presented in subsection 6.3.2 is subject to two major limitations: First, it implicitly assumes that the data $\mathcal{R}_n$ consist of "similar" staffing segments, i.e., that all past arrival processes behave similarly. However, practice shows that the actual structure of call arrival processes is often driven by external factors such as day of the week, week of the month or by holiday periods. Typically, one would cast such information into *features*, i.e., summarized representations of these factors that can be considered in vectorial form, e.g., whether a particular historical staffing decision was taken on a Monday or a Saturday. In the following, we denote such a feature vector for a particular staffing segment l as $\vec{x}_l$. Second, while the arrival rate uncertainty is considered, the fluid model approximation ignores the stochastic variability of the modeled processes, i.e., the variability in the inter-arrival times of single calls, the service requirements as well as the abandonment times.

In the following, we introduce a novel data-driven approach that combines the optimization logic that was formalized in subsection 6.3.1 with state-of-the-art machine learning techniques to *learn* a staffing prescription function $b(\vec{x})$ from historical data. As a prerequisite, assume for now that there exists a cost function $C : \mathcal{F} \times \mathcal{B} \longrightarrow \mathbb{R}_+$ that assigns a real-valued staffing cost to each combination of a call arrival pattern $F \in \mathcal{F}$ over the time horizon $[0, T]$ and a staffing decision $b \in \mathcal{B}$. This assumption as well as a way how to approximate such a function $C(\cdot, \cdot)$ are further detailed in the next paragraph. Moreover, we require a functional relationship between the features $\vec{x}$ and the call arrival pattern $F(t)$ with joint distribution function $f_{F \times X}$. Then, we find a function $b : \mathcal{X} \longrightarrow \mathbb{R}_+$ that maps from the domain of features to the respective optimal staffing quantity. We determine this function $b(\cdot)$ by minimizing the expected

total staffing costs $C(F, b(\cdot))$ given the vector of auxiliary features $\vec{x}$:

$$\min_{b(\cdot)\in\mathcal{B}} \mathbb{E}_{F\times X}[C(F, b(\vec{x}))|X = \vec{x}]. \tag{6.14}$$

However, in a practical setting, the distribution function $f_{F\times X}$ is not observable and estimating such a distribution from data is error-prone in high-dimensional feature spaces ("big data").

F_l	
t	$F_l(t)$
00:00:00	0
00:00:01	0
...	...
00:03:21	1
...	...
00:04:37	2

l	x_{mon}	...	x_{sat}	x_{staff_segm1}	...	x_{staff_segm11}	$x_{is\_holiday}$	$x_{lagged\_call\_vol}$
1	0	...	0	0	...	0	0	312
2	0	...	0	0	...	0	0	247
3	0	...	0	0	...	0	1	210
...	...	...	...	...	...	...	...	...
l	0	...	0	0	...	0	0	207
...	...	...	...	...	...	...	...	...
n	0	...	0	0	...	0	1	265

Figure 6.1: Overview of the structure of the data set $\mathcal{T}_n$. For each historical staffing segment l, a vector $\vec{x}_l$ of descriptive features as well as a record of the sequence of call arrivals $F_l(t)$ within the staffing segment is available.

For this reason, we choose a different approach: We apply the well-established machine learning principle of empirical risk minimization to directly learn the prescription function $b(\cdot)$ from historical data. To that end, we first augment the data set $\mathcal{R}_n$ with vectors of the respective historical feature values $\vec{x}$. The data set $\mathcal{T}_n$ which is used in the subsequent step consists of tuples $(F_l, \vec{x}_l),\ l = 1, ..., n$ and hence $\mathcal{T}_n := \bigcup_{l=1}^{n}(F_l, \vec{x}_l)$. Figure 6.1 provides an overview of the data set: For each collection of observed call arrivals we have a vector of additional information, e.g. whether the considered staffing segment was on a Monday or a Friday, and during which time period of the day the according staffing decision had to be made. Now we can replace (6.14) and instead minimize the empirical counterpart over the training data $\mathcal{T}_n$:

$$\min_{b(\cdot)\in\mathcal{B}} \sum_{l=1}^{n} C(F_l, b(\vec{x}_l)). \tag{6.15}$$

Learning the prescription function Our approach is based on the assumption that when two historical staffing segments are similar in their known properties, e.g., the same weekday, or the same time slot within the day, they are also similar in their unknown properties, i.e., in the call arrival pattern and hence, in the optimal staffing decision. For this reason, our goal is to retrace the functional relationship between the properties known in advance – the features $\vec{x}$ – and the optimal staffing decision b^*. Clearly, considering the complexity of the problem as well as the dimensionality of the potential feature space, it is infeasible to find a globally optimal $b(\vec{x})$ from all possible functions that map the features to staffing decisions. For this reason, we have to choose a function class that restricts the degrees of freedom the ultimate staffing function has. Since the cost function $C(F_l, b(\vec{x}))$, i.e., the evaluation of (6.1) for a given record of call arrivals F_l and a staffing quantity $b(\vec{x})$ is highly complex and not analytically defined (see the following subsection for our approach of approximating $C(F_l, b(\vec{x}))$), the potential candidate space of function classes is limited. As an example, artificial neural networks, one of the currently most powerful approaches to predictive problems, rely on the gradient descent method that is not applicable to that kind of loss function. For this reason, we propose to use a regression tree model [cf. Breiman et al., 1984], respectively its bagged variant, the random forest, which do not need to explicitly determine a gradient. These models are able to generally model very complex feature-demand relationships and have proven to perform well in various settings [see, e.g., Caruana and Niculescu-Mizil, 2006, Caruana et al., 2008]. In the following, we describe the adapted variant of the tree-learning algorithm that lets us "learn" optimal staffing decisions.

Let $\mathbb{L}_{\mathcal{T}'_n}(b)$ be the loss function in terms of total staffing costs for a staffing decision b over the historical staffing segments $\mathcal{T}'_n \subset \mathcal{T}_n$:

$$\mathbb{L}_{\mathcal{T}'_n}(b) = \sum_{l:(F_l,\vec{x}_l)\in\mathcal{T}'_n} \Big(C(F_l, b)\Big). \tag{6.16}$$

The procedure is then as follows: Start with the data sample of all historical

demand observations $\mathcal{T}_n$ and calculate the optimal staffing decision as follows:

$$\begin{aligned} b_{\mathcal{T}_n} &= \arg\min_{b \in \mathbb{R}_+} \mathbb{L}_{\mathcal{T}_n}(b) \\ &= \arg\min_{b \in \mathbb{R}_+} \left\{ \sum_{l:(F_l, \vec{x}_l) \in \mathcal{T}_n} \Big(C(F_l, b) \Big) \right\}. \end{aligned} \tag{6.17}$$

Then, we aim at finding a combination of a feature x_ϕ and splitting point σ that optimally splits the feature data space $\mathcal{X}$ along x_ϕ at σ into two subregions $\mathcal{T}_n^1 \subset \mathcal{T}_n | x_\phi \leq \sigma$ and $\mathcal{T}_n^2 \subset \mathcal{T}_n | x_\phi > \sigma$. The optimal splitting combination (x_ϕ, σ) is found by solving the following problem:

$$\begin{aligned} (\phi^*, \sigma^*) = & \underset{(\phi,\sigma) \in \Big(\Phi, \mathcal{X}_\phi\Big)}{\operatorname{argmin}} \Big(\mathbb{L}_{\mathcal{T}_n^1}(b) + \mathbb{L}_{\mathcal{T}_n^2}(b) \Big) \\ = & \underset{(\phi,\sigma) \in \Big(\Phi, \mathcal{X}_\phi\Big)}{\operatorname{argmin}} \Big(\sum_{l:(F_l, \vec{x}_l) \in \mathcal{T}_n^1} \Big(C(F_l, b) \Big) + \sum_{l:(F_l, \vec{x}_l) \in \mathcal{T}_n^2} \Big(C(F_l, b) \Big) \Big), \end{aligned} \tag{6.18}$$

where Φ is the index set of all features and $\mathcal{X}_\phi$ the set of all values of the ϕ-th feature in the learning data. Then, the procedure sorts all historical observations into the subsamples defined by the determined split above and repeats this procedure greedily for the subsamples $\mathcal{T}_n^1$ and $\mathcal{T}_n^2$ until no further loss reduction can be achieved. We interpret each final subsample as a leaf node or region r in the feature space and denote the set of samples in each region by $\mathcal{T}_n^r$ with $r = 1, ...R$ and R the number of regions. We then obtain a staffing decision for a new, unseen staffing segment by sorting the new instance into the tree based on its feature configuration $\vec{x}_{\text{new}}$ and returning the optimal staffing prescription depending on the respective region $\mathcal{T}_r$:

$$b(\vec{x}_{\text{new}}) = \sum_{r=1}^{R} b_{\mathcal{T}_n^r} \mathbb{1}(\vec{x}_{\text{new}} \in \mathcal{T}_n^r). \tag{6.19}$$

Figure 6.2 visualizes an exemplary tree:

Approximating the cost function In order to apply the tree-based learning algorithm described above, we require a function that assigns a particular cost

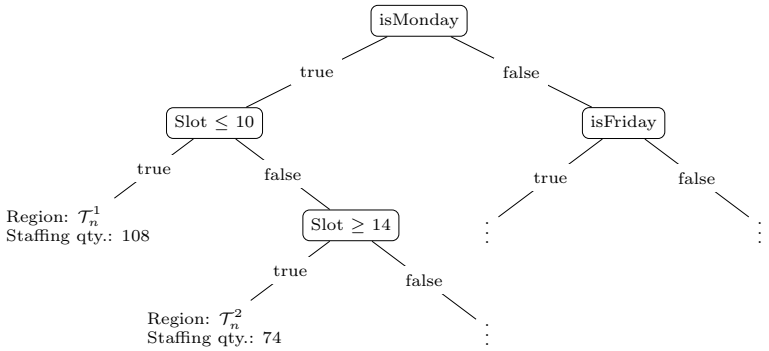


Figure 6.2: Exemplary outcome of our prescriptive analytics approach for one tree. The slot feature represents the staffing slot on a particular day (e.g., on a Monday, from 8.00am to 9.00am).

to a staffing decision given a call arrival pattern F_l:

$$C(F_l, b) = c \cdot b + pN_l^A(b), \tag{6.20}$$

with $N_l^A(b)$ denoting the number of abandoned customer calls due to excessive waiting times. Clearly, this number of abandonments depends on the actual call arrival pattern, i.e., the timing of individual call arrivals, as well as the number of staffed call center agents b. Since we possess the time stamps of actual call arrivals $f_1, \ldots, f_k$, we can closely retrace the actual call arrival arrival process. In the following, we focus on the call arrival uncertainty and for this reason assume service times s per customer and customer patience times v to be deterministic. However, we note that our approach is independent of such an assumption and would also let us reproduce the timing of events under much more complex service and customer patience time distributions. Algorithm 4 in the appendix presents the logic we then implemented in order to determine the set of abandoned calls A for a specific staffing segment l and a staffing decision b. The number of abandoned calls is then given by: $N_l^A(b) = |A|$, the size of the set A. We note that with algorithm 4 we can also consider transient effects, that is, we can account for busy servers and queues from previous time-slots when starting new intervals.

6.4 Evaluation

In this section, we evaluate and compare our *prescriptive staffing model (PSM)* from the previous section with a stochastic fluid model (SFM) as described in subsection 6.3.2.

6.4.1 Evaluation strategy

The goal of our evaluation is two-fold: First, we explore the performance of the PSM method in comparison to the SFM benchmark under fair conditions, that is, when both approaches are fed with the same information. To this end, we analyze the cases of two different call centers. The first call center is situated in the Netherlands and answers more than 400 calls/hour on average. Many of the staffing methods developed in the literature are particularly dedicated to call centers of a similar size since on the one hand, large call volumes and, as a consequence, high arrival rates shrink the observation error [Bassamboo and Zeevi, 2009] which allows for the internal representation of the call arrival pattern via arrival rates with sufficient accuracy. On the other hand, staffing large call centers allows for neglecting the impact of integrality constraints of the number of prescribed call center agents in a staffing slot. The second call center under consideration however, is much smaller (≈ 90 calls/hour on average) and hence provides us with the opportunity to retrace the impact of these factors on the staffing performance in a different setting.

Second, our prescriptive staffing model is designed to process further information that might or might not be relevant for the staffing decision via the input feature vector $\vec{x}$. For this reason, we perform a second series of analyses where we consider additional features such as lagged information about past call arrival patterns, holiday and weekday information.

Figure 6.3 provides an overview of the performed analyses. In the following, we detail our data sets and describe the design choices of our implementation of the respective models. Finally, we define the metrics being used to measure performance in our settings.

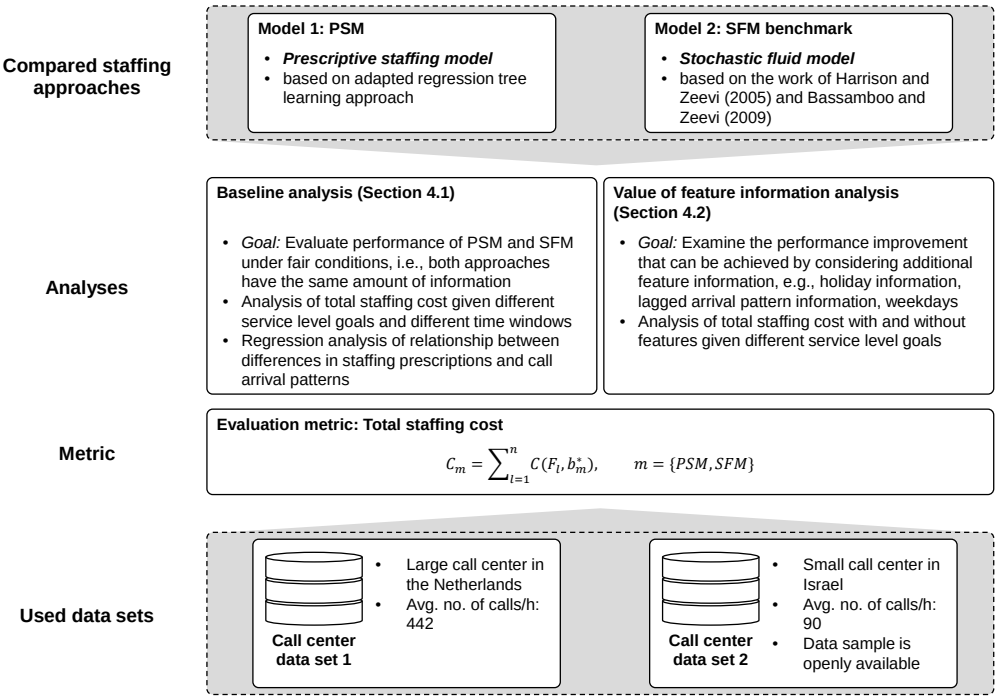


Figure 6.3: Overview of our evaluation procedure

Data and descriptives

We consider two different data sets from real-world call centers for our evaluations. Both data sets contain records of individual call arrival times along with information about waiting times, service times or, if applicable, the times when a customer has abandoned a call.

Call center 1: Public services in the Netherlands The first data set captures the call arrivals in a call center of a large city in the Netherlands over the period from 01/01/2014 to 12/31/2014. The call center offers services regarding, e.g., local taxes or parking fees. At peak times, a maximum of 113 call center agents handle calls from different lines, with call volumes of 442 calls/hour on average. Table 6.1 summarizes central characteristics of the incoming call pattern for one of the planning slots, from 8 am to 9 am. At a first glimpse, we note a strong intra-week seasonality in the call volume – with Mondays being by far the busiest weekdays whereas Fridays – the slowest days

– only see about 2/3 of the call volume of a Monday. Also, we have calculated the empirical coefficients of variation of the call arrivals, i.e., the standard deviation of arrival volume divided by its mean (CV_{emp}). This actual CV is contrasted with the theoretical coefficient of variation that we would expect if we assumed the arrivals to follow a Poisson process (CV_{Pois}). Clearly, the empirical values show a considerably larger variation than the Poisson values. This "overdispersion" is a commonly observed phenomenon in the call center staffing literature. Hence, it is obvious that the call arrivals within these data samples – which already reflect single planning slots – cannot be accurately modeled by a homogeneous Poisson process.

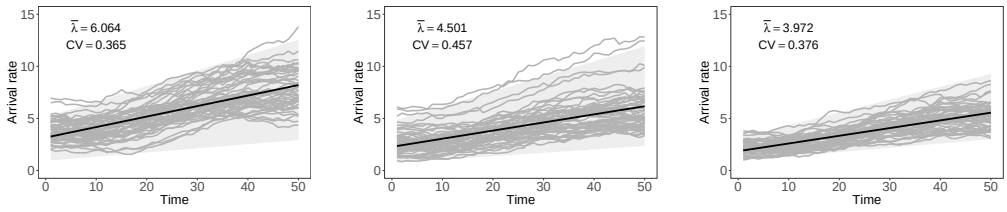
Day	**Calls/hour**	CV_{emp} **(in %)**	CV_{Pois} **(in %)**
Monday	329	21.7	5.5
Tuesday	243	34.1	6.4
Wednesday	241	23.5	6.4
Thursday	212	20.1	6.9
Friday	204	24.4	7.0

Table 6.1: Empirical data of arrivals in time slot 8am to 9am by week day for the large call center.

Figure 6.4 visualizes the underlying arrival patterns in more detail. We find that within these slots, the incoming calls follow a clearly identifiable trend with almost steady increases in the call arrival rate. We would expect that, given such a non-random pattern, the prescriptive staffing model PSM uses the timing of these single call arrivals to its advantage over the stochastic fluid model which only considers the empirical distribution of arrival rates.

Call center 2: "Anonymous Bank" in Israel The second data set contains calls to a telephone call center of a bank in Israel. The reported call data ranges over a period of 12 months, from 01/01/1999 to 12/31/1999, and is described in detail in Mandelbaum et al. [2001][16]. Table 6.2 provides an

[16]The data set is freely available and can be accessed at `http://iew3.technion.ac.il/serveng/callcenterdata/index.html`.



(a) Mondays (b) Tuesdays (c) Thursdays

Figure 6.4: Arrival rates on different weekdays from 8.00 to 9.00 am as well as the coefficient of variation of the slopes calculated for a single planning segment. The black line represents the mean slope of the arrival rates, the area shaded grey represents the 0.95 prediction interval of the slope of arrival rates.

excerpt of the provided information. A caller would first be connected to a voice response unit (vru) where one must identify oneself and then has the option to perform self-service transactions. After that, the caller could be either connected to an agent, or is assigned to the queue until the next agent becomes available.

Call_id	Date	Vru_entry	Q_start	Q_exit	Ser_start	Ser_exit	Server
33116	1999-01-01	00:00:31	00:00:36	00:03:09	00:00:00	00:00:00	NO_SERV
33117	1999-01-01	00:34:12	00:00:00	00:00:00	00:00:00	00:00:00	NO_SERV
33118	1999-01-01	06:55:20	06:55:26	06:55:43	06:55:43	06:56:37	MICHAL
33119	1999-01-01	07:41:16	00:00:00	00:00:00	07:41:25	07:44:53	BASCH
33120	1999-01-01	08:03:14	00:00:00	00:00:00	08:03:23	08:05:10	MICHAL

Table 6.2: Excerpt of the data structure from the small call center

This second call center possesses very differing properties from the first call center: First, we have a much smaller call volume: $\approx 37,000$ calls per month result in about 90 calls per hour on average. Second, the call center agents also perform outbound calls that might change the behavior of queuing and service completions. For our purposes, however, we only focus on the call arrival process.

Table 6.3 reveals that, similar to the situation in the larger call center, the empirical CV is considerably higher than the theoretical CV if we assumed a homogeneous Poisson call arrival process. This is again a sign for

Day	**Mean arrivals** $\frac{1}{\hat{\lambda}}$	$CV_{empirical}$ **(in %)**	$CV_{Poisson}$ **(in %)**
Sunday	93	21.0	10.3
Monday	89	19.6	10.6
Tuesday	88	23.1	10.6
Wednesday	85	27.0	10.8
Thursday	83	26.0	10.9
Friday	70	25.1	11.9

Table 6.3: Empirical data of arrivals in time slot 8am to 9am by week day for the small call center.

overdispersion of call arrivals.

Feature engineering

Due to its granularity, the raw data described above cannot directly be processed in order to generate recommendations for staffing decisions. Also, despite a lot of predictive information such as trend and seasonality patterns can usually be extracted from historical information, further auxiliary data like information about national holidays or special events and campaigns often provide relevant information that should be considered for staffing decisions. Hence, in order to apply the PSM as described in section 6.3, some data transformation and preprocessing steps are necessary to obtain predictive features, i.e., summarized representations of auxiliary data.

For our experiments, we consider three different groups of such features which are provided in Table 6.4. In our first analysis, only features that are directly related to the specific planning slot, i.e., information about the particular weekday and time of the day that has to be staffed are provided as information to the planning methods. Then, in the second series of experiments in subsection 6.4.3, further features are derived: On the one hand, we derive time series-related features such as the number of calls in the same planning segment on the previous day (and the week), relative to the average number of calls for all previous segments from similar weekday and time slot. Second, we also derived features that capture the information whether the

respective day is a national holiday in the country the call center is situated in which is expected to have a considerable impact on the call volume.

Source	Feature
Planning Features	Binary week day features: E.g., is the particular day a Monday? Binary planning slot features: E.g., is the planning slot 8:00am to 9:00am?
Time Series	Number of arrivals in the same slot on previous day (and week) relative to mean number of arrivals in historical slots at the same weekdays
Calendar	Binary holiday feature: Is the particular day a holiday, within school holidays, a bridge day, or in the first week of a month?

Table 6.4: Overview of the included features

Implementation and parameter choices

In order to apply PSM and SFM to the two call center settings, both approaches require several design and implementation choices. First, a central input for our benchmark method is the empirical distribution of the arrival rates. As Bassamboo and Zeevi [2009] note, the accuracy of such an empirical distribution depends strongly on the choice of the window length w over which the number of call arrivals is counted and then divided by the window length in order to determine the marginal arrival rates in a rolling window approach as described in equation (6.11). The same authors propose a rule of thumb to determine such a window length depending on the maximum of call arrivals by:

$$\widetilde{w} = \frac{1}{\sqrt[4]{N_{max}}}. \tag{6.21}$$

Based on this approach, we determine $\widetilde{w}$ for the large call center to be 10 minutes, and for the small call center 12 minutes. Then, we assume customers to be assigned to agents following a simple first-in, first-out control policy.

Furthermore, caller patience v as well as service time s are assumed to be deterministic for each customer and hence are determined for each of the data sets separately. Moreover, in our experiments we also control for the impact of the cost configuration between penalty costs for abandoned calls and for capacity costs for servers, i.e., the service level that is calculated as:

$$SL = 1 - \frac{c}{Tp\mu}. \tag{6.22}$$

We assume several service level configurations and report the realized costs for both staffing approaches. The following table 6.5 summarizes the parameter configurations that we have calculated from the actual arrival data for both call centers.

Parameters	**Large call center**	**Small call center**
Arrival window $\widetilde{w}$	10 min	12 min
Patience time v	1.69 min	1.31 min
Service time s	5.56 min	3.2 min

Table 6.5: Parameter settings for our analyses

Procedure and metrics

In the following subsections, two analyses are performed to identify drivers for the performance of the new prescriptive staffing method PSM in comparison to the SFM benchmark method. We use data from the two call centers detailed above which both contain 12 months of call arrival data. These data sets are split into five roughly equally-sized subsamples. Then, we follow a five-fold cross-validation approach: We take the first four subsamples to "learn" respectively calibrate the staffing models and use the fifth subsample as a separate test set to evaluate the performance of the respective staffing prescriptions. Then, we permutate the folds and repeat this procedure until we have used each fold once as a test set and four times within the training set and hence have generated out-of-sample prescriptions for each historical

staffing segment.

As described in Figure 6.3, our first analysis aims at examining the performance of both PSM and SFM under fair conditions. Here, only the features that define the respective planning slot – the week day as well as the time slot within that day – are considered as input to the methods. The PSM method is directly trained with these features whereas the SFM is calibrated for each planning slot separately by pre-clustering the training data as to provide the stochastic fluid model only with historical instances that are "similar" to the staffing segment that should be planned for.

In the second analysis, also the impact of feature information is examined. Here, all the features described in table 6.4 are handed over to a second PSM approach. Then, the latter is again trained on each respective (augmented) training data set and the staffing prescriptions are evaluated similarly to the analysis in section 6.4.2.

In order to report cost performance, we first calculate the *ex-post optimal cost* for the respective data set:

$$C^* = \min_b \sum_{l=1}^{n} C(F_l, b). \tag{6.23}$$

Then, for each staffing model $m = \{PSM, SFM\}$, the respective out-of-sample cost is calculated per fold and then summarized over all five folds as follows:

$$C_m = \sum_{f=1}^{5} \sum_{l=1}^{n_{f,test}} C(F_l, b_m(\vec{x}_l). \tag{6.24}$$

Finally, we report the relative gap to optimality which is defined as follows:

$$\Delta_{rel,m} = \frac{\delta_{abs,m}}{C^*} = \frac{C_m - C^*}{C^*}. \tag{6.25}$$

Based on this metric, we assess the performance of both models and derive structural insights.

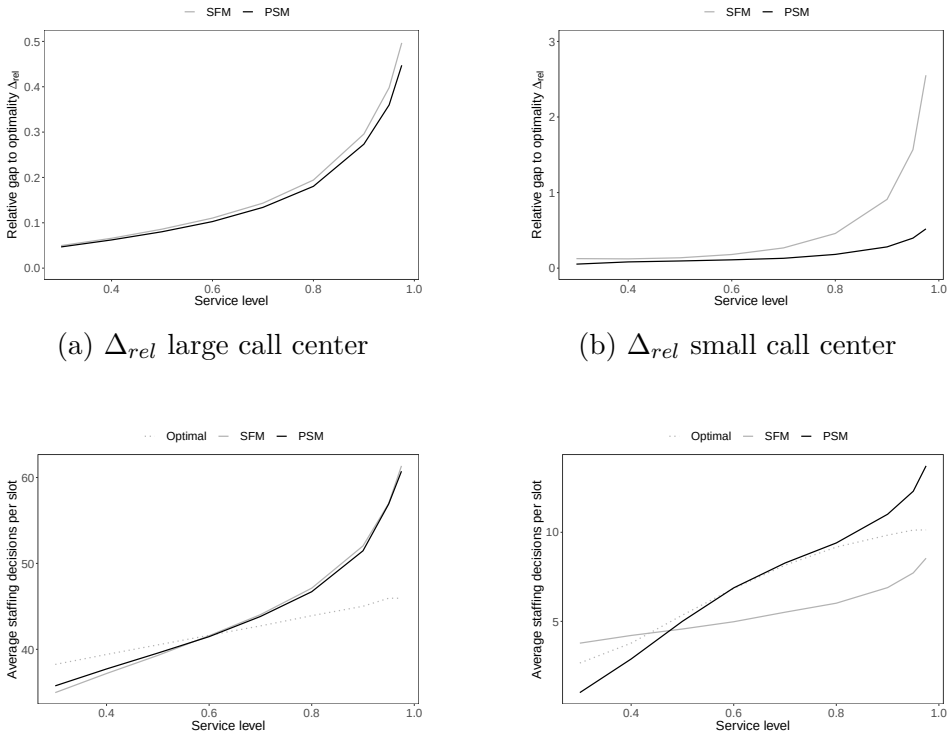
6.4.2 Baseline analysis

In our first analysis, we examine the performance of PSM and SFM when both approaches use the same information, that is, we only consider planning slot (i.e., hour of the day) and day of the week as features.

Figure 6.5 visualizes the cost performance of both methods reported as gap to optimality depending on the service level for both call centers separately. We find that for low service levels, both methods perform similarly well but the relative gap to optimality increases with increasing service levels. The reason for the increasing gap with increasing service level is that protecting against uncertain call arrival volumes becomes more expensive due to higher call abandonment costs and higher cost for additional safety capacities in terms of additional call center agents. The baseline, i.e., the ex-post optimal decision has perfect information and is not affected by uncertainty. The effect of the service level is also reflected in Figure 6.5 where particularly for the large call center setting (Figure 6.5c), both methods tend to overstaff in comparison to the ex-post optimal staffing decisions for high service levels.

However, we also note structural differences between the performances of both methods: Apparently, for high service levels, the PSM method results in notably lower costs (measured as gap to optimality) than the SFM method. For example, for the large call center and a service level of 95% (which corresponds to the actual service level), we find that PSM leads to a 3.81 percentage points lower gap to optimality compared to SFM which translates into a calculated cost difference of 8.97 units (where 7.78 are the costs per server and 4.17 are the costs per abandonment). In the case of the smaller call center, the SFM leads to an even larger gap of optimality for the highest service levels. As an example, for a service level of 90% (corresponding to the actual service level in the small call center) we find that PSM leads to a 62.9 percentage points lower gap to optimality compared to SFM which translates into a difference in calculated costs of 48.97 units (where 5.12 are the costs per server and 9.48 are the costs per abandonment).

These large performance differences between both approaches depending on the specific call center setting motivate our subsequent analyses. We can



(a) Δ_{rel} large call center

(b) Δ_{rel} small call center

(c) Decisions large call center

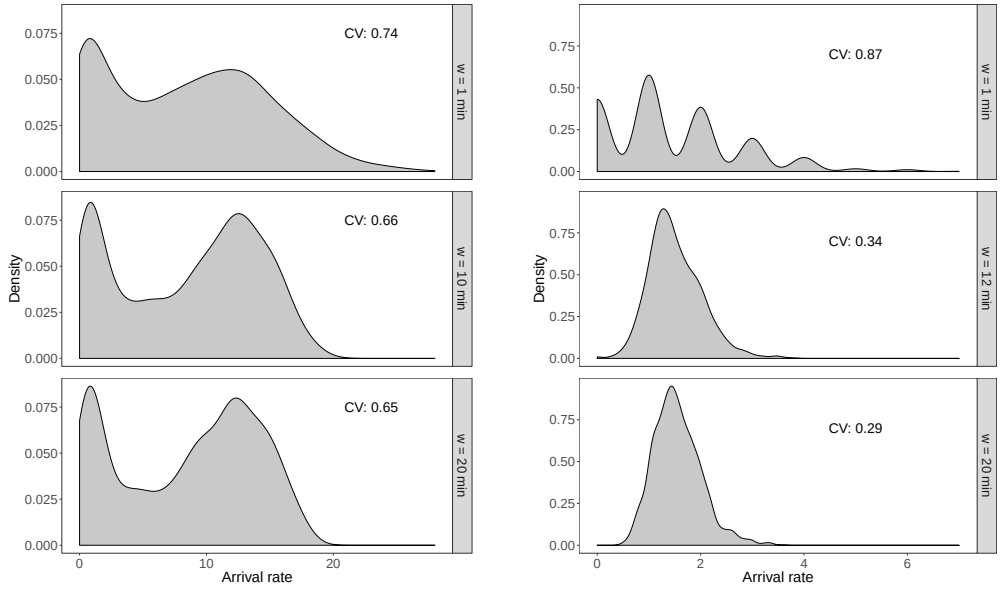
(d) Decisions small call center

Figure 6.5: Relative gap to optimality Δ_{rel} and average staffing decisions depending on service level configuration.

explain the very large performance gap between PSM and SFM in the small call center by the observation that the new PSM method is able to better capture the uncertainty that is inherent in the call arrival pattern. This is confirmed by the behavior illustrated in Figure 6.5d: SFM on average leads to significantly understaffed planning slots, planning even less call center agents than the ex-post optimal solution. This is very counterintuitive since we would expect a similar behavior as in the large call center setting where both staffing methods that have to consider uncertainty would add a generous safety capacity buffer compared to the ex-post optimal staffing decision.

We retrace the observed performance difference to two different factors: First, as Bassamboo and Zeevi [2009] point out, the correct choice of the window length w might strongly affect the observation error (of arrival rates) and

as such have an impact on the staffing performance. Figure 6.6 illustrates the effect of different window lengths w on the calculated arrival rate distributions for both call centers. The middle panels represent the choices of w that were calculated by equation (6.21) according to the approach proposed by Bassamboo and Zeevi. We find that while for the large call center, the coefficients of variation of the arrival rates fluctuate in the relatively small range of $[0.65, 0.74]$, the impact on the coefficients of variation in the small call center is much higher with values between 0.29 and 0.87.

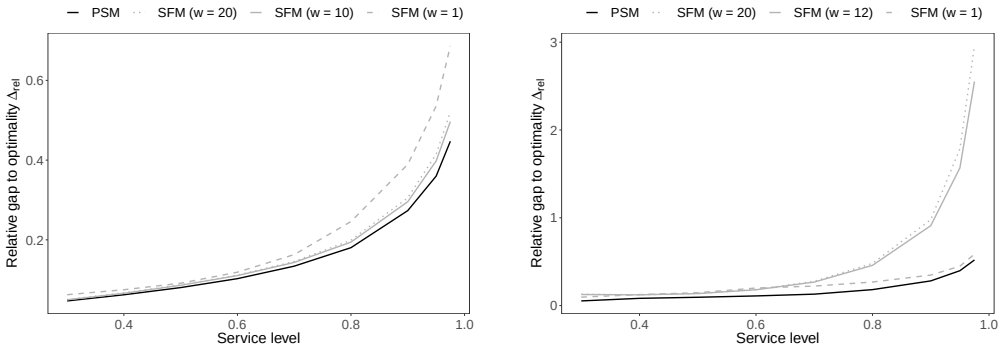


(a) Large call center (b) Small call center

Figure 6.6: Distribution of calculated arrival rates for different window sizes w

Figure 6.7 visualizes the impact of these different choices of w on the performance of the SFM method. We see that the shorter window length (resulting in the considerably higher coefficient of variation) helps the SFM method to better capture the uncertainty of the arrival rates in the small call center setting and hence leads to a much better staffing performance for high service levels. Nevertheless, although part of the performance difference between PSM and SFM can be explained by a suboptimal choice of the window length, the SFM's performance remains inferior to the PSM's performance

under all examined configurations.



(a) Large call center (b) Small call center

Figure 6.7: Staffing performances for different window sizes w

For this reason, we retrace the rest of the performance differences to the call arrival patterns themselves or, more specifically, we presume that our PSM method is better in handling intra-slot structure of the call arrivals, e.g., if the call arrival pattern within a planning slot reveals a trend. Figure 6.8 shows boxplots of the average cost differences between SFM and PSM from the large call center setting depending on the average absolute slope of each planning slot. We find that the cost advantage is significantly higher for slots with the highest third ordered by average absolute slopes compared to the third with the lowest average slopes. The median cost differences are 16.0, 11.6 and 6.10 for high, medium and low slope. For the large call center, the average absolute slope ranges between 0.5% and 9.0%. For the small call center, the largest average absolute slope value is only 1.5%. Given these negligeable absolute slopes, we can not perform a similar analysis as for the large call center.

6.4.3 Value of feature information analysis

The previous analysis has shown that the new prescriptive staffing method PSM is able to outperform its state-of-the-art competitor, SFM, given the same information on the slot that has to be planned. However, besides exploiting structural similarities between the call arrivals in single slots, a dis-

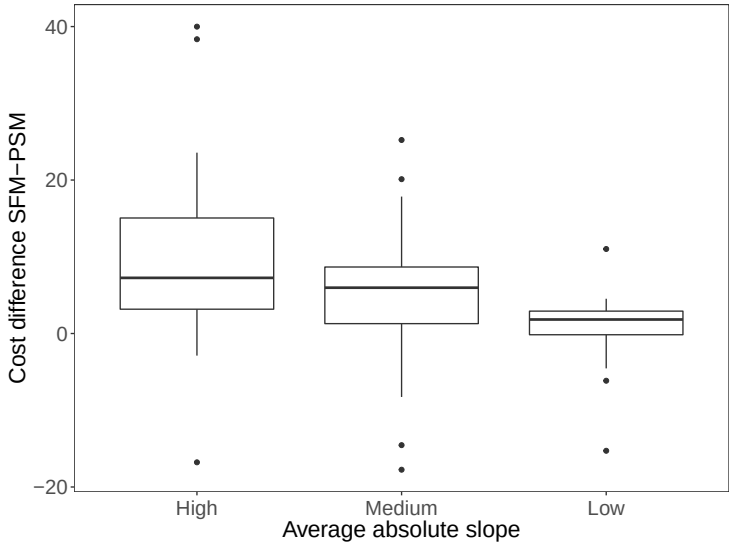
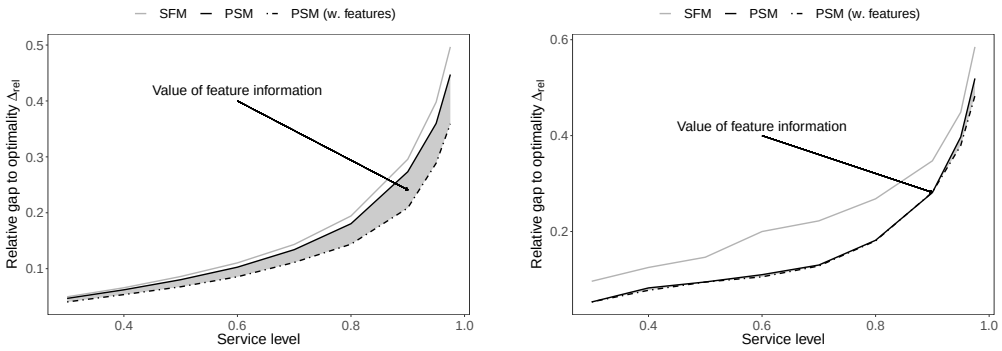


Figure 6.8: Boxplot of average cost differences between SFM and PSM for all slot/day-combinations for the large call center.

tinguishing property of the PSM method is that it allows to consider auxiliary data in the form of feature information that might be relevant for the staffing prescription. In this section, we evaluate the performance of the PSM with additional features derived from historical time series (e.g., call volume in the same slot at the previous day) and calendar information (e.g., holidays) and compare it to the performance of its counterpart without feature information as well as the SFM method.

Figure 6.9a provides an overview of the staffing performances for the large call center reported as relative gap to optimality Δ_{rel} depending on the service level configuration. We observe that the feature information significantly improves the staffing performance and even gains in importance with increasing service level up to a 8.8% lower gap to optimality than the PSM method without feature information for the 97.5% service level.

The performances for the small call center which are reported in Figure 6.9b however provide a different picture. In this setting, the included feature information does not add a large benefit to the staffing performance of the PSM method. We assume that the included features do not have predictive information for the call arrival rates in this setting which might be



(a) Large call center (b) Small call center

Figure 6.9: Value of feature information

explained by the fact that the call center is working in an entirely different domain (banking) than the larger call center (public services) and hence, call volume might be driven by different factors.

Still, while the PSM method cannot profit from the additional feature information, it remains robust, leading to similar prescription performance as the PSM method without feature information.

6.5 Conclusion and further research

In this paper we presented a prescriptive method to call center staffing. The proposed approach is entirely novel in that it prescribes optimal staffing levels by using an adapted machine learning algorithm that exploits the predictive information being available in the form of feature data, e.g., seasonal effects or national holidays. Our main contribution is the integration of the specific staffing objective into the machine learning algorithm. The application of this procedure to real-world problem instances is enabled by a novel preprocessing routine that efficiently calculates ex-post optimal decisions which then serve as an important input to the subsequent staffing model. The latter model then learns the functional relationship between these optimal decisions and the feature data and exploits it for future, unseen instances.

We find that our approach performs particularly well when uncertain arrival rates follow a nonrandom pattern (e.g., a trend) and when features with predictive information are available. Under such conditions, our prescriptive staffing method achieves up to 3% lower staffing costs than the optimized stochastic fluid model benchmark in the large call center and up to 8% lower costs in the small call center. More importantly, we are able to beat the benchmark in all examined scenarios, i.e., under differing service level assumptions both with and without auxiliary information. We conclude that by using the actual arrival patterns and not making parametric assumptions, our approach handles the uncertainty around the call arrival structure particularly well.

We leave it to further research work to extend our approach to settings with multiple customer classes as well as different server types. In these problem instances, routing calls from a specific customer class to an available server requires a considerably more complicated routing logic that we have ignored so far. Moreover, we see further research potential in exploring the benefits of utilizing more advanced underlying machine learning techniques. As an example, bagging multiple instances of the core regression trees of our PSM method would result in prescriptive random forests whose staffing decisions should be less prone to overfitting than regression trees.

7 Summary and Conclusion

This dissertation introduced and analyzed new ways for using data to make better decisions in operations management. It showed that, across a broad variety of industries, such as maintenance, repair and overhaul businesses, restaurants, call centers and public services, available data can be used to improve decision making. As discussed in Chapter 1, there is increasing awareness of the importance of data in decision-making, but the traditional approach, which uses data mainly for making predictions to support the subsequent decision, which typically involves manual adaptations based on intuition or gut feeling, stands in the way of achieving substantial improvements in productivity by automating the entire process using prescriptive models. Therefore, this thesis introduced and analyzed new prescriptive models that integrate estimation and optimization into a single step. Such JEO approaches were implemented and evaluated on real-world data sets for inventory management (Chapters 3 and 4) and capacity management problems (Chapters 5 and 6).

The first paper in the thesis (Chapter 2) sought ways to use data that is distributed between potentially competing players and that contains sensitive information, so it cannot be shared openly. We introduced and implemented a concept that allowed us to evaluate classification and regression trees on encrypted data. While the use case in our work was to forecast demand for maintenance and spare parts, with minor adaptations, the same concept could be used to evaluate tree-based JEO approaches and enable automated decision-making with distributed and sensitive data.

In Chapter 3 we analyzed the performance drivers of data-driven prescriptive inventory management in a Newsvendor setting with non-stationary demand. Our main conclusion is that, in a typical practical setting, the tree-

based approach provides more robust results than a linear approach because it provides lower-cost decisions for feature-demand relationships that are not predominantly linear, it provides better results in cases of heteroscedasticity, and it performs better in real-world settings with very small data sets because of its built-in feature selection, which is important in terms of usability in practice.

Chapter 4 analyzed the fundamental structural differences between SEO and JEO approaches. We used an inventory problem with non-stationary demand, where variations in demand are driven by observable features. We introduced a novel JEO approach based on random forests, a powerful and flexible ML technique, and compared its performance with that of its SEO counterpart, which uses a similar ML concept. An extensive simulation study revealed that, in cases of feature-dependent uncertainty, using JEO has significantly better results than using its SEO counterpart does. These results were backed up by similar analysis with kernel-based JEO and SEO approaches from the literature and by analytical results we obtained with linear models. However, the results from real-world data suggested that the difference in the two approaches' performance is only marginal and that SEO's performance is surprisingly robust. Hence, we concluded that the greater effort that is required for the implementation of problem-specific JEO approaches is justified only for high-impact decisions or in case of heteroscedasticity.

In Chapter 5, we considered a capacity-management problem in a public service office in Germany. We presented an innovative JEO approach to prescribe capacities that requires no assumptions about the underlying arrival process. We formalized service goals like "No more than 20% of the customers wait more than 20 minutes" and integrated them into am ML algorithm to learn a functional relationship between features and prescribed capacity from historical data. We analyzed our JEO approach's performance on a real-world dataset and compared it to an SEO approach that first uses out-of-the-box ML to predict arrival rates and then determines the capacity using queuing models. We found that the performance of both data-driven approaches is significantly better than that of a naive benchmark, but the JEO approach significantly outperformed the commonly used SEO benchmark. We

concluded that the JEO approach is especially useful if arrival rates are not stationary and the non-stationarity is feature dependent.

Chapter 6 introduced a JEO approach for staffing inbound call centers, where the main difference from the setting in Chapter 5 was that we assigned costs to waiting time and call abandonments. We integrated abandonment cost functions into a ML algorithm to learn a functional relationship between features and optimal capacity from historical data. An analysis of our JEO approach's performance on two real-world datasets compared to that of a state-of-the art data-driven benchmark revealed that our approach significantly outperforms the benchmark in both settings. We also found that our approach is especially useful in cases that have non-stationary arrival rates, which is in line with the results from Chapter 5.

From our results, we conclude that JEO approaches are promising ways to improve decision-making considerably. First, JEO approaches can significantly reduce costs compared to their traditional SEO counterparts when there is feature-dependent uncertainty (e.g., if demand predictions are more accurate for some days than for others). Second, JEO approaches are a perfect fit for automating decision-making since they prescribe actual decisions instead of making predictions like standard ML approaches do and that then require a second step that typically involves manual intervention. Hence, we see a large potential for increasing productivity by moving from predictive to prescriptive analytics with JEO approaches as part of automated decision-making processes.

The JEO approaches that were developed as part of this thesis will serve as foundation for a start-up that provides AI-based solutions for operations management. The various research projects conducted with companies in developing this dissertation suggest the enormous practical usefulness and potential for significant productivity gains in a variety of industries.

In addition to this obvious potential for practical applications, there are also many opportunities for further research. A downside of JEO approaches is that they are less flexible than SEO approaches, where the underlying ML technique that is used for making predictions can easily be exchanged. For JEO approaches, the ML algorithms have to be adapted to the specific prob-

lem class. Hence, implementing and analyzing JEO approaches for other problem classes such as pricing or multi-period inventory settings is an important field for future research.

A Appendix of Chapter 2

A.1 Aggregate classification results to demand distribution

We first provide the algorithm that returns the distribution and then illustrate the importance of having a measure for the forecast accuracy with an numeric example.

A.1.1 Algorithm

The following Algorithm 3 takes a vector with the individual replacement probabilities $\bar{\pi}$ and returns a distribution with probabilities for 0, 1, ... N replacements, where N is the number of instances in $\bar{\pi}$.

Then the initial distribution vector has length $N+1$ and is given by $\mathbf{p} = (1; 0; \ldots; 0)$. This vector already fulfills all necessary criteria of a probability distribution since:

- $0 \leq \mathbf{p}[i] \leq 1$ for $i = 1, \ldots, length(\mathbf{p})$
- and $\sum_{i}^{length(\mathbf{p})} \mathbf{p}[i] = 1$ hold.

For the initial distribution none of the classified instances is considered. Hence, the probability for 0 replacements is 1. We then add, iteratively, the individual replacement probabilities from $\bar{\pi}$ according to the following logic: If we add an additional component from $\bar{\pi}$, the probability of $j = 1, \ldots, N$ replacements is updated by adding the following two terms:

- the product of the probability of j replacements without the additional

```
Function distribution(π̄)
    N ← length(π̄)
    p ← vectorwithlength = N + 1            // initialize the
     distribution vector; all entries are 0

    p[1] ← 1                                // set first entry to 1

    for i in 2 : N + 1    // in each step one component is added
     do
        for j in i : 2          // backward iteration from i to 2
         do
            p[j] = p[j − 1] ∗ (1 − π̄[i − 1]) + p[j] ∗ π̄[i − 1]  // update
             the probabilities

        end
        p[1] = p[1] ∗ π̄[i − 1]    // this is the probability that
         all components have to be replaced
    end
    return p  // now p[i] contains the probability for N + 1 − i
    replacements
```

Algorithm 3: Iterative algorithm to compute a probability distribution given a vector with individual replacement probabilities

component and the probability that the additional component does not have to be replaced

- the product of the probability for $j-1$ replacements without the additional component and the probability that the additional one has to be replaced

Each step of the algorithm maintains the properties of a probability distribution.

A.1.2 Example

If the condition data changes, the reliability of the forecast can also change. Even if the point forecast remains the same, we now can measure this uncertainty, which gives us the opportunity to react accordingly. We illustrate

this with the following example. Let's assume the MRO is responsible for the overhaul of 100 engines currently in use by different customers and all equipped with the same kind of oil pump. The MRO is interested in the expected number of pumps he needs to replace within the next month (the lead time for this part). The classification results are given in Figure A.1. The 100 instances of data are classified using the tree given in Figure 2.3, assuming this tree was learned on historical data related to oil pump condition. Assume that, in Example 1, $\bar{\pi}_1 = 70$ instances belong to leaf 1 with a replacement probability of 0.1 and $\bar{\pi}_3 = 30$ to leaf 3 with probability 0.9.

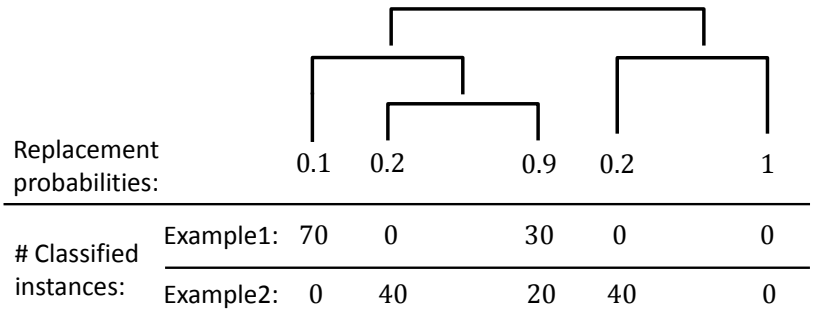


Figure A.1: Examples of classification results with 100 instances

The forecast is then just the mean:

$$F_{spares,Example1} = \sum_{l=1}^{L} \bar{\pi}_l \pi_l = 70*0.1+0*0.2+30*0.9+0*0.2+0*1 = 34 \quad \text{(A.1)}$$

In the classification results of Example 2, $\bar{\pi}_2 = 40$ instances belong to leaf 2, $\bar{\pi}_3 = 20$ to leaf 3 and $\bar{\pi}_4 = 40$ to leaf 4 with respective replacement probabilities of 0.2, 0.9 and 0.2 we obtain the same mean forecast:

$$F_{spares,Example2} = \sum_{l=1}^{L} \bar{\pi}_l \pi_l = 0*0.1+40*0.2+20*0.9+40*0.2+0*1 = 34 \quad \text{(A.2)}$$

Given only the mean, we have no information about the reliability of the forecasts. However, applying Algorithm 3 gives us the distributions depicted in Figure A.2.

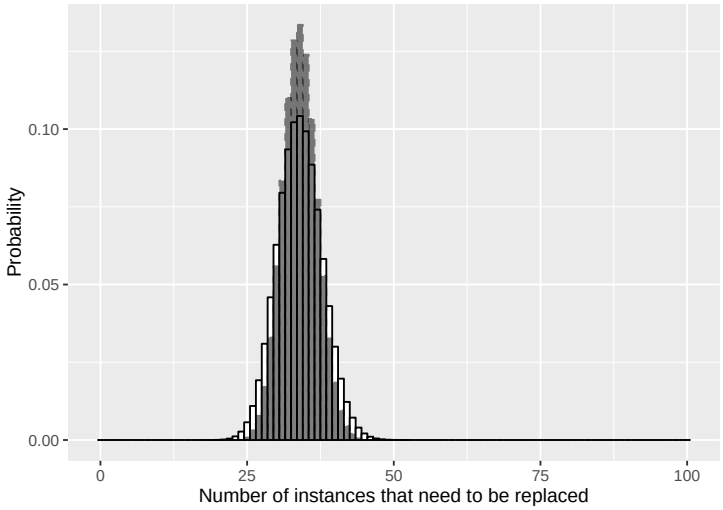


Figure A.2: Distributions resulting from the two different examples in Figure A.1; Example 1 in grey bars; Example 2 in white bars

We see that in Example 1 the distribution is more centered around the mean. Hence, although the forecast is the same, Example 2 comes with a higher uncertainty. This provides us with valuable information for subsequent inventory decisions.

B Appendix of Chapter 3

B.1 Expected costs for SAA with uniformly distributed demand

For a sample of demands with $d \sim \mathcal{U}[a, b]$ we determine the expected costs with respect to SL for SAA as follows. We note that we consider normalized underage-and overage costs, that is, $c_u + c_o = 1$ and therefore $SL = \frac{c_u}{c_u + c_o} = c_u$. From

$$E[C_{SAA}(SL)] = E[SL(d - q_{SAA})^+ + (1 - SL)(q_{SAA} - d)^+]$$

with $E[q_{SAA}] = a + SL(b - a)$ we get:

$$\begin{aligned} E[C_{SAA}(SL)] &= SL\frac{1}{2}(b - (a + SL(b - a))) + (1 - SL)\frac{1}{2}(a + SL(b - a) - a) \\ &= SL(b - a)(1 - SL) \\ \frac{\delta}{\delta SL} &= (b - a)(1 - 2SL) = \begin{cases} < 0 & \text{if } SL > 0.5 \\ > 0 & \text{if } SL < 0.5 \end{cases} \end{aligned}$$

Hence, the expected costs decrease with an increase of SL for $SL > 0.5$.

B.2 Combined effect of nonlinearity and heteroscedasticity

Figure B.1 shows the difference of mean costs of both models compared to SAA for different levels of nonlinearity ($p = 0, 0.4, 1, 2$) with respect to γ. We see that the behavior of TBR-NV does not change for different values of p. While the curve of LQR-NV is shifted towards SAA as p increases. However, the shape or slope of the curve of LQR-NV does not change hence the influence of nonlinearity and heteroscedasticity simply add up.

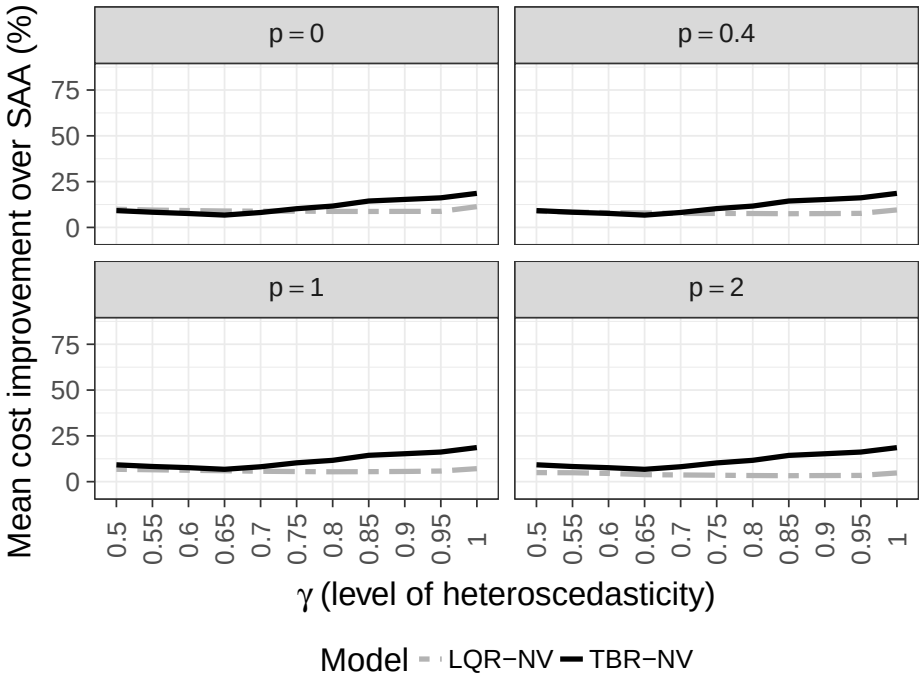


Figure B.1: Mean cost deltas depending on the level of heteroscedasticity for $cv_{noise} = 0.5, SL = 0.8$ and different levels of nonlinearity ($p = 0, 0.4, 1, 2$).

B.3 Features for Steak

The following tables B.1 and B.2 provide an overview of the derived features we originally fed into the LQR-NV and TBR-NV models.

Feature	Description
Weather_Di	Weather $\in$ {Wind, Sun, Rain, Cloudiness, Temperature} $i \in \{1, 2, 3\}$ days ago
Weather_M_Di	Average Weather $\in$ {Wind, Sun, Rain, Cloudiness, Temperature} over the last $i \in \{1, ..., 7\}$ days
Weather_H_7D	Number of days Weather $\in$ {Wind, Sun, Rain, Cloudiness, Temperature} was higher than 7day average
Weather_L_7D	Number of days Weather $\in$ {Wind, Sun, Rain, Cloudiness, Temperature} was lower than 7day average

Table B.1: List of features derived from weather data

Feature	Description
Steak_Di	Demand i days ago $i \in \{1, ..., 7\}$
Steak_C_Di	Demand cumulated over last i days $i \in \{1, ..., 7\}$
Steak_H_7D	Number of days demand was higher than 7day average
Steak_L_7D	Number of days demand was lower than 7day average
Steak_M_iW	Average aggregate demand for Steak at same weekday for the last i weeks $i \in \{1, 2, 3\}$
ProdCat_Di	Demand for products from $ProdCat \in$ {Meat, Fish, All} i days ago $i \in \{1, ..., 7\}$
ProdCat_C_Di	Demand for products from $ProdCat \in$ {Meat, Fish, All} cumulated over last i days $i \in \{1, ..., 7\}$
Steak_R_7D	Range of demand for Steak over last 7 days
ProdCat_R_7D	Range of demand for products from products from $ProdCat \in$ {Meat, Fish, All} over last 7 days
ProdCat_H_7D	Number of days demand for products from $ProdCat \in$ {Meat, Fish, All} was higher than 7day average
ProdCat_M_iW	Average aggregate demand for the products from $ProdCat \in$ {Meat, Fish, All} at same weekday for the last i weeks $i \in \{1, 2, 3\}$
is_Month	1 if observation is from Month, else 0
is_Weekend	1 if observation is from a weekend, else 0
is_DayOfWeek	1 if observation is from DayOfWeek, else 0
is_Year	1 if observation is from Month, else 0
Is_Outlier	Is special day (Event, Holiday, etc.)

Table B.2: List of features derived from time series for product Steak

C Appendix of Chapter 4

C.1 Proof of Proposition 4.1

Proof. From Koenker [2005] we obtain that the coefficient of the quantile regression $\hat{\beta}_{SL}$, converges for $n \to \infty$ to $\beta + \gamma F_u^{-1}(SL)$. This implies that the linear empirical risk minimization of $\hat{q}_{JEO-Lin}(x)$ provides consistent decisions and hence asysmptotically optimal costs.

For the same setting, we analyze $\hat{q}_{SEO-Lin}(x)$. We obtain

$$\begin{aligned}
\hat{\beta}_{LSE} &= \min_{\beta'} \sum_{i=1}^{n} (d_i - x_i\beta')^2 = \min_{\beta'} \sum_{i=1}^{n} (x_i\beta + x_i\gamma u_i - x_i\beta')^2 \\
&= \min_{\beta'} \sum_{i=1}^{n} x_i(\beta^2 + 2\beta\gamma u_i - 2\beta\beta' + (\gamma u_i)^2 - 2\beta'\gamma u_i + \beta'^2) \\
&= \min_{\beta'} \left(\sum_{i=1}^{n} x_i(\beta^2 - 2\beta\beta' + \beta'^2) + \sum_{i=1}^{n} x_i\gamma u_i(2\beta - 2\beta') + \sum_{i=1}^{n} x_i(\gamma u_i)^2 \right) \\
&= \min_{\beta'} \left(\sum_{i=1}^{n} x_i(\beta - \beta)^2 + \sum_{i=1}^{n} x_i\gamma u_i(2\beta - 2\beta') \right) \xrightarrow{n\to\infty} \beta
\end{aligned} \tag{C.1}$$

since $\sum_{i=1}^{n} x_i(\gamma u_i)^2$ is independent of β and $\sum_{i=1}^{n} x_i\gamma u_i(2\beta - 2\beta') \xrightarrow{n\to\infty} 0$ since X and u are independent and u has mean zero. Hence, the least squares estimate is not biased by heteroscedasticity.

$\hat{q}_{\varepsilon}(SL)$, i.e., the empirical quantile of the residuals does not consider the feature x and converges to some constant $const_{SL}$. Hence, the estimator in the SEO approach is still unbiased, the decision however, does not reflect the feature-dependent uncertainty, since $const_{SL}$ shifts the regression line similarly for all x.

Since the cost function is convex and JEO provides asymptotically optimal decisions, we obtain $\mathbb{E}\left[C(q_{JEO-Lin}(x), D)\right] \leq \mathbb{E}\left[C(q_{SEO-Lin}(x), D)\right]$. We do not have strictly lower costs for JEO due to special cases such as $x = const.$

□

C.2 Proof of Proposition 4.2

Proof. For $SL = 0.5$ we show that both approaches lead to the same expected decision. For the linear JEO approach we obtain $\mathbb{E}_{X\times D}\left[q_{JEO-Lin}(x)\right] = x(\beta + \gamma F_u^{-1}(0.5)) = x\beta$ since $F_u^{-1}(0.5) = 0$ because f_u is symmetrical with mean zero. For the linear SEO approach we have $\mathbb{E}_{X\times D}\left[q_{SEO-Lin}(x)\right] = x\beta + F_\varepsilon^{-1}(0.5)$ where the distribution of ε is given by the residuals of the least squares estimator:

$$\begin{aligned} \varepsilon_i &= x_i\beta + \gamma x_i u_i - x_i\hat{\beta} \\ &= x_i(\beta - \hat{\beta}) + \gamma x_i u_i \end{aligned} \tag{C.2}$$

Since the product distribution of Xu is still symmetric with mean zero and $(\beta - \hat{\beta}) \xrightarrow{n\to\infty} 0$ we obtain $F_\varepsilon^{-1}(0.5) = 0$ and hence $\mathbb{E}_{X\times D}\left[q_{JEO-Lin}(x)\right] = \mathbb{E}_{X\times D}\left[q_{SEO-Lin}(x)\right]$. Due to the piece-wise linear newsvendor cost function, similar expected decisions also imply similar expected costs.

For $SL > 0.5$, we first show that $\exists x_0 \in [0,1] : \mathbb{E}_{X\times D}\left[q_{JEO-Lin}(x_0)\right] = \mathbb{E}_{X\times D}\left[q_{SEO-Lin}(x_0)\right]$.

$$\begin{aligned} \mathbb{E}_{X\times D}\left[q_{JEO-Lin}(x_0)\right] &= \mathbb{E}_{X\times D}\left[q_{SEO-Lin}(x_0)\right] \\ \Leftrightarrow x_0\beta + F_\varepsilon^{-1}(SL) &= x_0\beta + x_0\gamma F_u^{-1}(SL) \\ \Leftrightarrow F_\varepsilon^{-1}(SL) &= x_0\gamma F_u^{-1}(SL) \\ \Leftrightarrow x_0 &= \frac{F_\varepsilon^{-1}(SL)}{\gamma F_u^{-1}(SL)} \end{aligned} \tag{C.3}$$

Hence, we need to show that $0 \leq \frac{F_\varepsilon^{-1}(SL)}{\gamma F_u^{-1}(SL)} \leq 1$. The left inequality we get since $F_\varepsilon^{-1}(SL) \geq 0$ and $F_u^{-1}(SL) \geq 0$ since $SL \geq 0.5$ and both distributions of u and ε are symmetric with mean zero.

For the right inequality we have:

$$\begin{aligned} F_{\varepsilon}^{-1}(SL) &\leq \gamma F_u^{-1}(SL) \\ \Leftrightarrow F_{\varepsilon}(q) &\geq \frac{1}{\gamma} F_u(q) \; \forall q > 0.5 \\ \Leftrightarrow P(\varepsilon \leq q) &\geq \frac{1}{\gamma} P(u \leq q) \\ \Leftrightarrow P(\gamma X u \leq q) &\geq \frac{1}{\gamma} P(u \leq q) \\ \Leftrightarrow P(z \leq q) := P(\gamma u \leq q) &\geq \frac{1}{\gamma} P(u \leq q) \\ \Leftrightarrow \int_{-\infty}^{q} f_{\gamma u}(z) dz &\geq \frac{1}{\gamma} \int_{-\infty}^{q} f_u(u) du \\ \Leftrightarrow \int_{-\infty}^{q} \frac{1}{|\gamma|} f_u(\frac{z}{\gamma}) dz &\geq \frac{1}{\gamma} \int_{-\infty}^{q} f_u(u) du \\ \Leftrightarrow \int_{-\infty}^{q} \frac{1}{|\gamma|} f_u(u) du &\geq \frac{1}{\gamma} \int_{-\infty}^{q} f_u(u) du \end{aligned} \tag{C.4}$$

where we use that $P(\gamma X u \leq q) \geq P(\gamma u \leq q)$ since $0 \leq X \leq 1$ and γ a scale parameter of f_u such that for $z := \gamma u$, we have $f_{\gamma u}(z) = \frac{1}{|\gamma|} f_u(\frac{z}{\gamma}) = \frac{1}{|\gamma|} f_u(u)$.

Since $\exists x_0 \in [0,1] : \mathbb{E}\left[q_{JEO-Lin}(x_0)\right] = \mathbb{E}_{X \times D}\left[q_{SEO-Lin}(x_0)\right]$, we have $\mathbb{E}_{X \times D}\left[q_{JEO-Lin}(x)\right] - \mathbb{E}_{X \times D}\left[q_{JEO-Lin}(x)\right] = (x - x_0)\gamma F_u^{-1}(SL)$ which is increasing in SL as $F_u^{-1}(SL)$ is increasing in SL. Since $C(.)$ is convex, we obtain that $\mathbb{E}_{X \times D}\left[C(q_{SEO-Lin}(X), D)\right] - \mathbb{E}_{X \times D}\left[C(q_{JEO-Lin}(X), D)\right]$ increases in SL. □

C.3 Mean versus median estimation

Given the demand model in (4.23), estimating the sample median instead of the mean provides better results in terms of costs when there is strong heteroscedasticity because then the observations with low noise are closely centered on the true mean. The sample median ignores the observations that have errors from the distributions with high variance, while the sample mean is similarly affected by all observations. We can show this reasoning more formally by comparing the variance of sample mean $\bar{\mathbf{d}}$ and sample median $\tilde{\mathbf{d}}$

in homoscedastic and heteroscedastic settings, respectively.

Proposition C.1. *For homoscedastic settings and a sample* $\mathbf{d}$ *from demand model (4.23) the following holds:*

$$Var(\tilde{\mathbf{d}}) > Var(\bar{\mathbf{d}}).$$

Proof. In the homoscedastic setting, we have $x \sim \mathcal{N}(0, \sigma^2_{base})$ with sample size n. Then we have:

$$Var(\bar{x}) = \frac{\sigma^2_{base}}{n} < \frac{\sigma^2_{base}\pi}{2n} \approx Var(\tilde{x}) \tag{C.5}$$

where we use that for n large, $Var(\tilde{x}) \approx \frac{1}{4nf^2(\theta)}$, with f the density function of x and θ the true median [Maritz and Jarrett, 1978]. □

Proposition C.2. *For heteroscedastic settings and a sample* $\mathbf{d}$ *from demand model (4.23) the following holds:*

$$Var(\tilde{\mathbf{d}}) > Var(\bar{\mathbf{d}}).$$

Proof.

$$Var(\bar{x}) = \frac{\sigma^2_{base}}{n} > \frac{\sigma^2_{Low}\pi}{2n_{Low}} \approx Var(\tilde{x}) \tag{C.6}$$

if $\sigma^2_{Low} < \frac{2n_{Low}}{\pi n}\sigma^2_{base}$, which holds in our setting in which σ^2_{Low} is as low as zero, and about half of the observations are drawn from a distribution with the lower variance. Hence, even for symmetric cost settings, JEO-RF can lead to more robust results in the case of strong heteroscedasticity. □

D Appendix of Chapter 6

D.1 Algorithm to approximate the cost function

Data: Ordered set of historical arrival times $\{f_1, \ldots, f_k\}$ in slot l, service time s, customer patience p, Number of servers b, end of planning slot T.
Result: Set of abandoning customers A
init Initialize parameters
    $S \longleftarrow \emptyset$ // set of customers currently served
    $Q \longleftarrow \emptyset$ // set of customers waiting for service
    $A \longleftarrow \emptyset$ // set of customers having abandoned
    $\tau = f_1$ // First event is arrival
end
begin
    while $\tau \leq T$ **do**
        if $\tau = f_i$ **then** next event is arrival
            if $S.length() < b$ **then** server available
                $c_i = \tau + s$ // calculate completion time
                $S.add(c_i)$ // add to server set
            end
            else must wait
                $a_i = \tau + v$ // calculate abandonment time
                $Q.add(a_i)$ // add to queue
            end
        end
        else if $\tau = c_i$ **then** next event is service completion
            $S.remove(c_i)$ // remove customer from server set
            if $Q.length() > 0$ **then** there is a queue
                $a_j = Q.first()$ // determine first customer in queue
                $Q.remove(a_j)$ // remove customer from queue
                $c_j = \tau + s$ // calculate new service completion time
                $S.add(c_j)$ // add next customer to server set
            end
            else do nothing
            end
        end
        else next event is abandonment
            $a_j = Q.first()$ // determine customer that abandons
            $Q.remove(a_j)$ // remove customer from queue
            $A.add(a_j)$ // save abandonment
        end
        $c_{i+1} = S.first()$ // Update c
        $a_{i+1} = Q.first()$ // Update a
        $\tau = \min\{f_{i+1}, c_{i+1}, a_{i+1}\}$ // Update τ
        $i = i + 1$ // Increase i and continue with next iteration
    end
    return A
end

Algorithm 4: Cost approximation

List of Figures

List of Tables

Bibliography

Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 563–574, New York, NY, USA, 2004. ACM.

Alp Akcay, Bahar Biller, and Sridhar Tayur. Improved inventory targets in the presence of limited historical demand data. *Manufacturing & Service Operations Management*, 13(3):297–309, 2011.

Zeynep Aksin, Mor Armony, and Vijay Mehrotra. The modern call center: A multi-disciplinary perspective on operations management research. *Production and operations management*, 16(6):665–688, 2007.

Mariz B. Arias and Sungwoo Bae. Electric vehicle charging demand forecasting model based on big data technologies. *Applied Energy*, 183:327–339, 2016.

Dimitrios Asteriou and Stephen G. Hall. *Applied Econometrics*. Palgrave Macmillan Ltd., 2011.

Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 492–499. IEEE Computer Society, 2010.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019.

Athanassios N. Avramidis, Alexandre Deslauriers, and Pierre L'Ecuyer. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7): 896–908, 2004.

M. Z. Babaï, A. A. Syntetos, Y. Dallery, and K. Nikolopoulos. Dynamic re-order point inventory control with lead-time uncertainty: Analysis and empirical investigation. *International Journal of Production Research*, 47 (9):2461–2483, 2009.

Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2018.

Sharad Barkataki and Hassan Zeineddine. On achieving secure collaboration in supply chains. *Information Systems Frontiers*, 17(3):691–705, 2015.

Mark Barratt. Understanding the meaning of collaboration in the supply chain. *Supply Chain Management: An International Journal*, 9(1):30–42, 2004.

Achal Bassamboo and Assaf Zeevi. On a data-driven method for staffing large call centers. *Operations Research*, 57(3):714–726, 2009.

Achal Bassamboo, J. Michael Harrison, and Assaf Zeevi. Design and Control of a Large Call Center: Asymptotic Analysis of an LP-Based Method. *Operations Research*, 54(3):419–435, 2006.

Achal Bassamboo, Ramandeep S. Randhawa, and Assaf Zeevi. Capacity sizing under parameter uncertainty: Safety staffing principles revisited. *Management Science*, 56(10):1668–1686, 2010.

Amos Beimel. Secret-sharing schemes: A survey. In *Proceedings of the Third International Conference on Coding and Cryptology*, IWCC'11, pages 11–46, Berlin, Heidelberg, 2011. Springer-Verlag.

Dimitris Bertsimas and Xuan Vinh Doan. Robust and data-driven approaches to call centers. *European Journal of Operational Research*, 207(2):1072–1085, 2010.

Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 2019.

Dimitris Bertsimas and Aurélie Thiele. A data-driven approach to newsvendor problems. *Working Paper, Massachusetts Institute of Technology*, 2005.

Anna Lena Beutel and Stefan Minner. Safety stock planning under causal demand forecasting. *International Journal of Production Economics*, 140 (2):637–645, 2012.

Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman and Jerome H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580, 1985.

Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984.

Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. Statistical analysis of a telephone call center. *Journal of the American Statistical Association*, 100(469):36–50, 2005.

Robert Goodell Brown. *Statistical forecasting for inventory control*. McGraw-Hill, 1959.

Erik Brynjolfsson and Andrew McAfee. *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. WW Norton & Company, 2014.

Emilio Carrizosa, Alba V Olivares-Nadal, and Pepa Ramírez-Cobo. Robust newsvendor problem with autoregressive demand. *Computers & Operations Research*, 68:123–133, 2016.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 96–103, 2008.

Hyunyoung Choi and Hal Varian. Predicting the present with google trends. *Economic Record*, 88:2–9, 2012.

Leon Yang Chu, J. George Shanthikumar, and Zuo-Jun Max Shen. Solving operational statistics via a Bayesian analysis. *Operations Research Letters*, 36(1):110–116, 2008.

William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829, 1979.

A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari. Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):136–153, 2016.

Ronald Cramer, Ivan Bjerre Damgrd, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, New York, NY, USA, 1st edition, 2015.

Ruomeng Cui, Santiago Gallino, Antonio Moreno, and Dennis J. Zhang. The operational value of social media information. *Production and Operations Management*, 27(10):1749–1769, 2018.

Haim Dahan, Shahar Cohen, Lior Rokach, and Oded Maimon. *Proactive data mining with decision trees*. SpringerBriefs in Electrical and Computer Engineering. Springer, New York, NY, 2014.

Rommert Dekker, Çerağ Pinçe, Rob Zuidwijk, and Muhammad Naiman Jalil. On the use of installed base information for spare parts logistics: A review of ideas and industry practice. *International Journal of Production Economics*, 143(2):536–545, 2013.

E. Deloux, B. Castanier, and C. Bérenguer. Predictive maintenance policy for a gradually deteriorating system subject to stress. *Reliability Engineering & System Safety*, 94(2):418–431, 2009.

Vinayak Deshpande, Ananth V. Iyer, and Richard Cho. Efficient supply chain management at the u.s. coast guard using part-age dependent supply replenishment policies. *Operations Research*, 54(6):1028–1040, 2006.

Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.

Alaa H. Elwany and Nagi Z. Gebraeel. Sensor-driven prognostic models for equipment replacement and spare parts inventory. *IIE Transactions*, 40(7): 629–639, 2008.

Qi Feng and J. George Shanthikumar. How research in production and operations management may evolve in the era of big data. *Production and Operations Management*, 27(9):1670–1684, 2018.

Organisation for Economic Co-operation and Development. *OECD Compendium of Productivity Indicators 2018*. OECD, 2018.

Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 145–156, London, UK, 2001. Springer-Verlag.

Noah Gans, Ger Koole, and Avishai Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141, 2003.

Noah Gans, Haipeng Shen, Yong-Pin Zhou, Nikolay Korolev, Alan McCord, and Herbert Ristock. Parametric forecasting and stochastic programming models for call-center workforce scheduling. *Manufacturing & Service Operations Management*, 17(4):571–588, 2015.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

A.A Ghobbar and C.H Friend. Sources of intermittent demand for aircraft spare parts within airline operations. *Journal of Air Transport Management*, 8(4):221–231, 2002.

Sharad Goel, Jake M Hofman, Sébastien Lahaie, David M Pennock, and Duncan J Watts. Predicting consumer behavior with web search. *Proceedings of the National academy of sciences*, 107(41):17486–17490, 2010.

Linda Green and Peter Kolesar. The pointwise stationary approximation for queues with nonstationary arrivals. *Management Science*, 37(1):84–97, 1991.

J. Michael Harrison and Assaf Zeevi. A method for staffing large call centers based on stochastic fluid models. *Manufacturing & Service Operations Management*, 7(1):20–36, 2005.

Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer series in statistics. Springer, New York, NY, 2nd edition, 2013.

R. H. Hayes. Statistical estimation problems in inventory control. *Management Science*, 15(11):686–701, 1969.

Bernd Hellingrath and Ann-Kristin Cordes. Conceptual approach for integrating condition monitoring information and spare parts forecasting methods. *Production and Manufacturing Research: An Open Access Journal*, 2(1): 725–737, 2014.

Tingliang Huang and Jan A. van Mieghem. Clickstream data and inventory management: Model and empirical analysis. *Production and Operations Management*, 23(3):333–347, 2014.

Jakob Huber, Sebastian Müller, Moritz Fleischmann, and Heiner Stuckenschmidt. A data-driven newsvendor problem: From data to decision. *European Journal of Operational Research*, 278(3):904 – 915, 2019.

Rouba Ibrahim and Pierre L'Ecuyer. Forecasting call center arrivals: Fixed-effects, mixed-effects, and bivariate models. *Manufacturing & Service Operations Management*, 15(1):72–85, 2013.

Rouba Ibrahim, Han Ye, Pierre L'Ecuyer, and Haipeng Shen. Modeling and forecasting call center arrivals: A literature survey and a case study. *International Journal of Forecasting*, 32(3):865–874, 2016.

Andrew K. S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510, 2006.

Marek Jawurek and Florian Kerschbaum. Fault-tolerant privacy-preserving statistics. In *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, pages 221–238, 2012.

Geurt Jongbloed and Ger Koole. Managing uncertainty in call centres using poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17 (4):307–318, 2001.

Florian Kerschbaum. Adapting privacy-preserving computation to the service provider model. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009*, pages 34–41, 2009.

Moutaz Khouja. The single-period (news-vendor) problem: Literature review and suggestions for future research. *Omega*, 27(5):537–553, 1999.

Song-Hee Kim and Ward Whitt. Are call center and hospital arrivals well modeled by nonhomogeneous poisson processes? *Manufacturing & Service Operations Management*, 16(3):464–480, 2014.

Diego Klabjan, David Simchi-Levi, and Miao Song. Robust stochastic lot-sizing by means of histograms. *Production and Operations Management*, 22 (3):691–710, 2013.

Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

Roger Koenker. *Quantile Regression*. Cambridge University Press, Cambridge, 2005.

Roger Koenker. quantreg: Quantile regression, 2016. URL `https://CRAN.R-project.org/package=quantreg`.

Julian Kurz. Capacity planning for a maintenance service provider with advanced information. *European Journal of Operational Research*, 251(2): 466–477, 2016.

Hau L Lee and Steven Nahmias. Single-product, single-location models. volume 4, pages 3–55. Elsevier, 1993.

S. Letourneau, F. Famili, and S. Matwin. Data mining to predict aircraft component replacement. *IEEE Intelligent Systems*, 14(6):59–66, 1999.

Retsef Levi, Georgia Perakis, and Joline Uichanco. The data-driven newsvendor problem: New bounds and insights. *Operations Research*, 63(6):1294–1306, 2015.

Siqiao Li, Qingchen Wang, and Ger Koole. Optimal contact center staffing and scheduling with machine learning. *Working Paper*, 2019.

Shuangqing Liao, Ger Koole, Christian van Delft, and Oualid Jouini. Staffing a call center with uncertain non-stationary arrival rate and flexibility. *OR Spectrum*, 34(3):691–721, 2012.

X. Lin, R.J.I. Basten, A.A Kranenburg, and G. J. van Houtum. Condition based spare parts supply. *Beta Working Paper series 371*, (371), 2012.

Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *Advances in cryptology-CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54, Berlin and New York, 2000. Springer.

Liwan H. Liyanage and J.George Shanthikumar. A practical inventory control policy using operational statistics. *Operations Research Letters*, 33(4):341–348, 2005.

Susan Lomax and Sunil Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys*, 45(2):1–35, 2013.

Mengshi Lu, J. George Shanthikumar, and Zuo-Jun Max Shen. Technical note - operational statistics: Properties and the risk-averse case. *Naval Research Logistics (NRL)*, 62(3):206–214, 2015.

Avishai Mandelbaum, Anat Sakov, and Sergey Zeltyn. Empirical Analysis of a Call Center. Technical report, Technion - Israel Institute of Technology, 2001. URL `http://iew3.technion.ac.il/serveng/References/references.html`.

J. S. Maritz and R. G. Jarrett. A note on estimating the variance of the sample median. *Journal of the American Statistical Association*, 73(361): 194–196, 1978.

Günter Matt. Adaptive Einflussgröenkombination (aek) - Prognosen mit schrittweiser Regression und adaptivem Gewichten. In Peter Mertens and Susanne Rässler, editors, *Prognoserechnung*, pages 125–168. Physica-Verlag, Heidelberg, 2005.

Nicolai Meinshausen. Quantile regression forests. *Journal of Machine learning Research*, 7:983–999, 2006.

Jan Meller, Fabian Taigel, and Richard Pibernik. Prescriptive analytics for inventory management: A comparison of new approaches. *Available at SSRN: https://ssrn.com/abstract=3229105*, 2018.

Peter Mertens, Andrew J. Zeller, and Jörn Große-Wilde. Kooperative Vorhersage in Unternehmensnetzwerken. In *Prognoserechnung*, pages 621–637. Springer, 2012.

Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis, Solutions Manual (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.

Kevin Patrick Murphy. *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.

E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Steven Nahmias. *Production and operations analysis*. The McGraw-Hill/Irwin series operations and decision sciences. McGraw-Hill/Irwin, Boston, Mass., 4th edition, 2001.

Afshin Oroojlooyjadid, Lawrence V. Snyder, and Martin Takác. Applying deep learning to the newsvendor problem. *CoRR*, abs/1607.02177, 2016.

Ying Peng, Ming Dong, and Ming Jian Zuo. Current status of machine prognostics in condition-based maintenance: A review. *The International Journal of Advanced Manufacturing Technology*, 50(1-4):297–313, 2010.

Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 463–477, Washington, DC, USA, 2013. IEEE Computer Society.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Vivek Ramamurthy, J. George Shanthikumar, and Zuo Jun Max Shen. Inventory policy with parametric demand: Operational statistics, linear correction, and regression. *Production and Operations Management*, 21(2): 291–308, 2012.

Lior Rokach and Oded Maimon. *Data mining with decision trees: Theory and applications*, volume 69 of *Series in machine perception and artificial intelligence*. World Scientific, Singapore, 2008.

Nicola Saccani. Forecasting for capacity management in call centres: Combining methods, organization, people and technology. *IMA Journal of Management Mathematics*, 24(2):189–207, 2013.

Herbert Scarf, K. J. Arrow, and S. Karlin. A min-max solution of an inventory problem. *Studies in the mathematical theory of inventory and production*, 10:201–209, 1958.

Bruce Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc, New York, NY, USA, 1995.

Erwan Scornet. Random forests and kernel methods. *IEEE Transactions on Information Theory*, 62(3):1485–1500, 2016.

Dirk Seifert. Konzepte des supply-chain-managements - cpfr als unternehmensübergreifende lösung. In Joachim Zentes, editor, *Handbuch Handel*, pages 781–794. Gabler, Wiesbaden, 2006.

Nikolai Stein, Jan Meller, and Christoph M Flath. Big data on the shop-floor: sensor-based decision-support for manual processes. *Journal of Business Economics*, 88(5):593–616, 2018.

Fabian Taigel, Anselme K. Tueno, and Richard Pibernik. Privacy-preserving condition-based forecasting using machine learning. *Journal of Business Economics*, 88(5):563–592, 2018.

Fabian Taigel, Jan Meller, and Alexander Rothkopf. Data-driven capacity management with machine learning: A novel approach and a case-study for a public service office. In H. Yang and R. Qiu, editors, *Advances in Service Science: Proceedings of the 2018 INFORMS International Conference on Service Science*. 2019.

James W. Taylor. Density forecasting of intraday call center arrivals using models based on exponential smoothing. *Management Science*, 58(3):534–549, 2012.

Theja Tulabandhula and Cynthia Rudin. Machine learning with operational costs. *Journal of Machine Learning Research*, 14:1989–2028, 2013.

E. van Wingerden, R.J.I. Basten, R. Dekker, and W. D. Rustenburg. More grip on inventory control through improved forecasting: A comparative study at three companies. *International Journal of Production Economics*, 157:220–237, 2014.

Albert W. Veenstra, Rob Zuidwijk, and Bart Geerling. Maintenance logistics in the dutch dredging industry. In *Service Operations and Logistics, and Informatics, 2006. SOLI'06. IEEE International Conference on*, pages 436–441, 2006.

S. Viswanathan, Handik Widiarta, and Rajesh Piplani. Value of information exchange and synchronization in a multi-tier supply chain. *International Journal of Production Research*, 45(21):5057–5074, 2007.

Geoffrey S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372, 1964.

Marvin N. Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.

David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *PoPETs*, 2016(4):335–355, 2016.

R. C.M. Yam, P. W. Tse, L. Li, and P. Tu. Intelligent predictive decision support system for condition-based maintenance. *The International Journal of Advanced Manufacturing Technology*, 17(5):383–391, 2001.

Yang Yang, Bing Pan, and Haiyan Song. Predicting hotel demand using destination marketing organization's web traffic data. *Journal of Travel Research*, 53(4):433–447, 2014.

Alice Zheng. *Mastering Feature Engineering: Principles and Techniques for Data Scientists*. Oreilly & Associates Inc, 2017.

Antonio Zilli, Richard Pibernik, Julian Kurz, and Fabian et al. Taigel. D24.2 - business modeling: Practice technical report, 2015.

Paul H. Zipkin. *Foundations of inventory management*. McGraw-Hill, Boston, MA, 2000.

Eidesstattliche Erklärung (Statement of Academic Integrity)

Hiermit erkläre ich gemäß § 6 Abs. 2 Nr. 2 der Promotionsordnung der wirtschaftswissenschaftlichen Fakultät der Universität Würzburg, dass ich diese Dissertation eigenständig, d.h. insbesondere selbständig und ohne Hilfe eines kommerziellen Promotionsberaters angefertigt habe. Ausgenommen davon sind jene Abschnitte, bei deren Erstellung ein Koautor mitgewirkt hat. Diese Abschnitte sind entsprechend gekennzeichnet und die Namen der Koautoren sind vollständig und wahrheitsgemäß aufgeführt. Bei der Erstellung der Abschnitte, bei denen ein Koautor mitgewirkt hat, habe ich einen signifikanten Beitrag geleistet, der meine eigene Koautorschaft rechtfertigt.

Außerdem erkläre ich, dass ich außer den im Schrifttumsverzeichnis angegebenen Hilfsmitteln keine weiteren benutzt habe und alle Stellen, die aus dem Schrifttum ganz oder annähernd entnommen sind, als solche kenntlich gemacht und einzeln nach ihrer Herkunft nachgewiesen habe.

Würzburg, den 29. Oktober 2019

Fabian Michael Taigel

Lebenslauf

Persönliche Daten

geboren	20. September 1987 in Tübingen

Ausbildung

April 2015	Master Wirtschaftsmathematik an der Universität Würzburg
April 2013	Bachelor Wirtschaftsmathematik an der Universität Würzburg

Praktische Tätigkeit

April 2015 – August 2019	Wissenschaftlicher Mitarbeiter am Lehrstuhl Logistik und Quantitative Methoden der Universität Würzburg

Ausgewählte Veröffentlichungen

2018	Taigel, F., Tueno, A. K. & Pibernik, R. Privacy-preserving condition-based forecasting using machine learning. Journal of Business Economics
2019	Taigel, F., Meller, J., & Rothkopf, A. Data-driven capacity management with machine learning: A novel approach and a case-study for a public service office. In Advances in Service Science: Proceedings of the 2018 INFORMS International Conference on Service Science.