Predictive and Prescriptive Analytics in Operations Management

by

Omar Skali Lami

B.S. and M.S., École Centrale Paris (2017)
M.BAn., Massachusetts Institute of Technology (2017)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Sloan School of Management
April 05, 2022

Certified by .....
Georgia Perakis
William F. Pounds Professor of Management Science
Co-director, Operations Research Center
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

Predictive and Prescriptive Analytics in Operations Management

by

Omar Skali Lami

Submitted to the Sloan School of Management
on April 05, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

The recent surge in data availability and advances in hardware and software and the recent developments and democratization of analytics highlight the critical importance of prediction and prescription in harnessing the power of data to create value through optimal, data-driven decision making. This thesis proposes novel Machine Learning (ML) and optimization methods in (i) predictive analytics, (ii) prescriptive analytics, and (iii) their high-impact applications in operations management.

On the predictive side, this thesis tackles the problems of interpretability and predictive power within the context of tree ensembles. The first chapter introduces the Extended Sampled Trees (XSTrees) method, a novel tree ensemble ML method for classification and regression. Instead of learning a single decision tree like CART, or an collection of trees like Random Forests or Gradient Boosting methods, XSTrees learns the entire probability distribution over the tree space. This approach results in good theoretical guarantees and has a significant edge over other ensemble methods in terms of performance. Analytically, we prove that XSTrees converges to the true underlying tree model with rate $\mathcal{O}\left(\frac{\log(n)}{n}\right)$, where $n \in \mathbb{N}$ is the number of training observations. Experimentally, we show on publicly available datasets, synthetic data, and two real-world case studies that XSTrees is very competitive with the state-of-the-art models, with an average accuracy between 2.5% and 50% higher than competitors for classification and an average R^2 between 2% and 85% higher for regression.

We further highlight the need and impact of more powerful and interpretable tree-based methods in the second chapter through the problem of ancillary services in targeted advertising under an ML lens. This chapter aims to predict the Net Present Value (NPV) of these services, estimate the probability of a customer subscribing to each of them depending on what services are offered to them, and ultimately prescribe the optimal personalized service recommendation that maximizes the expected long-term revenue. First, we propose a novel method called Cluster-While-Classify (CWC). This hybrid optimization-ML method performs joint clustering and classification and subsequently fits a tree-based classifier on the corresponding assignment to predict the

sign-up propensity of services based on customer, product, and session-level features. CWC is competitive with the industry state-of-the-art and can be represented in a simple decision tree, making it interpretable and easily actionable. We then use Double Machine Learning (DML) and Causal Forests, another tree-based ML method, to estimate the NPV for each service and finally propose an iterative optimization strategy — that is scalable and efficient — to solve the personalized ancillary service recommendation problem. CWC achieved a competitive 74% out-of-sample accuracy which, alongside the rest of the personalized holistic optimization framework, resulted in an estimated 2.5-3.5% uplift in revenue, which in turn translates to $80-100 million increase in revenue and $15-20 million increase in profits.

On the prescriptive side, this thesis moves away from the predict-then-optimize paradigm by doing the prediction and the prescription jointly, resulting in a lower prescription error and higher robustness. The third chapter presents a holistic framework for prescriptive analytics. Given side data $\boldsymbol{x}$, decisions $\boldsymbol{z}$, and uncertain quantities y, that are functions of $\boldsymbol{x}$ and $\boldsymbol{z}$, we propose a framework that simultaneously predicts y and prescribes the "should be" optimal decisions $\bar{\boldsymbol{z}}$. The algorithm can accommodate a large number of predictive machine learning models and continuous and discrete decisions of high cardinality. It also allows for constraints on these decision variables. We show wide applicability and strong computational performances on synthetic experiments and two real-world case studies.

Additionally, we illustrate the impact of these predictive and prescriptive analytics methods in two additional real-world, high-impact applications: healthcare and industrial operations. The fourth chapter proposes an end-to-end framework to help mitigate the COVID-19 pandemic and its impact through the case and death prediction, true prevalence, and fair vaccine distribution. We present the methods we developed for predicting cases and deaths using a novel ML-based aggregation method to create a single prediction we call `MIT-Cassandra`. We further incorporate COVID-19 case prediction to determine the true prevalence and incorporate this prevalence into an optimization model for efficiently and fairly managing the operations of vaccine allocation. This also allows us to provide insights into how prevalence and exposure of the disease in different parts of the population can affect vaccine distribution.

In the last chapter, we propose a novel, machine learning-based methodology to improve the efficiency of maintenance operations, from description to prediction to intervention. The proposed methodology has three main components, applied sequentially to the maintenance scheduling problem. First, a data-driven failure modes and effects analysis to fully describe the state of equipment at a given time in a data-driven way, including probabilities of each failure mode and its respective causes. Second, a unified predictive model which slightly adjusts its parameters for each specific piece of equipment to predict the state of some equipment in the future. Third, a holistic prescriptive model to optimize maintenance interventions.

Thesis Supervisor: Georgia Perakis
Title: William F. Pounds Professor of Management Science
Co-director, Operations Research Center

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Georgia Perakis, for her guidance throughout all these years and for being always there for me, especially when I needed her the most. I am forever grateful for her help, but most importantly, her mentorship and friendship. Beyond her incredible technical expertise, I am thankful for all the time she gives her students, her eternal positivity and optimism, and her contagious good mood, not to mention her loyalty, sincerity, and love. My experience at MIT and beyond would not have been the same without her.

Then I would like to thank Prof. Dimitris Bertsimas, who first reached out to me six years ago in 2016, offering the opportunity to join the MBAn program and MIT. Without him, I would probably not be where I am today. His life philosophy on merit, integrity, having high aspirations for oneself, being master of one's destiny, and surrounding oneself with exceptional people to positively impact the world has inspired me. In addition, his passion for research, teaching, and operations research is unparalleled and pushed me to delve further into the field that is today a central component of my career and my life.

After that, I would like to thank my thesis committee members, Prof. Retsef Levi and Prof. Stephen Graves, for their support and help throughout this journey. For example, Retsef taught me my final class in Operations Management, and his teachings aroused my curiosity for the field a notch further, which impacted my research directions. In addition, they believed in me, and they were always here for guidance, advice, and recommendations to help me and push me for the better.

I would also like to thank Prof. Alexandre Jacquillat for his friendship and advice, Prof. Vivek Faris, Prof. Rahul Mazumder, Prof. Nikos Trichakis, and Prof. Thodoris Lykouris, and Prof. Negin Golrezaei for investing their time to help me with my research and my various endeavors.

In terms of collaborators, I would like to extend my special thanks to Prof. Divya Singhvi from New York University, whom I have been working with since Day 1, who helped me academically, personally, and professionally. I could not have hoped for a

better co-author for all these years. Also, I want to thank Prof. Dessi Pachamanova from Babson College, who was essential to many of my research papers and was a delight to work with. I would also like to mention Asterios Tsiourvas, Leann Thayaparan, Ioannis Spantidakis, Angela Lin, Amine Bennouna, Michael Li, and Vassilis Digalakis from the Operations Research Center, Joshua Joseph from MIT Quest for Intelligence, Abdennour Jbili, Omar Amrani, and Ayman Bentourki from OCP, and Ankit Mangal and Stefan Poninghaus from Wayfair. But also, Hamza Tazi, David Nze Ndong, Jonah Adler, Shane Weisberg, Maureen Canellas, Allison Borenstein, Jiong Wei Lua, and many others. I would also like to add Laura Rose, Michelle Li, and Giada Tridello, the backbones of the ORC and the MBAn program.

I want to express my immense gratitude and love for my family: My mother, Fatima Zahra Naciri Bennani, and my father, Khalid Skali Lami, for their unconditional love and support, not only during my Ph.D. but during my entire life; My sister Kenza Skali Lami and my brother-in-law Lotfi Belmahi, who were always there for me, as well as my nephews Mamoun and Taoufiq Belmahi and the rest of my family; And, of course, my partner Sophia Sayah whose love and support were invaluable in many ways.

I am also very thankful for all my friends at MIT: El Ghali Zerhouni and Driss Lahlou Kitane, who helped me improve myself as a person and brought me a lot of laughter and good times; As well as Ivan Paskov, Amal Rar, and Youssef Berrada; But also many others: Adil Benkiran, Mouad Harbaz, Mehdi Berrada Mnimene, Hamza Zerhouni, Rim Harris, Kenza Mhaimdat, Zakaria El Hajjouji, Cheikh Fall, Léonard Boussioux, Luca Mingardi, Alessandro Previero, Francois Caprasse, Louis Lécluse, Tomas Littrel, Tamar Cohen-Hillel, the MBAn Class of 2017, especially Eric Green, Afshin Amidi, Matthieu Hummeau, Luc Missoum, Derek Chu, and Lorena Dominguez; and the ORC community as a whole – the generosity and collegiality of its people makes the ORC what it is.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A key objective of Operations Research (OR) and Operations Management (OM) is to go from data to decisions. This process often involves a predictive component and a prescriptive component, both critical in harnessing the value of data. The traditional approach in OR and OM is to build models from which we can derive decisions. By and large, data in these models has only played a supporting role. In contrast, data in Machine Learning (ML) models has played a protagonistic role.

Recent years however have seen a tremendous increase in data availability due to the democratization of analytics and the decline in the cost of sensors and microcontroller units. Advances in OR, ML, and computing power have the ability of turning these large amounts of data into real-world value. For example, [124] shows how predictive and prescriptive analytics could impact the top-line (optimizing external growth, e.g., pricing, churn prevention, and promotion optimization), the bottom-line (optimizing internal processes, e.g., predictive maintenance, supply chain optimization, and fraud prevention), as well as developing new business models. [1] shows how companies like Netflix realized more than $1B annual revenue by deploying individualized content. At the same time, healthcare organizations like Kaiser Permanente was able to decrease antibiotic use from its patients by 50% through data-driven health services. In OM, [150] shows the impact and increasing use of predictive and prescriptive techniques in various revenue management problems, including pricing, assortment optimization, and inventory management.

Arguably, tree methods such as CART ([39]) and Optimal Trees ([22]) are among the most prominent methods in predictive analytics, mainly due to their interpretability and scalability. Nevertheless, these methods struggle to compete in terms of accuracy with other state-of-the-art methods. On the other hand, ensemble methods such as Random Forests ([40]) and Gradient Boosting Trees ([49]) have very good predictive power but lack in terms of probabilistic guarantees and interpretability. One of the goals of this thesis on the predictive side is to bridge this gap and propose tree-based ML methods such as Extended Sampled Trees (Chapter 2) and Cluster-While-Classify (Chapter 3) that (i) have strong theoretical guarantees, (ii) a significant edge over other state-of-the-art ensemble methods in terms of performance, and (iii) intepretability, all equally critical in high-stakes applications.

On the prescriptive side, the standard paradigm in real-world OR problems involving prediction and prescription is predict-then-optimize, where ML tools are used to predict an uncertain point estimate of an uncertain quantity, that is then plugged in a nominal optimization problem to solve for the optimal decisions. This thesis tries to move away from this paradigm by doing the prediction and the prescription jointly (Chapters 3, 4, and 6) or by reducing the uncertainty in the predictions through ensemble learning (Chapters 2 and 5).

Last but not least, this thesis demonstrates how these novel predictive and prescriptive methods spanning ML, OR, and OM, can be applied in high-stakes applications in healthcare and healthcare operations, pricing and revenue management, and preventive maintenance. Consequently, we propose in this thesis additional insights on how to bridge the gap between theory and application through a focus on interpretability (Chapters 2, 3, 4), causality (Chapter 3, through the use of Double Machine Learning and Causal Forests), and subject-matter expertise in the fields of applications leading to considerable, measurable impact (Chapters 3, 5, and 6).

1.1 Contributions

Our main contributions in this thesis are as follows:

1. **Novel Tree-based ML Methods that are Interpretable and Powerful:** We propose new general methods for prediction. In Chapter 2, we introduce the **XSTrees**, or Extended Sampled Trees, a novel tree-based ensemble ML method for classification and regression. Instead of learning a single single decision tree such as CART or a finite collection of trees such as Random Forests or Boosting Trees, XSTrees learn the entire distribution over a carefully defined tree space. We also introduce in Chapter 3 the **CWC**, or a hybrid optimization-ML method that performs joint clustering and classification and subsequently fits a tree classifier on the corresponding assignment.

2. **Novel Robust Method for Joint Prediction and Prescription:** We also propose new general methods for prescription and data-driven decision-making. In Chapter 4, we introduce the **HPA**, or Holistic Prescriptive Analytics, a point-predictive-prescriptive method that jointly regroups observations into clusters with similar behaviors, learns a predictive model over each of these clusters and prescribes the optimal decisions under constraints.

3. **Adapt and Combine Analytics Methods for new OM Problems:** We expand ML and Optimization methods to be able to solve OM problems with novel approaches. In Chapter 3, we propose a framework that combines **Double Machine Learning** with **Causal Forest** to evaluate the Net Present Value of services in online retail, and we propose a curated optimization approach, the **ASDO**, or Ancillary Service Display Optimization for assortment optimization in this context. In Chapter 5, we adapt and combine four different methods, including a reinforcement learning method: the **MRL** for Minimal Representation Learning and an epidemiological method: the **C-SEIRD** for Chained SEIRD, two methods we have developed in different papers and that we have adapted to the COVID-19 data and problem. We also proposed an ensemble method for combining these predictions which we adapted to our methods and to the time-series context, as well as a totally-unimodular formulation of the fair vaccine allocation problem. Finally, in Chapter 6, we generalize our own **SSV**

method for Sparse and Slowly Varying Regression to the Sensor Prediction and Preventive Maintenance problems.

4. **Provide a Rigorous Framework for our Methods through strong Theoretical Guarantees:** In order to give a rigorous framework for our methods and applications, we prove guarantees on the methods mentioned above. In Chapter 2, we prove asymptotic and finite-sample bounds on the XSTrees methods. In the case of finite samples, we prove a finite-sample bound on the average distance between the position of the splits of the true underlying tree model and the learned XSTrees and show that this distance converges to 0 with rate $\mathcal{O}(\frac{\log(n)}{n})$, when $n \to \infty$, where n is the number of training observations. In Chapter 3, we prove the convergence of both the CWC and the ASDO methods. In Chapter 4, we prove the convergence of the HPA framework and evaluate the complexity and scalability of the method under different properties of the predictive and prescriptive costs. In Chapter 5, we prove robustness and variance bounds on the aggregation method, as well as total-unimodularity of the vaccine optimization problem.

5. **Prove Wide Applicability of the Methods proposed through Extensive Computational Experiments:** We perform extensive numerical experiments on (i) synthetic data, (ii) publicly-available datasets, (iii) real-world case studies across all chapters. This includes 30+ datasets from University of California Irvine (UCI) Machine Learning repository, 100+ synthetic experiments, and case studies on real-world data including healthcare data, revenue management data, and industrial sensor data. We shows that our methods proposed above are at least competitive with the state-of-the-art with improvements in accuracy going from 2% to 85% on average.

6. **Highlight the Real-World Impact of the Methods:** Through our collaboration with a leading online retailer on revenue management, described in Chapter 2 and 3, we have deployed an end-to-end framework for targeted advertising and assortment optimization, which generated $6M in the A/B testing

phase, with an estimated impact of $80-$100M in incremental revenue annually for our collaborator. In Chapter 5, we highlight the outcome of our collaboration in the MIT COVID-19 Response System (MCRS) with MIT Quest for Intelligence. Our methods were key in understanding the appropriate degree of returning to campus and re-opening the institute. They were also used on the CDC website (under the name of `MIT-Cassandra`) to help the CDC and government entities understand and mitigate the spread of the pandemic. Finally, in Chapter 6 on preventive maintenance and industrial operations, we show how our framework was deployed by OCP and generated 2.5M MAD (∼$250,000) in the testing phase, with 200M MAD (∼$20M) in expected profit after full-scale deployment.

In what follows, I will provide an outline of what each chapter contains.

1.2 Predictive Analytics: from Data to Predictions

1.2.1 XSTrees: Extended Sampled Tree Ensembles for Classification and Regression

In this chapter, we propose a new tree-based ensemble method for classification and regression we refer to as Extended Samples Trees, or XSTrees. Instead of learning a single decision tree such as CART or a finite collection of trees such as Random Forests or Boosting Trees, XSTrees learn the entire distribution over a carefully defined tree space. A key goal for XSTrees has been to be interpretable and with good theoretical guarantees, both asymptotic and for finite samples. XSTrees has a significant edge over other state-of-the-art machine learning methods in terms of performance (outperforming them by more than 2% on average). All three of these aspects have proven to be critical in important revenue management and healthcare applications. For example, we have used XSTrees in choice modeling and sales forecasting with an online retailer, as well as in terms of predicting hospital short-term "demand" (length of stay of a patient) for UMass Memorial Hospital.

1.2.2 Application to Revenue Management: Ancillary Services in Targeted Advertising

One of our most successful collaboration while developing XSTrees was with the leading online retailer for home furniture and décor. This chapter introduces an end-to-end framework for targeted advertising and assortment optimization, which generated \$6M in the A/B testing phase, with an estimated impact of \$80-\$100M in incremental revenue annually for our collaborator. The framework has three main components: (i) a novel Cluster-While-Classify (CWC) method to estimate the probability of service sign-up given customer and product features and a service assortment, (ii) a causal inference method consisting of Double Machine Learning (DML) and Causal Forests to estimate the incremental NPV, or Net Present Value, which captures the short- and long-term increase in revenue due to subscribing to one or more ancillary service, and (iii) a scalable, provably-optimal optimization formulation for optimal ancillary service display in a personalized fashion.

1.3 Prescriptive Analytics: from Predictions to Decisions

1.3.1 Holistic Prescriptive Analytics for Continuous and Constrained Optimization Problems

One of the many goals of Operations as a field is to go from data to operational decisions that create "value". Consequently, while the predictive component is important, there is fundamental need for prescriptive analytics as well. This chapter proposes a point-predictive-prescriptive framework called HPA, for Holistic Prescriptive Analytics, that jointly regroups observations into clusters with similar behaviors, learns a predictive model over each of these clusters and prescribes the optimal decisions under constraints. The intuition behind this framework is that clustering allows us to achieve higher accuracy by aggregating data into clusters and to divide the training

process into smaller subproblems, while performing all the tasks (clustering, prediction, and prescription) jointly allows us to find the right trade-off between accurate predictions and optimal decisions, in a tractable way that can account for continuous and constrained decision spaces. The HPA framework also allows for interpretable decision making, while maintaining a high-level of performance, as illustrated in a diabetes treatment case study, where it improves over other methods by up to 14% in terms of treatment efficacy.

1.3.2 Application to Healthcare with MIT Quest for Intelligence: COVID-19 Predictions, Prevalence, and the Operations of Vaccine Allocation

Given the unique circumstances in the last couple of years on a global scale, a big area of application was also on combating the COVID-19 pandemic and its effects. I have been part of two research groups. First, the covidanalytics.io group, where I helped develop an epidemiological model called DELPHI. In addition, in the MIT-Cassandra group, I led the efforts in (i) creating a high-performing ensemble model to predict the COVID-19 cases and deaths throughout the country, with different levels of granularity ranging from country to state, county, to even ZIP code, but also (ii) creating a probabilistic prevalence model to estimate not only the detected cases and deaths, but also the true underlying cases and deaths beyond the limited testing capabilities. Our predictions were used by the CDC, and were out-performing other methods, being consistently in the top 10, and very often ranked 1st. MIT-Cassandra was also ranked 1st on average for the short-term 1-week ahead predictions. Finally, (iii) we have also developed an optimization framework, with interesting insights for optimal vaccine allocation and roll-out, which we have applied to the state of Massachusetts. Our predictions were deployed jointly and used by MIT Quest for Intelligence in the context of safely and promptly re-opening the institute, which was in hybrid format in Spring 2021, and has fully re-opened since Summer 2021. We discuss the latter in this chapter.

1.3.3 Application to Industrial Operations with OCP: A Machine Learning Approach to Preventive Maintenance

Last but not least, in collaboration with OCP, the largest the largest phosphates mining company in the world, based in Morocco, this chapter proposes a preventive maintenance pipeline, which consists of a descriptive, a predictive, and a prescriptive component. The descriptive component builds a dynamic Failure Mode & Effects Analysis tree based on sensor data and text data only. The predictive component predicts the evolution of failures and of sensor data with the Slowly Varying Sparse Regression method we have developed, and the prescriptive component schedules maintenance interventions based on the output of the two previous components. This framework has already been partly deployed and generated 2.5M MAD (∼$250,000) in the testing phase, with 200M MAD (∼$20M) in expected profit after full-scale deployment.

Chapter 2

Extended Sampled Tree Ensembles for Classification and Regression

2.1 Introduction

Machine Learning (ML) and Artificial Intelligence (AI) have transformed decision making in diverse fields such as retail, healthcare, military, and policy among others (see for e.g. [41], [13] and [28]). The rise of big data has led to tremendous growth in ways in which data can inform decisions. One of the main focuses of ML today is predictive analytics: given features x, we want to predict some target variable y, which can be categorical (in which case, we talk about classification, e.g., choice modeling) or continuous (in which case, we talk about regression, e.g., demand prediction), as accurately as possible with models trained on historical data. Predictive analytics is at the very core of revenue management and is the key in driving improved decisions. For example, y can represent a customer's purchase decision when presented an assortment of products. Or, in another application, y can represent the overall units sold of a particular product (with features x, that includes price p). In each case, predictive analytics can be used to predict how y relates to the underlying decision, and in turn optimize it. Recent studies include [50] that develop a decision forest model that can predict more general consumer choice behavior. [48] focus on Binary Choice Forests for estimating discrete choices. Similarly, [5] construct a tree based mar-

ket segmentation model for personalized response prediction. Since these predictive models directly aid in decision making, there is a need for high-performing predictive models, with provable guarantees, as well as simple ways to interpret the outcomes from these models. Arguably, tree methods such as CART ([39]) and Optimal Trees ([22]) are among the most prominent methods in predictive analytics, mainly due to their interpretability and scalability. Nevertheless, these methods struggle to compete in terms of accuracy with other state-of-the-art methods. On the other hand, ensemble methods such as Random Forests ([40]) and Gradient Boosting Trees ([49]) have very good predictive power but lack in terms of probabilistic guarantees and interpretability.

In this chapter, we propose a new framework that has intuition similar to both of these families of methods. While tree-based models have a simple decision-rule structure that helps in interpretability, ensemble methods leverage the power of a family of tree models to improve accuracy. The proposed **Extended Sampled Trees (XSTrees)** framework considers a family of such decision trees but instead of aggregating predictions like in the Random Forest model, our model estimates a distribution of trees from this family.

Our proposed approach results in good theoretical guarantees, but most importantly, in a significant edge over other state-of-the-art ML methods in terms of predictive performances. To gain more intuition, one can view the learning of the appropriate tree for a regression or a classification as an estimation task. CART would then be a local-minimum point-estimator for this estimation task, Optimal Trees would be the "best" point-estimator, Random Forests and Boosting Trees would be a collection of estimators, while XSTrees would, in contrast, learn the entire distribution over these estimators. To accomplish this, we introduce the notions of *tree extensions* and *tree distributions* but also present in detail the XSTrees approach to learn these distributions and generate predictions from them using Monte-Carlo ([115]) sampling. We then prove asymptotic and finite-sample guarantees on the XSTrees algorithm and benchmark it against state-of-the-art methods on publicly available datasets, synthetic data, and two real-world case studies — the first one in a revenue management

setting with our industry collaborator, a leading e-commerce retailer that sells home goods, and the second in a healthcare setting. Finally, we illustrate how to explain and extract intepretable insights from the XSTrees predictions with a visualization tool and through several examples.

Contributions

The main contributions of this chapter are the following:

- **Novel tree ensemble framework which is scalable, accurate, and widely applicable.** Motivated by the tremendous success of tree ensembles such as Random Forests and Boosted Trees but also driven by the need to develop more interpretable ML methods, we introduce a novel tree sampling framework that is scalable, accurate, and applicable to a wide range of classification and regression tasks. We formalize the notions necessary for understanding and using the framework, and introduce the mathematical formulation of the XSTrees algorithm in §2.3.

- **Asymptotic and finite-sample guarantees on the learning of the underlying true model.** We establish that the XSTrees framework enjoys strong theoretical performance. In particular, we show that the framework is asymptotically consistent: as the number of training samples grows large, the expected model estimation error goes to zero. In the case of finite samples, we prove a finite-sample bound on the average distance between the position of the splits of the true underlying tree model and the learned XSTrees and show that this distance converges to 0 with rate $\mathcal{O}(\frac{\log(n)}{n})$, when $n \to \infty$, where n is the number of training observations. We detail this analysis in §2.4.

- **Strong computational performance to show high predictive power, wide applicability, and better interpretability.** We perform extensive numerical experiments to test the accuracy, sensitivity, robustness, and interpretability of the XSTrees framework. These experiments are performed on:

1. *UCI Machine Learning Repository*: In §2.5, we perform extensive numerical experiments on publicly available and widely used datasets from the UCI Machine Learning Repository. We find that out of the 25 randomly selected datasets, both for classification and regression tasks, the XSTrees algorithm outperforms other benchmark algorithms (for example, Random Forests, XGBoost, and Feed-Forward Neural Networks) in terms of average out-of-sample accuracy with on average between 2.5% and 50% higher accuracy for classification, and between 2% and 85% higher R^2 for regression. Furthermore, the proposed algorithm has the least variability (in terms of out-of-sample accuracy) which provides further proof of its consistent performance across a wide range of applications.

2. *Synthetic Data*: In §2.6, we perform a numerical study with synthetically generated data to compare different methods in terms of out-of-sample accuracy. We find that XSTrees outperforms other benchmark algorithms, especially in the case when the number of training samples is small, or the data is noisy. We also introduce in §2.3.3 a framework to visualize the XSTrees distribution and extract useful explanations such as, but not limited to, the importance of the features and the most probable tree, or *mode tree*, in this distribution, which is similar to a single CART tree.

3. *Real-World Case Studies*: We perform two extensive case studies. The first one on revenue management from real world data from our industry collaborator (§2.7). This case study focuses on sales forecasting, and how XSTrees outperforms the state-of-the-art industry standards, while providing decision-makers with tools to interpret the predictions and act upon them. We also discuss a second case study on blood transfusion (§2.8) to illustrate the interpretability of the XSTrees model, as discussed in §2.3.3. We show through this case study how XSTrees maintain a high level of accuracy, while providing more explanations on how the predictions are made than alternative ensemble methods.

2.2 Related Literature

The objective of this chapter is to propose a probabilistic tree ensemble that outperforms the state-of-the-art ML methods in terms of accuracy, while providing provable guarantees and better ways to interpret the results. Consequently, we discuss in this section bodies of literature that relate to (i) high-performance predictive ML methods that we benchmark the XSTrees against, (ii) probabilistic extensions of tree ensembles and (iii) interpretation of tree-based models; and finally (iv) application on data driven forecasting techniques to revenue management.

State-of-the-art ML models for classification and regression: The two best-performing models in Kaggle ([1]) by number competitions won, based on predictive accuracy only, are XGBoost ([49]) for structured data, and Deep Learning ([107]) for unstructured data (See [15] and [37]). While unarguably powerful methods, both lack in terms of theoretical guarantees and interpretability, with the later needing a significant amount of data to achieve good performances. Inspired by the success of XGBoost, as well as the wide popularity of Random Forests ([40]), there has been throughout the years a tremendous growth in the development of new tree ensemble methods, such as AdaBoost ([135]) and LightGBM ([105]). All these methods have in common that they train single decision trees, often CART ([39]), which are called "weak learners" and aggregate them together. Random Forest constructs these single trees by sampling different subsets, both in terms of observations and features, from the data, then training CART trees on each of these subsets. XGBoost does this construction in a sequential way, by training a CART tree on the entire data, then subsequent CART trees on the residuals of the predictions of the ensemble of previously trained trees. AdaBoost differs from XGBoost by focusing on harder-to-classify examples through weighting the trees, while LightGBM has the particularity of growing trees by level (depth)-wise instead of splitting leaf nodes in the training process. XSTrees also aggregates tree-based weak learners, but does it in a more structured way, which allows it to train an entire distribution on the tree space, instead of an

[1] https://www.kaggle.com/competitions

independent collection of trees.

Last but not least, a certain number of decision makers prefer to use simpler, but less powerful methods ([84]) such as Linear Regression ([118] for personalized pricing), k-Nearest Neighbors ([3])), Support Vector Machines ([148]), or just single CART Trees ([80] for sales forecasting) because of their interpretability and scalability. [123] review the applications of these methods in operations management in general, while [81] focus on retail forecasting in particular. Consequently, we will benchmark the XSTrees algorithm against all the methods cited above in the experimental section (§2.5) on (i) publicly available data, (ii) synthetic data, and (iii) real-world case studies.

Probabilistic extensions of tree ensembles: Bayesian interpretation of decision trees was first provided by [53] where the authors assumed priors on tree split probabilities and then used stochastic search to guide improved tree generation. Similarly, [90] use a Gaussian process model to generate Bayesian trees. More recently, [149] generate candidate ensemble trees based on a non-parametric prior distribution fitted using the training sample, while [14] propose a PAC-Bayesian framework for decision trees, which weights the majority vote of tree ensembles in order to get probabilistic guarantees without trying to learn the underlying distribution over the tree space. Nevertheless, most of the existing Bayesian frameworks rely on Markov Chain Monte Carlo (MCMC) methods which can be computationally slow and can suffer from ensemble outcomes ([106]). Our proposed method also generates empirical distributions on the number of splits and the location of these splits in each feature dimension to generate potentially infinite candidate trees. Nevertheless, we *extend* our trees in the decision space, which leads to considerably different prior assumptions on the sampling distribution. Furthermore, we also show theoretical convergence results for the proposed algorithm, and perform extensive numerical experiments to show considerable gains over other state-of-the-art algorithms.

Interpretation of tree-based models: Arguably one of the most interpretable machine learning prediction framework was introduced by [39] in the form of CART. Since then, many attempts have been made to improve the accuracy of CART using

tree-based ensembles, sometimes at the expense of interpretability ([40], [49]). Nevertheless, the importance of interpretable methods has been recognized since many years. For example, [42] considers a simple tree predictor obtained from synthetic data generated from the training set to bring accuracy to tree ensembles. Similarly, [120] generates optimal partitions of the data region in order to generate sparse prediction representations. [93] poses the interpretability problem as a Bayesian model selection problem with the objective of a simplified tree-based model. Finally, more recent attempts by [70] and [113] aim to develop a rigorous framework of interpretability and define the characteristics of interpretable models. Finally, more recently, [67] constructs simplified trees by learning frequent feature interactions in a tree ensemble. Instead, the algorithm proposed in this chapter generates an empirical distribution on the splits as well as the location of these splits and we use this distributional information to make the tree ensemble more interpretable. As such the proposed visualization framework aims at making the underlying method more transparent. We comment on the interpretation of the XSTrees in §2.6, §2.8, and §2.7.

Data-driven forecasting in revenue management: Data driven forecasting techniques have played an important role in various revenue management problems, including pricing, assortment optimization and inventory management. We refer the interested readers to [150] for a general overview of these topics. In recent years, the number of papers using data-driven techniques has increased a lot, mainly due to the wider availability of data. Some examples include, [80] who demonstrate how demand forecasting can be done for a multi-product price optimization problem. [75] show how to use forecasting for bundle pricing optimization. Others, including [5, 50, 48], focus on the problem of assortment optimization. On the problem of forecasting demand, [4, 160] consider the problem of forecasting demand with censored sales observations. [10] consider the problem of forecasting demand for new products. The current work is complementary to these studies since it develops a general purpose, non-parametric prediction algorithm. We further demonstrate that the method outperforms state of the art prediction algorithms and how the output can be used to generate insightful visualizations.

2.3 Approach

2.3.1 Preliminaries

Let $x \in \mathbb{R}^d$, $d \in \mathbb{N}$, a $d-$dimensional vector. We denote $x[i]$ the i^{th} component of the vector and $[d]$ the set $\{1, \ldots, d\}$. We discuss some preliminaries that will be used for the tree sampling framework we introduce next:

Definition 1 (Parallel Split) *A parallel split is a constraint of the form* $\{x[i] \leq b\}$*, defined by a dimension* $i \in [d]$ *and a position* $b \in \mathbb{R}$*. If* c *is this parallel split, we note* $c(d)$ *its dimension, i.e.,* $c(d) = i$*, and* $c(p)$ *its position, i.e.,* $c(p) = b$*. e.g., if* c *is the split* $\{x[3] \leq 5\}$*,* $c(d) = 3$ *and* $c(p) = 5$*.*

Definition 2 (Parallel Splits Tree) *A parallel splits tree is a partitioning of the space* $\mathbb{R}^d$ *using exclusively a finite number of parallel splits. More formally, let* $\{c^1, \ldots, c^r\}$*,* $r \in \mathbb{N}$*, be the set of all parallel splits used in this partitioning. Each region of the space defined by this partitioning is called a leaf node, and is fully characterized by a subset of* $\{c^1, \ldots, c^r\}$ *and whether or not each constraint in this subset is true or false (satisfied, or its opposite is satisfied). We denote* $\mathcal{L}(x)$ *the leaf node where* x *belongs.*

Definition 3 (Extended Tree) *An extended tree is a parallel splits tree for which each leaf node is defined by the same set of parallel splits* $\{c^1, \ldots, c^r\}$*,* $r \in \mathbb{N}$ *and whether or not each of these constraints is true or false. i.e. the number of leaf nodes for an extended tree is always* 2^r*.*

Definition 4 (Decision Tree) *We refer here to a decision tree as a parallel splits tree for which each leaf node is associated with a prediction. Note that a decision tree is fully defined by a set of leaf nodes, and a predictive function* $y : \mathbb{R}^d \to \mathbb{R}$*, such that* $\forall\ x_1,\ x_2 \in \mathbb{R}^d,\ \mathcal{L}(x_1) = \mathcal{L}(x_2) \implies y(x_1) = y(x_2)$*.*

Definition 5 (Extended Decision Tree) *We finally define an extended decision tree as a tree that is an extended tree and a decision tree at the same time.*

Example 1 *The tree in Figure 2-1 is a decision tree, where the possible splits are* $c^1 = \{x[1] \leq 10\}$ *and* $c^2 = \{x[2] \leq 5\}$*, there are three leaf nodes* $\mathcal{L}_1$*,* $\mathcal{L}_2$*,* $\mathcal{L}_3$ *defined respectively by* $\{\bar{c}^1\}$ *(*c^1 *is false),* $\{c^1, c^2\}$*, and* $\{c^1, \bar{c}^2\}$*. The predictive function* y *of this tree is such that* $y(x) =$ *"Green" if* $x \in \mathcal{L}_3$*, "Red" otherwise.*

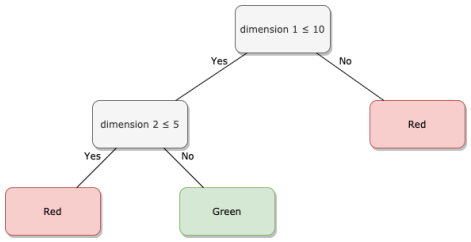


Figure 2-1: 2-dimensional example of a decision tree of depth 2.

Similarly, the tree extension $\mathcal{T}^*$*, of the decision tree in Figure 2-1 is represented in Figure 2-2. The set of possible constraints is still the same:* $c^1 = \{x[1] \leq 10\}$ *and* $c^2 = \{x[2] \leq 5\}$ *and so is the predictive function, however, there are now four leaf nodes* $\mathcal{L}_1^*$*,* $\mathcal{L}_2^*$*,* $\mathcal{L}_3^*$ *and* $\mathcal{L}_4^*$ *defined respectively by* $\{c^1\, c^2\}$*,* $\{c^1\, \bar{c}^2\}$*,* $\{\bar{c}^1\, c^2\}$ *and* $\{\bar{c}^1\, \bar{c}^2\}$*.*

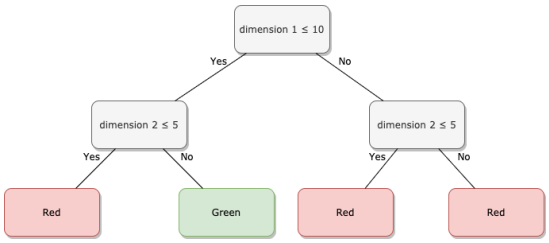


Figure 2-2: Linear extension of the tree in Figure 2-1.

Proposition 1 *For each decision tree, there exists an extended decision tree with the same predictive function.*

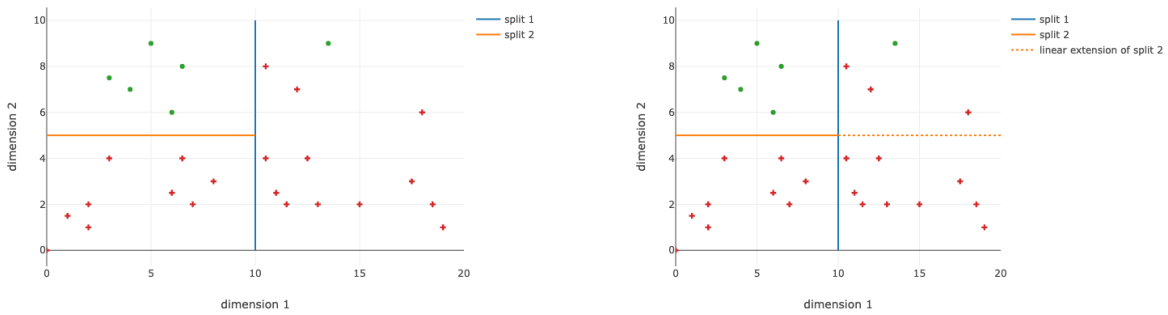
Proof of Proposition 1.

Let $\mathcal{T}$ a decision tree with predictive function y, and let $\{c^1, \ldots, c^r\}$, $r \in \mathbb{N}$ the union of all the parallel splits that define its leaf nodes. We define the tree $\mathcal{T}^*$ where the leaf nodes are defined by the entire set $\{c^1, \ldots, c^r\}$ and whether or not each of these constraints is true or false. $\mathcal{T}^*$ is, by definition, an extended tree. Let $x \in \mathbb{R}^d$. Let $\mathcal{L}^*$ a leaf node of $\mathcal{T}^*$ containing x, and satisfying all constraints in $\{\hat{c}^1, \ldots, \hat{c}^r\}$,

where $\hat{c}^i = c^i$ if c^i is true in $\mathcal{L}^*$, and $\hat{c}^i = \bar{c}^i$ if c^i is false, for $i \in [r]$. Since the leaf nodes of $\mathcal{T}$ form a partition, there exist a leaf node $\mathcal{L}$ of $\mathcal{T}$, such that $x \in \mathcal{L}$. Since $\{c^1, \ldots, c^r\}$ is the union of all the parallel splits that define the leaf nodes of $\mathcal{T}$, the constraints of $\mathcal{L}$ are a subset of the constraints of $\mathcal{L}^*$, i.e. $\mathcal{L}^* \subseteq \mathcal{L}$. We can consequently define a predictive function y^* for $\mathcal{T}^*$ such that $y^*(x) = y(x),\ \forall x \in \mathbb{R}^d$, while ensuring that all points in the same leaf node of $\mathcal{T}^*$ have the same prediction, and hence maintaining a decision tree structure. $\mathcal{T}^*$ is an extended tree and a decision tree, and has the same predictive function as $\mathcal{T}$. ■

Remark 1 *Note that for the leaf nodes to form a partition, and hence have a tree structure, it is necessary to use a lexicographical order on the constraints to define these leaf nodes. This makes the splits of the parallel split trees, and by extension, the decision trees order-dependent. The intuition behind the tree extensions is that it removes the conditions on the order of the splits, at the cost of increasing the number of leaf nodes.*

Example 2 *In $\mathbb{R}^d$, this procedure is equivalent to linearly extending the splits that only cover a part of the space in the partition, as illustrated in Figure 2-3 for the same example as before.*



(a) Tree where the order of the splits matters. (b) Order-independent linear extension.

Figure 2-3: 2-dimensional visualization of the tree extension procedure.

2.3.2 Additional Notations

Let $\mathcal{D}_n$ a dataset of size $n \in \mathbb{N}$, containing observations $\{x_1, \dots, x_n\}$ in $\mathbb{R}^d$. Let θ a random sample of $\mathcal{D}_n$, both in terms of observations and in terms of dimensions. CART($\mathcal{D}_n,\ \theta$) denotes the decision tree resulting in fitting a Classification and Regression Tree (CART, [39]) on the sample θ of the dataset $\mathcal{D}_n$. Each leaf node of such a tree contains training samples from $\{x_1, \dots, x_n\}$. For any vector $x \in \mathbb{R}^d$, we note $\mathcal{L}_n(x,\ \theta)$ the leaf node where x belongs for CART($\mathcal{D}_n,\ \theta$), and $N_n(x,\ \theta)$ the number of training set in the corresponding leaf node, that is

$$N_n(x,\ \theta) = \sum_{i=1}^{n} \mathbb{1}\{x_i \in \mathcal{L}_n(x,\ \theta)\}\,.$$

Additionally, for any tree for which the set of possible constraints that appear in any of its leaf nodes is $\{c^1, \dots,\ c^r\}$, $r \in \mathbb{N}$, we denote

$$\text{sp}_j = \bigcup_{i=1}^{r} \{c^i : c^i(d) = j\},\ \forall j = 1, \dots,\ d\,,$$

the set of all splits that occur on dimension j and $\vec{s} = (|\text{sp}_1|, \dots, |\text{sp}_d|)$ denotes the compact representation of the number of splits in each dimension. Similarly,

$$\vec{l_j} = \{c^i(p),\ \forall c^i \in \text{sp}_j\}\,,$$

denotes the location of the $|\text{sp}_j|$ splits in the j^{th} dimension (ordered in increasing order of the split location). Note that an extended tree is fully defined by $(\vec{s},\ \vec{l_1},\ \dots,\ \vec{l_d})$. Moreover, we refer to the size of a leaf node, noted $||.||$ as the euclidean volume of the hyper-cube defined by this leaf node ($+\infty$ if the leaf node is not bounded). Finally, $e_i \in \mathbb{R}^d$ denotes the i^{th}-dimensional unit vector in the d-dimension, and for any estimator $\hat{r}_n(\mathbf{X})$ (where n is the size of the training sample) of an unknown function $r(\mathbf{X}) : \mathbb{R}^d \to \mathbb{R}$, we say that $\hat{r}$ is asymptotically consistent if $\mathbb{E}_{\mathbf{X}}\left[(\hat{r}_n(\mathbf{X}) - \mathbf{r}(\mathbf{X}))^2\right] \to 0$ as $n \to \infty$.

2.3.3 Intuition

In this section, we introduce our sampling-based distributional tree framework. Like any other ML method, XSTrees uses training data to make out-of-sample predictions. In particular, let $X_{train} \in \mathbb{R}^{n \times d}$ denote the training sample with n rows and d features. Similarly, let Y_{train} denote the outcomes corresponding to the n training samples. Assume that the n samples are i.i.d. and generated from the joint distribution $\mathbb{P}_{X,Y}$. Then, the objective is to estimate a regression (classification) function $r_n(x^{test}) : \mathbb{R}^d \to \mathbb{R}$ that minimizes the out-of-sample prediction error:

$$\min_{r_n} \mathbb{E}[(Y(\mathbf{X}) - r_n(\mathbf{X}))^2],$$

where the expectation is taken over the joint distribution of features X and outcome variable $Y(X)$. CART locally solves this minimization problem over a single decision tree, while Random Forests (RF, [40]) solves it by generating a family of decisions trees by fitting CARTs on a random sample (generated through boot-strapping or sub-sampling) of the training data. Each of these trees is used to generate a prediction for the test data point, and the final prediction is an ensemble of all the individual predictions. Our proposed method also generates tree ensembles, but combines them in a more structured way. More precisely, instead of learning an independent family of decisions trees like RF does, XSTrees learns the entire distribution of the tree space, and the samples from this distribution constitute our ensemble.

While it is theoretically possible to learn the distribution over any tree space within this framework, data complexity issues make this impractical. This is why we learn tree distributions on extended decision trees only. Proposition 1 ensures that it can be done without any loss of generality, and the order-independent structure of extended trees ensures that the approach is tractable. Indeed, for an extended tree, we only need to learn the distribution of the number of splits on each dimension, and the distribution of the position of each of these splits. The idea is to generate a collection of decision trees from the data and extend them. Then, extract and order the positions of the splits in each dimension separately, and use these samples

to either (i) update similarly-ordered priors on the distribution of these splits, for a Bayesian approach or (ii) compute an empirical distribution on these quantities.

Example 3 *We illustrate this in a simplified example in Figure 2-4.*

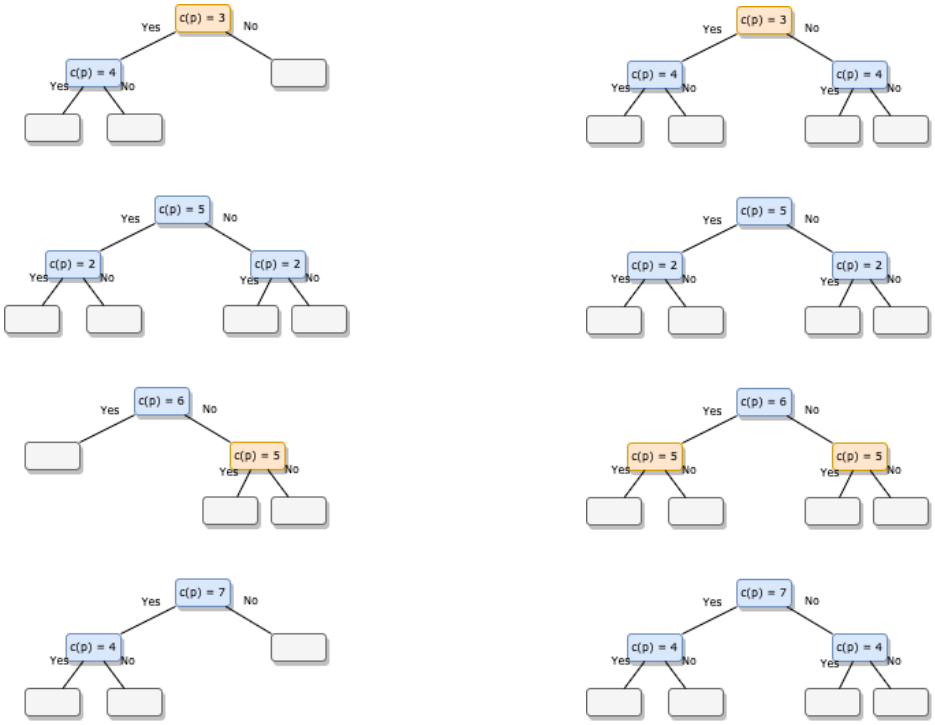
In the Bayesian approach, another way to choose which split to update is simply by selecting the one that maximizes the likelihood (computed according to the prior distribution) of observing the sampled split at each iteration. In the following section we focus exclusively on the first update procedure, but both are tested in the computational experiments, as the second one sometimes achieves better performances.

2.3.4 Algorithm

We present the XSTrees Algorithm (Algorithm 1), the sampling based tree algorithm that we propose. The algorithm takes as an input the training data $\mathcal{D}_n$, the total number of CART trees (K) that are generated through a pre-selected randomization procedure (denoted through θ), the maximum number of splits in any dimension S^{max}, and the size of the tree ensemble T^{max}, for generating final predictions. Finally, the objective is to predict the outcome for a new test point x.

We divide the algorithm into three main steps. The *model fitting* step uses CART trees generated on random sub-samples to generate distributions of the splits and their location in each dimension. Each tree is first *extended* (see §2.3.1). Then, the tree extensions are used to estimate the average number of splits and the location of these splits and the variance around each of the splits. The mean and the variance ($\hat{s}^{mean}$ and $\hat{s}^{var}$) provide moment information to sample and create an ensemble of trees, which is executed in the *prediction sampling* step.

In the prediction sampling step, we generate a total of T^{max} trees by sampling the number of splits in each dimension as well as the location of the splits based on the distribution estimated in the *model fitting* step. The sampled trees are all extended trees. This ensures ease of sampling since splits in different dimensions can be sampled independently from one another.



(a) Collection of trees trained from the data. (b) Linear extensions of these trees.

Figure 2-4: This is an illustration of the intuition behind the XSTrees algorithm in a 4-sample, 2-depth example. We train a collection of CARTs from 4 samples of the data to obtain 4 trees (a), then we extend the splits of (a) to obtain 4 extended trees (b). The blue splits refer to splits on dimension 1, and the orange splits refer to splits on dimension 2. We notice that for dimension 1, it appears twice with 1 split, and twice with 2 splits. Empirically, we set $P(|\text{sp}_1| = 0) = 0$ and $P(|\text{sp}_1| = 1) = P(|\text{sp}_1| = 2) = 0.5$, and from the smallest split in each of the 4 trees, we conclude $\vec{l}_1[1] \sim \mathcal{N}(4, 1.41)$ assuming a normal distribution, and by computing the average and standard deviation of $[4, 2, 6, 4]$. Similarly, we set $\vec{l}_1[2] \sim \mathcal{N}(6, 1)$, $P(|\text{sp}_2| = 0) = P(|\text{sp}_2| = 1) = 0.5$, and $\vec{l}_2[1] \sim \mathcal{N}(4, 1)$ to fully define the distribution of the tree space, that we can then sample using Monte Carlo sampling to make our final predictions.

Nevertheless, it also increases the total number of leaf nodes in the generated tree. Hence, the prediction of a test point (using h^{XST} (Algorithm 2)) is done using advanced pruning procedures and a nearest neighbor approach in regions of the space which do not have enough training data. For example, any generated extended tree defines a partition of the data space. To predict the outcome for a test point, we first find the partition that contains the test point. The predicted outcome is the average outcome from all the training samples in the same partition as the test point. In case there are no training points (or not enough training points $n_{\min} \in \mathbb{N}$) in the partition, we use one of two heuristic methods to make the prediction: the first randomly selects a dimension and find the closest non empty data partition by following that dimension and averaging over the training samples in the non-empty partition. This approach replicates what a regular CART does by merging the empty region of the space with a neighboring leaf but has the advantage of adding randomization through a large number of sampled trees to improve the estimation. The second one relies on an approximate k-nearest neighbor search algorithm to find the nearest data points that belong to neighbouring non-empty grid cells. The heuristic to use, as well as $n_{\min}$ and k are chosen through hyper-parameter tuning (See Appendix §A.2). It is important to note that these procedures, in addition to the common tree regularization techniques, help limit over-fitting.

The final prediction is an average of the predicted outcomes from all sampled trees and is performed in the final *averaging predictions* step of the algorithm. This can be seen as the expected value of the predicted value knowing the tree distribution. Note that in the case of classification, this step is replaced by a majority vote weighted by the probability of each tree.

Remark 2 *Note that the normality assumption is on the position of the splits only, not on the underlying data distribution, nor on the predictive function of the tree and it can be easily replaced by an empirical distribution or other discrete (Dirichlet) and continuous distributions. The current version of the code supports all* `numpy` *([94]) distributions as well as empirical distribution for the non-bayesian approach. In practice, we have benchmarked different distributions, including the exponential*

distribution and the Cauchy distribution, but the normal distribution was the best performing one. The theoretical results hold regardless of this choice, as long as the averages converge at similar rates. Once such distributions are generated, sampling can be then used to generate candidate trees for prediction.

Algorithm 1 XSTrees($\mathcal{D}_n$, K, T^{max}, S^{max}, x)

Model Fitting

for $i \in [K]$ **do**

Let h_i = CART($\mathcal{D}_n$, θ_i), [θ_i is a randomization] and $(\vec{s},\ \vec{l}_1, \ldots,\ \vec{l}_d)_i$ = TreeExtension(h_i) (see §2.3.1 for details).

end for

for $j \in [d]$ **do**

Let $\hat{s}_j^{mean} = \frac{1}{K}\sum_{i=1}^{K} \vec{s}_j[i]$, $\hat{s}_j^{var} = \frac{1}{K}\sum_{i=1}^{K}\left(\vec{s}_j[i] - \hat{s}_j^{mean}\right)^2$.

for $m \in [S^{max}]$ **do**

Let $\kappa_j^{(m)} = \{i : s_j[i] \geq m\}$, $\mu_j^{(m)} = \frac{1}{|\kappa_j^{(m)}|}\sum_{i\in\kappa_j^{(m)}} \vec{l}_j^{\,i}[m]$ & $\sigma_j^{(m)} = \frac{1}{|\kappa_j^{(m)}|}\sum_{i\in\kappa_j^{(m)}}\left(\vec{l}_j^{\,i}[m] - \mu_j^{(m)}\right)^2$.

end for

end for

Prediction Sampling

for $q \in T^{max}$ **do**

for $j \in [d]$ **do**

$\tilde{\nu}_j^q = \mathcal{N}\left(\hat{s}_j^{mean},\ \hat{s}_j^{var}\right)$

for $m \in \lfloor \tilde{\nu}_j^q \rfloor$ **do**

$\tilde{ls}_j^q(m) = \mathcal{N}(\mu_j^{(m)},\ \sigma_j^{(m)})$

end for

end for

Let $\hat{y}_q(x) = h^{XST}(\tilde{ls}_1^q, \ldots,\ \tilde{ls}_d^q, x)$ (see Algorithm 2)

end for

Averaging Predictions

Predict $\hat{y}(x) = \frac{1}{T^{max}}\sum_{r=1}^{T^{max}} \hat{y}_q(x)$.

Algorithm 2 $h^{XST}(\mathcal{D}_n,\ \tilde{l}_1, \ldots,\ \tilde{l}_d,\ x)$.

if $|\mathcal{L}_n(x,\ \tilde{l}_1, \ldots,\ \tilde{l}_d)| > 0$ **then**
  Predict $\hat{y}(x) = \sum_{i=1}^{n} y_i \mathbb{1}\{x_i \in \mathcal{L}_n(x,\ \tilde{l}_1, \ldots,\ \tilde{l}_d)\}$
else
  Let i^r be chosen randomly from 1, ...,d.
  Let $\epsilon_{min} = \min_\epsilon\{\epsilon : |\mathcal{L}_n(x + \epsilon \cdot e_{i^r},\ \tilde{l}_1, \ldots,\ \tilde{l}_d)| > 0|\}$
  Predict $\hat{y}(x) = \sum_{i=1}^{n} y_i \mathbb{1}\{x_i \in \mathcal{L}_n(x + \epsilon_{min} \cdot e_{i^r},\ \tilde{l}_1, \ldots,\ \tilde{l}_d)\}$
end if

2.4 Theoretical Guarantees

While decision tree based ensemble methods have been widely popular, statistical properties of such methods are hard to establish because of the non-parametric structure of the predictor function. However, the distributional nature of XSTrees results in good theoretical guarantees, both in terms of finite samples and asymptotic convergence. In this section, we provide bounds for the XSTrees algorithm on the estimation error.

2.4.1 Assumptions

We consider without loss of generality the case of a classification problem with $m \in \mathbb{N}$ classes $\{1, \ldots, m\}$. We have training data $x_1, \ldots, x_n \in \mathbb{R}^d$, with labels $y_1, \ldots, y_n \in [m]$, where the true underlying classification model follows a decision tree structure $\mathcal{T}$ with predictive function y (see §2.3.1). From Proposition 1, we can assume that $\mathcal{T}$ is an extended tree, again without loss of generality. We impose the following assumptions (as is often done in this literature):

Assumption 1 (A1) *The feature space is bounded. That is,* $\forall i \in [n], x_i \in [0,1]^d$.

Assumption 2 (A2) *The observations are stochastic, i.e., for any* $i \in [n]$*, the label* y_i*, for* x_i *in the training data, is generated from the predictive function* y *of* $\mathcal{T}$ *with some error* $\epsilon \in [0, \frac{1}{2}]$*. We write* $P(y_i = y(x_i)) = 1 - \epsilon$.

Assumption 3 (A3) *The training data used for the learning is i.i.d. and uniformly sampled from* $[0,\ 1]^d$.

Assumption **(A1)** and **(A3)**, of bounded training data, and uniform i.i.d sampling, are standard in the machine learning literature. Assumption **(A2)** ensures that the class assignments through the underlying data generation process, $\mathcal{T}$, is not deterministic and there is some probability of observing an incorrect class for the sampled data. Note that ϵ in assumption **(A2)** does not have to be the same across different leaves. This part of the assumption is made in order to simplify the notation, while $\epsilon \leq \frac{1}{2}$ is necessary to make inference possible (otherwise the data is more often wrong than not). Our goal is to learn $\mathcal{T}$ from the training data, using the XSTrees procedure, and accurately approximate y.

2.4.2 Asymptotic Convergence

In this section, we focus on the asymptotic convergence of a single extended tree estimator to the true underlying extended tree. Let $\mathcal{V}$ be the class of single decision tree predictors such as CART or Optimal Trees ([22]), with regularization constraints (such as the minimum number of training observations in each leaf). We write the following optimization problem, minimizing the prediction error on the training data over $\mathcal{V}$ (Equation 2.1):

$$\begin{aligned} \min_{f} \quad & \sum_{i=1}^{n} l(y_i, f(x_i)) \\ \text{s.t.} \quad & f \in \mathcal{V}, \end{aligned} \tag{2.1}$$

where $l(a, b) = 0$ if $a = b$, and 1 otherwise.

We refer to this problem as $\mathcal{F}_n$, and note that what we mean by optimizing over $\mathcal{V}$ is learning the finite set of parameters of the tree. We refer the reader to [22] for an explicit mixed-integer formulation. CART is a local-optimum of $\mathcal{F}_n$ and Optimal Trees are theoretically a global-optimum of $\mathcal{F}_n$. Finally, we let $\mathcal{S}_n$ the set of all locally optimal solutions of $\mathcal{F}_n$.

Theorem 1 *Under assumptions **(A1)** - **(A3)**, there exists a sequence $\{\mathcal{T}_n \in \mathcal{S}_n\}$ which converges asymptotically to the true extended tree $\mathcal{T}$ when $n \to \infty$.*

Let ρ be the depth of $\mathcal{T}$, since $\mathcal{T}$ is an extended tree, there are 2^ρ leaf nodes in $\mathcal{T}$. For each leaf node, we take the geometrical center (i.e. the point obtained by averaging the upper bound and the lower bound in each dimension within this leaf node). This is possible since the feature space is bounded **(A1)**, we denote it g_i for each leaf node $i \in [2^\rho]$. Let $\delta \in \mathbb{R}_+$ such that the size of each leaf node (i.e. the size of a region delimited by the leaf node) is greater than δ. We write $||\mathcal{L}_\mathcal{T}(x)|| \geq \delta$, $\forall x \in \mathbb{R}^d$. Let p_{min}, such that $\delta > p_{min} > 0$, and let $n_{min} = n \times p_{min}$.

Let $\mathcal{T}_n^*$ denote the extended decision tree defined as follows. For all splits $c^j,\ j \in [\rho]$ of $\mathcal{T}$, we define a split c_n^{j*}, such that $c_n^{j*}(d) = c^j(d)$ and

$$c_n^{j*}(p) = \frac{\min_{i\in[n]}\{x_i[c^j(d)] : x_i[c^j(d)] \geq c^j(p)\} + \max_{i\in[n]}\{x_i[c^j(d)] : x_i[c^j(d)] \leq c^j(p)\}}{2}.$$

This provides a mapping between the leaf nodes of $\mathcal{T}$ and the leaf nodes of $\mathcal{T}_n^*$. We set the predictive functions to be equal according to this mapping. We denote $\mathcal{F}_n(n_{min})$ the problem $\mathcal{F}_n$ under the regularization constraint that the minimum number of observations in a leaf node is n_{min}, and $\mathcal{S}_n(n_{min})$ the set of locally optimal solutions of $\mathcal{F}_n(n_{min})$.

We argue that when $\mathcal{T}_n^*$ is feasible in $\mathcal{F}_n(n_{min})$ (Lemma 1), there exists $\mathcal{T}_n \in \mathcal{S}_n(n_{min})$ such that $\mathcal{T}_n^*$ and $\mathcal{T}_n$ have the same number of splits in each dimension, i.e., $\mathcal{T}_n$ is obtained by locally moving up (increase the value of the position) or down (decrease the value of the position) the splits of $\mathcal{T}_n^*$ (Lemma 2). This will allow us to prove the convergence of $\mathcal{T}_n \to \mathcal{T}$ when $n \to \infty$.

Let Ψ_i, for any leaf node i, be defined as

$$\Psi_i = \begin{cases} 1 \text{ if } \{|\{j \in [n],\ x_j \in \mathcal{L}_\mathcal{T}(g_i)\}| \geq n_{min}\} \\ 0,\ \text{otherwise,} \end{cases}$$

where Ψ_i takes value 1 if the leaf node i contains at least n_{min} data points and hence also depends on the size of the data set, n.

Lemma 1 *When Ψ_i is true for all $i \in [2^\rho]$, then $\mathcal{T}_n^*$ is feasible for $\mathcal{F}_n(n_{min})$. Addi-*

tionally, we have $\lim_{n\to\infty} P\left(\bigcap_{i=1}^{2^\rho} \Psi_i\right) = 1$. *We note this probability* p_1^n.

Proof of Lemma 1. $\mathcal{T}_n^*$ has by construction a tree structure, so the only other condition to verify for feasibility is the regularization constraint. By definition, Ψ_i for all $i \in [2^\rho]$ ensures that each leaf node of $\mathcal{T}$ has at least n_{min} observations. The definition of the splits of $\mathcal{T}_n^*$ guarantees that there are not observations between c_n^{j*} and c^j for all $j \in [\rho]$, so $\mathcal{T}_n^*$ and $\mathcal{T}$ define the same partitioning of the data. Consequently the number of observations in each leaf node of $\mathcal{T}_n^*$ is at least n_{min}, i.e. $\mathcal{T}_n^*$ is feasible.

Let us bound the probability $P\left(\bigcap_{i=1}^{2^\rho} \Psi_i\right)$:

$$\begin{aligned} P\left(\bigcap_{i=1}^{2^\rho} \Psi_i\right) &= 1 - P\left(\bigcup_{i=1}^{2^\rho} \bar{\Psi}_i\right) \\ &\geq 1 - \sum_{i=1}^{2^\rho} P(\bar{\Psi}_i) \\ &\geq 1 - \sum_{i=1}^{2^\rho} \sum_{k=0}^{\lfloor n_{min} \rfloor} \binom{n}{k} ||\mathcal{L}_\mathcal{T}(g_i)||^k (1 - ||\mathcal{L}_\mathcal{T}(g_i)||)^{n-k} \end{aligned} \tag{2.2}$$

This terms bounds the probability of having less than n_{min} observations in a given leaf node i, from assumption **(A3)**, we know that the training data is uniformly distributed, and hence the probability of being in a given leaf node is exactly the size of the leaf node, divided by the total size of the features space. The size of the features space is 1 **(A1)**, so this is exactly $||\mathcal{L}_\mathcal{T}(g_i)||$ for leaf node i. The independence of the observations **(A3)** gives this binomial structure.

We have, for $i \in [2^\rho]$, that $\sum_{k=0}^{\lfloor n_{min} \rfloor} \binom{n}{k} ||\mathcal{L}_\mathcal{T}(g_i)||^k (1 - ||\mathcal{L}_\mathcal{T}(g_i)||)^{n-k}$ is the cumulative density function for a binomial distribution of n samples, with individual probability $||\mathcal{L}_\mathcal{T}(g_i)||$ evaluated at point $\lfloor n_{min} \rfloor$. Since $\lfloor n_{min} \rfloor \leq n_{min} = n \times p_{min} \leq n \times \delta \leq n \times ||\mathcal{L}_\mathcal{T}(g_i)||$, we know this probability goes to 0 when $n \to \infty$. Since ρ is finite, so is 2^ρ, so we have a finite sum of terms converging to 0 when $n \to \infty$. We conclude $\lim_{n\to\infty} p_1^n = \lim_{n\to\infty} P\left(\bigcap_{i=1}^{2^\rho} \Psi_i\right) = 1$. We have proven Lemma 1. ■

Additionally, for a region A (e.g. a leaf node), we define the random variable

$$\Phi(A,\ n_{min}) =$$
$$\begin{cases} 1 \text{ if } \{\forall I \subseteq [n] : |\{j \in I : x_j \in A\}| \geq n_{min}, \\ \qquad |\{j \in I : x_j \in A,\ y(x_j) \neq y_j\}| \leq |\{j \in I : x_j \in A,\ y(x_j) = y_j\}|\} \\ 0 \text{ otherwise.} \end{cases}$$

That is, for region A, there is no subset of the training data that is in A, of size at least n_{min}, for which the proportion of misclassified points is greater than the proportion of correctly classified points. We denote Φ_i the random variable $\Phi(\mathcal{L}_{\mathcal{T}}(g_i), n_{min})$, for $i \in [2^\rho]$.

Lemma 2 *When Φ_i is true for all $i \in [2^\rho]$, no further feasible splits on $\mathcal{T}_n^*$ can improve the objective function, and $\lim_{n\to\infty} P\left(\bigcap_{i=1}^{\rho} \Phi_i\right) = 1$. We note this probability p_2^n.*

Proof of Lemma 2.

By contradiction, consider a feasible split c on $\mathcal{T}_n^*$ that improves the objective function of $\mathcal{F}_n(n_{min})$. Let $\mathcal{L}_{\mathcal{T}}(g_i)$, $i \in [\rho]$ be a leaf of $\mathcal{T}$ such that splitting on this leaf improves the objective function (there exists at least one by assumption). Let $\mathcal{L}_1$, $\mathcal{L}_2$ be the two leaves obtained by splitting this leaf with c. We know that at least one of these two leaves has a different prediction from $y(g_i)$, otherwise the predictive function on this leaf is identical to $\mathcal{T}_n^*$ (and hence there is no improvement on the objective). This means that the percentage of misclassified points in this leaf is greater than that of correctly classified points. Since Φ_i is true, this leaf contains less than n_{min} points, and hence is not feasible, we obtain a contradiction and conclude that no further feasible splits on $\mathcal{T}_n^*$ can improve the objective function. Let n_i the number of training points in leaf node $\mathcal{L}_{\mathcal{T}}(g_i),\ i \in [2^\rho]$. We also have:

$$
\begin{aligned}
P\left(\bigcap_{i=1}^{2^\rho}\Phi_i\right) &= 1 - P\left(\bigcup_{i=1}^{2^\rho}\bar{\Phi}_i\right) \\
&\geq 1 - \sum_{i=1}^{2^\rho} P(\bar{\Phi}_i) \\
&\geq 1 - \sum_{i=1}^{2^\rho}\sum_{k=\lceil n_{min}\rceil}^{n_i}\sum_{k'=\lceil \frac{k}{2}\rceil}^{k}\binom{k}{k'}\epsilon^{k'}(1-\epsilon)^{k-k'}
\end{aligned}
\tag{2.3}
$$

Since we have $\lceil \frac{k}{2}\rceil \geq \frac{k}{2} \geq k\epsilon$, we bound the interior sum by $\exp(-2k(\frac{\lceil \frac{k}{2}\rceil}{k} - \epsilon)^2)$, which can be bounded by $\exp(-2k(\frac{1}{2} - \epsilon)^2)$. The sum $\sum_{k=1}^{n}\exp(-2k(\frac{1}{2} - \epsilon)^2)$ is a convergent series in n, and $\lim_{n\to\infty}\lceil n_{min}\rceil = \infty$, so $\lim_{n\to\infty}\sum_{k=\lceil n_{min}\rceil}^{n_i}\sum_{k'=\lceil \frac{k}{2}\rceil}^{k}\binom{k}{k'}\epsilon^{k'}(1-\epsilon)^{k-k'} = 0$. Since 2^ρ is finite, we can conclude that $\lim_{n\to\infty} P\left(\bigcap_{i=1}^{2^\rho}\Phi_i\right) = 1$. We have proven Lemma 2. ■

Proof of Theorem 1.

We do not penalize the number of splits in the objective function and we do not impose any constraint on the predictive function, besides that it is the same within each leaf, hence, pruning will not improve the objective function. Consequently, Lemma 2 allows us to conclude that there exists $\mathcal{T}_n \in \mathcal{S}_n(n_{min})$ with the same number of splits as $\mathcal{T}_n^*$ and $\mathcal{T}$. We call these splits c_n^j for $j \in [\rho]$, mapped to the splits c^j of $\mathcal{T}$ by ordering of the splits similarly to the XSTrees update procedure. Finally, we call $d_j^n = |c_n^j(p) - c^j(p)|$, and $\mathcal{U}_{(k,n)}$ the distribution of a k-order statistic of n i.i.d samples from a uniform distribution in [0, 1] (i.e. the distribution of the k^{th} smallest element of these samples, for $k \in [n]$.

Let $j \in [\rho]$ and $k(n) \in [n]$, and $\Gamma^+_{j,k(n),n} = \Phi(\{x_i, i \in [n] : x_i[c^j(d)] > x_i[c^j(p)]\}, k(n))$ (resp. $\Gamma^-_{j,k(n),n} = \Phi(\{x_i, i \in [n] : x_i[c^j(d)] < x_i[c^j(p)]\}, k(n))$). Recall from the definition that it is the event that no subset of point that are above (resp. below) split c^j of size more than k has a greater proportion of misclassified points than correctly classified points. When $\Gamma^+_{j,k(n),n}$ and $\Gamma^-_{j,k(n),n}$ are true, there exists a random variable $u^{j+}_{k(n),n} \sim \mathcal{U}_{(k(n),n)}$ (capturing the move above) and $u^{j-}_{k(n),n} \sim \mathcal{U}_{(k(n),n)}$ (capturing the move below) such that d_j^n is bounded by $u^{j+}_{k(n),n} + u^{j-}_{k(n),n}$. Let $k(n)$ s.t. $k(n) \to \infty$

and $\frac{k(n)}{n} \to 0$. We have:

$$
\begin{aligned}
E\left[\max_{j\in[\rho]} d_j^n\right] &\leq E\left[\sum_{j=1}^{\rho} d_j^n\right] \\
&\leq \sum_{j=1}^{\rho} E\left[d_j^n\right] \\
&\leq \sum_{j=1}^{\rho} \left(E\left[u^{j+}_{k(n),n} + u^{j-}_{k(n),n}\right] P(\Gamma^+_{j,k(n),n} \cap \Gamma^-_{j,k(n),n}) + 1 - P(\Gamma^+_{j,k(n),n} \cap \Gamma^-_{j,k(n),n})\right)
\end{aligned} \tag{2.4}
$$

We have that $\lim_{n\to\infty} P(\Gamma^+_{j,k(n),n} \cap \Gamma^-_{j,k(n),n}) = 1$ since $k(n) \to \infty$ similarly to the proof of Lemma 2, and $\lim_{n\to\infty} E[u^{j+}_{k(n),n}] = \lim_{n\to\infty} E[u^{j-}_{k(n),n}] = 0$ because $\mathcal{U}_{(k(n),n)}$ follows a Beta distribution $\text{Beta}(k(n), n+1-k(n))$, with an expected value $\frac{k(n)}{n+1} \to 0$ when $n \to \infty$. We conclude that $\lim_{n\to\infty} E[\max_{j\in[\rho]} d_j^n] = 0$. i.e., we have proven that with probability at least $p_1^n p_2^n \to 1$ when $n \to \infty$, there exist a sequence of extended trees in $\mathcal{S}_n$ that converge to the true extended tree $\mathcal{T}$. ■

For any function $k(n)$ satisfying $k(n) \to \infty$ and $\frac{k(n)}{n} \to 0$, we denote $p_3^{n,k(n)} = P\left(\bigcap_{j=1}^{\rho} (\Gamma^+_{j,k(n),n} \cap \Gamma^-_{j,k(n),n})\right)$ for the purpose of proving finite sample guarantees in the next subsection.

Remark 3 *Note that this proof is not constructive, as we do know the parameters of the true underlying tree.*

2.4.3 Finite Sample Guarantees on the XSTrees

This learning procedure also gives us bounds on the error of the estimation for finite samples with the averaging effect of the XSTrees.

Theorem 2 *Let c^j for $j \in [\rho]$ a split of the underlying true decision tree $\mathcal{T}$, and c_n^j the estimated split for training data of size $n \in \mathbb{N}$. Using the same notations as before, we have:*

$$
E\left[|c_n^j(p) - c^j(p)|\right] \leq \frac{2k(n)}{n+1} p_1^n p_2^n p_3^{n,k(n)} + (1 - p_1^n p_2^n p_3^{n,k(n)}),
$$

For all function $k(n)$ satisfying $k(n) \to \infty$ and $\frac{k(n)}{n} \to 0$, and where p_1^n is defined in Lemma 1, p_2^n is defined in Lemma 2, and $p_3^{n,k(n)}$ is defined at the end of the proof of Theorem 1. Additionally, we can write:

$$E\left[|c_n^j(p) - c^j(p)|\right] = \mathcal{O}\left(\frac{\log(n)}{n}\right).$$

This theorem ensures that the splits of the XSTrees within this setting converge with rate $\mathcal{O}\left(\frac{\log(n)}{n}\right)$ to the splits of the true underlying tree. Note that this bound does not explicitly depend on d, but it does depend exponentially on the total number of splits ρ through $p_1^n p_2^n p_3^{n,k(n)}$. However, the extended tree structure of the XSTrees ensures that this dependency appears only in the multiplicative constants. Note that a similar exponential dependency on d can be found in the convergence results of Random Forests, with rate $\mathcal{O}(n^{-1/(8d+2)})$ ([86]) against which the XSTrees compare favorably.

Remark 4 (Remark on the notations) *If the estimated solution does not have the same number of splits as $\mathcal{T}$, we set $c^j(p)$ for $j \in [\rho]$ to the closest split that is in the same dimension, or to 1 if such split does not exist. Otherwise, we keep using the ordering of the XSTrees procedure.*

Proof of Theorem 2.

Let Ω the random variable $\left(\bigcap_{j=1}^{\rho}(\Gamma^+_{j,k(n),n} \cap \Gamma^-_{j,k(n),n})\right) \cap \left(\bigcap_{i=1}^{2^\rho}(\Psi_i \cap \Phi_i)\right)$. We have proven in the Proof of Theorem 1, that under Ω, the constructed sequence $\mathcal{T}_n$ is feasible and in $\mathcal{S}_n$ of the optimization problem $\mathcal{F}_n$ under the regularization constraints discussed in the proof of Theorem 1, and that the distance between each estimated split and the true split of $\mathcal{T}$ is bounded by the sum of two beta-distributed random variables u_n^+ and u_n^- with mean $\frac{k(n)}{n+1}$. If Ω is not true, we cannot say anything about any solutions of $\mathcal{F}_n$, however, since the space is $[0,1]^p$ from assumptions **(A1)**, we know that the distance between true and estimated split is at most 1. We write:

$$E\left[|c_n^j(p) - c^j(p)|\right] = E\left[|c_n^j(p) - c^j(p)||\Omega\right] P(\Omega) + E\left[|c_n^j(p) - c^j(p)||\bar{\Omega}\right](1 - P(\Omega))$$
$$\leq \frac{2k(n)}{n+1} p_1^n p_2^n p_3^{n,k(n)} + 1 \times (1 - p_1^n p_2^n p_3^{n,k(n)}). \tag{2.5}$$

Additionally, we have seen that through their binomial structure, each of the p_i^n, $i \in \{1,2\}$ can be bounded by $(1 - C_i \exp(-\alpha_i n))$, and by bounding the infinite sum by the integral, we also have directly that $p_3^{n,k(n)}$ can be bounded by $(1 - C_3 \exp(-\alpha_3 k(n)))$, $\forall n \in \mathbb{N}$ for some positive constants C_i and α_i, $i \in \{1,2,3\}$ that depend on ρ and δ.

As a result, $\exists\ C,\ \alpha > 0$, s.t. $\forall n \in \mathbb{N},\ p_1^n p_2^n p_3^{n,k(n)} \leq (1 - C\exp(-\alpha k(n)))$. By setting $k(n) = \frac{\log(n)}{\alpha}$, which satisfies both $k(n) \to \infty$ and $\frac{k(n)}{n} \to 0$, we get that $p_1^n p_2^n p_3^{n,k(n)} \leq (1 - \frac{C}{n})$. From Equation 2.5, we conclude that $E\left[|c_n^j(p) - c^j(p)|\right] = \mathcal{O}\left(\frac{\log(n)}{n} + \frac{1}{n}\right) = \mathcal{O}\left(\frac{\log(n)}{n}\right)$.■

The exact values of p_1^n, p_2^n, $p_3^{n,k}$ can be computed if we know the data generation process, the true underlying extended tree, and p_{min}. The results and the convergence hold for any $\delta > p_{min} > 0$, but the higher this p_{min}, the faster the convergence. A good p_{min} can be easily obtained with cross-validation. With the exact same method, similar results can be obtained for the variance of the XSTrees estimator and for how the "closeness" of the estimator to the true tree translates to in-sample and out-of-sample prediction error. The XSTrees accurately evaluates the number of splits and their average positions with high probability, while accounting for variability and uncertainty in the data by introducing a standard deviation on these positions. Thanks to its distributional nature, it also introduces a continuum in the predictions, which are purely discrete for single decision trees.

Last but not least, even in the absence of assumptions **(A1)**-**(A3)**, we propose an alternative version of XSTrees, HXSTrees (Honest XSTrees, Algorithm 8) where splits in different dimensions follow a certain structure and the training data is randomized. We establish that the HXSTrees is asymptotically consistent even outside of **(A1)**-**(A3)**. HXSTrees has the property that for any test sample x, the tree distribution is

independent of x. This is ensured by splitting the data into two independent halfs: one is used for tree generation, and the other is used for out-of-sample prediction. We present the corresponding algorithm, the theorem and the proof for the HXSTrees in §A.1 of the Appendix.

2.5 Computational Experiments

We test the XSTrees algorithm on (i) publicly available data, (ii) synthetic data and (iii) real-world case studies. We show on both classification and regression tasks that XSTrees significantly out-performs the state-of-art models in terms of out-of-sample prediction accuracy, while being more robust to noise and requiring a relatively low amount of data to train. We also argue that the XSTrees provides more interpretability than alternative ensemble methods through several examples.

Remark 5 (Note on implementation) *Our code for the XSTrees is open source, and all the hyper-parameters used for these experiments can be found in §A.2 of the Appendix. Also note that we have carefully implemented pruning strategies (e.g., by setting a maximum depth, a minimum number of samples per split, or a minimum number of samples per leaf for both the learning of the trees and the prediction, see Appendix §A.2), and regularization heuristics (see §2.3.4) to avoid over-fitting.*

First, we test the XSTrees algorithm on the UCI Publicly Available Datasets (see [71]).

2.5.1 Classification Benchmark

We benchmark the algorithm against a Baseline Model (most frequent label in the training data), Linear Models (LIN - [118]), Support Vector Machines (SVM - [148]), K-Nearest Neighbors (KNN - [3]), Classification and Regression Trees (CART - [39]), AdaBoost (ADA - [135]), Feed-Forward Neural Networks (FNN - [107]), LightGBM (LGBM - [105]), Random Forest (RF - [40]), and Extreme Gradient Boosting Trees (XGBoost - [49]) on 20 popular classification datasets from UCI Machine Learning

Public Repository. We do not benchmark the algorithm against Optimal Trees ([22]) because of (i) reproducibility, as the Optimal Tree implementation is a proprietary software, and (ii) because the base learner for all the tree methods above is CART, and can be itself replaced by an Optimal Tree for an apple-to-apple comparison.

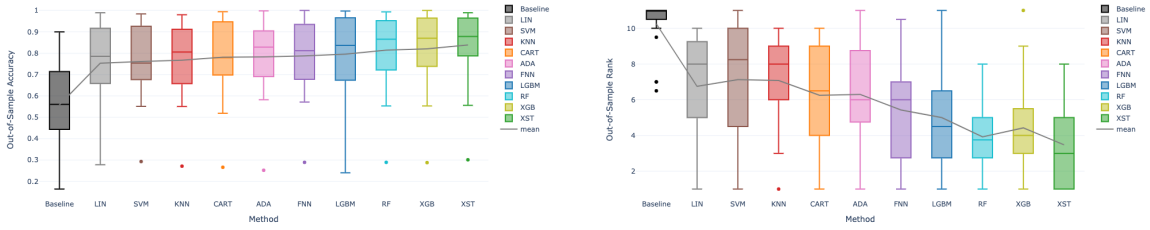
These datasets were randomly chosen with the number of observations going from 100 to 20,000 and with number of features going from 3 to 1,000, including text, categorical variables, integer variables and continuous variables. Categorical variables are transformed to binary variables for which the XSTrees algorithm is identical, with the exception that at most one split can exist on the corresponding dimension, and it has to be at value 0.5. No data imputation is performed and missing rows are removed, and all the algorithms are hyper-parameter tuned on each dataset (see §A.2 in the Appendix for all the details). 75% of the data is assigned to the training set. Each test is repeated 5 times with a different random seed, and the following results are the average of the out-of-sample accuracy (in terms of correctly classified test set points) results for these tests (See Table 2.1 and Table 2.2 below for more details). The standard deviation of these results can be found in §A.3 in the Appendix. We also report the overall rank ordering of the XSTrees algorithm in comparison to the other methods.

Dataset	Baseline	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST	XST Rank
Abalone	0.166	0.278	0.293	0.271	0.266	0.252	0.289	0.240	0.289	0.288	**0.301**	**1**
Adult	0.763	0.802	0.757	0.803	0.807	0.864	0.795	**0.874**	0.863	0.873	0.865	3
Anneal	0.765	0.989	0.914	0.874	**0.994**	0.906	0.937	0.984	0.992	0.988	0.988	4.5
Car	0.695	0.935	**0.984**	0.921	0.942	0.900	0.978	0.977	0.954	0.977	0.969	5
Contraception	0.423	0.517	0.557	0.550	0.519	**0.582**	0.581	0.556	0.553	0.579	0.556	5.5
Credit Approval	0.548	0.851	0.652	0.741	0.737	0.843	0.854	0.833	**0.868**	0.859	0.863	2
Banknote Authentification	0.567	0.989	0.949	0.955	0.986	0.998	**1.000**	0.997	0.993	0.987	0.973	8
Haberman	0.733	0.752	0.717	**0.761**	0.739	0.734	0.742	0.713	0.752	0.722	0.728	8
Heart Disease	0.600	0.678	0.613	0.667	0.665	0.777	0.805	0.811	0.833	0.827	**0.843**	**1**
Congressional Vote	0.632	0.639	0.700	0.636	0.625	0.598	0.610	0.625	0.656	0.652	**0.701**	**1**
Iris	0.300	0.898	0.920	0.980	0.980	0.920	0.980	0.980	0.980	**1.000**	0.989	2
Chess	0.528	0.969	0.979	0.939	0.992	0.960	**0.993**	0.992	0.987	0.990	0.982	6
King Rook vs. King	0.164	0.328	0.551	0.649	0.718	0.647	0.622	0.713	0.691	0.881	**0.886**	**1**
Magic04	0.649	0.789	0.753	0.808	0.820	0.841	0.818	**0.878**	0.877	0.867	0.871	3
MONK-01	0.415	0.754	0.754	0.816	0.804	0.779	0.757	0.634	0.878	**0.902**	0.892	2
MONK-02	0.554	0.554	0.725	0.569	0.678	0.619	0.571	0.625	0.660	0.553	**0.895**	**1**
MONK-03	0.463	0.829	0.932	0.902	0.951	0.903	0.906	0.923	0.951	0.951	**0.959**	**1**
Part Failures	0.899	**0.966**	0.935	0.922	0.910	0.956	0.932	0.955	0.923	0.940	0.936	5
Sonar	0.551	0.783	0.810	0.826	0.753	0.816	**0.840**	**0.840**	0.826	0.811	0.820	5
Transfusion	0.754	**0.766**	0.733	0.754	0.744	0.755	0.733	0.758	0.758	0.754	0.755	4.5
Average	*0.558*	*0.753*	*0.761*	*0.767*	*0.782*	*0.783*	*0.787*	*0.795*	*0.814*	*0.820*	***0.839***	***3.475***
Median	*0.560*	*0.786*	*0.754*	*0.805*	*0.779*	*0.829*	*0.812*	*0.837*	*0.866*	*0.870*	***0.879***	***3***

Table 2.1: Mean Out-of-Sample Accuracy for the UCI Classification Datasets. In bold, the top-performing algorithm for each row. Ranks of the XSTrees are reported in the last column.

Dataset	Baseline	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST
Abalone	11	6	2	7	8	9	3.5	10	3.5	5	**1**
Adult	10	8	11	7	6	4	9	**1**	5	2	3
Anneal	11	3	8	10	**1**	9	7	6	2	4.5	4.5
Car	11	8	**1**	9	7	10	2	3.5	6	3.5	5
Contraception	11	10	4	8	9	**1**	2	5.5	7	3	5.5
Credit Approval	11	5	10	8	9	6	4	7	**1**	3	2
Banknote Authentification	11	5	10	9	7	2	**1**	3	4	6	8
Haberman	7	2	10	**1**	5	6	4	11	3	9	8
Heart Disease	11	7	10	8	9	6	5	4	2	3	**1**
Congressional Vote	7	5	2	6	8.5	11	10	8.5	3	4	**1**
Iris	11	10	8.5	5	5	8.5	5	5	5	**1**	2
Chess	11	8	7	10	2.5	9	**1**	2.5	5	4	6
King Rook vs. King	11	10	9	6	3	7	8	4	5	2	**1**
Magic04	11	9	10	8	6	5	7	**1**	2	4	3
MONK-01	11	8.5	8.5	4	5	6	7	10	3	**1**	2
MONK-02	9.5	9.5	2	8	3	6	7	5	4	11	**1**
MONK-03	11	10	5	9	2	8	7	6	3.5	3.5	**1**
Part Failures	11	**1**	6	9	10	2	7	3	8	4	5
Sonar	11	9	8	3	10	6	**1.5**	**1.5**	4	7	5
Transfusion	6.5	**1**	10.5	6.5	9	4.5	10.5	2.5	2.5	8	4.5
Average	*10.25*	*6.75*	*7.125*	*7.075*	*6.25*	*6.3*	*5.425*	*5*	*3.925*	*4.425*	***3.475***
Median	*11*	*8*	*8.25*	*8*	*6.5*	*6*	*6*	*4.5*	*3.75*	*4*	***3***

Table 2.2: Accuracy Rank for the UCI Classification Datasets. In bold, the top-performing algorithm for each row. For consistency, in case of a draw, the average rank is taken: for example, if two methods both achieve the 3rd best accuracy, their rank is 3.5.



(a) Out-of-Sample Accuracy

(b) Out-of-Sample Rank.

Figure 2-5: Boxplots for the Out-of-Sample Results (Accuracy and Rank) for the Classification Benchmark.

The results can also be summarized in Figure 2-5 (for both accuracy and rank). We make the following observations: the XSTrees algorithm takes the *top spot* (i.e. is the best-performing method) on 30% of the datasets (6 out of 20), more than any other method. The second best methods for this metric are LGBM and FNN who take the top spot on 15% of the datasets (3 out of 20). Additionally, XSTrees has an average of 0.839 out-of-sample accuracy, in front of XGB and RF with respectively 0.820 and 0.814. The 4th best method on average is the LGBM method, further behind with 0.795 average out-of-sample accuracy. To account for the variability within

each dataset, we also look at the average rank across all tasks. XSTrees achieves on average a rank of 3.475 against the 10 other benchmark methods. The second most consistent method is the RF method with 3.925 average rank and the XGB method with 4.425 average rank. Similarly, to account for outliers in the results, we repeat this comparison for the median out-of-sample accuracy and the median rank, and obtain very similar conclusions, with the XSTrees ranking first on both metrics, with a median accuracy of 0.879 and a median rank of 3. Last but not least, XSTrees also has the lowest standard deviation in terms of out-of-sample accuracy (0.169) and the smallest interquantile range, i.e. difference between the 75% percentile and the 25% percentile in the boxplot for the accuracy (0.158), proving the wide applicability of the method. However, it only ranks 3rd in terms of standard deviation of the rank (2.342), behind the baseline (1.526) and the RF algorithm (1.771), with a relatively high interquantile range for the rank (4.0), but it is the only method where rank 1 is below the 25% percentile. We also highlight one of limitations of the XSTrees method through the Banknote Authentication dataset, where XSTrees achieves its worst rank (8th out of 11), suggesting that the distributional component of the XSTrees is not enough to bridge the gap between tree-based methods and Neural Networks (FNN ranks 1st for this dataset) for computer vision applications. These findings, as well as the full summary of the benchmark can be found in Table 2.3.

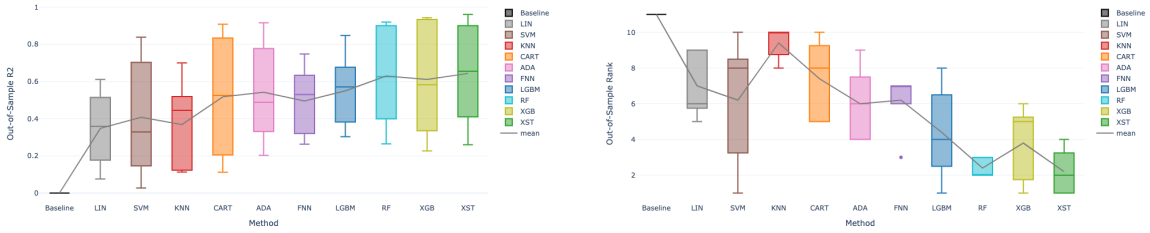
Metric	Baseline	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST
Number of Top Spots	0 (11)	1 (7.5)	1 (7.5)	1 (7.5)	1 (7.5)	1 (7.5)	3 (2.5)	3 (2.5)	1 (7.5)	2 (4)	**6 (1)**
Average Accuracy	0.558 (11)	0.753 (10)	0.761 (9)	0.767 (8)	0.782 (7)	0.783 (6)	0.787 (5)	0.795 (4)	0.814 (3)	0.82 (2)	**0.839 (1)**
Average Rank	10.25 (11)	6.75 (8)	7.125 (10)	7.075 (9)	6.25 (6)	6.3 (7)	5.425 (5)	5.0 (4)	3.925 (2)	4.425 (3)	**3.475 (1)**
Median Accuracy	0.56 (11)	0.786 (8)	0.754 (10)	0.805 (7)	0.779 (9)	0.829 (5)	0.812 (6)	0.837 (4)	0.866 (3)	0.87 (2)	**0.879 (1)**
Median Rank	11.0 (11)	8.0 (8)	8.25 (10)	8.0 (8)	6.5 (7)	6.0 (5)	6.0 (5)	4.5 (4)	3.75 (2)	4.0 (3)	**3.0 (1)**
Standard Deviation of Accuracy	0.196 (10)	0.205 (11)	0.176 (3.5)	0.173 (2)	0.183 (7)	0.176 (3.5)	0.182 (6)	0.193 (9)	0.178 (5)	0.184 (8)	**0.169 (1)**
Interquantile Range of Accuracy	0.251 (10)	0.239 (8)	0.235 (6)	0.245 (9)	0.236 (7)	0.192 (2)	0.228 (5)	0.267 (11)	0.215 (4)	0.212 (3)	**0.158 (1)**
Standard Deviation of Ranks	**1.526 (1)**	3.08 (10)	3.312 (11)	2.364 (4)	2.84 (7)	2.731 (6)	2.966 (8)	3.022 (9)	1.772 (2)	2.617 (5)	2.342 (3)
Interquantile Range of Ranks	**0.25 (1)**	4.125 (9)	5.25 (11)	3.0 (4)	4.5 (10)	3.75 (6)	3.875 (7)	3.375 (5)	2.125 (2)	2.25 (3)	4.0 (8)

Table 2.3: Summary of the UCI Classification Benchmarks. Between parentheses, how each method ranks for the corresponding metric. In bold, the top-performing algorithm for each metric.

2.5.2 Regression Benchmark

We perform a similar benchmark and analysis on regression tasks on 5 popular datasets from the same UCI Machine Learning Public Repository. We report the

results in terms of out-of-sample coefficient of determination (R^2), and in terms of ranks in Table A.2 and Table A.3. Note that we take as a baseline model the average value of the target variable in the training data, which means the R^2 of the baseline model is approximately 0, so we ignore it in the rank comparison.



(a) Out-of-Sample R^2 (b) Out-of-Sample Rank.

Figure 2-6: Boxplots for the Out-of-Sample Results (R^2 and Rank) for the Regression Benchmark.

We further summarize these results in Figure 2-6. We find extremely similar results to the classification benchmark, with the XSTrees having the highest average out-of-sample R^2 (0.643), the highest median out-of-sample R^2 (0.655), the highest average rank (2.2) and the highest median rank (2, placed equal first with RF). It is also very consistent, with a standard deviation of 1.303 for ranks, but a relatively high standard deviation for R^2 (0.290), coming mainly from the wide difference between the considered regression tasks. The full analysis can be found in Table 2.4.

Metric	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST
Number of Top Spots	0 (8)	1 (3)	0 (8)	0 (8)	0 (8)	0 (8)	1 (3)	0 (8)	1 (3)	**2 (1)**
Average R^2	0.348 (10)	0.408 (8)	0.369 (9)	0.518 (6)	0.542 (5)	0.495 (7)	0.55 (4)	0.63 (2)	0.611 (3)	**0.643 (1)**
Average Rank	7.0 (8)	6.2 (6.5)	9.4 (10)	7.4 (9)	6.0 (5)	6.2 (6.5)	4.4 (4)	2.4 (2)	3.8 (3)	**2.2 (1)**
Median R^2	0.359 (9)	0.329 (10)	0.445 (8)	0.525 (6)	0.488 (7)	0.53 (5)	0.571 (4)	0.626 (2)	0.583 (3)	**0.654 (1)**
Median Rank	6.0 (5.5)	8.0 (8.5)	10.0 (10)	8.0 (8.5)	6.0 (5.5)	7.0 (7)	4.0 (3)	**2.0 (1.5)**	5.0 (4)	**2.0 (1.5)**
Standard Deviation of R^2	0.212 (3)	0.335 (9)	0.249 (4)	0.347 (10)	0.284 (6)	**0.196 (1)**	0.209 (2)	0.284 (5)	0.323 (8)	0.291 (7)
Interquantile Range of R^2	0.271 (3)	0.472 (8)	0.333 (4)	0.573 (10)	0.357 (5)	0.257 (2)	**0.211 (1)**	0.45 (7)	0.56 (9)	0.421 (6)
Standard Deviation of Ranks	1.871 (5)	3.633 (10)	0.894 (2)	2.302 (8)	2.121 (6)	1.789 (4)	2.702 (9)	**0.548 (1)**	2.168 (7)	1.304 (3)
Interquantile Range of Ranks	3.0 (6.5)	4.0 (9.5)	1.0 (2.5)	4.0 (9.5)	3.0 (6.5)	**0.0 (1)**	3.0 (6.5)	1.0 (2.5)	3.0 (6.5)	2.0 (4)

Table 2.4: Summary of the UCI Regression Benchmarks. Between parentheses, how each method ranks for the corresponding metric. In bold, the top-performing algorithm for each metric.

In conclusion, XSTreees scale to reasonably-sized datasets, can be applied consistently to a very wide set of classification and regression tasks, and significantly outperform all other tested methods (between 2.5% and 50% higher average accuracy

for classification, and between 2% and 85% higher average R^2 for regression across all 25 supervised tasks).

2.6 Synthetic Data

The goal of this experiment is to evaluate how much data is needed to get a good level of performance for the XSTrees method compared to other tree-based methods, as well as investigate the method's sensitivity to noise.

2.6.1 Experimental Set-Up

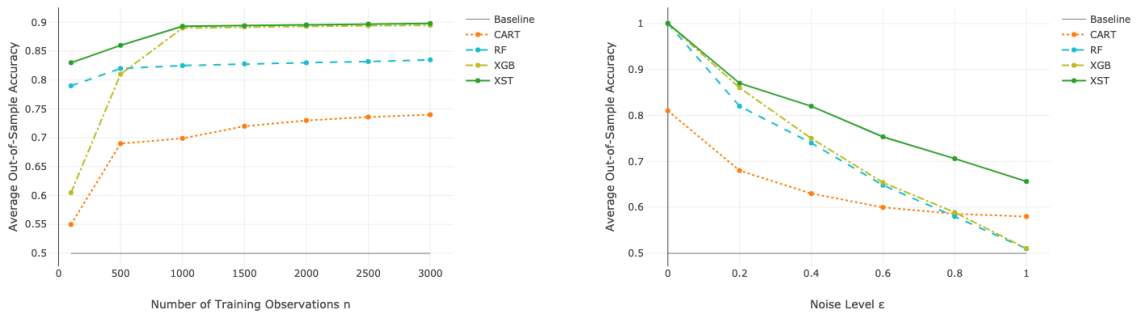
We consider the following experimental set-up: let $n \in \mathbb{N}$ be a positive integer denoting the number of observations, $p \in \mathbb{N}$, the number of features, and $\epsilon^0 \in \mathbb{R}_+$ a parameter controlling for the noise. For $i \in [n]$ denoting one observation, let $X_{i,j}$ denote the feature j of observation i, be drawn uniformly from $[-1, 1]$ and $\epsilon_{i,j}$ the noise on feature j of observation i, be drawn from $\mathcal{N}(0, \epsilon)$. This is unobserved in the data. Additionally, let $y_i = \mathbb{1}\left(\prod_{j=1}^{p}(X_{i,j} + \epsilon_{i,j}) > 0\right)$ denote the binary outcome for observation i. This is the target variable for the considered classification task.

We want to learn for any given vector $x \in [-1, 1]^p$ the probability that x is labeled 0 or 1 from this data. We compare XSTrees to the other two best tree-based methods from the benchmark: XGB and RF, which are arguably the most popular tree ensemble methods, as well as CART, which is the base learner for all three methods. We vary the size of the training data (n) and the magnitude of the unobserved noise (ϵ). Simulations are repeated 30 times for each set of parameters.

2.6.2 Results

We get the results for $p = 4$ in Figure 2-7. We notice that XSTrees and XGB significantly outperform the other methods in the asymptotic regime. For small n, XSTrees has an edge versus its competitors, achieving 4% more out-of-sample accuracy for the smallest n ($n = 100$) than the second best method in the low-data regime, RF.

In Figure 2-7(b), we also notice that XSTrees outperforms all other methods when ϵ increases. It has similar performances to XGB and RF for $\epsilon = 0$, but it outperforms them by a significant margin for $\epsilon \geq 0.4$.



(a) Out-of-Sample accuracy vs n. (b) Out-of-Sample accuracy vs ϵ.

Figure 2-7: Out-of-Sample results for the two synthetic experiments. XSTrees outperforms other benchmarks in the limited data regime and when the outcome variance is high.

In conclusion, this simulation study shows that XSTrees needs overall less data to perform well and are less sensitive to noise than alternative tree-based methods.

Remark 6 *Despite having in this case exponentially (in the number of dimensions) many leaves, the aggregation procedure of the XSTrees allows the algorithm to perform very well, since the number of splits that are being learned are in the same order of magnitude as p, and we have with the results of §2.4 that the fast convergence to the true position of these splits is ensured. It also confirms the results of §2.5, where the XSTrees outperformed its competitors in high-dimensional examples (more than 1,000 features). This is partly due to the fact that XSTrees focus on learning splits (for which the procedure scales linearly) rather than learning the leaves directly (for which the procedure scales exponentially). When it comes to correlation between features, the XSTrees are also particularly well-suited, because (i) the base learner, here CART, already does feature selection, and (ii) if splits happen on highly-correlated features, this will create redundant regions, but the predictions are ultimately likely to be correct in each of these regions separately, only impacting computational complexity and interpretability, but not predictive power.*

2.6.3 Visualizing XSTrees

The resulting XSTrees is fully characterized by the distribution on the number of splits in each dimension, and for each of these splits, the distribution of the position of the split. If $m_j,\ j \in [p]$ is the maximum number of splits in dimension j, then we can fully represent the distribution on the tree space by $\sum_{j=1}^{p} m_j$ splits c characterized by (i) the dimension of the split $c(d)$, (ii) the normal distribution of its position $c(p)$, and (iii) the probability of the split existing p directly given by the distribution on the number of splits in $c(d)$, resulting in (iv) a prediction for each point $P(y = 1)$. We illustrate the trained XSTree on the synthetic example with $p = 4$ in Figure 2-8.

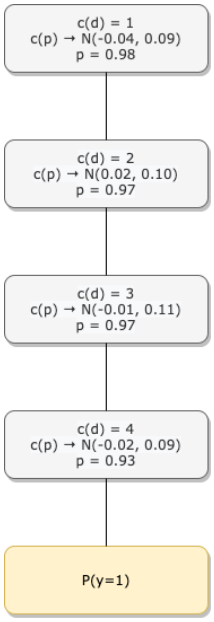


Figure 2-8: Representation of the XSTrees trained on the synthetic example. There are at most 4 possible splits, at most 1 per dimension. These splits are represented by descending order in terms of probability of existence, e.g. the first split is for dimension $c(d) = 1$, in position $c(p)$ normally distributed around -0.04 with standard deviation 0.09, and has 98% probability of existing when sampling the individual extended trees.

Example 4 *An example of sample from this tree distribution, and how a new observation $x \in \mathbb{R}^4$ is treated within this sample, is shown in Figure 2-9. Remember that these samples are what allow us to compute $P(y = 1)$ for x with Monte-Carlo simulation.*

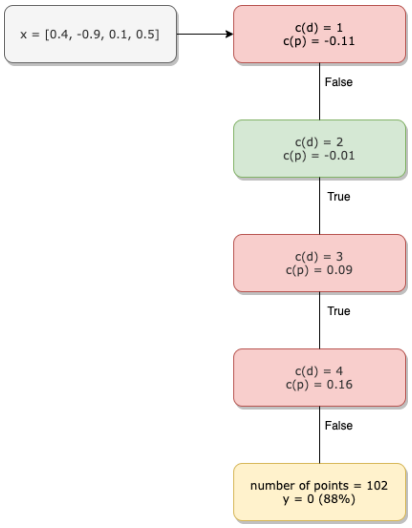


Figure 2-9: An extended tree sampled from the trained XSTrees in Figure 2-8. It behaves exactly the same as a regular decision tree, but has the significant advantage of being represented in compact form due to its symmetry (every depth of the tree has the same split, hence the representation above). In this example, the new point x satisfies $x[1] = 0.4 > -0.11$, $x[2] = -0.9 \leq -0.01$, $x[3] = 0.1 > -0.09$ and $x[4] = 0.5 > 0.16$. Hence, x is assigned to the leaf of the sampled tree satisfying the same constraints. This leaf contains 102 training observations, 88% of them being labeled 0, so we predict y = 0.

This XSTrees framework is one of the most useful interpretations of tree ensembles which allows one to generate interpretations similar to a single decision tree, while maintaining predictive performance levels of gradient boosted and bagged ensembles. We provide a visualization tool with our open-source code.

Remark 7 *Note in the case of very deep XSTrees, we can also "trim" the tree distribution by removing the possible splits that have a probability of existing smaller than some threshold. It allows us to get a low-depth approximation of the XSTrees. We can even compute, thanks to the distributional nature of the XSTrees, probabilistic metrics to quantify how good this approximation is, resulting in a high-quality trade-off between interpretability and performance. For example, given numbers of splits on each dimension, and given ranges for the positions of these splits, we can compute the probability that an extended tree within these ranges is sampled from a particular XSTrees distribution.*

Also note that we can compute the mode of this distribution, i.e. the most prob-

able tree in the XSTrees distribution. This is done by keeping the splits that have a probability of existing above 50%, and setting their positions to the mean of the corresponding normal distribution. For example, the mode tree of the XSTrees in Figure 2-8 is represented in Figure 2-10.

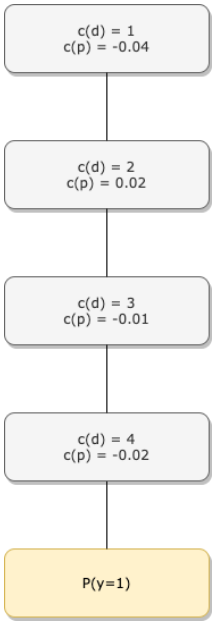
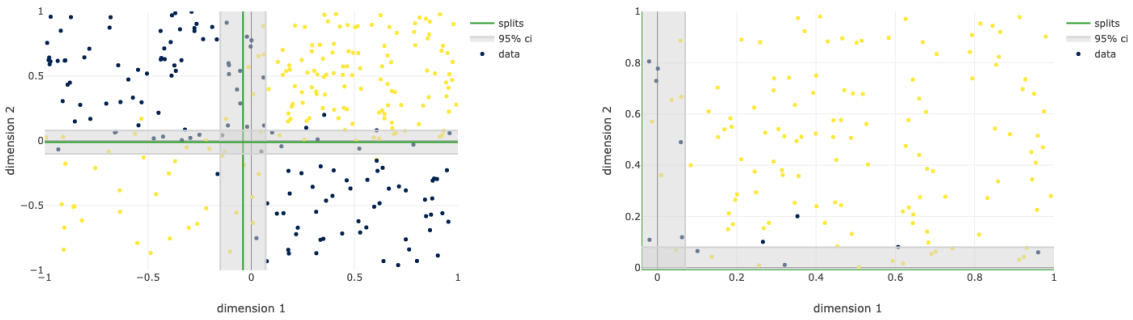


Figure 2-10: Mode of the XSTrees for the synthetic example, as shown in Figure 2-8. It is obtained by keeping only the splits with a probability of existing above 50%, and by taking the mean (which is also the mode) of the corresponding normal distribution as the position of the split. It is the most probable tree given the XSTrees distribution and behaves exactly the same as a regular decision tree, but has the advantage of being represented in compact form due to its symmetry.

2-dimension example: we can further visualize the trained XSTrees on 2 or 3 dimensions at the time. Even if the problem is higher dimension, that allows us to capture the interactions between two dimensions in particular and extract more insight from the XSTrees distribution.

Example 5 *In the previous example for $p = 2$, we obtain Figure 2-11. We can see that XSTrees captures both the true positions of the splits around 0, and two sets of uncertainties: (i) the uncertainty from the lack of data as observed in the split on dimension 1 in Figure 2-11(a) with high standard deviation (and large confidence interval), and (ii) the uncertainty from the error ϵ in the data itself as the width of the*

confidence intervals covers some of the misclassified points in Figure 2-11(b). This is one of the reasons the XSTrees performs so well, as it introduces continuity and robustness in a fundamentally discrete tree structure.



(a) Representation of the entire XSTrees. (b) Representation of a leaf node.

Figure 2-11: 2-d Representation of XSTrees splits and their 95% confidence interval.

2.7 Real-Life Case Study for an Online Retailer

In addition to testing the method on synthetic as well as publicly available data, we also investigate the performance of our method with an industry partner (a leading American e-commerce company that sells home goods) on sales forecasting, which is key to their day-to-day operations. We periodically (in this case, bi-weekly) predict the sales of each SKU based on time-series data and product features. The goal is to approximate the currently used forecasting algorithm — a proprietary concatenation of several XGBoost models — with an XSTree.

2.7.1 Experimental Set-Up

Forecasting sales is critical in managing cash flow and purchasing, but also in planning the best way to take advantages of future changes and monitoring the performances of both the SKUs individually and the company as a whole. This dataset consists of 1 million observations and 200 features, including:

- **Product Features**: e.g. price, category, color, material, quality, and user ratings.
- **Seasonality Features**: e.g. day of the month, month, big weekends, and events.
- **Time-Series Features**: e.g. past sales and past changes in price.

Our target variable y, is the 2-week ahead forecast used by our industry partner. The goal is to accurately approximate this forecast with our model that is both more stable and more interpretable. By splitting the dataset into training (75% of the data) and testing (the remaining 25%) in chronological order and not at random, and using the same hyper-parameter tuning and training procedure described in Section §A.2.

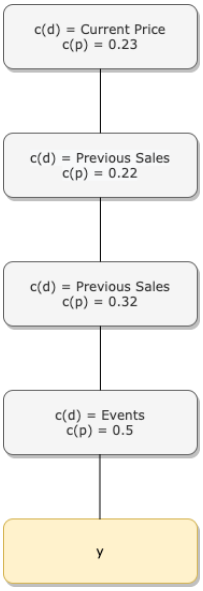
2.7.2 Results

We obtain an in-sample and an out-of-sample accuracy both equal to 99.6%, with an out-of-sample MAPE (Mean Absolute Percentage Error) of 0.2%. We create a visualization of the trained XSTrees distribution. For confidentiality purposes, the displayed values are normalized. We obtain the compact representation of the trained XSTrees and of its mode in Figure 2-12.

Remark 8 *Note that for binary features, the position of a potential split is always 0.5, and the uncertainty is only on whether the split occurs or not.*

By evaluating y for each of the 16 leaf nodes of the mode tree, we can draw the following observations: low-price popular items will sell well regardless of the season, contrary to high-cost popular items which sell more in big weekends and promotional events. We also observe that price is not as important of an indicator of future sales for unpopular items as it is for popular ones. Finally, low-price medium-popularity items benefits more than their more popular counterparts from big weekends and promotional events.

(a) XSTrees trained on the Choice Modeling Data.



(b) Mode tree.

Figure 2-12: Compact representation of the trained XSTrees distribution on the Online Retail Case Study, and mode of this distribution.

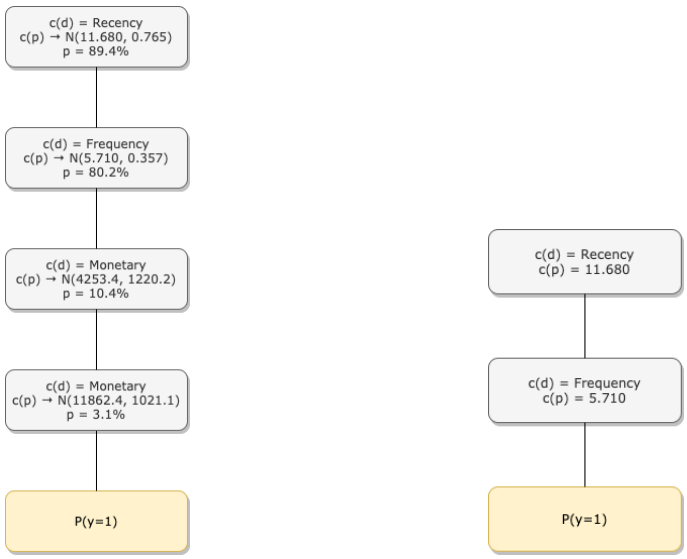
In conclusion, our XSTrees can nearly perfectly replicate the industry standards in terms of forecasting accuracy, while giving important business insights to our partner.

2.8 Blood Transfusion Case Study

We take the last dataset from the computational experiment (Blood Transfusion Service Center Dataset ([71]) in §2.5. The dataset contains 748 observations, each one corresponding to a particular blood donor, with data on: Recency (months since last donation), Frequency (total number of donations), Monetary (total blood donated in c.c.) and Time (months since first donation). The objective is to predict whether he/she is going to donate blood on a particular visit. Our model achieves an out-of-sample accuracy of (75.5%) (See Table 2.1 and Table 2.2).

We claim in this section that in addition to the main purpose of XSTrees — its state-of-the-art accuracy —, we can also extract useful takeaways from the interpretability of the model. The other two top-performing ensemble methods overall

(XGB and RF) provide little insight on why predictions are made and on the important features that capture the desired behavior, with the exception of some general black-box interpreters such as LIME ([136]). We plot the XSTrees trained for the Blood Transfusion Case Study problem in Figure 2-13(a). Figure 2-13(a) shows that the two most important features are Recency and Frequency, with the two main splits on these two dimensions occurring on average at Recency of 11.680 (did the patient donate blood or not during the last 11.680 months, i.e. in the last year approximately), and Frequency of 5.71 (did the patient donate less or more than 6 times in total). These splits have both a probability of occurring in sampled trees above 80%. We see that there are two other possible splits of occurring in this XSTrees distribution, both on Monetary, but both with a very low probability of occurring (10.4% and 3.1% respectively), and a high standard deviation. These characteristics are indicative of a feature of low importance, and a high degree of uncertainty on these splits. This can be explained by the fact that Monetary is highly correlated with Frequency, so these are more about second-order correction terms than critical splits in the problem. This results in the simple mode tree in Figure 2-13(b).



(a) XSTrees trained on the Blood Transfusion Data. (b) Mode tree.

Figure 2-13: Compact representation of the trained XSTrees distribution on the Blood Transfusion Data, and mode of this distribution.

We plot these 2 main splits in a 2 dimensional graph in Figure 2-14(a) and observe

that according to our trained XSTrees, people who did not give blood in the last 12 months are unlikely to give blood in the next visit regardless of the frequency. While if they have donated blood in the last 12 months, then if their Frequency is above 6 times, we will predict that they will give blood again with high probability. This probability decreases to about $\frac{1}{3}$ if their frequency is below the given threshold. The XSTrees method also captures the fact that there is a higher error rate (uncertainty) close to the position of the splits, within the confidence intervals, and adjusts the predicted probabilities accordingly. Figure 2-14(b) on the other hand, further shows that the information from Time and Frequency only is not enough to understand the prediction of the XSTrees, although a trend on the importance of Frequency is already appearing.

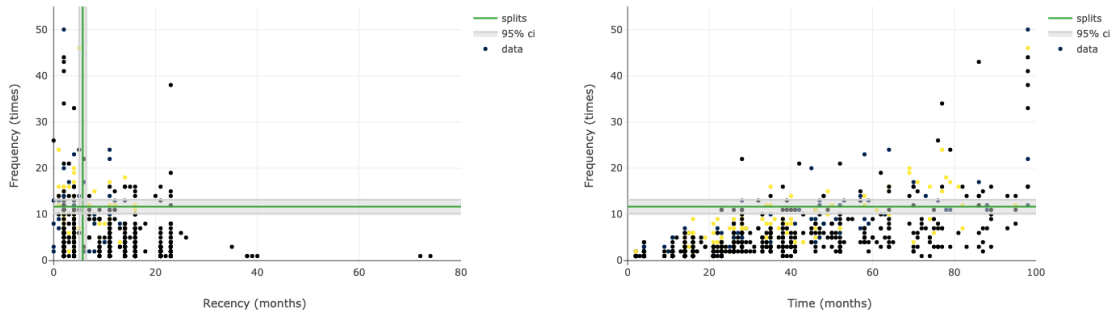


Figure 2-14: The splits of the XSTrees for Recency and Frequency (a) and Time and Frenquency (b)
Purple points denote people who donated blood and yellow points denote people who did not.

This example illustrates how we can achieve high accuracy, with a powerful ensemble model, while still being able to get some explainability on how the predictions are made.

2.9 Conclusions

The theoretical results — both asymptotic and on finite samples — and the computational experiments — on publicly available datasets, on synthetic data, and on real-world case studies —, provide strong evidence that XSTrees is extremely general,

scalable, and competitive in terms of accuracy with state-of-the-art predictive model (between 2.5% and 50% higher average accuracy for classification, and between 2% and 85% higher average R^2 for regression). It also provides a framework to explain these predictions in an interpretable way. Finally, its intrinsic distributional structure allows it to perform well when only limited training data is available as well as in a noisy environment.

Chapter 3

Ancillary Services in Targeted Advertising

3.1 Introduction

The retail industry has experienced a huge growth in terms of its online presence in the last decade. The online sector alone impacts over $1.5 trillion of total retail sales just in the United States. In fact, according to Forrester Research (2019) eCommerce will drive two thirds of retail growth by 2023. Not only have many retailers shifted to have a larger presence in the online space but new ones keep emerging as fully online retailers. This growth has further speeded-up recently due to COVID-19. This surge has allowed retailers to have access to a lot of data about their customers. This increase in availability of data has also in fact led to new business ideas in the last decade, to provide personalized recommendations and services to consumers. Apart from Amazon, other examples of online retailers who offer personalized recommendations include Stichfix (a personalized clothing styling service), Rent the Runway (a personalized rental service of designer clothing) and Wayfair (an online furniture retailer) among many others. As demonstrated by these companies, most businesses with an online presence nowadays utilize recommendation systems. In fact, these industry trends in the online sector have led to more sophisticated product recommendation systems which have been developed in order to provide the necessary competitive edge to on-

line sellers, increasing profits on the order of millions.Similarly to the retail practice, academics have also been addressing the problem of targeted product offerings in the recent years both in the marketing literature and the operations literature.

The next step to personalized product offerings that has emerged in the recent years, has been for online retailers to provide additional recommendations of ancillary services at the time of purchase. Pioneers in this idea have been businesses in the travel and hospitality industries. They provide recommendations to the traveler by offering ancillary services at the time of purchase. For example, airlines offer supplementary products to improve the traveler's experience before, during and after their ticketed trip such as VIP lounge access, priority boarding, seat upgrades, in-flight wifi. These services are offered to the traveler throughout the online purchase process. Our industry partner also offers to the consumer at the time of purchase the option to purchase an additional service. Examples of such a service include subscribing to a credit card and/or ordering assembly services for the product the customer is buying. These ancillary services are also personalized for each individual consumer. As a result, a key question to address is *which ancillary service(s) to provide to which customer at the time of purchase* with the goal to get the customer to sign up for this ancillary service. This is the question we aim to answer in this chapter and demonstrate its value in collaboration with our industry partner.

To answer this question there are two important associated issues to address: (i) determine what personalized services to offer so that they have the highest propensity for the customer to buy these additional services offered, (ii) understand what is the long term incremental revenue that will be associated with these services if offered to the customer at the time of purchase. This chapter addresses these two issues.

Industry Collaborator

In this work we collaborated with one of the largest online furniture and home goods retailers in the world. Their net revenue was over $9 billion solely in 2019. The problem we discussed above is one of the central problems faced by retailers because apart from providing their customers with millions of products to choose from,

our collaborator also offers many accompanying services aimed at alleviating home-shopping-experience anxieties. Our collaborator's service offerings span the areas of home services, professional services, design, and financing. Examples of individual services include Assembly, Warranty, Private Label Credit Card (PLCC), and Business-to-Business (B2B).

Our collaborator's services are displayed to customers on product display pages (PDPs), add-to-cart pages (ATCs), and checkout pages, as well as through intermediary pop-up screens or sidebars. This is in fact the case with many retailers. Presenting irrelevant services leads to increased cognitive burden on the customer navigating the page and increased overhead on page load times, harming the customer experience and possibly leading to lower conversion rates. However, if a retailer does not display the right services which customers are likely to sign-up for, then the retailer stands to lose a lot of revenue. Therefore, showing the right services to the right customers is an important question to answer that applies to most retailers beyond just our collaborator. In fact, we estimated that an average retailer leaves close to 4 % of additional revenue on the table if they do not address these issues. This in fact can translate to about $650-800 million in increased revenue.

When customers purchase or sign-up for these services, our collaborator generates immediate profit through revenue or cost savings. For example, in the case of the Warranty service, our collaborator collects additional revenue from the direct cost of the protection service the customer chooses, whereas in the case of the PLCC service, our collaborator benefits from cost savings on merchant fees. Beyond the immediate term, if customer experiences with these services foster customer loyalty and increase customers' likelihood of repeat purchases, the services can generate further revenue uplift for our collaborator. This longer term uplift is referred to as the "halo-effect," or incremental value, of a service. The sum of the immediate revenue generated and the incremental value of a service experience, minus the cost of service fulfilment, equals the net present value (NPV) of a service. The NPV of a service, combined with the individual customer's propensity to purchase that service should drive our collaborator's decision on what ancillary services to offer to the customer.

Contributions

This chapter aims to answer the following questions: *For each customer session, which service(s) should be presented during the online shopping experience in order to maximize the customer's propensity to buy this service?* In addition, *how can we maximize the net present value (NPV) of the service for this customer?* It is important to note that our collaborator is not the only retailer who has a similar business setting and for which this question is central to answer. In fact, one would argue that this is the central question to address for every retailer who has a presence in the online space. In order to address this question, we (i) predict a customer's probability (propensity) of signing up for a service, (ii) estimate the NPV of the service(s) the retailer offers at the individual customer level, (iii) determine the optimal combination of services to present to a customer, based on the customer's estimated NPV of engaging with a service, as well as his/her likelihood of signing up for a service at the product page.

Our solution strategies are built with the aim of providing interpretable insights to managers and stakeholders so that they can directly aid the decision making process. In doing so, we have developed methods that yield good estimation accuracy while providing insights into questions such as: *why the method may suggest that a particular customer segment has an especially high propensity of signing up for a service and what are the key factors that drive this fact?*

The chapter brings together several methods from statistics, machine learning and optimization in a holistic way but also develops a new methodology. We not only show how existing literature regarding the estimation of heterogeneous treatment effects can play an important role in this setting but also develop a new method for personalized response modeling, to devise a scalable, holistic framework that can have a considerable impact in the retail space (well beyond the question we are asking and well beyond our industry collaborator). Finally, we propose a holistic prescriptive framework that combines these estimates to recommend ancillary services to customers.

In what follows we discuss what we consider to be the main contributions of this

chapter.

1. **Predictive Cluster-While-Classify method:** One of the key goals of this work has been to determine customer sign-up propensities as a function of what services a retailer decides to show to its customers at a personalized level. A possible approach to accomplish this could be to apply a Cluster-Then-Predict method, that is, first apply a clustering method (e.g., K-means) for customer segmentation, and then apply a machine learning method (e.g., Random Forests) for the response modeling within each of the customer segments obtained. Unfortunately, this approach has one main drawback: while the customers in different segments may differ in terms of attributes (for example, age, income, and historical spending), there is no guarantee that the customers also differ in their sign-up behavior and propensities. Motivated by this limitation, we introduce a novel Cluster-While-Classify (CWC) method that performs joint clustering and classification. In fact, the Cluster-While-Classify method applies to general settings (well beyond retail) involving classification problems where there are subgroups in the data that exhibit different underlying behaviors. An important consideration when designing this method has been interpretability. As a result, we utilize a decision tree approach that allows a post-hoc interpretability. This interpretability allowed us to obtain stakeholder buy in so that service owners can understand the personalized recommendations for their customers. It is important to note that this method is general and applies to far more general settings than retail.

2. **Illustrate how this new method allows a retailer to estimate the propensity of a customer to buy a product or service offered by the retailer:** Although as discussed the method introduced in this chapter is general, we illustrate how it can help a retailer understand the probability of a customer to buy a service(s) and determine the key factors that can influence this decision. In our computations on our collaborator's data we find that CWC achieves 74% out-of-sample accuracy over 4 possible outcomes and 7 different

combinations of services for the propensity prediction. This result outperforms other popular machine learning such as random forests.

3. **Interpretable causal modeling of the long term valuation of an ancillary service:** A key objective of a retailer when showing a service to a customer, is to determine the long-term incremental value (NPV) of the service. We accomplish this through an interpretable causal model. To accomplish this goal, we leverage ideas from machine learning namely, causal forests and double machine learning. We are perhaps the first to address this NPV value estimation.

4. **Service Offering Optimization to a Customer:** Using the causal modeling and the Cluster-While-Classify method, we subsequently introduce an efficient optimization formulation that allows a retailer to personalize its service offerings to its customers and improve customer experience. We discuss the optimization formulation that maximizes not only the propensity to buy a service but also the NPV of the service as well as balances out the associated costs. Furthermore, we introduce a Linear Programming based iterative algorithm for solving this formulation. We show this algorithm determines the "right" service for the "right" customer efficiently.

5. **Approach effectiveness on a large e-commerce retailer:** We perform a study using our collaborator's data in order to establish the efficiency and interpretability of the ideas and methods introduced in this chapter. This allows us to show how the approach introduced in this chapter, gives our collaborator the capability to personalize their service offerings and improve the customer experience with a 2.5-3.5% revenue uplift. This uplift in turn translates to $80-100 million increase in revenue and $15-20 million increase in profits.

The results of this chapter will not only help retailers determine what services to display to which customer(s), it will also provide an understanding of how the NPV of a service varies across different customers, and help retailers make better-informed strategic business decisions about which services to prioritize, and as a result,

improve their service-specific marketing efforts. Furthermore, answering this question will help the retailer streamline the customer shopping experience through reduced page load times, fewer distractions, and increased personalization. By improving the shopping experience, the retailer can reap additional benefits with respect to customer retention, loyalty, and lifetime value.

3.2 Relevant Literature

This chapter explores a number of key questions facing online retailers seeking to harness the power of their clickstream data. In particular, we investigate the intersection of personalized recommendations and long-term incrementality, with a focus on interpretability. In this section, we discuss bodies of literature that relate to our work and we highlight how our contributions differ from, or build upon, these prior efforts.

To begin, we point out that aspects of our work relate to the study of targeted advertisement and personalization in ecommerce, both of which have been studied extensively in the operations management and marketing literature. With respect to targeted advertisement, areas of exploration with similarities to this chapter include how to best use clickstream data to meaningfully segment customers and how to efficiently personalize ad-serving decisions. A key component of successful targeted advertisements is having an understanding of how different types of customers respond to advertisement display decisions. [45] define a customer's utility function for soliciting information from a website and introduce a model to predict advertising click response using clickstream data gathered at the individual customer level, noting that segmenting customers improves the predictive ability of click response and identifies differences in response coefficients. Their segmentation is predetermined based on information regarding frequency of visits to the website being studied and, even with this simplistic approach, they find that multiple segment analysis predicts better than single segment analysis. In aiming to discover differentiated response to display decisions, our work also utilizes multiple segment analysis; however, we aim

to discover customer segments through a data-driven process, and we introduce the Cluster-While-Classify (CWC) algorithm for this purpose.

Another method with the same aim as CWC is the renowned Latent Class Analysis (LCA, [104]), which tries to discover underlying latent clusters in the data, and fit different response functions inside each of these clusters. The LCA often uses the Expectation-Maximization algorithm to find a solution, and while both the CWC and LCA based methods in principle use an iterative scheme to estimate model parameters, there are several core differences: (i) in the standard LCA (see [104] and [159]), the response function that is fitted within each latent class is a logistic regression. The CWC method instead supports any classification method, from logistic models to decision trees to neural networks. The iterative approach in CWC ensures that these methods remain tractable. In fact, contrary to LCA, the fitting process of the response function in CWC remains in the same complexity class as one-shot fitting the method on the entire data. (ii) assumptions made by these two methods on the input data are different. While neither LCA nor CWC assume any assumptions related to linearity or homogeneity, LCA often requires the observed class-related data to be categorical or ordinal ([127]), which CWC does not. The most popular LCA implementations (see for example [112]) enforce this assumption, breaking even further from the CWC approach. (iii) Last but not least, even on the same type of data (discrete), with the same class of response functions (logistic), the two approaches lead to different results. The CWC approach starts from a specific class assignment, which can be random or a warm start, driven by data or subject matter expertise, and converges towards a local minimum of the corresponding optimization problem. This optimization problem may have a different objective function from that of the EM approach, a different set of constraints on the classes, which LCA does not support, and depends on the starting class assignment. This leads to a different optimal solution from that of LCA. This is illustrated in both the real-world experiment and the synthetic benchmark (See §3.5), which shows that CWC, not only is different, but significantly outperforms the LCA approach.

Given the need to personalize offerings in real-time, another question being ad-

dressed in targeted advertisement literature is efficiency. Efficiency and scalability are key considerations in the display advertising framework presented by [44], which utilizes a simple machine learning framework, namely logistic regression, for modeling response predictions in display advertising and demonstrates that simple methods could outperform the then state-of-the-art models used in display advertising [2]. [63] present an algorithm to make personalized ad-serving decisions within milliseconds, using regularized logistic regression models to predict the probability of a revenue-generating user action and subsequently calculating the expected profit from displaying each ad to make an allocation decision. More recently, [10] propose a joint clustering and estimation framework for predicting future sales of new products and [5] use a tree-based approach to jointly estimate distinct market segments along with customer response modeling. Nevertheless, as is the case in this chapter, most targeted advertisement studies do not place emphasis on long-term incremental revenue in the decision making process. Our work aims to address problems of efficiency that arise in the face of decision making in an online setting, but also to incorporate not only immediate but also long-term revenue generation in the display decision. In [54], the authors note that improving personalization and reducing the intrusiveness of advertising messages can maximize marketing efficiencies, improve customer retention, and increase a firm's return on investment. [100] consider how to optimally choose the composition of the set of items offered to a customer by maximizing total sales over a given planning horizon. Although this work, like ours, aims to add to the body of research on net present value of long-term cash flows from a customer, [100] are more interested in learning a customer's preferences from several website sessions, whereas we are focused on making single session decisions based on estimates of long-term revenue that can be obtained from previous session data.

Causal relationships between online targeting decisions and revenue have seldom been explored in the literature. Recently, [9] used transactional data to estimate causal customer-to-customer trend effects in developing a personalized promotion tool. Through regularized least squares regression and instrumental variable methods they determine customer trend probability estimates, and then formulate a non-linear

mixed-integer optimization model for dynamic promotion target to maximize total expected revenue. While [9] focuses on the causal effect that one customer's purchase decision has on another, our goal in this chapter is not to investigate customer-to-customer interactions but rather instead we concentrate on the causal effect of a customer's engagement with a product or service on long-term revenue. Additionally, since in this chapter we model the long term causal impact of a service on the customer's purchase behavior through the estimation of the Net Present Value (NPV), we compare our methods to the Customer Lifetime Value (CLV) methods, which are widely used in the marketing literature. The simplest heuristic method to estimate CLV is to estimate three quantities - the average customer lifetime, the average customer revenue and the average customer interpurchase time - and predict future value based on these quantities alone. While simple, this approach requires fully observing the customer lifetime, and suffers from the limitation of not being able to capture customer heterogeneity. Another well-established method in the CLV literature is the NBD-Pareto method, initially developed in [139]. In this original model, a Pareto distribution timing model is used for modelling customer lifetime (time to customer dropout / defection is exponentially distributed with heterogeneity determined by a gamma distribution), while a negative binomial model is imposed on the number of transactions (number of transactions is Poisson distributed with heterogeneity determined by a gamma distribution). [140] extend the initial model to incorporate the dollar value of transactions rather than just modeling the number of transactions, and apply it to an industrial purchase use case. [78] propose an alternative NBD-BG model, which differs from the NBD-Pareto model in how they model customer lifetime. Instead of imposing an exponentially distribution on time to customer defection, the probability of customer defection is assumed to be geometrically distributed, with heterogeneity determined by a beta distribution. Both models are applied in a non-contractual settings, and require only two summary statistics for the purpose of estimating these parameters - the cutomer's "recency" (when the last transaction occurred) and "frequency" (how many transactions the customer made in a specified time period). However, compared to the NBD-Pareto model, the NBD-BG

model is less computationally challenging. While similar in many ways to CLV, the NPV estimation is a different problem. In the current chapter, our objective is not to estimate CLV, but instead to estimate the treatment effect of a service sign-up on the customer's spending on the platform over the next "T" months. To answer the estimation question we use a Double Machine Learning (DML) approach which lets use estimate heterogeneous treatment effects, with a large set of control variables and non-parametric model specifications. This modeling choice lets us control for not only important features that the CLV literature proposes (frequency, recency and monetary value of purchase) but many other customer-platform related features.

Another body of literature that is important to our work is the study of ancillary services. In the context of our industry partner, ancillary display services are additional services that may be offered to customers either to enhance a product (as in the case of assembly and warranty services) or may be offered independent of a particular product (as in the case of private label credit card service). The former type of ancillary display service may be considered as an "add-on" product. [98] provides a comprehensive list of add-on products that exist across multiple industries, such as: (1) Airports offering gyms to travelers during layovers; (2) Fitness centers offering amenities which are separate from basic membership, such as locker rentals, towels, and group fitness classes; (3) Hotels offering packages that combine lodging and meals or special activities; and (4) Car dealers offering accessory packages and extended warranties with the purchase of a vehicle. A large portion of add-on product or ancillary service literature focuses on pricing decisions. For example, [162] investigate heterogeneous customer choice behavior in the presence of main products and ancillary services with options of pay-per-use and subscription, with an aim to determine whether firms should sell subscriptions for ancillary services. [46] study the effect of free services on pricing and customer retention strategies for online retail by characterizing an online vendor's selection of augmenting services as a knapsack problem, and recommend that the online vendor should not only periodically reevaluate the set of services offered to satisfy the expected product requirements, but also assess the customer retention ability of the augmented product. Similar to these works, we

study the impact of add-on products and ancillary services on customer behavior. In contrast, however, we aim to develop optimal display decision strategies rather than to develop optimal pricing strategies.

Strategies developed for ancillary services share commonalities to those developed for cross-selling and product bundling. A trend of the stream of literature dedicated to cross-selling and bundling which relates, in part, to our work is the study of dynamic decision making and revenue optimization. [126] investigate the problem of dynamic cross-selling in the context of firms that can choose to offer each customer a choice between the requested product and a package containing the requested product as well as another product, referred to as a "packaging complement." The authors formulate the cross-selling problem as a stochastic dynamic program blended with combinatorial optimization, and propose heuristics to determine the packaging complement, as well as the pricing strategies. Similar to this study, we also present a solution for dynamic, as opposed to static, decisions in online retail. However, [126] focus on bundling and pricing strategies, whereas we focus on combinations of service add-ons to consider presenting to a customer. Thus, our problem may be considered as a question regarding "product framing," whereby a customer's choice is influenced by the way that products are displayed, or framed. One of the first framing-dependent models was proposed by [85], which introduces approximation algorithms for the problem of framing, assuming that consumers consider only products in the top pages, with heterogeneity across customers in terms of the number of pages they are willing to see, and that consumers select a product, if any, from these pages according to a general choice model. They prove that the problem is NP-hard and develop approximation algorithms to determine an assortment and a distribution of the products in the assortment into the different pages in order to maximize expected revenue. We also study how consumer choice is affected by the way products are framed, but we specifically aim to understand how different customer segments respond to display decisions from a company, whereas they concentrate more on the effects of where on a page a product is displayed.

Other cross-selling and product bundling studies concentrate on a different set

of constraints. [76] study the trade-off between profit maximization and inventory management, and present a personalized, discounted bundle recommendation model. In developing their model, they investigate consumers' buying propensity as well as inventory management for long-term profitability through consideration of a finite selling horizon with a fixed number of periods. They formulate the problem as a dynamic program where primary source of complexity can be attributed to the calculation of expected future revenue as a function of inventory levels and address this complexity and the need for real-time output with multiplicative and additive approximation algorithms.[60] show an efficient LP based reformulation of the assortment optimization problem with totally unimodular constraints. Similarly, [89] aim to solve the problem of personalizing an assortment of products for customers arriving at an online retailer while accounting for back-end supply chain constraints. They consider multiple products with limited inventories and consider a set of customer types, where a customer type makes a choice on which products to buy according to a general choice model and find that differentiating customer types, even just by location, can lead to significant increase in revenue. In this chapter, we also consider heterogeneous consumers with different choice models, but where the objective accounts for the negative effects of showing large assortments of ancillary services on customer conversion. We develop an efficient algorithm for the service display problem, accounting for such negative effects of displaying assortment of ancillary services.

We conclude this section by noting that an important aspect of our purchase probability and net present value estimation models is interpretability. Model transparency and understanding are critical components of the willingness of online retailers to adopt algorithm-driven decision making processes. With respect to purchase probability estimation, the algorithm we develop is adaptable in the sense that any applicable machine learning methods can be for the clustering and classification, however in this chapter we utilize highly interpretable models for each component and demonstrate their superior performance abilities. The output of the net present value estimation model in this chapter utilizes complex causal and machine learning models

and can be explained in an interpretable manner using machine learning approaches. To support our use of decision trees to approximate the incremental value estimates obtained through double machine learning and causal forests, we draw on papers that study post-hoc interpretability in machine learning. [92] provide a classification of the main problems addressed in the literature with respect to the notion of explanation and the type of black-box system. Their paper includes a comprehensive survey of methods that can be used to solve the outcome explanation problem, where an interpretable model is applied to explain the prediction of the black-box. One of the methods discussed is the "single-tree approximation," which uses a decision tree as the explanator. [68] applies this single-tree approximation method on several datasets, using a decision tree to explain the output of black-box models which were used for the original model-fitting process, such as KNN, Linear Discriminant Analysis, Multi-Layer Perceptron, AdaBoost, and Support Vector Machine. They find that maximum fidelity (accuracy with which the tree can simulate the original black-box model) can be reached with reasonable complexity and time requirements.

3.3 Problem Formulation: from Prediction to Prescription

In this section, we start by first describing the estimation of personalized customer purchase propensity for different services using the CWC method we propose. We then discuss the estimation of NPV of different ancillary services using machine learning. Finally, we discuss how to convert these predictions to prescribe optimal personalized ancillary services to customers through an iterative Linear Programming method we introduce.

We start by introducing some notation that will be used throughout the chapter. Let $K \in \mathbb{N}$ the total number of ancillary services. We consider a customer-platform session with features $X \in \mathbb{R}^d$, when $\mathcal{S} \subset \{1, .., K\}$ ancillary services are offered, which led to the sign-up of ancillary service $y \in \mathcal{S} \cup \phi$, note that ϕ implicitly denotes that the

customer left without selecting any service. The platform has access to data from n sessions that are compactly denoted as $\mathcal{D} = (\mathbf{Y}, \mathbf{X}, \mathcal{S})$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the matrix of session features, $\mathbf{Y} \in \mathbb{R}^n$ of outcomes and $\mathcal{S}_i$ is the set of services offered in the i^{th} service session. Let $f_{SSP}(Y|X, \mathcal{S})$ denote the expected service sign-up function that predicts the sign-up outcome (Y) of a session with features X when a set of $\mathcal{S}$ services is offered. Also let NPV_i denote the Net Present Value of service i, $\forall i = 1, .., K$. The platform's objective is to use the available data set $\mathcal{D}$ to estimate f_{SSP} and the NPV values of each service in order to optimize future service offerings for each session. We discuss both these estimations next.

3.3.1 The Cluster-While-Classify Method

In this section we introduce a new method for joint clustering and classification. In the retail setting this method allows us to determine the probability a customer will buy a service if presented to them. This is a fundamental problem to address in the retail space. A key step is to determine customers with similar features but who also exhibit similar buying behavior. To accomplish this, we first present a nonlinear mixed integer optimization formulation for joint clustering and classification. As the problem is not tractable, we present an iterative algorithm for solving it. Finally, we discuss a post-hoc interpretability technique using decision trees.

We start by discussing the estimation of the service sign up probability function (f_{SSP}). Consider the classical multinomial logistic framework which posits that for any feature vector X and any service $i \in \mathcal{S}$

$$f_{SSP}(Y = i|\beta, X, \mathcal{S}) = \frac{\exp(-\beta_i^\top X)}{1 + \sum_{j \in \mathcal{S}} \exp(-\beta_j^\top X)}\,, \tag{3.1}$$

where $\beta = [\beta_1, .., \beta_k] \in \mathbb{R}^{K \times d}$ are class specific parameters that are estimated with data. In particular, let $\hat{\beta}(\mathcal{D})$ denote the Maximum Likelihood Estimator of the problem parameters and $\hat{f}_{SSP}$ denote the corresponding probability estimate obtained by replacing β with $\hat{\beta}$ in the definition of f_{SSP}. Then, given $\mathcal{D}$, we can use $\hat{\beta}$ to indeed predict the purchase behavior of any customer. Unfortunately, this modeling

approach assumes that the response function is homogeneous across different sessions. Nevertheless, for ancillary services, customers can have distinct purchase behaviors. To incorporate more complex models, we can consider a clustering approach that estimates $L \in \mathbb{N}$ distinct customer response functions. A popular strategy to estimate such heterogeneous responses is to first cluster the data into different segments (based on session features) and then estimate individual models for each of these clusters. In this chapter, we take this approach one step further by *jointly* clustering and estimating L different response functions. In particular, we solve the following nonlinear mixed integer optimization problem:

$$\min_{z,p,\hat{\beta}} \quad -\frac{1}{N}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{k=1}^{K} z_{il}\delta_{ik}\log p_{ik,l} \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{k=1}^{\ell} z_{ik} = 1, \quad i = 1,\dots,n \tag{3.2b}$$

$$p_{ik,l} = \hat{f}_{SSP}(Y = k \mid \hat{\beta}(\hat{\mathcal{D}}_l), X_i, \mathcal{S}_i) \quad \forall i = 1,\dots,n; k \in \mathcal{S}_i; l = 1,\dots,L \tag{3.2c}$$

$$z_{ik} \in \{0,1\}, \quad i = 1,\dots,n; \quad k = 1,\dots,\ell. \tag{3.2d}$$

In the formulation above, $z_{il} \in \{0,1\}$ is a binary decision variable indicating whether observation i belongs to cluster l, δ_{ik} is a binary variable indicating whether observation i belongs to class k, and $p_{ik,l}$ is the probability estimate that observation i belongs to class k based on the model in cluster l. Finally, $\hat{\mathcal{D}}_l = \{i : z_{il} = 1\}$ denotes the set of observations assigned to cluster l. Notice that by construction, $\cup_{i=1}^{L}\hat{D}_i = \mathcal{D}$. For simplicity we denote the objective of the optimization problem in (2) as $\mathcal{L}(z,p)$. We ignore the coefficient $\hat{\beta}$ in this notation as $\hat{\beta}$ is determined by p. Note that δ is data directly obtained from Y and is not a decision variable.

Solving this problem to optimality is computationally hard, hence, we propose an iterative, approximate algorithm to solve this problem. The proposed iterative method described below is based on a simple idea: fixing cluster assignments z to some warm start $\hat{z}$, the problem of minimizing $\mathcal{L}(\hat{z},p)$ can be accomplished by estimating L distinct multinomial logit functions from data points assigned to the L clusters.

Vice versa, for a fixed p, we can optimize for z by assigning every observation to a cluster that leads to the minimum negative log likelihood for that observation. We present the Cluster While Classify (CWC) algorithm next.

CWC Iterative Algorithm: Given L clusters as input, T iterations,

1. Initialize the cluster assignment $\hat{z}_{il}^{(0)}$ either randomly or through an appropriate warm-start (possibly derived from a previous clustering process).

2. For iteration $t = 1, \dots T$:

   (a) For each cluster $l = 1, \dots L$, fit a multinomial logistic regression model to estimate $\hat{\beta}(\hat{\mathcal{D}}_l^{t-1})$, which minimizes the cluster-wise logistic loss based on the existing cluster assignment, $\hat{z}_{il}^{(t-1)}$. The cluster-wise logistic loss for cluster l is defined as:

   $$\mathcal{L}_l(z, p) = -\frac{1}{\sum_{i=1}^{N} \hat{z}_{il}^{(t-1)}} \sum_{i=1}^{N} \sum_{k=1}^{K} \hat{z}_{il}^{(t-1)} \delta_{ik} \log p_{ik,l} \,,$$

   and $p_{ik,l}$ is a function of the estimate parameters $\hat{\beta}(\hat{\mathcal{D}}_l^{t-1})$ that needs to be optimized (see eq. (3.1)).

   (b) For each data point $i = 1, \dots N$, update the cluster assignment $\hat{z}_{il}^{t}$ by assigning the data point to the cluster where the model's predictions yields the lowest logistic loss. Mathematically, update the assignment by solving

   $$\min - \sum_{l=1}^{L} \sum_{k=1}^{K} z_{il}^{(t)} \delta_{ik} \log \hat{p}_{ik,l} \text{ subject to } \sum_{l=1}^{L} z_{il}^{(t)} = 1$$

   where $\hat{p}_{ik,l}$ is the probability estimate that observation i belongs to class k from the fitted model for cluster l within the current iteration t and has been estimated in step (a).

   (c) Terminate with $z_{il}^{(t)}$, $\hat{\beta}(\hat{\mathcal{D}}_l^{t-1})$ when $t = T$, or when the number of data points in the most infrequent class in any of the L clusters fall below a predetermined threshold. This alternative termination criterion is to

prevent overfitting with each cluster becoming a "pure" cluster with only one class.

We note that the number of clusters L and number of iterations T are hyper-parameters which require tuning when training the CWC model. Furthermore, we can replace the multinomial logistic loss when estimating the probability functions with other classification functions and regularize the objective with $\mathcal{L}-1$ or $\mathcal{L}-2$ regularization to avoid overfitting. In fact, in our numerical section we use $\mathcal{L}-1$ regularized objective function.

In Proposition 2, we show that the run time complexity of the algorithm is primarily driven by the complexity of estimating multinomial logistic regression. Furthermore, the algorithm is also guaranteed to converge to a cluster assignment of different observations.

Proposition 2 *The CWC algorithm has the following two properties:*

(1) If the estimation method for the multinomial logistic model runs in $O(p(n,d,L))$ polynomial time, where $p(n,d,L)$ is any polynomial of n, d, and L, then the CWC algorithm runs in $O(Tp(n,d,L)+TndL)$ polynomial time.

(2) The CWC algorithm converges, which happens when $z_{ik}^{(t)}=z_{ik}^{(t-1)}, \forall i=1,...,n; k=1,...,\ell$.

Proof: See §B.1 of the Appendix.

Finally, while the CWC procedure lets us estimate sign-up propensities of existing observations, to enable out-of-sample predictions, we can fit a secondary cluster-assignment model using the obtained cluster assignments $\hat{z}_{ik}^{t}$ as the dependent variable and features X as predictors. This model can then be used to determine the cluster membership for out-of-sample data points, and pick the right cluster-wise models for predictions. The model for estimating these clusters can be any multi-class classification function (for e.g. another multinomial logistic regression model). Nevertheless, to add interpretability to the CWC framework, we propose using a decision tree to estimate out-of-sample cluster assignment. In particular, let $\mathcal{T}(\mathcal{D})$ denote a decision

tree model that uses data set $\mathcal{D}$ to estimate the outcome of interest (either continuous or multi-class outcome). Then, we use $\mathcal{D} = [\hat{z}, X]$ to estimate an *interpretable* model of cluster assignment based on the estimate cluster assignments of the CWC algorithm on the training data. The use of a tree-based method to estimate cluster assignments provides decision rules that are easy to understand for managers.

The CWC framework is flexible because for both the cluster-wise classification models and the secondary cluster-assignment model, the modeler can choose any arbitrary machine learning model that is deemed appropriate for the task. Furthermore, the modeler can also exercise discretion over what features to use for each of the two tasks—for example, the modeler might want to use a small subset of features for the cluster-wise models for parsimony, and another customer-segmentation related subset of features for the secondary cluster-assignment model. In §3.4, we further elaborate on this flexibility as well as the interpretability of the tree-based model.

3.3.2 Interpretable Estimation of the Net Present Value of Ancillary Services

Having described the procedure for estimating purchase probabilities, we next investigate to the problem of estimating the NPV of different ancillary services. It is useful to first illustrate why the NPV estimation is a causal question and not a predictive one. To do so, we come back to the problem of our industry collaborator. Let us use the PLCC service as an example, the question we are seeking to answer is: For a given customer who signed up for the PLCC service, *how much more* is she spending with our collaborator compared to how much she would have spent with our collaborator *otherwise*? In other words, the objective in this case is to estimate the long term *causal effect* of a customer engaging with the PLCC service (*intervention*) on this customer's spending? Notice that the estimation becomes challenging since we never observe customer's spending in the alternate situation where she did not get a PLCC service – her counterfactual spending is not observed. This is the fundamental challenge in causal inference.

Causal inference is straightforward in situations where the intervention is randomly assigned. However, for ancillary services, customers *choose* to sign-up for a service, rather than being randomly assigned to services, thus introducing selection bias. In particular, there might be variables which affect both a customer's decision to sign-up for a service (the intervention) and their future spending (the outcome). With respect to these variables *confounders*, customers who sign-up for a service might be systematically different from customers who do not. For example, PLCC customers might be more credit-constrained or rebate-sensitive than non-PLCC customers, and simultaneously have lower spending capacity than non-PLCC customers. It might also be the case that customers who purchased assembly service are less price-sensitive and, consequently, more willing to spend than those who were not assembly-purchasers. Therefore, we need to control for such differences, which could otherwise confound our estimates of incremental value generated by the services.

It is a non-trivial task, however, to identify which variables should be considered confounders and to specify exactly how these variables interact with one another. This process, known as model selection, is typically a "time-consuming, tenuous and somewhat arbitrary process that can make a significant difference in the final result" of the estimated intervention effect [88]. Model selection is even harder in a high-dimensional environment (i.e., when there are many observed variables which could affect both the intervention and outcome), which is precisely the setting that we were operating in. The Double Machine Learning (DML) framework [51], which we briefly discuss below, integrates machine learning within a carefully constructed econometric framework to make model selection more principled and automated.

With respect to the NPV estimation task, another noteworthy consideration is that the incremental value generated from a given service might exhibit variability across different customer segments (*heterogeneous treatment effects*). For example, there might be one-off customers who sign-up for the retailer's credit card for the sake of the initial $40 off (that is offered in conjunction with sign-up) yet never intend to return, but also customers who sign-up for the card because they expect to be a recurrent customer in the longer term. The incremental value for the former type of

customer will likely be significantly less than that of the latter customer. Estimating heterogeneous treatment effects is challenging for two broad reasons. First, there is a risk that investigators will search for narrow subgroups that exhibit more extreme treatment effects, when these effects may be entirely spurious [55]. Second, classical approaches such as matching and kernel-based methods suffer from the curse of dimensionality and are negatively affected when irrelevant covariates are used in the matching process [161]. Without having expert knowledge about what the relevant variables are in the matching process, relying on these methods could lead to model misspecification and inaccurate estimation results.

With all of these considerations in mind, we propose using Double Machine Learning [51] and Causal Forests [161] (a modified decision-forest algorithm for causal inference) for NPV estimation. We demonstrate how these methods can be utilized to have a more principled and data-driven approach for model selection and for discovering the structure of treatment heterogeneity, should heterogeneity exist in NPV estimation. Additionally, in what follows, we will also demonstrate how these methods can be used to obtain valid asymptotic confidence intervals on the estimated average and heterogeneous treatment effects. We start by briefly introducing some more notation. We let $\tilde{Y} \in \mathbb{R}$ denote the outcome variable, T denote the treatment and $X \in \mathbb{R}^d$ denote the associated features. For example, in our problem, since we are interested in the NPV of various ancillary services, $\tilde{Y}_i^t$ for a customer i is the total spending of the customer from a total of t-days after the treatment. Similarly, T_i^j denotes whether customer i purchased ancillary service j (conditional on being offered that service). Finally, $X_i \in \mathbb{R}^d$ denote the features associated with the customer as well as potentially the session in which the purchase of the service was made. As before, we will assume that the platform has access to data set of n observations per service j: $\tilde{\mathcal{D}}_j = (\tilde{\mathbf{Y}}, \mathbf{T^j}, \mathbf{X})$.

Double Machine Learning (DML)

Consider the regression specification given by

$$\tilde{Y} = \Theta(X) \cdot T + g(X) + \epsilon\,, \tag{3.3}$$

which relates the outcome $\tilde{Y}$ with explanatory control variables X and treatment T. Here, ϵ denotes the idiosyncratic noise term with conditional mean 0 ($\mathbb{E}[\epsilon|T, X] = 0$). We are interested in $\Theta(X) : \mathbb{R}^d \to \mathbb{R}$, the heterogeneous effect of treatment T on outcome $\tilde{Y}$. If T is randomly assigned, conditioned on explanatory features X, then the treatment effect $\Theta(X)$ can be directly estimated. But treatment T can itself be a function of explanatory features X, which can be specified as

$$T = f(X) + \nu\,, \tag{3.4}$$

where $f(X) : \mathbb{R}^d \to \mathbb{R}$ models the dependence of the treatment on the explanatory features. Equation (3.4) tracks the *confounding* due to the selection of treatment based on explanatory controls instead of random assignment. The main advantage of Double Machine Learning is that it does not make parametric assumptions on functions *f*, *g* and Θ. Furthermore, a wide variety of nonparametric machine learning methods can be used instead of linear models to estimate the regression model. The framework has strong analytical guarantees and has also been widely used due to its ease of implementation ([6]).

To estimate the treatment effect, DML uses a two step procedure. In the first stage, two sets of residuals

$$\hat{Y} = \tilde{Y} - E[\tilde{Y}|X] \;\&\; \hat{T} = T - E[T|X]$$

are estimated. These are called the *first stage regressions*. Notice that $\hat{Y}$ represents the variation in outcome $\tilde{Y}$ that cannot be explained by the variation in X. Similarly, $\hat{T}$ represents the variation in outcome T that cannot be explained by the variation in

X. Then, heterogeneous treatment effects $\Theta(X)$ can be calculated by solving

$$\hat{\Theta}(X) = \arg\min_{\Theta} E(\hat{Y} - \Theta(X) \cdot \hat{T})^2 .$$

This is called the *second stage regression*. Under the unconfoundedness assumption (i.e., that there are no other unobserved confounders and all confounders have been captured under the X matrix), all the residual variation in $\hat{Y}$ must be explained by the residual variation in $\hat{T}$; which is accomplished by the final-stage regression.

Since this method makes no parametric assumptions on the functional form of $E[\tilde{Y}|X]$, $E[T|X]$, it overcomes the problem of *model selection*. Similar generality for model selection also exists for the final stage regression where we use a "cross-fitting" [51] routine to prevent overfitting and to ensure that we get asymptotically unbiased estimates of the treatment effect. While this modification is model-agnostic in general, we explain an equivalent process utilized in Causal Forests known as "honest subsampling" (as Causal Forests is our model of choice for the final-stage regression estimator) in the section below.

Causal Forests

The DML framework accommodates a large family of parametric and non-parametric model specification to estimate treatment effects. We use the causal forest model for our final stage regression. Whereas decision-forest algorithms typically aim to achieve high predictive accuracy, the ultimate objective of Causal Forests is to accurately estimate conditional average treatment effects. Therefore, additional criteria are imposed on the forest-growing procedure. For each constituent causal tree in the Causal Forest, two additional criteria are imposed on the tree-growing procedure [7]. First, "honest" subsampling on the training dataset is required: the training dataset has to be further split into two sets, with one set used for growing the tree (the growing set) and the second set used for within-leaf estimation of treatment effects (the estimation set). Each observation can only be in either the growing set or the estimation set, but not both. Second, within each leaf node, there must be a minimum

number of examples belonging to both the treatment and control groups.

[161] prove that the estimates obtained from this approach have asymptotic normality properties which implies that, with a sufficient number of data points, the estimated conditional average treatment effects have a Gaussian sampling distribution and it is possible to construct confidence intervals around the obtained point estimates. The tightness of confidence intervals increases with the number of trees within the causal forest, as it addresses the Monte Carlo variability that arises due to the subsampling routine involved in the Causal Forests procedure.

Our selection of causal forests for estimating heterogeneous treatment effects is driven by three main considerations: (i) causal forests do not impose parametric assumptions on the structure of the heterogeneous treatment effect; (ii) causal forests can estimate highly accurate treatment effects even in the high dimensional setting when the number of features are very large; (iii) causal forests provide asymptotically valid confidence intervals for heterogeneous treatment effects, estimated at an individual customer level.

Adding Interpretability to NPV Estimation : While the causal forest model is inherently non-interpretable, we again use the paradigm of post-hoc interpretability and fit a decision tree over the forest to interpret the heterogeneous treatment effects estimated by the causal forest procedure. This approach is similar to the one adopted to add interpretability to the service sign-up propensity estimation. We discuss further details in §3.4.

3.3.3 Optimizing Service Displays

Having discussed the estimation of service sign-up propensities and net present value of different services, we now discuss how to combine these two predictive tasks in a holistic prescriptive framework for recommending personalized ancillary services to customers on the product display page.

Personalized offerings are very important for the platform but at the same time complex. On one hand, we would like to avoid displaying too many services because

this leads to message saturation and increased cognitive load for the customer, potentially harming conversion. On the other, we do not want to omit the display of relevant services to the customer because this leads to lost revenue if the customer would have signed up for the service.

Let $\mathbf{S}$ denote the set of all feasible service combinations that can be offered to the customer and $\mathcal{S}_j$ denote the j^{th} set in $\mathbf{S}$. For example, if the platform has K distinct ancillary services, then $|\mathbf{S}| = \sum_{j=1}^{K} \binom{K}{j}$. Also recall that $\hat{f}_{SSP}(Y = i|\hat{\beta}, X, \mathcal{S}_j), \forall i \in \mathcal{S}_j$ denotes the estimated service sign up propensity of service i when a set of S_j ancillary services are offered to the customer with session features X and NPV_i denotes the Net Present Value of any service i. Finally, to model the tradeoff between conversion rate and cost of displaying ancillary services we let $B \in \mathbb{R}$ denote the total revenue (or profit) generated from the purchase of product on display and c_j be the impact (negative) of displaying set S_j of ancillary services on the conversion of the product. This is on account of marketing fatigue that customers experience due to extra recommendations.

The platform's objective is to maximize overall revenue by choosing a particular set of ancillary services to display. For any $\mathcal{S}_j \in \mathbf{S}$, the total expected revenue is

$$r(\mathcal{S}_j) = \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = i|\hat{\beta}, X, \mathcal{S}_j) NPV_i - c_j B \,,$$

where the first part denotes the expected revenue from purchasing the ancillary services in the offered set and the second part denotes the expected loss due to reduction in the conversion rate of the product purchase. The platform solves the following Ancillary Service Display Optimization (ASDO) problem:

$$\begin{aligned} \max_{w_1,..,w_{|\mathbf{S}|}} & \sum_{j=1}^{|\mathbf{S}|} w_j r(\mathcal{S}_j) \\ \text{s.t.} & \sum_{j=1}^{|\mathbf{S}|} w_j = 1 \\ & w_j = \{0, 1\} \; \forall j = 1, .., |\mathbf{S}| \,. \end{aligned} \tag{3.5}$$

The optimization problem formulated above seeks to determine the set of services to show to a customer at a given product display page so that it yields the highest expected revenue. The decision variable w_j is a binary variable which is 1 if the platform offers the j^{th} set of ancillary services and 0 otherwise.

Notice that for any given set of ancillary services, $r(\mathcal{S}_j)$ can be estimated using the estimated values of NPV for each service in the set $\mathcal{S}_j$ and the estimates of the propensity of purchase. It also accounts for cross service effects, by modeling the purchase propensity as a function of other ancillary services being offered to the customers. Furthermore, the sign-up propensity for services can also be a function of the product's price (by including it as a feature in X). We can also enforce business level constraints such as restricting the display of certain ancillary services with specific products by adding additional constraints to the ASDO problem.

The ASDO problem is a combinatorial problem with exponential number of decision variables (in the total number of ancillary services) and the worst case computational complexity of directly solving the ASDO problem can be exponential in the number of ancillary services. Hence it can be computationally hard to solve. In what follows, we discuss a Linear Programming (LP) based algorithm that reduces the complexity of the problem substantially. The algorithm that we propose is based on the following intuition: the ASDO problem is closely related to the revenue maximizing assortment optimization problem except for cost of c_i that captures the cost of showing larger assortments. Hence, instead of solving the ASDO problem directly, we can solve the capacity constrained assortment optimization problem with varying capacities using an LP reformulation. Finally, the optimal set of ancillary products can be found by simply comparing the optimal revenue maximizing assortments of different sizes.

In what follows, we will assume that $c_j = c|\mathcal{S}_j| \ \forall j$, where c is the per unit decrease in conversion rate due to showing a single ancillary service. That is, the negative impact of displaying ancillary services on conversion is only related to the size of the set of ancillary services. This assumption simplifies the more general ASDO problem since instead of estimating $|\mathbf{S}|$ total cost parameters (one for every feasible

assortment), we now only have to estimate a single per unit cost parameter, c.

Before we formally present the algorithm, we introduce some more notation. In particular, let $q_i = \{0, 1\}$ be a binary decision variable that is 1 if we offer service i to the customer and 0 otherwise. Then, given a vector of decisions $Q = \{0, 1\}^K$, the probability that the customer chooses ancillary service i is given by

$$\hat{f}_{SSP}(Y = i|\hat{\beta}, X, Q) = \frac{v_i q_i}{1 + \sum_{j=1}^{K} v_j q_j},$$

where $v_i = \exp(-\hat{\beta}_i^\top X),\ \forall i = 1, , K$, denotes the preference weights of service i estimated from data (see §3.3). Notice that every Q directly maps to a set of services $\mathcal{S}_j \in \mathbf{S}$.

Using the notation above, and ignoring the expected cost due to reduction in the conversion rate for showing a set of ancillary services, we consider the following cardinality constrained assortment optimization problem:

$$\begin{aligned} \max_{q_1,..,q_K} & \sum_{j=1}^{K} \hat{f}_{SSP}(Y = j|\hat{\beta}, X, Q) \cdot NPV_j \cdot q_j \\ \text{s.t.} & \sum_{j=1}^{K} q_j \leq m \\ & q_j = \{0, 1\}\ \forall j = 1, .., K\,. \end{aligned} \tag{3.6}$$

Notice that in comparison to the ASDO problem, we have reduced the number of decision variables exponentially (from exponential to linear, in the number of ancillary services). Nevertheless, the objective becomes non-linear in the decision variables and we have now added a constraint on the size of the assortment. Hence, it is not yet clear how this reformulation can result in an optimal solution to the ASDO problem and if it is at all faster.

To motivate our approach, we first focus on the ASDO-Relaxed Linear Programming (ASD-RLP) reformulation of the assortment optimization problem, which is

given by

$$ASDO^{RLP}(m) = \begin{aligned} \max_{s_0,s_1,..,s_K} & \sum_{j=1}^{K} s_j \cdot NPV_j \\ \text{s.t.} & \sum_{j=1}^{K} s_j + s_0 = 1 \\ & \sum_{j=1}^{K} \frac{s_j}{v_j} \leq m s_0 \\ & 0 \leq \frac{s_j}{v_j} \leq s_0 \ \forall j = 1, .., K \, . \end{aligned} \tag{3.7}$$

Notice that instead of considering binary decision variables, the ASD-RLP problem considers continuous variables, $s_0, .., s_K$. A solution to the LP problem above directly yields a feasible solution to the assortment optimization problem with cardinality constraints, as shown by [60]. Using this reformulation, we propose the Iterative Ancillary Service Optimization algorithm (Algorithm 3). The algorithm iteratively finds optimal assortments of maximum size $1, .., K$. Then, for any given assortment size, the cost of reduction in conversation rate is accounted for by simply subtracting the associated cost from the optimal revenue. Since the algorithm loops through all potential assortments of different sizes, it guarantees that the identified assortment will be an optimal solution to the ASDO problem. We formalize this intuition in Theorem 3.

Algorithm 3 Iterative Ancillary Service Optimization (K)

for $m \in [K]$ **do**
Solve $ASDO^{RLP}(m)$ to get the optimal assortment of maximum m services.
Let $\mathcal{A}(m)$ and $r^*(m)$ denote the optimal assortment and the optimal expected revenue from the optimal assortment.
end for
Let $ASDO^{rev}(m) = r^*(m) - c|\mathcal{A}(m)|B$ and let $m^* = \arg\max_{i=1,,K} ASDO^{rev}(i)$.
Offer $\mathcal{A}(m^*)$ to the customer.

Theorem 3 *Consider the ASDO problem with $c_i = c|\mathcal{S}|$, for any feasible assortment $\mathcal{S} \in \mathbf{S}$. Then, the Iterative Ancillary Service Optimization Algorithm (Algorithm 3) computes correctly an optimal solution to the ASDO problem.*

Proof: See §B.2 of the Appendix.

Hence, under the linearly increasing cost structure, we devise a linear programming based algorithm to solve the ASDO problem. In the more general setting when the cost associated with an assortment can be arbitrary, we use an off-the-shelf Integer Programming solver to solve the ASDO problem. While the ASDO problem can have a exponential number of decision variables (in the total number of ancillary services), in general the number of ancillary services are considerably smaller than the total products. For example, our industry collaborator sells millions of products but only a handful of ancillary services (three that we considered in the current study). Hence, even the general ASDO problem is practically tractable. We will further discuss the details of the problem and the implementation strategy in §3.4.

3.4 Approach effectiveness on a large e-commerce retailer

In this section, we test our framework with our collaborator as described in §3.3. We first estimate the propensity of each service offered by our collaborator, i.e. the probability of purchasing this service given a displayed assortment, using the CWC algorithm. We then estimate the NPV of each of these services with DML and Causal Forests. This allows us to prescribe the optimal service offering for each product, personalized for each customer. In order to accomplish these prediction tasks, we have used a wide range of data sources, including historical customer data (that we do not discuss in detail for confidentiality purposes) and static and dynamic session information. More specifically, we use the following features:

- **Past Website Interactions**: Number of Distinct Devices Used, Number of Distinct Channels leading to the Website, Number of Add-to-Carts, Number and Frequency of Visits, etc.
- **Past Purchase Behavior**: Number and Price of Purchased Products, Average Order Value, etc.

- **Static Session Information**: Marketing Visitor Type, Days Since Last Visit, Days Since Acquisition, Same Day Total Order Value, etc.

- **Dynamic Session Information**: Current Page Display, Price of SKU, Number of PDPs, etc.

3.4.1 Service Sign-Up Propensity

In the subsection, we estimate the service sign-up propensity given each possible assortment of displayed services. Before we discuss more details of the experiments based on our collaborator's data, we first note that we also benchmark the performance of the CWC method against state of the art machine learning methods in a synthetic data setting as described in Appendix B. We find that the proposed method outperforms other methods with improvements ranging between 4.4% to 8.3% in terms of out-of-sample accuracy. Following the results from the synthetic experiments, we use random forests and CWC for the study, the two best performing methods. We provide more details on the experimental section in Appendix B.5. For our collaborator's estimation problem, the target variable for this problem is the mutually exclusive outcomes of whether the customer purchases Nothing, Assembly Service, Warranty Service or PLCC Service.

While CWC is a general purpose algorithm, we adapt it to our collaborator's requirements for interpretability purposes. We used a decision tree as the cluster assignment model to have interpretable groupings of the customers based on their features and provide insight into which variables determine cluster membership. Furthermore, the CWC used for this case study deterministically assigns them with the trained decision tree. This enables a more straightforward interpretation of the clusters and of the assignment mechanism.

For the classification task within each cluster, a key consideration is being able to provide a simple way to understand how customers respond to different "Retailer's Actions", which here refer to the choice of service assortment to show to the customer. For example, for a group of customers within the same cluster, with similar response

behavior, we want to know their probability of signing up for a particular service if this is the only service offered, versus if it shown alongside several other specific services. To accomplish this, we use a multinomial logistic regression for the cluster-wise models where the only features used for classification are the seven possible Retailer's Actions. This allows our collaborator to make direct comparisons of the response behavior of different clusters. Additionally, the use of a limited number of features makes the model more easily operational and hence also more easy to commercialize for our industry partner. We highlight the trade-off between accuracy and number of features used in Table 3.3.

Another benefit of the use of multinomial logistic regression is that, should our collaborator wish to increase the complexity of the model by incorporating additional features, the linearity of the model structure will allow it to remain highly interpretable. In addition, future enhancements through incorporation of additional features would enable increased personalization at the individual customer level and will provide information on the marginal effect of customer features (e.g., age, income) on the likelihood of signing up for various services. Furthermore, to prevent over-fitting, $\mathcal{L}1$ and $\mathcal{L}2$ regularization are imposed to regularize the coefficients associated with the multinomial logistic regression, adding robustness to the model and reducing the number of features required for prediction.

Table 3.1 and Table 3.2 show respectively the service propensity modeling out-of-sample results for CWC and Random Forests (RF). We detail the meaning of these metrics in §B.4 of the Appendix. We find that the CWC method achieves extremely comparable performance to the RF model while being much more interpretable. Furthermore, we note that while our algorithm utilizes all features for the purposes of cluster assignment, the cluster-wise classification models use only seven features which correspond to which services our collaborator chooses to show. In contrast, the Random Forest model utilizes all customer features, making it much more complex.

	Precision	Recall	F1-Score	Support
None Purchased	0.809	0.651	0.722	2922
Assembly	0.671	0.888	0.764	2863
Warranty	0.757	0.691	0.722	2861
PLCC	0.651	0.605	0.627	1468
Accuracy			0.723	10114
Macro-Average	0.722	0.709	0.709	10114
Weighted Average	0.732	0.723	0.720	10114

Table 3.1: Out-of-Sample Cluster-While-Classify Performance Using Only Action Features

	Precision	Recall	F1-Score	Support
None Purchased	0.836	0.663	0.739	2922
Assembly	0.637	0.971	0.770	2863
Warranty	0.801	0.614	0.695	2861
PLCC	0.687	0.582	0.630	1468
Accuracy			0.724	10114
Macro-Average	0.740	0.707	0.709	10114
Weighted Average	0.748	0.724	0.720	10114

Table 3.2: Out-of-Sample Random Forest Baseline Model Performance Using All Features

Table 3.3 shows respectively the CWC performance obtained using all features for classification, with $\mathcal{L}1$-regularization applied for sparsity, while Table 3.4 shows that the incorporation of additional features can lead to an approximate 3% increase in performance, as measured by the Macro-Average F1-Score. We conclude that we can attain a high level of performance with CWC using only seven features, striking the right trade-off between accuracy and interpretability, while allowing our collaborator

to incorporate additional features into the cluster-wise classification models in order to increase the level of personalization that can be achieved.

	Precision	Recall	F1-Score	Support
None Purchased	0.780	0.705	0.741	2922
Assembly	0.710	0.900	0.788	2863
Warranty	0.799	0.659	0.714	2861
PLCC	0.669	0.705	0.686	1468
Accuracy			0.739	10114
Macro-Average	0.737	0.734	0.731	10114
Weighted Average	0.747	0.739	0.737	10114

Table 3.3: Out-of-Sample Cluster-While-Classify Performance Using All Features

Features Used	# Features	F1 Macro-Avg	% Improvement
Retailer's Actions Only	7	0.709	-
All, $\mathcal{L}1$-regularized	101	0.731	3.10%

Table 3.4: Cluster-While-Classify Performance as a Function of Number of Features Used

In addition to these aggregate performance metrics, we found that the CWC approach was able to provide extremely granular customer segmentation based on the decision tree structure we use for cluster assignment, and we were able to identify the most likely responses to our collaborator's actions for each cluster. To illustrate this, consider the path highlighted in red in Figure B-1 of the Appendix.

By tracing the splits leading to the leaf node boxed in red, which corresponds to assignment to cluster 2, we can discern that customers falling into this leaf node (a) were acquired by our collaborator (i.e., shared their email with our collaborator) within the past $\approx$ 9 years, (b) have visited at least one assembly-eligible product display page during their current session, (c) were email acquired more than 1 year ago, but have never purchased (corresponding to a marketing visitor type of "Lapsed

Non-Past-Purchaser," (d) have not yet clicked on any PLCC call-to-actions during their current session, (e) have no service add-ons in their cart currently, and (e) are currently viewing a product whose display price is greater than $446.23.

After using rules such as those exemplified above to segment customers into clusters, for each cluster, we can also identify the most likely response to Wour collaborator's decisions to show them different combinations of services. Figure 3-1 below shows the probabilities of various service sign-up response outcomes associated with the actions that our collaborator chooses. For example, if only the PLCC service was shown to customers in this cluster, the associated cluster-wise multinomial logit model predicts that these customers have an 88.5% probability of signing up for the PLCC service. On the other hand, if all three services (Assembly, Warranty, and PLCC) were shown to these customers, the probability of choosing the PLCC service falls to 52.8%. The interactions between our collaborator's actions and response probabilities for this cluster signify that certain customer segments can be highly sensitive to our collaborator's service display decisions. We highlight this particular cluster as an example, and note that other clusters displayed greater stability with respect to their response probabilities. For example, for any of our collaborator's display decision that includes the Assembly service, customers in cluster 3 have over a 90% probability of selecting the Assembly service, according to the cluster-wise multinomial logit model. As demonstrated by the example above, there is significant value

Response Probabilities → / **Action ↓**	*None Purchased*	*Assembly Purchased*	*Warranty Purchased*	*PLCC Purchased*
Assembly, Warranty, & PLCC Shown	0.104	0.242	0.126	**0.528**
Assembly & Warranty Shown	0.110	**0.504**	0.380	-
Assembly & PLCC Shown	0.144	0.201	-	**0.632**
Warranty & PLCC Shown	0.087	-	0.209	**0.700**
Assembly Shown	0.157	**0.788**	-	-
Warranty Shown	0.132	-	**0.859**	-
PLCC Shown	0.107	-	-	**0.885**

Figure 3-1: Action to Customer Response Matrix for Cluster 2

from employing a segment-based classification approach for service display response modeling. In addition to discovering groups of customers with similar response be-

havior, we also found that different customer segments have different sensitivities to competing service call-to-actions. These findings serve as key inputs to the optimal service presentation component of the study.

3.4.2 NPV Estimation

The goal of this section is to estimate the NPV of each service independently, using similar features as for the propensity estimation as described at the beginning of the section. The target variable for this problem becomes the t-day Post-Intervention Gross Revenue Stable (GRS) from a given customer, where $t \in \{30, 90, 180, 365\}$. We refer to the subset of customers that were offered a particular service as the cohorts. Customers that signed up for this service fall into the treatment group, and those who did not fall into the control group.

Each cohort contains between 2800 and 5000 customers in the treatment group, depending on the service. The per-cohort control groups contain anywhere from 80,000 to 300,000 observations, depending on the service and covering the year 2019.

We use the the Double Machine Learning and Causal Forests methodology as presented in §3.3 to obtain estimates for the 6-month incremental revenue generated for the PLCC, Warranty, and Assembly services. We emphasize that the following incremental revenue estimates are the "halo-effect" component of the NPV, and do not include the immediate profit generated when a customer purchases a service. For the PLCC service, we estimated the 6-month incremental revenue to be around **$200** with a 95% confidence interval of approximately **[$150, $250]**. For the Warranty service, we estimated the 6-month incremental revenue to be less than **$50** with a 95% confidence interval of approximately **[$5, $80]**. For the Assembly service, we estimated the 6-month incremental revenue to be above **$130** with a 95% confidence interval of approximately **[$100, $200**]. The exact number are not given for confidentiality purposes. In addition to obtaining aggregated point estimates for each service, we also utilized interpretable surrogate explainers to provide clarity surrounding the individual treatment effect estimates that were obtained. Figure B-2 is a visualisation of a single-tree that approximates the full causal forest utilized to estimate heteroge-

neous NPV. Customers are partitioned into different subgroups, represented by the various nodes. The darker the shade of green, the higher average NPV associated with customers belonging to that corresponding nodes.. It can be observed that approximately half of the customers (10,325 of 21,895) fall into the white-colored leaf node and have an average incremental value above $130. However, the customers who fall into the green colored leaf nodes—customers who exhibit higher engagement with our collaborator's website prior to sign-up—have slightly higher incremental values, with customers in the rightmost leaf node having an average incremental value above $150. Our collaborator analysts and service owners can use the surrogate explainers we have created for each service to inform their customer targeting efforts. For example, for the Assembly service, targeting strategy should be concentrated towards higher engagement customers, as they have the potential to provide 15% greater incremental value as compared to lower engagement customers.

We also give insight on two wider questions that are of interest to our industry partner:

1. What does the *time trajectory* of incremental value for the services look like?

2. How far into the future can we be *confident* in estimating incremental value?

Regarding the time trajectory, we created multiple models to estimate the revenue at specified time periods from the day of intervention—30, 90, 180, and 365 days. From each of these models, we could obtain a point estimate for the average incremental value up to the corresponding time period, associated standard errors, and hence confidence intervals around the estimates. As shown in Figure 3-2 below, we can see that the PLCC service has a linear trajectory over time, while the Assembly and Warranty services have an asymptotic (increase-then-plateau) trajectory over time.

Comparison of incremental value trajectories across the different services reveals a number of key insights. To begin, we found that PLCC has the highest average incremental value of the three services, with an increase in incremental value of close to $1/day. The linear trend associated with the PLCC service, as compared to the

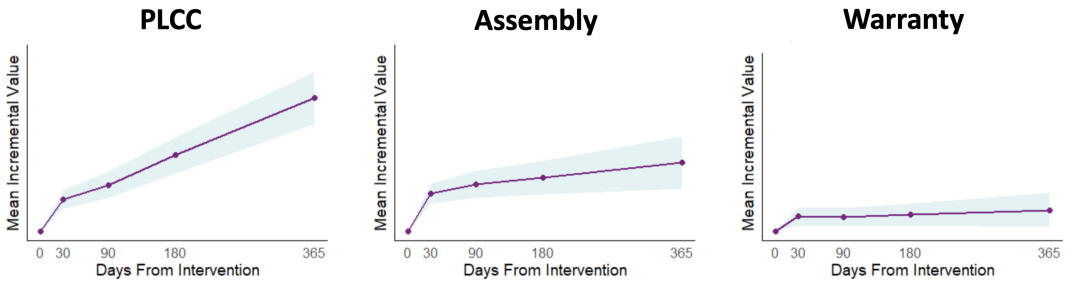


Figure 3-2: Time Trajectories of Incremental Value for 3 Services

asymptotic trends of the Assembly and Warranty services, suggests that differences in incremental value might arise from long-term engagement services versus short-term engagement services. This insight could be especially useful for our collaborator as they continue to expand their offerings by introducing new services in the future. With respect to prospective services it is important to note, of course, that profit margins must be considered in addition to incremental revenue dollar values and trends compared to lower engagement customers.

3.4.3 Optimal Service Offering Prescription

Finally, we are now in a position to transform the estimates from our NPV models and predictions from our service sign-up propensity model into prescriptions for what services to showcase to customers on the product display page.

We use the formulation described in the optimization section of §3.3. The decision variable $\mathcal{S}$ corresponds here to the display of a particular combination of services (e.g., one action could be displaying PLCC and Assembly; another could be displaying PLCC and Warranty). The set of seven possible options that our collaborator has in terms of actions are *(i) Display Assembly, (ii) Warranty, Display Assembly & Warranty, (iii) Display Assembly & PLCC, (iv) Display Warranty & PLCC, (v) Display (vi) Assembly Only, Display Warranty Only,* and *(vii) Display PLCC Only.*

Recall that $f_{SSP}(Y = i|\mathcal{S}_j)$ for product i given assortment $\mathcal{S}_j$ is the probability of sign-up estimated by our propensity model, while NPV_i is the estimated Net Present Value for this product i. The other parameters are given. We implement this for-

mulation and run a simulation on historical customer session while tracking the 3 key metrics: the estimated increase in profits and incremental 6-month revenue, the estimated percentage of session-PDPs in which we could have reduced the number of services shown and the Distribution of the new set of actions compared against the distribution of the historical set of actions. We summarize the detailed breakdown for the prescribed service offering, in comparison with the historical breakdown in Figure 3-3. We anonymize the name of the services for confidentiality purposes. The numbering in the pie chart does not correspond to the numbering of the services presented above. We observe that our optimization formulation prescribed a shift away from displaying all the services that the customer could sign-up for towards displaying individual services which our service sign-up model indicates the customer is highly likely to sign-up for. Our simulations also indicate that our collaborator can reduce the number of services displayed in 60-70% of eligible customer sessions, resulting in a significant opportunity for our collaborator to curate their service messaging efforts.

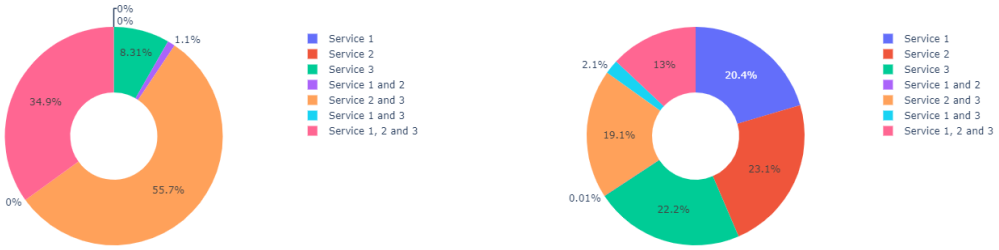


Figure 3-3: Historical (LEFT) vs Prescribed (RIGHT) breakdown of the service offering (anonymized)

Based on a simulation involving over 21,000 historical eligible customer sessions, we estimated that our collaborator would have been able to generate about 2.5-3.5% uplift in revenue, which translates to about $650-800 in increased revenue in dollar terms for these sessions. Scaling this to all eligible customers based on our collaborator's 2019 website traffic data, we would expect that we would be able to generate about $80-100 million in revenue uplift for our collaborator when this model will be fully deployed. Assuming a 20% gross profit margin, this translates to around

$15-20 million in profits.

3.4.4 Managerial Insights

The framework introduced in this chapter can be used to inform both macro and micro level business decisions. We discuss how different components of the framework (NPV estimation, customer segmentation and service optimization) can be used independently and in-tandem to inform business decisions.

- *Ancillary service management:* the NPV estimation discussed in the chapter provides a framework of leveraging platform data to estimate long term benefits of ancillary services on customer purchase behavior. Ancillary services are popular, not only in retail but also in other industries (notably, the airline industry). With the size of ancillary offerings growing in recent years, understanding which services would lead to the most bottom line benefit for the platform becomes crucial.

- *Heterogeneous customer behavior:* customers have highly heterogeneous tastes. To meet these heterogeneous needs, platforms must estimate personalized models that can guide decision making. We find that grouping customers into segments, and understanding "personalized" tastes leads to substantial improvement in customer ancillary service purchase behavior.

- *Incorporating long term effects in service display optimization:* Long term effects of service purchases can be considerable, Hence, simply optimizing for immediate reward can lead to long-term sub-optimal decisions. We propose that platforms should account for both long term revenue potential from a service, along with immediate revenue generation while optimizing for service displays.

3.5 Insights from Synthetic Experiments

While in the previous section we showed the effect of the proposed framework for our industry collaborators, in this section we discuss the performance of the proposed

framework against other benchmark methods. We first compare the CWC algorithm against other benchmark algorithms and then discuss the sensitivity of the ASDO algorithm with respect to different problem parameters. In this section, we test the accuracy of the CWC method through synthetic experiments, and extract insights on the scalability of the ASDO algorithm and its sensitivity to the maximum number of displayed ancillary services and to the sensory overload.

3.5.1 CWC Accuracy Benchmark

We perform synthetic experiments to compare the performance of CWC with other state-of-the-art benchmark methods.

Benchmark algorithms: We benchmark the CWC method against state-of-the-art predictive methods. These methods include Decision Trees ([39], CART), Latent Class Analysis ([104], LCA, with the EM algorithm) and Random Forest ([40], RF), although the later does not provide the same interpretability as the CWC method.

Data generation process: We create a dataset with $n \in \mathbb{N}$ observations, each observation represents a customer at a certain time, with particular features, as well as the features of the product shown to this customer. There are $p \in \mathbb{N}$ possible services that can be offered alongside this product, and hence the target variable is which of these services, if any, did this customer buy, making it a $p+1$-class classification task.

We generate each observation as follows: (i) We draw the customer features as binary variables that are sampled uniformly at random; (ii) service features are drawn as continuous variables drawn from a uniforms distribution $[0, 1]$; (iii) these observations are then clustered according to a clustering algorithm (for e.g. using the k-means clustering algorithm, $k \in \mathbb{R}$) based on customer features only to create customers segments; (iv) the response class is generated from a multinomial logit response function for each cluster, using only the product features from normal distributions. This allows us to create the labels for each observation. We split the final data set into a training set (75% of the observations) and a testing set (the remaining 25%). We

then train the CWC, LCA, CART and RF on the training set and evaluate the out-of-sample accuracy, i.e. the percentage of time the predictions are correct on the testing set. All hyper-parameters of the models are decided using 10-fold cross validation. We set the number of observations to be $n = 10,000$ and the number of ancillary products to be $p = 3$. The number of clusters are selected to be $k = 4$ and we split the total features into 40 customer features and 40 product features. we obtain the following results (Table 3.5 (Experiment 1)). Similarly, in Table 3.5 (Experiment 2), we present results from changing the proportion of customer features and increasing it to 60 versus 20 product features.

Out-of-Sample Accuracy	Experiment 1	Experiment 2
LCA	0.152	0.2372
CART	0.4408	0.3852
RF	0.5172	0.4408
CWC	0.605	0.506

Table 3.5: Results of the benchmark for the two experiments (Experiment 1: 40 customer features and 40 product features, Experiment 2: 60 customer features and 20 product features).

We notice that in both cases, CWC outperforms the other benchmark algorithms, with more than 7% improvement in terms of out-of-sample accuracy on average compared to RF, despite being more interpretable. We also obtain more than 12% improvement on average compared to CART. The LCA method does not perform well in these experiments due to the lack of latent variables.

3.5.2 ASDO Computational Scalability

While the number of ancillary services for our collaborator were only three, in this section we show that the proposed ASDO algorithm scales well for optimization problems with a much larger set of services. Recall that the ASDO framework solves a Linear Program p times, where p is the total number of ancillary services. Its complexity is consequently polynomial in p. We test its scalability by varying p in a

synthetic simulation similar to the previous subsection. For different values of p, we randomly sample the product utilities, revenue from the underlying product (B) is assigned value \$10 and c is fixed to be 0.1. In Figure 3-4 we present the computational time of the ASDO algorithm as we change the number of ancillary services. We find that the ASDO algorithm is able to find the optimal assortment of ancillary services in minutes, even when the number of services scale to 1000 showing the practical applicability of the model.

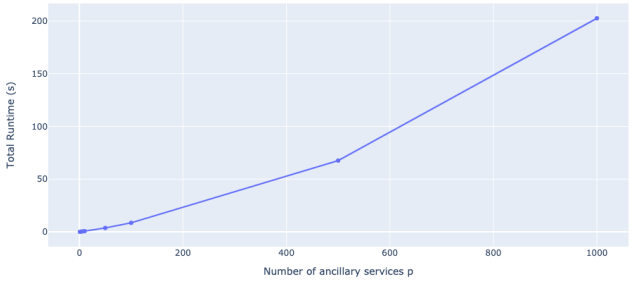


Figure 3-4: Evolution of the total run-time versus the number of ancillary services.

Both the NPV estimation, the CWC predictions and the ASDO optimization scale well for reasonably sized datasets. Our computational testing leads us to believe that our end-to-end framework is scalable.

Sensitivity to the cost parameter c: In this section, we show how our framework depends on the cost parameter parameter c, that penalizes the number of displayed products. Recall that c models the reduction in the probability of purchasing the original product as a function of the number of ancillary services. As one might expect, the expected revenue goes down for a given size assortment we increase the cost parameter c. Nevertheless, even in the case when c=0, it is not optimal to show all ancillary services. This is because adding more ancillary services reduces the probability of purchasing existing services in the assortment (follows from the MNL model). In fact, the optimal assortment in this case is of size 40 out of a potential set of 100 products.

Figure 3-5 confirms that despite having $c = 0$ and total number of ancillary

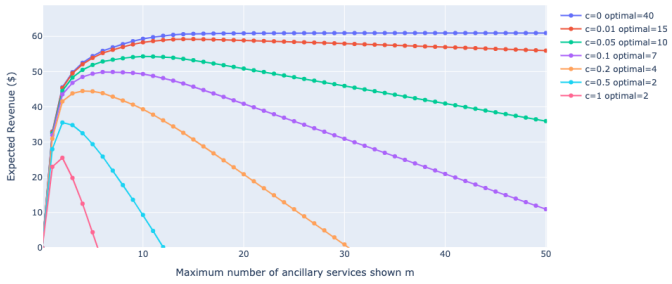


Figure 3-5: Expected revenue (in $) versus maximum number of services shown. In this example the number of possible ancillary services is $p = 100$.

services $p = 100$, the optimal number of services to show is 40. It also illustrates that the optimal number of product shown decreases with c, and that their is a linear dependency. We conclude that the solution to the ASDO algorithm is non trivial, and that the framework can properly account for sensory overload as it was designed to do.

3.5.3 Testing the end-to-end framework on a synthetic example

The framework that we propose is not restricted to a retail setting. In order to showcase this, we discuss a synthetic case study inspired from the airline industry and show the usefullness of the framework in this setting. We note that the case study is purely for illustrative purposes and because of lack of data, we sample different problem parameters for the experiments.

We start by discussing the ancillary service optimization problem in the context of the airline industry first.

- The ancillary services are the auxiliary products provided by the airline company: seat choice, premium meal, priority privileges, excess baggage, etc. Note that the synthetic nature of the experiment allows us to artificially increase the number of such ancillary services as much as desired.

- The customer-related features are booking date, number of flights booked in

the past, average price of these flights, number of miles and status with the airline. We generate the booking date as the number of days in advance the flight is being booked, through an exponential distribution. The number of flights booked in the past as well as the status with the airline are generated as categorical variables (0 for first-time users, 1 for infrequent users or 2 for frequent users) from a uniform distribution. The average price is drawn from normal distribution. We vary the parameters of these distribution throughout the case study for more robust results.

- The features related to the product itself here include time of the flight (drawn from a normal distribution), whether it is connecting flight or not, the size of the aircraft, the popularity of the destination (all three drawn from uniform distributions), and finally the price of the product, which is drawn using a linear response to the previous features, as well as unobserved, latent features such as whether there is an event at the destination.

- Additionally, we add the price of the ancillary products as a feature, and we draw it from different normal distributions.

- Finally, the response function is which ancillary product(s) did the customer buy, if any. The underlying assumption here is that there exist different types of travelers (classes) and each one of these types have a different response function to features of the product and the services, which we take as a multinomial logit for the purpose of the case study.

We generate $40,000$ observations, with 40 customer features and 40 product features, grouped into 4 clusters. We consider $p = 20$ ancillary services. We also generate the Net Present Values of the services at random. We first test the CWC method on this case study, and compare it to RF, the only clear competitor from the synthetic experiments. We obtain an out-of-sample accuracy for CWC of 0.594 (vs. 0.5288 for RF). We then apply the ASDO algorithm to get the optimal service display(Figure 3-6). Importantly, we find that segmentation plays an important role and not all

services are displayed to customers of all segments. Furthermore, both the optimal assortment size, as well as the services shown, vary from cluster to cluster.

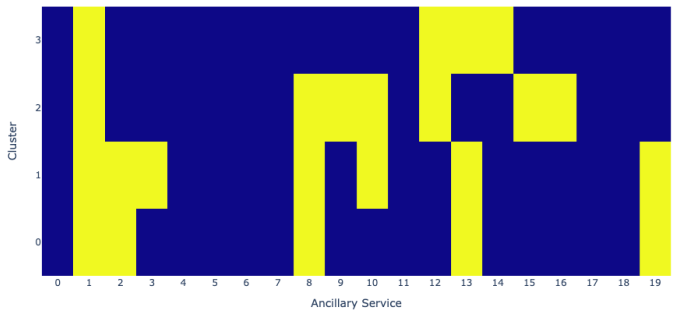


Figure 3-6: Optimal Assortment of Ancillary Services for each Cluster. Yellow means the service is shown, Blue means the service is not shown. For example, service 1 is shown to every type of customer, while service 4 is never shown, and service 8 is shown to everyone except customer class 3. We also note that not the same number of products is shown to each class of customers.

In conclusion, our end-to-end framework is general, flexible, interpretable, and competitive in terms of accuracy with alternative methods.

3.6 Conclusions

In this chapter, we have tackled the problem of personalized ancillary service advertising in a holistic way: from prediction and causal estimation to prescription. We developed a novel method to estimate the propensity of each service, called Cluster-While-Classify (CWC), that jointly creates clusters of customers based on their features and historic behavior, and trains classification predictive models within each of these clusters to model their unique responses. This method is accurate, with more than 74 % out-of-sample accuracy across 7 different combination of services, and is interpretable —and directly actionable— for decision makers. We also leveraged Double Machine Learning (DML) and Causal Forests (CF) to estimate the NPV, or Net Present Value of these services. Finally, we formulated the corresponding optimization problem for personalized prescription of service offering that maximizes the long-term revenue and proposed an algorithm to solve it efficiently. This holistic

framework offers a transformative and sustainable impact for our industry partner, that is actively testing it and deploying it, with an estimate 2.5-3.5% uplift in revenue, with translates to roughly $80-100 million.

Chapter 4

Holistic Prescriptive Analytics for Continuous and Constrained Optimization Problems

4.1 Introduction

A key objective in Operations Research (OR) is to obtain decisions based on data. The traditional approach in OR is to build models from which we can derive decisions. By and large, data in OR models has only played a supporting role. In contrast, data in Machine Learning (ML) models has played a protagonistic role.

We are given training data $\{\boldsymbol{x}_i,\ y_i,\ \boldsymbol{z}_i\}$ and a cost function $c(y_i, \boldsymbol{z}_i)$ for $i \in \{1, \ldots, n\} = [n]$, $n \in \mathbb{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ is the vector of covariates for observation i, $y_i \in \mathbb{R}$ is the outcome, and $\boldsymbol{z}_i \in \mathcal{Z}$ the decision set. Our goal is to prescribe an optimal treatment that minimizes the cost function c for a new observation $\boldsymbol{x}_0$. We introduce the decision variable $\bar{\boldsymbol{z}}_i \in \mathcal{Z}$ for $i \in \{0, \ldots, n\}$ which is the treatment that we prescribe for observation i, in contrast with $\boldsymbol{z}_i \in \mathcal{Z}$ which is the observed treatment, and which is available only for $i \in [n]$, i.e., training data.

Since the outcome of the new observation $\boldsymbol{x}_0$ is unknown, we need to predict it in order to be able to compute its cost, and ultimately prescribe an optimal $\bar{\boldsymbol{z}}_0$. If we

are given the function $f(\boldsymbol{x}_0, \bar{\boldsymbol{z}}_0)$, $\forall \bar{\boldsymbol{z}}_0 \in \mathcal{Z}$, to predict the outcome for $\boldsymbol{x}_0$ under any treatment $\bar{\boldsymbol{z}}_0$, the problem we are trying to solve can be written as in Problem (4.1).

$$\begin{aligned} \min_{\bar{\boldsymbol{z}}_0} \; & c(f(\boldsymbol{x}_0, \bar{\boldsymbol{z}}_0), \, \bar{\boldsymbol{z}}_0) \\ \text{s.t.} \; & \bar{\boldsymbol{z}}_0 \in \mathcal{Z}. \end{aligned} \tag{4.1}$$

The standard paradigm for this type of prescriptive problems has been to learn f separately within a class of ML functions $\mathcal{F}$ such that it minimizes the mean-squared error on the predictions over the training data as described in Problem (4.2), then to use it in order to solve Problem (4.1) in a predict-then-optimize fashion.

$$\begin{aligned} \min_{f} \; & \sum_{i=1}^{n} (y_i - f(x_i, z_i))^2 \\ \text{s.t.} \; & f \in \mathcal{F}. \end{aligned} \tag{4.2}$$

Note that what we mean by minimizing over f in Problem (4.2) is minimizing over a finite number of parameters that fully define the function f, for example the weights $\boldsymbol{W}$ of the neurons if f is a neural network, the design T of the tree if f is a decision tree, or the coefficients $\boldsymbol{\beta}$ of the regression if f is a linear, polynomial or convex regression.

However, going beyond the standard predict-then-optimize paradigm, recent efforts have tried to go directly from data to decisions using ML methods. [27] propose a framework that can accommodate a large number of ML algorithms, works for both continuous and discrete variables, and can also accommodate constraints, but still utilizes the two-stage framework of first training a predictive model, and then prescribing in a sequential fashion, despite not using the predictions themselves in the second stage. [24] propose an extension of Optimal Classification and Regression Trees ([22]) called Optimal Prescriptive Trees (OPTs), that simultaneously predicts and prescribes. While the method is powerful and interpretable, it is limited by that

the decisions need to be discrete and small in cardinality, and cannot be constrained.

In this chapter, we propose a generalization of OPTs to a larger class of ML methods than trees, that simultaneously predicts and prescribes, can accommodate continuous decisions as well as discrete decisions of high cardinality, and also allow constraints on the decision variables.

To prescribe an optimal treatment $\bar{\boldsymbol{z}}_0$, we propose a framework for prescriptive analytics that jointly regroups observations into clusters with similar behaviors, learns a predictive model over each of these clusters and prescribes the optimal decisions under constraints. The intuition behind this framework is that clustering allows to achieve higher accuracy by aggregating data into clusters and to divide the training process into smaller subproblems, while performing all the tasks (clustering, prediction and prescription) jointly allows us to find the right trade-off between accurate predictions and optimal decisions, in a tractable way that can account for continuous and constrained decision spaces. This framework can be used with a wide range of predictive ML methods and cost functions for the prescriptions, with both constrained and unconstrained problems, as well as with a decision space that can be infinite or even continuous. We show that the proposed method, which we will refer to as **Holistic Prescriptive Analytics method** or **HPA**, is scalable, and that it significantly improves state-of-the-art performance.

Like OPTs, our framework also provides interpretability when the ML methods used are interpretable, which can be vital in some applications (See Section 4.6 for a Case Study).

4.2 Relevant Literature

The standard paradigm in real-world analytics problems involving prediction and optimization is predict-then-optimize, where ML tools are used to predict a point estimate of an uncertain quantity, that is then plugged in a nominal optimization problem to solve for the optimal decisions.

4.2.1 Predict-then-Optimize Methods

Due to the poor performances of this paradigm, researchers have tried to account for the variability of the prediction by solving the optimization problem over the expected value of the prescription cost, and by approximating this expected value by the empirical sample average (Sample Average Approximation SAA), see for example [35]. On the other hand, researchers proposed to use Robust Optimization instead of Stochastic Optimization to solve for the optimal decisions once the prediction is made, see for example [16] and [25]. In this case, the uncertainty of the prediction is accounted for by considering the worst-case scenario within an uncertainty set.

[103] combines both approaches by proposing a more robust estimation of SAA using propensity scoring. [73] extends the predict-then-optimize framework into a Smart Predict-then-Optimize (SPO) by introducing an SPO loss function which measures the decision error induced by a prediction as the new objective function for the prediction phase. This prediction is then used as a point estimate in the optimization phase in the same way as the standard predict-then-optimize. A similar idea is explored in [164], which, while remaining a two-stage sequential approach, specifically trains the predictive model to perform well on the prescriptive problem by incorporating the latter in the gradient-based training of the former.

[65] also proposes a way to make ML and combinatorial optimization interact in the predict-then-optimize framework, first by proposing an algorithm with ranking objectives in the case of learning linear functions ([65]), and then by extending it to the case of optimization problems solvable by dynamic programming ([144]) with the same linearity constraints.

4.2.2 Direct Methods

The difference between direct methods and the predict-then-optimize framework is that the prescriptions are made at the same time as the predictions, or there is a feedback-loop between the prediction process and the prescription process instead of having a sequential structure.

[157] introduces the idea of integrating operational cost into the training of ML models by adding it directly into the objective function for the prediction with either an optimistic or a pessimistic bias and shows that the simultaneous optimization is more effective than the sequential one. However, the proposed formulation can be computationally very difficult.

[27] also moves away from the predict-then-optimize paradigm and uses a framework that leverages previously trained ML models in a prescriptive optimization problem without actually using the point estimate by optimizing over a weighted combination of training samples instead with the weights computed based on the ML methods trained for the prediction. [24] introduces OPTs, which change the objective function of the Optimal Classification and Regression Trees [22, 23] to account for the prescription cost, and hence making the prediction and the optimal decisions at the same time instead of sequentially. However, the decisions need to be discrete and small in cardinality and cannot be constrained, due to the complexity of the formulation of the prescriptive tree model.

[64] draws a similar conclusion by benchmarking a wide range of predict-then-optimize method and direct methods on the Knapsack problem: direct methods outperform alternative indirect methods; however, their tractability seems limited. [64] highlights the need for better direct methods and more automatic ways to create semi-direct approaches to a new optimization problem, which is exactly what we do in this chapter. We propose a tractable, holistic approach with clustering and an alternative iterative search algorithm to keep the framework tractable even when other methods fail, while still leveraging the benefits of simultaneous optimization.

4.2.3 Contributions

Motivated by these recent efforts in advancing the field of prescriptive analytics and the limitations of some of the existing algorithms, we develop a prescriptive framework that

1. **Can accommodate a wide range of ML methods for prediction:** for

example, Linear Regression, Elastic Net, Support Vector Machines, Convex Regression ([29]), and Convex Neural Networks ([17]) in the case of a convex prescriptive cost function, and any ML method (e.g., Decision Trees) in specific cases where the predictive loss and the prescriptive cost have the same structure (e.g., any ML method minimizing the squared loss if the cost function is of the form $c(y, z) = y^2$).

2. **Simultaneously predicts and prescribes**, moving away from the predict-then-optimize paradigm.

3. **Allows continuous and constrained decisions.**

4. **Allows the uncertain outcome y to be also a function of the decisions $\boldsymbol{z}$.**

5. **Regroups the observations into clusters of similar behavior** which enables us to achieve higher accuracy by aggregating data into clusters and to divide the training process into smaller subproblems for tractability purposes.

6. **Has strong computational performance and wide applicability** for both synthetic and real world datasets.

We test different versions of this framework and perform extensive numerical experiments to benchmark the predictive accuracy, prescriptive cost, scalability and interpretability of the framework (see Section 4.6). We perform these experiments on synthetic data, i.e., generated data where the underlying behavior, and hence the counterfactuals, are known, and on two real-world case studies: a diabetes case study where we prescribe an optimal treatment in order to minimize the expected average blood sugar levels of patients, and an assortment optimization case study where we optimize display for a food retailer in order to maximize revenue. In the former case study we use an interpretable approach due to the medical nature of the application. In these experiments, we show that the HPA framework is scalable and provides a performance edge over alternative methods. Across the 3 computational

experiments that we detail in this chapter, with datasets up to 100,000 observations and about 100 features, we observe between 14% and 30% improvement in terms of out-of-sample prescriptive cost compared to the predict-then-optimize baseline, and between 5% and 17% improvement compared to the state-of-the-art predictive-prescriptive method ([27]).

4.3 Proposed Approach

The first idea of our approach combines Problem (4.1) and Problem (4.2) by jointly learning the predictive function and prescribing an optimal treatment. This is done by minimizing over f and $\bar{\boldsymbol{z}}$ the weighted average of the prediction error on the training data and the prescriptive cost on both the training data and the new data point $\boldsymbol{x}_0$. We denote $\lambda \in [0, 1]$ the weight of the prediction error and $(1 - \lambda)$ the weight of the prescriptive cost. We obtain Problem (4.3):

$$\begin{aligned} \min_{f, \bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1 - \lambda) \sum_{i=0}^{n} c(f(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\ \text{s.t} \ & \bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n \in \mathcal{Z}. \end{aligned} \tag{4.3}$$

Note that the first part of the objective function of Problem (4.3) is exactly the objective function of Problem (4.2), weighted by λ. The second part of the objective function of Problem (4.3) is just the cost function used in Problem (4.1), weighted by $1 - \lambda$, and applied to both the new data point $i = 0$ and the training data $i \in [n]$ in order to account for the entire prescriptive cost, even though the final objective is just to get an optimal treatment $\bar{\boldsymbol{z}}_0$ only. The weights given to the prescriptive cost and the predictive cost are controlled by λ. For $\lambda = 1$ the problem is purely predictive, and for $\lambda = 0$, the problem is purely prescriptive. Both extreme values will result in poor performances out-of-sample on the final cost function. That is why this λ is chosen through cross-validation to properly calibrate the weight given to each of these two costs in the optimization problem.

Our overall approach extends Problem (4.3) by clustering each observation $i \in \{0, \ldots, n\}$ into a cluster $j \in [k]$, where k is a predefined number of clusters. Then, within each cluster j, we train a different predictive model f_j to predict outcomes based on covariates and treatments. Since we do not have any outcome for observation $\boldsymbol{x}_0$, we assign it to the cluster that minimizes the average distance in terms of features of its points to $\boldsymbol{x}_0$. Here, we take the ℓ_2-norm, but any other tractable distance can be considered. The key of our approach is that all three steps (clustering, prediction and prescription) are done jointly.

We also define the following additional notation:

(a) $u_{i,j}$: the binary outcome of whether observation i is assigned to cluster j.

(b) f_j: a function that learns the prediction of the outcome y based on the covariates $\boldsymbol{x}$ and the observed prescriptions $\boldsymbol{z}$.

(c) N_{min}: hyper-parameter defining the minimum number of training observations in each cluster.

Problem (4.3), with the additional layer of clustering becomes Problem (4.4).

$$
\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{f},\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n}\sum_{j=1}^{k} u_{i,j}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda)\sum_{i=0}^{n}\sum_{j=1}^{k} u_{i,j} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
\text{s.t} \ & \bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n \in \mathcal{Z} \ \text{(constraints on decisions)}, \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min}, \ \forall j \in [k] \ \text{(minimum number of training observations per cluster)}, \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \forall i \in \{0, \ldots, n\} \ \text{(one observation is assigned to exactly one cluster)}, \\
& u_{0,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k] \ (\boldsymbol{x}_0 \text{ is assigned to the closest cluster}).
\end{aligned}
\tag{4.4}
$$

Note that the last constraint of Problem (4.4) enforcing that $\boldsymbol{x}_0$ is assigned to the closest cluster is an inequality instead of an equality to ensure that the problem

remains feasible if several clusters satisfy the closeness condition.

To highlight the fact that $\mathcal{Z}$ can be constrained, we replace, without loss of generality, $\mathcal{Z}$ by $\{(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_n) \mid h_r(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_n) \leq b_r,\ r \in [s]\}$ for some $s \in \mathbb{N}$. These constraints on the decisions can for example be budget constraints. Our proposed approach can then be summarized in Problem (4.5).

$$
\begin{aligned}
\min_{\boldsymbol{u}, \boldsymbol{f}, \bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n} \sum_{j=1}^{k} u_{i,j} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
\text{s.t} \ & h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \ \forall r \in [s] \ \text{(constraints on decisions)}, \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min}, \ \forall j \in [k] \ \text{(minimum number of training observations per cluster)}, \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \forall i \in \{0, \ldots, n\} \ \text{(one observation is assigned to exactly one cluster)}, \\
& u_{0,j} \leq \mathbb{1} \left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k] \ (\boldsymbol{x}_0 \text{ is assigned to the closest cluster}).
\end{aligned}
\tag{4.5}
$$

Note that the use of one single x_0 is for illustrative purposes, the framework remains identical when we have $n_{test} \in \mathbb{N}$ new observations we want to prescribe an optimal decision for. See Appendix §C.1.

Importance of the Clustering in the HPA approach.

The clustering has two main advantages: increasing the predictive accuracy of simpler tractable models (for example training a Linear Regression model on the entire dataset yields worse results than training a different Linear Regression model on each subgroup with similar underlying behaviors), and reducing the size of the training set for the different models by training them into smaller batches.

It is also important to note that clustering and predicting at the same time adds a value in comparison to clustering then prediction, especially when the right distance

metric to use is unknown. To illustrate that, consider the following example (Figure 4-1(a)) where we get labeled data points from two types of clusters (yellow and blue). In addition to this cluster assignment, we consider that each data point is further characterized by a single one-dimensional feature $x \in \mathbb{R}$. We are interested in a continuous target variable $y \in \mathbb{R}$. A point in the blue region behaves according to the true target function $y = f_1(x) = -50x$, while a point in the yellow region behaves according to the true target $y = f_2(x) = 100x$. The goal is to use clustering and supervised learning, here linear regression, to learn the true underlying model and make a prediction for a new data point (the point in grey, for which we consider for simplicity of the example that $x = 1$). See Figure 4-1(b).

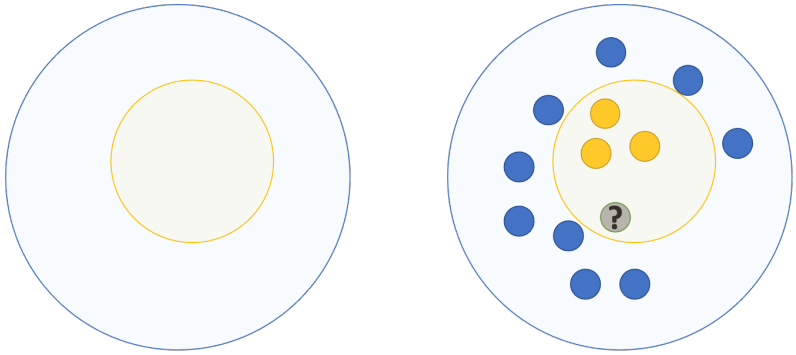


Figure 4-1: (a) True underlying model. (b) Data samples from this model.

By running the k-Means clustering first, and **then** training a linear regression within each cluster, we obtain the results in Figure 4-2. The function $f_1(x) = -50x$ is learned correctly, however the function $f_2(x)$ is estimated to be $14x$ instead of $100x$, and the new point is assigned to the wrong cluster, with a prediction $y = -50$ instead of $y = 100$, i.e., with an absolute error of 150.

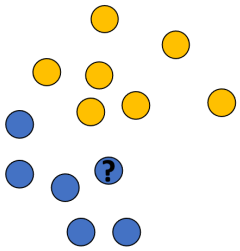


Figure 4-2: Results of Cluster **then** Predicting.

However, by clustering and predicting at the same time similarly as we do in

the HPA framework, we obtain the results in Figure 4-3, where the model perfectly learns the true underlying model, and assigns the new point to the correct clusters, predicting $y = 100$, i.e., with 0 absolute error.

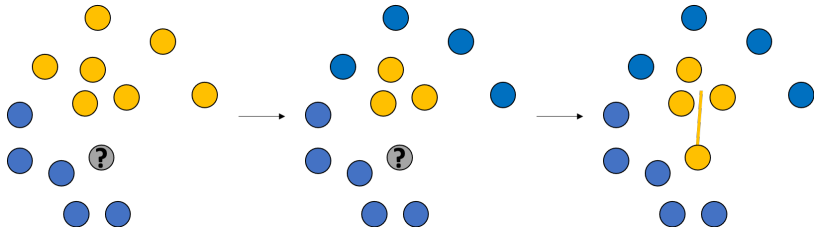


Figure 4-3: Results of Clustering and Predicting **at the same time**.

We show more experiments in Appendix §C.2, and point the reader to related literature that shows the advantages of clustering and predicting simultaneously instead of doing in sequentially: e.g, [10] proposes exhaustive benchmarks showing that a Cluster-While-Regress (CWR) approach outperforms a Cluster-then-Regress approach in many cases, and [130] similarly shows on synthetic data and real-world experiments that a Cluster-While-Classify (CWC) approach outperforms a sequential Cluster-then-Classify one.

4.3.1 Solving the General Problem

In this section, we outline Algorithm 4 for HPA. We assume that optimizing over the predictive function $\boldsymbol{f}$ involves a collection of parameters. We cover in Algorithm 1 the case of learning models that are fully defined by a set of parameters of fixed size which can be globally optimized over a custom loss function. This definition includes, but is not limited to neural networks, where the parameters are the weights of the neurons $\boldsymbol{W}_j$, optimal trees, where the parameters are the design of the tree T_j and linear, polynomial, and convex regression, where the parameters are the coefficients $\boldsymbol{\beta}_j$ of the regression for each cluster j. When we say optimize over the function f_j, we mean optimize over the finite parameters that define f_j.

We propose an alternative algorithm for non-parametric and black-box models that minimize the mean-squared error loss in Section 4.4 and expand extensively on

the cases of decision trees and linear regression in Section 4.5.1 and Section 4.5.2, respectively.

We describe the Algorithm in Algorithm 4. We use a local search iterative approach with multiple restarts to solve Problem (4.5). We start with a warm start $\boldsymbol{u}^{(1)}$ and $\bar{\boldsymbol{z}}^{(1)}$ for $\boldsymbol{u}$ and $\bar{\boldsymbol{z}}$. Then, for $t \in [T_{max} - 1]$, we fix $\boldsymbol{u}^{(t)}$ and $\bar{\boldsymbol{z}}^{(t)}$ to find $\boldsymbol{f}^{(t+1)}$ as in Equation (4.6) in Algorithm 4, then we fix $\boldsymbol{u}^{(t)}$ and $\boldsymbol{f}^{(t+1)}$ to find $\bar{\boldsymbol{z}}^{(t+1)}$ as in Equation (4.7) in Algorithm 4, and finally we fix $\bar{\boldsymbol{z}}^{(t+1)}$ and $\boldsymbol{f}^{(t+1)}$ to find $\boldsymbol{u}^{(t+1)}$ as in Equation (4.8) in Algorithm 4.

Algorithm 4 IterativeSolve($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{u}^{(1)}$,$\bar{\boldsymbol{z}}^{(1)}$, N_{min}, T_{max}, k)

for $t \in [T_{max} - 1]$ **do**

for $j \in [k]$ **do**

$$f_j^{(t+1)} = \arg\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t)}), \bar{\boldsymbol{z}}_i^{(t)}), \quad (4.6)$$

end for

$$\bar{\boldsymbol{z}}^{(t+1)} = \arg\min_{\bar{\boldsymbol{z}}} \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j}^{(t)} c(f_j^{(t+1)}(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \text{ s.t } h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s], \quad (4.7)$$

$$\begin{aligned} \boldsymbol{u}^{(t+1)} = \arg\min_{\boldsymbol{u}} & \lambda \sum_{i=1}^{n} \sum_{j=1}^{k} u_{i,j} (y_i - f_j^{(t+1)}(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 \quad (4.8) \\ & + (1-\lambda) \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j} c\left(f_j^{(t+1)}(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t+1)}), \bar{\boldsymbol{z}}_i^{(t+1)}\right) \\ & \text{s.t } \sum_{i=1}^{n} u_{i,j} \geq N_{min} \ \ \forall j \in [k] \\ & \sum_{j=1}^{k} u_{i,j} = 1, \ \ \forall i \in \{0, \ldots, n\} \\ & u_{0,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}}\right), \forall j \in [k], \end{aligned}$$

end for

Return $f_1^{(T_{max})}, f_2^{(T_{max})}, \ldots, f_k^{(T_{max})}$, $\bar{\boldsymbol{z}}^{(T_{max})}$, $\boldsymbol{u}^{(T_{max})}$.

Algorithm 4 converges to a local minimum of Problem (4.5), which is why we use multiple restarts. We can get warm starts $\bar{\boldsymbol{z}}^{(1)}$ and $\boldsymbol{u}^{(1)}$ by setting $\bar{\boldsymbol{z}}_i^{(1)}$ to $\boldsymbol{z}_i$ for $i \in [n]$ and $\bar{\boldsymbol{z}}_0^{(1)}$ at random, and getting $\boldsymbol{u}^{(1)}$ from a clustering method such as k-means on the $\boldsymbol{x}_i$'s. Note that Problem (4.8) in Algorithm 4 can be solved optimally similarly to [30], or solved greedily by first optimizing over the values of $\boldsymbol{u}_{i,j}^{(t+1)}$ for $i \in [n]$ and

$j \in [k]$ then assigning $\boldsymbol{x}_0$ to the closest cluster in average.

The tractability of Problem (4.6) in Algorithm 4 for each $j \in [k]$ depends on the complexity of f_j and that of the cost function c. It does remain tractable for a large class of functions, for example, if f_j is convex, and c is convex, the optimization is a convex optimization problem as a composition and sum of convex functions, which makes this HPA approach very general, contrary to [157] which requires restrictive properties on c or [65] which requires f_j to be linear. This applies for example for Naive Bayes, Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Logistic Regression, Support Vector Machines, and even Convex Regression ([29]) and Convex Neural Networks ([17]).

Furthermore, even if f_j has a much more complex structure like Neural Networks, it can still be solved in our setting in some special cases of c. We explore one of these special cases where c is quadratic in the next section.

4.4 Quadratic Cost Function

When the cost function can be written $c(y, \bar{\boldsymbol{z}}) = y^2$, we can solve Problem (4.6) in Algorithm 4 more efficiently. In this case, we outline Algorithm 5 for HPA, which generalizes to any ML method that minimizes the mean-squared error loss, regardless of whether or not it can formulated as an explicit optimization problem. This includes for example greedy, non-parametric models such as CART (Section 4.5.2) and Random Forests, and black-box models, in addition to the class of models covered by Algorithm 4. In this section, optimizing over function f_j means selecting the "best" f_j with regards to the mean-squared error on a dataset that we define, within a selected class of functions. We further discuss conditions for tractability in the complexity analysis.

In Algorithm 5, instead of changing the objective function of the ML method f_j for each cluster j to get Problem (4.6) in Algorithm 4, which is hard to do for some ML

methods such as CART ([39]) or even some complex neural network structures ([107]), we change the dataset over which we minimize regular mean-squared error loss. If we assume λ is rational, i.e. can be written $\frac{p}{p+q}$, $p, q \in \mathbb{Z}_+$, with p and q preferably small, we create p copies of every training observation i in cluster j to account for the predictive part of the objective function as in Problem (4.9) in Algorithm 5. Then we create q modified copies of every observation i in cluster j where we set the target variable to 0 to get exactly the prescriptive part of the objective function as in Problem (4.10) in Algorithm 5.

Algorithm 5 PredictiveFit($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{l}$, $\bar{\boldsymbol{z}}$, k, p, q)

Let $\lambda = \frac{p}{p+q}$, $p, q \in \mathbb{Z}_+$.

for $j \in [k]$ **do**

Create p copies of every training observation i in cluster j: $\boldsymbol{x}_i$, $\boldsymbol{z}_i$, y_i, (4.9)

Create q modified copies of every observation i in cluster j: $\boldsymbol{x}_i$, $\bar{\boldsymbol{z}}_i$, 0, (4.10)

Solve the regular mean-squared error model fitting for f_j on this new dataset,

i.e. $f_j = \arg\min_f \sum_{i'=1}^{p\bar{n}_j+qn_j} (y'_i - f(\boldsymbol{x}'_i, \boldsymbol{z}'_i))^2$ with the notations that variables with a " $'$ " superscript indicate variables in the new created dataset, n_j the number of observations in cluster j and $\bar{n}_j$ the number of training observations in cluster j.

end for

Return $f_1, f_2, \ldots, f_k$.

Proposition: If there exist p and $q \in \mathbb{Z}_+$ such that $\lambda = \frac{p}{p+q}$, then the output of Algorithm 5, and that of Problem (4.6) in Algorithm 4 are equal.

Proof: Let $j \in [k]$ a cluster with n_j observations and $\bar{n}_j$ training observations. The minimization problem in Algorithm 5 is equivalent to Equation (4.11).

$$
\begin{aligned}
&\min_{f_j} \sum_{i'=1}^{p\bar{n}_j+qn_j} (y'_i - f_j(\boldsymbol{x}'_i, \boldsymbol{z}'_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} p(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} q(0 - f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \frac{p}{p+q}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} \frac{q}{p+q}(0 - f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \lambda(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} (1-\lambda)(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i))^2 \\
\iff &\min_{f_j} \sum_{i=1}^{\bar{n}_j} \lambda(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + \sum_{i=1}^{n_j} (1-\lambda) c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
\iff &\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i)
\end{aligned}
\tag{4.11}
$$

Which proves the equivalence. ■

Note that if we consider different loss functions for the ML method (e.g., the absolute error), Algorithm 2, and subsequent proposition still applies for cost functions with the same structure (e.g., $c(y, z) = |y|$).

Complexity of the algorithm

In order to bound the complexity of Algorithm 4 in the quadratic case, we assume that Problem (4.12), which minimizes the mean-squared error loss for each ML method f_j over a dataset of size n can be solved with complexity $\mathcal{O}(\rho_1(n))$.

$$
\min_{f_j} \sum_{i=1}^{n} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2, \tag{4.12}
$$

We also assume that Problem (4.13) which minimizes, given f_j and x, the prescriptive cost for decisions z over a dataset of size $n+1$ can be solved with complexity $\mathcal{O}(\rho_2(n))$.

$$\begin{aligned} \min_{\bar{\boldsymbol{z}}} \ & \sum_{i=0}^{n} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\ \text{s.t. } & h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \ \forall r \in [s], \end{aligned} \tag{4.13}$$

We have that complexity of solving Problem (4.6) in Algorithm 4 at each time t which consists of fitting $f_1, \ldots, f_k$ using Algorithm 5 is $\mathcal{O}(\sum_{j=1}^{k} \rho_1((p+q) \times n_j))$, where n_j is the number of observations in cluster j. Complexity of solving Problem (4.7) at each time t is $\mathcal{O}(\rho_2(n))$, while Problem (4.8) in Algorithm 4 is an assignment problem, completely independent of the complexity of the ML model or that of the constraints on the decisions, its complexity is negligible compared to the 2 previous steps.

So we have a total complexity of q_T bounded by $O(T_{max}(k\rho_1((p+q) \times n) + \rho_2(n)))$. On average, for evenly-distributed clusters, this complexity becomes $q_T = O(T_{max}(k\rho_1((p+q) \times \frac{n}{k}) + \rho_2(n)))$, which is tractable for reasonably small p and q. Note that if $T_{max} = 2$ (1 iteration), then the algorithm becomes a predict-then-optimize framework, using the information of the prescription cost to train a predictive function, and then using this predictive function to decide on a treatment, similarly to the idea of Smart PTO ([73]) and [157], but with a clustering component. However, for $T_{max} > 2$, f_j and $\bar{\boldsymbol{z}}$ are evaluated jointly since one is iteratively used to solve for the other in a repeated fashion until convergence to a local minimum of the joint predictive-prescriptive problem.

4.5 Examples of Parameterizations

We can further improve the efficiency of solving Problem (4.5) for particular ML methods. We discuss the cases of Linear Regression and Classification and Regression Trees.

4.5.1 Linear Parametrization

In this subsection, we discuss the case where f_j is linear for each cluster $j \in [k]$, i.e. there exist coefficients $\boldsymbol{\beta}_j$ and γ_j such that $f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i) = \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \bar{\boldsymbol{z}}_i \ \forall i \in [n]$. Problem (4.5) can be then rewritten to get Problem (4.14).

$$
\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{\beta},\boldsymbol{\gamma},\bar{\boldsymbol{z}}} \ & \lambda \sum_{i=1}^{n}\sum_{j=1}^{k} u_{i,j}(y_i - \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \boldsymbol{z}_i)^2 + (1-\lambda)\sum_{i=0}^{n}\sum_{j=1}^{k} u_{i,j}(\boldsymbol{\beta}_j^T \boldsymbol{x}_i + \gamma_j \bar{\boldsymbol{z}}_i)^2 \\
\text{s.t} \ & h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s], \\
& \sum_{i=1}^{n} u_{i,j} \geq N_{min} \ \ \forall j \in [k], \\
& \sum_{j=1}^{k} u_{i,j} = 1, \ \ \forall i \in \{0, \ldots, n\} \ , \\
& u_{0,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k]
\end{aligned}
\tag{4.14}
$$

For the linear parametrization, Problem (4.6) in Algorithm 4 can easily be formulated as a tractable optimization problem, to obtain Problem (4.15) in Algorithm 6, which can be solved directly. The full corresponding iterative procedure to solve Problem (4.14) is described in Algorithm 6, which is similar, but more efficient than Algorithm 4.

Algorithm 6 IterativeSolveLinear($\boldsymbol{X}$,$\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{u}^{(1)}$,$\bar{\boldsymbol{z}}^{(1)}$, N_{min}, T_{max}, k)

for $t \in [T_{max} - 1]$ **do**

for $j \in [k]$ **do**

$$\boldsymbol{\beta}_j^{(t+1)}, \gamma_j^{(t+1)} = \arg\min_{\boldsymbol{\beta}_j, \gamma_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - \boldsymbol{\beta}_j^T \boldsymbol{x}_i - \gamma_j \boldsymbol{z}_i)^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} (\boldsymbol{\beta}_j^T \boldsymbol{x}_i + \gamma_j \bar{\boldsymbol{z}}_i^{(t)})^2, \quad (4.15)$$

end for

$$\bar{\boldsymbol{z}}^{(t+1)} = \arg\min_{\bar{\boldsymbol{z}}} \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j}^{(t)} ((\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i + \gamma_j^{(t+1)} \bar{\boldsymbol{z}}_i)^2 \text{ s.t } h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \forall r \in [s],$$

$$\boldsymbol{u}^{(t+1)} = \arg\min_{\boldsymbol{u}} \sum_{j=1}^{k} \left[\sum_{i=1}^{n} u_{i,j} \lambda (y_i - (\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i - \gamma_j^{(t+1)} \boldsymbol{z}_i)^2 + \sum_{i=0}^{n} u_{i,j} (1-\lambda) ((\boldsymbol{\beta}_j^{(t+1)})^T \boldsymbol{x}_i + \gamma_j^{(t+1)} \bar{\boldsymbol{z}}_i^{(t+1)})^2 \right]$$

$$\text{s.t } \sum_{i=1}^{n} u_{i,j} \geq N_{min} \quad \forall j \in [k]$$

$$\sum_{j=1}^{k} u_{i,j} = 1, \quad \forall i \in \{0, \ldots, n\}$$

$$u_{0,j} \leq \mathbb{1} \left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=1}^{n} u_{i,j'} \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2^2}{\sum_{i=1}^{n} u_{i,j'}} \right), \forall j \in [k],$$

end for

Return $\boldsymbol{\beta}_1^{(T_{max})}, \boldsymbol{\beta}_2^{(T_{max})}, \ldots, \boldsymbol{\beta}_k^{(T_{max})}$, $\gamma_1^{(T_{max})}, \gamma_2^{(T_{max})}, \ldots, \gamma_k^{(T_{max})}$, $\bar{\boldsymbol{z}}^{(T_{max})}$, $\boldsymbol{u}^{(T_{max})}$.

Note that this approach can also be applied to regularized linear regression, such as Ridge Regression or Elastic Net.

4.5.2 Tree-Based Parametrization

In this subsection we set f_j for $j \in [k]$ to be equal to a tree ML method, such as CART ([39]) or Optimal Trees ([22]). We apply Algorithm 4 and Algorithm 5 to this parametrization.

In order to solve Problem (4.6) in Algorithm 4, i.e. learning the predictive function $f_j^{t+1} = \arg\min_{f_j} \lambda \sum_{i=1}^{n} u_{i,j}^{(t)} (y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} u_{i,j}^{(t)} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i^{(t)}), \bar{\boldsymbol{z}}_i^{(t)})$ for each cluster j at iteration t, we use Algorithm 5 as described in Section 4.4. Since Algorithm 5 only requires to minimize the mean-squared error loss over a new dataset, it can be used with both the globally optimal trees, or with the iterative CART algorithm.

For Problem (4.7) in Algorithm 4, there are two approaches, depending on whether the problem is constrained or not:

1. If the problem is unconstrained, then we can find an optimum just by iterating over each observation $i \in [1, n]$ independently. For each i, we find all the possible

leaves of the trained tree to which this observation can belong, and pick one treatment that minimizes the corresponding cost. We note that even though this number of leaves is theoretically bounded by 2^d where d is the depth of the tree, it is usually extremely small, as it corresponds to the number of splits on the treatment $\bar{\boldsymbol{z}}_i$ for the region of space where X_i, i.e. the covariates of observations i, belongs. Even in the case where the treatment space is large or continuous, the fact that the cost function is a function of the outcome y_i only, which is itself a tree-based function of X_i and $\bar{\boldsymbol{z}}_i$, the structure of the tree makes the minimization computationally tractable: any $\bar{\boldsymbol{z}}_i$ that verifies the constraints that would lead to the optimal leaf can be chosen as the optimal solution.

2. If the problem is constrained, then it is necessary to use heuristics, or formulate Problem (4.7) in Algorithm 4 as an Mixed-Integer Program. The exact formulation is given by [24]. In our case however, the problem is simpler because it does not require to retrain the tree, hence reducing the number of variables of the problem by several orders of magnitude.

Problem (4.8) in Algorithm 4 is again an assignment problem that does not depend on the chosen ML method.

We also note that once this model is trained, the decision-making process that outputs an optimal decision $\bar{\boldsymbol{z}}$ given covariates $\boldsymbol{x}$, can be described with a simple decision tree, which makes it interpretable. In Section 4.6, we show computational results for the tree-based formulation applied in a medical setting where interpretability is particularly relevant.

4.6 Experimental Results

We perform extensive experiments for both the linear and the tree-based parametrizations (applied with Classification and Regression Trees - [39]). In the following tests, we use multiple restarts for the local-search algorithm.

For the real-world data, since the counterfactuals are unknown, we evaluate the

results by learning the true function with an XGBoost ([49]) and a Feed-Forward Neural Network ([107]), with the hyper-parameters described in §C.3, we then pick the best-performing model out-of-sample and consider it ground-truth for the underlying behavior.

4.6.1 Linear Parametrization

We test the linear parametrization on synthetic data and on an assortment problem in a real-world case study, against:

1. `predict-then-optimize`: the predict-then-optimize framework with linear regression, i.e., we train a linear regression on the output y, then separately optimize for the z given the parameters of this regression for each new data point.

2. `saa`: Sample Average Approximation ([35]), similarly to `predict-then-optimize`, we first train a linear regression, but instead of optimizing on the prediction, we simulate data points from the linear regression, accounting for the trained error ϵ, and then we optimize z over the sample average.

3. `pred-presc`: Linear Predictive-Prescriptive, where we train a linear regression on y separately, then we use this regression to compute weights which are used directly in the optimization problem over z instead of using the predictions themselves. See [27] for more details. No hyper-parameter tuning is necessary for these first three approaches.

4. `lin-sim`: Simultaneous Optimization with Optimistic Bias ([157]), where we optimize over the same objective as Equation (4.14) but without any clustering component nor iteration. The values tested for the hyper-parameter tuning of λ can be found in Appendix §C.3 and are identical to the ones tested for the HPA approach.

5. `lin-sim`: Simultaneous Optimization with Optimistic Bias ([157]), where we optimize over the same objective as Equation (4.14) but without any clustering

component nor iteration. The values tested for the hyper-parameter tuning of λ can be found in Appendix §C.3.

6. `elastic-sim`): Identical to `lin-sim` but with a penalized linear regression, i.e., we add an elastic net regularization to the objective ([167]). The values tested for the hyper-parameter tuning of this penalization can be found in Appendix §C.3.

Synthetic Data

We simulate a group of $n \in \mathbb{N}$ patients. Each patient $i \in [n]$ is characterized by $p \in \mathbb{N}$ features and reacts differently to a binary treatment $z_i \in \{0, 1\}$ that affects a target variable $y_i \in \mathbb{R}$ that we need to predict and that we want to minimize. We note $\boldsymbol{x}_i \in [0, 1]^p$ the vector of features for patient i.

We have $r \in \mathbb{N}$ different reactions to the treatment, for each type of reaction $j \in [r]$, we set $y_i = \boldsymbol{\beta_j}^T \boldsymbol{x}_i + \gamma_j z_i$ if and only if i belongs to group j, where $\boldsymbol{\beta_j} \in \mathbb{R}^p$ and $\gamma_j = 10 \cdot ((-1)^j) \cdot (j + 1)$. Note that the direction and the magnitude of the effect of z_i on y_i differs in each group.

We set $n = 10,000$, $r = 5$, and $p = 4$. Then, we create the groups from the following tree structure:

- i belongs to group 1 $\iff x_{i,1} \leq 0.5$ and $x_{i,2} \leq 0.5$.
- i belongs to group 2 $\iff x_{i,1} \leq 0.5$ and $x_{i,2} > 0.5$
- i belongs to group 3 $\iff x_{i,1} > 0.5$ and $x_{i,3} \leq 0.33$
- i belongs to group 4 $\iff x_{i,1} > 0.5$, $x_{i,3} > 0.33$ and $x_{i,4} \leq 0.66$
- i belongs to group 5 $\iff x_{i,1} > 0.5$, $x_{i,3} > 0.33$ and $x_{i,4} > 0.66$

We draw patients uniformly from 1 to 5 groups, and then we draw the features of the patient uniformly and independently within the bounds that define each group. We minimize Mean Squared Error (MSE) for the predictive cost, and $c(y, z) = y^2$ for

the prescriptive cost, similarly to Section 4.4. We separate the dataset into training set (60% of the data), validation set (20%), and test set (20%).

We take the predict-then-optimize approach as a baseline, and test it against our HPA approach, while varying the number of clusters k from 1 to 10, as well as the weight λ (see §C.3) and selecting the best ones in the validation set.

We refer in the experiments to the linear HPA with and without restarts by respectively `lin-hpa-r` and `lin-hpa`. Similarly, we refer to the HPA with an Elastic Net penalization with and without restarts by respectively `elastic-hpa-r` and `elastic-hpa`. In this experiment, restarts are random, i.e., the data points are initially randomly assigned to clusters, then we run the iterative HPA algorithm until convergence, and we select the best local minimum as our global HPA solution.

We obtain the results in Table 4.1 for training data and Table 4.2 for unseen, testing data, for the best values for k and λ.

Model	**Training MSE**	**In-Sample R^2**	**Prescriptive Cost**	**Difference vs Baseline**
`predict-then-optimize`	172546	78.4%	12490	0%
`saa`	172546	78.4%	12146	-3%
`lin-sim`	268174	66.4%	12006	-4%
`elastic-sim`	276846	65.3%	12332	-2%
`pred-presc`	177045	77.9%	11912	-5%
`lin-hpa`	190831	76.1%	9315	-25%
`elastic-hpa`	196943	75.4%	9306	-25%
`lin-hpa-r`	**171094**	**78.6%**	8706	-30%
`elastic-hpa-r`	180244	77.5%	**8166**	**-35%**

Table 4.1: Results of the Synthetic Data simulation (Minimization) - Training.

We notice that the HPA method substantially outperforms the other linear prescriptive methods both in terms of prediction accuracy and most importantly in terms of prescription cost with over 30% improvement compared to baseline in both training and testing data.

In addition to that, Figure 4-4 (train) and Figure 4-5 (test) summarize the results for all tested number of clusters $k \in [1, 10]$.

Figure 4-4 and Figure 4-5 outline that our framework was able to recover the correct number of clusters $k = 5$ from the data samples. By looking at the clusters with the majority of observations from the initial groups and matching them accordingly, we also observe that more than 91% of the observations a classified correctly

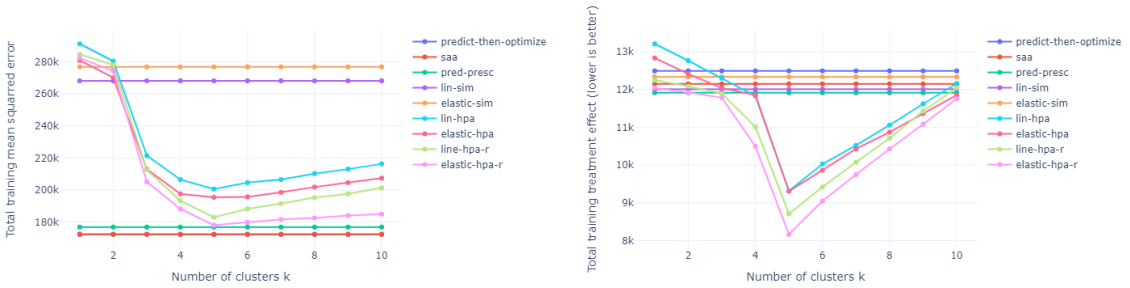


Figure 4-4: For top models: in-sample MSE on Synthetic Simulation (on left), and in-sample Treatment on Synthetic Simulation (Lower is better on right.

Model	Testing MSE	Out-of-Sample R^2	Prescriptive Cost	Difference vs Baseline
`predict-then-optimize`	184771	76.9%	11896	0%
`saa`	184771	76.9%	11189	-6%
`lin-sim`	285815	64.3%	10876	-9%
`elastic-sim`	274931	65.6%	10541	-11%
`pred-presc`	208937	73.9%	10350	-13%
`lin-hpa`	200743	74.9%	9380	-21%
`elastic-hpa`	195601	75.5%	9250	-22%
`lin-hpa-r`	183144	77.1%	8622	-28%
`elastic-hpa-r`	**178244**	**77.7%**	**8351**	**-30%**

Table 4.2: Results of the Synthetic Data simulation (Minimization) - Testing

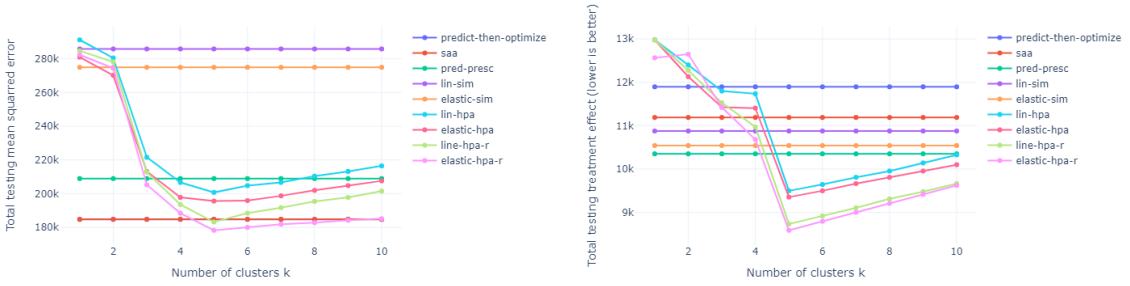


Figure 4-5: For top models: out-of-sample MSE on Synthetic Simulation (on left), and out-of-sample Treatment on Synthetic Simulation (Lower is better) on right. Our framework outperforms other benchmarks and recovers the correct number of clusters.

(i.e. in the correct group) with the HPA method, and that without imposing any tree structure.

Tractability

The average run-time of each iterations of the `elastic-hpa-r` algorithm for select values of n, k and p on this problem is as follows:

n	k	p	q	Run-time (seconds)
1,000.00	5	1	100	0.2218398
1,000.00	5	1	1	0.0000867
1,000.00	5	100	1	0.2204847
1,000.00	10	1	100	0.0550053
1,000.00	10	1	1	0.0000214
1,000.00	10	100	1	0.0550880
1,000.00	100	1	100	0.0005463
1,000.00	100	1	1	0.0000002
1,000.00	100	100	1	0.0005596
10,000.00	5	1	100	21.6410225
10,000.00	5	1	1	0.0084438
10,000.00	5	100	1	21.7576244
10,000.00	10	1	100	5.5379600
10,000.00	10	1	1	0.0021578
10,000.00	10	100	1	5.5038581
10,000.00	100	1	100	0.0549852
10,000.00	100	1	1	0.0000218
10,000.00	100	100	1	0.0553235
100,000.00	5	1	100	2,223.2307948
100,000.00	5	1	1	0.8653242
100,000.00	5	100	1	2,241.9927715
100,000.00	10	1	100	539.9380922
100,000.00	10	1	1	0.2155644
100,000.00	10	100	1	541.8205072
100,000.00	100	1	100	5.5351108
100,000.00	100	1	1	0.0021908
100,000.00	100	100	1	5.5895802

Table 4.3: Average run-time of each iterations of the `elastic-hpa-r` algorithm for select values of n, k and p on the synthetic experiment

We further visualize these results in 4-6 in both linear and logarithmic scale. Because the dependency of the run-time in p and q only appears through $p+q$, we only plot the results for different values of k and different values of $p+q$.

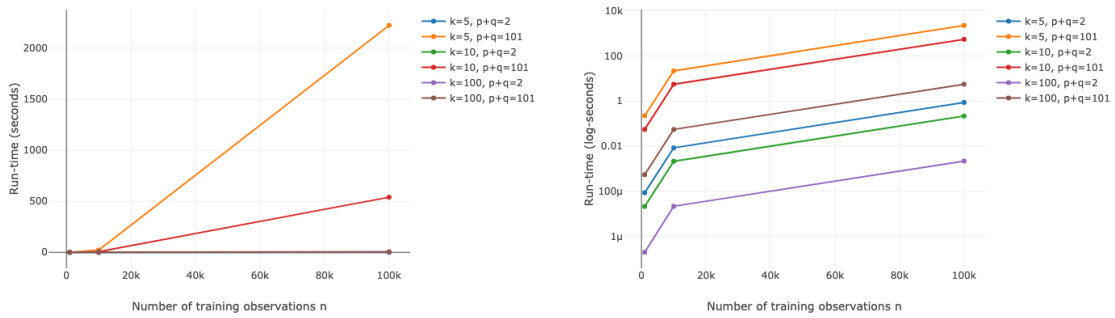


Figure 4-6: Run-time of the `elastic-hpa-r` on the synthetic experiment in linear scale (a), and logarithmic scale (b).

The details on the hardware used can be found in Appendix §C.4.

Typically, the algorithm converges for $T_{max} \approx 10$ iterations. For comparison, the smallest training and prescription run-time is for `predict-then-optimize` , with respectively $5.3e{-}4$ seconds for $n = 1,000$, $6.4e{-}2$ seconds for $n = 10,000$ and 4.8 seconds for $n = 100,000$.

This reinforces the conclusion of §4.4 on the theoretical tractability of the HPA framework.

Assortment Case Study

The dataset consists of 25,000 observations, each one corresponding to a product sales at a particular date with:

- **Products Features**: ex. type of product, brand, whether it is perishable or not.
- **Pricing Decisions**: price, whether it is promotion or not.
- **Time-series Data**: historical sales for the product.
- **External Indicators**: market features, competitors' prices.

- **Decision Variable**: whether to display the product or not.
- **Target Variable**: number of units sold for this product SKU at this time t.

The goal is to select products for the display of the retailer to maximize revenue. We assume that the target variable is representative of the demand as long as the product is displayed (no inventory limitation) and we ignore the cannibalisation effect of the products (i.e. the product demand only depends on the features and not on the selected portfolio). We consider here the case of a linear cost function, i.e. $c(y, z) = -y$ (since it is a maximization problem), which makes Steps (4.6) and (4.7) of Algorithm 4 highly tractable. Specifically, Equation (4.6) has the same complexity as a simple linear regression, and (4.7) is a linear program. Note that this is a constrained optimization problem, since the shelf space is limited. Again, we split the dataset into training (60%), validation (20%), and test (20%). No additional data-processing or data imputation is performed and all other methods and hyper-parameters are the same as the previous experiment.

We summarize the results in Table 4.4.

Model	**Out-of-Sample R^2**	**Prescriptive Cost**	**Improvement From Baseline**
`predict-then-optimize`	74 %	1404.93	0%
`saa`	74 %	1545.79	10%
`lin-sim`	70%	1560.04	11%
`elastic-sim`	69%	1576.66	12%
`pred-presc`	73 %	1580.13	12%
`lin-hpa`	71 %	1508.7	7%
`elastic-hpa`	75 %	1597.22	14%
`lin-hpa-r`	79 %	1641.99	17%
`elastic-hpa-r`	**83 %**	**1643.42**	**17%**

Table 4.4: Results for the Invetory Dataset (Maximization)

On this real-world experiment, the linear HPA outperforms its competitors in both out-of-sample accuracy and prescriptive cost. We might reasonably assume that the improvement in predictive accuracy comes mainly from the fact that the clustering allows the model to capture non-linear relationships (that are linear within each cluster but not linear over all the dataset), but that the improvement in the predictive cost comes from performing the clustering, the prediction and the prescription jointly instead of sequentially.

4.6.2 Tree-based Formulation

In this section, we move beyond linear models and use Classification and Regression Trees within the HPA method, using Algorithm 5.

We benchmark the tree-based HPA (which we refer to as `cart-hpa-r` and `cart-hpa` respectively depending on whether we use multiple restarts or not) on a diabetes case-study, against predict-then-optimize based on point estimate with CART (`predict-then-optimize`). sample average approximation (`saa`), tree-based predictive-prescriptive (`pred-presc`), and prescriptive trees (`presc-trees`, [24]). These methods are the same as in §7.1, but using a CART Tree instead of a Linear Regression. For the Prescriptive Trees, we use the Interpretable AI (https://www.interpretable.ai/) implementation. All used hyper-parameters can be found in §C.3.

Diabetes Case Study

The dataset is proprietary and consists of 100,000 observations, with patient-level data with:

- **Patient Characteristics**: age, sex, medical history, BMI, vitals.
- **Previous Treatments**: drugs previously taken, number of visits to care centers, treatment history.
- **Decision Variable**: treatment given at this time t (11 possible treatments).
- **Target Variable**: average blood sugar levels.

The goal is to prescribe an optimal treatment at time t given patient's characteristics and the previous sequence of treatments in order to minimize his average blood sugar levels. Note that interpretability is key is this experiment since doctors need to understand the decision-making process to prescribe the treatment to real patients. We summarize the results in Table 4.5. We also report run-time of our algorithm in Appendix §C.4.

The `cart-hpa-r` Framework significantly outperforms all other prescriptive methods in terms of prescriptive cost, with a 14% improvement from the standard paradigm

Method	Out-of-Sample R2	Prescriptive Cost	Improvement From Baseline
`predict-then-optimize`	75%	13510.1	0%
`saa`	75%	12980.95	4%
`pred-presc`	73%	12624.8	7%
`presc-trees`	**85%**	11682.5	14%
`cart-hpa`	71%	13058.97	3%
`cart-hpa-r`	82%	**11623.18**	**14%**

Table 4.5: Results for the Diabetes Dataset (Minimization).

predict-then-optimize and 7% improvement from the state-of-the-art predictive-prescriptive approach. Only Prescriptive Trees (`presc-trees`)) are on par with HPA in terms of prescriptive cost, and outperform it in terms of out-of-sample accuracy. However, the HPA Framework can accommodate constraints and continuous treatments, which is not the case for Prescriptive Trees.

In this example, the prediction and the prescription can both be represented with a simple decision tree, explaining how the patients are clustered into groups, and how decisions are made based on their characteristics. This ensures the model is interpretable and easily explainable to the medical doctors, which are in fact the decision makers in this case study.

By delving into these clusters, we observe that the grouping happens mainly over 4 features: patient's age, sexe, BMI and number of previous visits. The splits separates the BMI into 3 categories: Low, Average, High, the age groups into less than 30 years old, between 30 and 69 years old, and above 69 years old. The split on age only happens for patients in one of the two first age groups ($\leq$ 69 years old), while the cutoff for the previous visits is 5 visits. Which results in a total of 30 intepretable groups. Within each cluster, a different predictive tree model is trained, which splits on drug count and previous treatments to predict sugar level and prescribe an optimal next treatment. This example illustrates the claim that the HPA framework is interpretable when used with interpretable ML methods.

4.7 Extensions

We discuss in this section two potential extensions of the HPA framework. The first one regarding the probabilistic assignment of observations to clusters, and the second about the retraining of the model when new training data is available or when we want to prescribe a treatment for a new data point.

4.7.1 Probabilistic Cluster Assignment

Our framework summarized in Equation (4.5) assumes that each observation is deterministically assigned to one and exactly one cluster. One possible extension is to relax the integer condition on the assignment $\boldsymbol{u}$ similarly to [10], thus making the assignment probabilistic. $\boldsymbol{u}$ would then represent the probability of assignment instead of the assignment itself, resulting in potentially more robust, tractable solutions. Some additional modeling is however needed to decide on the assignment of unseen data and to control the averaging effect for problems where the predictions are widely different across clusters that are close given a chosen distance metric, for example the ℓ_2-norm which is independent of the HPA objective function.

4.7.2 Retraining of the Model

When new training data is available or when we want to prescribe a treatment for a new data point, it is not always necessary to re-solve Problem (4.5) from scratch. Two observations can be made in that regard: the first one is that the iterative approach described in Algorithm 4 can be used with the new objective function with the new training data and/or the new data points for which we want to make a prescription. By setting the starting parameters $\boldsymbol{u}^{(1)}$ and $\bar{\boldsymbol{z}}^{(1)}$ to the previously trained value, and depending on how much new data is added, the algorithm would converge in very few iterations, i.e. for T_{max} small. The second option is to make prescriptions without retraining the model, by directly assigning the new observations to a cluster based on some distance metric or classification algorithm, and only solve the prescriptive problem with the resulting prediction. This option would however be considered

a predict-then-prescribe approach, limiting the positive effect of joint optimization, which is the premise of this chapter. Other extensions might include using gradient-based method in the case where the HPA objective is convex and the prescriptive problem unconstrained.

4.8 Conclusions

In this chapter, we introduced a novel method for prescription we refer to as Holistic Prescriptive Analytics (HPA) framework. This framework allows for the use for a large variety of predictive ML methods and prescriptive cost functions. Moreover, it accommodates constrained and continuous problems, while being scalable and providing a performance edge over the state-of-the-art prescriptive methods. It also preserves interpretability of the ML models that are being used within the framework.

The complexity analysis of the algorithm as well as the computation experiments, on synthetic data and on two real-world case studies provide strong evidence to these claims. Its fundamental holistic structure allows us to combine strong predictive accuracy within a powerful prescriptive framework thanks to the clustering and the fact the three tasks (clustering, prediction and prescription) are optimized jointly instead of sequentially.

Chapter 5

COVID-19: Prediction, Prevalence, and the Operations of Vaccine Allocation

5.1 Introduction

The COVID-19 pandemic has quickly changed the nature of society and resulted in massive loss of life, dramatically different societal interactions, and significant economic problems. All levels of government, institutions, and private organizations have rushed to respond to this pandemic. Nevertheless, the changing nature of the pandemic has made short- and long-term planning of activities difficult. Organizations are using trends in infection levels to make decisions, plan the use of resources, and craft policies. However, high levels of travel due to society's globalization, fluctuating government policies in different parts of the world, and social restlessness due to isolation have all contributed to highly unpredictable infection rates. So far the United States has experienced at least three waves of the pandemic due to potential new strains of the virus and changes in behavior of the population. Another challenge that has impacted organizations' ability to respond to the pandemic is the high number of asymptomatic carriers. Asymptomatic individuals are considered responsible

for over half of all COVID-19 spread ([99]), yet with limited testing capabilities it is hard to quantify how many asymptomatic cases exist. Ideally, population-wide testing would lead to the identification and isolation of asymptomatic carriers, but this is not a feasible solution with current testing capabilities.

This leaves governments and institutions unable to assess the true risk they face when making decisions. Understanding true infection levels has become particularly important as the United States has begun the roll out of different vaccines. As of June 2021, the United States Food and Drug Administration (FDA) has approved three COVID-19 vaccines, two of which require two doses to be fully effective ([158]). Especially during the initial roll-out, the supply of these vaccines has been limited. State-level governments need to decide how to allocate these vaccines within their state, including whether to prioritize first or second doses. If the state (or national government) prioritizes first doses, then it can achieve wide-ranging (but not complete) immunity. If it prioritizes second doses, then a small part of the population will obtain highly effective immunity, but the rest of the population will be fully susceptible to the disease. In order to make such decisions, local governments need to have an understanding of (i) how positivity rates might grow in the short and long term in their regions and (ii) what positive tests say about the true level of infection in the different regions of a state.

In this chapter, we tackle these issues by proposing a novel, end-to-end framework for case and death prediction. We then propose a model for determining through the detected cases what are the true cases, that is, the true prevalence of the disease. Finally, we optimize vaccine allocation among different regions in a fair way under operational constraints, based on the predicted prevalence of our model.

Contributions

1. **We introduce an ensemble method that accounts for different aspects of the COVID-19 case and death evolution**: We first develop four individual predictive models, each of which captures different aspects of the disease spread. These four models are then aggregated through Machine Learning (ML)

to create more accurate predictions that account for additional factors that drive changes in the pandemic. We demonstrate that the aggregate model's predictions are more accurate and robust to changes in the pandemic. We prove that the aggregate model's prediction error is lower than each individual model's in-sample and lower than at least one of the four models out-of-sample and that the aggregate model has lower variability than its individual components.

2. **We demonstrate accurate short- and long-term predictions for both cases and deaths**: We compare our model to the other models used by the Center for Disease Control (CDC), which are considered to be the top models in the country. We show that our models consistently perform among the best both in the short-term and long-term future. The model we propose in this paper has ranked 1^{st} for several months (in predicting both deaths and cases) and is consistently among the top 10 models out of more than 50 state-of-the-art models.

3. **We propose a prevalence method to estimate true disease spread**: We propose a method for determining the "true" case counts of COVID-19 in different regions (that is, states and counties across the United States). We test our method using data from the CDC's randomized serology testing.

4. **We introduce and study an optimization model for determining the distribution of different vaccines and discuss interesting insights**: The optimization model we propose captures first and second dose vaccine distribution while accounting for differences between counties (regions), population groups (e.g. age) and different vaccines (e.g. efficacy, time between doses). Using this model, we create recommendations for state-level governments and the corresponding counties and show insights into the structure of optimal vaccine allocation. We quantify the importance of fast vaccination, introduce a condition on when to prioritize first versus second dose vaccines and tackle the trade-off between area prevalence, exposure, vaccine efficacy and mortality rate when allocating vaccines to particular sub-populations. Interestingly, we find

that the US government's strategy of completely vaccinating one age group before moving to another is not necessarily optimal, especially when the level of exposure differs from one age group to another.

5. **We discuss the impact of this work through our collaboration with MIT and the CDC:** This work has been the outcome of a collaboration on the MIT COVID-19 Response System (MCRS). MCRS is a joint effort between the MIT Quest for Intelligence and Lincoln Labs in order to model the effects of returning to campus. The prediction and prevalence models in this paper were developed as part of the MCRS effort as accurate forecasts of local prevalence rates are crucial for understanding the appropriate degree of returning to campus. Furthermore, the models in this paper are used on the CDC website (under the name of `MIT-Cassandra`) to help the CDC and government entities understand and mitigate the spread of the pandemic. This end-to-end framework is summarized in Figure D-5.

5.2 Relevant Literature

There has been a renewed interest in the operations community in modeling epidemics and analyzing their impact on society. The literature on each of these topics is growing rapidly. In this section, we briefly discuss some of the most relevant literature related to each topic we touch upon in this chapter.

5.2.1 Predictive Models

In this chapter we introduce different predictive models that we then aggregate. The first model is a feature-based Markovian representation approach and is related to offline Reinforcement Learning (RL). While RL [147, 20] deals with learning in a dynamic environment when exploration is feasible, offline RL [108] tackles situations where experimentation is not feasible and learning is performed only from a fixed batch of transition data. The second model is a Nearest Neighbor Approach inspired

by the KNN algorithm [57, 58]. To the best of our knowledge, despite some early work by [166], there have not been as many applications of KNN to time series prediction problems until recently. The third model is a Deep Learning approach based on Recurrent Neural Networks (RNNs), specifically Bidirectional Long Short-Term Memory (LSTM) Networks. LSTMs were first introduced in the seminal work of [96] with a modification in [87] that led to their final form. The final prediction method we use is a generalized SEIRD (Susceptible, Exposed, Infectious, Recovered, and Deceased) model that can account for multiple waves of the pandemic, introduced in [131]. We refer the interested readers to [38] and the references therein, for a discussion on compartmental models and their extensions.

5.2.2 Aggregation Methods

In addition to constructing four different models for COVID-19 case and death prediction, we also aggregate these outputs into final combined predictions for cases and deaths. We refer the interested reader to [69] and [138] and the references therein for an in-depth discussion of ensemble methods for aggregation. [165] introduced stacked generalization, which can be seen as a more sophisticated version of cross-validation. [154] study the method of stacked generalization by combining models from different subsets of a training dataset and merging their predictions in a majority vote manner. [155] address two issues; the type of regularizer that is suitable to derive the higher-level model and the kind of attributes that should be used as its input. [143] study an ensemble of linear networks trained on different but overlapping training sets. The authors consider ensemble error and average error of individual predictors before obtaining the generalization error and they study convergence to the optimum under assumptions. [72] empirically evaluate several state-of-the-art methods for constructing ensembles of heterogeneous classifiers with stacking and they show experimentally that they perform comparably to selecting the best classifier from the ensemble by cross validation. [142] present a linear method that incorporates meta-features for improved accuracy in the aggregation process [137] studies why ensembling methods work well in terms of MSE. Our method departs from the existing literature since

it is the only method that at the same time (1) combines machine learning models with different structure to obtain the best of all worlds, (2) uses general machine learning models (instead of linear, voting e.t.c. models) to combine the predictions of the initial (base-0) models in a smart way and (3) is applied to a time-series problem. Moreover, our work provides novel provable guarantees on the robustness and the variance of the predictions. Compared to previous works our guarantees are for general distance functions and therefore can be applied to many well-known metrics such as the MSE, the MAE and more.

5.2.3 Prevalence Extrapolation

Apart from the methods we discussed above, a key contribution of this chapter is estimating the true prevalence of the disease using detected cases and deaths. Testing to identify detected cases is extremely useful; [12] provide a method they devised in designing a system to manage border crossings that they tested in Greece. [134] argues that even if the accuracy of available tests is low, testing a lot with less accurate tests can be useful. Due to these limited testing capabilities and disproportionately high levels of asymptomatic cases, extrapolating these numbers to true prevalence of the disease is a challenge. Most studies, including [132], [83], and [125], use infection fatality rate (IFR) to back-cast true infection from the recorded COVID-19 deaths. For a more thorough review of the IFR approach, we refer the interested reader to [122]. [110] propose a different method for modeling true infection by assuming that undetected individuals will have a different transmission rate than detected individuals. We differentiate ourselves from this literature by modelling the relationship between positive detection rates and testing rates rather than considering the difference in transmission rate. Furthermore the difference between the proportion of positive tests for an epidemic and true prevalence has been a known issue in the epidemiology community for a while. However much of the Bayesian estimates have focused on solving for misclassification error in the tests themselves, either false positives or false negatives [101, 19, 66, 74]. [121] consider the problem of how small sample size will affect the quality of prevalence estimate, but they evaluate sample size and disease

prevalence within the sample in order to determine what distribution to assume on the number of new cases (binomial or hypergeometric). [61] use a Bayesian network to detect whether an epidemic has started in a population based on surveillance medical data, by calculating a probability density function on the true prevalence. However none of these works address the filtering that happens with non-random testing and small samples, specifically the relationship between the prevalence of testing, likeliness to be tested and probability of actually having the disease. Our work quantifies this relationship and allows us to predict true prevalence for a variety of areas with different testing capabilities.

Vaccine Allocation

Finally, this chapter also studies the problem of optimal vaccine allocation to different regions and a heterogeneous population in a fair way. There has been a recent increase in the operations management literature on COVID-19 related work. [34] use a spatial epidemiology model to optimize targeted lockdown policies in different neighborhoods of a city. [59] propose a dynamic program for optimal hospital care scheduling to reduce the strain on the health system by prescribing optimal care for individual COVID-19 patients. [32] use system dynamics and time series modelling to model short and long term bed capacity demand. Similarly, [77] use epidemiology and ML to target lockdowns based on clinical severity risk. Nevertheless, very few papers have explored vaccine allocation for COVID-19. [133] frames a generic epidemic where both preventative (vaccines) and corrective (antidotes) interventions are available. [119], [117], [145] and [156], focus on the influenza vaccine allocation problem. [116] take a modeling approach to analyze how quickly the population is vaccinated under different allocation policies for two dose vaccines in the presence of limited supply. Similarly, [114] use an epidemiological model to first forecast disease evolution and then analyze the performance of different resource allocation policies to minimize the number of new infections. [26] tackles the problem of COVID-19 vaccine allocation at country-level for the US using a system of differential equations named DELPHI ([109]) as the underlying truth for the progression of the pandemic under different

scenarios. These differential equations are then used in an optimization problem that is solved locally using an iterative algorithm. Our chapter considers the challenge of one- and two-dose vaccines and their allocation under fairness and other operational constraints.

5.3 Predicting COVID-19 Detected Cases

In this section we present a novel method that aggregates different predictive methods. Aggregation (ensembling) methods typically combine models that are structurally similar. Our method instead combines different models that each bring a different type of representation to the table. For each method there are different situations in which it will perform accurately or be prone to error. By strategically combining these methods that at times some overestimate and other underestimate the errors, we are able to create an aggregate model that has a strong performance in the majority of situations. We mainly focus on presenting our approach for predicting COVID-19 detected cases and deaths.

5.3.1 Notations

$N \in \mathbb{N}$ represents the number of regions we want to make a prediction for. $P \in \mathbb{N}$ represents the number of features used for the predictions, (e.g. lagged variables, temperature or mobility levels) and $T \in \mathbb{N}$ the number of time periods used in the data to train the models. In this work, time periods refer to number of days. $H \in \mathbb{N}$ represents the time horizon for the predictions, (i.e., we make predictions for time $t \in [T+1, T+H]$), $S \in \mathbb{N}$ is the number of outcomes to predict at each time step for each region, (e.g. if we are predicting deaths and cases jointly, $S = 2$). $\boldsymbol{X} \in \mathbb{R}^{N \times T \times P}$ represents the vector of features, where $\boldsymbol{X}_{i,t}$ is the feature vector for region i at time t and $\boldsymbol{Y} \in \mathbb{R}^{N \times T \times S}$ the outcome vector, and $\boldsymbol{Y}_{i,t}$ represents the vector of outcomes for region i at time t. For example, $\boldsymbol{Y}_{i,t,1}$ is the predicted number of cases for region i, at time t. Other outcome variables may include deaths and active cases. For simplification we use a lower case $\boldsymbol{y} \in \mathbb{R}^{N \times T}$ for a uni-dimensional

outcome, (e.g., $\boldsymbol{y}_{i,t} = \boldsymbol{Y}_{i,t,1}$). The goal is to learn f, the set of functions $f_{i,t}$ so that $\boldsymbol{Y}_{i,t} = f_{i,t}(\boldsymbol{Y}_{j,\gamma}, \boldsymbol{X}_{j,\gamma}, \forall j, \gamma \in [N] \times [t-1])$.

5.3.2 A Markovian-based learning approach: Minimum Representation Learning

Summary

COVID-19 evolution in different regions can be seen as the evolution of a dynamic continuous state space system. At each step t and for each region i, the system has features $\mathbf{X}_{i,t} \in \mathbb{R}^N$ that can include growth rates, cases, mobility, healthcare quality, and weather, among others. These features correspond to the states of the dynamic system. At each step, a region is at state $\mathbf{X}_{i,t}$, takes an action $A_{i,t} \in \mathcal{A}$ (e.g., a restrictive mobility measure), observes a cost $R_{i,t} \in \mathbb{R}$ (e.g., growth rates, number of cases, number of deaths), and then transitions to a new feature $\mathbf{X}_{i,t+\Delta t} = F(\mathbf{X}_{i,t}, a) \in \mathbb{R}^N$, with Δt a chosen time step (e.g. number of days). Each region at a given date corresponds to a set of features and, therefore, to a state of the dynamic system.

Model Formulation

We seek to construct from observed data $(\mathbf{X}_{i,t}, A_{i,t}, R_{i,t})_{i \leq N, t \leq T}$ a reduced representation of this dynamic system by learning a finite state space deterministic Markov Decision Process (MDP) representing the system accurately. Our approach is based on the Minimal Representation Learning algorithm (`MRL`) introduced in [18]. `MRL` aggregates the features into groups (regions of the feature space) that have similar cost and dynamics. It then maps each region of the feature space into a state of the reduced finite MDP representation. Figure D-1 illustrates this process. This aggregation allows the construction of a concise MDP model of the system that is easier to learn from data.

Actions as a reflection of changes in mobility `MRL` allows to effectively account for restrictive measures in the prediction by introducing *actions* on the COVID-19 dynamic system. To define this set of actions, we consider a *mobility index* that

quantifies the flux of people within a state across time. The underlying assumption is that a significant change in the value of that index corresponds to a generalized change in the behavior of the population, which is the consequence of a state-level government decision. We discretize the *change in mobility index* into a finite increasing set of level $\mathcal{A}$ of thresholds $\bar{a}_1 < \ldots < \bar{a}_{|\mathcal{A}|}$.

Variance reduction using randomization — Randomized-MRL model In this chapter, we introduce an extension of the `MRL` method that follows closely the spirit of the *Random Forests* algorithm introduced by [40]. We refer to this new version of the `MRL` as the *Randomized-MRL* model or `r-MRL`. `r-MRL` uses *random batch-sampling* from the set of paths as well as *bootstrapping* from the set of features in order to learn different representations of the MDP. Each MDP provides an estimate of the target and the final prediction consists of the aggregation of the individual predictions (e.g, the empirical mean and the median among others). Similarly to the Random Forests, `r-MRL` presents strong robustness properties. It is also more flexible, more accurate, and more scalable with respect to the dimension of the feature space.

The second method we describe next is a generalization of the traditional kNN algorithm modified to handle time series data.

5.3.3 A Nearest-Neighbor Approach: Similarity-Weighted Time-Series

Summary

The `KNN` method takes advantage of the similarity of conditions and trajectories across regions and time in order to predict the disease's evolution in the future. Our method generalizes the k-nearest neighbors approach ([56]) for time-series prediction. Instead of imposing any underlying structure, it explores the *similarity* between the observed time-series and its relation to the time-series we want to predict. To model this relationship, we applied an inverse distance metric along with a neighbor cutoff threshold as weights in order to determine relevant time series neighbors and demonstrate that the method performs very well on real data.

In particular, for a given region and time period, this method uses as features the recent trajectory of the growth rates of the disease, as well as other information describing the states' conditions at the given time (e.g. population density, average temperature, percentage of people vaccinated, etc). Assuming that at time t, c_t is the number of cases (or deaths), we define the growth rate GR_t as the ratio between two consecutive days, that is, $GR_t = \frac{c_t}{c_{t-1}}$. Our goal is to predict $Y_\tau = GR_\tau$, where $\tau \in [T+1, T+H]$ using the previous GR_t as well as other features. Next, we formalize the method by characterizing the weight function and the neighbor selection method.

Model Formulation

Let $(\boldsymbol{X}, Y)_{i,t}$ be the feature and target variable pair of a particular neighbor where $i, t \in [N], [T]$. We define a potential neighbor as the combination of a region $i \in [N]$ at a time $t \in [T]$, thus the total number of neighbors in our training space are $N \cdot T$. Given a region-time pair $j, \tau \in [N], [T+1, T+H]$ for which we want to make a prediction, a given distance function, $d(\cdot, \cdot)$ and a threshold $C \geq 1$, we define the set of neighbors $I_C(j, \tau)$ such that:

$$I_C(j,\tau) = \big\{(i,t) \in [N], [\tau - 1] : d(X_{j,\tau}, X_{i,t}) \leq C \min_{i,t} d(X_{j,\tau}, X_{i,t})\big\}. \tag{5.1}$$

Using the above set, we predict the target variable $Y_{j,\tau}$ using the following equation:

$$Y_{j,\tau} = \sum_{i,t \in I_C(j,\tau)} \frac{\frac{Y_{i,t}}{d(X_{j,\tau}, X_{i,t})}}{\sum_{i,t \in I_C(j,\tau)} \frac{1}{d(X_{j,\tau}, X_{i,t})}}. \tag{5.2}$$

Note that the threshold C makes the method more robust by excluding skewed data and the denominator normalizes the selected weights. Finally, the threshold parameter and the distance function $d(\cdot, \cdot)$ are tuned based on their prediction performance in a validation set that consists of the final days of the training data. We discuss the mathematical formulation of the `KNN` model in more details in the Appendix, including a discussion of Theorem 7 that establishes uniform almost complete convergence of the estimator. Next, we consider a Deep Learning method involving

RNNs that allows us to capture more complex structures.

5.3.4 A Deep Learning Approach: Bidirectional LSTM

Summary

In this approach, we employ a Deep Learning architecture based on the Dynamic Time Warping (DTW) Clustering Algorithm and a Bidirectional LSTM Network, that discovers hidden patterns related to the growth of COVID-19 deaths and cases. As a result, this method allows us to accurately predict future cases and deaths.

Model Formulation

The proposed architecture consists of two distinct components. The first component is the Dynamic Time Warping (DTW) clustering algorithm that creates clusters of states with similar growth rates. As different states may be in different phases of the pandemic, it is more effective to train separate models for each different cluster of states. More specifically, we use the DTW clustering algorithm with cases and death growth rates as features, in order to cluster regions with similar growth rates together. The second component is a Bidirectional LSTM Network. The final output of this model is the weighted average of the two individual networks. By using the Bidirectional LSTM network we increase the amount of information available to the network and improve the quality of predictions. We refer the reader to Appendix D for more details about RNNs, LSTMs, and the DTW distance.

The strength of the Bidirectional LSTM model lies in its ability to analyze and discover previously unobserved patterns in large amounts of sequentially dependent data. This makes it ideal for predicting COVID-19 cases and deaths. We trained and validated on four differently-sized architectures in order to find one that is appropriate. The architectures differ in terms of the number of layers and units. The complete block architecture of the system can be seen in Figure D-3.

Inspired from [47], we establish in Appendix §D.3 a generalization bound for the LSTM network. This bound holds asymptotically for the Bidirectional LSTM.

The last data-driven method we consider in the next subsection is an epidemiological model specifically designed to capture multiple waves of COVID-19, which we refer to as the `C-SEIRD`.

5.3.5 An Epidemiological Approach: Multi-peak SEIRD

Summary

The Multi-peak SEIRD (or Chained SEIRD (`C-SEIRD`)) is an epidemiology model that leverages existing knowledge about the progression of an epidemic. Most ML models are dependent on historical data in order to understand how the cases of COVID-19 will progress. The `C-SEIRD` model instead brings a structural understanding of the epidemic. The model we propose identifies when new waves are occurring and the timing of peaks. Our approach is based on the C-SEIRD model proposed in [131].

Model Formulation

The traditional SEIRD (Susceptible, Exposed, Infectious, Recovered, and Deceased) model is a compartmental epidemiology model and assumes parameters are static. Unfortunately, this means that it can only predict one peak. For a full description of the SEIRD model, we refer the reader to Appendix D. In order to account for multiple waves, we use a multi-peak `C-SEIRD` model. In the n-peak `C-SEIRD` model, the model considers n-waves of the disease, with each wave, w, starting at T_w and ending at T_{w+1}. The differential equations for the model are described by differential equations:

$$\frac{dS}{dt} = \frac{-\beta_w SI}{N}, \ \frac{dE}{dt} = \frac{\beta_w SI}{N} - \frac{E}{\alpha_w} \qquad \text{for } t \in [T_w, T_{w+1}] \tag{5.3}$$

$$\frac{dI}{dt} = \frac{E}{\alpha_w} - (\gamma_w + \mu_w)I, \ \frac{dR}{dt} = \gamma_w I, \ \frac{dD}{dt} = \mu_w I \qquad \text{for } t \in [T_w, T_{w+1}] \tag{5.4}$$

This structure allows the multi-peak `C-SEIRD` to model changing infection, recovery, and mortality rates while still using the fundamental structure of the SEIRD model.

Given the differential equations we described, the question becomes how to learn the change points T_w and the corresponding parameters for each wave. The multi-peak model provides a dynamic process of first learning the parameters of a given wave and then using those parameters to identify the wave changes. The key idea behind the change-point is that as long as all the data comes from the same wave, the prediction error of the model should be exchangeable. However, when a new wave starts, this no longer holds and we can leverage this change to quickly identify new waves using a martingale for detection. [131] provide bounds on how fast new waves are detected for the complex parametric structure of the `C-SEIRD`.

5.3.6 An Aggregate Predictive Method: MIT-Cassandra

Summary

So far we have discussed four different predictive methods that apply for each region $i \in [n]$, at each time period $t \in [T+1, T+H]$. This section will discuss how to aggregate these predictions in order to obtain one single "best" prediction. The American Centers for Disease Control and Prevention (CDC) uses a simple aggregation algorithm for its forecasts for both COVID-19 cases and deaths. This is an average of select models that are submitted to the CDC with an accuracy above a certain threshold that is chosen by the CDC. We will extend this idea of aggregating different models based on historical performances, but we will go one step further by using ML on an appropriate validation set in order to create the final prediction through an ensemble of the four proposed methods in this chapter.

Model Formulation

Let $\hat{y}_{i,t,m}$ denote the predicted outcome value (for simplicity, assume this represents the cumulative number of cases) for region i at time t made by model $m \in \mathcal{M}$. $\mathcal{M}$ is the set of all individual models that we want to aggregate. $y_{i,t}$ represents the outcome value for i at time t. We assume all models were trained up to a time $T_{val} < T$. We choose a class of ML functions $\mathcal{F}$. Our goal is to find for each region i, the best

aggregator f_i within this class that takes predictions $\{\hat{y}_{i,t,m}, \forall m \in \mathcal{M}\}$ and outputs a single prediction $\hat{y}_{i,t}$, i.e., $\hat{y}_{i,t} = \hat{y}_{i,t}^{agg} = f_i(\hat{y}_{i,t,m}, m \in \mathcal{M})$. We assume that for each individual model $m_0 \in \mathcal{M}$, the function $f_i(\hat{y}_{i,t,m}, m \in \mathcal{M}) = \hat{y}_{i,t,m_0}, \forall i, t$ belongs to the set $\mathcal{F}$.

We formulate the training problem, for $i \in [N]$, in terms of cases as the problem of minimizing the sum of absolute errors in problem (5.5).

$$\begin{aligned} \min_{f_i} \quad & \sum_{t=T_{val}}^{T} |y_{i,t} - f_i(\hat{y}_{i,t,m}, m \in \mathcal{M})| \\ \text{s.t.} \quad & f_i \in \mathcal{F}, \end{aligned} \tag{5.5}$$

When $\mathcal{F}$ is the class of linear functions, for example, we obtain problem (5.6).

$$\min_{\boldsymbol{\beta}_i} \sum_{t=T_{val}}^{T} |y_{i,t} - \sum_{m \in \mathcal{M}} \boldsymbol{\beta}_{i,m} \hat{y}_{i,t,m}|. \tag{5.6}$$

Notice that the simple average used by the CDC is a special case of this framework. Indeed if we denote as M the number of models selected by the CDC in their ensemble, then by setting $\boldsymbol{\beta}_{i,m} = \frac{1}{M}, \forall i \in [N]$ if model m has been selected by the CDC, $\boldsymbol{\beta}_{i,m} = 0, \forall i \in [N]$ otherwise, then we get a feasible solution to problem (5.6).

In the final model built for MIT and posted on the CDC website, we set $\mathcal{F}$ to be within several ML function forms: Regularized Linear Models, Support Vector Machines (SVM), Classification and Regression Trees (CART), Random Forests (RF), and XGBoost.

Analytical Results

In this subsection we show that the proposed aggregation yields more accurate and more robust results than the individual models do separately.

Theorem 4

1. **(In-Sample Predictions)** *The trained aggregate model in problem (5.5) has lower in-sample mean absolute error than each of its individual models. i.e.*

$\sum_{t=T_{val}}^{T} |y_{i,t} - f_i(\hat{y}_{i,t,m}, m \in \mathcal{M})| \leq \min_{m \in \mathcal{M}} \sum_{t=T_{val}}^{T} |y_{i,t} - \hat{y}_{i,t,m}| \leq \sum_{t=T_{val}}^{T} |y_{i,t} - \hat{y}_{i,t,m_0}| \quad \forall m_0 \in \mathcal{M}, \ \forall i \in [N]$. *This result can also be generalized to general distance functions.*

2. **(Out-of-Sample Predictions)**

(i) **Robustness:** *The trained aggregate model in problem (5.5) has a lower out-of-sample mean absolute percentage error than at least one of its individual models: that is,* $E[|y_i - \hat{y}_i|] \leq \max_{m \in \mathcal{M}} E[|y_i - \hat{y}_{i,m}|]$ *when* $\mathcal{F}$ *is the set of convex combinations.*

(ii) **Variance:** *The trained aggregate model in problem (5.5) has a lower out-of-sample variance than at least one of its individual models: that is,* $var(\hat{y}_i) \leq \max_{m \in \mathcal{M}} var(\hat{y}_{i,m})$. *This result can be generalized to absolutely homogeneous distance functions.*

The proofs can be found in Appendix §D.6. Theorem 4.1 guarantees that the aggregate model always has a better in-sample fit in terms of predictive error. Theorem 4.2.(i) and 4.2.(ii) show that the aggregation method reduces variability and also adds robustness to the final prediction. While these results compare the aggregate model to the worst performing model, note that these results are for out-of-sample predictions. As such, the worst-performing model is not known in advance and may be different for the worst expected value and the worst variability. The aggregate model ensures a minimum level of performance out-of-sample while being the best in-sample.

5.4 Results with COVID-19 Data

To evaluate the success of our models, we benchmark them against models selected by the CDC and made publicly available on the COVID-19 Forecast Hub. This comparison is discussed extensively in §5.7. In this section, we discuss the regional predictions we use to inform the prevalence and vaccination models that we introduce.

5.4.1 Data Sources and Features

We utilized multiple data sources in order to construct the final dataset that was then used to train the models we presented above. We collected cases and deaths related data, state-level social distancing policies data, global population mobility reports and weather data. More details on the data sources and on how the features are used can be found be in the Appendix D.7.

5.4.2 Model Predictions

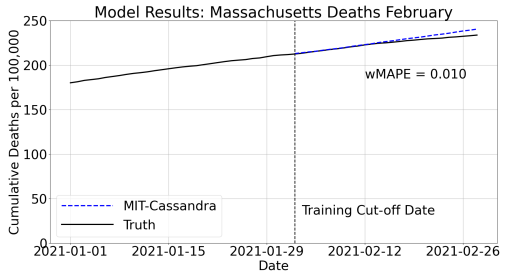
The component and aggregate models predict the cumulative number of deaths and cases for each state or county in the US. To measure prediction accuracy, we use the mean absolute percent error weighted by the true number of deaths or cases in each state (wMAPE, i.e. weighted Mean Absolute Percentage Error). Table 5.1 shows prediction accuracy for the aggregate model across different US regions, as defined by worldatlas.com, as well as across the country as a whole. We test the model on four months at different points during the pandemic. For each of these months, the models were trained and validated up to the last date of the previous month, and all predictions beyond that date were out-of-sample. Due to data issues, the states of Alaska and Hawaii (as well as the District of Columbia and other US territories) are excluded from the averages shown here.

Month	Metric	Midwest	Northeast	South	West	USA
June, 2020	Deaths	0.118	0.029	0.062	0.026	0.051
	Cases	0.427	0.716	0.462	0.305	0.440
September, 2020	Deaths	0.015	0.004	0.024	0.009	0.013
	Cases	0.210	0.087	0.159	0.367	0.205
November, 2020	Deaths	0.056	0.009	0.038	0.030	0.031
	Cases	0.111	0.267	0.387	0.356	0.262
February, 2021	Deaths	0.022	0.020	0.027	0.016	0.022
	Cases	0.499	0.525	0.477	0.219	0.438

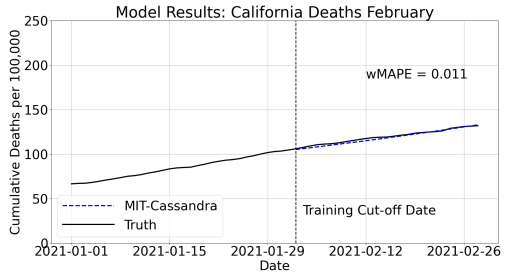
Table 5.1: wMAPE of the MIT-Cassandra model at various points during the pandemic in US regions. Regions defined by worldatlas.com. Alaska, Hawaii excluded.

For the purpose of estimating prevalence and optimally allocating vaccine doses,

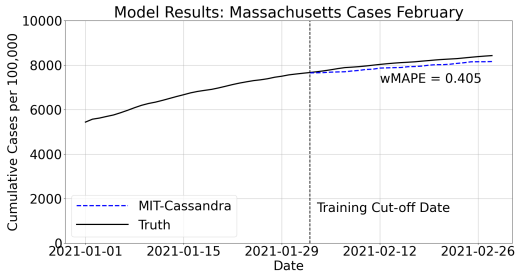
it is important to demonstrate that the aggregate model makes accurate predictions on the level of individual states as well. In Figure 5-1 we show cumulative deaths and cases alongside aggregate model predictions for Massachusetts and California for the month of February. The model is able to closely approximate the true evolution of cases and deaths in these states at this time: average percentage errors for these predictions over this month are also shown on the plots in Figure 5-1 below. It is worth acknowledging that the death predictions are more accurate than the case predictions. This is largely due to the way case counts are significantly more volatile, able to dramatically change their trajectory with little warning. Despite the errors, there is value in the case predictions to decision makers as they still generally trace the evolution of the disease. Understanding the broad evolution is crucial to long-term planning when there is uncertainty about the exact number of cases.



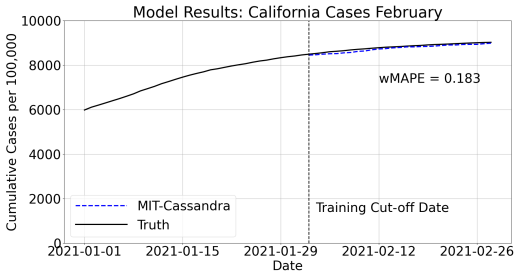
(a) February Deaths: MA



(b) February Deaths: CA



(c) February Cases: MA



(d) February Cases: CA

Figure 5-1: Statewide predictions (black) from the aggregate model alongside true death and detected case counts (gray) for the month of February in Massachusetts (left) and California (right).

The next section will discuss estimating the true prevalence of the disease, as opposed to the number of detected cases, which is essential for optimizing vaccine

allocation. This will require making predictions on individual counties, which will be discussed in the next section.

5.5 From Detected Cases to True Cases

Most COVID-19 predictive methods are trained on confirmed cases; however, decision makers care about determining the true number of cases. In this section, we propose a model that allows us to determine the true cases, also referred to as prevalence, from the detected cases. To determine this information, we use random testing serology data from the CDC.

5.5.1 Summary

COVID-19 testing is limited and is not done uniformly at random. Instead, it is biased towards the population that is more likely to test positive. We do not assume that the ratio between true cases and detected cases is constant; rather we assume a linear relationship between the probability of a person being infected knowing that she has not been tested and the probability of a person being infected knowing that she has been tested.

5.5.2 Model Formulation

For a random person in the population of interest, we denote $\mathcal{I}$ the random variable of being infected, $\mathcal{T}$ the random variable of being tested, $\mathcal{N}$ the random variable of not being tested ($\mathcal{N} = \bar{\mathcal{T}}$). In what follows we assume that tests for COVID-19 are 100% accurate, i.e., the probability of being positive if an infected person takes a test is 1 and the probability of being negative if a person who is not infected takes the test is 1 as well. Note that this assumption can easily be relaxed without any major changes to the model but is imposed for ease of exposition.

The linearity assumption can then be written as follows $P(\mathcal{I}|\mathcal{N}) = \alpha P(\mathcal{I}|\mathcal{T})$, where α is a constant capturing how likely it is for someone to be infected knowing

that one is not tested vs one is tested. This gives rise to Equation (5.7).

$$\begin{aligned} P(\mathcal{I}) &= P(\mathcal{I}|\mathcal{T})P(T) + P(\mathcal{I}|\mathcal{N})P(N) = P(\mathcal{I}|\mathcal{T})P(\mathcal{T}) + P(\mathcal{I}|N)\big(1 - P(\mathcal{T})\big) \\ &= P(\mathcal{I}|\mathcal{T})P(\mathcal{T}) + \alpha P(\mathcal{I}|\mathcal{T})\big(1 - P(\mathcal{T})\big) = P(\mathcal{I}|\mathcal{T})\big(\alpha + (1-\alpha)P(\mathcal{T})\big). \end{aligned} \tag{5.7}$$

We notice that $P(\mathcal{T})$ can be evaluated empirically by $\frac{\#\text{ Tests}}{\text{Population}}$ and $P(\mathcal{I}|\mathcal{T})$ can be evaluated by $\frac{\#\text{ Cases}}{\#\text{ Tests}}$. Hence, we obtain equation (5.8), which we can compute from the data and the predictions as long as we know the value of constant α.

$$\#\text{ Infected} = \frac{\#\text{ Cases}}{\#\text{ Tests}} \times \text{Population} \times \left(\alpha + (1-\alpha)\frac{\#\text{ Tests}}{\text{Population}}\right). \tag{5.8}$$

5.5.3 Evaluating α

In order to evaluate α, we use serology and random testing data from the CDC. This data assumes uniform random testing across a specific region at a particular time. Consequently we know the overall prevalence of this region at that time $P(\mathcal{I})$. By inserting this value into (5.8), we obtain an estimate of α for this region. The advantage of this method is that while the different probabilities are time-dependent, α is not. This implies that we can use this α for back-testing and future predictions. In Table 5.2, we show the estimation of α for some select states in the US. Note that some of these numbers may vary from the overall estimates, as they were performed on sub-sets of the state, e.g. the experiment for California was performed only on the Bay Area. Seroprevalence here is the estimate of the CDC of the percentage of the population that has been infected with COVID-19. Using this framework, we can

State	**Date**	**Seroprevalence**	**Detected**	**Population**	**% Detected**	**% Tested**	$\boldsymbol{\alpha}$
California	4/27/2020	1	45000	39510000	0.113	1.400	0.110
Connecticut	5/3/2020	4.9	29000	3562000	0.814	2.874	0.148
Massachusetts	5/15/2020	9.9	85400	6893000	1.238	6.528	0.111
Minnesota	5/12/2020	2.7	12500	5640000	0.221	2.142	0.244
Missouri	4/26/2020	2.7	6800	6110000	0.111	1.052	0.247
Pennsylvania	4/25/2020	3.2	41200	12800000	0.321	1.507	0.136
Utah	5/3/2020	2.2	4800	3206000	0.149	3.808	0.542

Table 5.2: Estimation of α for select states in the US based on the CDC Serology and Random Testing Data.

predict the true number of cases across regions instead of the detected ones. As is evident from equation (5.8), there is a linear relationship between α and the predicted prevalence ratio, $\frac{\#\text{ Infected}}{\text{Population}}$, where the slope of this relationship is determined by $P(\mathcal{T})$ and $P(\mathcal{I}|\mathcal{T})$.

It is worth noting that while the CDC serology dataset is one of most reliable in the United States, it also brings with it a set of assumptions and biases. First, while the study attempted to get a sample of seroprevalence tests nationwide, biases introduced based on the sampling process (who donated/submitted blood for labratory testing) might skew the results. Second, the seroprevalence survey doesn't not account for changing levels of antibodies over time. A complicating factor is that the serology tests only determine the presence of antibodies, thus patients who get Covid-19 multipletimes will not be accounted for. However we have since learned that antibodies last at least 3 months after the infection and these studies were conducted within the first four months of the pandemic. Third there is always the possibility in the tests themselves, introduced error through false negatives and false positives. Despite these factors, the CDC serology dataset is the closest to the ground truth of COVID-19 prevalence and is far less censored than the Covid-19 tests.

We show examples of true prevalence predicted by our method for different values of α for the states of Massachusetts and California on February 2, 2021. The value of α for both these regions was estimated to be 0.11. It is worth observing that the prevalence percentage varies by less than 2%, for a fairly wide range of α. Because the prevalence estimation is robust with respect to α, using the estimate of α from the serology data from the CDC, the subsequent estimate of true prevalence can be relied upon to be accurate. Note that the relationship between $P(\mathcal{I})$ and α while linear on a particular time, is not linear over time, as $P(\mathcal{I}|\mathcal{T})$ and $P(\mathcal{T})$ both vary over time.

By comparing Massachusetts and California, on February 2, 2021 a few interesting dynamics arise. First, it is worth noting that even though both states have similar α and true prevalence, the characteristics of each state are very different at this time. Specifically, the probability of being tested, $P(\mathcal{T})$, in Massachusetts (19.3%) is

approximately double the probability of being tested in California (9.7%). However the positivity rate, $P(\mathcal{I}|\mathcal{T})$, in Massachusetts (4.2%) is a little more than half that of California (7.5%). This counterbalancing effect explains why prevalence for the two states is so close. That being said, while the point prediction for the states is similar, because testing is more common in Massachusetts, the prevalence result is more robust. This is reflected in Figure 5-2 by the lower slope for Massachusetts compared to California. ‘

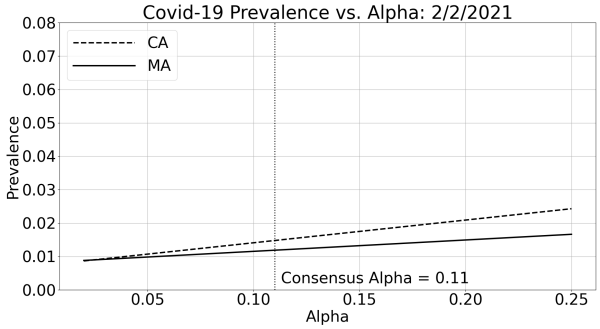
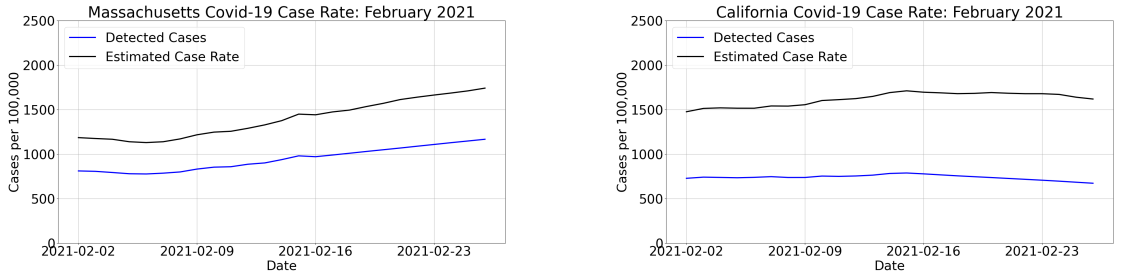


Figure 5-2: Possible true values of prevalence in Massachusetts and California on February 2, 2021. Consensus alpha refers to the estimated alpha from the CDC serology data in Table 5.2.

It is true that if the testing policies change, the α does change, so we have no guarantee that the previously evaluated α still holds. However, with new data, we can accurately re-evaluate this α, and could also assume a functional form for a time-varying α, which allows us to extrapolate for future changes in α. Note that one of the assumptions of this approach is that this α is constant between the time of evaluation and the time of prediction. However, if the testing policies change, the α does change, and we have no guarantee that the previously evaluated α still holds. However, with new data, we can easily and accurately re-evaluate this α, and could also assume a functional form for a time-varying α, which allows us to extrapolate for future changes in α.

5.5.4 Model Predictions

Using the value of α determined above, we can compare detected cases and estimated prevalent cases in these two states during the month of February 2021. This comparison is shown in Figure 5-3 with Massachusetts on the left and California on the right. As shown in Figure 5-2, at this time and α, the prevalence ratio is higher for California than for Massachusetts, and this relationship is reflected in this plot. California is much larger in population, but it is clear that the ratio between prevalent and detected cases is higher in California during this time.



(a) Cases in Massachusetts in February

(b) Cases in California in February

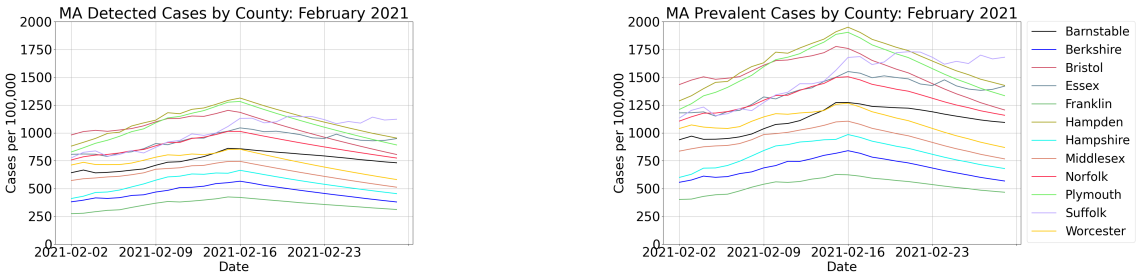
Figure 5-3: Detected (gray) and estimated prevalent cases (black) in Massachusetts (left) and California (right) during the month of February 2021. Total prevalence estimated using $\alpha = 0.11$.

For the purpose of allocating vaccines optimally to make the most impact, we need to estimate this true prevalence for each county. As an example, Figure 5-4 shows detected cases (left) and estimated prevalent cases during this span (right) for each county in Massachusetts, as our case study focuses on county-level allocation for this state. The next section discusses the formulation of this optimization problem and examines the prescribed vaccine allocations.

5.6 Application to Vaccine Allocation

Summary

In this section, we use the prevalence estimation and case and death predictions determined in §5.5.1 and §5.5.4, in particular Figure 5-4, to optimize the vaccine



(a) Detected Cases (b) Estimated Total Cases

Figure 5-4: Detected cases (left) and estimated total cases (right) in each Massachusetts county during the month of February 2021. Total prevalence estimated using $\alpha = 0.11$.

allocations across counties for different population types, vaccine types, and across first and second doses. The goal is to minimize the expected number of deaths or cases under capacity and fairness constraints.

Model Formulation

We consider the problem of a centralized planner who needs to allocate a finite number of 1-dose (K_1) and 2-dose (K_2) vaccines across J counties over T time periods. Each county is heterogeneous and is resided by I different population types. Population types differ from one another based on characteristics that drive the spread of the virus and the efficacy of the vaccine. The objective of the centralized planner is to minimize the expected number of deaths over T. To model this objective, we first define other population level parameters.

For an individual in population $i \in [I]$ and vaccine $k \in [K_1 + K_2]$, let $\boldsymbol{p_{1,i,k}}$ denote the probability of being immune to the virus after receiving only one dose of the vaccine and $\boldsymbol{p_{2,i,k}}$ denote the probability of being immune to the virus after receiving two doses. We assume that this probability is independent of the population type and simplify this notation to $\boldsymbol{p_{1,k}}$ and $\boldsymbol{p_{2,k}}$. Also, let $\boldsymbol{\hat{p}_{t,i,j}}$ denote the probability of being infected by the virus without any immunity at time t, in county j. This probability is given by our predicted prevalence (by using our prevalence model from §5.5.1 on our predictions from §5.3.6 to obtain the results shown on Figure 5-4 broken

down by age group). Once an individual from a population is infected, we let the mortality of the individual be given by $\boldsymbol{m_{t,i,j}}$. Finally, since we are solving a multi-period problem, we must model the evolution of the pandemic over the population. To accomplish this, we let $\boldsymbol{n^{0}_{t,i,j}}, \boldsymbol{n^{1,k}_{t,i,j}}, \boldsymbol{n^{2,k}_{t,i,j}}$ and $\boldsymbol{c^{0}_{t,i,j}}, \boldsymbol{c^{1,k}_{t,i,j}}, \boldsymbol{c^{2,k}_{t,i,j}}$ denote the number of susceptible people and the number of cases that received 0, 1 and 2 doses of vaccine k, respectively, at time t, in county j, from population i. (We only define $\boldsymbol{n^{2,k}_{t,i,j}}$ for $k \in [K_1 + 1, K_1 + K_2]$, as these represent the 2-dose vaccines.) Finally, we let $\boldsymbol{v^{1,k}_{t,i,j}}$ and $\boldsymbol{v^{2,k}_{t,j}}$ denote the integer decision variable of the central planner which denotes the number of *allocated* vaccines for the first and the second dose to different counties, population types, and time. Note that the susceptible population evolves based on the allocated vaccine according to the following equations:

$$n^{0}_{t,i,j} = n^{0}_{t-1,i,j} - \sum_{k=1}^{K_1+K_2} v^{1,k}_{t,i,j} - c^{0}_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \tag{5.9a}$$

$$n^{1,k}_{t,i,j} = n^{1,k}_{t-1,i,j} + v^{1,k}_{t,i,j} - c^{1,k}_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K_1], \tag{5.9b}$$

$$n^{1,k}_{t,i,j} = n^{1,k}_{t-1,i,j} - v^{2,k}_{t,i,j} + v^{1,k}_{t,i,j} - c^{1,k}_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K_1 + 1, K_2], \tag{5.9c}$$

$$n^{2,k}_{t,i,j} = n^{2,k}_{t-1,i,j} + v^{2,k}_{t,i,j} - c^{2,k}_{t,i,j}, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K_1 + 1, K_1 + K_2], \tag{5.9d}$$

Equations (5.9a) to (5.9d) represent the constraints describing the population dynamics when people of particular subgroups get infected and get vaccinated with one dose and two doses, respectively. For example, if an individual in the 1-dose population $n^{1,k}_{t-1,i,j}$ for population i in county j receives the second dose of vaccine k at time t, they are moved to the 2-dose population $n^{2,k}_{t,i,j}$ for the next time period. The constant vectors $\boldsymbol{c}^0$, $\boldsymbol{c}^1$ and $\boldsymbol{c}^2$ account for the estimated new infections and deaths that are removed from the pool of eligible candidates for vaccination.

The centralized planner's objective is to minimize the expected number of deaths

given by

$$\min_{\boldsymbol{v}} \quad \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}(n^{0}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j} + \sum_{k=1}^{K_1+K_2} n^{1,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k}) + \tag{5.10}$$

$$\sum_{k=K_1+1}^{K_1+K_2} n^{2,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k})), \tag{5.11}$$

The expected number of deaths in this objective from the vaccine allocation is divided into three components: (i) the expected number of deaths for populations that have received no vaccine doses (number of susceptible people $n^{0}_{t,i,j}$, times the probability of getting infected $\hat{p}_{t,i,j}$, times the mortality rate, or probability of dying knowing that one is infected $m_{t,i,j}$) plus (ii) the populations that received one dose and finally, (iii) those who received two doses. For these last two cases, we additionally account for the probability of not being immune, i.e. $1-p_{1,k}$ and $1-p_{2,k}$ respectively.

Our model formulation also allows us to account for fairness and other operational constraints. For example, centralized planners need to ensure that the allocation across regions is equitable. Furthermore, there might be capacity constraints across time for different regions. This can be done by ensuring that not too many vaccines of any type are allocated to a region or allocated in a given time period. Let $\boldsymbol{V_{max,k,t}}$ denote the maximum number of vaccines k available for the whole state at time t and $\boldsymbol{V_{min,t,j}}$ denote the minimum number of vaccines that need to be allocated to county j, at time t. Then we can ensure a fair allocation by enforcing the following constraints

$$\sum_{i=1}^{I}\sum_{j=1}^{J} v^{1,k}_{t,i,j} \leq V_{max,k,t}, \quad \forall t \in [T], \forall k \in [K_1] \tag{5.12a}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} (v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \leq V_{max,k,t}, \quad \forall t \in [T], \forall k \in [K_1+1, K_1+K_2] \tag{5.12b}$$

$$\sum_{i=1}^{I}(\sum_{k=1}^{K_1+K_2} v^{1,k}_{t,i,j} + \sum_{k=K_1+1}^{K_1+K_2} v^{2,k}_{t,i,j}) \geq V_{min,t,j}, \quad \forall t \in [T], \forall j \in [J], \tag{5.12c}$$

Equations (5.12a) and (5.12b) represent the capacity constraints for the 1-dose and

the 2-dose vaccines, respectively. Equation (5.12c) represents the fairness constraint, i.e., a minimum number of vaccines allocated to each county, at each time period. Additionally, we set $K = K_1 + K_2$ and $n^{2,k}_{t,i,j} = v^{2,k}_{t,i,j} = 0, \forall i, j, t$, for all 1-dose vaccines (for $k \in [K_1]$) (note this is without loss of generality). Furthermore, $c^{2,k}_{t,i,j} = 0$ from the definition of the 1-dose vaccines.

Putting this together gives us Formulation (5.13):

$$
\begin{aligned}
\min_{\boldsymbol{v}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}(n^0_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j} + \sum_{k=1}^{K}(n^{1,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k}) + n^{2,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k}))), \\
\text{s.t.} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \le V_{max,k,t}, \quad \forall t \in [T], \forall k \in [K], \\
& \sum_{i=1}^{I}\sum_{k=1}^{K}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \ge V_{min,t,j}, \quad \forall t \in [T], \forall j \in [J], \\
& n^0_{t,i,j} = n^0_{t-1,i,j} - \sum_{k=1}^{K} v^{1,k}_{t,i,j} - c^0_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \\
& n^{1,k}_{t,i,j} = n^{1,k}_{t-1,i,j} - v^{2,k}_{t,i,j} + v^{1,k}_{t,i,j} - c^{1,k}_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K], \\
& n^{2,k}_{t,i,j} = n^{2,k}_{t-1,i,j} + v^{2,k}_{t,i,j} - c^{2,k}_{t-1,i,j}, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K], \\
& n^0_{t,i,j}, n^{1,k}_{t,i,j}, n^{2,k}_{t,i,j}, v^{1,k}_{t,i,j}, v^{2,k}_{t,i,j} \ge 0, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K]. \\
& n^{2,k}_{t,i,j} = v^{2,k}_{t,i,j} = 0, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K_1]. \\
& v^{1,k}_{t,i,j}, v^{2,k}_{t,i,j} \ge 0, \quad \text{integer}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K].
\end{aligned}
\tag{5.13}
$$

It is important to note that in this formulation, the prevalence and the number of cases c are exogenous variables. This assumption can be relaxed by considering a dynamic program version of the formulation, as presented in Appendix D.10. However, in practice, this optimization formulation is frequently resolved, only implementing the here-and-now allocation decision, i.e., the allocation decision at time $t = 0$, or within a certain time interval with lag δ, from time $t = 0$ to time $t = \delta$. Then we use the output to update the prevalence and the expected cases, which allows us to resolve for the wait-and-see allocation decision ($t > 0$, or $t > \delta$). This framework provides a good, scalable approximation because high priority patients will remain high priority

regardless of the evolution of the overall prevalence, and the optimization is trying to vaccinate as much as possible as early as possible under the capacity constraints, which makes the update less impactful on the wait-and-see decisions. Moreover, the vaccination decisions made now do not impact prevalence until 14-days at least after vaccination, which makes the problem stable in terms of short-term input, allowing us to re-optimize over multiple time horizons.

To condense the formulation further and make it easier to solve, we use the following new notations: $\beta^{1,k}_{t,i,j} = \sum_{s=t}^{T} \hat{p}_{s,i,j} m_{s,i,j} p_{1,k}$ and $\beta^{2,k}_{t,i,j} = \sum_{s=t}^{T} \hat{p}_{s,i,j} m_{s,i,j} (p_{2,k} - p_{1,k})$ and $V^{1}_{t,i,j,max} = n^{0}_{0,i,j} - \sum_{s=0}^{t-1} c^{0}_{s,i,j}$ and $V^{2,k}_{t,i,j,max} = n^{1,k}_{0,i,j} - \sum_{s=0}^{t-1} c^{1,k}_{s,i,j}$ (note that these quantities are known and can be pre-computed). This re-formulation transforms the problem from minimizing the expected number of deaths to maximizing the expected reduction in deaths through vaccination. We argue that Formulation (5.13) can be rewritten into Formulation (5.14):

$$
\begin{aligned}
\max_{\boldsymbol{v}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(\beta^{1,k}_{t,i,j} v^{1,k}_{t,i,j} + \beta^{2,k}_{t,i,j} v^{2,k}_{t,i,j}), \\
\text{s.t.} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \le V_{max,k,t}, \quad \forall t \in [T], \forall k \in [K], \\
& \sum_{i=1}^{I}\sum_{k=1}^{K}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \ge V_{min,t,j}, \quad \forall t \in [T], \forall j \in [J], \\
& 0 \le \sum_{k=1}^{K}\sum_{s=1}^{t} v^{1,k}_{s,i,j} \le V^{1}_{t,i,j,max}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \\
& 0 \le \sum_{s=1}^{t} v^{2,k}_{t,i,j} \le V^{2,k}_{t,i,j,max} + \sum_{s=1}^{t} v^{1,k}_{s,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K], \\
& v^{2,k}_{t,i,j} = 0, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K_1]. \\
& v^{1,k}_{t,i,j}, v^{2,k}_{t,i,j} \ge 0, \quad \text{integer}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K].
\end{aligned}
\tag{5.14}
$$

Proposition 3 *Formulation (5.13) and Formulation (5.14) are equivalent.*

The proof can be found in Appendix §D.9. Formulation (5.14) is exactly our initial vaccine allocation problem, but notice that it does not include the evolution

constraints ((5.9a) to (5.9d)); these are replaced by upper bounds on the number of people eligible for vaccination in each sub-population. This results in a linear integer program with a **totally uni-modular matrix** defining its feasible region. As a result, (5.14) can be solved fast and efficiently with a simple linear relaxation.

Additionally, note that there are three main sources of uncertainty in this formulation: (i) the predicted prevalence $\hat{p}$: from both the prediction of the number of detected cases itself and from the choice of α, (ii) the mortality rate m, and (iii) the vaccine efficacy p_1 and p_2. While in this chapter, we focus on the nominal version, mainly because of the demonstrated high accuracy of our models, a robust re-formulation for Formulation (5.14) and Formulation (5.13) can easily be written and solved. See Appendix §D.11 for more details.

Intuition on the vaccine allocation policy

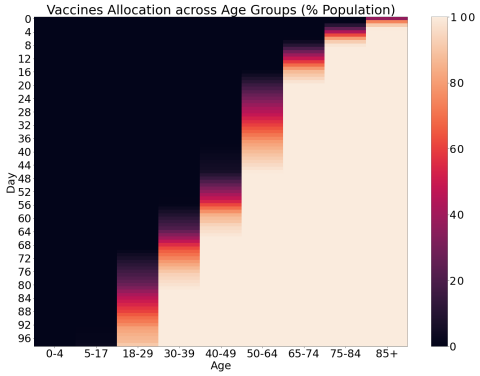
We notice, however, that the vaccination priority (between the different populations and due to the fairness constraints) across counties, populations, time, vaccine type, and first versus second-dose is determined depending on the values of the coefficients β (these are computed in advance, and depend on the prevalence, the mortality rate, and the vaccine dose efficacy).

Given these observations, we obtain the following takeaways: (i) $\beta^{1,k}_{t,i,j} > \beta^{1,k}_{t+1,i,j} \forall t, j$ and $\beta^{2,k}_{t,i,j} > \beta^{2,k}_{t+1,i,j}, \forall t, j$. This confirms mathematically that we should vaccinate any given population as soon as possible. That is, if the capacity constraint in Problem (5.14) is replaced by $\sum_{i=1}^{I}\sum_{j=1}^{J}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \leq V_{max,k}$, then the available capacity would be allocated to the earliest time regardless of how the disease prevalence and number of cases evolve. (ii) Note that $\beta^{1,k}_{t,i,j} - \beta^{2,k}_{t,i,j} = \sum_{s=t}^{T} \hat{p}_{s,i,j} m_{s,i,j}(2p_{1,k} - p_{2,k})$. For the same time t, population i, and county j, priority between vaccinating first and second doses (for 2-dose vaccines) is entirely determined by the sign of $2p_{1,k} - p_{2,k}$. If $2p_{1,k} > p_{2,k}$, then priority is given to vaccinating the entire population with the first-dose of the vaccines and only after that administering second doses. The opposite holds for $2p_{1,k} < p_{2,k}$. For example, for a vaccine with $p_{1,k} = 40\%$ and $p_{2,k} = 90\%$ efficacy (i.e. the first dose of vaccine k gives a 40% immunity to the virus, and the second dose

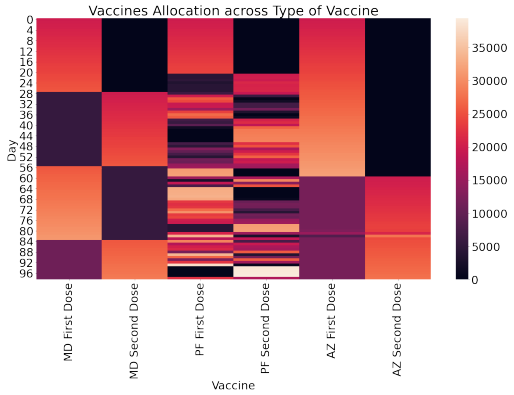
a 90% immunity), administering first and second doses of vaccines should be given the priority, contrary to some government policies. Note that different efficiencies, or level of immunity of the vaccine doses per population type, can be incorporated into the model by replacing $p_{1,k}$ by $p_{1,i,k}$ and $p_{2,k}$ by $p_{2,i,k}$ (representing different vaccine efficiencies for different population types i). This might change the order of priority for administering the first versus the second dose. However, there is not enough data on the breakdown of the efficiency by population type. (iii) We also observe several trade-offs between population i, county j, and vaccine type k. The proposed formulation allows us to answer questions such as whether to prioritize highly-effective vaccines in low-prevalence areas or less-effective vaccines in high-prevalence areas, or vaccinating the high-susceptible population in low prevalence areas or the low-susceptible population in high-prevalence areas. (iv) Finally, note that (5.14) does not assume a lag between first and second dose vaccines. Nevertheless, it can be directly incorporated by replacing the second population limit constraint by $0 \leq \sum_{s=1}^{t} v_{t,i,j}^{2,k} \leq V_{t,i,j,max}^{2,k} + \sum_{s=1}^{max(t-t_{lag},0)} v_{s,i,j}^{1,k}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K]$, where t_{lag} denotes the lag, which typically equals 21 days, but can vary across vaccines.

Results with COVID-19 Data

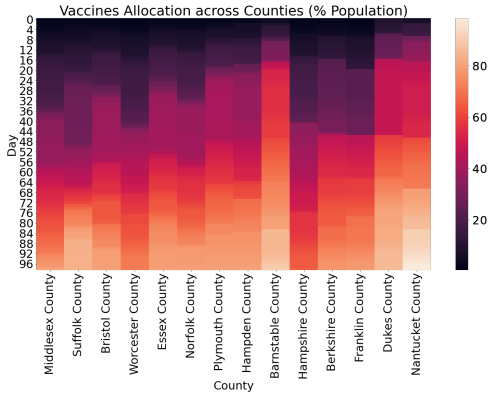
We test this approach on the different Massachusetts (MA) Counties, with three vaccines that we call vaccine `MD`, vaccine `PF` and vaccine `AZ`, with respectively immunity rates for 1 dose of $p_{1,1} = 40\%$, $p_{1,2} = 50\%$ and $p_{1,3} = 75\%$, and immunity rates for 2 doses of respectively $p_{2,1} = 90\%$, $p_{2,2} = 95\%$ and $p_{2,3} = 85\%$. We utilize the prevalence in each of the 14 MA Counties on February 1^{st} and predict the prevalence and the number of cases in the following 100 days regardless of vaccination. In particular, we split the population into nine age groups: 0-4, 5-17, 18-29, 30-39, 40-49, 50-64, 65-74, 75-84, and 85+, similar to the process of the CDC. These parameters, including the time-horizon are for illustrative purposes only. Note that for our 100-days ahead forecast at the beginning of the vaccination (January 2021), our aggregate model has a wMAPE of 29% only for cases. In this first experiment, we use as prevalence the output of the aggregate model, without using any age breakdown, i.e., we consider that for the same county, the prevalence across different age groups is



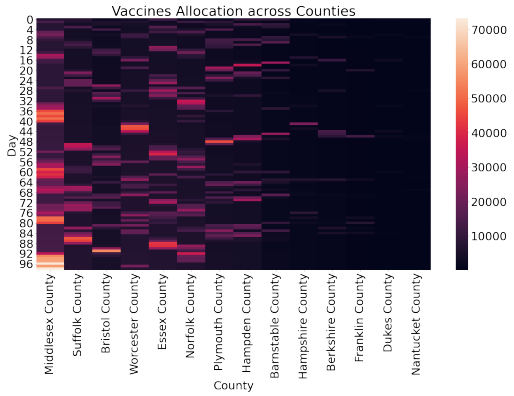
(a) Cumulative Optimal Vaccine Allocation by Age Group (% of Age Group Population).



(b) Optimal Vaccine Allocation by Vaccine Type.



(c) Optimal Vaccine Allocation by County (Represented in % of Population)



(d) Optimal Vaccine Allocation by County

Figure 5-5: Output of the Vaccine Optimization in MA Counties. The graph in (a) should be read top to bottom, and the bar for each age group becomes white when 100% of the population of this age group received at least one shot of any vaccine.

the same: $\hat{p}_{t,i,j} = \hat{p}_{s,j}\ \forall t, i, j$. We also assume that the mortality is time and county-independent, i.e., $m_{t,i,j} = m_i\ \forall s, i, j$. We use the values presented in Table D.7 in the Appendix, which has been estimated at the state-level. Additionally, we set the fairness constraint to be equal to half of what the population-proportional allocation would have given. i.e. $V_{min,t,j} = \sum_{k=1}^{K} V_{max,k,t} \times \frac{\text{population of county j}}{2\times\text{total population of MA}}\ \forall t \in [T] j \in [J]$ (for illustrative purposes, but centralized planner can set this fairness constraint to any value). We impose a minimum of 28 days-lag between a first and second dose vaccination for the `MD` vaccine, 21 for the `PF` vaccine, and 30 days for the `AZ` vaccine.

We show our results in Figures 5-5a to 5-5b. Figure 5-5a shows that the most at-risk population should be vaccinated first, while some people that are less at risk but are in high-prevalence areas should also be vaccinated in parallel but sparingly. Figures 5-5d and 5-5c show that high-prevalence areas should be prioritized, while maintaining a certain level of vaccination in other counties for fairness considerations. Figure 5-5c also shows that the areas that are heavily vaccinated first are not necessarily those that will reach 100% vaccination first.

Finally, Figure 5-5b shows interesting insights on the vaccine distribution, both among vaccines, but also between first and second doses. For vaccine `MD`, we see that the property $2p_{1,1} \leq p_{2,1}$ applies, which means second dose vaccines should be always prioritized (everything else being equal). The figure also shows clearly in the first two columns that as the population is vaccinated with a second dose as soon as they are eligible (28 days after first vaccination for `MD`, and 21 for `PF`), first doses are given only in an alternate fashion when the same population cannot receive a second dose yet. For the `PF` vaccine, $2p_{1,1} > p_{2,1}$, however these two values are very close, so although the prioritization should go to first doses, when a subgroup (age group i, in county j) is entirely vaccinated with a first dose, it is sometimes more effective to start a second dose vaccination for this subgroup instead of moving to another subgroup. This is why the alternation between first and second dose vaccines is less dominant than what we see for vaccine `MD`.

For `AZ`, where the second dose is a booster, we have $2p_{1,1} >> p_{2,1}$; we observe that the optimization model prioritizes the first dose vaccination over the second dose. This translates to few or no second doses of `AZ` being administered until the end of the experiment ($t > 60$).

Prevalence Breakdown by Age Groups

In this part, we perform a second experiment where we relax the assumption that prevalence is the same across age groups: $\hat{p}_{t,i,j} = \hat{p}_{s,j}\ \forall t, i, j$. Since few reliable data on the number of cases and deaths per age groups can be found ([26]), we estimate it using CDC exposure rates. The ratios of cases and deaths for different age groups

compared to the 5-17 year old age group according to the CDC are shown in Figure D-4 in the Appendix. The CDC computed these numbers from reputable sources such as NCHS and COVID-NET, and our findings are robust to changes on the order of 10%. These ratios can be interpreted as follows: all else equal, a person within the 75-84 year old age group, for example, is twice as likely to be detected positive for COVID-19 and 2800 times more likely to die from COVID-19. We normalize these results by population to extract the breakdowns on cases and deaths of COVID-19 by age group both in absolute value (number of cases per age group) and in terms of probability of being infected given membership in a particular age group. Below we show more details on this.

Denote $\mathcal{I}$ and $\mathcal{T}$ the events of being infected by COVID-19 and being tested for COVID-19, respectively. Also denote $\mathcal{B}_j$ as the event of belonging to age group $j \in [J]$.

Given the reference age group 5-17 and by replacing $P(\mathcal{I} \cap \mathcal{T}|\mathcal{B}_j)/P(\mathcal{I} \cap \mathcal{T}|\mathcal{B}_{5-17})$ with the ratios for cases from Figure D-4, denoted by $\gamma(j)$, for $j \in [J]$, we obtain Equation (5.15):

$$\begin{aligned} P(\mathcal{I} \cap \mathcal{T}) = \sum_{j \in [J]} P(\mathcal{I} \cap \mathcal{T}|\mathcal{B}_j)P(\mathcal{B}_j) = \\ \sum_{j \in [J]} \gamma(j)P(\mathcal{I} \cap \mathcal{T}|B_{5-17})P(\mathcal{B}_j) = P(\mathcal{I} \cap \mathcal{T}|B_{5-17}) \sum_{j \in [J]} \gamma(j)P(\mathcal{B}_j). \end{aligned} \tag{5.15}$$

We observe that $P(\mathcal{I} \cap \mathcal{T}) \times$ (population) at a given time t is exactly the number of cases, and that $P(\mathcal{B}_j)$ can be evaluated empirically by the ratio of the population in age group j divided by total population $\forall j \in [J]$. From this we can conclude that $\forall j \in [J]$:

$$\frac{\#\text{ cases in age group } j}{\text{total number of cases}} = \frac{P(\mathcal{I} \cap \mathcal{T} \cap \mathcal{B}_j)}{P(\mathcal{I} \cap \mathcal{T})} = \frac{\gamma(j)P(\mathcal{B}_j)}{\sum\limits_{l \in [J]} (\gamma(l)P(\mathcal{B}_l))} = \frac{\gamma(j) \times \text{population of } j}{\sum\limits_{l \in [J]} (\gamma(l) \times \text{population of } l)}. \tag{5.16}$$

Using census data for the population age breakdown by state, we compute these ratios in Equation (5.16) as displayed in Table D.8 of the Appendix. These ratios are far from uniform across age groups, and they vary from county to county. For example, in Barnstable County, our model indicates that the most exposed group is the 50-64 year old population with 24.9%, whereas in Middlesex County, it is the 18-29 year old with 27.4%. This allows us to have different levels of prevalence for each age group in Equation (5.14). The resulting vaccination strategy is very similar to the first approach in terms of time, county and vaccine type. However, it highlights an important difference from an age group perspective (see Figure 5-6).

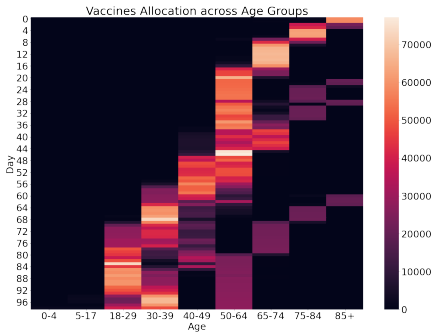


Figure 5-6: Optimal Vaccine Allocation by Age Group in Experiment 2.

Figure 5-6 shows, for example, that it is better to vaccinate some 75-84 years old, and even some 50-64 and 65-74 before completing the 85+ population. This distinction arises from two main factors: (i) Even though the mortality rate of 85+ years old is higher, if we account for the exposure in the disease prevalence per age group, the overall probability of dying from COVID-19 might still be lower than other age groups in some counties. (ii) Once we vaccinate the older population with the first dose in the high-prevalence areas, it is unclear without the optimization whether we should vaccinate the older population in low-prevalence areas with a first shot or with a second shot instead of a younger population in high-prevalence areas with a first shot. This is where the trade-offs discussed in the previous section play a role.

In conclusion, the optimization approach we discuss in this chapter, allows us to understand optimal vaccine allocation in a data-driven way. The broad strokes of the allocation policy align with widely held expectations of how vaccines should

be distributed. Namely, high risk age groups and high prevalence areas should be prioritized. But our proposed optimization approach more explicitly captures the tension between these two rules. For example, Figure 5-5a suggests that we need to begin to vaccinate lower risk groups if they are in high prevalence areas before finishing with the higher risk groups in low prevalence areas. The figure also shows how the two dose allocation policy should play out for different vaccines given their respective first and second dose protection. Figure 5-5b is particularly interesting in the context of [116]. In their comparison of second-dose hold back (i.e., at each time reserving half of the supply of vaccines for the second dose) versus second-dose release (i.e., not holding back second doses but switching between almost fully allocating first doses to fully allocating in second doses) under linearly increasing supply, the second-dose release policy dominates. While we do not explicitly model the policies as [116] does, our optimal policy has a similar structure to the second-dose release policy. For MD and PF vaccines, especially early on, we alternate between fully allocating the first dose and fully allocating the second dose, which reflects the second-dose release policy that [116] models and finds to be more optimal than trying to distribute both at the same time. However, our approach differs in that we consider prevalence of age groups and counties as well. This creates more intricacies in the allocation policy as the approach in this chapter balances trade-offs between prevalence and risk.

Most regions in the United States followed a phased roll-out of vaccines, starting first with essential and healthcare workers and then working through the population is decreasing order of risk (as evaluated by age and co-morbidity). These phases had clearly delineated starts, when subsections of the population all became eligible to receive the vaccine, and people were automatically assigned to receive their second dose (if their vaccine was two-dose) as soon as medically recommended. The policies were created at state level and applied throughout the counties within the state and without requirements for individuals to be vaccinated within their own state. While these policies had the benefit of simplicity and clarity, both in terms of the recommendation itself and the logic behind it, they are not represented in the outcomes of the optimal vaccine allocation policy. We can see the cost of the simplicity when we

evaluate how different the objective function of such a discrete policy is from optimal. There is a price to simplicity in terms of human lives and deaths when we prioritize interpretable, straightforward, enforceable allocations and this vaccine allocation optimization helps quantify that. In this regard the vaccine allocation optimization can be used as a crucial tool when determining vaccine policy. It allows decision makes to understand what is optimal and what are the costs of various relaxations, such as how many more deaths are expected when we maintain a strict hierarchy in terms of age regardless of prevalence in different counties or when we enforce the same policy across counties within a state to discourage people from travelling to be vaccinated sooner.

Limitations: Note that this vaccine allocation model assumes we control which counties and age groups receive vaccine doses, while in practice in the US, local governments can only assign vaccine stocks to distributors. There has been a far less granular control on age groups being vaccinated, and a significant fraction of the population got vaccinated in a different county they live in. This however has been implemented in different countries with more stringent vaccination roll-outs. The second limitation of the vaccination model in its current form is that it assumes an eligible individual who is offered the vaccine at time t will take it at this same time t, while experience has shown that some delay their vaccination, and other refuse it altogether, contradicting the 100% vaccine uptake. This can fixed by adding a probabilistic component to the formulation and through heavy incentives to vaccinate for the hesitant populations. Finally. our model does not currently account for reinfection directly. Although this can be done by further subdividing the vaccinated population and updating the estimated prevalence for these different subdivision accordingly.

5.7 Impact

This work has been the outcome of a collaboration on the MIT Covid-19 Response System (MCRS). MCRS is a joint effort between the MIT Quest for Intelligence and Lincoln Lab to model the effects of returning to campus. MCRS estimates these ef-

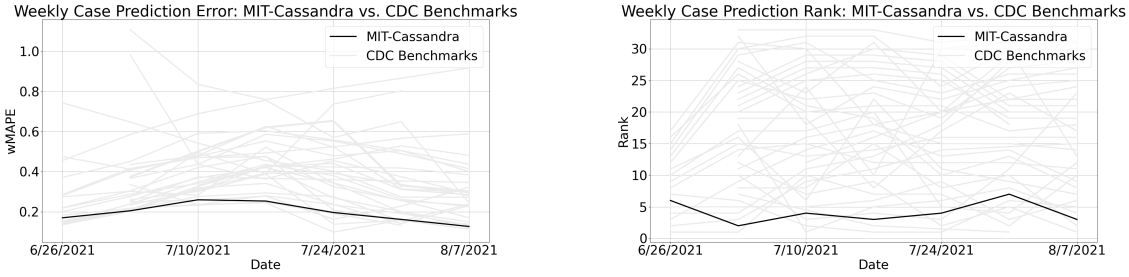
fects using de-identified data of campus mobility, with data access and usage overseen by MIT's Legal, Ethical, and Equity Committee and the IT Governance Committee. This work was funded and developed as part of the MCRS effort, as accurate forecasts of local prevalence rates are crucial to understanding the appropriate degree of returning to campus. The prevalence predictions have been important for reopening the institute by providing MIT senior administration with the forecasts they need to consider as they make policy decisions on the degree of allowing access to the campus for different groups (students, faculty, staff and visitors among others).

In addition, `MIT-Cassandra`, the predictive method discussed in this chapter is also part of the group of models that are used by the CDC to predict the cases and deaths of COVID-19 in different parts of the US. Of these models, our group's model performance consistently ranks among the top 10 models (out of more than 50 models) at different stages of the pandemic and is ranked 1^{st} overall in several months for both detected cases and detected deaths. These results are explored further in the following subsection. Also note that the proposed optimization formulation for vaccination is general, and it can easily be applied to different vaccination centers throughout the world.

5.7.1 CDC Model Comparison

We now evaluate the success of our models by benchmarking them against the rest of the models submitted to the CDC at several points during the pandemic. Our predictions, along with those from the other CDC models, are made publicly available on the COVID-19 Forecast Hub. Predictions from all models are uploaded weekly and forecast weekly deaths and cases up to 4 weeks out. We show that `MIT-Cassandra` consistently performs among the most accurate CDC models in terms of predicting cases one week in advance and is the most accurate model on average after excluding submissions that are purely ensembles of other CDC forecasts. Figure 5-7 shows accuracy (specifically wMAPE) and rankings for each model actively making predictions at this time. These accuracies are calculated for all predictions made for COVID-19 cases one week in advance. We see that `MIT-Cassandra` is consistently among the

five most accurate models by this metric. In fact, on average, the `MIT-Cassandra` model is the second-most accurate active CDC model by average wMAPE during this time span, and the most accurate by this metric excluding CDC models that are just ensembles of other CDC models. Additionally, we see in the left plot of Figure 5-7 that as the summer 2021 infection wave began in late summer 2021, the range of accuracies by these models widens as the predictive task became more difficult. `MIT-Cassandra` was able to continue making accurate case predictions throughout the entire period shown here.



(a) Prediction Error (wMAPE)

(b) Model Rankings

Figure 5-7: Prediction error (left) and model rankings (right) for all CDC models making COVID-19 detected case predictions during summer 2021. All comparisons are for predictions made one week in advance. MIT-Cassandra is shown in black and other CDC models in gray.

Model	Average wMAPE	Rank: Overall	Rank: Excluding Ensembles of CDC Models
MIT-Cassandra	**0.196**	**2**	**1**
UChicagoCHATTOPADHYAY-UnIT	0.200	3	2
Microsoft-DeepSTIA	0.215	4	3
USC-SI_kJalpha	0.226	5	4
IEM_MED-CovidProject	0.230	6	5
Geneva-DetGrowth	0.235	7	6
Karlen-pypm	0.250	8	7
JHU_CSSE-DECOM	0.259	9	8
COVIDhub-baseline	0.288	10	9
JHUAPL-Bucky	0.296	13	10

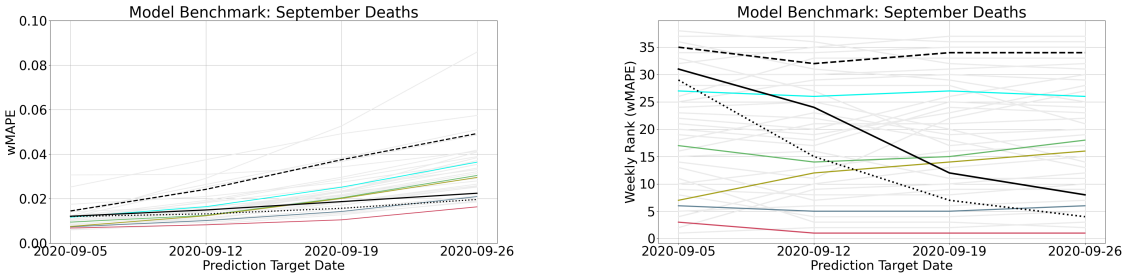
Table 5.3: Model accuracies for one-week ahead COVID-19 detected case predictions during summer 2021.

Performances Earlier in the Pandemic

Prior to publishing our results to the CDC, we were also able to extensively test our model performance against CDC models that were live earlier in the pandemic. We did this at several points during the pandemic to capture the model's ability to

make accurate forecasts during earlier stages of the pandemic. In Appendix §D.8.1 and D.8.2, we show the results for both cases and deaths predictions for the months of September 2020, November 2020, and February 2021.

Due to space limitations, we only show in this section the plots for September 2020 in Figures 5-8 and 5-9. For For the purpose of back-testing congruently, our methods are trained and validated on data up until the day the CDC predictions were posted. Predictions beyond that date are purely out-of-sample.



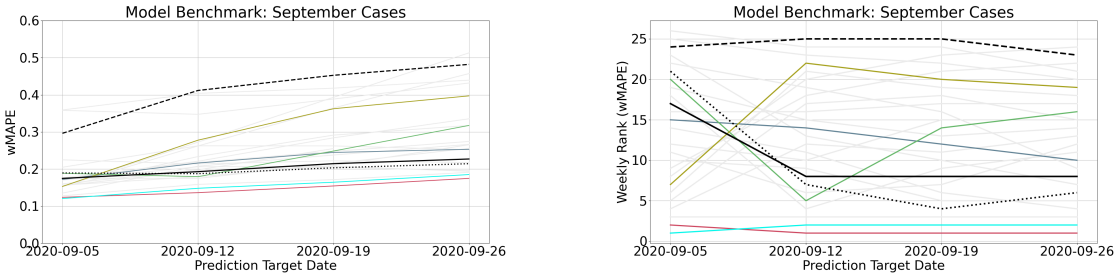
(a) September Deaths wMAPE

(b) September Deaths Ranks

Figure 5-8: Benchmarking the aggregate and best and worst component models (black) vs the top performing models of the CDC. The colored models are the top 5 by average rank to make predictions in September for cumulative deaths. Comparisons in terms of wMAPE (left) and overall ranking (right). All projections are out-of-sample predictions made 7 days before the first date displayed on the graphs.

First, we notice that our aggregate method performs very well. The aggregate model precisely identifies the most accurate component model and weights those predictions heavily. Figure 5-8 shows that the aggregate model is always much closer to the accuracy of the best component model than the worst component model. This clearly illustrates the benefits of our ensemble approach. It is also worth noticing the extremely small margins separating the top models for this month. In the first week of September, we observe that all of the models, including our aggregate model, are highly accurate and that 0.5% separates the majority of the models predicting deaths. Furthermore, in the month of September all models perform very well and the average error of our aggregate model is less than 15 deaths per state. We notice that only a few predicted deaths per state separate the best from the worst models in this month

because the number of deaths during that month were smaller relatively to others and more stable throughout the month. For February 2021 however, Appendix D shows the our model is ranked first for both the 1-week ahead and the 2-weeks ahead death predictions.



(a) September Cases wMAPE

(b) September Cases Ranks

Figure 5-9: Benchmarking the aggregate versus the best and worst component models in the ensemble (black) and a selection of CDC models. The colored models are the top 5 by average rank to make predictions in September for incident cases. Comparisons in terms of wMAPE (left) and overall ranking (right). All projections are out-of-sample predictions made 7 days before the first date displayed on the graphs.

As for the benchmark on cases predictions, When it comes to predicting cases, we see in Figure 5-9, showing the results both in terms of wMAPE (left) and rank (right) on September 2020, that the aggregate model is competitive with the best models used by the CDC at the time. In these months, the aggregate model is among the top-10 most accurate models for each time span except one (September 1-week ahead predictions) and is the most accurate model predicting cases two and three weeks out in November 2020 (See Appendix D). In general, it is clear that the ensemble method proposed in this chapter adds significant value and enables results in-line with or better than current top-performing models.

5.8 Conclusions

In this chapter, we proposed a holistic framework for predicting COVID-19 deaths, detected cases, and true disease prevalence. Among other applications, we use these predictions to optimize vaccine distribution. We presented four individual models

of different structure and showed how aggregation leads to greater accuracy and robustness. We applied the aggregate model to COVID-19 data to show strong performance in predicting short and long term case growth, which in turn has helped MIT and the CDC respond to the pandemic. The predictions were also used to create a probabilistic estimate for true prevalence, which was utilized in a vaccine allocation optimization. By varying prevalence in different sub-populations, we were able to present insights into the optimal allocation strategy and different trade-offs between groups. This holistic framework tackles the COVID-19 pandemic end-to-end from prediction to prevalence to prescription through vaccine allocation.

Chapter 6

Preventive Maintenance at OCP Maintenance Solutions: a Machine Learning Approach

6.1 Introduction

With modern advancements in hardware and software in industrial machines, and the availability of sensor data, machine learning (ML) has become increasingly important to reduce the cost of maintenance and ensure the reliability and the reactiveness of industrial systems. The use of ML in the context of failure detection is not recent. For example, in the context of electricity distribution, [91] predict failures using simple ML models such as linear regression and support vector machines, while more recent studies such as [97] use deep learning to predict the states of production lines, sacrificing intepretability and simplicity for higher accuracy. [43] show that, combined with the right intervention, predictive maintenance can save up between 18% and 25% in maintenance expenditures alone, with additional cost savings through reduced downtime.

In this chapter, we propose an end-to-end framework that leverages novel ML models to create data-driven tools for preventive maintenance from a descriptive, a

predictive, and a prescriptive perspective. We first start by proposing a natural language processing (NLP) model to create a dynamic failure modes and effects analysis (FMEA) from available text data. Then, we expand the FMEA using a probabilistic model to evaluate the likelihood of each failure mode and its causes, depending on the state of the system and based solely on sensor data. These two steps constitute the descriptive part of the analysis, allowing us to fully describe the current state of an industrial machine based on past observations. We then use a novel sparse and slowly varying regression model, first introduced in [21], to project this analysis to the future and predict the state of the system days and weeks in advance, and in an interpretable way. We empirically show that our methods substantially improve upon baselines in terms of accuracy, while remaining explainable to end-users and maintenance agents. Further, we discuss how these predictions are being used to schedule maintenance interventions, greedily or by applying the optimization framework proposed by [31].

Finally, we illustrate how the proposed framework was been successfully implemented and deployed by OCP Maintenance Solutions, subsidiary of OCP, the largest phosphates mining company in the world. This application led to significant performance improvement for a large number of internal and external clients, on a wide range of industrial machines, and to a measurably high business impact.

6.2 Relevant Literature

Recent years have seen a tremendous growth in the predictive maintenance field as one of the hot topics which come with an industry that relies heavily on data and models, referred to as industry 4.0, as well as the industrial Internet of Things (IoT). Since our work relates to both topics, we review recent work related to these areas.

FMEA is typically used as a method for identifying failure modes, thus improving the reliability of assets or components. In the state of the art, multiple studies and research work have been published to address the applications of the FMEA. For instance, [62] apply the FMEA approach to evaluate the impact of reliability-centered maintenance on a power generating system on hydraulic turbines. [11] present a

probabilistic modification of the FMEA model and establish its superiority over the conventional FMEA. [79] propose applying the FMEA approach for risk analysis of geothermal power plants. Recently, [82] develop a data-driven way to build an FMEA. [79] rely on both historical and operational data from the use stage of industrial machines and build the FMEA using deep learning methods. Our work differs from [79] in that it requires much less data and in that the FMEA is built in an interpretable way. Furthermore, we extend our dynamic FMEA to also account for probabilities of occurrence.

Vibration Analysis is a technique employed for rotating machines and manufacturing systems. Several studies have used vibration analysis for product data management (PdM). [36] suggest sensors positioning rules for PdM. [128] and [152] present experimental cases of studies on bearing failures. [52] propose indicators to describe the overall state of operation of a machine. We use vibration analysis as part of the feature engineering process.

Developing **ML models for PdM** is one of the key expansions of maintenance that comes with the rise of industry 4.0. PdM is technically intensive, engaging different technologies for maintenance, instrumentation, and information technology (IT). The practical implementation of a PdM policy faces two major problems: first, absence of any concrete statistical model for PdM [151] besides the general purpose machine learning regression models, e.g., LASSO Regression [153] and XGBoost ([49]), which are becoming increasingly popular in the industry but are yet to fully succeed; and second, requirements for sophisticated data acquisition and monitoring systems ([163]). [146] suggest a multiple classifier approach based on different horizons with two classes: faulty and not faulty. [102] use IoT vibration sensors with analysis capabilities to predict the remaining useful life of a product. The aforementioned solutions are industry-specific and require enormous investment for installation; therefore, their applicability to large organizations is limited. Moreover, the models developed using these approaches need modification when implemented on different companies within the same industry. In this chapter, we propose a general, low-cost machine learning framework for PdM.

6.3 Data & Pipeline

In this section, we describe the structure of the original (raw) data which we use, as well as the pipeline that the proposed framework applies to them.

6.3.1 Original Data Format

We collect tabular data $\mathtt{X}$ with the following structure:

- **Equipment type**: Equipment category, e.g., "ventilator." Let $\mathcal{M}$ be the distinct equipment types in the data with $|\mathcal{M}| = M$. We assume that all pieces of equipment of one particular type are identical in terms of their components, sensors, etc. We refer to this column in the data as $\mathtt{X}_{\mathrm{M}}$.

- **Equipment ID**: Equipment unique identifier. Let $\mathcal{N}_m$ be the set of pieces of equipment of type $m \in \mathcal{M}$ with $|\mathcal{N}_m| = N_m$. Then, we assume, without loss of generality, that the equipment ID takes values in the set $\mathcal{N}_m = \{1, \ldots, N_m\} := [N_m]$. We refer to this column in the data as $\mathtt{X}_{\mathrm{N}}$.

- **Component**: Component of the equipment. Let $\mathcal{C}_m$ be the set of components of equipment of type $m \in \mathcal{M}$ with $|\mathcal{C}_m| = C_m$. We refer to this column in the data as $\mathtt{X}_{\mathrm{C}}$.

- **Sensor**: Sensor of the component from which measurements are taken. Let $\mathcal{S}_{m,c}$ be the sensors for component $c \in \mathcal{C}_m$ of equipment of type $m \in \mathcal{M}$ with $|\mathcal{S}_{m,c}| = S_{m,c}$. We refer to this column in the data as $\mathtt{X}_{\mathrm{S}}$.

- **Sensor data**: Data obtained from the sensor. This includes both direct measurements, such as temperature, acceleration, and vibration, as well as spectral data, e.g., vibration spectrum. Any other sensor data can be used for the purpose of this analysis. We will refer to the sensor data as the **features** of the component. Let $\mathcal{D}_{m,c,s}$ be the set of features for sensor $s \in \mathcal{S}_{m,c}$ of component $c \in \mathcal{C}_m$ of equipment of type $m \in \mathcal{M}$. We refer to this column in the data as

$\mathtt{X}_{\mathrm{D}}$. We assume that $\mathtt{X}_{\mathrm{D}} \in \mathbb{R}^{|\mathcal{D}_{m,c,s}|}$ and let $\mathtt{X}_{r,\mathrm{D}[d]}$ extract feature $d \in \mathcal{D}_{m,c,s}$ from the sensor data of sensor s, component c, equipment type m.

- **Symptoms**: Free text of the description of the effects of the failure from subject matter experts after a maintenance intervention. We refer to this column in the data as $\mathtt{X}_{\mathrm{F}}$.

- **Timestamp**: Time measurements (sensor data and, if exist, symptoms) were taken. We assume that the measurements for all sensors and all components of each piece of equipment $n \in \mathcal{N}_m$ of type $m \in \mathcal{M}$ are taken at a fixed rate (e.g., one measurement per hour) and there is the same number of them. Then, we let, without loss of generality, $T_m \in \mathbb{Z}_+$ be the number of measurements taken for equipment of type $m \in \mathcal{M}$. We refer to this column in the data as $\mathtt{X}_{\mathrm{T}}$.

Denoting by R be the number of rows in $\mathtt{X}$, i.e., $|\mathtt{X}| = R$, then we refer to a specific row $r \in [R]$ as $\mathtt{X}_r$ and to a specific entry, e.g., row r in the "Component" column, as $\mathtt{X}_{r,\mathrm{C}}$. An example data point, corresponding to row $r \in [R]$ in the data table $\mathtt{X}$, is given in Table 6.1.

Data Field	Notation	Example
Equipment Type	$\mathtt{X}_{r,\mathrm{M}}$	Ventilator
Equipment ID	$\mathtt{X}_{r,\mathrm{N}}$	3
Component	$\mathtt{X}_{r,\mathrm{C}}$	Turbine
Sensor	$\mathtt{X}_{r,\mathrm{S}}$	2
Sensor Data (Temperature, Acceleration, Vibration)	$\mathtt{X}_{r,\mathrm{D}}$	(143.2, 2.4, 0.98)
Symptoms	$\mathtt{X}_{r,\mathrm{F}}$	Friction during turbine rotation
Time	$\mathtt{X}_{r,\mathrm{T}}$	12:23 08/12/2020

Table 6.1: An example data point that captures the format of the raw tabular data.

For simplicity in presentation, throughout the chapter, we work with simplified versions of the data, e.g., we focus on specific equipment types and only look at certain components or sensors. We explicitly state any simplifications when describing the corresponding part of the proposed framework.

6.3.2 Preliminaries

As discussed in the introduction, failure modes and effects analysis (FMEA) is a methodology to identity potential failure modes for an equipment and to assess the causes associated with each of these modes. The FMEA has a tree structure, first breaking down the equipment into its vulnerable components, and then identifying the observed failure modes for each of these components. Finally, the FMEA tries to identify the causes of each of these failure modes.

The FMEA tree is built based on some observed variables, e.g. temperature, vibration, etc. associated with the components. The general schema for the FMEA is represented in Figure 6-1. In some applications, the equipment can be further broken into asset and sub-assets.

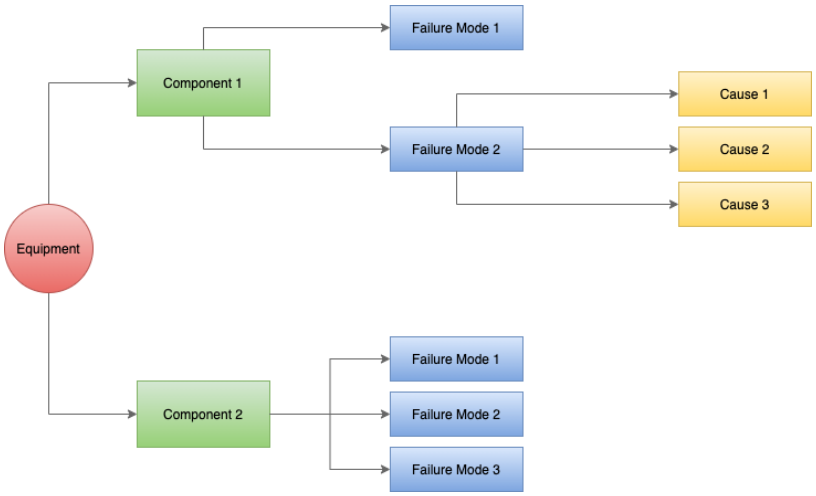


Figure 6-1: Failure Modes and Effects Analysis (FMEA) tree.

6.3.3 Data Pipeline

The proposed framework consists of three parts: a descriptive part, which, in turn, consists of a module that builds an FMEA tree in a data-driven fashion (Module 1a) and a module that predicts failure modes and their causes (Module 1b); a predictive part, which predicts *future* failures (Module 2); a prescriptive part, which, given predictions, performs maintenance scheduling (Module 3). We illustrate the pipeline in Figure 6-2.

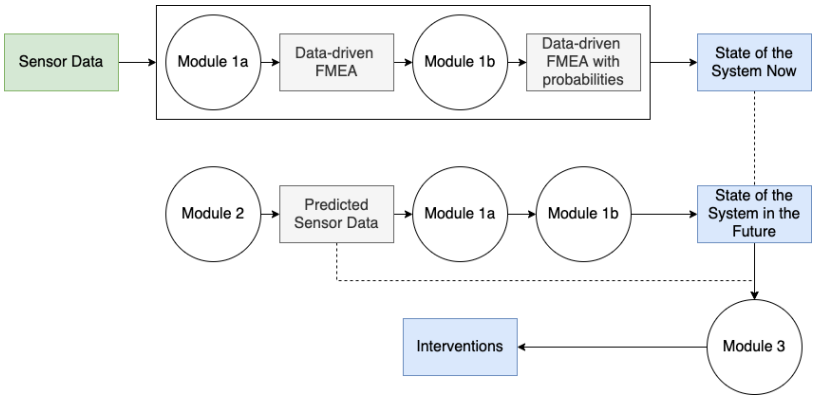


Figure 6-2: Flowchart of the Preventive Maintenance Framework. Input are in green, intermediary outputs are in grey, and final outputs are in blue.

Module 1a operates on a component level. For any given equipment $m \in \mathcal{M}$ and component $c \in \mathcal{C}_m$, Module 1a takes as input all corresponding symptoms that appear in the data, i.e.,

$$\mathcal{S}_{m,c} = \{[r, \mathtt{X}_{r,\mathrm{F}}] \mid r \in [R], \mathtt{X}_{r,\mathrm{M}} = m, \mathtt{X}_{r,\mathrm{C}} = c\}.$$

The output of Module 1a is the set $\mathcal{F}_{m,c}$ of possible failures for component c of equipment m and a function $f_{m,c} : \mathcal{S}_{m,c} \mapsto \mathcal{F}_{m,c}$ which maps symptoms (i.e., free text) to failure modes, as well as the corresponding FMEA tree.

Module 1b also operates on a component level. For any given equipment $m \in \mathcal{M}$ and component $c \in \mathcal{C}_m$, Module 1b takes as input the corresponding sensor, sensor data, and failure modes from the corresponding set of failures $\mathcal{F}_{m,c}$, i.e.,

$$\{[r, \mathtt{X}_{r,\mathrm{T}}, \mathtt{X}_{r,\mathrm{S}}, \mathtt{X}_{r,\mathrm{D}}, f_{m,c}(\mathtt{X}_{r,\mathrm{F}})] \mid r \in [R], \mathtt{X}_{r,\mathrm{M}} = m, \mathtt{X}_{r,\mathrm{C}} = c\}.$$

The output is a function which, given sensor data, estimates, the probability that a particular failure mode for component c of equipment m is happening, and for each sensor $s \in \mathcal{S}_{m,c}$, the probability that this failure is due to sensor s.

Module 2 operates on a sensor level. For any given equipment $m \in \mathcal{M}$, component $c \in \mathcal{C}_m$, and sensor $s \in \mathcal{S}_{m,c}$, Module 2 takes as input the corresponding sensor data

from all pieces of equipment and all timestamps, i.e.,

$$\{[r, \mathtt{X}_{r,\mathrm{T}}, \mathtt{X}_{r,\mathrm{N}}, \mathtt{X}_{r,\mathrm{D}}] \mid r \in [R], \mathtt{X}_{r,\mathrm{M}} = m, \mathtt{X}_{r,\mathrm{C}} = c, \mathtt{X}_{r,\mathrm{S}} = s\}.$$

The output is a set of models, each of which is tailored for a specific piece of equipment of type m, which predict future values of the sensor data of sensor s of component c.

Finally, Module 3 operates on a unified level, that is, it combines the outputs of all the aforementioned modules with maintenance data (available resources, constraints, etc.) to perform preventive maintenance scheduling.

6.4 Descriptive Part: Building a Data-Driven FMEA Tree

In this section, we describe Module 1a to build the two first layers of the FMEA tree.

6.4.1 First layer of the FMEA: Equipment-Components

Module 1a constructs the first layer of the FMEA tree, that is, the Equipment-Component layer. This can be fully obtained by listing all the observed combinations of an equipment (given by $\mathtt{X}_{\mathrm{M}}$) and its components (given by $\mathtt{X}_{\mathrm{C}}$) in the data-sets. For example, for an OCP “Ventilator" equipment, we extract two components, the “Pulley" and the “Turbine," as “susceptible components,” i.e., components which are potentially subject to failure. We obtain Figure 6-3.

6.4.2 Second layer of the FMEA: Component-Failure Modes

For a given equipment m, the second layer of the FMEA tree links each component c with its possible failure modes $\mathcal{F}_{m,c}$ (see Figure 6-1). The failure modes are not directly provided in the data and are not standardized; instead, they are indirectly

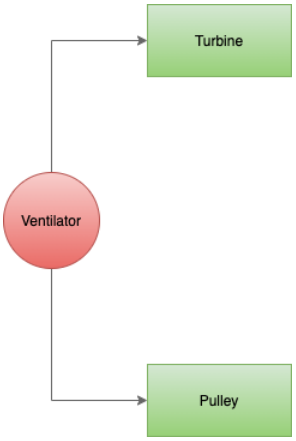


Figure 6-3: First layer of the FMEA tree. For illustrative purposes, the equipment is in red, the components are in green.

described through the observed symptoms in $\mathtt{X}_{\mathrm{F}}$:

$$\mathcal{S}_{m,c} = \{[r, \mathtt{X}_{r,\mathrm{F}}] \mid r \in [R], \mathtt{X}_{r,\mathrm{M}} = m, \mathtt{X}_{r,\mathrm{C}} = c\}.$$

Module 1a uses natural language processing (NLP) and clustering to map these symptoms to properly defined failures modes, using a function $f_{m,c} : \mathcal{S}_{m,c} \mapsto \mathcal{F}_{m,c}$. We describe the algorithm used in Module 1a in Algorithm 7.

Algorithm 7 Module 1a.

Input: Free-text symptoms column $\mathtt{X}_{\mathrm{F}}$, list of stop-words $\mathcal{L}$, minimum number of observations $N_{\min} \in \mathbb{Z}^+$, number of clusters $k \in \mathbb{Z}^+$.

Output: Mapping $f_{m,c} : \mathcal{S}_{m,c} \mapsto \mathcal{F}_{m,c}$.

1. **Stop-words**: we remove all stop-words from $\mathtt{X}_{\mathrm{F}}$. i.e., $\mathtt{X}_{r,\mathrm{F}} = \mathtt{X}_{r,\mathrm{F}} \setminus \mathcal{L}$, $\forall r \in [R]$. (The full list of stop-words can be found in Appendix E.1.)
2. **Stemming**: we stem the remaining words, i.e., we transform the words to their base root, for example if the word ends in "ed", we remove the "ed", if the word ends in "ing", we remove the "ing", if the word ends in "ly", remove the "ly". We also remove the prefixes and the plural forms. The words "rotation", "rotations", "rotated", "rotate" for example all become the same word: "rotat". i.e., $\mathtt{X}_{r,\mathrm{F}} = \mathtt{stem}(\mathtt{X}_{r,\mathrm{F}})$, $\forall r \in [R]$.
3. **N-grams**: we compute 2-grams. 2-grams are combinations of words that appear frequently next to each other and that should be considered as one term. For

example “New York" should be considered as one term, or in our case, “Failure Mode" refers to one single instance. This is done by computing the occurences of the terms appearing one next to another.

4. **Document-Term Matrix (DTM)**: we then create a matrix where the rows are the observations $r \in [R]$ and the columns are the terms resulting from the pre-processing steps $w \in \{[r, \mathtt{X}_{r,\mathrm{F}}] \mid r \in [R]\}$ s.t. the number of occurences of this term w is at least $N_{\min}$. Let W the number of such terms. Each entry of this matrix corresponding to row $r \in [R]$ and column $w \in [W]$ is equal to 1 if observation i contains the word w, and 0 otherwise.

5. **Term Frequency (TF) - Inverse Document Frequency (IDF) Weighting**: we weigh the entries of the matrix depending of the frequency of the word w in observation r noted $n_{r,w}$, as well as the frequency of the word w across all documents. We define the functions TF: $\mathrm{TF}(r, w) = \frac{n_{r,w}}{\sum_{w' \in [W]} n_{r,w'}}$ and IDF: $\mathrm{IDF}(w) = \log \frac{R}{|\{r \in [R] : n_{r,w} > 0|}$. We then multiply each entry r, w in the Document-Term Matrix by $\mathrm{TFIDF}(r, w) = \mathrm{TF}(r, w) \times \mathrm{IDF}(w)$. The goal of this step is to weight more the words that appear multiple times in the same text, and to weight less the words that appear too frequently across all the text, as they are less specific to the observation of interest.

6. **Clustering**: we then cluster the previous observations using k-means clustering with the cosine similarity distance. Similar observations, i.e., observations with similar text, get grouped together. Each group of “symptoms" $\mathcal{S}_{m,c}$ will represents a different failure model $\mathcal{F}_{m,c}$, resulting in the mapping $f_{m,c} : \mathcal{S}_{m,c} \mapsto \mathcal{F}_{m,c}$.

7. **Topic Extraction**: lastly, we give names to the failure modes for interpretability purposes. We compute for each word $w \in [W]$ the mean μ_w and the standard deviation σ_w across all the documents. Then, we computed the mean $\mu_{w,j}$ of each word within each cluster $j \in [k]$. We automatically name the detected failure mode j, i.e., the cluster, with the term that is most unique to it: $\arg\max_w \frac{\mu_{w,j} - \mu_w}{\sigma w}$.

8. **Return** $f_{m,c}$ and the names of the failures modes.

We assume, without loss of generality, that the rows $\mathtt{X}_{\mathrm{F}}$ are for the same equipment type m and the same component c; for the general case, we subset the data

(as shown in Section 6.3.3) and apply Algorithm 7 to all equipment type-component pairs separately. Concerning the inputs to Algorithm 7, we note that $N_{\min}$ and k are obtained by hyper-parameter tuning. After applying Algorithm 7, we only need to enumerate the combinations Equipment-Component-Failure Mode that appear in the (extended) data-set (whereby the failure modes have been added), i.e., all unique entries of $\{(m,\ c,\ f_{m,c}(s)),\ \forall(m,\ c,\ s) \in \mathtt{X}_{\mathrm{M}} \times \mathtt{X}_{\mathrm{C}} \times \mathtt{X}_{\mathrm{F}}\}$. We consequently get the second layer of the FMEA tree.

For our example of an OCP “Ventilator" equipment, we extract three possible modes: “Friction,” “Misalignment,” and “No Anomaly”. These failure modes are the same for both the “Pulley" and the “Turbine" components. We obtain Figure 6-4.

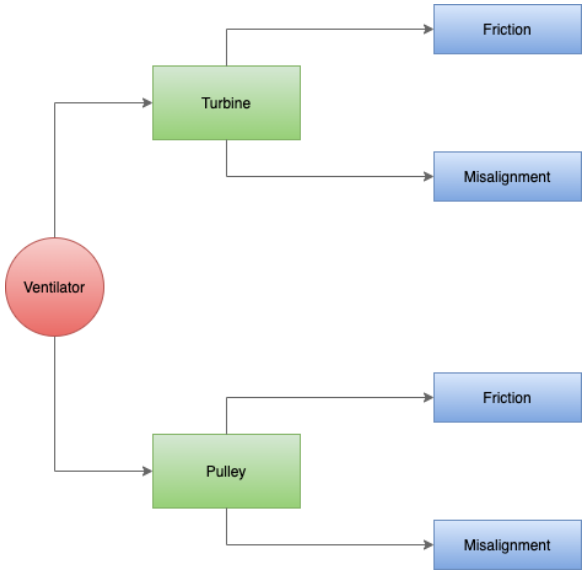


Figure 6-4: Two first layers of the FMEA tree. The failures modes are in blue.

6.5 Descriptive Part: Predicting Failure and Failure Causes

In this section, we present Module 1b, which is used to build the third layer of the FMEA tree (Failure Modes-Sensors) and extend it to account for probabilities of occurrence.

6.5.1 Model

As is often the case in practice, data to directly identify the exact cause of a failure is unavailable. The framework we develop for Module 1b detects the failure modes from the sensor data $\mathtt{X}_{\mathrm{D}}$ and assigns each failure mode to a particular sensor in $\mathtt{X}_{\mathrm{S}}$. We split the data-set such that we train a different model per component, because the failure modes interact differently depending on the type of equipment and the nature of the component, and the structure of the sensors is not the same either. Consequently, we again assume, without loss of generality, that there is one machine m and one component c in the dataset. We denote by $i \in [I]$ the set of all observations for the same component of the same equipment at the same time (aggregating the sensors), which we obtain by properly selecting a subset of the rows $r \in [R]$ in the data.

From Module 1a, we have a label $y_i \in \mathcal{F}_{m,c}$ in our data-set for the failure mode for each observation i. We denote by $\bar{\mathtt{X}}_{i,\Delta}$ the matrix of the sensor data (temperature, pressure, velocity, spectrum) of all sensors for a given observation i at a given timestamp; notice that $\bar{\mathtt{X}}_{i,\Delta}$ can be obtained by aggregating the data $\bar{\mathtt{X}}_{i,\mathrm{D}}$ from all sensors $s \in \mathcal{S}_{m,c}$ that correspond a specific component c of the same equipment m at the same time t. Then, $\bar{\mathtt{X}}_{i,s,\Delta}$ gives the vector of covariates for sensor $s \in \{\mathtt{X}_{i,\mathrm{S}},\ i \in [I]\}$ for the corresponding observation i at the same timestamp. We also note X the corresponding random variable, $S := S_{m,c} \in \mathbb{Z}^+$ the total number of sensors for this component, and $P \in \mathbb{Z}^+$ the total number of failure modes for this component.

For Module 1b, we develop the `assign-and-predict` method, a weighted logistic regression model with probabilistic cause assignment. We introduce the following parameters:

- $\boldsymbol{\alpha}_s$: The probability that the cause of the failure is sensor s, $\forall s \in [S]$.
- $\boldsymbol{\beta}_p^s$: The regression vector corresponding to sensor (cause) s, $\forall s \in [S]$, and failure p, $\forall p \in [P]$.
- $\boldsymbol{\gamma}_i$: The position or cause, where a sensor is, causing the failure for observation

$i \in [I]$.

Using the parameters defined above, we write the optimization problem shown in Equation (6.1), with the additional notation that failure $p = 0$ corresponds to no failure, and each $p > 0$ corresponds to a particular failure mode. For instance, in our ventilator example, $p = 1$ corresponds to $p =$ "friction" and $p = 2$ to $p =$ "misalignment".

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & -\sum_{i=1}^{I}\sum_{p=0}^{P} \mathbb{1}(y_i = p) \log\left(\sum_{s=1}^{S} \alpha_s \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum\limits_{l=0}^{P} \exp\left(-\beta_l^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}\right), \\ \text{s.t.} \quad & \sum_{s=1}^{S} \alpha_s = 1, \\ & \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{6.1}$$

The optimization in Problem (6.1) is the maximization of the likelihood of observing the data that we have, given the parametrization.

6.5.2 Interpretation and Results

Problem (6.1) directly gives the probability that the cause of the failure (if there exists a failure) is cause s, $\forall s \in [S]$, through the coefficient $\boldsymbol{\alpha}_s$. Furthermore, from the optimal solution to Problem (6.1), we can extract additional useful information, as explained next. For a given observation i, the probability that the failure mode is p, given that the cause of failure is sensor s, can be estimated as:

$$\mathbb{P}(y_i = p|\ \gamma_i = s, X = \bar{\mathtt{X}}_{i,\Delta}) = \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum\limits_{l=0}^{P} \exp\left(-\beta_l^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}.$$

The overall probability that failure mode is p can be estimated as:

$$\mathbb{P}(y_i = p|X = \bar{\mathtt{X}}_{i,\Delta}) = \sum_{s=1}^{S} \mathbb{P}(\gamma_i = s|X = \bar{\mathtt{X}}_{i,\Delta}) \mathbb{P}(y_i = p|\ \gamma_i = s, X = \bar{\mathtt{X}}_{i,\Delta})$$

$$= \sum_{s=1}^{S} \alpha_s \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum_{l=0}^{P} \exp\left(-\beta_l^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}.$$

The total probability of failure for observation i is given by:

$$\begin{aligned}\mathbb{P}(y_i = p | X = \bar{\mathtt{X}}_{i,\Delta}) &= \sum_{s=1}^{S} \mathbb{P}(\gamma_i = s | X = \bar{\mathtt{X}}_{i,\Delta}) \mathbb{P}(y_i = p |\ \gamma_i = s, X = \bar{\mathtt{X}}_{i,\Delta}) \\ &= \sum_{p=1}^{P} \sum_{s=1}^{S} \alpha_s \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum_{l=0}^{P} \exp\left(-\beta_l^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}.\end{aligned}$$

Finally, the probability that the cause is s, knowing that the identified failure mode is p, can be estimated (using Bayes rule) as:

$$\begin{aligned}\mathbb{P}(\gamma_i = s | y_i = p, X = \bar{\mathtt{X}}_{i,\Delta}) &= \frac{\alpha_s}{\mathbb{P}(y_i = p | X = \bar{\mathtt{X}}_{i,\Delta})} \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum_{l=0}^{P} \exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)} \\ &= \frac{\alpha_s}{\sum_{u=1}^{S} \alpha_u \frac{\exp\left(-\beta_p^u \bar{\mathtt{X}}_{i,u,\Delta}\right)}{\sum_{l=0}^{P} \exp\left(-\beta_l^u \bar{\mathtt{X}}_{i,u,\Delta}\right)}} \frac{\exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}{\sum_{l=0}^{P} \exp\left(-\beta_p^s \bar{\mathtt{X}}_{i,s,\Delta}\right)}.\end{aligned}$$

Module 1b, through the `assign-and-predict` method, allows to fully describe the state of the system given the sensor data $\mathtt{X}_\Delta$. We hence complete the data-driven FMEA tree and extend it to account for probabilities of occurrence at each of its levels. For the OCP "Ventilator", we obtain Figure 6-5, which is the final FMEA output from Module 1.

6.5.3 Benchmark of the Method

For comparison purposes, we use OCP data to benchmark the `assign-and-predict` method of Module 1b against a baseline, which consists of **(i)** running a logistic regression on the entire sensor data to predict the failure mode, and then **(ii)** randomly assigning the cause to one of the candidate sensors. We obtain the results shown in Table 6.2. The first column "Failure Mode" shows the out-of-sample accuracy in

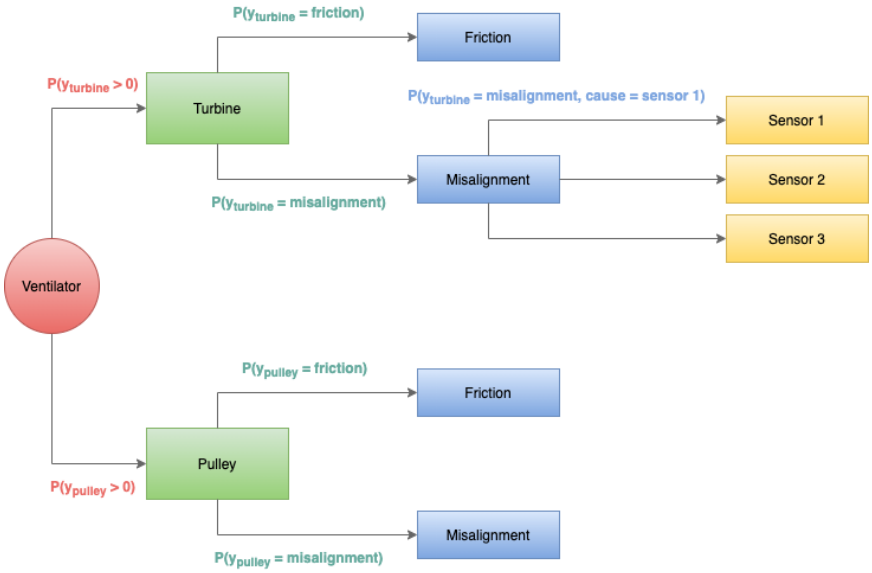


Figure 6-5: Complete FMEA tree. The sensors, or potential causes of failure are in yellow, and each edge represents a probability of occurrence for given sensor data.

predicting the failure mode for both methods; the second column “Cause of Failure" shows the out-of-sample accuracy in predicting the true cause (sensor) of this failure, if any; the last column “Cause of Failure, knowing the Failure Mode" is the same thing as the second column, but when the failure mode is actually known and not predicted.

Out-of-Sample Accuracy	Failure Mode	Cause of Failure	Cause of Failure, knowing the Failure Mode
`baseline`	19.9%	32.7%	63.3%
`assign-and-predict`	82.0%	58.1%	89.7%

Table 6.2: Comparison between the baseline and the assign-and-predict method.

Table 6.2 shows that the `assign-and-predict` method significantly outperforms the baseline in all three prediction tasks, going from under 20% in failure mode detection to 82%, and almost doubling the rate of identification of the correct cause of failure.

6.6 Predictive Part: Sparse and Slowly Varying Regression

As far as the predictive part is concerned, our goal is to use the available sensor data and train predictive models. Since the structure of the sensor data at different sensors of different components of different equipment types can be very different from each other, we need different models for each of them. Nevertheless, we can expect same sensors of the same component of pieces of equipment of the same type to produce data of the same structure; further, we can expect them to behave relatively similarly. Therefore, for each equipment type $m \in \mathcal{M}$, component $c \in \mathcal{C}_m$, and sensor $s \in \mathcal{S}_{m,c}$ we fit a single model across all such pieces of equipment $n \in \mathcal{N}_m$, which, at the same time, allows for small variations between the "submodels" that correspond to distinct pieces of equipment. For example, in the context of regression, we would estimate a different regressor $\boldsymbol{\beta}^n$ for each piece of equipment $n \in \mathcal{N}_m$ of one particular type $m \in \mathcal{M}$ (instead of estimating a single regressor $\boldsymbol{\beta}$ for the entire equipment type), while requiring that all such regressors are "similar."

The alternatives to the proposed approach would be, on the one extreme, to fit a single model for the entire equipment type and, on the other extreme, to fit completely independent models at the piece of equipment level. By training models at the equipment type level, we benefit in several ways:

- We train a single model using data from multiple sources hence increasing the dataset size. This exploits the structure of the problem and, specifically, the fact that pieces of equipment of the same type are expected to behave similarly.

- We can directly make predictions for new equipment of the same type without having to collect data and train a new model. In particular, we can utilize the submodel that corresponds to the most similar piece of equipment (of the same category) in the training data.

Additionally, we would want our model to be sparse and interpretable: we care about explaining our predictions to subject matter experts and hence need to use a

model with a small number of variables and with explainable interactions. To this end, we utilize the sparse and slowly varying regression framework, first introduced by [21], which satisfies the above requirements. We next give a description of the proposed framework, applied to the setting that we consider.

6.6.1 Model

We focus on equipment type $m \in \mathcal{M}$, component $c \in \mathcal{C}_m$, and sensor $s \in \mathcal{S}_{m,c}$. Our goal is to fit $N_m = |\mathcal{N}_m|$ regressions $(\boldsymbol{\beta}^n)_{n \in \mathcal{N}_m}$ over a graph G with vertices $\mathcal{N}_m$; each vertex in the graph corresponds to one of the N_m pieces of equipment. For $(n_1, n_2) \in \mathcal{N}_m \times \mathcal{N}_m$, the edge (n_1, n_2) is in the set of edges $\mathcal{E}$ if and only if pieces of equipment n_1 and n_2 are considered to be similar; for example, two pieces of equipment could be considered similar depending on their age. The similarity data, which is used to construct the similarity graph, has to be inputted to the system by subject matter experts.

Recall that the input to the predictive part is data of the form

$$\mathcal{X}_{\text{predictive}} = \{[r, \mathtt{X}_{r,\text{T}}, \mathtt{X}_{r,\text{N}}, \mathtt{X}_{r,\text{D}}] \mid r \in [R], \mathtt{X}_{r,\text{M}} = m, \mathtt{X}_{r,\text{C}} = c, \mathtt{X}_{r,\text{S}} = s\}.$$

That is, our data for this part consists of triplets of the form (Timestamp, Equipment ID, Sensor Data) for the specific equipment type $m \in \mathcal{M}$, component $c \in \mathcal{C}_m$, and sensor $s \in \mathcal{S}_{m,c}$ that we study. We are willing to make predictions for one feature $d \in \mathcal{D}_{m,c,s}$ in the future, given all features in $\mathcal{D}_{m,c,s}$ in the present. (Our approach directly generalizes to predicting the future values of all features in $\mathcal{D}_{m,c,s}$ through a multiple regression model.) Therefore, for $t \in [T_m - 1]$ and $n \in \mathcal{N}_m$, we introduce the following notations:

$$\begin{aligned} \boldsymbol{X}_t^n &= [\mathtt{X}_{r,\text{D}} \mid r \in \mathcal{X}_{\text{predictive}}, \mathtt{X}_{r,\text{T}} = t, \mathtt{X}_{r,\text{N}} = n] \in \mathbb{R}^{|\mathcal{D}_{m,c,s}|}, \\ y_t^n &= [\mathtt{X}_{r,\text{D}[d]} \mid r \in \mathcal{X}_{\text{predictive}}, \mathtt{X}_{r,\text{T}} = t+1, \mathtt{X}_{r,\text{N}} = n] \in \mathbb{R}, \end{aligned}$$

where recall that $\mathtt{X}_{r,\text{D}[d]}$ extracts feature $d \in \mathcal{D}_{m,c,s}$ from the sensor data in row r of the full dataset. In other words, $\boldsymbol{X}_t^n$ corresponds to the vector of features for

observation t of the n-th piece of equipment and y_t^n corresponds the response (target) value of observation t, i.e., the feature we are willing to predict at the next time step. Note that, for simplicity, we make the assumption that we have the same number of observations for all pieces of equipment; it is straightforward to drop this assumption and model the more general scenario of an unequal number of observations.

Then, the slowly varying regression problem with sparsity constraints can be formulated as below:

$$\min_{\boldsymbol{\beta}^1,\cdots,\boldsymbol{\beta}^{N_m}} \quad \sum_{n\in N_m}\sum_{t\in[T_m-1]}\left(y_t^n-(\boldsymbol{X}_t^n)^\top\boldsymbol{\beta}^n\right)^2+\lambda_\beta\sum_{n\in\mathcal{N}_m}\|\boldsymbol{\beta}^n\|_2^2+\lambda_\delta\sum_{(n_1,n_2)\in\mathcal{E}}\|\boldsymbol{\beta}^{n_2}-\boldsymbol{\beta}^{n_1}\|_2^2 \tag{6.2}$$

$$\text{s.t.} \quad |\text{Supp}(\boldsymbol{\beta}^n)|\le K_\text{L}, \quad \forall n\in\mathcal{N}_m, \tag{6.3}$$

$$\left|\bigcup_{n\in\mathcal{N}_m}\text{Supp}(\boldsymbol{\beta}^n)\right|\le K_\text{G}, \tag{6.4}$$

$$\sum_{(n_1,n_2)\in\mathcal{E}}|\text{Supp}(\boldsymbol{\beta}^{n_1})\triangle\text{Supp}(\boldsymbol{\beta}^{n_2})|\le K_\text{C}, \tag{6.5}$$

where $\text{Supp}(\boldsymbol{\beta})$ denotes the set that corresponds to the support of vector $\boldsymbol{\beta}$ and $S_1\triangle S_2$ denotes the symmetric difference of sets S_1, S_2. The objective function (6.2) penalizes both the least-squares loss of the N_m regressions and the l_2 coefficient distance between regressions that are similar with magnitude λ_δ. We also introduce an l_2 regularization term of magnitude λ_β for robustness purposes. There are three types of constraints on the regression coefficients $\boldsymbol{\beta}^n$:

- **Local Sparsity**: Each regression can have at most K_L relevant features (constraint (6.3)).
- **Global Sparsity**: There can be at most K_G relevant features across all M regressions (constraint (6.4)).
- **Sparsely Varying Support**: There can be a difference of at most K_C relevant features among similar regressions n_1, n_2 across all pairs of similar regressions (constraint (6.5)).

The parameters $K_{\mathrm{L}}, K_{\mathrm{G}}, K_{\mathrm{C}}$ have to be consistent, i.e., they need to satisfy $K_{\mathrm{L}} \leq K_{\mathrm{G}} \leq |\mathcal{D}_{m,c,s}|$ and $K_{\mathrm{C}} \leq 2K_{\mathrm{L}}N_m$.

As explained in [21], Problem (6.2)-(6.5) can be reformulated exactly as a binary convex optimization problem and, leveraging the convexity of the reformulated problem, it can be solved to optimality and at scale using a cutting plane-type algorithm. In particular, in combination with a heuristic stepwise method, which is used to find high-quality warm-start solutions, [21] develop a highly optimized implementation of such algorithm, which solves, in minutes, problems with $T_m \approx 10,000$ observations, $N_m \approx 50$ machines, and $|\mathcal{D}_{m,c,s}| \approx 600$ features.

6.6.2 Benchmark of the Method

We apply our proposed to the equipment type "Ventilator" (i.e., m = Ventilator). We use real-world data obtained from the same component $c \in \mathcal{C}_m$ and same sensor $s \in \mathcal{S}_{m,c}$ from $N_m = 6$ ventilators that OCP performs maintenance on. The collected features $\mathcal{D}_{m,c,s}$ include the temperature and 15 spectral features, for a total of $|\mathcal{D}_{m,c,s}| = 16$ features. Our goal is to predict the temperature at time $t + H$, where H denotes the prediction horizon (i.e., how much time ahead we are willing to predict) given all features at time t. We explore various values for H in our experiments (varying from four hours to one week). We use the same number $T_m \approx 1,000$ of observations for each distinct ventilator (piece of equipment); we use the first 70% of the observations as our training set and keep the remaining 30% as our test set.

As a baseline, we fit a single sparse regression model (with ridge regularization) applied to all ventilators, referred to as `baseline`. We use `ssv-regression` to refer to the proposed sparse and slowly varying regression model, solved using the exact cutting plane algorithm of [21]. We tune both methods' hyperparameters using cross validation. We report results for different sparsity levels. Note that, when the sparsity level for `baseline` is set to 1 and the temperature is the only feature selected, the resulting model coincides with a simple AR(1) model.

The results are presented in Table 6.3. The key takeaway is that `ssv-regression` is able to compute solutions that improve upon `baseline` in minutes. We note that

the performance of less sparse models, with more than 2 features, quickly deteriorates; this can be attributed to the high correlations among the features in the data (namely, the mean correlation between all pairs of features is ≈ 0.5). We also remark that, all models consistently select the temperature at time t as a relevant feature to predict that temperature at time $t + H$; when more features are allowed to be included in the model, `ssv-regression` uses all its degrees of freedom, alternating between the remaining features and estimating coefficients of slowly varying magnitudes.

	Parameters			**Out-of-sample R^2**				**Computational Time (in sec)**			
	K_L	K_G	K_C	$H = 4$ hours	$H = 12$ hours	$H = 3$ days	$H = 7$ days	$H = 4$ hours	$H = 12$ hours	$H = 3$ days	$H = 7$ days
`baseline`	1	1	-	0.690	0.541	0.336	0.186	7.5	8.4	8.3	7.8
	2	2	-	0.684	0.537	0.317	0.123	8.0	8.8	8.7	8.3
`ssv-regression`	1	3	2	0.692	0.541	**0.341**	0.183	33.2	69.7	227.8	213.5
	2	4	4	**0.695**	**0.557**	0.325	**0.189**	2172.9	2455.9	2467.3	4422.1

Table 6.3: Comparison between the baseline and the sparse and slowly varying (ssv) regression model.

6.7 Prescriptive Part: Holistic Prescriptive Analytics

In this section, we briefly reference one way of using the predictions from Modules 1 and 2 to schedule maintenance interventions and act before the failures occur, in an optimal way. We remark that, currently, OCP Maintenance Solutions uses a greedy approach, whereby the maintenance scheduling is performed based on the nominal values of the predictions made by combining Modules 1 and 2. In what follows, we propose an extension to the current system, Module 3, which would allow to do the predictions and the prescription jointly.

The proposed framework, dubbed `HPA` for holistic prescriptive analytics for continuous and constrained optimization problems, optimizes over a weighted average of the prediction loss, i.e., the error made by the predictive model that we train, and the prescriptive loss, i.e., the final objective we care about in the maintenance intervention (e.g., downtime or total intervention cost). The key intuition behind `HPA` is that it regroups observations, or machines at a particular time step, into groups of similar behavior, trains a sparse and slowly varying linear regression for each one of these

groups separately, and optimizes the maintenance interventions globally using the resulting predictions. All the above is optimized `jointly`, at the same time, instead of sequentially. We refer to [31] for the mathematical details of the formulation. The important thing to note is that `HPA` is particularly suitable for this application as **(i)** `HPA` can be easily solved in combination with the sparse and slowly varying regression method for prediction, **(ii)** the mixed-integer linear optimization problem for scheduling interventions is scalable, **(iii)** `HPA` eliminates a layer of errors in a framework that has multiple sequential modules, and consequently is subject to a more compounding effect from the prediction errors.

The aforementioned approach allows to bring together the predictive part (sparse and slowly varying regression) and the prescriptive part (maintenance operations scheduling), and drastically reduce the error by being more robust to prediction uncertainty in the final interventions. Ultimately, this end-to-end framework from description to prediction to prescription allows to have an exhaustive and detailed analysis of an industrial system and to take optimal, data-driven maintenance measures, resulting in significant and sustainable performance improvements. We outline such improvements in the next section.

6.8 Implementation and Impact

In this section, we discuss the status of implementation of the proposed preventive maintenance framework at OCP Maintenance Solutions and highlight its impact.

6.8.1 Implementation: the I-sense Platform

OCP Maintenance Solutions implements the proposed framework as part of its industrial internet of things (IIoT) platform named I-sense. The I-sense platform includes device management and integrated analysis tools that allow users to connect and manage devices, collect and analyze data to improve decision-making practices, all within a secure interconnected environment. The following modules of the I-sense platform are directly related to the proposed framework:

- **Measurements Module:** The measurements module provides all relevant information for a selected asset (i.e., piece of equipment), e.g., equipment type, equipment ID, components, etc. (see Figure 6-6). In addition, this module includes the "spectral analysis toolbox," which allows the user to select a component and monitor its sensors, measurement trends, and vibratory signals (see Figure 6-7); the system provides all the necessary functionalities and configurations for an effective vibration analysis process. The aforementioned tools can help vibration analysts (i.e., subject matter experts) diagnose the vibration signals and provide their recommendations for any failure and its cause (using the interface shown in Figure 6-6). We collect the provided data to label the signals that we have in the platform and retrain/improve our models.

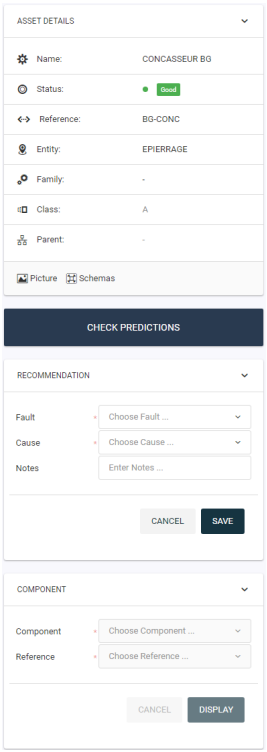


Figure 6-6: I-sense measurements module: asset management.

- **Alarms Management Module:** The alarms management module keeps track of incidents, automatically creates tickets for each alarm (or, in less urgent cases, alert) to be analysed, and monitors other key performance indicators, such as the tickets backlog trend, the alarms' age, and the time to intervention. Figure 6-8

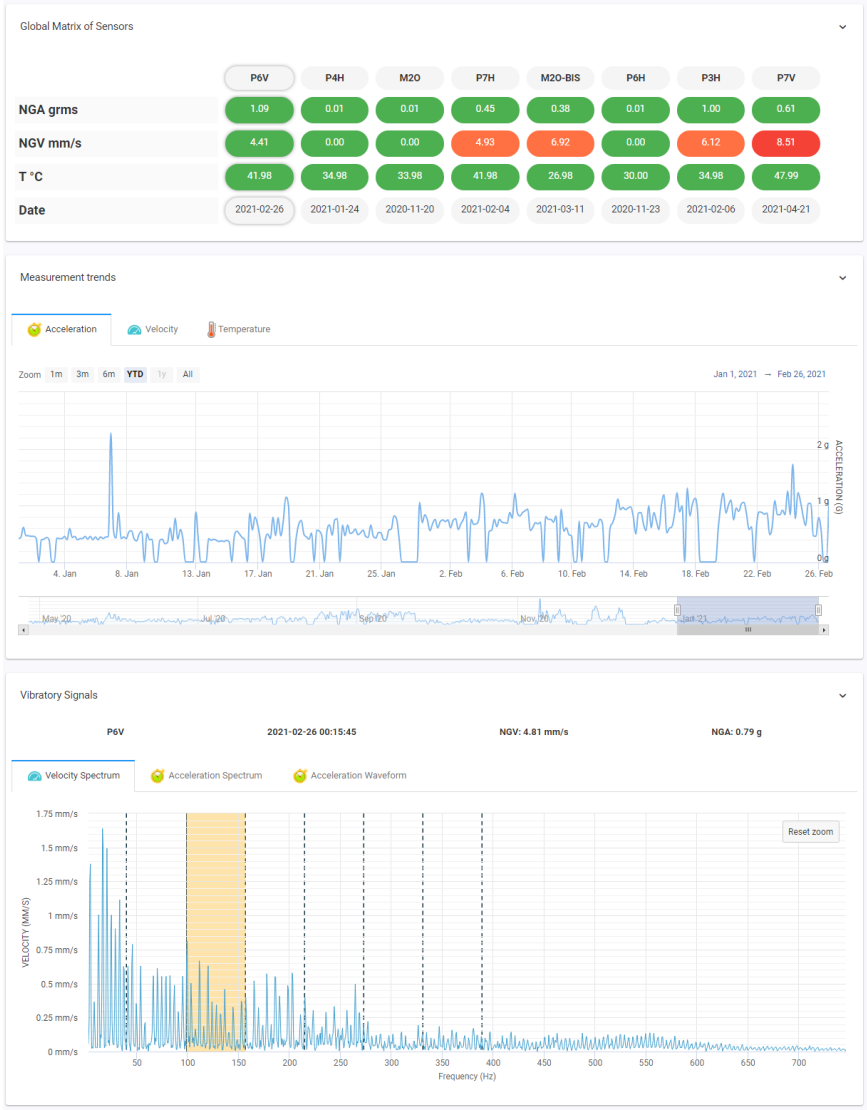
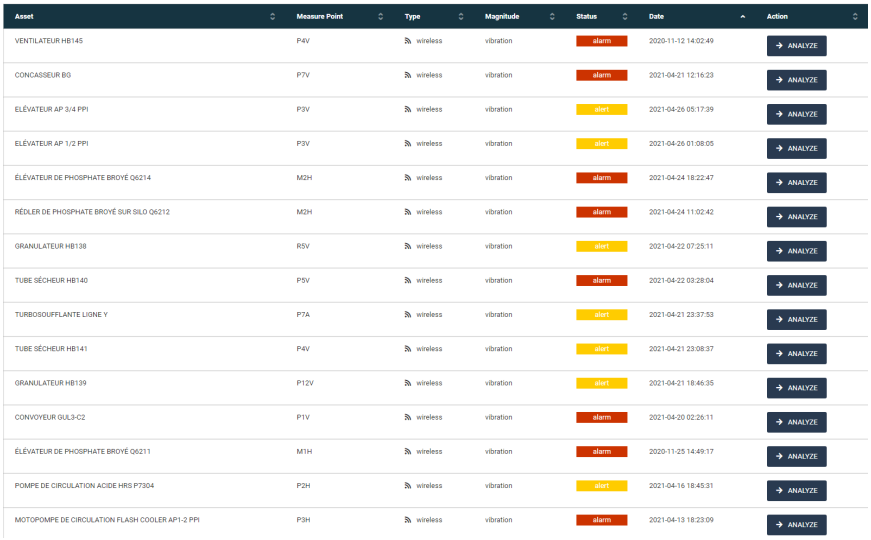


Figure 6-7: I-sense measurements module: monitoring components and the spectral analysis toolbox.

presents the alarms backlog, which is part of the alarms management module. The alarms backlog contains information about each alarm or alert, i.e., what is the failing equipment type, ID, and component.

- **Analytics Module:** The analytics module provides visualizations of predictions. The state prediction dashboard (Figure 6-9) provides state predictions for a given component of a piece of equipment.

  For a given timestamp in the state prediction dashboard, the failure and causes



Asset	Measure Point	Type	Magnitude	Status	Date	Action
VENTILATEUR HB145	P4V	wireless	vibration	alarm	2020-11-12 14:02:49	ANALYZE
CONCASSEUR BG	P7V	wireless	vibration	alarm	2021-04-21 12:16:23	ANALYZE
ELÉVATEUR AP 3/4 PPI	P3V	wireless	vibration	alert	2021-04-26 05:17:39	ANALYZE
ELÉVATEUR AP 1/2 PPI	P3V	wireless	vibration	alert	2021-04-26 01:08:05	ANALYZE
ÉLÉVATEUR DE PHOSPHATE BROYÉ Q6214	M2H	wireless	vibration	alarm	2021-04-24 18:22:47	ANALYZE
RÉDLER DE PHOSPHATE BROYÉ SUR SILO Q6212	M2H	wireless	vibration	alarm	2021-04-24 11:02:42	ANALYZE
GRANULATEUR HB138	R5V	wireless	vibration	alert	2021-04-22 07:25:11	ANALYZE
TUBE SÉCHEUR HB140	P5V	wireless	vibration	alarm	2021-04-22 03:28:04	ANALYZE
TURBOSOUFFLANTE LIGNE Y	P7A	wireless	vibration	alert	2021-04-21 23:37:53	ANALYZE
TUBE SÉCHEUR HB141	P4V	wireless	vibration	alert	2021-04-21 23:08:37	ANALYZE
GRANULATEUR HB139	P12V	wireless	vibration	alert	2021-04-21 18:46:35	ANALYZE
CONVOYEUR GUL3-C2	P1V	wireless	vibration	alarm	2021-04-20 02:26:11	ANALYZE
ÉLÉVATEUR DE PHOSPHATE BROYÉ Q6211	M1H	wireless	vibration	alarm	2020-11-25 14:49:17	ANALYZE
POMPE DE CIRCULATION ACIDE HRS P7304	P2H	wireless	vibration	alert	2021-04-16 18:45:31	ANALYZE
MOTOPOMPE DE CIRCULATION FLASH COOLER AP1-2 PPI	P3H	wireless	vibration	alarm	2021-04-13 18:23:09	ANALYZE

Figure 6-8: I-sense alarms management module: alarms backlog.

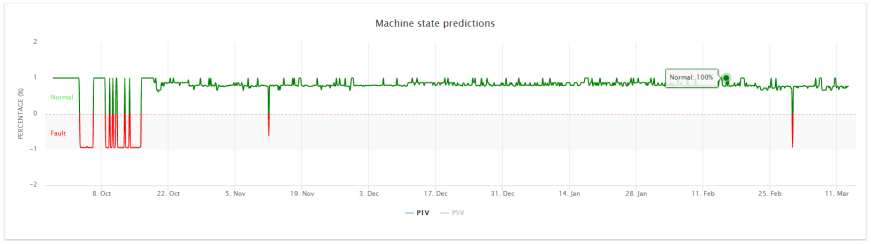


Figure 6-9: I-sense analytics module: state predictions.

prediction dashboard (Figure 6-10) depicts the probability of every failure class as estimated by the FMEA tree and the assign-and-predict model, along with the generated FMEA tree. For each pair (failure, cause), we assign an operation

that needs to be done to solve the issue.

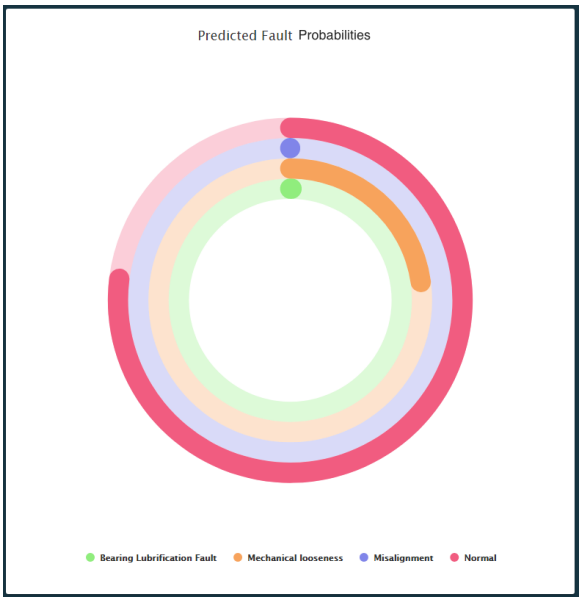


Figure 6-10: I-sense analytics module: failure probabilities.

In addition, the measurement predictions dashboard (Figure 6-11) presents actual and predicted values for each feature (temperature, vibration, etc.) and evaluates the sparse and slowly varying regression model's historic performance.

6.8.2 Deployment and Financial Impact

To illustrate measurable impacts of the proposed framework and the I-sense platform on OCP Maintenance Solutions customers, we present three examples:

- The first use case is about an alarm triggered by the exceeding vibration threshold on a sanitation fan on the south line at Maroc Chimie plant. The vibration levels at the bearing side on the fan reached 13.4 mm/s. The prescription was to control the misalignment and the fouling. An urgent intervention was made following these prescriptions, in addition to fan cleaning, and realignment and control of the bearing on the fan side. The vibration decreased considerably

Figure 6-11: I-sense analytics module: measurement predictions.

to 0.9 mm/s. The damages that were prevented include damage to the bearings and couplings, premature degradation of the motor, and a possible sudden interruption of the production line. On the business level, the fact that this intervention was carried out in time prevented an expected 14 hours of downtime, which corresponds to 826 tons of fertilizer produced and 2.5M MAD (around 290k USD) gained by the customer.

- The second use case concerns a reaction fan located on the north line of Maroc Chimie plant. The fan rose a vibration of 24.2 mm/s at the bearing fan side reached a global level, which was caused by a rotational imbalance, as well as a high acceleration of 4.2g on the same bearing, which was due to the friction caused by a lack of lubrication and by slack. Thanks to the timely predictions, we were able to clean the fouling of the fan, adjust the slack of bearing-roller on the fan side, and lubricate the bearings. As a result, the vibration decreased to 4.1 mm/s and we were able to avoid premature degradation of the bearings and a possible interruption of the production line. In this case, the intervention 8 hours of downtime, which corresponds to 472 tons of fertilizer produced and

1.4M MAD (around 160k USD) gained by the customer.

- The last example concerns an analysis performed on historical data from a grinder in Benguerir's site. We ran the model in a critical time slot where the machine had a failure. The asset was broken down for 18 days because of a rotating looseness. The model was able to predict the failure mode at its early stages, twelve days before the incident with a probability of 0.27 and one day before the incident with a probability of 0.71. We emphasize that this specific type of equipment has special mechanical structure and consists of custom parts that are extremely difficult to be replaced in case of failure. Therefore, having insights about which part is in danger is crucial and would minimize downtime.

6.9 Conclusion

In this chapter, we introduced a general framework for preventive maintenance from description to prediction to prescription. The proposed framework fully describes the state of an equipment at a given time in a data-driven way, predicts future failures and their causes, and recommends optimal maintenance intervention. We showed that the proposed framework significantly outperforms baselines for the same tasks. The proposed framework has been fully implemented and deployed by our industry collaborators, OCP Maintenance Solutions, resulting in a full-fledged platform for preventive maintenance, which we described in this chapter, as well as an important increase in performance. Specifically, the maintenance cost for the clients of OCP Maintenance Solutions decreased by more than 2.5 MAD over the course of a single year.

Chapter 7

Conclusions

This thesis was motivated by the fact that the current state-of-the-art in terms of predictive and prescriptive analytics in OR and OM can be improved upon with more powerful and interpretable machine learning methods for prediction, and a unified optimization framework for prescription, ultimately making the transition from data to prediction, and from prediction to prescription more data-based than model-based, and more streamlined. Additionally, it is crucial to bridge the gap between theory and application in OM, from revenue management to healthcare and healthcare operations to industrial operations.

This thesis considers such problems from broad perspectives, considering the Machine Learning approach under a modern Optimization lens while keeping in mind the end-users in the various fields of applications in Operations and beyond.

Chapter 2 looks purely at the predictive part of things by developing the XSTrees method, bringing the best of both worlds in terms of high computational performances and high interpretability for ease of use while keeping a rigorous theoretical framework. Chapter 4 combines the prediction with clustering and prescription to eliminate as many layers of error as possible and obtain a holistic framework to go directly from data to decisions. In Chapters 3, 5, and 6, we develop new methods and apply them to achieve high impact in revenue management through the collaboration with the leading online retailer for home furniture and decor, in healthcare through the work with MIT Quest for Intelligence and the CDC, and in industrial

operations through the collaboration with OCP. We show how machine learning and optimization methods can be expended and fine-tuned to tackle critical problems in operations and create value.

In conclusion, these areas of operations research and data-driven decision-making lie at the intersection of applied mathematics, optimization, learning, operations management, and subject-matter expertise. These thesis chapters shed light on how predictive and prescriptive analytics, particularly the discussed novel methods in the appropriate fields of application, can yield a significant edge over current practice, both in theoretical research and real-world applications.

Appendix A

Supplement for Chapter 2

A.1 HXTrees and more on the Asymptotic Convergence of the Framework

Similarly to what the literature does to prove convergence properties on tree ensembles (see [33] and [8]), we consider a version of the XSTrees algorithm where splits in different dimensions follow a certain structure and the training data is randomized. We establish that the HXSTrees algorithm (Honest XSTrees (Algorithm 8), is asymptotically consistent. HXSTrees have the property that for any test sample x, the tree distribution is independent of x. This is ensured by splitting the data into two independent halfs: one is used for tree generation, and the other is used for out-of-sample prediction.

Theorem 5 *Let RF model (T^*) of the HXSTrees algorithm be such that*

1. *Each candidate tree is a fully grown tree of depth ρ_n.*

2. *The split dimension (i.e. the dimension over which the split in a particular node occurs) is selected uniformly at random from any of the 1, . . . ,d dimensions.*

Also, assume that the distribution of $\mathbf{X}$ has support on $[0,1]^d$. Then, the HXSTree estimate $\bar{y}_n^{HXST}$ is consistent whenever $2^{\rho_n} \to \infty$ and $2^{\rho_n}/n \to 0$ as $n \to \infty$.

Algorithm 8 HXSTrees($\mathcal{D}_n$, K, T^{max}, S^{max}, T^*, x)

Randomly split $\mathcal{D}_n$ in two equal halves: $\mathcal{D}^1_{n/2}$ and $\mathcal{D}^2_{n/2}$.
Model Fitting
Randomly split $\mathcal{D}^1_{n/2}$ in two equal halves: $\mathcal{D}^1_{n/4}$ and $\mathcal{D}^2_{n/4}$.
for $i \in [K/2]$ **do**
  Let $h_i = \text{CART}(\mathcal{D}^1_{n/4}, \theta_i)$, and
  $(\vec{s}, \vec{l}_1, \ldots, \vec{l}_d)_i = \text{TreeExtension}(h_i)$.
end for
for $i \in [K/2+1, K]$ **do**
  Let $h_i = T^*(\mathcal{D}^2_{n/4}, \theta_i)$, and
  $(\vec{s}, \vec{l}_1, \ldots, \vec{l}_d)_i = \text{TreeExtension}(h_i)$.
end for
for $j \in [d]$ **do**
  Let $\hat{s}_j^{mean} = \frac{1}{K}\sum_{i=1}^K \vec{s}_j[i]$, $\hat{s}_j^{var} = \frac{1}{K}\sum_{i=1}^K \left(\vec{s}_j[i] - \hat{s}_j^{mean}\right)^2$.
end for

Prediction Sampling
for $q \in [T^{max}]$ **do**
  for $j \in [d]$ **do**
    $\tilde{\nu}_j^q = \mathcal{N}\left(\hat{s}_j^{mean}, \hat{s}_j^{var}\right)$.
    if $\tilde{\nu}_j^q \geq 2^{\rho_n}$ **then**
      for $m \in \lfloor \tilde{\nu}_j^q \rfloor$ **do**
        $\tilde{ls}_j^q(m) = \frac{m}{2^{\lfloor \tilde{\nu}_j^q \rfloor}}$
      end for
    end if
  end for
  Let $\hat{y}_q(x) = h^{XST}(\mathcal{D}^2_{n/2}, \tilde{ls}_1^q, \ldots, \tilde{ls}_d^q, x)$.
end for

Averaging Predictions
Predict $\bar{y}^{HXST}(x) = \frac{1}{T^{max}}\sum_{r=1}^{T^{max}} \hat{y}_q(x)$.

Proof of Theorem 5. We have that $\bar{y}^{HXST}(\mathbf{X})$ is a function of training data $\mathcal{D}_n$ and randomization parameter, θ (with a slight abuse of notation, representing the random number of splits and the location of these splits in each dimension). The randomization parameter θ is estimated from two tree construction methods. First, on the $\mathcal{D}^1_{n/4}$ samples, we use CART trees, and then, on the second random split of the data $\mathcal{D}^1_{n/4}$ we use fully grown trees with the structure as specified in the statement.

First note that by Jensen's inequality that

$$\mathbb{E}_{\mathbf{X}}\left[y(\mathbf{X}) - \bar{y}_n^{HXST}(\mathbf{X})\right]^2 = \mathbb{E}_{\mathbf{X}}\left[\mathbb{E}_{\theta}\left[\left(y(\mathbf{X}) - \bar{y}_n^{HXST}(\mathbf{X},\theta)\right)^2\right]\right]$$
$$\leq \mathbb{E}_{\mathbf{X}}\left[\left(y(\mathbf{X}) - \bar{y}_n^{HXST}(\mathbf{X},\theta)\right)^2\right].$$

Next, we will show that all extended trees constructed using θ, have the following two properties:

1. Let $\text{diam}(\mathcal{L}_n(x,\ \theta))$ define the maximum length of the rectangular region $\text{diam}(\mathcal{L}_n(x,\ \theta))$ in any dimension. Then, $\text{diam}(\mathcal{L}_n(x,\ \theta)) \to 0$.

2. $\lim_{n\to\infty} N_n(x,\ \theta) \to \infty$.

Recall that $\theta = (l_1, \ldots,\ l_d)$ is defined by the distribution of the location of splits, given the number of splits in each dimension. Now for any dimension i, we have that $\hat{s}_i = \lim_{K\to\infty} \frac{1}{K} \sum_{j=1}^{K} \vec{s}_j[i] = \lim_{K\to\infty} \frac{1}{K} \left(\sum_{j=1}^{K/2} \vec{s}_j[i] + \sum_{j=\frac{K}{2}+1}^{K} \vec{s}_j[i]\right) \geq \lim_{K\to\infty} \sum_{j=\frac{K}{2}+1}^{K} \vec{s}_j[i]$. By construction, we have that all the trees numbered (K/2 +1) to K are fully grown with depth ρ_n. Furthermore, the split dimension is also random and selected uniformly over the d dimensions. For all trees constructed with the above two restrictions we have that, $S_i \sim Binomial(2^{\rho_n},\ 1/d)$. Hence, for any n and $\forall i = 1, \ldots, d$

$$\lim_{K\to\infty} \sum_{j=\frac{K}{2}+1}^{K} \vec{s}_j[i] = \frac{2^{\rho_n}}{d}.$$

To show (1), first note that by definition, $\forall x$

$$\text{diam}(\mathcal{L}_n(x,\ \theta)) = \max_{j=1,\ldots,d} V_{nj}(x,\ \theta),$$

where $V_{nj}(\mathbf{X},\theta)$ is the size of the rectangle in the j^{th} dimension which contains $\mathbf{X}$. By construction, we have that for any given x, $V_{nj}(x,\ \theta) \sim 2^{-S_j}$. Hence, for any $j = 1, \ldots, d$

$$\mathbb{E}\left[V_{nj}(\mathbf{X},\ \theta)\right] \leq \mathbb{E}_{\mathbf{X}}\left[\mathbb{E}\left[2^{-S_j}|X\right]\right] \leq 2^{-\frac{\rho_n}{d}}.$$

Hence, $\mathbb{E}_{\mathbf{X}}\left[V_{nj}(\mathbf{X},\theta)\right]$ goes to 0 in probability, as ρ_n goes to infinity with n. Next, consider statement (2) above. By definition, $N_n(x) = \sum_{i=1}^{\frac{n}{2}} \mathbb{1}\{x_i \in \mathcal{L}_n(x,\theta)\}$. Finally, notice that the tree sampling step is constrained to have at least 2^{ρ_n} cuts in each dimension. Hence, after extending the splits, there are $(d-1)2^{\rho_n}$ rectangular

regions. Therefore,

$$\mathbb{P}\left(N_n(x) < M\right) = \mathbb{E}\left[\mathbb{P}\left(N_n(x) < M\right)|x\right] = \sum_{i=1}^{2^{d+\rho_n-1}} \frac{N_i}{\frac{n}{4}+1} \mathbb{1}\{N_n(x) < M\}$$
$$\leq \frac{2^{d+\rho_n+1} M}{n},$$

which converges to zero, by assumption. After having established that properties (1), (2) described above hold, the consistency result holds similarly to ([33]). ■

A.2 Hyper-parameters Tuning

For our computational tests in §2.5, we consider the following range of hyper-parameters:

1. **Linear Models (Elastic Net)**
   - L_1 Penalization Ratio: [0, 0.1, 0.2, 0.3, ..., 1]
   - Weight for L_1 and L_2 Penalization (α): [$1e^{-5}$, $1e^{-4}$, $1e^{-3}$, $1e^{-2}$, $1e^{-1}$, 1, 10, 100]
2. **Support Vector Machines**
   - Soft Margin Parameter (C): [.1, 1, 10, 100, 1000]
3. **K-Nearest Neighbors**
   - Number of Nearest Neighbors (K): [1, 5, 10, 20, 40]
4. **Classification and Regression Trees**
   - Maximum Depth: [3, 4, 5, 6, 10, None]
   - Minimum Samples per Leaf (%): [0.04, 0.06, 0.08]
   - Maximum Features to consider per split (%): [0.2, 0.4, 0.6, 0.8]
5. **AdaBoost**

- Number of Estimators: [50, 100, 300]
- Learning Rate: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 1]

6. **Feed-Forward Neural Networks**

   - Number of Hidden Layers: [1, 2, 3, 4, 5]
   - Number of Neurons per Layer: [2, 3, 5, 10, 20]
   - Activation Function: tanh and reLU
   - Solver: SGD and Adam
   - Alpha: [0.0001, 0.05]
   - Learning Rate: Constant and Adaptive

7. **LightGBM**

   - Maximum Depth: [5, 10, 30, 40, None]
   - Number of Estimators: [50, 100, 300]
   - Learning Rate: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 1]

8. **Random Forests**

   - Bootstrap: w/ and w/o
   - Maximum Depth: [5, 10, 30, 40, None]
   - Maximum Features to consider per split: None and square root of total number of features
   - Minimum Samples per Leaf: [1, 2, 4]
   - Minimum Samples per Split: [2, 5, 10]
   - Number of Estimators: [50, 100, 300]

9. **XGBoost**

   - Minimum Child Weight: [1, 2, 3, . . . , 20]

- Gamma: [1, 2, 3, . . . , 6]
- SubSample: [0.8, 1.0]
- Alpha: [0.5, 1, 2, 5]
- ColSample ByTree: [0.6, 0.8, 1.0]
- Maximum Depth: [3, 4, 5, 6, 10]
- Learning Rate: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 1]

10. **XSTrees**

- Bootstrap: w/
- Maximum Depth: [4, 5, 6]
- Minimum Samples per Leaf: [1, 2, 4, 14]
- Minimum Samples per Split: [2, 5, 10]
- Number of Estimators: [50, 100, 300]
- Number of Sampled Trees: [100, 150, 300]
- Number of K-Nearest Leaves (Heuristic): [4, 10, 20, 30]
- Update Procedure: [“ordering", “bayesian"]

Hyper parameter tuning is done using K-fold cross validation with $K = 10$, exclusively on the training data. Experiments are performed in a Python 3.6.1 environment. We use our own implementation for the XSTrees, which is open source, the **xgboost 0.90** [49] library for XGBoost, and **sklearn 0.21.1** [129] for the rest.

A.3 Standard Deviation of Computational Results

In this section, we detail the standard deviation of the out-of-sample accuracy over the 20 UCI classification datasets presented in §2.5 of the chapter. See Table A.1.

We notice that on average, XSTrees is much more stable (in terms of variability in out-of-sample accuracy for the same problems with different random seeds) than

Dataset	Baseline	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST
Abalone	0.0060	0.0068	0.0119	0.0084	0.0241	0.0246	0.0718	0.0187	0.0185	0.0103	0.0144
Adult	0.0091	0.0079	0.0000	0.0052	0.0415	0.0095	0.0375	0.0186	0.0138	0.0106	0.0130
Anneal	0.0062	0.0077	0.0060	0.0121	0.0298	0.0238	0.0320	0.0119	0.0195	0.0183	0.0112
Car	0.0093	0.0090	0.0125	0.0083	0.0479	0.0175	0.0351	0.0197	0.0195	0.0120	0.0136
Contraception	0.0139	0.0083	0.0166	0.0131	0.0395	0.0140	0.0399	0.0144	0.0299	0.0120	0.0073
Credit Approval	0.0081	0.0054	0.0147	0.0162	0.0570	0.0233	0.0534	0.0089	0.0137	0.0138	0.0130
Banknote Authentification	0.0000	0.0000	0.0000	0.0056	0.0242	0.0089	0.0000	0.0102	0.0186	0.0142	0.0000
Haberman	0.0107	0.0089	0.0000	0.0110	0.0510	0.0158	0.0464	0.0210	0.0066	0.0000	0.0121
Heart Disease	0.0110	0.0073	0.0112	0.0063	0.0221	0.0177	0.0377	0.0194	0.0174	0.0136	0.0114
Congressional Vote	0.0047	0.0093	0.0092	0.0177	0.0556	0.0137	0.0330	0.0172	0.0121	0.0182	0.0137
Iris	0.0064	0.0072	0.0197	0.0085	0.0360	0.0107	0.0668	0.0100	0.0181	0.0183	0.0104
Chess	0.0105	0.0083	0.0098	0.0072	0.0517	0.0189	0.0513	0.0197	0.0190	0.0000	0.0175
King Rook vs. King	0.0075	0.0067	0.0138	0.0076	0.0199	0.0190	0.0731	0.0148	0.0120	0.0131	0.0077
Magic04	0.0146	0.0126	0.0171	0.0132	0.0414	0.0182	0.0443	0.0192	0.0163	0.0248	0.0084
MONK-01	0.0123	0.0064	0.0132	0.0123	0.0220	0.0176	0.0790	0.0155	0.0148	0.0177	0.0068
MONK-02	0.0084	0.0068	0.0133	0.0190	0.0176	0.0214	0.0586	0.0138	0.0207	0.0000	0.0081
MONK-03	0.0040	0.0122	0.0128	0.0119	0.0525	0.0218	0.0316	0.0163	0.0153	0.0132	0.0124
Part Failures	0.0044	0.0069	0.0147	0.0079	0.0362	0.0253	0.0375	0.0153	0.0187	0.0146	0.0000
Sonar	0.0143	0.0074	0.0000	0.0125	0.0469	0.0203	0.0407	0.0190	0.0136	0.0058	0.0081
Transfusion	0.0132	0.0081	0.0136	0.0130	0.0400	0.0086	0.0530	0.0079	0.0119	0.0142	0.0197
Average	*0.0087*	*0.0077*	*0.0105*	*0.0108*	*0.0379*	*0.0175*	*0.0478*	*0.0156*	*0.0165*	*0.0133*	*0.0104*

Table A.1: Standard Deviation of the Out-of-Sample Accuracy for the UCI Datasets

their single tree counterpart (0.0104 average standard deviation versus 0.0379 for CART), as well as RF (0.0165). However, it has slightly more variability than XGB (0.0133). Unsurprisingly, FNN, ADA and LGBM suffer from a higher variability. Table A.1 indicates that the results and conclusions discussed in §2.5 of the chapter are significant.

A.4 Detailed Results for the UCI Regression Benchmark

.

We report the results in terms of out-of-sample coefficient of determination (R^2), and in terms of ranks for the UCI Regression Benchmark in Table A.2 and Table A.3. Note that we take as a baseline model the average value of the target variable in the training data, which means the R^2 of the baseline model is always equal to 0, so we ignore it in the rank comparison.

Dataset	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST	XST Rank
Automobile	0.076	0.027	0.700	0.908	0.916	0.748	0.848	0.920	0.943	**0.961**	**1**
Communities	0.611	**0.658**	0.459	0.525	0.488	0.530	0.620	0.626	0.583	0.655	2
Servo	0.482	0.839	0.445	0.809	0.731	0.596	0.571	0.894	**0.931**	0.881	3
Students	0.211	0.186	0.113	0.112	0.203	0.263	**0.303**	0.265	0.227	0.260	4
Wine	0.359	0.329	0.126	0.236	0.374	0.339	0.409	0.444	0.371	**0.460**	**1**
Average	*0.348*	*0.408*	*0.369*	*0.518*	*0.542*	*0.495*	*0.550*	*0.630*	*0.611*	***0.643***	***1***
Median	*0.359*	*0.329*	*0.445*	*0.525*	*0.488*	*0.530*	*0.571*	*0.626*	*0.583*	***0.655***	***1***

Table A.2: Mean Out-of-Sample R^2 for the UCI Regression Datasets. In bold, the top-performing algorithm for each row. Ranks of the XSTrees are reported in the last column.

Dataset	LIN	SVM	KNN	CART	ADA	FNN	LGBM	RF	XGB	XST
Automobile	9	10	8	5	4	7	6	3	2	**1**
Communities	5	**1**	10	8	9	7	4	3	6	2
Servo	9	4	10	5	6	7	8	2	**1**	3
Students	6	8	9	10	7	3	**1**	2	5	4
Wine	6	8	10	9	4	7	3	2	5	**1**
Average	*7*	*6.2*	*9.4*	*7.4*	*6*	*6.2*	*4.4*	*2.4*	*3.8*	***2.2***
Median	*6*	*8*	*10*	*8*	*6*	*7*	*4*	***2***	*5*	***2***

Table A.3: R^2 Rank for the UCI Regression Datasets. In bold, the top-performing algorithm for each row.

Appendix B

Supplement for Chapter 3

B.1 Proof of Proposition 2

Proof: We prove the two properties of Proposition 2 separately:

(1) The CWC algorithm has two main steps in each iteration. In step 2(a) we estimate multinomial logistic regression models of individual clusters based on the cluster assignments from the previous iteration. By assumption, this step takes $O(p(n, d, L))$ time where recall that n is the total number of observations, d is the dimension of each observation and L is the total number of clusters. Then, in Step 2(b) we iterate over each observation to check if the cluster assignment can be improved. This is accomplished by first estimating the log likelihood of each observation across all clusters and then assigning the point to the cluster with the maximum log-likelihood. Hence, this step takes $O(ndL)$ time. Now since there are T total rounds, the time complexity of the algorithm is $O(Tp(n, d, L) + TndL)$.

(2) During the algorithm, step 2(a) fixes $\widehat{z}_{ik}^{(t-1)}$ and minimizes $\mathcal{L}$ to estimate multinomial logistic models. Let the fitted logistic models be $\hat{p}^{(t)}$. Then, step 2b fixes $\hat{p}^{(t)}$ and minimizes $\mathcal{L}$ by re-clustering products to find $\widehat{z}_{ik}^{(t)}$, which means

$$\mathcal{L}(\widehat{z}^{(t)}, \hat{p}^{(t)}) \leq \mathcal{L}(\widehat{z}^{(t-1)}, \hat{p}^{(t)}) \leq \mathcal{L}(\widehat{z}^{(t-1)}, \hat{p}^{(t-1)}),$$

Thus, the objective decreases in every iteration of step 2. Since $\mathcal{L}$ is lower bounded (since we do not allow for pure clusters), the algorithm must converge to a solution. Furthermore, if convergence happens at time $t-1$, this can be checked from time t onwards if the solution remains unchanged, i.e., $\widehat{p}^{(\tau)} = \widehat{p}^{(t-1)}$ and $\widehat{z}^{(\tau)} = \widehat{z}^{(t-1)}$ for $\tau \geq t$. For this to happen, we only need that $\widehat{z}^{(t)} = \widehat{z}^{(t-1)}$, because the deterministic algorithms of steps 2a and 2b will find the same solutions at every iteration onwards.

■

B.2 Proof of Theorem 1

Proof: We will prove the result of Theorem 3 by contradiction. Assume that $\exists \mathcal{S}_j \in \mathbf{S}$: $\mathcal{S}_j \neq \mathcal{A}(m^*)$ & $r(\mathcal{S}_j) > r(\mathcal{A}(m^*))$. Let $\tilde{m} = |\mathcal{S}_j|$. We have two cases to analyze. Either $m^* = \tilde{m}$ or otherwise $m^* \neq \tilde{m}$. First consider the case when $m^* = \tilde{m}$. Since $\mathcal{S}_j$ is optimal, we have that that

$$
\begin{aligned}
& r(Z^*) > r(\mathcal{A}(m^*)) \\
& \implies \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = j|\mathcal{S}_j) NPV_i - c_j B > \sum_{i \in \mathcal{A}(m^*)} \hat{f}_{SSP}(Y = i|\mathcal{A}(m^*)) NPV_i - c_j B \\
& \implies \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = i|\mathcal{S}_j) NPV_i - c\tilde{m} B > \sum_{i \in \mathcal{A}(m^*)} \hat{f}_{SSP}(Y = i|\mathcal{A}(m^*)) NPV_i - cm^* B \\
& \implies \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = i|\mathcal{S}_j) NPV_i - cm^* B > \sum_{i \in \mathcal{A}(m^*)} \hat{f}_{SSP}(Y = i|\mathcal{A}(m^*)) NPV_i - cm^* B \\
& \implies \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = i|\mathcal{S}_j) NPV_i > \sum_{i \in \mathcal{A}(m^*)} \hat{f}_{SSP}(Y = i|\mathcal{A}(m^*)) NPV_i \,,
\end{aligned}
\tag{B.1}
$$

where we have suppressed the dependence of $\hat{f}_{SSP}$ on X and $\hat{\beta}$ for ease of exposition.

Also note by construction that $\mathcal{A}(m^*)$ is the optimal assortment of size at most

m^*. Since, $m^* = \tilde{m}$, we have that

$$\sum_{i \in \mathcal{A}(m^*)} \hat{f}_{SSP}(Y = i|\mathcal{A}(m^*))NPV_i \geq \sum_{i \in \mathcal{S}_j} \hat{f}_{SSP}(Y = i|\mathcal{S}_j)NPV_i\,.$$

Hence, we have reached a contradiction.

Next consider the case when $\tilde{m} > m^*$. To prove a contradiction, first consider all the assortments of at most size $\tilde{m}$. Then, by construction, we have that $\mathcal{A}(\tilde{m}) = \mathcal{S}_j$, which follows using the same argument as above. Furthermore, by the optimality of $\mathcal{A}(m^*)$, we also have that

$$\begin{aligned} & r^*(m^*) - c|\mathcal{A}(m^*)|B \geq r^*(k) - c|\mathcal{A}(k)|B,\ \forall k = 1, .., K. \\ \implies & r^*(m^*) - c|\mathcal{A}(m^*)|B \geq r^*(\tilde{m}) - c|\mathcal{A}(\tilde{m})|B, \\ \implies & r^*(m^*) - c|\mathcal{A}(m^*)|B \geq r^*(\mathcal{S}_j) - c|\mathcal{S}_j|B. \end{aligned} \tag{B.2}$$

Hence, we have reached a contradiction again since $\mathcal{S}_j$ was assumed to be the optimal solution ot the ASDO problem. Finally, the case when $\tilde{m} < m^*$ follows identically and we skip the details for the sake of brevity. ■

B.3 Visualization of the Results

In this section, we present the visualization (Figure B-1) of the cluster assignment tree in §3.4, as well as the single tree explainer ((Figure B-2)) for the incremental revenue of the assembly service from the same section. Both of these figures were placed in this appendix because of space constraints.

B.4 Definition of Performance Metrics

In this section we detail the definition of the Precision, Recall, F1-Score and Support metrics used in §3.4:

- **Precision**: or positive predictive value (PPV) is defined as the ratio between

true positives TP and predicted positive (true positives TP + false positives FP). We write PPV = $\frac{\text{TP}}{\text{TP+FP}}$.

- **Recall**: or true positive rate (TPR) is defined as the ratio between true positives TP and actual positives (true positives TP + false negatives FN). We write TPR = $\frac{\text{TP}}{\text{FN+FP}}$.
- **F1-Score**: the harmonic mean of precision and recall. We write F1-Score = $2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV+TPR}}$.
- **Support**: the total number of observations in the corresponding class.

B.5 Details on the Case Study

In this section, we discuss in detail how we calculate the estimated uplift based on implementing the proposed ancillary service optimization framework.

1. We consider 21144 unique customer sessions, where during the session the customer arrives at the product page.
2. For each customer in these sessions, we utilize our NPV models to estimate the NPV of the customer purchasing each of the 3 services. We obtain 3 values: Assembly NPV, Warranty NPV, and PLCC NPV.
3. For each customer at the product page, we utilize our CWC model to predict a vector of probabilities that the customer will purchase each of the services (including not purchasing any service) under the 7 different actions by our industry collaborator. We obtain a 7 x 4 matrix of probabilities, where entry (i,j) captures the probability that customer will sign up for service j under action i.
4. Using the NPV values from (2) above, we create a 4 dimensional "revenue" vector which captures the immediate and incremental revenue that the customer will generate for our collaborator if he buys a service

- Value from no service purchased: 0
- Value from assembly purchased: Assembly NPV + Assembly Margin * Assembly Price
- Value from warranty purchased: Warranty NPV + Warranty Margin * Warranty Price
- Value from PLCC purchased: PLCC NPV - PLCC Cost

Assembly Margin = 0.17, Warranty Margin = 0.15, and PLCC Cost = \$40. This \$40 cost for PLCC is associated with the store credit that our collaborator gives to new customers on the platform.

5. We take a dot product between the probability matrix from (3) and the value vector from (4), which yields a 7-dimensional vector which basically captures the "expected revenue" under each of the 7 actions.

6. To capture our assumption that more services impose a cognitive burden on customers, we impose higher "costs" for showing more services. Due to business considerations, we express this cost as a form of reduced conversion rates multiplied by the product price.

   For example, we assume that showing 3 services leads to a 0.3% reduction in conversion, while showing 2 services leads to a 0.2% reduction in conversion, while showing 1 service is the baseline (0% on conversion). These numbers were informed by other business units of our industry collaborator. We also evaluate the sensitivity of the estimated improvement, as we change the cost parameters. The results of the sensitivity analysis are provided in Table B.1.

7. The cost of showing 3 services is then 0.3% * Product price. We then use this to create a 7-dimensional cost vector which captures the "cost" of the 7 possible actions.

8. The "expected value" of each of the 7 actions is then given by:

   Expected Value = Expected Revenue (from (5)) - Cost (from (6))

We then prescribe the action with the highest expected value amongst the set of feasible actions. If the product in question was not assembly-eligible, then any combination involving the Assembly service would be deemed an infeasible action, and cannot be prescribed.

9. Our baseline for comparison is that our collaborator shows all services that the customer / product is eligible for. We find that by converting to the prescribed display actions we gain about 2.5% - 3.5% in incremental revenue which is about $600-850 for 21000 sessions (i.e. about $0.03 cents per customer session).

In particular, (8)-(9) show that the optimization framework only assigns feasible services to a sampled session. Hence, the estimated improvements are made by "'personalizing" eligible ancillary services for different customers.

Assumption	**Number of Services**	**Impact on Conversion(c)**	**Improvement**
A	1	0.1%	2.7%
	2	0.3%	
	3	0.2%	
B	1	0%	2.9%
	2	0.2%	
	3	0.3%	
C	1	0%	3.2%
	2	0.3%	
	3	0.4%	

Table B.1: Impact of Optimized Service Display on Revenue under different conversion impact assumptions

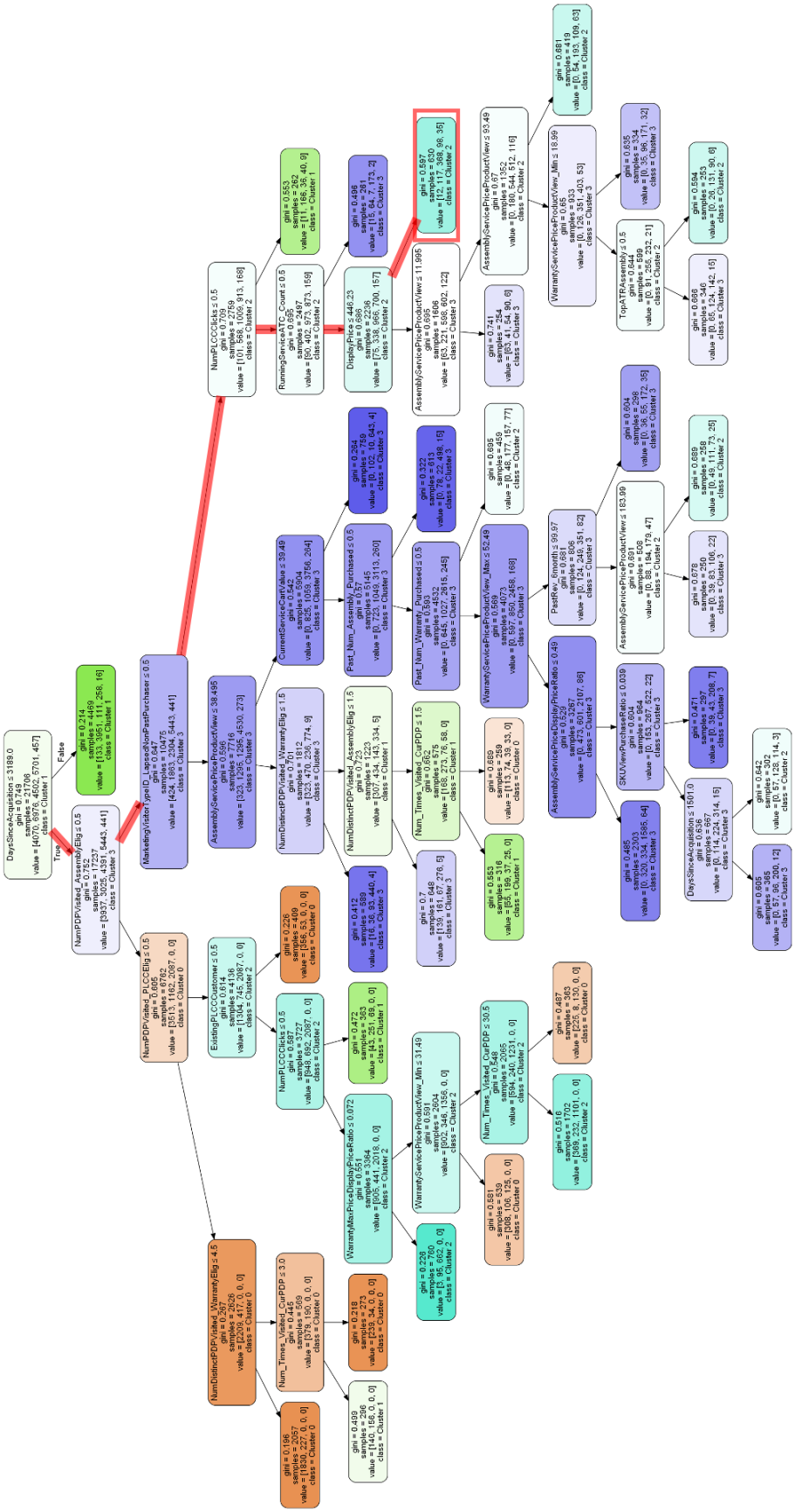


Figure B-1: Visualization of the Cluster Assignment Decision Tree

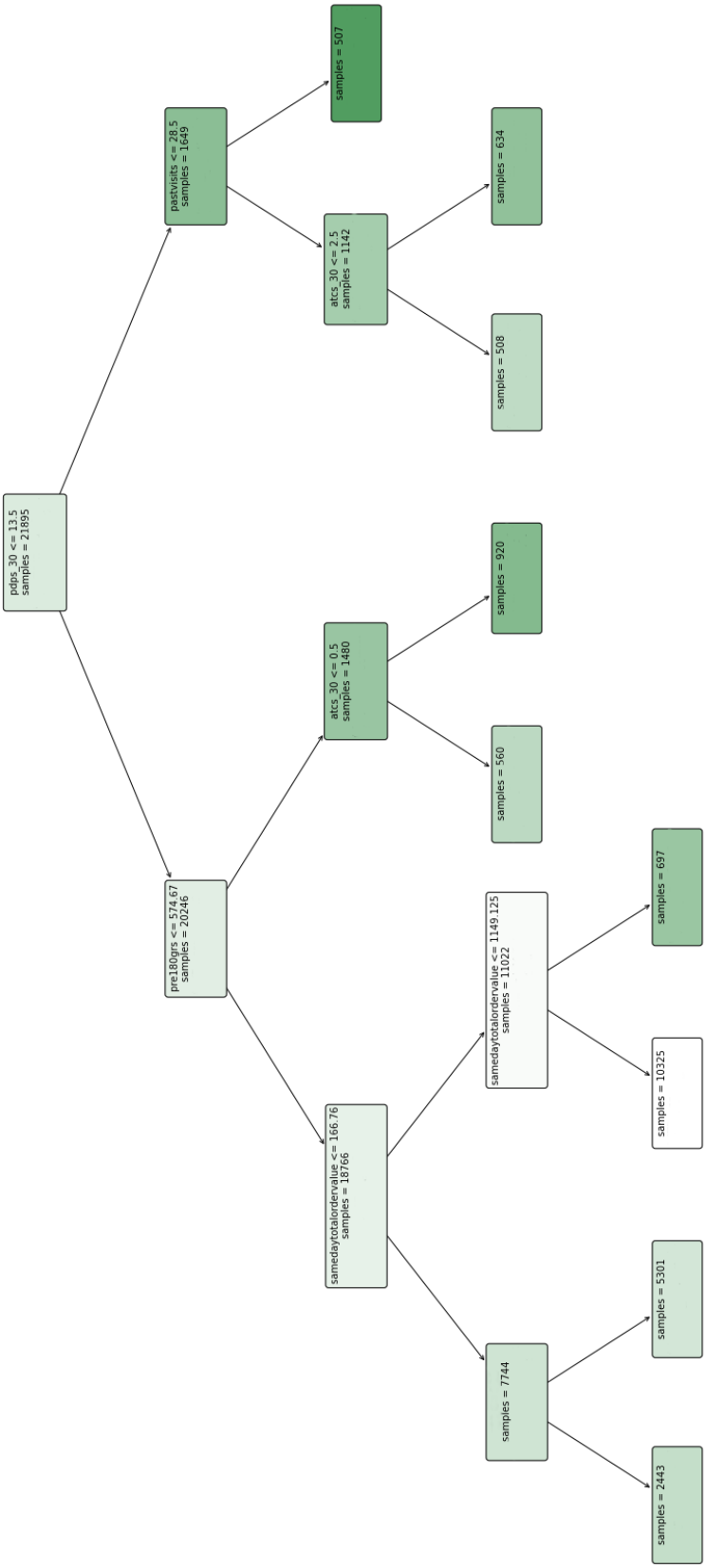


Figure B-2: Single Tree Explainer for the Incremental Revenue of the Assembly Service

Appendix C

Supplement for Chapter 4

C.1 Generalization of the Formulation to multiple new observations

The use of one single x_0 is for illustrative purposes, the framework remains identical when we have $n_{test} \in \mathbb{N}$ new observations we want to prescribe an optimal decision for. Equation (5) becomes:

$$
\begin{aligned}
&\min_{\boldsymbol{u},\boldsymbol{f},\bar{\boldsymbol{z}}} \ \lambda \sum_{i=n_{test}}^{n} \sum_{j=1}^{k} u_{i,j}(y_i - f_j(\boldsymbol{x}_i, \boldsymbol{z}_i))^2 + (1-\lambda) \sum_{i=0}^{n} \sum_{j=1}^{k} u_{i,j} c(f_j(\boldsymbol{x}_i, \bar{\boldsymbol{z}}_i), \bar{\boldsymbol{z}}_i) \\
&\text{s.t} \quad h_r(\bar{\boldsymbol{z}}_0, \ldots, \bar{\boldsymbol{z}}_n) \leq b_r, \ \forall r \in [s] \text{ (constraints on decisions)}, \\
&\qquad \sum_{i=n_{test}}^{n} u_{i,j} \geq N_{min}, \ \forall j \in [k] \text{ (minimum number of training observations per cluster)}, \\
&\qquad \sum_{j=1}^{k} u_{i,j} = 1, \ \forall i \in \{0, \ldots, n\} \text{ (one observation is assigned to exactly one cluster)}, \\
&\qquad u_{t,j} \leq \mathbb{1}\left(j = \arg\min_{j' \in [k]} \frac{\sum_{i=n_{test}}^{n} u_{i,j'} \|\boldsymbol{x}_t - \boldsymbol{x}_i\|_2^2}{\sum_{i=n_{test}}^{n} u_{i,j'}} \right), \forall j \in [k], \ \forall t \in [n_{test} - 1] \\
&\qquad \text{(test point } \boldsymbol{x}_t \text{ is assigned to the closest cluster).}
\end{aligned}
\tag{C.1}
$$

Which is exactly the formulation that is used in the experimental section. The contribution of test points to the prescriptive part of the objective is in $O(\frac{n_{test}}{n})$. This can be further adjusted by weighting the two types of observations with a parameter λ' that will then by hyper-parameter tuned on validation set similarly to λ. The prescriptive part of the objective function would become:

$$(1-\lambda)\left[\lambda'\left(\sum_{i=n_{test}}^{n}\sum_{j=1}^{k}u_{i,j}c(f_j(\boldsymbol{x}_i,\bar{\boldsymbol{z}}_i),\bar{\boldsymbol{z}}_i)\right)+(1-\lambda')\left(\sum_{i=0}^{n_{test}-1}\sum_{j=1}^{k}u_{i,j}c(f_j(\boldsymbol{x}_i,\bar{\boldsymbol{z}}_i),\bar{\boldsymbol{z}}_i)\right)\right]$$

Making the contribution of new points $O(\frac{(1-\lambda')n_{test}}{\lambda' n})$.

C.2 Experiment on the Cluster-While-Regress approach

We note for a vector x, $x[i]$, the i^{th} dimension of this vector.

In this simulation, we draw $n = 100$ 3-dimensional data points uniformly at random in $[0,1]^3$, and we set for each data point x, $y = 100x[3]$ if $x[1]^2 + x[2]^2 \leq 0.5$ and $y = -50x[3]$ otherwise. Then, we run a k-means clustering with $k = 2$, and then train within each resulting cluster a linear regression, and then we compare that with the Cluster-While-Regress that we highlight in the HPA approach. We get the following results (Table C.1):

Method	Estimated Coefficient in Cluster 1	Estimated Coefficient in Cluster 2	R2
Cluster then Regress	0.5	-1.3	-2.82
Cluster while Regress	100	-50	1

Table C.1: Comparison between Cluster then Regress and Cluster while Regress

This confirms computationally that there exists some advantage to clustering and regressing at the same time.

C.3 Hyper-parameters Tuning

For our computational tests in §4.6, we consider the following range of hyper-parameters:

1. **Linear Models (Elastic Net)**
    - L_1 Penalization Ratio: [0, 0.1, 0.2, 0.3, ..., 1]
    - Weight for L_1 and L_2 Penalization (α): [$1e^{-5}$, $1e^{-4}$, $1e^{-3}$, $1e^{-2}$, $1e^{-1}$, 1, 10, 100]
2. **Weight of the Predictive vs. Prescriptive Cost** λ: [$1e^{-5}$, $1e^{-4}$, $1e^{-3}$, $1e^{-2}$, $1e^{-1}$, 1, 10, 100, $1e^{3}$]
3. **Number of Clusters** k: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
4. **Classification and Regression Trees**
    - Maximum Depth: [3, 4, 5, 6, 10, None]
    - Minimum Samples per Leaf (%): [0.04, 0.06, 0.08]
    - Maximum Features to consider per split (%): [0.2, 0.4, 0.6, 0.8]
5. **XGBoost**
    - Minimum Child Weight: [1, 2, 3, ..., 20]
    - Gamma: [1, 2, 3, ..., 6]
    - SubSample: [0.8, 1.0]
    - Alpha: [0.5, 1, 2, 5]
    - ColSample ByTree: [0.6, 0.8, 1.0]
    - Maximum Depth: [3, 4, 5, 6, 10]
    - Learning Rate: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 1]
6. **Feed-Forward Neural Networks**
    - Number of Hidden Layers: [1, 2, 3, 4, 5]

- Number of Neurons per Layer: [2, 3, 5, 10, 20]
- Activation Function: tanh and reLU
- Solver: SGD and Adam
- Alpha: [0.0001, 0.05]
- Learning Rate: Constant and Adaptive

C.4 Hardware and Computation Time

For these computations and for the reported run-times, we have used 2.3 GHz Quad-Core Intel Core i7 Processor, with 16 GB of RAM and an Intel Iris Plus Graphics 1536 Graphic Card.

We also report the run-time of the tree-based HPA in Figure C-1.

Figure C-1: Average run-time of each iterations of the `cart-hpa-r` algorithm for select values of n, k and p on the synthetic experiment. Instance with above the 10,000 seconds threshold are instances above the time-limit that we have set, and are not solved to optimality.

Appendix D

Supplement for Chapter 5

D.1 Minimum Representation Learning Model

In this section we describe further the Minimal Representation Learning algorithm (`MRL`). We refer the reader to [18] for a detailed description of the general approach and theoretical analysis.

`MRL` uses the observed data from the dynamic system to partition the feature space into states of a finite deterministic MDP $(\mathcal{S}, \mathcal{A}, f, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $r : \mathcal{S} \to \mathbb{R}$ are the transition and outcome functions. Each feature $x \in \mathbb{R}^N$ is mapped to a unique MDP's state $\phi(x) = s \in \mathcal{S}$ with a mapping $\phi : \mathbb{R}^N \longrightarrow \mathcal{S}$. Figure D-1 illustrates this process. The constructed MDP constitutes a concise reduced representation of the system that approximates accurately the system's values, and therefore models COVID-19's evolution in our case.

Once the model is constructed, we can predict values of the dynamic system at features x and taking a sequence of actions $a_1, \ldots, a_h \in \mathcal{A}$ (e.g., restrictive mobility measures), by mapping the feature vector x to its corresponding state $s := \phi(x)$ in the MDP representation and then extracting the value of the representation MDP

starting at s and taking actions $a_1, \ldots, a_H$, i.e.,

$$\hat{V}(x) = r(f(s)) + \sum_{t=1}^{h} r(f(s, a_1 \ldots a_t)), \tag{D.1}$$

where $\hat{V}(x)$ is the predicted value and $f(s, a_1 \ldots a_t) = f(\ldots f(f(s, a_1), a_2) \ldots, a_t)$.

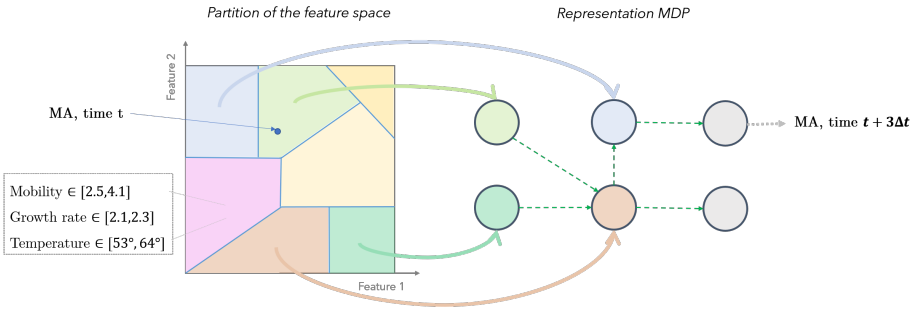


Figure D-1: Illustration of the representation MDP construction. The feature space is partitioned into a finite number of regions (left). A representation MDP is then constructed and each region is mapped (full arrows) to a state of the MDP (right). To make a prediction, a region at a given date is mapped with its features to a region of the feature space, then mapped to the corresponding state in the representation MDP.

To learn this concise MDP representation using transition data, the `MRL` algorithm iteratively splits the feature space to approximate an MDP representation that is consistent with the transition data. Such a representation is called *coherent* (see Definition 4.1 in [18]). More precisely, a mapping of the feature space $\hat{\phi}$ into states of an MDP is defined to be coherent if any two data points (feature vectors) are mapped to the same state and share the same outcome, i.e.,

$$\hat{\phi}(X_{i_1,t_1}) = \hat{\phi}(X_{i_2,h_2}) \implies R_{i_1,t_1} = R_{i_2,t_2}, \quad \forall i_1, i_2, \ \forall t_1, t_2.$$

and every two data points mapped to the same state and having observed the same action transition to data points mapped to the same state, i.e.,

$$\hat{\phi}(X_{i_1,t_1}) = \hat{\phi}(X_{i_2,t_2}) \ \text{ and } \ A_{i_1,t_1} = A_{i_2,t_2} \implies \hat{\phi}(X_{i_1,t_1+1}) = \hat{\phi}(X_{i_2,t_2+1}), \quad \forall i_1, i_2, \ \forall t_1, t_2.$$

[18] show that any coherent representation of the system converges to the most

concise (with smallest number of states) representation of the system. Hence, the `MRL` method learns the most concise MDP representation of the COVID-19 evolution. The `MRL` approach also provides provable guarantees as we summarize in Theorem 6 below.

Theorem 6 (see [18]) *Let N be the number of sample paths and T be the length of each path in the training data. Let V be the true value function of the system and $\hat{V}$ be the value estimated as in* (D.1) *for a prediction horizon $h \in \mathbb{N}$ with a coherent MDP representation. For all $\delta > 0$, with probability $1 - \delta$,*

$$|\hat{V} - V| = O\left(\sqrt{\frac{h}{NT}\left[\log(NT) + \log\left(\frac{1}{\delta}\right)\right]}\right).$$

We note that the same bound holds when the prediction is of the form $e^{\hat{V}}$, which is the case in COVID-19. That is, asymptotically $|e^{\hat{V}} - e^{V}| = O(e^{|\hat{V}-V|} - 1) = O(\hat{V} - V)$.

Model Formulation

In this section, we detail how we model COVID-19 evolution as a dynamic system. The prediction target $Y_{i,t}$ for a given region i, at day t can be either the *cumulative number of cases* or the *cumulative number of deaths*.

Introducing a time discretization. The day-to-day growth rate, cases and deaths are volatile due to fluctuations of testing numbers and seasonality among others. As a result, aiming to make accurate daily predictions is a complex problem. Furthermore, such a precise number is not necessary for the purpose of policy making. Instead of learning a day-to-day outcome, we introduce a number of days d onto which we *aggregate* the observations. Hence, for some arbitrary starting time t_0, we consider the aggregated outcomes on the set of times $t \in \{t_0, t_0 + d, t_0 + 2d, \ldots\}$ defined as

$$Y_{i,h}^{(d)} = \frac{1}{d}\sum_{\tau=0}^{d} Y_{i,t_0+ht-\tau}, \quad \forall h \leq T/d$$

where d is chosen as a trade-off between *learnability* and *precision*. Moreover, we observe some seasonal behavior on a short-term in the target time series. Averaging the curve allows us to recover the trend in a more stable and easier way that translates into an MDP.

From growth rate to outcomes. We define the MDP costs R as the logarithm of the aggregated growth rate:

$$R_{i,h} = \log\left(Y^{(d)}_{i,h+1}/Y^{(d)}_{i,h}\right).$$

Suppose the goal is to predict the outcome in region i for a horizon $h_0 + H$, where the available data stops at time h_0. We use the constructed finite MDP representation by `MRL` to predict sequences of costs of the MDP | i.e., log aggregated growth rates | $\{\hat{R}_{i,h_0+h},\ h \in [1, H]\}$, deduce an estimate $\hat{V}_{i,h_0+H}$ of the value function $V_{i,h_0+H} = \sum_{h=0}^{H} R_{i,h_0+h} = \log\left(Y^{(d)}_{i,h_0+H}/Y^{(d)}_{i,h_0}\right)$, and recover an estimate of the target :

$$\hat{Y}^{(d)}_{i,h_0+H} = \hat{Y}^{(d)}_{i,h_0}\exp(\hat{V}_{i,h_0+H}) \approx Y^{(d)}_{i,h_0}\exp(V_{i,h_0+H}) = Y^{(d)}_{i,h_0+H}$$

Finally, for days t within two time steps, we assume that the growth rate is constant between two time steps, equal to the next step growth rate and predict the outcome as

$$\begin{aligned}\hat{Y}_{i,t_0+h_0d+t} &= \hat{Y}^{(d)}_{i,h_0+\lfloor t/d\rfloor}\cdot\exp\left(\frac{t-d\lfloor t/d\rfloor}{d}\cdot\hat{R}_{h_0+\lfloor t/d\rfloor}\right)\\ &\approx Y^{(d)}_{i,h_0+\lfloor t/d\rfloor}\cdot\exp\left(\frac{t-d\lfloor t/d\rfloor}{d}\cdot R_{h_0+\lfloor t/d\rfloor}\right) = Y_{i,t_1+d\lfloor t/d\rfloor}\left(\frac{Y_{i,t_1+d\lfloor t/d\rfloor+d}}{Y_{i,t_1+d\lfloor t/d\rfloor}}\right)^{\frac{t-d\lfloor t/d\rfloor}{d}}\end{aligned}$$

where $t_1 = t_0 + h_0 d$.

Clustering of the outcomes. While the set of outcomes is populated by *log growth rates* observed over time across regions, the `MRL` method requires this set to be finite. As suggested in [18], this additional characteristic could be achieved by defining a *distance threshold* ϵ, that will be used as a proxy for clustering the observed outcomes beforehand. This clustering creates batches of outcomes of the

form $[R-\epsilon, R+\epsilon]$ that we consider to be similar outcomes. In other words, the algorithm does not distinguish between two outcomes in the same batch and identifies outcomes within ϵ. Hence, this introduces an additional error in the prediction of order $H\epsilon$, where H is the horizon of the prediction. Naturally, there is a trade-off captured. Taking ϵ too small increases the number of states of the constructed MDP (as a high precision in the prediction is required), creating a more complex structure to learn and generalize, while ϵ too large tends to oversimplify the learnt MDP and increases the discretization error, in which case the algorithm's prediction, within an error $H\epsilon$, would be of low accuracy.

D.2 Nearest-Neighbor and Similarity-Weighted Time-Series

In this section, we discuss the convergence of the `KNN` model:

Let us define $m(\boldsymbol{x_{j\tau}}) = \boldsymbol{E}[Y_{j\tau}|\boldsymbol{X_{j\tau}} = \boldsymbol{x_{j\tau}}]$ the unknown regression function that we want to estimate from a compact subset $S_F \subset F$ to $\mathbb{R}$ where F is an infinite dimensional functional space.

The k-nearest neighbor estimator is defined as:

$$\hat{m}_{N,T}(\boldsymbol{x_{j\tau}}) = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} W_{it}(\boldsymbol{x_{j\tau}}; \boldsymbol{X_{it}})Y_{l\tau}}{\sum_{i=1}^{N}\sum_{t=1}^{T} W_{it}(\boldsymbol{x_{j\tau}}; \boldsymbol{X_{it}})} \tag{D.2}$$

where $N * T$ is the total number of observations, k is the number of nearest neighbors and $W_{it}(\boldsymbol{x_{j\tau}}; \boldsymbol{X_{it}})$ is the weight function usually taking the form of a kernel: $K\big(\frac{d(\boldsymbol{x}, \boldsymbol{X_{it}})}{h(\boldsymbol{x})}\big)$ for a distance metric d and a bandwidth h.

Let S_F be a compact subset of F, and $N_\delta(S_F)$ be the minimal number of open balls with radius δ in F which is necessary to cover S_F with centers $\chi_1, ..., \chi_{N_\delta(S_F)}$ respectively. In addition, let $\psi_{S_F}(\delta)$ be the Kolmogrov's δ-entropy of S_F. For all

$\chi \in S_F$, denote $k(\chi) = \arg\min_{k\in\{1,2,\ldots,N_\delta(S_F)\}} d(\chi, \chi_k)$,

$$s^2_{N*T,1} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(Y_{i,t}u_{i,t}, Y_{j,m}u_{j,m})|, \quad s^2_{N*T,2} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u'_{i,t}, u'_{j,m})|$$
$$s^2_{N*T,3} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u_{i,t}, u_{j,m})|, \qquad s^2_{N*T,4} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{N}\sum_{m=1}^{T} |Cov(u''_{i,t}, u''_{j,m})| \tag{D.3}$$

where

$$\begin{aligned} u_{i,t} &= I_{B(\chi_{k(\chi)}, Ch_{N*T}+\delta)}(\chi_{i,t}) \text{ for some } C > 0 \text{ and } 0 < h_{N*K} \to 0 \\ u'_{i,t} &= \frac{Y_{i,t}K\left(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\right)}{EK\left(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\right)} - \frac{E\left(Y_{i,t}K\left(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\right)\right)}{EK\left(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\right)} \\ u''_{i,t} &= \frac{K\left(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\right)}{EK\left(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\right)} - \frac{EK\left(\frac{d(\chi_k,\chi_{i,t})}{h_{N*T}(\chi_k)}\right)}{EK\left(\frac{d(\chi_k,\chi_1)}{h_{N*T}(\chi_k)}\right)} \end{aligned} \tag{D.4}$$

We define

$$s^2_{N*T} = \max(s^2_{N*T,1}, s^2_{N*T,2}, s^2_{N*T,3}, s^2_{N*T,4}) \tag{D.5}$$

Let C, C', C_1, C_2 be some strictly positive generic constants. The assumptions under which the theorem holds are the following:

$(A1)$ $\forall \delta > 0, P(\chi \in B(\chi, \delta)) =: \varphi_\chi(\delta) > 0$

and $\varphi_\chi(\cdot)$ is continuous and strictly increasing around 0 with $\varphi_\chi(0) = 0$.

$(A2)$ $\exists$ function $\phi(\cdot) \geq 0$, a bounded function $f(\cdot) > 0, \alpha > 0$ and $\tau > 0$ such that

(i) $\phi(0) = 0$ and $\lim_{\delta \to 0} \phi(\delta) = 0$.

(ii) $\lim_{\delta \to 0}(\phi(u\delta)/\phi(\delta)) = u^\alpha$ for $u > 0$

(iii) $\sup_{\chi \in S_F} |\varphi_\chi(\delta)/\phi(\delta) - f(\chi)| = \mathcal{O}(\delta^\tau),$ as $\delta \to 0$.

$(A3)$ The kernel function $K(.)$ is a nonnegative function over its support $[0, 1]$, and its derivative $K'(.)$ exists on $[0, 1]$

with $-\infty < C_1 < K'(t) < C_2 < 0$ and $K(1) > 0 \quad \forall t \in [0, 1]$.

$(A4)$ (i) $m(.)$ is a bounded Lipschitz operator of order β on $S_F,$ that is,

$\exists \beta > 0$ such that $\forall \chi_1, \chi_2 \in S_F, |m(\chi_1) - m(\chi_2)| \leq C d^\beta(\chi_1, \chi_2)$.

(ii) $\forall m \geq 2, E(|Y|^m | X = \chi) = \delta_m(\chi) < C$ with $\delta_m(\cdot)$ being continuous on S_F.

$(A5)$ The Kolmogorov's $\delta -$ entropy of S_F satisfies $\sum_{n=1}^{\infty} e^{(1-\omega)\psi_{S_F}(\frac{\log n}{n})} < \infty$

for some large enough ω.

$(A6)$ $\exists \alpha > 1$ and $p > 2$ such that $s_{N*T}^{-((\alpha+1)p/(\alpha+p))} = o((N*T)^{-\theta})$ for some large enough θ. (D.6)

For the convergence of the $\hat{m}_{N,T}(\boldsymbol{x_{j\tau}})$ in strong mixing time series data with a kernel as a weight function we will cite the following theorem.

Theorem 7 (see [111]) *Under the assumptions (A1)-(A6), if* $\lim_{N*T \to \infty} \frac{k}{N*T} = 0$, $\log^2 \frac{N*T}{k} < \psi_{S_F}(\frac{\log N*T}{N*T}) < \frac{k}{\log N*T}$ *and* $0 < C_1 < \frac{k}{\log^2 N*T} < C_2 < \infty$ *for* $N*T$ *large enough and* C_1,C_2 *some constants then*

$$\sup_{\boldsymbol{x} \in S_F} |\hat{m}_{N,T}(\boldsymbol{x}) - m(\boldsymbol{x})| = \mathcal{O}_{a.co}\left(\phi^{-1}\left(\frac{k}{N*T}\right)^\beta + \sqrt{\frac{s_{N*T}^2 \psi_{S_F}(\frac{\log N*T}{N*T})}{(N*T)^2}}\right)$$

Theorem 7 establishes uniform almost complete convergence of the estimator. The assumption stated in the theorem are quite usual for time series data. For those interested in this topic we recommend reading [111] and the references within.

D.3 Deep Learning Approach

In the time-series variation of k-means, the similarity between two time series is measured by the DTW distance metric, and the cluster centroids are computed with respect to it. More specifically, given two time series $A = (a_0, ..., a_n)$ and $B = (b_0, ..., b_m)$ the DTW distance from A to B is formulated as the following optimization problem.

$$DTW(A, B) = \text{minimize}_{\pi} \quad \left(\sum_{(i,j)\in\pi} d(a_i, b_j)^2 \right)^{1/2}, \tag{D.7}$$

with $\pi = [p_0, ..., \pi_K]$ being a path which is a list of index pairs such that for each $k \in \{0, 1, ..., K\}$, $\pi_k = (i_k, j_k)$, $i_k \in \{0, 1, ..., n-1\}$ and $j_k \in \{0, 1, ..., m-1\}$. Moreover, $\pi_0 = (0, 0)$, $\pi_K = (n-1, m-1)$, $i_{k-1} \leq i_k \leq i_{k-1}+1$ and $j_{k-1} \leq j_k \leq j_{k-1}+1$ where the distance function $d(\cdot, \cdot)$ in this case is the Euclidean distance. This method calculates an optimal match between the two sequences that has the minimal cost according to the used distance function.

Regarding Recurrent Neural Networks or RNNs, they are a category of neural networks that processes sequential data. RNNs model sequences of data so that each sample is assumed to be dependent on the previous one. Even though RNNs theoretically can process long sequences (also referred to as long term dependencies), in practice they do not perform well (see [141]). Moreover, they suffer from what is referred to as the gradient vanishing and exploding problem. LSTMs are designed to be a solution to those problems ([95]). Additionally, their special architecture helps them remember and learn long term dependencies. LSTM models use special types of units that help them remember previous data and while simultaneously addressing gradient problems. Furthermore, they are capable of processing and predicting large time series.

In our system, each LSTM Network consists of two distinct layers. Each LSTM

layer consists of multiple LSTM units connected sequentially. Each LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell is the key component of the LSTM unit as it stores all the useful previous information. The LSTM can add and/or remove information to/from the cell via regulators called gates. By using specific activation functions, gates determine how much of each component of the unit goes through to the cell state. In order to formally describe the LSTM unit we introduce the following notation. Let $x_t \in \mathbb{R}^d$ be the input vector in the LSTM unit, $f_t \in \mathbb{R}^h$ the forget gate's activation vector, $i_t \in \mathbb{R}^h$ the input gate's activation vector, $o_t \in \mathbb{R}^h$ the output gate's activation vector, $h_t \in \mathbb{R}^h$ the output vector of the unit, $\tilde{c}_t \in \mathbb{R}^h$ the cell input activation vector, and $c_t \in \mathbb{R}^h$ the cell state vector.

The forget gate's activation vector is given by the following equation $f_t = \sigma(U_f h_{t-1} + W_f x_t + b_f)$, where σ is the sigmoid activation function and $U_f \in \mathbb{R}^{h \times h}, W_f \in \mathbb{R}^{h \times d}$ and $b_f \in \mathbb{R}^h$. This gate controls the self-loop weight (between 0 and 1) of the RNN. Intuitively, a value equal to 0 means that the LSTM decides to forget the previous unit's output, while a value equal to 1 means that the LSTM decided to keep completely the information from the previous unit.

At the same time, we calculate the input gate activation vector using the equation $i_t = \sigma(U_i h_{t-1} + W_i x_t + b_i)$, where σ is the sigmoid activation function, $U_i \in \mathbb{R}^{h \times h}, W_i \in \mathbb{R}^{h \times d}$, and $b_i \in \mathbb{R}^h$. The input gate activation vector is element-wise multiplied (Hadamard product) with the cell input activation vector, which is given through equation $\tilde{c}_t = \sigma(U_c h_{t-1} + W_c x_t + b_c)$, where σ is the hyperbolic tangent activation function, $U_c \in \mathbb{R}^{h \times h}, W_c \in \mathbb{R}^{h \times d}$, and $b_c \in \mathbb{R}^h$. The cell input activation function denotes the new candidate value for the cell and the input gate activation denotes the quantity of the new candidate value that will eventually be stored in the cell. Similar to what we described above, a value equal to 0 means that the LSTM decides to not take into consideration the new cell candidate value, while a value equal to 1 represents that the LSTM forwards completely the new candidate value to the cell.

The cell state vector is calculated using equation $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$, where $\circ$ denotes the Hadamard product. The new cell state is given by summing the remaining, multiplied by f_t, information from the previous cell, with the new candidate cell

value multiplied by i_t. As before, this shows how much consideration we put into the candidate cell value.

Finally, the output value h_t is given by a filtered state of the cell state. As before, by computing the output's gate activation vector $o_t = \sigma(U_o h_{t-1} + W_o x_t + b_o)$, where σ is the sigmoid activation function, $U_f \in \mathbb{R}^{h \times h}, W_f \in \mathbb{R}^{h \times d}$, and $b_f \in \mathbb{R}^h$, we decide which part of the cell state will remain. Then, the cell state c_t passes through a hyperbolic tangent activation function, that scales the values between -1 and 1, and is element-wise multiplied with o_t, resulting in the output $h_t = o_t \circ \sigma(c_t)$, where σ is the hyperbolic tangent activation function. In Figure D-2, a complete LSTM unit is depicted.

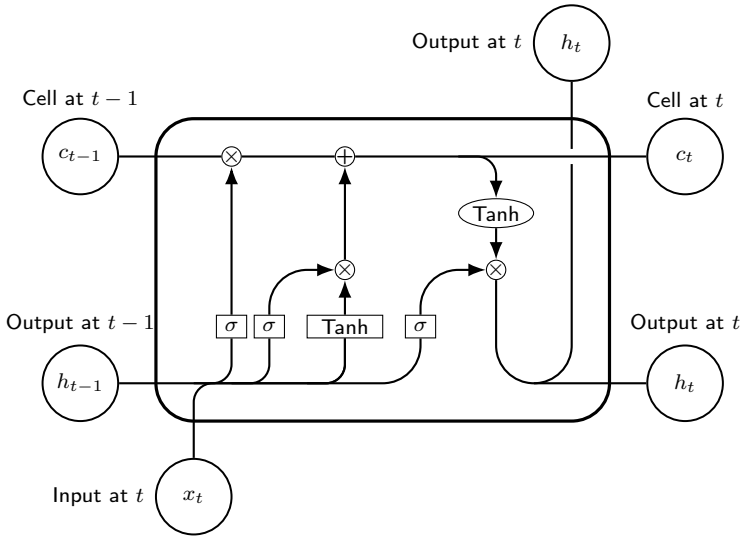


Figure D-2: An LSTM unit.

The complete architecture of our system is depicted in the following figure.

Bounds on Prediction Accuracy

In this subsection, we establish a theoretical bound on the predictive accuracy of the `Bi-LSTM` network. To accomplish this we impose the following reasonable assumptions:

Assumption 1: The input data are bounded.

Assumption 2: The spectral norms of weight matrices are bounded. More specif-

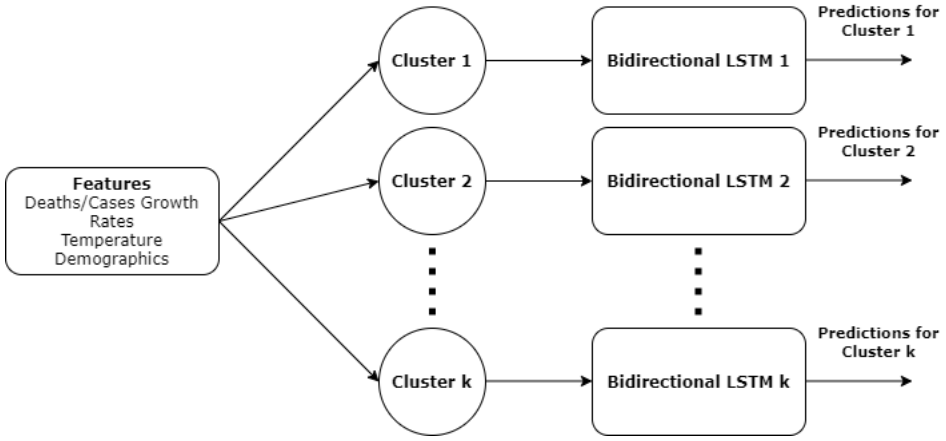


Figure D-3: Architecture of the complete system.

ically, $||W_f||_2 \leq B_{W_f}$, $||W_i||_2 \leq B_{W_i}$, $||W_c||_2 \leq B_{W_c}$, $||W_o||_2 \leq B_{W_o}$, $||U_f||_2 \leq B_{U_f}$, $||U_i||_2 \leq B_{U_i}$, $||U_c||_2 \leq B_{U_c}$, $||U_o||_2 \leq B_{U_o}$.

Assumption 3: The activation function of the output σ is Lipschitz continuous with parameter ρ and $\sigma(0) = 0$.

Theorem 8 (see [47]) *If Assumptions 1-3 hold, then for* $(x_t, z_t)_{t=1}^T$ *and* $S = \left\{(x_{i,t}, z_{i,t})_{t=1}^T, i = 1, ..., m\right\}$ *i.i.d. samples drawn from any underlying distribution over* $\mathbb{R}^{d_x \times T} \times \{1, .., K\}$*, with probability at least* $1 - \delta$ *over* S*, for every margin value* $\gamma > 0$ *and every* $f_t \in \mathcal{F}_{g,t}$ *for integer* $t \leq T$ *we have that:*

$$\mathbb{P}[\hat{z} \neq z] \leq \hat{\mathcal{R}}_\gamma(f_t) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}} + \mathcal{O}\left(\frac{d\rho B_V \min\{\sqrt{d}, B_{W_c} B_x \frac{\beta^t - 1}{\beta - 1}\}\sqrt{\log(\frac{\theta^t - 1}{\theta - 1} d\sqrt{m})}}{\sqrt{m}\gamma}\right), \tag{D.8}$$

where $\hat{z}$ *is the estimation of the LSTM,* $\hat{\mathcal{R}}_\gamma(f_t)$ *is the empirical risk,* $\mathcal{F}_{g,t} = \{(X_t, z_t) \mapsto g(f_t(X_t), z_t) : f_t \in \mathcal{F}_t\}$*, where* $\mathcal{F}_t$ *is the class of mappings from the first* t *inputs to the* $t-th$ *output and* g *the loss function,* $d_x = m$ *is the number of features,* K *is the total number of labels,* d *is the maximum dimension of the matrices,* B_V *is the bound on the spectral norm of the matrix applied to the output* h_t*, and finally,* B_x *is the bound of the second norm of each data point,* $\beta = \max\{||f_j||_\infty + B_{U_c}||i_j||_\infty||o_j||_\infty\}$ *and* $\theta = \beta + B_{U_f} + B_{U_i} + B_{U_o}$.

This result follows from [47]. The bound it establishes holds asymptotically for

regression and is of the order of $\mathcal{O}(\frac{\sqrt{\log\frac{1}{\delta}}\gamma+d\min\{\sqrt{d},\beta^t\}\sqrt{\log(\theta^t d\sqrt{m})}}{\sqrt{m}\gamma})$. Since, the bidirectional LSTM consists of two independent LSTMs the generalization bounds hold asymptotically also for the Bi-directional LSTM.

D.4 Epidemiology Approach

We begin by briefly describing the original SEIRD model and then build on it to formulate the multi-peak `C-SEIRD` model. The SEIRD model is a compartmental epidemiology model. It assumes that the total population is of size N and can be broken into five sub-populations: susceptible (S), exposed (E), infected (I), recovered (R) and deceased (D). Individuals move through these sub-populations, and the aggregate of this movement is described by the following set of differential equations:

$$\frac{dS}{dt} = -\frac{\beta SI}{N}, \quad \frac{dE}{dt} = \frac{\beta SI}{N} - \frac{E}{\alpha}, \quad \frac{dI}{dt} = \frac{E}{\alpha} - (\gamma+\mu)I, \quad \frac{dR}{dt} = \gamma I, \quad \frac{dD}{dt} = \mu I \tag{D.9}$$

In these equations β represents the infection rate of the disease, α represents the average incubation time (i.e., how long it takes for an individual to move from getting the disease to being contagious), γ denotes the recovery rate of infected individuals, and μ denotes the mortality rate.

Nevertheless, this formulation assumes that the parameters are static. If this were the case, then any area with COVID-19 would have only experienced a single wave (peak) of the disease before the virus had time to mutate or people had enough time to lose their immunity. However the multiple waves of COVID-19 seen across the United States demonstrate that the parameters are not static. Instead, population behavior and environmental conditions have driven multiple waves, leading to time-dependent parameters.

D.5 Trade-offs between Models

When choosing a prediction model for the evolution of the disease, there are several trade-offs to consider. Some models, such as epidemiological models, impose a lot of structure on the problem based on prior knowledge, while others, such as ML methods, learn general classes of functions from the data, with little to no problem-specific structure. There is a trade-off between leveraging subject-matter expertise, and being able to learn complex patterns from historical data and additional features such as weather or mobility. Additionally, some models perform better than others in terms of predictive accuracy but are less interpretable or need significantly more data to train. Based on these trade-offs, it is often unclear which model to choose; this results in a wide variety (more than 50) of state-of-the-art models used by the CDC for this prediction task. We consider four classes of models: reinforcement learning models, machine learning models, deep learning methods, and epidemiological models. Instead of choosing one of these classes for the prediction task, we combine all of these to bring together the best of all worlds. More specifically:

(i) **Feature-based Markovian representation approach**: This approach is designed to exploit the fact that the time series of growth rates present some similarities across time and region, conditioned on what has been observed in the recent history. Formalized under the *Markovian* assumption, the MRL method constructs a discrete Markov Decision Process from data. While the discretization we perform allows us to summarize information more efficiently, it induces a systematic lower bound in the error, resulting from the fact that the system is continuous. This limiting factor can be mitigated by increasing the granularity of the state space, but this might impact the stability of the constructed MDP. An alternative solution that we have developed in this paper consists in constructing an MRL-ensemble model.

(ii) **Nearest-neighbor weighted time series approach**: The main benefit of the Nearest Neighbor method is that is can make very accurate predictions with only a few data points as training. It captures behaviors that were observed in

the past and uses them in order to make predictions for the future. In addition, contrary to other methods that are often black box, one can identify from which neighbors the predicting patterns come from. This is a very desirable trait; it can be used to cluster many regions using the degree of similarity among them, as well as to interpret and explain the origin of the predictions. On the other hand, the fact that it uses only the observed patterns to make predictions can be a drawback if there is very little data available. If a new pattern arises, a nearest neighbor might not be able to accurately predict its evolution. However, after observing the first few time periods, the method can use those to make estimates about the future.

(iii) **Deep Learning approach**: The goal of the Bi-LSTM model is to analyze and discover previously unobserved patterns in large amounts of sequentially dependent data. By introducing a time-series clustering layer that groups states based on their individual characteristics, we create distinct Bi-LSTM networks to produce more accurate predictions for states in each cluster. The Bi-LSTM method has the advantage that it can also incorporate external features beyond historical death/cases growth rates including weather data and demographics. Nevertheless, there are two limitations in this method. First, the Bi-LSTM requires a substantial amount of data in order to be trained properly and produce accurate predictions. Additionally, the architecture we consider uses random initialization of the cluster centroids, (e.g. the DTW clustering algorithm randomly initializes the initial centroids). As a consequence, the final clusters are not always the same, and therefore, the predictions of the model may not always be consistent.

(iv) **Epidemiological approach**: The goal of the C-SEIRD model is to understand the progression of diseases through a population. The C-SEIRD brings with it a structure that includes information about epidemiology beyond what is contained in the data. Even if cases have had an upward trajectory through the entirety of the dataset, and all the other models predict that cases will continue

to grow, the C-SEIRD will identify that there is a point where the cases will peak. This means that the epidemiology model will perform the best when there is little data. This model is able to compensate for the lack of information with its structure. However the structure is also the main limitation of the model. It will not be able to hyper-fit to specific variations in the data, as the structure forces the model to average between fluctuations. This means that the model, when trained well, will often produce small errors, wMAPES often under the 10% mark, but it will rarely get as close to less than 1% errors as our aggregate model often produces. The C-SEIRD model is also very interpretable as it is easy to see how and why the model is making its predictions by taking a look at the estimated infection/recovery rates as well as the population size.

D.6 Proof of Theorem 1

Useful Definitions

Definition 1: *A metric on a set X is a function (called distance function) $d : X \times X \to [0, \infty)$. For any $x, y, z \in X$, the following three axioms are satisfied:*

1. $d(x, y) = 0 \iff x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

Definition 2: *A distance function is called absolutely homogeneous if for $x, y \in X$ and $\alpha \in \mathbb{R}$ the following holds: $d(\alpha x, \alpha y) = |\alpha| d(x, y)$.*

Proof:

1. **(In-Sample Predictions)**

   The result follows by contradiction. Let $m_0 \in \mathcal{M}$, we assume that the function defined by $f^0(\hat{y}_{i,t,m}, m \in \mathcal{M}) = \hat{y}_{i,t,m_0}, \forall i, t$ belongs to the set $\mathcal{F}$. That

is, f^0 is feasible in Problem (5.5). We can deduce directly by optimality of f_i that the objective function is lower, that is, $\sum_{t=T_{val}}^{T} |y_{i,t} - f_i(\hat{y}_{i,t,m}, m \in \mathcal{M})| \leq \sum_{t=T_{val}}^{T} |y_{i,t} - f^0(\hat{y}_{i,t,m}, m \in \mathcal{M})| = \sum_{t=T_{val}}^{T} |y_{i,t} - \hat{y}_{i,t,m_0}|$.

General Distance Functions: Again, the result follows by contradiction. Let $m_0 \in \mathcal{M}$, we assume that the function defined by $f^0(\hat{y}_{i,t,m}, m \in \mathcal{M}) = \hat{y}_{i,t,m_0}, \forall i, t$ belongs to the set $\mathcal{F}$. That is, f^0 is feasible in Problem (5.5). We can deduce directly by optimality of f_i that the objective function is lower, that is, $\sum_{t=T_{val}}^{T} d(y_{i,t}, f_i(\hat{y}_{i,t,m}, m \in \mathcal{M})) \leq \sum_{t=T_{val}}^{T} d(y_{i,t}, f^0(\hat{y}_{i,t,m}, m \in \mathcal{M})) = \sum_{t=T_{val}}^{T} d(y_{i,t}, \hat{y}_{i,t,m_0})$.

■

2. **(Out-of-Sample Predictions)**

For simplicity of notations, we denote y_i, for $i \in [N \times H]$ the random variable of the true target value (for example cases and deaths) out-of-sample for a particular region at a particular time, $\hat{y}_i$ the corresponding predicted value by the aggregate model and $\hat{y}_{i,m}$ the predicted value for individual model $m \in \mathcal{M}$.

(i) **Robustness:** Using the same notations, let $\boldsymbol{\lambda} > 0$ s.t. $\sum_{m \in \mathcal{M}} \lambda_m = 1$. We have that:

$$
\begin{aligned}
E[|y_i - \hat{y}_i|] &= E[|y_i - \sum_{m \in \mathcal{M}} \lambda_m \hat{y}_{i,m}|] = E[|\sum_{m \in \mathcal{M}} \lambda_m (y_i - \hat{y}_{i,m})|] \\
&\leq E[\sum_{m \in \mathcal{M}} |\lambda_m (y_i - \hat{y}_{i,m})|] \\
&= E[\sum_{m \in \mathcal{M}} \lambda_m |(y_i - \hat{y}_{i,m})|] = \sum_{m \in \mathcal{M}} \lambda_m E[|(y_i - \hat{y}_{i,m})|] \\
&\leq \sum_{m \in \mathcal{M}} \lambda_m \max_{k \in \mathcal{M}} E[|(y_i - \hat{y}_{i,k})|] \\
&= \max_{k \in \mathcal{M}} E[|(y_i - \hat{y}_{i,k})|](\sum_{m \in \mathcal{M}} \lambda_m) = \max_{k \in \mathcal{M}} E[|(y_i - \hat{y}_{i,k})|] \quad ■
\end{aligned}
\tag{D.10}
$$

Absolutely Homogeneous Distance Functions:

$$
\begin{aligned}
E[d(y_i, \hat{y}_i)] = E[d(y_i, \sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m})] = E[d(\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_i, \sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m})] \leq \\
\leq E[d(\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_i, \sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m^*})] = E[\sum_{m\in\mathcal{M}} \lambda_m d(\hat{y}_i, \hat{y}_{i,m^*})] = \\
= \sum_{m\in\mathcal{M}} \lambda_m E[d(\hat{y}_i, \hat{y}_{i,m^*})] = E[d(\hat{y}_i, \hat{y}_{i,m^*})] \quad \blacksquare
\end{aligned}
\tag{D.11}
$$

where $m^* = \arg\max_{m\in\mathcal{M}} d(\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_i, \sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m})$.

(ii) **Variance:** Using the same notations, let $\boldsymbol{\lambda} > 0$ s.t. $\sum_{m\in\mathcal{M}} \lambda_m = 1$.
We have that:

$$
\begin{aligned}
var(\hat{y}_i) = var(\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m}) = \mathbb{E}[\big(\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m}\big)^2] - \big(\mathbb{E}[\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m}]\big)^2 \\
= \mathbb{E}[\sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n \hat{y}_{i,m} \hat{y}_{i,n}] - \big(\mathbb{E}[\sum_{m\in\mathcal{M}} \lambda_m \hat{y}_{i,m}]\big)^2 \\
= \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n \mathbb{E}[\hat{y}_{i,m} \hat{y}_{i,n}] - \big(\sum_{m\in\mathcal{M}} \lambda_m \mathbb{E}[\hat{y}_{i,m}]\big)^2 \\
= \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n \mathbb{E}[\hat{y}_{i,m} \hat{y}_{i,n}] - \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n \mathbb{E}[\hat{y}_{i,m}] \mathbb{E}[\hat{y}_{i,n}] \\
= \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n \big(\mathbb{E}[\hat{y}_{i,m} \hat{y}_{i,n}] - \mathbb{E}[\hat{y}_{i,m}] \mathbb{E}[\hat{y}_{i,n}]\big) \\
= \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}} \lambda_m \lambda_n Cov(\hat{y}_{i,m}, \hat{y}_{i,n}) \\
= \sum_{m\in\mathcal{M}} \lambda_m^2 var(\hat{y}_{i,m}) + 2 \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{M}, n>m} \lambda_m \lambda_n Cov(\hat{y}_{i,m}, \hat{y}_{i,n}).
\end{aligned}
\tag{D.12}
$$

Now, let $V^* = \max_{m\in\mathcal{M}} var(\hat{y}_{i,m})$. The equation (D.12) becomes:

$$
\begin{aligned}
\sum_{m\in\mathcal{M}} \lambda_m^2 var(\hat{y}_{i,m}) + 2 \sum_{m\in\mathcal{M}} \sum_{n>m} \lambda_m \lambda_n Cov(\hat{y}_{i,m}, \hat{y}_{i,n}) \\
\leq V^* \big(\sum_{m\in\mathcal{M}} \lambda_m^2 + 2 \sum_{m\in\mathcal{M}} \sum_{n>m} \lambda_m \lambda_n\big)
\end{aligned}
\tag{D.13}
$$

We need to maximize $\sum_{m\in\mathcal{M}}\lambda_m^2 + 2\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{M},n>m}\lambda_m\lambda_n$ subject to the constraint that $\sum\limits_{m\in\mathcal{M}}\lambda_m = 1$ and $\lambda > 0$. Since, $\boldsymbol{\lambda}$ belongs to a convex subset of $\mathbb{R}^{|\mathcal{M}|}$, we can use Lagrange multipliers. Let p the corresponding Lagrange multiplier, $f(\boldsymbol{\lambda}) = \sum_{m\in\mathcal{M}}\lambda_m^2 + 2\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{M},n>m}\lambda_m\lambda_n$ and $g(\boldsymbol{\lambda}) = \sum\limits_{m\in\mathcal{M}}\lambda_m - 1$. We have that:

$$\nabla f(\boldsymbol{\lambda}) - p\nabla g(\boldsymbol{\lambda}) = 0 \implies$$
$$\begin{pmatrix} 2\lambda_1 \\ 2\lambda_2 \\ . \\ . \\ 2\lambda_{|\mathcal{M}|} \end{pmatrix} + 2\begin{pmatrix} \lambda_2 + ... + \lambda_{|\mathcal{M}|} \\ \lambda_1 + \lambda_3 + ... + \lambda_{|\mathcal{M}|} \\ . \\ . \\ \lambda_1 + \lambda_2 + ... + \lambda_{|\mathcal{M}|-1} \end{pmatrix} - p\begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix} = 0 \tag{D.14}$$

From the relation above, we observe that for every $i = 1, ..., |\mathcal{M}|$, we have that $\lambda_i = p/2 - \mathbf{1}^T\boldsymbol{\lambda}_{-i}$, where $\boldsymbol{\lambda}_{-i}$ is the $|\mathcal{M}| - 1$ dimensional vector that contains every λ_j for $j \neq i$ and $\mathbf{1}$ is the $|\mathcal{M}| - 1$ dimensional vector that contains 1's at every coordinate. By symmetry, we obtain that $\lambda_1 = \lambda_2 = ... = \lambda_{|\mathcal{M}|}$. Since, $\sum\limits_{m\in\mathcal{M}}\lambda_m = 1$, we obtain that $\lambda_1 = \lambda_2 = ... = \lambda_{|\mathcal{M}|} = \frac{1}{|\mathcal{M}|}$. Therefore, returning to equation (D.13), we obtain that:

$$V^*\Big(\sum_{m\in\mathcal{M}}\lambda_m^2 + 2\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{M},n>m}\lambda_m\lambda_n\Big) \leq$$
$$\underbrace{V^*\Big(\sum_{m\in\mathcal{M}}\frac{1}{|\mathcal{M}|^2} + 2\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{M},n>m}\frac{1}{|\mathcal{M}|^2}\Big)}_{=1} = V^* \tag{D.15}$$

■

D.7 Data Sources and Features

We collected cases and deaths related data from the Center for Systems Science and Engineering of John Hopkins. State-level social distancing policies data were retrieved from the related COVID19StatePolicy Github repository, while global population

mobility reports were collected from Google. Moreover, we used weather historical data provided by the National Climatic Data Center (NCDC) of National Oceanic and Atmospheric Administration (NOAA) and Demographics data provided by the U.S Census Bureau.

Our proposed models use this data as features in different ways. The `MRL` (and by extension `r-MRL`) feature space was constructed based on the historical growth rates, on both cases and deaths, including multiple lags to capture short-term and long-term effects. Ratios on differences between cases and deaths helped as well in refining the MDP state space. Regarding the `Bi-LSTM` and the `KNN` models, experimental results showed that in addition to deaths- and cases-related features, for case prediction, past mobility and temperature significantly improved (by more than 20%) the accuracy of the models. For death prediction, past case growth rates also improved the predictive power of the models. Finally, in the predictions starting on January 2021, we have also included vaccination rates as a feature in all our models, except the `C-SEIRD`, which is not feature-based.

D.8 Results with COVID-19 Data: CDC Benchmark Figures and Tables

We present here the figures and tables corresponding to the graphs in §5.4, for the CDC Benchmark on both deaths and cases.

D.8.1 Benchmark for Deaths

	9/5/2020		9/12/2020		9/19/2020		9/26/2020	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
COVIDhub-ensemble	0.007	3	0.008	1	0.011	1	0.016	1
LNQ-ens1	0.008	11	0.009	3	0.013	3	0.019	2
JCB-PRM	0.006	1	0.009	2	0.013	2	0.019	3
YYG-ParamSearch	0.008	8	0.010	4	0.014	4	0.020	4
UCLA-SuEIR	0.007	6	0.010	5	0.014	5	0.021	5
Covid19Sim-Simulator	0.007	5	0.010	6	0.015	6	0.021	6
`MIT-Cassandra`	**0.012**	**30**	**0.015**	**23**	**0.019**	**11**	**0.022**	**7**
Karlen-pypm	0.008	13	0.011	7	0.018	8	0.025	8
USC-SI_kJalpha	0.008	9	0.011	8	0.017	7	0.025	9
OliverWyman-Navigator	0.009	15	0.014	15	0.019	10	0.026	10
SteveMcConnell-CovidComplete	0.006	2	0.012	9	0.018	9	0.027	11
IowaStateLW-STEM	0.010	23	0.015	21	0.021	17	0.028	12
NotreDame-mobility	0.011	24	0.016	24	0.023	19	0.029	13
CEID-Walk	0.007	4	0.012	10	0.020	12	0.030	14
COVIDhub-baseline	0.008	7	0.012	12	0.020	13	0.030	15
⋮								
RPI_UW-Mob_Collision	0.012	31	0.029	33	0.053	35	0.086	35

Table D.1: Selection of model results for predicting September deaths.

	11/07/2020		11/14/2020		11/21/2020		11/28/2020	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
`MIT-Cassandra`	**0.013**	**23**	**0.024**	**39**	**0.043**	**39**	**0.06**	**38**
MSRA-DeepST	0.01	1	0.016	5	0.027	7	0.038	13
GT-DeepCOVID	0.01	2	0.015	1	0.025	4	0.034	4
LNQ-ens1	0.01	3	0.016	3	0.025	5	0.036	5
OliverWyman-Navigator	0.011	4	0.017	8	0.028	14	0.04	16
LANL-GrowthRate	0.011	5	0.019	11	0.03	17	0.043	19
COVIDhub-ensemble	0.011	6	0.018	10	0.029	16	0.039	15
USC-SI_kJalpha	0.011	7	0.017	7	0.027	6	0.036	9
Karlen-pypm	0.012	8	0.016	4	0.023	1	0.029	1
UMass-MechBayes	0.012	9	0.016	6	0.025	3	0.032	3
Google_Harvard-CPF	0.012	10	0.016	2	0.023	2	0.03	2
SteveMcConnell-CovidComplete	0.012	11	0.018	9	0.027	8	0.037	11
UCSD_NEU-DeepGLEAM	0.012	12	0.019	12	0.032	19	0.045	20
MOBS-GLEAM_COVID	0.012	13	0.019	13	0.032	21	0.046	21
CEID-Walk	0.012	14	0.021	25	0.037	31	0.053	31
COVIDhub-baseline	0.012	15	0.021	26	0.037	30	0.052	30
⋮								
TTU-squider	0.054	50	0.068	49	0.088	49	0.11	49

Table D.2: Selection of model results for predicting November deaths.

	2/6/2021		2/13/2021		2/20/2021		2/27/2021	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
`MIT-Cassandra`	**0.011**	**1**	**0.019**	**1**	**0.028**	**5**	**0.041**	**14**
OliverWyman-Navigator	0.020	2	0.023	2	0.031	11	0.039	10
LNQ-ens1	0.020	3	0.023	4	0.030	10	0.037	9
UCLA-SuEIR	0.021	4	0.027	15	0.034	17	0.042	16
Microsoft-DeepSTIA	0.021	5	0.026	8	0.034	16	0.051	22
COVIDhub-ensemble	0.021	6	0.023	3	0.028	6	0.033	4
USC-SI_kJalpha	0.021	7	0.023	5	0.028	7	0.035	8
COVIDhub-baseline	0.021	8	0.028	20	0.044	32	0.059	33
CEID-Walk	0.022	9	0.029	22	0.045	33	0.061	35
MOBS-GLEAM_COVID	0.022	10	0.026	9	0.034	15	0.041	12
UCSD_NEU-DeepGLEAM	0.022	11	0.027	13	0.035	19	0.043	18
Karlen-pypm	0.022	12	0.027	14	0.032	12	0.039	11
COVIDhub-trained_ensemble	0.023	13	0.025	7	0.026	1	0.031	1
SteveMcConnell-CovidComplete	0.023	14	0.026	11	0.029	9	0.033	5
BPagano-RtDriven	0.023	15	0.028	19	0.039	21	0.048	21
⋮								
JHUAPL-Bucky	0.071	42	0.086	41	0.099	41	0.107	41

Table D.3: Selection of model results for predicting February deaths.

D.8.2 Benchmark for Cases

	9/5/2020		9/12/2020		9/19/2020		9/26/2020	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
LNQ-ens1	0.124	2	0.136	1	0.154	1	0.175	1
COVIDhub-ensemble	0.120	1	0.148	2	0.164	2	0.185	2
Covid19Sim-Simulator	0.128	3	0.159	3	0.171	3	0.191	3
CEID-Walk	0.170	15	0.206	10	0.207	5	0.194	4
COVIDhub-baseline	0.155	10	0.195	8	0.205	4	0.195	5
UMich-RidgeTfReg	0.225	22	0.214	12	0.222	9	0.218	6
`MIT-Cassandra`	**0.168**	**14**	**0.189**	**7**	**0.211**	**7**	**0.224**	**7**
IowaStateLW-STEM	0.205	21	0.263	18	0.262	15	0.252	8
Karlen-pypm	0.172	16	0.216	13	0.245	11	0.253	9
JHU_IDD-CovidSP	0.175	18	0.175	4	0.215	8	0.256	10
USC-SI_kJalpha	0.156	11	0.179	6	0.209	6	0.266	11
UCLA-SuEIR	0.135	4	0.208	11	0.238	10	0.266	12
JHUAPL-Bucky	0.176	19	0.218	14	0.248	12	0.274	13
CU-scenario_low	0.153	5	0.234	16	0.292	17	0.317	14
LANL-GrowthRate	0.189	20	0.179	5	0.249	13	0.318	15
⋮								
CU-scenario_high	0.153	8	0.273	19	0.393	22	0.513	22

Table D.4: Selection of model results for predicting September cases.

	11/7/2020		11/14/2020		11/21/2020		11/28/2020	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
Karlen-pypm	0.235	3	0.258	2	0.264	2	0.273	1
`MIT-Cassandra`	**0.260**	**6**	**0.254**	**1**	**0.254**	**1**	**0.278**	**2**
LANL-GrowthRate	0.234	2	0.275	3	0.304	3	0.310	3
CU-scenario_high	0.284	11	0.328	5	0.341	5	0.326	4
LNQ-ens1	0.218	1	0.276	4	0.321	4	0.340	5
CU-scenario_mid	0.283	9	0.331	7	0.361	6	0.363	6
CU-select	0.283	10	0.331	8	0.361	7	0.363	7
CU-nochange	0.286	12	0.340	10	0.370	9	0.366	8
UCSB-ACTS	0.331	21	0.375	14	0.406	11	0.404	9
JHU_UNC_GAS-StatMechPool	0.337	22	0.387	17	0.414	13	0.405	10
UMich-RidgeTfReg	0.283	8	0.342	11	0.380	10	0.406	11
JHU_CSSE-DECOM	0.313	16	0.364	13	0.406	12	0.414	12
JCB-PRM	0.313	15	0.377	15	0.430	15	0.451	13
COVIDhub-ensemble	0.319	18	0.393	19	0.443	18	0.455	14
JHUAPL-Bucky	0.345	26	0.391	18	0.432	16	0.457	15
⋮								
CovidAnalytics-DELPHI	0.589	30	0.636	29	0.675	29	0.712	29

Table D.5: Selection of model results for predicting November cases.

	02/06/2021		02/13/2021		02/20/2021		02/27/2021	
Model	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**	**wMAPE**	**Rank**
`MIT-Cassandra`	**0.652**	**39**	**0.45**	**26**	**0.39**	**19**	**0.352**	**15**
Geneva-DetGrowth	0.107	1	0.213	7	0.188	10	0.292	11
LNQ-ens1	0.12	2	0.107	1	0.168	8	0.195	5
Microsoft-DeepSTIA	0.139	3	0.231	8	0.393	21	0.489	27
JHU_IDD-CovidSP	0.163	4	0.164	2	0.273	15	0.312	12
UChicagoCHATTOPADHYAY-UnIT	0.163	5	0.187	3	0.156	3	0.154	3
UVA-Ensemble	0.166	6	0.278	12	0.39	20	0.424	21
LANL-GrowthRate	0.173	7	0.205	5	0.174	9	0.201	7
CU-select	0.196	8	0.356	17	0.586	36	0.714	36
Karlen-pypm	0.2	9	0.194	4	0.146	1	0.139	1
RobertWalraven-ESG	0.207	10	0.212	6	0.158	4	0.14	2
IEM_MED-CovidProject	0.21	11	0.268	10	0.16	5	0.184	4
USC-SI_kJalpha	0.218	12	0.247	9	0.346	17	0.361	18
BPagano-RtDriven	0.242	13	0.338	14	0.481	28	0.525	28
JHU_CSSE-DECOM	0.242	14	0.4	21	0.618	39	0.756	39
IowaStateLW-STEM	0.245	15	0.356	16	0.514	32	0.587	29
⋮								
CU-scenario_mid	0.691	41	0.919	41	0.238	13	0.395	19

Table D.6: Selection of model results for predicting February cases.

D.9 Proof of Proposition 1

Proof: We have under the constraints of (5.13) the following:

$$
\begin{aligned}
&\min_{\boldsymbol{v}} \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}(n^{0}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}+ \\
&\quad \sum_{k=1}^{K}(n^{1,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k})+n^{2,k}_{t,i,j}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k}))) \iff \\
&\min_{\boldsymbol{v}} \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}((n^{0}_{t-1,i,j}-\sum_{k=1}^{K}v^{1,k}_{t,i,j}-c^{0}_{t-1,i,j})\hat{p}_{t,i,j}m_{t,i,j}+ \\
&\quad \sum_{k=1}^{K}((n^{1,k}_{t-1,i,j}-v^{2,k}_{t,i,j}+v^{1,k}_{t,i,j}-c^{1,k}_{t-1,i,j})\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k})+ \\
&\quad (n^{2,k}_{t-1,i,j}+v^{2,k}_{t,i,j}-c^{2,k}_{t,i,j})\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k}))) \iff \\
&\min_{\boldsymbol{v}} \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}((n^{0}_{0,i,j}-\sum_{s=1}^{t}(\sum_{k=1}^{K}v^{1,k}_{s,i,j}+c^{0}_{s-1,i,j}))\hat{p}_{t,i,j}m_{t,i,j}+ \\
&\quad \sum_{k=1}^{K}((n^{1,k}_{0,i,j}-\sum_{s=1}^{t}(v^{2,k}_{s,i,j}-v^{1,k}_{s,i,j}+c^{1,k}_{s-1,i,j}))\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k})+ \\
&\quad (n^{2,k}_{0,i,j}+\sum_{s=1}^{t}(v^{2,k}_{s,i,j}-c^{2,k}_{s-1,i,j}))\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k}))) \iff \\
&\min_{\boldsymbol{v}} \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\hat{p}_{t,i,j}m_{t,i,j}[-p_{1,k}\sum_{s=1}^{t}v^{1,k}_{s,i,j}+(p_{1,k}-p_{2,k})\sum_{s=1}^{t}v^{2,k}_{s,i,j}] \iff \\
&\max_{\boldsymbol{v}} \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}((\sum_{s=t}^{T}\hat{p}_{s,i,j}m_{s,i,j}p_{1,k})v^{1,k}_{t,i,j}+((\sum_{s=t}^{T}\hat{p}_{s,i,j}m_{s,i,j}(p_{2,k}-p_{1,k}))v^{2,k}_{t,i,j})) \qquad \blacksquare
\end{aligned}
$$

D.10 Dynamic Program for the Vaccine Allocation Problem

Alternatively to the Mixed-Integer Program proposed in §5.6, we can use a Dynamic Program that accounts for the feedback mechanism. We formulate the problem as follows:

- State: $S_t = (n^{0}_{t,i,j}, n^{1,k}_{t,i,j}, n^{2,k}_{t,i,j}, \hat{p}_{t,i,j})$.

- Decision Variable: $\boldsymbol{v}_{t+1} = (v^{1,k}_{t+1,i,j}, v^{2,k}_{t+1,i,j})$.

- Update: $S_{t+1} = (n^0_{t+1,i,j}, n^{1,k}_{t+1,i,j}, n^{2,k}_{t+1,i,j}, \hat{p}_{t+1,i,j})$ with:

  1. $n^0_{t+1,i,j} = n^0_{t,i,j} - \sum_{k=1}^{K} v^{1,k}_{t+1,i,j} - \hat{p}_{t,i,j} n^0_{t,i,j}$,
  2. $n^{1,k}_{t+1,i,j} = n^{1,k}_{t,i,j} - v^{2,k}_{t+1,i,j} + v^{1,k}_{t+1,i,j} - \hat{p}_{t,i,j}(1 - p_{1,k}) n^{1,k}_{t,i,j}$,
  3. $n^{2,k}_{t+1,i,j} = n^{2,k}_{t,i,j} + v^{2,k}_{t+1,i,j} - \hat{p}_{t,i,j}(1 - p_{2,k}) n^{2,k}_{t,i,j}$,
  4. $\hat{p}_{t+1,i,j} = f_{agg}(\hat{p}_{t,i,j}, n^0_{t+1,i,j}, n^{1,k}_{t+1,i,j}, n^{2,k}_{t+1,i,j})$.

- Initial Conditions: at time t=0, S_t is fully known (population in each sub group in each county, how any people are already vaccinated, with which vaccine, as well as the initial prevalence estimation today).

- Value Function: $\mathcal{F}_t(S_{t-1}) = \min_{\boldsymbol{v}_t} \sum_{i=1}^{I} \sum_{j=1}^{J} (n^0_{t,i,j}(S_{t-1}, v_t) \hat{p}_{t,i,j}(S_{t-1}, v_t) m_{t,i,j} + \sum_{k=1}^{K} (n^{1,k}_{t,i,j}(S_{t-1}, v_t) \hat{p}_{t,i,j}(S_{t-1}, v_t) m_{t,i,j}(1-p_{1,k}) + n^{2,k}_{t,i,j}(S_{t-1}, v_t) \hat{p}_{t,i,j}(S_{t-1}, v_t) m_{t,i,j}(1-p_{2,k}))) + \mathcal{F}_{t-1}(S_{t-2})$
  s.t. $n^0_{t,i,j}, n^{1,k}_{t,i,j}, n^{2,k}_{t,i,j}, v^{1,k}_{t,i,j}, v^{2,k}_{t,i,j} \geq 0$.

- Objective: Compute $\mathcal{F}_T$ and get $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_T$.

While this problem can be solved theoretically thanks to its Dynamic Programming structure, the size of the feature space ($I \times J \times (3K+1)$) as well as the complexity of the function f_{agg} to estimate the prevalence with both past data and vaccination makes the problem very hard to solve.

D.11 Adding Robustness to the Vaccine Allocation Problem

We discuss three main sources of uncertainty in Formulation (5.14):

- The predicted prevalence $\hat{p}$: from both the prediction of the number of detected cases itself and from the choice of α.

- The mortality rate m.

- The vaccine efficacy p_1 and p_2.

While uncertainty can be taken into account for all of these three inputs, both mortality rate and vaccine efficacy are easier to evaluate accurately at any given time. Consequently, the predicted prevalence, which is an output of MIT-Cassandra is the uncertain variable that we want to account for. It can easily be done by introducing an uncertainty set $\mathcal{U}$ and a placeholder decision variable θ to put the objective constant in the constraint.

$$
\begin{aligned}
\min_{\boldsymbol{v},\theta} \quad & \theta \\
\text{s.t.} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J}(v_{t,i,j}^{1,k}+v_{t,i,j}^{2,k}) \le V_{max,k,t}, \quad \forall t\in[T],\forall k\in[K], \\
& \sum_{i=1}^{I}\sum_{k=1}^{K}(v_{t,i,j}^{1,k}+v_{t,i,j}^{2,k}) \ge V_{min,t,j}, \quad \forall t\in[T],\forall j\in[J], \\
& n_{t,i,j}^{0} = n_{t-1,i,j}^{0} - \sum_{k=1}^{K} v_{t,i,j}^{1,k} - c_{t-1,i,j}^{0}, \quad \forall t\in[T],\forall i\in[I],\forall j\in[J], \\
& n_{t,i,j}^{1,k} = n_{t-1,i,j}^{1,k} - v_{t,i,j}^{2,k} + v_{t,i,j}^{1,k} - c_{t-1,i,j}^{1,k}, \quad \forall t\in[T],\forall i\in[I],\forall j\in[J],\forall k\in[K], \\
& n_{t,i,j}^{2,k} = n_{t-1,i,j}^{2,k} + v_{t,i,j}^{2,k} - c_{t-1,i,j}^{2,k}, \quad \forall t\in[T], i\in[I],\forall j\in[J],\forall k\in[K], \\
& n_{t,i,j}^{0}, n_{t,i,j}^{1,k}, n_{t,i,j}^{2,k}, v_{t,i,j}^{1,k}, v_{t,i,j}^{2,k} \ge 0, \quad \forall t\in[T], i\in[I],\forall j\in[J],\forall k\in[K]. \\
& n_{t,i,j}^{2,k} = v_{t,i,j}^{2,k} = 0, \quad \forall t\in[T], i\in[I],\forall j\in[J],\forall k\in[K_1], \\
& \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}(n_{t,i,j}^{0}\hat{p}_{t,i,j}m_{t,i,j} + \sum_{k=1}^{K}(n_{t,i,j}^{1,k}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{1,k}) + n_{t,i,j}^{2,k}\hat{p}_{t,i,j}m_{t,i,j}(1-p_{2,k}))) \\
& \le \theta, \quad \forall \hat{p}\in\mathcal{U}.
\end{aligned}
\tag{D.16}
$$

Note that an uncertainty set on α only can be translated without loss of generality on an uncertainty set $\mathcal{U}$ on $\hat{p}$.

Additionally, if $\mathcal{U}$ is a box uncertainty set, i.e. each of the $\hat{p}_{t,i,j}$ has its own

uncertainty set $\mathcal{U}_{t,i,j}$, uncorrelated from the rest. Then by noting $\hat{p}^0_{t,i,j} = \max \mathcal{U}_{t,i,j}$, we have that the robust formulation is exactly the same as the nominal formulation, by replacing the prevalence by the worst-case scenario within the considered uncertainty set.

$$
\begin{aligned}
\min_{\boldsymbol{v}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}(n^0_{t,i,j}\hat{p}^0_{t,i,j}m_{t,i,j} + \sum_{k=1}^{K}(n^{1,k}_{t,i,j}\hat{p}^0_{t,i,j}m_{t,i,j}(1-p_{1,k}) + n^{2,k}_{t,i,j}\hat{p}^0_{t,i,j}m_{t,i,j}(1-p_{2,k})))\\
\text{s.t.} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \leq V_{max,k,t}, \quad \forall t \in [T], \forall k \in [K],\\
& \sum_{i=1}^{I}\sum_{k=1}^{K}(v^{1,k}_{t,i,j} + v^{2,k}_{t,i,j}) \geq V_{min,t,j}, \quad \forall t \in [T], \forall j \in [J],\\
& n^0_{t,i,j} = n^0_{t-1,i,j} - \sum_{k=1}^{K} v^{1,k}_{t,i,j} - c^0_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J],\\
& n^{1,k}_{t,i,j} = n^{1,k}_{t-1,i,j} - v^{2,k}_{t,i,j} + v^{1,k}_{t,i,j} - c^{1,k}_{t-1,i,j}, \quad \forall t \in [T], \forall i \in [I], \forall j \in [J], \forall k \in [K],\\
& n^{2,k}_{t,i,j} = n^{2,k}_{t-1,i,j} + v^{2,k}_{t,i,j} - c^{2,k}_{t-1,i,j}, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K],\\
& n^0_{t,i,j}, n^{1,k}_{t,i,j}, n^{2,k}_{t,i,j}, v^{1,k}_{t,i,j}, v^{2,k}_{t,i,j} \geq 0, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K].\\
& n^{2,k}_{t,i,j} = v^{2,k}_{t,i,j} = 0, \quad \forall t \in [T], i \in [I], \forall j \in [J], \forall k \in [K_1],
\end{aligned}
\tag{D.17}
$$

Which can be solved exactly as described in §5.6.

Identical observations can be drawn for robustness of the model in Formulation (5.13), which we have proven to be equivalent to Formulation (5.14). One of the advantages of this (5.13) is that we can construct uncertainty on β directly, to account for the error on the three sources mentioned earlier at once: prevalence $\hat{p}$, mortality m, and vaccine efficacy p_1 and p_2, instead of just $\hat{p}$. Two examples of how to construct these uncertainty sets are:

- To use box uncertainty sets, calibrated in a data-driven fashion by using the confidence intervals outputted by the model.
- To use a Central Limit Theorem uncertainty set on $\hat{p}_t$'s separately. Correlating

the regions and populations groups, but limiting the repeated extreme variations over multiple time periods.

D.12 Mortality Rate Table By Age Group

By using the age breakdown approach described, in §5.6, we estimate the mortality rates by age group at state-level, which we use for the vaccine allocation experiment. These estimations can be found in Table D.7 below.

Age Group	**Mortality Rate**
85+ years old	43.4%
75-84 years old	25.2%
65-74 years old	9.9%
50-64 years old	3.5%
40-49 years old	1.0%
30-39 years old	0.3%
18-29 years old	0.1%
5-17 years old	0.03%
0-4 years old	0.01%

Table D.7: Estimated Mortality Rate per Age Group.

D.13 Rate Ratios for Exposure and Mortality for COVID-19 by Age Group

The CDC computed the rate ratios of cases and deaths for different age groups compared to the 5-17 year old group. The full table is shown in Figure D-4 and can be interpreted as follows: everything else being equal, a person within the 75-84 years old age group, for example, is 2 times more likely to be detected positive for COVID-19 and 2800 times more likely to die from COVID-19.

Rate ratios compared to 5-17 year olds

	0—4 years	5—17 years	18—29 years	30—39 years	40—49 years	50—64 years	65—74 years	75—84 years	85+ years
Cases	<1x	Reference group	3x	2x	2x	2x	2x	2x	2x
Hospitalization	2x	Reference group	7x	10x	15x	25x	35x	55x	80x
Death	2x	Reference group	15x	45x	130x	400x	1100x	2800x	7900x

Figure D-4: Risk ratios of different age groups compared to the 5-17 year old age group can be used to estimate infection and death probabilities disaggregated by age group.

D.14 Results on the Estimated Age Breakdown of COVID-19 Cases for MA Counties

We evaluate the breakdown of detected COVID-19 cases in each MA county using census data and the approach described in Equation (5.16) and obtain Table D.8:

County	0-4	5-17	18-29	30-39	40-49	50-64	65-74	75-84	85+
Barnstable County	0.0	0.058	0.179	0.092	0.096	0.249	0.186	0.099	0.041
Berkshire County	0.0	0.065	0.228	0.111	0.111	0.238	0.14	0.072	0.035
Bristol County	0.0	0.08	0.244	0.134	0.133	0.226	0.104	0.053	0.025
Dukes County	0.0	0.069	0.181	0.119	0.122	0.243	0.169	0.071	0.025
Essex County	0.0	0.082	0.243	0.132	0.129	0.227	0.108	0.053	0.026
Franklin County	0.0	0.068	0.195	0.129	0.123	0.243	0.154	0.06	0.027
Hampden County	0.0	0.084	0.268	0.134	0.121	0.212	0.104	0.052	0.026
Hampshire County	0.0	0.053	0.408	0.097	0.093	0.18	0.104	0.045	0.021
Middlesex County	0.0	0.074	0.274	0.153	0.131	0.205	0.092	0.047	0.023
Nantucket County	0.0	0.076	0.218	0.163	0.161	0.222	0.097	0.042	0.021
Norfolk County	0.0	0.082	0.239	0.139	0.135	0.223	0.102	0.053	0.028
Plymouth County	0.0	0.085	0.224	0.123	0.131	0.24	0.117	0.058	0.023
Suffolk County	0.0	0.055	0.383	0.178	0.108	0.156	0.069	0.035	0.016

Worcester County	0.0	0.082	0.254	0.136	0.132	0.227	0.099	0.048	0.022

Table D.8: Fraction of the Population Infected (and Detected) by COVID-19 by Age Group for each MA County.

D.15 Flowchart Summary of the end-to-end Approach

The flowchart summarizing the entire end-to-end framework used in this paper can be found in Figure D-5.

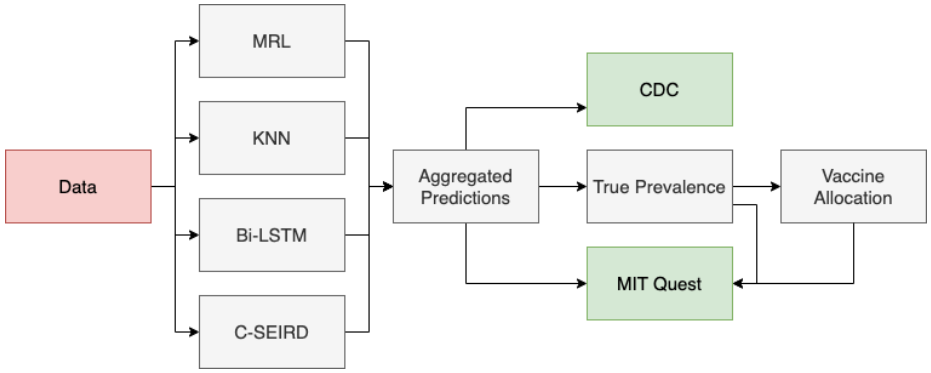


Figure D-5: Flowchart of the end-to-end approach presented in this paper.

Appendix E

Supplement for Chapter 6

E.1 Stop-words

The full list of stop-words removed in §**??** is:

{“ourselves", “hers", “between", “yourself", “but", “again", “there", “about", “once", “during", “out", “very", “having", “with", “they", “own", “an", “be", “some", “for", “do", “its", “yours", “such", “into", “of", “most", “itself", “other", “off", “is", “s", “am", “or", “who", “as", “from", “him", “each", “the", “themselves", “until", “below", “are", “we", “these", “your", “his", “through", “don", “nor", “me", “were", “her", “more", “himself", “this", “down", “should", “our", “their", “while", “above", “both", “up", “to", “ours", “had", “she", “all", “no", “when", “at", “any", “before", “them", “same", “and", “been", “have", “in", “will", “on", “does", “yourselves", “then", “that", “because", “what", “over", “why", “so", “can", “did", “not", “now", “under", “he", “you", “herself", “has", “just", “where", “too", “only", “myself", “which", “those", “i", “after", “few", “whom", “t", “being", “if", “theirs", “my", “against", “a", “by", “doing", “it", “how", “further", “was", “here", “than"}

Bibliography

[1] Mark Abraham, Steve Mitchelmore, Sean Collins, Jeff Maness, Mark Kistulinec, Shervin Khodabandeh, Daniel Hoenig, and Jody Visser. Profiting from personalization. *Boston Consulting Group*, 2017.

[2] Deepak Agarwal, Rahul Agrawal, Rajiv Khanna, and Nagaraj Kota. Estimating rates of rare events with multiple hierarchies through scalable log-linear models. 07 2010.

[3] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] Ravi Anupindi, Maqbool Dada, and Sachin Gupta. Estimation of consumer demand with stock-out based substitution: An application to vending machine products. *Marketing Science*, 17(4):406–423, 1998.

[5] Ali Aouad, Adam N Elmachtoub, Kris J Ferreira, and Ryan McNellis. Market segmentation trees. *arXiv preprint arXiv:1906.01174*, 2019.

[6] Susan Athey. 21. the impact of machine learning on economics. In *The economics of artificial intelligence*, pages 507–552. University of Chicago Press, 2019.

[7] Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

[8] Susan Athey and Guido W Imbens. The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives*, 31(2):3–32, 2017.

[9] Lennart Baardman, Setareh Borjian Boroujeni, Tamar Cohen-Hillel, Kiran Panchamgam, and Georgia Perakis. Detecting customer trends for optimal promotion targeting. *Available at SSRN: https://ssrn.com/abstract=3242529*, 2018.

[10] Lennart Baardman, Igor Levin, Georgia Perakis, and Divya Singhvi. Leveraging comparables for new product sales forecasting. *Available at SSRN 3086237*, 2017.

[11] Dirk M Barends, Margryt Teatske Oldenhof, Marjo J Vredenbregt, and Maarten J Nauta. Risk analysis of analytical validations by probabilistic modification of fmea. *Journal of pharmaceutical and biomedical analysis*, 64:82–86, 2012.

[12] Hamsa Bastani, Kimon Drakopoulos, Vishal Gupta, Jon Vlachogiannis, Cristos Hadjicristodoulou, Pagona Lagiou, Gkikas Magiorkinis, Dimitris Paraskevis, and Sotirious Tsiodras. Deploying an artificial intelligence system for covid-19 testing at the greek border. *SSRN*, 2021.

[13] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017.

[14] Luc Bégin, Pascal Germain, François Laviolette, and Jean-Francis Roy. Pac-bayesian bounds based on the rényi divergence. In *Artificial Intelligence and Statistics*, pages 435–444, 2016.

[15] Souhaib Ben Taieb and Rob J. Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International Journal of Forecasting*, 30(2):382–394, 2014.

[16] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[17] Yoshua Bengio, Nicolas Le Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. *Advances in neural information processing systems*, 18:123, 2006.

[18] Amine Bennouna, Dessislava Pachamanova, Georgia Perakis, and Omar Skali Lami. Learning the minimal representation of a dynamic system from transition data. *https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3785547*, 2021.

[19] Dirk Berkvens, Niko Speybroeck, Nicolas Praet, Amel Adel, and Emmanuel Lesaffre. Estimating disease prevalence in a bayesian framework using probabilistic constraints. *Epidemiology*, pages 145–153, 2006.

[20] Dimitri P Bertsekas et al. *Dynamic programming and optimal control: Vol. 1*. Athena scientific Belmont, 2000.

[21] Dimitris Bertsimas, Vassilis Digalakis Jr, Michael Linghzi Li, and Omar Skali Lami. Slowly varying regression under sparsity. *arXiv preprint arXiv:2102.10773*, 2021.

[22] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[23] Dimitris Bertsimas and Jack Dunn. *Machine Learning under a Modern Optimization Lens*. Dynamic Ideas, 2019.

[24] Dimitris Bertsimas, Jack Dunn, and Nishanth Mundru. Optimal prescriptive trees. *INFORMS Journal on Optimization*, 1(2):164–183, 2019.

[25] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *Mathematical Programming*, 167(2):235–292, 2018.

[26] Dimitris Bertsimas, Joshua Ivanhoe, Alexandre Jacquillat, Michael Li, Alessandro Previero, Omar Skali Lami, and Hamza Tazi Bouardi. Optimizing vaccine allocation to combat the covid-19 pandemic. *medRxiv*, 2020.

[27] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

[28] Dimitris Bertsimas, Nathan Kallus, Alexander M Weinstein, and Ying Daisy Zhuo. Personalized diabetes management using electronic medical records. *Diabetes care*, 40(2):210–217, 2017.

[29] Dimitris Bertsimas and Nishanth Mundru. Sparse convex regression. *INFORMS Journal on Computing*, 33(1):262–279, 2021.

[30] Dimitris Bertsimas, Agni Orfanoudaki, and Holly Wiberg. Interpretable clustering: An optimization approach. *Machine Learning*, (to appear), 2020.

[31] Dimitris Bertsimas and Omar Skali Lami. Holistic prescriptive analytics for continuous and constrained optimization problems. 2021.

[32] Lidia Betcheva, Feryal Erhun, Antoine Feylessoufi, Peter Fryers, Paulo Gonçalves, Houyuan Jiang, Paul A. Kattuman, Tom Pape, Anees Pari, Stefan Scholtes, and Carina Tyrrell. An adaptive research approach to covid-19 forecasting for regional health systems in england. *https://ssrn.com/abstract=3695258*, 2021.

[33] GÃŠrard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr):1063–1095, 2012.

[34] John R Birge, Ozan Candogan, and Yiding Feng. Controlling epidemic spread: reducing economic losses with targeted closures. *University of Chicago, Becker Friedman Institute for Economics Working Paper*, (2020-57), 2020.

[35] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[36] F Bogard, K Debray, and YQ Guo. Determination of sensor positions for predictive maintenance of revolving machines. *International Journal of Solids and Structures*, 39(12):3159–3173, 2002.

[37] Casper Solheim Bojer and Jens Peder Meldgaard. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603, 2021.

[38] Fred Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008.

[39] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole Statistics-Probability Series. Taylor & Francis, 1984.

[40] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[41] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[42] Leo Breiman and Nong Shang. Born again trees. 1996.

[43] C Richard Cassady and Erhan Kutanoglu. Integrating preventive maintenance planning and production scheduling for a single machine. *IEEE Transactions on reliability*, 54(2):304–309, 2005.

[44] O. Chapelle, Eren Manavoglu, and Rómer Rosales. Simple and scalable response prediction for display advertising. *ACM Trans. Intell. Syst. Technol.*, 5:61:1–61:34, 2014.

[45] Patrali Chatterjee, Donna L. Hoffman, and Thomas P. Novak. Modeling the clickstream: Implications for web-based advertising efforts. *Marketing Science*, 22(4):520–541, 2003.

[46] Ramnath K. Chellappa and K. Ravi Kumar. Examining the role of "free" product-augmenting online services in pricing and customer retention strategies. *Journal of Management Information Systems*, 22(1):355–377, 2005.

[47] Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks. *CoRR*, abs/1910.12947, 2019.

[48] Ningyuan Chen, Guillermo Gallego, and Zhuodong Tang. The use of binary choice forests to model and estimate discrete choices. *Available at SSRN 3430886*, 2019.

[49] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[50] Yi-Chun Chen and Velibor Mišić. Decision forest: A nonparametric approach to modeling irrational choice. *Available at SSRN 3376273*, 2020.

[51] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and causal parameters, 2016.

[52] X Chiementin, F Bolaers, L Rasolofondraibe, and J-P Dron. Restoration of a temporal indicator specific to each vibratory sources for a predictive maintenance. *Mechanical systems and signal processing*, 23(6):1909–1919, 2009.

[53] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[54] Jin-A Choi and Kiho Lim. Identifying machine learning techniques for classification of target advertising. *ICT Express*, 6(3):175 – 180, 2020.

[55] David I Cook, Val J Gebski, and Anthony C Keech. Subgroup analysis in clinical trials. *Medical Journal of Australia*, 180(6):289, 2004.

[56] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[57] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[58] Thomas M Cover. Rates of convergence for nearest neighbor procedures. In *Proceedings of the Hawaii International Conference on Systems Sciences*, volume 415, 1968.

[59] J D'Aeth, Shubhechyya Ghosal, Fiona Grimm, David Haw, Esma Koca, Krystal Lau, Huikang Liu, Stefano Moret, Dheeya Rizmie, P Smith, et al. Optimal hospital care scheduling during the sars-cov-2 pandemic. *Optimization Online*, 2021.

[60] James Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment planning under the multinomial logit model with totally unimodular constraint structures. 2013.

[61] Peter Dawson, Ralph Gailis, and Alaster Meehan. Detecting disease outbreaks using a combined bayesian network and particle filter approach. *Journal of theoretical biology*, 370:171–183, 2015.

[62] Rodrigo de Queiroz Souza and Alberto José Álvares. Fmea and fta analysis for application of the reliability centered maintenance methodology: case study on hydraulic turbines. In *ABCM Symposium Series in Mechatronics*, volume 3, pages 803–812, 2008.

[63] Bert De Reyck, Ioannis Fragkos, Yael Grushka-Cockayne, Casey Lichtendahl, Hammond Guerin, and Andrew Kritzer. Vungle inc. improves monetization using big data analytics. *INFORMS Journal on Applied Analytics*, 47(5):454–466, 2017.

[64] Emir Demirović, Peter J Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction+ optimisation for the knapsack problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 241–257. Springer, 2019.

[65] Emir Demirovic, Peter J Stuckey, James Bailey, Jeffrey Chan, Christopher Leckie, Kotagiri Ramamohanarao, and Tias Guns. Predict+ optimise with ranking objectives: Exhaustively learning linear functions. In *International Joint Conference on Artificial Intelligence*, pages 1078–1085, 2019.

[66] Nandini Dendukuri, Elham Rahme, Patrick Bélisle, and Lawrence Joseph. Bayesian sample size determination for prevalence and diagnostic test studies in the absence of a gold standard test. *Biometrics*, 60(2):388–397, 2004.

[67] Houtao Deng. Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics*, pages 1–11, 2018.

[68] Federica Di Castro and Enrico Bertini. Surrogate decision tree visualization interpreting and visualizing black-box classification models with surrogate decision tree. *CEUR Workshop Proceedings*, 2327, 2019. 2019 Joint ACM IUI Workshops, ACMIUI-WS 2019 ; Conference date: 20-03-2019.

[69] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[70] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[71] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[72] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, March 2004.

[73] Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *ArXiv*, 1710.08005, 2017.

[74] Claes Enøe, Marios P Georgiadis, and Wesley O Johnson. Estimation of sensitivity and specificity of diagnostic tests and disease prevalence when the true disease state is unknown. *Preventive veterinary medicine*, 45(1-2):61–81, 2000.

[75] Markus Ettl, Pavithra Harsha, Anna Papush, and Georgia Perakis. A data-driven approach to personalized bundle pricing and recommendation. *Manufacturing & Service Operations Management*, 22(3):461–480, 2020.

[76] Markus Ettl, Pavithra Harsha, Anna Papush, and Georgia Perakis. A data-driven approach to personalized bundle pricing and recommendation. *Manufacturing & Service Operations Management*, 22(3):461–480, 2020.

[77] T Evgeniou, M Fekom, A Ovchinnikov, R Porcher, C Pouchol, and N Vayatis. Pandemic lock-down, isolation, and exit policies based on machine learning predictions. 2020.

[78] Peter S Fader, Bruce GS Hardie, and Ka Lok Lee. "counting your customers" the easy way: An alternative to the pareto/nbd model. *Marketing science*, 24(2):275–284, 2005.

[79] Hamid Reza Feili, Navid Akar, Hossein Lotfizadeh, Mohammad Bairampour, and Sina Nasiri. Risk analysis of geothermal power plants using failure modes and effects analysis (fmea) technique. *Energy Conversion and Management*, 72:69–76, 2013.

[80] Kris Johnson Ferreira, Bin Hong Alex Lee, and David Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2016.

[81] Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 2019.

[82] Marc-André Filz, Jonas Ernst Bernhard Langner, Christoph Herrmann, and Sebastian Thiede. Data-driven failure mode and effect analysis (fmea) to enhance maintenance planning. *Computers in Industry*, 129:103451, 2021.

[83] Seth Flaxman, Swapnil Mishra, Axel Gandy, H Juliette T Unwin, Thomas A Mellan, Helen Coupland, Charles Whittaker, Harrison Zhu, Tresnia Berah, Jeffrey W Eaton, et al. Estimating the effects of non-pharmaceutical interventions on covid-19 in europe. *Nature*, 584(7820):257–261, 2020.

[84] L. Fridley. Improving online demand forecast using novel features in website data : a case study at zara. 2018.

[85] Guillermo Gallego, Anran Li, Van-Anh Truong, and Xinshang Wang. Approximation algorithms for product framing and pricing. *Operations Research*, 68(1):134–160, 2020.

[86] Wei Gao and Zhi-Hua Zhou. Towards convergence rate analysis of random forests for classification. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9300–9311. Curran Associates, Inc., 2020.

[87] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[88] Matt Goldman and Brian Quistorff. Pricing engine: Estimating causal impacts in real world business settings, 2018.

[89] Negin Golrezaei, Hamid Nazerzadeh, and Paat Rusmevichientong. Real-time optimization of personalized assortments. *Management Science*, 60(6):1532–1551, 2014.

[90] Robert B Gramacy and Herbert K H Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

[91] Philip Gross, Albert Boulanger, Marta Arias, David L Waltz, Philip M Long, Charles Lawson, Roger Anderson, Matthew Koenig, Mark Mastrocinque, William Fairechio, et al. Predicting electricity distribution feeder failures using machine learning susceptibility analysis. In *AAAI*, pages 1705–1711, 2006.

[92] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51, 02 2018.

[93] Satoshi Hara and Kohei Hayashi. Making tree ensembles interpretable: A bayesian model selection approach. *arXiv preprint arXiv:1606.09066*, 2016.

[94] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[95] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6:107–116, 1998.

[96] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[97] Jing Huang, Qing Chang, and Jorge Arinez. Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, 160:113701, 2020.

[98] Teng Huang, David Bergman, and Ram Gopal. Predictive and prescriptive analytics for location selection of add-on retail products. *Production and Operations Management*, 28(7):1858–1877, 2019.

[99] Michael A Johansson, Talia M Quandelacy, Sarah Kada, Pragati Venkata Prasad, Molly Steele, John T Brooks, Rachel B Slayton, Matthew Biggerstaff, and Jay C Butler. Sars-cov-2 transmission from people without covid-19 symptoms. *JAMA network open*, 4(1):e2035057–e2035057, 2021.

[100] Monica Johar, Vijay Mookerjee, and Sumit Sarkar. Selling vs. profiling: Optimizing the offer set in web-based personalization. *Information Systems Research*, 25(2):285–306, 2014.

[101] Lawrence Joseph, Theresa W Gyorkos, and Louis Coupal. Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard. *American journal of epidemiology*, 141(3):263–272, 1995.

[102] Deokwoo Jung, Zhenjie Zhang, and Marianne Winslett. Vibration analysis for iot enabled predictive maintenance. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1271–1282. IEEE, 2017.

[103] Nathan Kallus, Brenton Pennicooke, and Michele Santacatterina. More robust estimation of sample average treatment effects using kernel optimal matching in an observational study of spine surgical interventions. *ArXiv*, 1811.04274, 2018.

[104] Wagner A Kamakura and Gary J Russell. A probabilistic choice model for market segmentation and elasticity structure. *Journal of marketing research*, 26(4):379–390, 1989.

[105] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[106] Balaji Lakshminarayanan, Daniel Roy, and Yee Whye Teh. Top-down particle filtering for bayesian decision trees. In *International Conference on Machine Learning*, pages 280–288, 2013.

[107] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[108] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[109] Michael Lingzhi Li, Hamza Tazi Bouardi, Omar Skali Lami, Thomas A. Trikalinos, Nikolaos K. Trichakis, and Dimitris Bertsimas. Forecasting covid-19 and analyzing the effect of government interventions. *medRxiv*, 2020.

[110] Ruiyun Li, Sen Pei, Bin Chen, Yimeng Song, Tao Zhang, Wan Yang, and Jeffrey Shaman. Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (sars-cov-2). *Science*, 368(6490):489–493, 2020.

[111] Nengxiang Ling, Shuyu Meng, and Philippe Vieu. Uniform consistency rate of k nn regression estimation for functional time series data. *Journal of Nonparametric Statistics*, 31(2):451–468, 2019.

[112] Drew A Linzer, Jeffrey B Lewis, et al. polca: An r package for polytomous variable latent class analysis. *Journal of statistical software*, 42(10):1–29, 2011.

[113] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[114] Elisa F Long, Eike Nohdurft, and Stefan Spinler. Spatial resource allocation for emerging epidemics: A comparison of greedy, myopic, and dynamic policies. *Manufacturing & Service Operations Management*, 20(2):181–198, 2018.

[115] D. J. C. Mackay. *Introduction to Monte Carlo Methods*, pages 175–204. Springer Netherlands, Dordrecht, 1998.

[116] Ho-Yin Mak, Tinglong Dai, and Christopher S Tang. Managing two-dose covid-19 vaccine rollouts with limited supply. *Available at SSRN 3790836*, 2021.

[117] Laura Matrajt and Ira M Longini Jr. Optimizing vaccine allocation at different points in time during an epidemic. *PloS one*, 5(11):e13767, 2010.

[118] Peter McCullagh. *Generalized linear models*. Routledge, 2018.

[119] Jan Medlock and Alison P Galvani. Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708, 2009.

[120] Nicolai Meinshausen et al. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010.

[121] Locksley L McV Messam, Adam J Branscum, Michael T Collins, and Ian A Gardner. Frequentist and bayesian approaches to prevalence estimation using examples from johne's disease. *Animal Health Research Reviews*, 9(1):1–23, 2008.

[122] Gideon Meyerowitz-Katz and Lea Merone. A systematic review and meta-analysis of published research data on covid-19 infection-fatality rates. *International Journal of Infectious Diseases*, 2020.

[123] Velibor V Mišić and Georgia Perakis. Data analytics in operations management: A review. *Manufacturing & Service Operations Management*, 22(1):158–169, 2020.

[124] Niko Mohr and Holger Hürtgen. Achieving business impact with data. *Digital McKinsey*, 2018.

[125] Siuli Mukhopadhyay and Debraj Chakraborty. Estimation of undetected covid-19 infections in india. *medRxiv*, 2020.

[126] Serguei Netessine, Sergei Savin, and Wenqiang Xiao. Revenue management through dynamic cross selling in e-commerce retailing. *Operations Research*, 54(5):893–913, 2006.

[127] Daniel Oberski. Mixture models: Latent profile and latent class analysis. In *Modern statistical methods for HCI*, pages 275–287. Springer, 2016.

[128] Sadettin Orhan, Nizami Aktürk, and Veli Celik. Vibration monitoring for defect diagnosis of rolling element bearings as a predictive maintenance tool: Comprehensive case studies. *Ndt & E International*, 39(4):293–298, 2006.

[129] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[130] Georgia Perakis, Divya Singhvi, Omar Skali Lami, Ankit Mangal, Stephan Poninghaus, Alison Borenstein, and Jiong Wei Lua. Ancillary services in targeted advertising: from prediction to prescription. *Manufacturing & Service Operations Management*, 2021.

[131] Georgia Perakis, Divya Singhvi, Omar Skali Lami, and Leann Thayaparan. Fighting covid-19: A multipeak sir based model for learning waves and optimizing testing. 2021.

[132] Steven J Phipps, R Quentin Grafton, and Tom Kompas. Robust estimates of the true (population) infection rate for covid-19: a backcasting approach. *Royal Society open science*, 7(11):200909, 2020.

[133] Victor M Preciado, Michael Zargham, Chinwendu Enyioha, Ali Jadbabaie, and George Pappas. Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks. In *52nd IEEE conference on decision and control*, pages 7486–7491. IEEE, 2013.

[134] Kamalini Ramdas, Ara Darzi, and Sanjay Jain. 'test, re-test, re-test': using inaccurate tests to greatly increase the accuracy of covid-19 testing. *Nature medicine*, 26(6):810–811, 2020.

[135] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[136] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[137] Jonathan Rougier. Ensemble averaging and mean squared error. *Journal of Climate*, 29(24):8865 – 8870, 2016.

[138] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[139] David C Schmittlein, Donald G Morrison, and Richard Colombo. Counting your customers: Who-are they and what will they do next? *Management science*, 33(1):1–24, 1987.

[140] David C Schmittlein and Robert A Peterson. Customer base analysis: An industrial purchase process application. *Marketing Science*, 13(1):41–67, 1994.

[141] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[142] Joseph Sill, Gábor Takács, Lester W. Mackey, and David Lin. Feature-weighted linear stacking. *CoRR*, abs/0911.0460, 2009.

[143] Peter Sollich and Anders Krogh. Learning with ensembles: How overfitting can be useful. In D. Touretzky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[144] Peter J Stuckey, Tias Guns, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, Jeffrey Chan, et al. Dynamic programming for predict+ optimise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1444–1451, 2020.

[145] Peng Sun, Liu Yang, and Francis De Véricourt. Selfish drug allocation for containing an international influenza pandemic at the onset. *Operations Research*, 57(6):1320–1332, 2009.

[146] Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, and Alessandro Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820, 2014.

[147] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[148] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[149] Matt Taddy, Chun-Sheng Chen, Jun Yu, and Mitch Wyle. Bayesian and empirical bayesian forests. *arXiv preprint arXiv:1502.02312*, 2015.

[150] Kalyan T Talluri, Garrett Van Ryzin, and Garrett Van Ryzin. *The theory and practice of revenue management*, volume 1. Springer, 2004.

[151] Cher Ming Tan and Nagarajan Raghavan. Imperfect predictive maintenance model for multi-state systems with multiple failure modes and element failure dependency. In *2010 Prognostics and System Health Management Conference*, pages 1–12. IEEE, 2010.

[152] Hamdi Taplak, Selçuk Erkaya, and Ibrahim Uzmay. Experimental analysis on fault detection for a direct coupled rotor-bearing system. *Measurement*, 46(1):336–344, 2013.

[153] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[154] Kai Ming Ting and Ian H. Witten. Stacking bagged and dagged models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 367–375, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[155] Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *CoRR*, abs/1105.5466, 1999.

[156] Ashleigh R Tuite, David N Fisman, Jeffrey C Kwong, and Amy L Greer. Optimal pandemic influenza vaccine allocation strategies for the canadian population. *PloS one*, 5(5):e10520, 2010.

[157] Theja Tulabandhula and Cynthia Rudin. Machine learning with operational costs. *The Journal of Machine Learning Research*, 14(1):1989–2028, 2013.

[158] US Food and Drug Administration. Fda statement on following the authorized dosing schedules for covid-19 vaccines, 2021.

[159] Jeroen K Vermunt and Jay Magidson. Latent class models for classification. *Computational Statistics & Data Analysis*, 41(3-4):531–537, 2003.

[160] Gustavo Vulcano, Garrett Van Ryzin, and Richard Ratliff. Estimating primary demand for substitutable products from sales transaction data. *Operations Research*, 60(2):313–334, 2012.

[161] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

[162] Ruxian Wang, Maqbool Dada, and Ozge Sahin. Pricing ancillary service subscriptions. *Management Science*, 65(10):4712–4732, 2019.

[163] Wendai Wang and Dan Dragomir-Daescu. Reliability quantification of induction motors-accelerated degradation testing approach. In *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No. 02CH37318)*, pages 325–331. IEEE, 2002.

[164] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

[165] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[166] S Yakowitz. Nearest-neighbour methods for time series analysis. *Journal of time series analysis*, 8(2):235–247, 1987.

[167] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 67(2):301–320, 2005.