Interpretable Contextual Newsvendor Models: A Tree-Based Method to Solving Data-Driven Newsvendor Problems

Author:

Parisa Keshavarz

Supervisors:

Dr. Jonathan Yumeng Li

Dr. Erick Delage

A thesis submitted to the University of Ottawa
in fulfilment of the thesis requirement for the degree
of Master of Science in Management

Telfer School of Management
University of Ottawa

Abstract

In this thesis, we consider contextual newsvendor problems where one seeks to determine ordering quantities of perishable products based on the observations of past demands and some features (such as seasonality, weather forecasts, economic indicators, etc.) related to the demand. We propose solving the problems via a single-step optimal decision-tree approach. Unlike the traditional two-step approach that first predicts a demand distribution based on given features and then optimizes the order quantity, our approach seeks to determine a tree-based ordering policy that directly maps given features to optimal order quantities. We show how the optimal policies can be found by solving mixed-integer programming (MIP) problems. The tree structure overcomes the black-box nature of most machine learning algorithms while reaching better performance than simple solutions such as linear regression. In addition to risk-neutral newsvendor problems, we further extend the method to address risk-averse newsvendor problems formulated based on Conditional Value-at-Risk (CVaR). Numerical experiments on synthetic and real-world data suggest that our approach outperforms existing approaches with the same objective function, such as the ERM-based convex optimization model which is referred to as Ban and Rudin's big data newsvendor model, and quantile regression decision trees.

Contents

List of Tables

List of Figures

List of Abbreviations

CART	Classification and Regression Trees
CDF	Cumulative Distribution Function
CVaR	Conditional Value at Risk
DNN	Deep Neural Network
ERM	Empirical Risk Minimization
KNN	K-Nearest Neighbors
KO	Kernel Optimization
MIO	Mixed-Integer Optimization
MIP	Mixed-Integer Programming
OCT	Optimal Classification Tree
RF	Random Forest
SAA	Sample Average Approximation
SVM	Support Vector Machine
VaR	Value at Risk

List of Symbols

α	Complexity parameter
β	Degree of risk aversion
ϵ	Vector consisting of smallest non-zero distance between adjacent values of each feature
ϵ_{max}	Maximum component of vector ϵ
γ	Threshold above which the conditional expectation of losses would fall at probability level β
λ	Regularization parameter
$\mathcal{L}(t)$	Set of ancestors of node t whose left branch was followed on the path from the root node to node t
$\mathcal{R}(t)$	Set of ancestors of node t whose right branch was followed on the path from the root node to node t
$\mathcal{T}_B$	Set of branch nodes in a decision tree
$\mathcal{T}_L$	Set of leaf nodes in a decision tree
$\tilde{c}_i$	Actual cost of $i-th$ observation
a_{jt}	Binary variable that is 1 if feature j is selected to branch on in branch node t; 0 otherwise
b_t	Break point in branch node t

c_{it}	Cost incurred if $i-th$ observation is using $t-th$ decision policy
D_i	Demand (output variable) of $i-th$ observation
d_t	Binary variable that is 1 if branch node t applies a split; 0 otherwise
l_t	Binary variabe that is 1 if leaf node t contains any points; 0 otherwise
N_{min}	Minimum number of points required in each leaf node
o_{it}	Overage cost of $i-th$ observation in leaf t
$p(t)$	Parent node of node t
q_t^j	Policy parameter corresponding to feature j in leaf node t
u_{it}	Underage cost of $i-th$ observation in leaf t
x_i^j	Value of feature j for the $i-th$ observation
z_{it}	Binary variable that is 1 if point i is assigned to leaf node t; 0 otherwise
b	Unit backordering cost
D_T	Depth of the decision tree
h	Unit holding cost
i	Index for number of observations
j	Index for number of features
M	A large value
m	Index for ancestors of a node
n	Number of observations
p	Number of features
T	Number of nodes in a fully-grown tree
t	Index for leaves in the decision tree

Chapter 1

Introduction

The newsvendor problem is one of the most popular problems in the area of operations and supply chain management, where a retailer needs to decide on the inventory level/order quantity of a perishable product before observing the actual demand. The retailer incurs an *overage cost* for each unit of product that remains unsold at the end of each period (unsold items are discarded at the end of each period) and an *underage cost* for each unit of unsatisfied demand. This problem aims to choose an order quantity that minimizes the expected sum of these costs.

In traditional approaches to solving the newsvendor problem (see [1] for a review), it is assumed that the demand follows a known family of probability distributions up to unknown parameters that need to be estimated [2]. This assumption leads to two-step solutions where first a demand distribution is estimated, and then the order quantity is optimized. Although these approaches lead to intuitive solutions, estimating the probability distribution of the demand is prone to errors which are propagated into the second step and might lead to solutions that are not optimal [3]. In reality, decision-makers usually make decisions without having any prior knowledge of the demand distribution [2]. As a result, there have been many efforts to relax the assumption of a known demand distribution in the newsvendor problem using historical data.

Even though most of the above-mentioned efforts to date are only based on past demand observations, decision-makers potentially have access to large amounts of

relevant information about factors that affect the demand in addition to the past demand observations. In the the area of contextual optimization, these factors are called covariates, features, attributes, or explanatory variables. Ban and Rudin [4] defined covariates as the "exogenous variables that are predictors of the demand and are available to the decision-maker before the ordering occurs", and analytically quantified the value of feature information by comparing decisions made with and without the use of any features. In the case of contextual newsvendor problems, the two-step approach would estimate the conditional distribution of the demand with respect to the features and then calculate the order quantity achieving the optimal decision. However, the feature information is not used in the second step. Our focus is to solve the newsvendor problem by exploiting contextual data where instead of two separate steps, we use a single step (without having to estimate the demand distribution) and solve the problem to optimality using the Empirical Risk Minimization (ERM) principle [5] from machine learning. The single-step approach avoids the complexity of demand model specification in high dimensional feature space and the errors caused by the conditional probability estimation.

It is important to mention that even though we are trying to create a single step machine learning algorithm to avoid the errors caused by distribution estimation in the conventional two-step approaches, there are still some challenges that arise when working with contextual optimization models in machine learning that should not be ignored. The usual criteria for success in machine learning is considered to be the predictive accuracy (which in the case of the newsvendor problem results in lower costs). However, observations from various domains show that sometimes users are more interested in the interpretation of the results induced from the data [6]. Interpretability[1] in the context of machine learning is defined as "the ability to explain or to present in understandable terms to a human" [8].

Molnar explains in his book on Interpretable Machine Learning [9] that when using a machine learning algorithm for predictive modeling, one should always ask "Do I just need to know **what** is predicted? Or is it equally important to know **why** the prediction was made, and possibly pay for the interpretability with a drop in the

[1]Note that interpretability and explainability are often used interchangeably [7]

predictive performance?". The answer to this question is highly dependant on the application and domain being studied. According to Doshi-Velez and Kim [8], there are two scenarios in which explanation is not required. The first concerns problems in low-risk environments that do not require explanation since making a mistake in such cases does not have serious consequences (e.g., movie recommendation systems). The second is when the problem is well-studied and validated in a real application in such a way that the system's decisions are considered to be trustworthy (e.g., postal code sorting, where a camera is used to capture the images of postal envelops and Optical Character Recognition (OCR) approach is used to recognize the images of handwritten postal codes).

A common belief in machine learning is that there is a trade-off between a model's performance and its interpretability. Even though some suggest that this trade-off often might not be true [10], it has resulted in a tendency among scholars and practitioners to adopt more complicated models known as black-boxes to reach a better performance, even if interpretability is sacrificed. The inability of humans to interpret the results of these black-box models has proven to be problematic in different domains [11]. One solution to better understand black-box models that has been widely used in recent years is *interpretable machine learning*, where a second model is created to explain the black-box model. However, Rudin [10] explains that there are critical issues[2] associated with this solution that makes using an inherently interpretable model a better strategy.

These issues convinced us to prioritize interpretability and seek for a naturally interpretable solution. Among machine learning methods, decision trees is a popular approach known for its interpretability. Decision trees are known for their resemblance to human decision-making and they provide advantages that can contribute to any decision-making problem. Aside from interpretability [9], decision trees have a natural visualization structure that separates data points into distinct groups based on their features and makes the interpretation and explanations quite easy in

[2]For instance, explanations provided in the second model are usually not enough to understand what the black-box model is doing, and these explanations can lead to an overly complicated decision that humans can not troubleshoot in case it is flawed.

comparison to black-box models such as neural networks.

This chapter will review the different approaches used to solve the newsvendor problem, briefly review the risk-averse extension, explain the research gap that inspired this research, and discuss our contributions.

1.1 Literature Review

In this subsection, we review the literature in the area of the newsvendor problem. We start by introducing the problem and early solution approaches, and then discuss more recent ones that involve the use of data. The data-driven solutions are categorized as contextual and non-contextual based on the type of data used. In the non-contextual approaches, the only available data is the historical demand observations whereas in the contextual case, the decision maker has access to features in addition to the historical demand observations. Finally, a brief overview of the related literature in the area of risk-averse newsvendor problems is provided.

1.1.1 Introduction to the Newsvendor Problem

The first appearance of the problem that we know as "The Newsvendor Problem", dates back to Edgeworth [12], when he first used it in the banking context in order to set the level of cash reserves of a bank to cover customers' demand using a normal distribution. However, the term "Newsboy" was first used by Morse and Kimball [13] in an example given in their book [14]. The example was about a newsboy purchasing newspapers to sell before knowing how much the demand will be in a specific day, and from this example the name of the problem was coined. Later, Arrow et al., [15] investigated the problem in more depth, and derived the classical solution that we know as the "critical fractile" approach under the assumption that the demand distribution is known to the decision maker. Based on their solution, the optimal order quantity was calculated using the critical fractile of the demand's distribution [1] where the critical fractile is a ratio calculated using the unit underage and overage costs.

Early papers investigating this problem assume that the demand distribution is either fully known [16, 17] or the true distribution of the demand belongs to a parametric family of distributions, but the specific values of parameters are unknown [2]. In the Bayesian approach, demand is considered to belong to a known family of distributions, which is chosen based on intuition and past experience, up to some unknown parameters. Estimates of these parameters are obtained dynamically using the Bayes' rule as the demand gets updated and new information becomes available [18, 19, 20]. Liyanage and Shanthikumar [21] used *operational statistics* to perform optimization and estimation simultaneously. In this approach, it is assumed that demand belongs to a parametrized family of distribution functions. However, unlike the Bayesian approach, there is no prior knowledge about the unknown parameter values.

Two other well-known approaches are the minimax and maximin. In the Maximin approach, the mean and variance of demand are known, and the goal is to maximize the worst-case profit [22, 23]. On the other hand, there is the Minimax approach in which maximum regret or opportunity cost from choosing a specific demand distribution with partial information is minimized demand distributions within a specified uncertainty set [24, 25].

There are many other similar approaches. However, it is now evident that a known demand distribution is not a realistic assumption. More importantly, assuming known parameters leads to solutions that require two separate steps to determine the optimal order quantity, first the demand distribution should be estimated and then the order quantity can be predicted using the estimations from the first step. This two-step procedure amplifies errors from the first step in the second step and results in solutions that might be far from optimal.

It is possible to bypass the errors from the two-step approaches by allowing the decisions to be made based on covariates in one step. As a result, more recent approaches to solve the newsvendor problem put efforts into developing a single step algorithm. Since these efforts are too numerous to enumerate here, we divide these approaches into two general categories, "non-contextual" and "contextual", depending on how the data is used by the methods. If a solution is only based on

historical demand observations it is regarded as non-contextual, and if it exploits covariate/feature data in addition to historical demand it is classified as contextual. Despite the close similarity of our work to contextual solutions, we will also briefly point out the non-contextual approaches.

It is worth mentioning that regardless of this classification which is created for the ease of discussion, there are different extensions of the newsvendor problem. For instance, additional constraints can be added to the model to consider capacity or budget limitations when ordering more than one product or the objective of the problem can be changed to take into account the risk preferences of the decision-makers in addition to minimizing (maximizing) the expected costs (profit).

1.1.2 Non-contextual Methods

One of the most popular data-driven methods that use historical demand data instead of any prior knowledge about the true distribution is the sample average approximation (SAA) method [26, 27]. In the SAA approach, one has access to independent and identically distributed (i.i.d) samples drawn from the true distribution from which an empirical distribution can be induced. Levi et al. [28] first applied this method to the newsvendor problem where they used a counterpart objective function based on many independent random samples drawn from the probability distribution of the demand either based on historical data or by means of *Monte Carlo* sampling. Levi et al. [28] also established bounds on the number of samples required to guarantee that, with a high probability, the difference of the expected cost of an optimal solution with respect to their counterpart SAA is within a small pre-specified error. Levi et al. [2] then further improved the aforementioned bound.

Burnetas and Smith [29], Kunnumkal and Topaloglu [30], and Huh and Rusmevichientong [31] used stochastic gradient algorithms, and they all incorporated censored demand data (i.e. data that are on sales instead of demand) in their algorithms. Godfrey and Powell [32] and Powell et al. [33] used a sampling method called Concave Adaptive Value Estimation (CAVE) that recursively constructs piecewise linear approximations of the cost function. Bertsimas and Thiele [34] found order quan-

tities based on empirical data and incorporated a scalar parameter in their model that can be adjusted to achieve an appropriate level of protection against demand uncertainty.

1.1.3 Contextual Methods

One of the first papers to consider contextual data in the newsvendor problem is He et al. [35], who used the newsvendor model to determine optimal staffing levels using two features (number of cases and the hours of usage of operating rooms) about operating rooms' workload. They consider the problem of finding optimal staffing levels with different information sets available at the time of making decisions: no information, information on the number of cases, and information on the number and type of cases. Their comparisons showed that hospitals could reduce staffing costs by $39\% - 49\%$ if they postponed the staffing decisions until the procedure information is available.

Although the work of He et al. [35] is a good starting point to incorporate feature data in the newsvendor problem, it only uses a small number of features. Hannah et al. [36] also used what they called the "state variable", which is the same as a feature in definition, and they proposed solving the contextual newsvendor problem by a weighted empirical stochastic optimization approach where they estimate the weights given to past data using the Kernel Optimization (KO) method, group similar observations with some weights, and then determine the optimal decision based on these similar observations. However, they only consider discrete features.

The optimal solution in the newsvendor problem is a quantile of the demand distribution, and it should be chosen in a way that minimizes the underage and overage costs. As a result, a good way to solve this problem is to model it as a quantile regression problem. Ban and Rudin [4] proposed a linear machine learning algorithm that considers both small and big data (where the size of data is decided based on a ratio that is simply the number of features divided by the number of observations), and they prove that their algorithm is equivalent to a high-dimensional quantile regression problem. In this algorithm, they relate the optimal order quantity and

demand features using an affine decision function that is basically a weighted sum of features. To avoid overfitting in their model, a regularization term is also added to the original objective function. Moreover, Ban and Rudin [4] proposed another method based on the kernel regression method introduced by Nadaraya [37] and Watson [38], called Kernel Optimization (KO) that outperformed their machine learning algorithm (which is based on empirical risk minimization) in terms of the out-of-sample accuracy.

Another novel and recent approach in solving the contextual newsvendor problem is proposed by Oroojlooyjadid et al. [3]. They used a deep neural network (DNN) with a revised loss function that takes into account the impact of back-ordering and holding costs and minimizes the cost function directly. Oroojlooyjadid et al. [3] proved that their algorithm's real value to is for problems where there is lack of historical data, problems with volatile historical data, or problems with unknown/unfitted probability distributions. Huber et al. [39] also proposed single-step approaches that are a combination of machine learning algorithms and quantile regression. More specifically, they use artificial neural networks and gradient boosted decision trees which are ensembles of traditional decision trees such as Classification and Regression Trees (CART). They tested their algorithms on a real-world dataset for a large German bakery chain and showed that given sufficiently large amounts of data, their methods outperform traditional two-step solutions. Within their conclusions, they also found that overall performance of two-step approaches depend heavily on the demand estimation method employed, meaning that poor forecasts cannot be compensated for by the choice of the subsequent optimization approach which empirically proves why two-step approaches of demand estimation and optimization are not a reliable choice.

Punia et al. [40] also applied different machine learning algorithms to solve the mutli-product newsvendor problem with a capacity constraint. Their porposed algorithms use quantile regression to predict order quantities in a single step. They demonstrated that not only their models, namely multiple regression, neural networks, and random forest which is another ensemble of traditional decision trees, outperform empirical demand distribution model (where they assume that the de-

mand follows a normal distribution and they use the critical fractile method to predict the order quantities after estimating the demand distribution using the empirical mean and variance of the demand data), but also an accurate order quantity estimation can have a significant impact on the total cost of the newsvendor model. They also conclude that external variables, which is the same as feature data in our work, play an important part in enhancing the performance of their model.

Ban et al. [41] proposed a practice-oriented method called "The Residual Tree" based on the scenario tree method of Heitsch and Römisch [42] for optimizing the order quantity of a new, short-life-cycle product assuming that there are multiple suppliers and some feature information about new, and past similar products. Bertsimas and Kallus [43] applied five different machine learning algorithms, namely random forest(RF), kernel method, classification and regression trees (CART), k-nearest neighbours (KNN), and locally weighted scatter plot smoothing (LOESS), to prescribe optimal decisions in a wide range of decision-making problems including the contextual newsvendor problem. They derived weights from each algorithm and performed a weighted SAA to predict the order quantity in the newsvendor problem. Yu et al. [44] used a support vector machine (SVM) algorithm in order to predict newspaper demand according to available features. In the SVM method, demand observations are clustered, then these clusters are used to predict the demand value in future periods. While this approach has been used many times in the context of inventory management [45, 46], it does not take into account the fact that the predicted demand should be chosen in a way that underage and overage costs are balanced. The work of Bertsimas and Kallus [43] also suffers from a similar pitfall.

1.1.4 Risk-Averse Newsvendor Problem

All the above mentioned literature corresponds to the risk-neutral newsvendor problem where the goal is to minimize (maximize) the expected cost (profit). However, in practice, managers' decisions do not always revolve around cost (profit) minimization (maximization). Schweitzer and Cachon [47] provide evidence suggesting that managers may have different risk preferences depending on product character-

istics. Therefore, risk-averse newsvendor problems have attracted more attention from both academia and industry in the recent years.

There are different situations, such as unpredictable disasters or economic crises, where managers might be particularly concerned about the downside risk, i.e. focusing on the events of losses rather than long term average profits. Common risk assessment measures are Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR). Despite the popularity of VaR, its undesirable mathematical characteristics such as lack of convexity, makes CVaR a better choice when it comes to convex optimization settings [48] such as ours. A large body of literature on the risk-averse newsvendor problem using CVaR uses two-step approaches [49, 50, 51, 52, 53] where the solution is based on estimation of demand distribution. However, the focus of this thesis is on the contextual newsvendor problem where the estimation of the demand distribution is avoided using a one-step approach as explained before. The only work that has a similar approach and uses feature data is proposed by Sawik [54] where the predictive performance of three different models is compared in a risk-averse setting. However, the proposed models are created based on non-contextual data.

1.2 Research Gap

There are numerous approaches to solve the newsvendor problem, as discussed in the literature review section. However, a handful of these solutions use overly simplistic or have unrealistic assumptions regarding the demand distribution. Among the solutions that do not rely on any prior knowledge of the demand distribution and use historical observations instead, only a few of them exploit contextual data. These big data predictive models typically focus only on the performance of the model without taking into account their interpretability.

There is a need for a solution method for the newsvendor problem that is both interpretable and has a reasonable performance. To date, contextual solutions to the newsvendor problem are either too simple and thus do not perform well in the case of non-linearity [4], or too complex to be understood by decision-makers [3]. If

an algorithm can reveal how various features contribute (or do not contribute) to final predictions, the model can be validated and interpreted more easily, and biases that were not obvious during the training can be uncovered. This motivated us to propose an inherently interpretable algorithm to solve the newsvendor problem. Our proposed approach relies directly on data to avoid unnecessary assumptions about the demand distribution, and to take advantage of the proven value of contextual data [4].

For the purpose of interpretability, we will use a decision tree structure. In traditional decision tree algorithms, the greedy top-down approach will result in trees that are only locally optimal and do not capture well the characteristics of the underlying dataset and have poor performance when working with previously unseen data points [55]. Our goal is to find globally optimal decision trees. To identify the optimal decision tree, one should enumerate every possible decision tree to come up with the optimal one within a reasonable time, which sounds and is inefficient. Therefore, the main challenge here is to create optimal regression trees that can predict the optimal order quantity of the newsvendor problem in a single step and in a reasonable amount of time.

Moreover, we are also interested in the risk-averse newsvendor problem since the classical newsvendor problem ignores the risk preference of decision-makers. A recent and common choice for measuring risk is CVaR. Although there have been recent newsvendor models that incorporate CVaR [52, 53, 51], there seems to be a gap in the literature when it comes to applying CVaR to the case of contextual newsvendor problem.

1.2.1 Research Questions

To bridge the gap mentioned in the previous subsection, we need to answer the following questions:

1. How can one solve the contextual newsvendor model to optimality using a tree-based structure that is designed in a single step?

2. How can we extend the proposed single-product, risk-neutral newsvendor algorithm to the risk-averse and multi-product newsvendor?

1.2.2 Contributions

We summarize our contributions below:

- Interpretability: Decision trees provide inherently interpretable solutions. The trees produced by our approach can provide intuition on the important features that lead to a decision policy being assigned to an observation, and how the final order quantity is calculated. Our approach provides higher predictive accuracy in comparison to simple regression and better interpretability than complex machine learning models.

- Optimality: In heuristic decision trees such as CART, first a locally optimal tree is made by a greedy top-down approach and then this tree is pruned in a separate step. However, our algorithm provides a globally optimal pruned decision tree in a single step.

- Multi-Product case: We have extended our tree-based algorithm to the multi-product newsvendor problem as well. This extension is particularly important when a capacity constraint is added to the model.

- Risk-Averse Newsvendor: In addition to the risk-neutral approach, we investigate the risk-averse newsvendor problem with contextual data using the Conditional Value at Risk (CVaR) assessment measure. To the best of our knowledge, the risk-averse newsvendor problem with the contextual data has not been broadly investigated.

1.3 Overview and Structure of the Thesis

A chapter by chapter description of the remainder of this thesis is as follows:

Chapter 2: Provides an overview of decision tree algorithms, their main advantages and shortcomings, a description of the CART algorithm, and the motivation and idea behind Optimal Classification Trees.

Chapter 3: Discusses the solution for the classical newsvendor problem, Ban and Rudin's Big Data Newsvendor model, and proposes an interpretable newsvendor model for a single product along with other previously mentioned extensions of the problem.

Chapter 4: Includes a numerical experiment based on real world data on historical demand observations and seasonality features.

Chapter 5: Presents results, analyses, and conclusions obtained from the numerical experiment along with the future research directions.

Chapter 2

Overview of Decision Tree Methods

Decision trees are one of the widely-used supervised machine learning methods for creating interpretable predictive models. The main reason behind this method's popularity is that it closely resembles human reasoning and therefore is easy to understand compared to black-box models such as neural networks. In other words, it is much simpler to follow and interpret some logical rules that end up with a decision than the numeric weights and connections in a neural network.

Depending on the type of target variable being predicted, decision trees can be categorized into classification and regression trees. Classification trees are applied when the target variable is categorical or discrete in nature whereas regression trees are used when the outcome variable is continuous. Regardless of the nature of target variable, the goal of a decision tree is to divide a dataset into smaller segments based on a series of simple tests and use these segments for future predictions. Based on training data $(X_i, y_i),\ i = 1, \ldots, n$, with p features $X_i \in \mathbb{R}^p$, each test splits the data points by comparing a numeric feature against a threshold value or a nominal feature against a set of possible values [56]. It is important to mention that finding the optimum partitions in a decision tree is typically done by heuristics that follow a greedy search; meaning that the decision tree searches through the data points in a given node and calculates the gain from all possible tests in order to pick the

split with the greatest gain that results in the most homogeneous sub-nodes. This procedure is then applied recursively until a predefined stopping criteria is met and a decision tree is formed. The resulting decision tree is a fully-grown possibly deep tree with a high predictive accuracy on the data that has been trained with. However, the main goal of a predictive model is its generalizability, and deep decision trees tend to perform poorly when it comes to unseen data. Not surprisingly then, there have been many efforts over the years to create trees of smaller sizes and better generalizability. Some examples of such decision tree algorithms are C4.5 [57], ID3 [58], and CART [59].

In the following section, we give an overview of the CART algorithm and some of its limitations that motivated Bertsimas and Dunn [55] to come up with the Optimal Classification Trees.

2.1 Overview of Classification and Regression Trees (CART)

Classification and Regression Trees, or CART for short, is a term introduced by Leo Breiman [59] to refer to decision tree algorithms that can be used for both classification and regression predictive modeling problems. The representation of the CART model is a binary tree that is created using a top-down approach to partition the feature space. An example of a decision tree with depth two is shown in the Figure 2.1. Decision tree nodes are categorized as either a branch node or a leaf node. Nodes that are shown with rectangles in Figure 2.1, and have sub-nodes, are called branch nodes, where the starting branch node of a decision tree is called the root node, and nodes that are shown with circles and are not further split are called leaves of the tree. Similar to any other decision tree algorithm, the main elements of the CART are:

- A cost/loss function based on which splits are chosen at each branch node.
- Stopping criteria for deciding when a branch is terminal and can be split no more.

- A prediction for the target variable in each leaf node.

Starting from the root node, each split in a decision tree is determined by solving an optimization problem. At each branch node, a split is applied by determining the values of a and b where a is a vector of binary variables that shows which feature is chosen to split on and b is a continuous variable corresponding to the splitting threshold. Since CART produces univariate decision trees, a single component of a will be 1 and all others will be 0. For each data point i, $a^T x_i$ shows the value of the chosen feature for that point. If $a^T x_i < b$, the point follows the left branch from that node; otherwise, it will follow the right branch. Determining the best split in each node in CART (finding the optimal values for a and b) is done based on a loss function which is chosen according to the classification/regression nature of the problem. Optimizing the loss function in CART is essentially equivalent to choosing the best split at each branch node such that the points in each node are divided into two most homogeneous sub-nodes. Common choices of loss functions are the *gini index* for classification, and *mean squared error*, *mean absolute error*, or *quantile loss* for regression problems. It is worth mentioning that there are also some extra terms that can be added to the cost function to improve the performance of the tree and avoid overfitting.

The recursive splitting continues until one of the several predetermined stopping criteria (such as the minimum number of nodes, the maximum desired depth of the tree, etc.) is met. When the partitioning is done, each leaf node is assigned a label/numeric value that is used for future predictions. This label/numeric value is determined based on the classification/regression nature of the tree:

- Classification: In classification trees, a label is assigned to each leaf based on the most common label among the labels of the training points that fall into that leaf.

- Regression: In regression trees, a fixed numeric value is assigned to each leaf which is determined using the mean of the target values of the training points that fall into that leaf.

As mentioned above, the stopping criterion is important as it strongly influences the performance of the decision tree. However, a primary concern when using decision trees is to avoid overfitting the tree to the training points. It is possible to reach very high accuracy on the training points with a deep tree, but such a tree most likely will perform poorly when making predictions on unseen data. This overfitting issue is controlled using the trade-off between the training accuracy and the *complexity* of the decision tree, which is typically calculated as the number of splits. CART uses a complexity parameter, α, in order to control the trade-off between accuracy and complexity. After the tree is grown, at each split, CART considers replacing the split and its sub-nodes with a single leaf node and calculates the change in the predictive accuracy resulting from this change. If the change is lower than α, the corresponding split is replaced with a leaf. This process continues until all splits in the tree provide an improvement of at least α in their predictive accuracy [55]. This procedure is known as *cost-complexity pruning* and, as mentioned before, is done after the tree is grown.

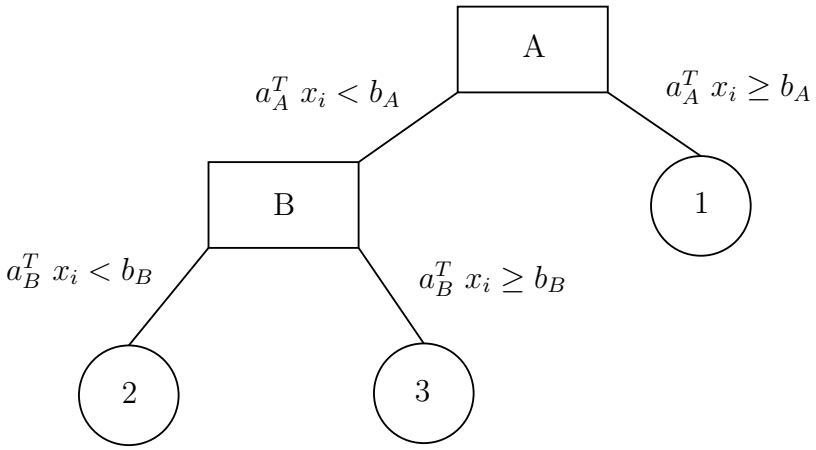

Figure 2.1: A Decision Tree with Depth 2

Despite their interpretability and ease of use, decision trees have some limitations because of their greedy nature. The main shortcoming of the CART and other popular decision tree algorithms is that the top-down approach selects each split of the tree in isolation and without considering its impact on the future splits [55]. This might result in poor performance when using unseen points for prediction after training the tree. As a result, decision trees algorithms usually require a second

phase after training to prune the tree. The pruning phase is separated from the training because the top-down approach is unable to handle complexity penalties while growing the tree since it might lead to powerful splits being hidden behind weaker splits [55]; meaning that if the complexity penalty is too high, it might prevent the first weaker splits from being chosen. To resolve this, the most common approach is to grow the tree as deep as possible and prune it using the complexity penalty afterwards.

As mentioned before, an important limitation of CART is that the top-down approach selects each split separately based on optimizing a loss function which makes the tree only locally optimal. This issue has also been acknowledged by Breiman et al. [59] when introducing the algorithm:

> *Finally, another problem frequently mentioned (by others, not by us) is that the tree procedure is only one-step optimal and not overall optimal ... If one could search all possible partitions ... the two results might be quite different ... At this stage of computer technology, an overall optimal tree growing procedure does not appear feasible for any reasonably sized dataset.*

According to Bertsimas and Dunn [55], this acknowledgement means that the two-step approach of creating and pruning taken by CART was largely due to practical limitations related to the execution time, and not because the procedure was thought to be inherently better. Therefore, one question remains unanswered. Can one develop an optimal decision tree algorithm in a single step?

2.2 Optimal Classification Trees (OCTs)

To address the limitations of CART and other popular decision tree algorithms, Bertsimas and Dunn [55] proposed a decision tree algorithm called "Optimal Classification Trees". What distinguishes their formulation from previously used decision tree algorithms is that they create the decision tree in one step instead of recursively creating a tree and then pruning it. However, as the name suggests, the algorithm

is designed only for classification problems. Bertsimas and Dunn [55] claim that the two-step procedure of first creating and then pruning the tree might result in solutions that are only locally optimal, and that the lack of success in developing the optimal decision tree algorithm is largely due to failure to address the underlying nature of decision trees. They argue that the natural way to create the whole tree in one step and achieve global optimality is through Mixed-Integer Optimization (MIO) since at each node of a decision tree several binary decisions must be made such as either branch or stop, what feature to branch on, etc. They argue that using an MIO approach to create a decision tree allows to consider all these decisions in a single problem with the knowledge of the whole tree , and it is practical because of the speed-up of MIO solvers in the last few decades.

It is worth mentioning that using ensemble algorithms for decision trees such as random forests results in an improvement over a single decision tree in terms of generalizability. The idea behind random forests is to create many decision trees that are each trained using a random subset of data and features, and use the mean prediction in cases of regression and the majority vote in classification problems. However, the increased predictive accuracy in random forests and other ensembles of decision trees (such as boosted trees) comes at the cost of losing interpretability. That is why Bertsimas and Dunn [55] created an algorithm for a single decision tree using MIO so one could achieve optimality while preserving the interpretable nature of decision trees.

Bertsimas and Dunn [55] then provided evidence that their formulation leads to trees that better capture the data in comparison to CART. They show that optimal classification trees are likely to outperform CART by 1.2–1.3% in situations where the CART accuracy is high and we have sufficient training data, while this improvement can increase by 4–7% when the CART accuracy or dimension of the dataset is low. For the performance comparison purposes, they used 53 data sets from the UCI machine learning repository [60]. This motivated us to believe that such an improvement is also possible for regression problems, and that we can create optimal decisions tree to predict the order quantity in the newsvendor problem.

Chapter 3

Methodology

3.1 Classical Newsvendor Problem

As mentioned before, the objective in the newsvendor problem is to choose an order quantity for a perishable product before observing the actual demand. When sales are repetitive, a sensible goal is to order a quantity that minimizes the total expected cost [4] according to

$$\min_{q \geq 0} EC(q) := \mathbb{E}[C(q; D)], \tag{3.1}$$

where q is the order quantity, D is the unknown future demand, and the cost associated with order q and demand D with respect to unit backordering and holding costs, denoted by b and h respectively, is as follows:

$$C(q; D) := b(D - q)^+ + h(q - D)^+ \tag{3.2}$$

where the superscript plus sign enforces the value in parenthesis to be non-negative. If one follows the assumption of a known demand distribution, the optimal decision can be calculated using the $\frac{b}{b+h}$ quantile. However, in practice, it is unlikely that that the decision-makers know the demand distribution. More importantly, various features (that are known before making a decision) affect the demand. A recent paper that takes into account the availability of feature information without depending on the demand distribution is the big data newsvendor model proposed by Ban and Rudin [4], which is explained in more depth in the following section.

3.2 Ban and Rudin's Big Data Newsvendor Model

Consider a newsvendor who has access to n past demand observations as well as p features where $S_n = \{(X_i, D_i)\}_{i=1}^n$ represents the historical data and $X_i = [x_i^1, \ldots, x_i^p]$ the features about the demand. The goal of Ban and Rudin's big data newsvendor model is to determine the order quantity of period $n+1$ after observing features X_{n+1} without knowing the actual demand.

To achieve this goal, Ban and Rudin [4] proposed a machine learning algorithm where they find the optimal function $q(\cdot)$ that maps the observed features $X_{n+1} \in \mathcal{X}$, with $\mathcal{X} \subset \mathbb{R}^p$, to an order $q(X_{n+1}) \in \mathbb{R}$ with respect to unit backordering and holding costs. Their formulation offers preliminary insights and allows decision makers to come up with a tractable decision policy which is contextual. The Big Data approach to solving the contextual newsvendor problem is

$$\min_{q(\cdot)\in\mathcal{Q}} \hat{R}(q(\cdot); S_n) = \frac{1}{n}\sum_{i=1}^{n}[b(D_i - q(X_i))^+ + h(q(X_i) - D_i)^+], \tag{3.3}$$

where $\mathcal{Q} \subseteq \{q\colon \mathcal{X} \to \mathbb{R}\}$, $\hat{R}$ is called the *empirical risk* of function q with respect to dataset S_n and the $\hat{}$ notation is used to emphasize that quantities are estimated from data. *Empirical risk minimization* is a principle that is widely used for classification and regression problems (refer to [5] for a complete discussion). To solve this problem, Ban and Rudin [4] defined the function class $\mathcal{Q}$ of the form

$$\mathcal{Q} = \{q\colon \mathcal{X} \to \mathbb{R} |\ \exists q^j,\ q(X) = \sum_{j=1}^{p} q^j x^j\}, \tag{3.4}$$

with $x^1 = 1$ to allow for an intercept term (feature-independent term). Ban and Rudin [4] explained that their choice of function class $\mathcal{Q}$ is not restrictive since one can easily consider nonlinear transformations of basic features and expand the feature space to accommodate nonlinear dependencies as well. In other words, nonlinear decisions can be transformed into linear decisions with the addition of new features. However, they mention that if this transformation results in $p \gg n$, the solution of model (3.4) does not have a good out-of-sample performance. Equation

(3.3) is equivalent to the following formulation

$$\begin{aligned}
\min_{q=[q^1,\ldots,q^p]} \quad & \frac{1}{n}\sum_{i=1}^{n}(bu_i + ho_i) \\
\text{s.t.} \quad & \\
& u_i \geq D_i - q^1 - \sum_{j=2}^{p} q^j x_i^j,\ i = 1,\ldots,n \\
& o_i \geq q^1 + \sum_{j=2}^{p} q^j x_i^j - D_i,\ i = 1,\ldots,n \\
& u_i, o_i \geq 0,
\end{aligned} \tag{3.5}$$

where u_i and o_i are dummy variables that account for the underage and overage costs of data point i. Furthermore, in the case of high dimensional data or when p is relatively large, Ban and Rudin [4] suggest the following model where a regularization term is added to the objective to avoid overfitting

$$\begin{aligned}
\min_{q=[q^1,\ldots,q^p]} \quad & \frac{1}{n}\sum_{i=1}^{n}(bu_i + ho_i) + \lambda\|q\|_k \\
\text{s.t.} \quad & \\
& u_i \geq D_i - q^1 - \sum_{j=2}^{p} q^j x_i^j,\ i = 1,\ldots,n \\
& o_i \geq q^1 + \sum_{j=2}^{p} q^j x_i^j - D_i,\ i = 1,\ldots,n \\
& u_i, o_i \geq 0
\end{aligned} \tag{3.6}$$

where $\lambda > 0$ is the regularization parameter and k refers to the type of norm that is used. Although this model proves to outperform previous models such as the SAA, it is still restrictive when working with nonlinear data. To solve this limitation, the authors propose using predefined breaking points enlarging the feature space to allow for piecewise linear decisions. However, there are two important issues when using this method. First, the choice of breaking points, as the authors mention [4], is arbitrary since it is determined before solving the model, and second, in order to incorporate these breaking points, one should increase the number of features by creating new ones which add to the complexity of the model. We believe this issue can be solved if we could determine the optimal breaking points where the model is solved and thus create a piecewise affine function that is optimal.

It is worth mentioning that both this model and the interpretable newsvendor model proposed in the next section are not limited to the inventory management context. For example, Ban and Rudin [4] case-studied a nurse staffing problem at a UK teaching hospital's emergency room, assuming a mandatory nurse-to-patient ratio (with daily data for a year). This setting is similar to the newsvendor problem since the hospital would incur an underage cost if due to the high number of patients, expensive emergency nurses have to be called, and an overage cost if there are too many regular nurses in comparison to the number of patients. They also knew that the hourly wage of an emergency nurse is 2.5 times that of a regular nurse, and based on this, they considered $b = \frac{2.5}{3.5}$ and $h = \frac{1}{3.5}$, which are the unit backordering and holding cost, respectively. They consider the day of the week, time of the day, and the number of days as the features in this case study.

3.3 The Interpretable Newsvendor Model

3.3.1 Single-Product Model

Consider the same data structure as mentioned in the previous section. A newsvendor has access to past demand data along with some features for a single product and our goal is to choose the order quantity of the next period after observing features but without knowing the actual demand using an interpretable decision tree structure. In order to formulate the interpretable newsvendor problem, one can divide the model's variables and constraints into two groups, one that corresponds to the decision tree structure and another that corresponds to the newsvendor context of the model. In this section, I will begin by introducing the notation, variables, and constraints pertaining to the first group. After that, the second group's definitions and constraints are discussed and finally, the objective function is introduced.

Without loss of generality, throughout this thesis, we assume that both the values for features (input) and past demand (target variable) are normalized, meaning that each $x_i^p, D_i \in [0, 1], i = 1, \ldots, n$.

Decision Tree Structure

Consider trying to construct a decision tree with a depth of D_T. Given this depth, we can construct a fully-grown tree with $T = 2^{(D_T+1)} - 1$ nodes, indexed by $t \in [T]$ as shown in Figure 3.1. It is important to mention that here the decision tree structure as a whole is analogous to the work of Bertsimas and Dun [55], with some modifications since we are trying to build optimal regression trees instead of optimal classification trees.

We will use $p(t)$ and $\mathcal{A}(t)$ to denote the parent node and set of ancestors (all the nodes on the path from the root node to node t) of node t, respectively. Moreover, similar to Bertsimas and Dunn [55], we let $\mathcal{L}(t)$ and $\mathcal{R}(t)$ to respectively denote the set of ancestors of node t whose left and right branches have been followed on the path from the root node to node t. $\mathcal{A}(t) = \mathcal{L}(t) \cup \mathcal{R}(t)$. For instance, using the above notation for node 6 in Figure 3.1, $\mathcal{L}(6) = \{3\}$, $\mathcal{R}(6) = \{1\}$, $\mathcal{A}(6) = \{1, 3\}$,

and $p(6) = 3$.

Similar to other decision tree structures such as CART (discussed in Chapter 2), we divide nodes of the tree into two sets of branch nodes and leaf nodes and denote them by $\mathcal{T}_B$ and $\mathcal{T}_L$ respectively. Branch nodes, $t \in \mathcal{T}_B = \{1, \ldots, \lfloor \frac{T}{2} \rfloor\}$, are the nodes that apply a split, and leaf nodes, $t \in \mathcal{T}_L = \{\lfloor \frac{T}{2} \rfloor + 1, \ldots, T\}$, are the nodes that make a prediction for points that fall into them.

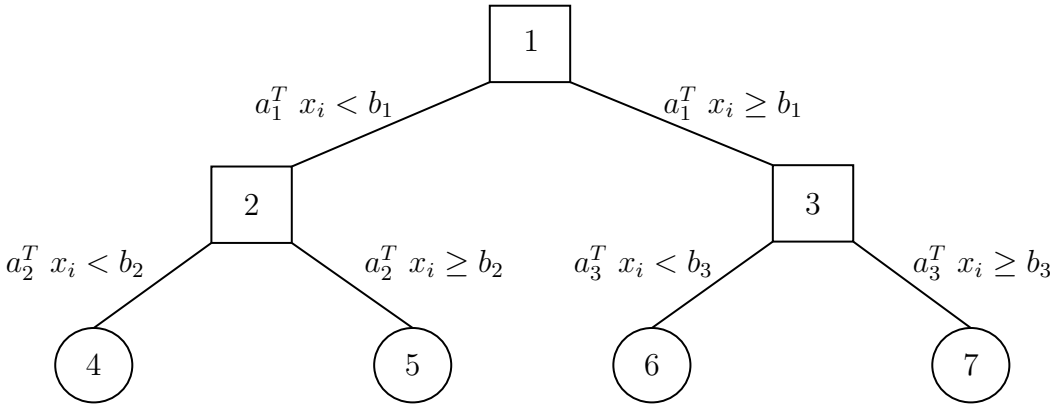

Figure 3.1: A Fully-Grown Decision Tree with Depth 2

In order to track the splits applied at branch node $t \in \mathcal{T}_B$, we use the indicator variables $a_{jt} \in \{0, 1\}$, where $j = 1, 2, \ldots, p$, and $b_t \in \mathbb{R}$. Similarly, we use the indicator variable $d_t \in \{0, 1\}$ to track which branch nodes apply a split, where $d_t = 1$ if node t applies a split (it is important to mention that the model can be adjusted to account for multivariate trees. For an in depth discussion, refer to [55]). Within this thesis, we restrict the model to univariate decision trees, meaning that the hyperplane in each branch node only involves one variable. This can be enforced by making $\sum_j a_{jt}$ to be equal to 1 in each branch node, if a split is applied in that node (In other words, the sum of each column of a_{jt} is equal to 1 if and only if $d_t = 1$). Also, the option of not splitting at a branch node is allowed. If a branch node does not apply a split, both $d_t = 0$ and $a_{jt} = 0$ for that node, and all points will follow the right branch, since the condition for left branch node is $0 < 0$, which is never satisfied (the logic is straightforward based on the decision tree in Figure 3.1). The following two constraints summarize the above explanations and correspond to

the univariate branching nature of the tree.

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in \mathcal{T}_B, \tag{3.7}$$

$$1 - d_t \leq b_t \leq 1, \ \forall t \in \mathcal{T}_B \tag{3.8}$$

We need the splitting point b_t to be in the same range as the data, and since all feature data is normalized, the second constraint is valid.

The next constraint enforces the hierarchical structure of the tree, meaning that a branch node is prevented from splitting if its parent node does not apply a split. As mentioned above, this is because if a node does not apply a split, all the points in that node must end up together in the same leaf and the node must be prevented them from further splitting.

$$d_t \leq d_{p(t)}, \ \forall t \in \mathcal{T}_B \setminus \{1\} \tag{3.9}$$

Obviously, such a constraint is not required for the root node. In order to track the allocation of data points to the leaves after branching, we need to introduce the indicator variable z_{it} and l_t for $t \in \mathcal{T}_L$. The binary variable $l_t = 1$ if leaf t contains any points, and z_{it} is introduced to track the assignment of points to the leaves, where $z_{it} = 1$ if data point i is assigned leaf node t, and $z_{it} = 0$ otherwise.

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \ \forall i \in [n] \tag{3.10}$$

$$z_{it} \leq l_t, \ \forall t \in \mathcal{T}_L \tag{3.11}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in \mathcal{T}_L \tag{3.12}$$

Constraint (3.10) enforces each point to be assigned to one leaf. Constraints (3.11) allows a point to be assigned to a leaf only if the leaf exists, and if a leaf does not contain any points (does not exist), then no points should be assigned to it, and Constraint (3.12) enforces a minimum number of points, given by N_{min}, at each leaf node to prevent overgrowing the tree and creating extra leaf nodes that do not include at least N_{min} number of points.

Lastly, we need to apply constraints enforcing the splits in the structure of the tree to be able to track the path that each point follows from the root node to its assigned

leaf. Bertsimas and Dunn [55], introduce the following constraints for this purpose:

$$a_m^T x_i < b_m + M_1(1 - z_{it}),\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \forall m \in \mathcal{L}(t) \tag{3.13}$$

$$a_m^T x_i \geq b_m - M_2(1 - z_{it}),\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \forall m \in \mathcal{R}(t) \tag{3.14}$$

Note that constraint (3.13) is a strict inequality which is not supported by MIP solvers. As a result, the authors added a small constant ϵ to the left-hand side of equation (3.13) to make the inequality non-strict.

$$a_m^T x_i + \epsilon \leq b_m + M_1(1 - z_{it}),\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \forall m \in \mathcal{L}(t) \tag{3.15}$$

To avoid instabilities in the MIP (Where the numerical solution diverges from the actual solution), ϵ should be as big as possible without affecting the feasibility of valid solutions to the problem. Consequently, the authors propose using different ϵ_j for each feature j as follows:

$$\epsilon_j = min\{x_j^{(i+1)} - x_j^{(i)} | x_j^{(i+1)} \neq x_j^{(i)},\ i = 1, \ldots, n-1, \} \tag{3.16}$$

where values of each feature are first sorted in an ascending order to do the above calculation. Then we can use the above calculated values to replace ϵ in Constraint (3.15), where the value of ϵ_j is selected according to the feature that is used in that split. We also need to choose values for the big-M constants M_1 and M_2. As mentioned before, both $a_t^T x_i \in [0, 1]$ and $b_t \in [0, 1]$ so the largest possible value of $a_t^T(x_i + \epsilon) - b_t$, is $1 + \epsilon_{max}$, where $\epsilon_{max} = max_j\{\epsilon_j\}$. Therefore, we can set $M_1 = 1 + \epsilon_{max}$. Similarly, the largest possible value of $b_t - a_t^T x_i$ is 1, so we set $M_2 = 1$. As a result, we can rewrite constraints (3.13) and (3.14) as

$$a_m^T(x_i + \epsilon) \leq b_m + (1 + \epsilon_{max})(1 - z_{it}),\ \forall i \in [n],\ \forall t \in \mathcal{T}_L,\ \forall m \in \mathcal{L}(t) \tag{3.17}$$

$$a_m^T x_i \geq b_m - (1 - z_{it}),\ \forall i \in [n],\ \forall t \in \mathcal{T}_L,\ \forall m \in \mathcal{R}(t) \tag{3.18}$$

Before discussing the remainder of the model, it is worth mentioning that Bertsimas and Dunn [55] formulated Equation (3.8) in their model as

$$0 \leq b_t \leq d_t,\ \forall t \in \mathcal{T}_B \tag{3.19}$$

This design choice was intended by the authors to force all training observations down the right branch by making the left split direction constraint (3.17) infeasible

when a branch node does not apply a split. In other words, if a branch node is considered to be inactive ($d_t = 0$), then its associated split parameters a_{jt} and b_t are set to the zero vector and zero, respectively. When a_{jt} and b_t are zero in a branch node, both (3.17) and (3.18) are feasible which is in contrast with the authors' intention. This issue was also noticed in [61], and an alternate equation was proposed there as a replacement which is the one used in our model (3.8) that allows b_t to be equal to one when a branch node does not apply a split (as opposed to $b_t = 0$ in a similar setting in Bertsimas and Dunn's formulation [55]). This makes Constraint (3.18) infeasible when a and b are both zero and forces all observations to follow the left branch.

Newsvendor Context

The remaining constraints in our model address the optimal ordering policy corresponding to the newsvendor context. We assume that each leaf in the final decision tree corresponds to an ordering policy, which is linear in the features. This ordering policy assigns a weight to each feature and these weights are used to calculate the order quantities of the points that fall into that leaf based on their feature values. Similar to the function that Ban and Rudin [4] defined, we define a function for each leaf that corresponds to the optimal ordering policy within that leaf:

$$q_t(X) = q_t^T X = \sum_{j=1}^{p} q_t^j x^j, \ t \in \mathcal{T}_L \tag{3.20}$$

Furthermore, we will modify the variables corresponding to the underage and overage costs introduced by Ban and Rudin [4] to be able to track the underage and overage costs in each leaf separately. Thus,

$$u_{it} \geq D_i - q_t^1 - \sum_{j=2}^{p} q_t^j x_i^j, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L \tag{3.21}$$

$$o_{it} \geq q_t^1 + \sum_{j=2}^{p} q_t^j x_i^j - D_i, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L \tag{3.22}$$

As mentioned previously, the overall goal in the newsvendor problem is to choose an order quantity that minimizes the subsequent costs. Therefore, we need a way to keep track of the total cost associated with each observation based on the ordering

policy it is assigned to. We will add the following big M constraint to force the actual cost to be calculated using the policy corresponding to the leaf that each observation falls into.

$$c_{it} \leq \tilde{c}_i + (1 - z_{it})M, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L, \tag{3.23}$$

where $c_{it} \in \mathbb{R}$ is the cost incurred if the t-th policy is assigned to the i-th observation, and $\tilde{c}_i \in \mathbb{R}$ is the actual cost of i-th observation. In other words, we are only concerned about the cost c_{it} if i-th data point is in the t-th leaf. The following constraint is also added to impose that the total cost, sum of backordering and holding costs should be less than or equal to the cost incurred if i-th data point is using t-th policy:

$$bu_{it} + ho_{it} \leq c_{it}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L \tag{3.24}$$

Objective Function

We want the objective function to be defined in a way that minimizes the cost for the newsvendor problem and at the same time controls the trade-off between the accuracy and complexity of the decision tree structure. Consequently, it consists of two parts. The first part corresponds to the goal of the newsvendor problem which is minimizing cost, so we add the following objective that is similar to the empirical risk objective in the big data newsvendor model:

$$\min \left(\frac{1}{n} \sum_{i=1}^{n} \tilde{c}_i \right) \tag{3.25}$$

Second, we need to control the complexity of the decision tree created to avoid overfitting by creating an unnecessarily deep tree. We will use complexity parameter α to control the complexity of the tree, and update the previously mentioned objective function (3.25) as follows:

$$\min \left(\frac{1}{n} \sum_{i=1}^{n} \tilde{c}_i + \alpha \sum_{t \in \mathcal{T}_B} d_t \right) \tag{3.26}$$

where the complexity of the decision tree is calculated as the number of splits. Aside from overfitting caused by the depth of the tree, overfitting might also happen

because of the choice of policy parameters. One of the important criteria for a model is its generalizability, meaning that the model should be able to perform accurately on unseen data. For example, if there is a large degree of freedom in the choice of the in-sample decision policies, then it may be possible to have unrealistically small or even zero in-sample cost, leading the decision maker to believe perfect ordering is possible [4]. However, having low in-sample cost and fitting the training data perfectly does not guarantee a reasonable out-of-sample performance on unseen data. Using a regularization parameter $\lambda > 0$, we can control the degree of freedom (which we define as the effective number of parameters) in choosing the policy parameters which in turn can help avoid overfitting. We will once again update the objective function as follows:

$$\min \left(\frac{1}{n} \sum_{i=1}^{n} \tilde{c}_i + \alpha \sum_{t \in \mathcal{T}_B} d_t + \lambda \|q\| \right) \tag{3.27}$$

Where $\|q\|$ is the L_1 norm of the policy parameters' matrix.

The following is the complete model for a single-product tree-based newsvendor problem.

$$\min \quad \left(\frac{1}{n} \sum_{i=1}^{n} \tilde{c}_i + \alpha \sum_{t \in \mathcal{T}_B} d_t + \lambda \|q\| \right) \tag{3.28}$$

s.t.

$$bu_{it} + ho_{it} \leq c_{it}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$u_{it} \geq D_i - q_t^1 - \sum_{j=2}^{p} q_t^j x_i^j, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$o_{it} \geq q_t^1 + \sum_{j=2}^{p} q_t^j x_i^j - D_i, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$c_{it} \leq \tilde{c}_i + (1 - z_{it})M, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$a_m^T x_i \geq b_m - (1 - z_{it}), \ \forall i \in [n], \forall t \in \mathcal{T}_L, \forall m \in \mathcal{R}(t),$$

$$a_m^T (x_i + \epsilon) \leq b_m + (1 + \epsilon_{max})(1 - z_{it}),$$

$$\forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall m \in \mathcal{L}(t),$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \ \forall i \in [n],$$

$$z_{it} \leq l_t, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in \mathcal{T}_L,$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in \mathcal{T}_B,$$

$$1 - d_t \leq b_t \leq 1, \ \forall t \in \mathcal{T}_B,$$

$$d_t \leq d_{p(t)}, \ \forall t \in \mathcal{T}_B \setminus \{1\},$$

$$z_{it}, l_t \in \{0, 1\}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$a_{jt}, d_t \in \{0, 1\}, \ j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B,$$

$$u_{it} \geq 0, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$o_{it} \geq 0, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L.$$

3.3.2 Multi-Product Model with A Capacity Constraint

Our model can also be extended to the multi-product newsvendor problem where a decision needs to be made for more than one product. In this case, we assume that there are n historical demand observations for m products, and for each demand observation there are p features. The decision matrix then becomes $q = (q_{ptk}), t \in \mathcal{T}_L$, in order to be able to consider different policies for different products in each leaf of the tree. Moreover, products might have distinct underage and overage costs that can be shown as b_k and h_k respectively as opposed to b and h values in the case of single product. Then the model can easily be adjusted according to these new variables and values.

It is important to mention that the multi-product case can be divided into different single-product problems if there are no additional constraints, and the products are independent. However, for some extensions of the problem, it is not possible to do so. One example of these extensions is the newsboy problem with budget/capacity constraint, also known as the capacitated newsvendor problem. Often times it happens that in practice companies have a limited budget for procurement or limited space in the inventory or other similar capacity limitations. These kinds of restrictions can be implemented in the model by adding a capacity constraint which has

been used before in many newsvendor settings. The overall form [62, 63] of this kind of constraint is as the following:

$$\sum_{k=1}^{m} r_k q_k \leq R \tag{3.29}$$

where r_k is the resource consumption of the k-th product, R is the overall budget/capacity, and q_k is the order quantity of the k-th product. We assume in our problem that there is a capacity constraint such as space in the inventory and R is the maximum number of products that the newsvendor can order. As a result, we can rewrite this constraint as $\sum_{k=1}^{m} q_k \leq R$. Similar to what was mentioned before, this constraint can be easily added to the case of the multi-product newsvendor problem. However, in order to simplify the model and avoid nonlinear constraints, we consider the capacitated newsvendor problem where the predictions in each leaf are only a constant value instead of a set of policy parameters corresponding to each feature. We can incorporate this either by putting policy parameters corresponding to features equal to zero or change the size of the defined variable so that in each leaf there would be only one variable. We chose the second alternative and defined q with the size $[tm], t \in \mathcal{T}_L$, where m shows the number of products. Similarly, in order to solve the multi-product newsvendor problem with capacity constraint, some of the variables such as underage and overage costs should change to 3 dimensional variables in order to incorporate different products. The following is the complete model for multi-product newsvendor with capacity constraint.

$$\min \quad \frac{1}{n} \sum_{k=1}^{m} \sum_{i=1}^{n} \tilde{c}_{ik} + \alpha \sum_{t \in \mathcal{T}_B} d_t + \lambda \|q\| \tag{3.30}$$

s.t.

$$b_k u_{itk} + h_k o_{itk} \leq c_{itk}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall k \in [m],$$

$$u_{itk} \geq D_{ik} - q^1_{tk}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall k \in [m],$$

$$o_{itk} \geq q^1_{tk} - D_{ik}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall k \in [m],$$

$$c_{itk} \leq \tilde{c}_{ik} + (1 - z_{it})M, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall k \in [m],$$

$$\sum_{k=1}^{m} q_{tk} \leq R, \ \forall t \in \mathcal{T}_L,$$

$$a_m^T x_i \geq b_m - (1 - z_{it}),\ \forall i \in [n], \forall t \in \mathcal{T}_L, \forall m \in \mathcal{R}(t),$$

$$a_m^T (x_i + \epsilon) \leq b_m + (1+\epsilon_{max})(1 - z_{it}),$$

$$\forall i \in [n],\ \forall t \in \mathcal{T}_L,\ \forall m \in \mathcal{L}(t),$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1,\ \forall i \in [n],$$

$$z_{it} \leq l_t,\ \forall i \in [n],\ \forall t \in \mathcal{T}_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t,\ \forall t \in \mathcal{T}_L,$$

$$\sum_{j=1}^{p} a_{jt} = d_t,\ \forall t \in \mathcal{T}_B,$$

$$1 - d_t \leq b_t \leq 1,\ \forall t \in \mathcal{T}_B,$$

$$d_t \leq d_{p(t)},\ \forall t \in \mathcal{T}_B \setminus \{1\},$$

$$z_{it}, l_t \in \{0, 1\},\ \forall i \in [n],\ \forall t \in \mathcal{T}_L,$$

$$a_{jt}, d_t \in \{0, 1\},\ j = 1, \ldots, p,\ \forall t \in \mathcal{T}_B,$$

$$u_{it} \geq 0,\ \forall i \in [n],\ \forall t \in \mathcal{T}_L,$$

$$o_{it} \geq 0,\ \forall i \in [n],\ \forall t \in \mathcal{T}_L.$$

3.3.3 Risk-Averse Newsvendor Problem

The methodology that we have discussed so far is based on the assumption of risk neutrality, meaning that managers and decision-makers simply order such that the expected costs (profits) are minimized (maximized), and the variability of random outcomes is not considered. However, different managers might have different risk preferences depending on the product characteristics. As a result, risk-averse inventory models such as the newsvendor problem have received increasing attention in recent years.

We have decided to extend our single-product model to incorporate the different degrees of risk-aversion using the CVaR criterion [48]. To do so, we need to change the previous objective function (3.27) to the following:

$$\min_{\gamma \in \mathbb{R}} \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n} \sum_{i=1}^{n} (\tilde{c}_i - \gamma)^+ \tag{3.31}$$

which can be equivalently formulated as:

$$\begin{aligned} \min_{\gamma\in\mathbb{R},\ y_i\in\mathbb{R}} \quad & \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} y_i \\ \text{s.t.} \quad & \\ & y_i \geq \tilde{c}_i - \gamma,\ i \in [n] \\ & y_i \geq 0,\ i \in [n] \end{aligned} \tag{3.32}$$

where β is a value between 0 and 1 and the value $(1-\beta)$ corresponds to the case where we consider the worst $(1-\beta)$ percentage of the costs. In other words, the β-CVaR is the conditional expectation of losses above the β-quantile, where increasing the value of β indicates increasingly more risk-averse cases. Note that that the terms that we used before in the objective function that correspond to the complexity of the decision tree and the policy parameters should still be added to the objective for the purpose of regularization. Therefore, the overall objective function for the risk-averse newsvendor becomes:

$$\min \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} (\tilde{c}_i - \gamma)^+ + \alpha \sum_{t\in\mathcal{T}_B} d_t + \lambda\|q\| \tag{3.33}$$

And the following constraints should be added to the model:

$$y_i \geq \tilde{c}_i - \gamma,\ i \in [n] \tag{3.34}$$

$$y_i \geq 0,\ i \in [n] \tag{3.35}$$

The complete tree-based model for the single-product risk-averse newsvendor problem then becomes:

$$\begin{aligned} \min \quad & \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} y_i + \alpha \sum_{t\in\mathcal{T}_B} d_t + \lambda\|q\| \\ \text{s.t.} \quad & \\ & y_i \geq \tilde{c}_i - \gamma,\ i \in [n], \\ & bu_{it} + ho_{it} \leq c_{it},\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \\ & u_{it} \geq D_i - q_t^1 - \sum_{j=2}^{p} q_t^j x_i^j,\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \\ & o_{it} \geq q_t^1 + \sum_{j=2}^{p} q_t^j x_i^j - D_i,\ \forall i \in [n],\ \forall t \in \mathcal{T}_L, \end{aligned} \tag{3.36}$$

$$c_{it} \leq \tilde{c}_i + (1 - z_{it})M, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$a_m^T x_i \geq b_m - (1 - z_{it}), \ \forall i \in [n], \forall t \in \mathcal{T}_L, \forall m \in \mathcal{R}(t),$$

$$a_m^T (x_i + \epsilon) \leq b_m + (1+\epsilon_{max})(1 - z_{it}),$$
$$\forall i \in [n], \ \forall t \in \mathcal{T}_L, \ \forall m \in \mathcal{L}(t),$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \ \forall i \in [n],$$

$$z_{it} \leq l_t, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \ \forall t \in \mathcal{T}_L,$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \ \forall t \in \mathcal{T}_B,$$

$$1 - d_t \leq b_t \leq 1, \ \forall t \in \mathcal{T}_B,$$

$$d_t \leq d_{p(t)}, \ \forall t \in \mathcal{T}_B \setminus \{1\},$$

$$z_{it}, l_t \in \{0, 1\}, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$a_{jt}, d_t \in \{0, 1\}, \ j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B,$$

$$u_{it} \geq 0, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$o_{it} \geq 0, \ \forall i \in [n], \ \forall t \in \mathcal{T}_L,$$

$$y_i \geq 0, \ \forall i \in [n].$$

Chapter 4

Numerical Experiments

In this chapter, we discuss the dataset used in order to evaluate the models developed in Chapter 3. The main goal of this chapter is to describe the steps taken to preprocess and prepare the data in a way that it consists of both demand data and feature information. Additionally, we will also discuss our approach for tuning the hyper parameters used in models (3.28) and (3.36).

4.1 Data

In order to provide a comprehensive benchmark of real-world performance, we used a dataset obtained from the UCI Machine Learning Repository [60] which is a source for datasets that are used regularly for reporting and comparing the performance of different supervised and unsupervised learning algorithms. The dataset we have chosen is an online transaction dataset that belongs to a UK retail from 2010-2011 [64, 65], where instead of demand, we have information about the sales quantities (in other words, we work with censored demand data). Originally, the dataset had columns corresponding to the invoice number and date, country in which the product was sold, product and stock codes, quantity sold, unit price, and a short description. Although not all these data inputs have an effect on the demand, an important factor that can have meaningful effects on the sales is seasonality. Using the invoice dates, we extracted the year, month of the year, day of the month, day

of the week, and hour of sale for each record and regarded each of these columns as a feature in addition to the unit price which is given as a column in the dataset. This means that the input to our algorithms consists of six features in total along with the sales records which is the dependant variable.

Initially, the dataset had a column showing in which country the sale took place. However, in reality, it seems more reasonable to order for each country separately in a retail that might have different branches in different countries. As a result, we filtered the dataset on different countries and treated each country as a separate dataset. However, not all datasets were suited for the purpose of our numerical experiment. We developed some criteria in order to choose the dataset(s) that could best serve our purpose. First, we looked for a dataset that has at least 400 records since our ultimate goal was to use our model on datasets with sizes in the hundreds. Moreover, with lower number of records, when splitting the dataset for validation and testing purposes, the number of records might not be enough. Second, one of the advantages of decision trees is mapping nonlinear inputs to the output. When the relationship between features and the dependant variable is linear, using a simpler algorithm such as multivariate linear regression might be a better choice since decision trees most probably overfit the data and add unnecessary complexities to the model. As a result, we looked for a dataset that has a nonlinear relation between the output and at least one of the features, and at the same time is representative of the whole dataset (in terms of correlations, having records for both years, type of relations between features and output, etc.). Using these criteria, we found a dataset with $n = 684$ records to use for our numerical experiments explained in the following sections of this chapter.

Additionally, based on the descriptions provided in the dataset, one can quickly notice that all the records belong to one category of products. For this reason, we decided to work with the dataset as if it is a single-product newsvendor problem.

In order to briefly showcase the performance of the multi-product model with and without the capacity constraint, and due to unavailability of real-world data for this purpose, we used synthetic datasets with a varying number of observations. These datasets were created using the make_regression from the Scikit-learn package in

python that can create random linear regression problems by generating an arbitrary number of input features, output targets, controllable degree of informative coupling between them, and Gaussian centered noise with adjustable scale. In order to add nonlinearity to the data, we squared the output variables created by the function. For simplicity, the number of output variables (which is the same as the number of products), and the number of features were kept the same and only the number of observations were changed in each dataset. We created datasets with 3 features, 2 products, and 100, 150, 200, 250 number of observations.

4.2 Regularization

An important challenge when working with machine learning algorithms is hyper-parameter tuning. There are two hyper-parameters in this problem:

- The complexity parameter α that controls the trade off between the performance and the number of splits in the decision tree in a way that ensures the tree will not be unnecessarily large. This hyper parameter ensures that increasing the number of splits in the tree leads to a an increase in the predictive performance (decrease in cost) that is sufficiently large.
- The regularization parameter λ that affects the generalizability of the algorithm. In other words, using this hyper parameter, we are able to control the number of features used in each leaf of the model and avoid unnecessary complications added to the model.

In order to avoid overfitting to the training data, we need to find the right value for the above mentioned hyper parameters. Typically in the process of tuning the hyper parameters, a part of the training data is reserved as a validation set to assess the performance of the trained model and to guide the choice of hyper parameters. The performance of the model on the validation set can be used as a proxy for the real out-of-sample performance and we expect the hyper parameters that performed well on the validation set to also perform well on the unseen data (test data).

A well-known approach to tuning hyper parameters is performing a grid search, where the model is trained on every possible combination of the hyper parameters using the training data set. Each model then is evaluated on the validation set in order to choose the combination of hyper parameters that give us the best performance. Although this seems to be a reasonable method, it might not be feasible to do a grid search when hyper parameters are continuous or have a large range of possible values. A potential solution for performing grid search on continuous-valued hyper-parameters is to discretize the range of possible values and select the best from this set. Bertsimas and Dunn [55] showed that despite the complexity parameter being continuous-valued, it has a discrete effect on the tree and they managed to use this method to tune the hyper parameters in OCTs and this has motivated us to follow the same method when tuning hyper parameters in our model since both of the hyper parameters are continuous and doing a grid search is not a feasible solution as discussed before.

In the following sections, we will discuss the implementation of the numerical experiments on (3.28) and (3.36) models in more depth.

4.3 Experiment Setup

4.3.1 Single-Product Newsvendor Problem

In order to assess the performance of the tree-based single-product newsvendor algorithm given by model (3.28), we used Ban and Rudin's big data newsvendor model (3.6) [4] as a benchmark. We used the same implementations for the tree-based model and Ban and Rudin's big data newsvendor model as in Chapter 3. Additionally, we were also curious to compare the performance of our algorithm with heuristics such as CART. Similar to standard linear regression, CART uses the mean squared error approach to calculate the predicted outcome variable. However, as mentioned before, the cost function in the newsvendor problem is equivalent to the loss function in quantile regression (the only difference is that in the newsvendor problem, we investigate the cost of the estimated quantile as opposed to the

estimated quantile itself) [4]. Quantile regression is an extension of standard linear regression that estimates the conditional quantile of the outcome variable with a general linear model that assumes a non-parametric form for the conditional distribution of the response variable. Therefore, in order to make a more meaningful comparison, we used the CART algorithm with a quantile loss (the quantile used in CART is calculated as $\frac{b}{b+h}$ where, similar to our model, b and h stand for the unit backordering and holding costs respectively) instead of the default objective of the algorithm which is minimizing the mean squared error (MSE).

We limited our experiments to decision trees with depth two because of the time limitations and speed of the algorithm, and similarly, we limited the quantile regression trees' depth to two for the purpose of comparison. It is worth mentioning that the quantile regression algorithm was readily available in the machine learning package that was used and the automatic tuning of the algorithm was not changed. Because of the low depth of the tree in the experiments, we chose to set the complexity parameter α equal to zero meaning that we decided not to perform the cost-complexity pruning in the decision tree algorithms since a depth two decision tree is not considered as complex due to the low number of leaves in comparison to decision trees with larger depths.

In order to tune the regularization parameters in the tree-based and big data newsvendor algorithms, in each experiment, we randomly divided the input data into three sets of training, validation, and test sets using the $80-10-10$ split percentages respectively, and the regularization method discussed in Section 4.2 was performed. We started with different powers of 10 (10^{-1}, 10^{-2}, etc.) as different regularization parameters to find the discretized range of possible values for our experiment. If the result did not change between two subsequent values, we used the mid-point as a new value to explore. Using this method, we developed vectors of possible values for the regularization parameters for each of the two algorithms (tree-based and big data newsvendor).

Additionally, to investigate the effect of increasing the number of observations on the performance of the algorithms, we started by randomly sampling 50 points from the dataset and, in each experiment, we increased the number of points by 50 until

we reached 400 points. However, in order to minimize the effect of the particular sampling and splitting of the data into training, validation and testing sets, the entire process was conducted thirty times for each experiment (each experiment differs in the number of observations used as input data), with a different splitting and random sampling each time. The final in-sample and out-of-sample costs were then obtained by averaging the results across these thirty runs.

4.3.2 Risk-Averse Newsvendor Problem

In order to evaluate the performance of the risk-averse newsvendor algorithm, we altered Ban and Rudin's big data newsvendor model (3.6) so that it would represent a risk-averse newsvendor model as well. This is mainly because of the fact that we found no other similar algorithm (in terms of being non-parametric, using covariate data, and using the same risk measurement) to use as a benchmark. There are many risk-averse newsvendor models, but most of them are parametric, or they investigate a specific scenario in the context of inventory management.

Similar to what we discussed in Section 3.3.3, we need to change the objective of the model first. However, before changing the objective, we will first change the previous model to the following:

$$\begin{aligned}
\min \quad & \frac{1}{n}\sum_{i=1}^{n} c_i + \lambda\|q\|_k \\
\text{s.t.} \quad & \\
& u_i \geq D_i - q^1 - \sum_{j=2}^{p} q^j x_i^j,\ i = 1, \ldots, n \\
& o_i \geq q^1 + \sum_{j=2}^{p} q^j x_i^j - D_i,\ i = 1, \ldots, n \\
& bu_i + ho_i \leq c_i,\ i = 1, \ldots, n \\
& u_i,\ o_i \geq 0
\end{aligned} \tag{4.1}$$

The objective of the model then should change to the following to correspond to the big data risk-averse newsvendor model:

$$\min_{\gamma \in \mathbb{R}} \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} (c_i - \gamma)^+ \tag{4.2}$$

which is equivalent to the following:

$$\begin{aligned} \min_{\gamma\in\mathbb{R},\ y_i\in\mathbb{R}} \quad & \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} y_i \\ \text{s.t.} \quad & \\ & y_i \geq c_i - \gamma,\ i \in [n] \\ & y_i \geq 0,\ i \in [n]. \end{aligned} \tag{4.3}$$

We can then reformulate the complete big data risk-averse newsvendor model as the following in order to use it as a benchmark.

$$\begin{aligned} \min \quad & \gamma + \frac{1}{(1-\beta)} \cdot \frac{1}{n}\sum_{i=1}^{n} y_i + \lambda\|q\| \\ \text{s.t.} \quad & \\ & y_i \geq c_i - \gamma,\ i \in [n], \\ & bu_i + ho_i \leq c_i,\ \forall i \in [n], \\ & u_i \geq D_i - q^1 - \sum_{j=2}^{p} q^j x_i^j,\ i = 1, \ldots, n \\ & o_i \geq q^1 + \sum_{j=2}^{p} q^j x_i^j - D_i,\ i = 1, \ldots, n \\ & u_i,\ o_i, y_i \geq 0. \end{aligned} \tag{4.4}$$

We will use the best values found for regularization parameters of the tree-based model and big data newsvendor model in the previous experiments since the data and sampling will be the same. However, similar to single-product implementations, we will use thirty samples in each experiment and average the results as the final in-sample and out-of-sample costs for both models. We will investigate different values of $\beta = 0.5,\ 0.8,\ 0.9,\ 0.95$, which indicate increasingly more risk-averse cases in order to assess the performance of the algorithms.

Chapter 5

Results and Discussion

In this chapter, we present the results obtained from solving the tree-based single-product newsvendor algorithms for both cases of risk-neutrality and risk-aversion. All models were coded in MATLAB using YALMIP, which is a free toolbox for rapid prototyping of optimization problems, and solved using Gurobi solver on a PC with the following specifications:

Processor: Intel(R) Core(TM) i7, CPU 2.70GHz, 2.90 GHz

RAM: 8.00 GB

System type: 64 – bit Operating System, x64 – based Processor

The analysis of the results is divided into two main sections. First, we compare the in-sample and out-of-sample costs of the risk-neutral tree-based newsvendor model with those from the big data newsvendor model and the CART algorithm with a quantile loss. Then, we investigate the performance of the risk-averse tree-based newsvendor model with different degrees of risk aversion, and compare the results with those from the risk-averse big data newsvendor model described in Chapter 4.

5.1 Results

In this section, the performance of the single-product risk-neutral newsvendor, multi-product risk-neutral newsvendor, and single-product risk-averse newsvendor models' are presented.

5.1.1 Single-Product Risk-Neutral Newsvendor

The in-sample and out-of-sample costs for the risk-neutral single-product newsvendor models are presented in Table 5.1. Each of the costs in Table 5.1, is the average of thirty random samples drawn from the data.

Table 5.1: Comparison of Risk-Neutral Single-Product Models' Performances

n	Method	In-sample Cost	Out-of-sample Cost
50	CART	0.9064	1.0126
	Big Data Model	0.8852	1.0877
	Big Data Regularized	0.9391	1.0419
	Tree-based Model	0.0512	1.3678
	Tree-based Regularized	0.0513	0.8420
100	CART	0.4920	1.1328
	Big Data Model	0.4449	1.0179
	Big Data Regularized	0.4448	1.0193
	Tree-based Model	0.0846	0.5404
	Tree-based Regularized	0.0853	0.4362
150	CART	0.3536	0.8003
	Big Data Model	0.3408	0.7788
	Big Data Regularized	0.3421	0.7739
	Tree-based Model	0.1047	0.6103
	Tree-based Regularized	0.1047	0.3024
200	CART	0.3301	0.7735
	Big Data Model	0.2970	0.7614

	Big Data Regularized	0.2970	0.7612
	Tree-based Model	0.1198	0.3040
	Tree-based Regularized	0.1198	0.3099
250	CART	0.3053	0.6838
	Big Data Model	0.2771	0.6741
	Big Data Regularized	0.2771	0.6741
	Tree-based Model	0.1250	0.2259
	Tree-based Regularized	0.1250	0.2257
300	CART	0.3115	0.5928
	Big Data Model	0.2948	0.5823
	Big Data Regularized	0.2948	0.5823
	Tree-based Model	0.1325	0.2108
	Tree-based Regularized	0.1358	0.2004
350	CART	0.2852	0.5400
	Big Data Model	0.2788	0.5295
	Big Data Regularized	0.2788	0.5295
	Tree-based Model	0.1395	0.1905
	Tree-based Regularized	0.1394	0.1882
400	CART	0.2736	0.5296
	Big Data Model	0.2676	0.5179
	Big Data Regularized	0.2676	0.5179
	Tree-based Model	0.1350	0.2061
	Tree-based Regularized	0.1349	0.2071

As expected, the in-sample costs of the tree-based algorithms with and without regularization is considerably lower than that for the big data models and the CART with quantile loss. It is worth mentioning that unlike interpretable tree algorithm, CART uses constant predictions in each leaf, meaning that after the decision tree is grown and observations are assigned to leaves, in each leaf, the final predicted value will be the average of observations' target values in that leaf.

Based on Table 5.1, it can be seen that the in-sample costs of the tree-based model is lower than CART and the big data model in all cases. Moreover, the performance of the tree-based model is more stable in comparison to that of the other models where increasing the number of points has a substantial effect on their performance. In other words, one of the advantages of the tree-based model is its performance in cases where there is a lack of enough historical observations.

Although the proposed algorithm outperforms the big data newsvendor and CART with quantile loss on the training data, it is more important to compare the performance of these models on the unseen data (test data) in order to evaluate the predictability of the tree-based model and make sure that the algorithm did not overfit or underfit to the training data.

Figure 5.2 compares the performance of different models' on the unseen data and the out-of-sample costs as the number of observations increases. The out-of-sample cost for the tree-based algorithm with 50 points is higher than the other models. However, after regularizing the algorithm, as visualized in figure 5.1, the cost decreased significantly and the regularized model outperformed all the other models. This is because we train the algorithm on 80% of the data, which in this case will be only 40 points, and this is a very small amount of data for training and the algorithm overfit to the training data which was later solved by regularizing the policy parameters.

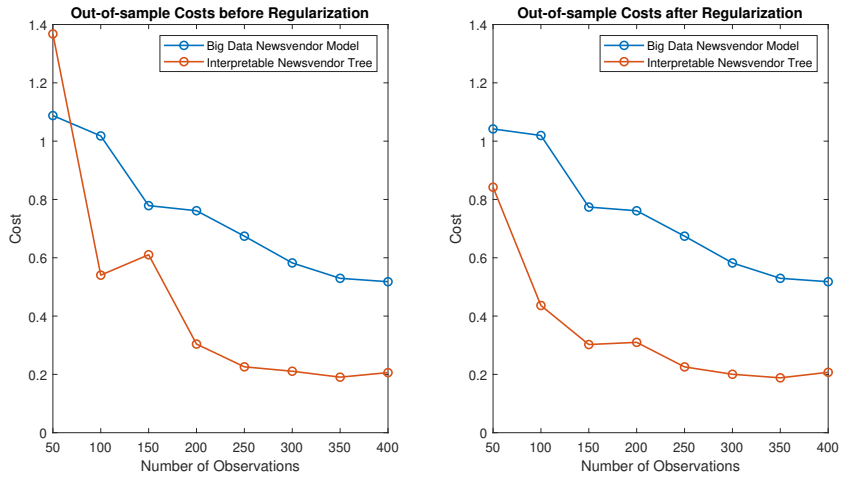

Figure 5.1: Comparison of Out-of-sample Costs of Big Data and Tree-based Models before and after Regularization

Additionally, Figure 5.2 shows that the tree-based algorithm outperforms the other models as the number of points increases above 50 even without regularization. By increasing the number of observations, the difference between the cost of models with regularization and without regularization becomes smaller in both big data and tree-based algorithms. Moreover, there are some cases where the CART with quantile loss outperforms the big data newsvendor model, but regardless of the number of observations, the cost of CART is always higher than the regularized tree-based model.

The out-of-sample performance of the model shows that the regularized tree-based algorithm is able to find an optimal solution that outperforms the other two models regardless of the number of observations. This means that even if the decision-maker only has access to limited data, the tuned tree-based algorithm will provide one a solution with considerably lower costs than Ban and Rudin's big data newsvendor models and quantile regression trees. Figure 5.3 further shows the comparison of the different models out-of-sample performance after regularization.

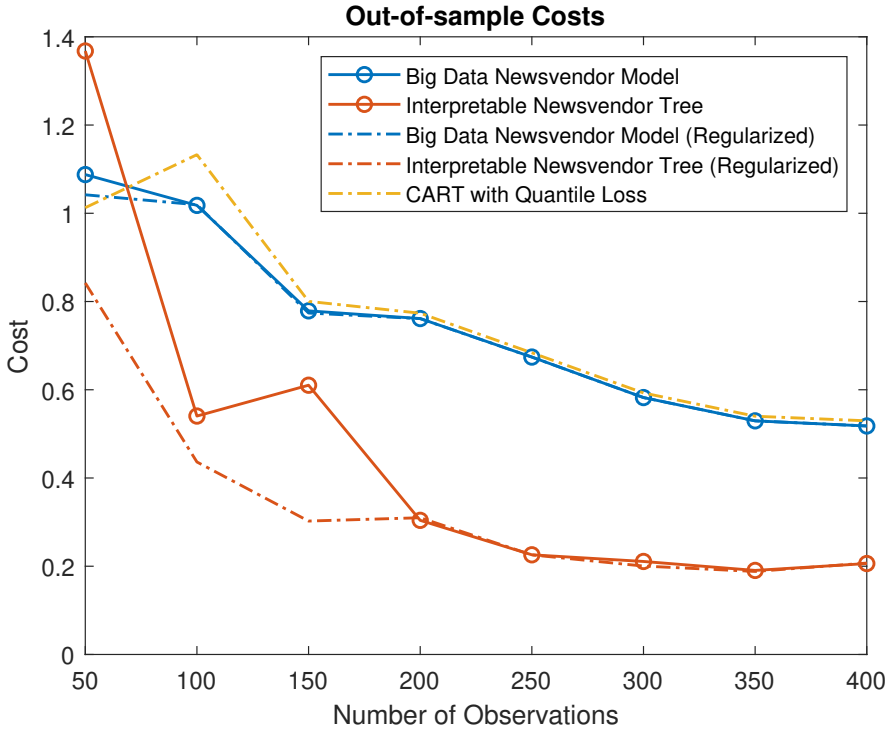

Figure 5.2: Comparison of Out-of-Sample Costs for Different Models

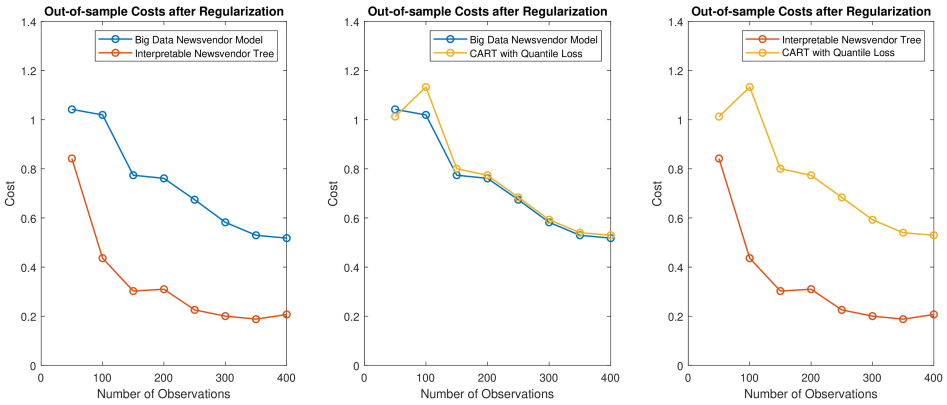

Figure 5.3: Detailed Comparison of Out-of-Sample Costs for Different Models

In addition to the improved performance of the tree-based model, as discussed before, one of the advantages of this algorithm is its interpretability. One can easily visualize the tree to see how a decision was made and gain valuable information. To further discuss this interpretability and show how to read the results obtained from the model, a fully grown optimal decision tree with $n = 100$ and without regularization is shown in Figure 5.4. Figure 5.4 shows the resulting tree after solving the model and getting the optimal solution of variables a_{jt} and b_t, $t \in \mathcal{T}_B$, that are shown below.

$$b = \begin{bmatrix} 0.0614 & 0.1666 & 0.4545 \end{bmatrix} \tag{5.1}$$

$$a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.2}$$

Each column in the vector b and matrix a represent a branch node starting from the root node. For example, for the root node in Figure 5.4, we need the values a_1 and b_1 to replace in the left and right branches. The value of b_1 is the first value in vector b which is 0.0614 as shown above, and a_1 is the first column of matrix a. Since our algorithm creates univariate regression trees, in each column of matrix a only one value is 1 and the rest is 0. In other words, only one feature is chosen to branch on in each branch node. In this case, the first feature, which is the price, is

chosen in the first branch. If the value of first feature in any observation is less than $b_1 = 0.0614$, this observation will follow the left branch. Otherwise, it will follow the right branch. The same process then continues in the second and third branch until each point is assigned to only one leaf. It is important to mention that the optimal values of vector b are based on the normalized features. In order to retrieve the actual values before normalizing, we should multiply each feature's breaking point by the maximum value of that feature (note that the outcome is ultimately calculated based on the normalized values but we converted the breaking points to show how one can make sense of the tree). The resulting tree is shown in figure 5.4 where "Day" stands for day of the month and "Hrs" stands for hour of the day.

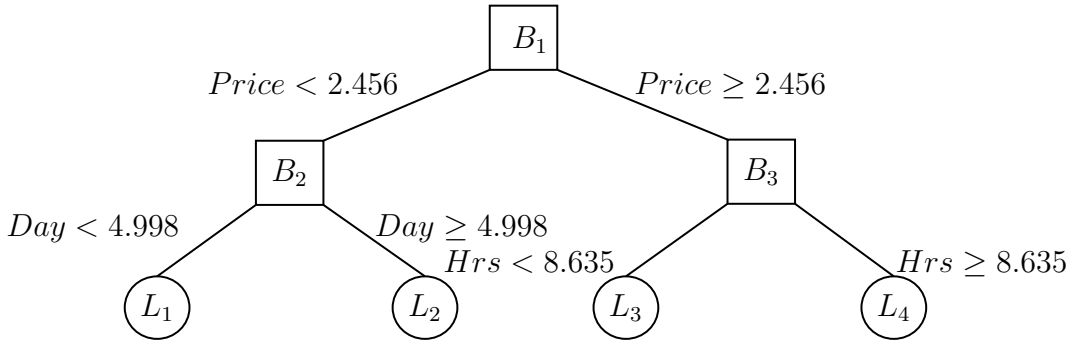

Figure 5.4: Example of an Interpretable Decision Tree with $n = 100$ Observations

Based on the optimal values of matrix z, we can also figure out the number of observations going through each branch node and in each leaf node as well as the index of the points in each leaf node. This information can be useful in classifying observations and figuring out how common each classification is as well as figuring out where they are lacking data for future data collections. Moreover, decision makers can easily follow the decisions based on the features of each observation to the leaf it is assigned to, and figure out the decision policies that has led to such decision.

Although gaining insightful information about training data can be important, since this is a predictive model, it is more important to know how a prediction was made on unseen data. The above mentioned structure of the tree is used to classify observations but after optimizing the model, the decision-maker also has access to

the decision policies in each leaf. These decision policies assign weights to different features in the different leaves. For this example, the policy parameters are shown in Table 5.2.

Table 5.2: The Slope and Intercept of Ordering Policies in Different Leaves of the Interpretable Tree with $n = 100$ Observations

	Leaf 1	Leaf 2	Leaf 3	Leaf 4
Intercept	-2.6707	0.9916	1.1114	-22.4240
Price	-1.5936	-1.7448	0	-0.1784
Hour	4.6663	0.0352	-0.1094	-0.9990
Weekday	4.0932	-0.0403	0.0416	-0.9474
Day of Month	3.0873	0.0041	-0.0062	0.4385
Month	-4.5789	-0.0013	-0.0196	-0.3219
Year	2.5189	-0.8435	-1.698	23.8590

Based on Table 5.2, decision-makers can identify the type of relationship between a feature and the dependant variable (whether increasing the value of a feature would increase the dependant variable or decrease it) as well as its weight where both the relationship and the weight differ from one leaf to another. For example, based on Figure 5.4, we can see that observations in leaves 1 and 2 differ in the fourth feature which is the day of the month. This means that for observations with the day of the month less than 4.998, this feature has a positive and significant impact on the predicted order quantity whereas observations with the day of the month being more than 4.998, the effect of this feature although still positive, becomes insignificant. The relationship of some features with the outcome are consistently positive or negative while for some features, it can differ. For instance, the relationship between the month and the outcome is negative in all leaves. However, the hour of the day affects the outcome positively in leaves 1 and 2, and negatively in leaves 3 and 4, meaning that for lower priced instances in leaves 1 and 2, the later it gets during the day, the higher the demand gets and it is more important to pay attention to the stocking of these products during the later hours of the day, but for higher priced products in leaves 3 and 4, the demand decreases based on the hour of day and the

negative effect is greater in leaf 4 that represents the later hours of the day based on the breaking point in branch 3. This is only one example of the interpretation and insights that a decision tree can provide to the decision makers in comparison to more complex models. Using such insights can heavily affect the understanding of managers and decision makers of the given predictions, and as it was discussed, it is quite straightforward to analyze and investigate interpretations even for non-experts that might be in charge of decision making. It is worth mentioning that this algorithm can also be used in other similar settings such as staffing in healthcare, as mentioned before in Section 3.2, where interpretation is usually preferred over slightly more accurate but more complex models [55].

5.1.2 Mutli-Product Risk-Neutral Newsvendor

The results of the multi-product newsvendor problem without the capacity constraint are presented in Table 5.3. Each row of Table 5.3 represents a different dataset with the specified number of observations used. However, the number of features and products in all experiments remains the same where the number of features used to create the datasets is 3, and the number of products is 2 (the low number of features and products are mainly because of the limited time to conduct experiments on the multi-product extension of the model and not the inability of the model in solving bigger problems). The results of the interpretable newsvendor model is compared against the big data newsvendor model for multi-product case.

Table 5.3: Comparison of Risk-Neutral Mutli-Product Models' Performances without Capacity Constraint

n	Big Data In-Sample Cost	Big Data Out-of-Sample Cost	Tree-Based In-Sample Cost	Tree-Based Out-of-Sample Cost
100	0.5159	0.6505	0.2241	0.5129
150	0.1856	0.2755	0.1287	0.2337
200	0.2295	0.2812	0.1637	0.2607
250	0.1894	0.2106	0.1416	0.1926

It is important to mention that none of the models are regularized (since the trees are shallow and the predictions are constant in each leaf after adding the capacity constraint) and each dataset is divided into training and test sets where 80% of the data is used for training and 20% is used for assessing the performance of the models on unseen data. As expected, the in-sample and the out-of-sample costs of the tree-based model is lower than the big data model in all experiments.

We have also incorporated the capacity constraint in both the tree-based and the big data newsvendor models for all four datasets. For these experiments, we started with a randomly chosen limited capacity and in each experiment increased it by a certain value. The following tables show the results of capacitated models as we increase the capacity on each dataset.

Table 5.4: Comparison of Risk-Neutral Mutli-Product Models' Performances with Capacity Constraint for $n = 100$ Observations

Capacity	Big Data In-Sample Cost	Big Data Out-of-Sample Cost	Tree-Based In-Sample Cost	Tree-Based Out-of-Sample Cost
0.6	0.6826	0.6231	0.6139	0.5792
0.8	0.5559	0.5902	0.4850	0.5622
1.0	0.5475	0.5751	0.4318	0.5467
1.2	0.5475	0.5751	0.4038	0.5243

Table 5.5: Comparison of Risk-Neutral Mutli-Product Models' Performances with Capacity Constraint for $n = 150$ Observations

Capacity	Big Data In-Sample Cost	Big Data Out-of-Sample Cost	Tree-Based In-Sample Cost	Tree-Based Out-of-Sample Cost
0.6	0.9991	1.2942	0.9442	1.2317
0.8	0.7026	0.9057	0.5556	0.8599
1.0	0.5575	0.7210	0.3521	0.5299

1.2	0.5436	0.6680	0.2838	0.4020
1.4	0.5436	0.6680	0.2712	0.3386
1.6	0.5436	0.6680	0.2684	0.3123

Table 5.6: Comparison of Risk-Neutral Mutli-Product Models' Performances with Capacity Constraint for $n = 200$ Observations

Capacity	Big Data In-Sample Cost	Big Data Out-of-Sample Cost	Tree-Based In-Sample Cost	Tree-Based Out-of-Sample Cost
0.6	1.3261	1.4222	1.2334	1.3121
0.8	0.8864	0.9503	0.7554	0.8201
1.0	0.6341	0.6605	0.4668	0.4948
1.2	0.5420	0.5750	0.3352	0.3567
1.4	0.5420	0.5750	0.2957	0.3204
1.6	0.5420	0.5750	0.2953	0.3195

Table 5.7: Comparison of Risk-Neutral Mutli-Product Models' Performances with Capacity Constraint for $n = 250$ Observations

Capacity	Big Data In-Sample Cost	Big Data Out-of-Sample Cost	Tree-Based In-Sample Cost	Tree-Based Out-of-Sample Cost
0.6	0.9849	1.1247	0.9539	1.1079
0.8	0.6231	0.6871	0.5538	0.6438
1	0.4137	0.4528	0.3075	0.3293
1.2	0.3586	0.3793	0.2272	0.2394
1.4	0.3586	0.3793	0.2210	0.2335

Similar to the results for the non-capacitated multi-product newsvendor, the in-

sample and out-of-sample costs of the capacitated multi-product tree-based model is lower than the big data newsvendor model in all experiments regardless of the value used for capacity. As expected, by increasing the available capacity, both in-sample and out-of-sample costs of the tree-based and big data models decrease until the capacity is big enough that it no longer limits the optimal solution and after that the costs remain the same no matter how much we increase the capacity (in other words, in each table above, after the last capacity value presented, the costs remain the same. This upper-bound on capacity above which costs will no longer improve is dependant on the data and that is why this value differs in each table). The capacity threshold after which the costs no longer improve, is lower for the big data model in comparison to the tree-based model. However, for both models, the capacitated costs will remain higher than the non-capacitated models since in the capacitated models the policy parameters are simplified to constant predictions.

5.1.3 Single-Product Risk-Averse Newsvendor

In this section, results for the single-product risk-averse newsvendor model are presented. We have tested the tree-based algorithm and the big data newsvendor models with a risk-averse objective on varying number of observations and increasing values of beta. The detailed results of the in-sample and out-of-sample risks for models with $\beta = 0.5$ are shown in Table 5.8 and the detailed results for $\beta = 0.8, 0.9, 0.95$ are presented in Appendix A. Based on the results, it can be seen that, as predicted, the in-sample risks of the tree-based models remain less than Ban and Rudin's big data newsvendor models with the risk-averse objective in all cases.

Table 5.8: Comparison of Risk-Averse Single-Product Models' Performances with $\beta = 0.5$

n	Method	In-sample Risk	Out-of-sample Risk
50	Big Data Model	0.9868	1.0032
	Big Data Regularized	1.0167	0.9075
	Tree-based Model	0.0546	0.7625

	Tree-based Regularized	0.0547	0.7608
100	Big Data Model	0.4684	0.9939
	Big Data Regularized	0.4680	0.9934
	Tree-based Model	0.0956	0.4247
	Tree-based Regularized	0.0966	0.3844
150	Big Data Model	0.3681	0.7762
	Big Data Regularized	0.3661	0.7775
	Tree-based Model	0.1150	0.4495
	Tree-based Regularized	0.1149	0.3120
200	Big Data Model	0.3241	0.7567
	Big Data Regularized	0.3239	0.7564
	Tree-based Model	0.1304	0.4216
	Tree-based Regularized	0.1306	0.3644
250	Big Data Model	0.3043	0.6750
	Big Data Regularized	0.3043	0.6750
	Tree-based Model	0.1341	0.2625
	Tree-based Regularized	0.1341	0.2255
300	Big Data Model	0.3202	0.5891
	Big Data Regularized	0.2302	0.5891
	Tree-based Model	0.1421	0.2316
	Tree-based Regularized	0.1422	0.2326
350	Big Data Model	0.3081	0.5434
	Big Data Regularized	0.3081	0.5434
	Tree-based Model	0.1486	0.2104
	Tree-based Regularized	0.1487	0.2114
400	Big Data Model	0.2973	0.5341
	Big Data Regularized	0.2973	0.5341
	Tree-based Model	0.1436	0.2231
	Tree-based Regularized	0.1436	0.2232

As mentioned before, we are mainly interested in the out-of-sample performance of the models. Results of the out-of-sample risks for different models are summarized in Figures 5.5 to 5.8.

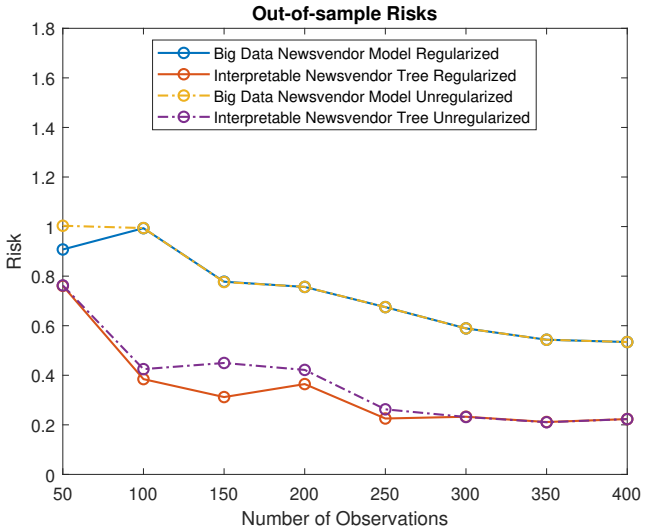

Figure 5.5: Out-of-Sample Risk Comparison of Risk-Averse Models with $\beta = 0.5$

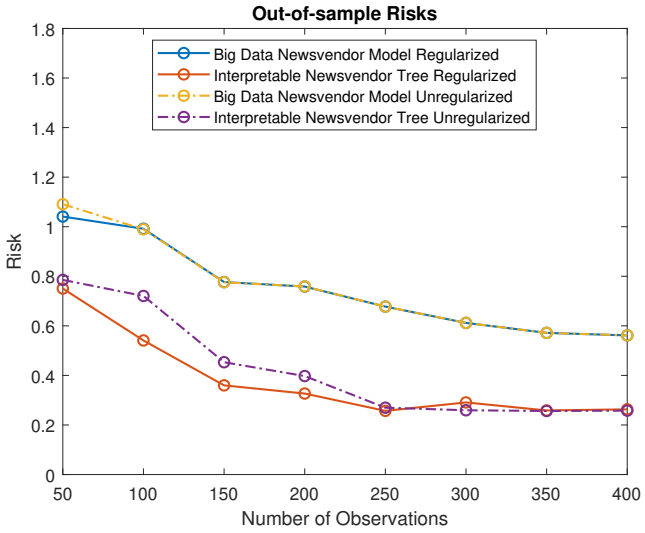

Figure 5.6: Out-of-Sample Risk Comparison of Risk-Averse Models with $\beta = 0.8$

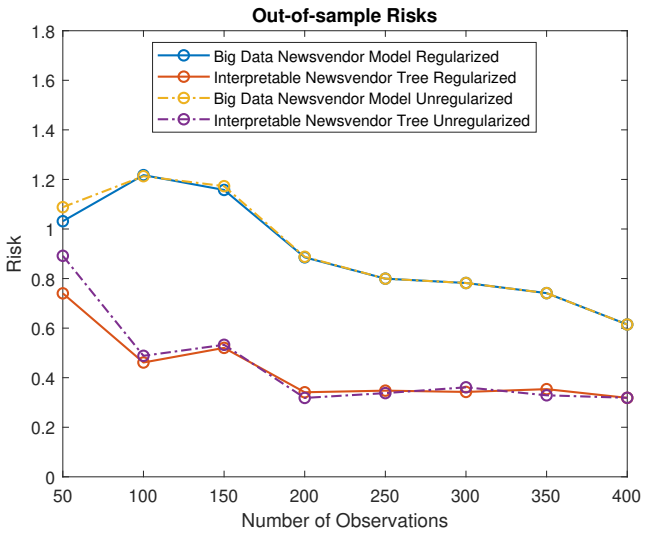

Figure 5.7: Out-of-Sample Risk Comparison of Risk-Averse Models with $\beta = 0.9$

It can be seen that the out-of-sample risks of the regularized tree-based model is lower than the big data models with and without regularization in all cases.

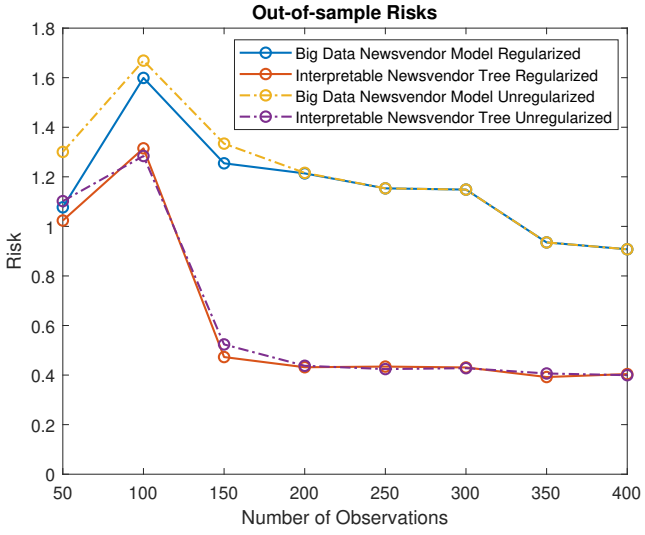

Figure 5.8: Out-of-Sample Risk Comparison of Risk-Averse Models with $\beta = 0.95$

5.2 Conclusion

In this section, we conclude that the proposed risk-neutral tree-based algorithm is successful at reducing the cost in the newsvendor problem without having any information about the demand distribution. We compared our algorithm against Ban and Rudin's big data newsvendor algorithm which is a novel contextual solution to the

newsvendor problem and reached comparatively lower costs. We also showed that the tree-based algorithm outperforms the heuristic decision tree algorithm CART by creating the optimal tree in a single step. Although our algorithm was aimed to predict based on large amounts of data, we observed that our proposed algorithm outperforms both CART and Band and Rudin's big data newsvendor model even in cases where the decision maker does not have access to large datasets and is trying to make a prediction based on limited historical data. This is important because decision trees tend to overfit to training data and have poor out-of-sample performance when they are grown on small amounts of data. However, we were able to take advantage of regularization in order to avoid overfitting on small samples and reach optimal decision policies that would ultimately lead to lower costs. As we showed previously, in such cases, the big data newsvendor model and CART predictions result in considerably higher costs even when the models are regularized in order to avoid overfitting.

We also investigated two extensions of the risk-neutral single-product newsvendor problem, the risk-neutral multi-product newsvendor problem, and the risk-averse single-product newsvendor model. We updated Ban and Rudin's big data newsvendor model in each extension for the purpose of comparison since it was originally aimed for single-product risk-neutral newsvendor problem. In the risk-averse single-product newsvendor extension, we tested our model against the risk-averse version of Ban and Rudin's newsvendor model and showed that regardless of the degree of risk-aversion, our model resultrf in lower costs both for in-sample and out-of-sample results.

For the multi-product risk-neutral newsvendor problem, we used synthetic datasets with different number of observations to test our model against the multi-product extension of Ban and Rudin's newsvendor model. Aside from the non-capacitated multi-product newsvendor, we also proposed a simplified model for the multi-product capacitated newsvendor since in reality, it is likely to have ordering, budget, or similar restraining constraints. We showed that our algorithm outperforms Ban and Rudin's model both in non-capacitated and capacitated extensions where we tried different values for the maximum capacity available.

5.3 Challenges and Limitations

The main limitation of this research is the execution time of the tree-based model. Generally, one of the main challenges of MIPs is their speed and the fact that depending on the type and number of the variables, the model's execution time can increase dramatically. For the tree-based newsvendor problem increasing the depth of the tree had a worse effect on the time of the problem than increasing the number of observations and features. As a result, we only tested the algorithm on decision trees with depth two. Mixed integer optimization models are known to speed up greatly when they are initiated with a feasible solution as a *warm start* [66]. Warm starts provide initial upper bounds on the optimal solution, and as the quality of warm start increases, MIP solver reaches an optimal solution more quickly [55, 61]. In the early stages of testing our algorithm, we provided the solution of CART as a warm start. However, the solution provided by CART is far from the final optimal solution and it did not speed up the problem as anticipated. As future work, it is possible to come up with a heuristic algorithm that can find a local optimal solution that is closer to the global optimum than the CART solution and use it as a warm start. Moreover, one can use the results from the shallow trees as a warm start for trees with higher depths. It is also worth mentioning that in order to deal with a large feature space and the increased execution time resulted from it, one can use dimensionality reduction algorithms such as Principal Component Analysis (PCA).

The next limitation is the unavailability of large-scale multi-product data. Despite the desirable performance of our algorithm in the single-product case, in order to effectively demonstrate the model's performance in a multi-product setting, we need to test it on real-world data. Unfortunately, we were unable to find information about real-world features to create synthetic multi-product data with covariates ourselves. Testing the model more rigorously with different data sets with a varying number of observations, features, and products can be a future work.

Additionally, there is still a lot to do in the area of the risk-averse newsvendor problem. Most of the literature to date, relies on model-based approaches even though the value of data-driven models and feature information were demonstrated

in the risk-neutral newsvendor solutions. There is room for exhaustive numerical experiments to determine the effect of feature information on the ordering behaviour and objective of the data-driven newsvendor model when the decision-maker is risk-averse.

Appendix A

Results of Risk-Averse Newsvendor Models

Here we present the detailed result of the risk-averse newsvendor models (both the tree-based, and the updated big data model).

Table A.1: Comparison of Risk-Averse Single-Product Models' Performances with $\beta = 0.8$

n	Method	In-sample Risk	Out-of-sample Risk
50	Big Data Model	1.2546	1.0907
	Big Data Regularized	1.3188	0.409
	Tree-based Model	0.0903	0.7853
	Tree-based Regularized	0.0919	0.7504
100	Big Data Model	0.8045	0.9895
	Big Data Regularized	0.8041	0.9917
	Tree-based Model	0.1270	0.7206
	Tree-based Regularized	0.1285	0.5410
150	Big Data Model	0.3785	0.7759
	Big Data Regularized	0.3819	0.7766
	Tree-based Model	0.1489	0.4531
	Tree-based Regularized	0.1478	0.3598

200	Big Data Model	0.3276	0.7589
	Big Data Regularized	0.3274	0.7584
	Tree-based Model	0.1707	0.3974
	Tree-based Regularized	0.1721	0.3268
250	Big Data Model	0.3080	0.6776
	Big Data Regularized	0.3080	0.6776
	Tree-based Model	0.1723	0.2696
	Tree-based Regularized	0.1733	0.2569
300	Big Data Model	0.3744	0.6118
	Big Data Regularized	0.3744	0.6118
	Tree-based Model	0.1814	0.2593
	Tree-based Regularized	0.1836	0.2909
350	Big Data Model	0.3658	0.5716
	Big Data Regularized	0.3658	0.5716
	Tree-based Model	0.1909	0.2564
	Tree-based Regularized	0.1913	0.2586
400	Big Data Model	0.3540	0.5614
	Big Data Regularized	0.3540	0.0.5614
	Tree-based Model	0.1868	0.2578
	Tree-based Regularized	0.1860	0.2631

Table A.2: Comparison of Risk-Averse Single-Product Models' Performances with $\beta = 0.9$

n	Method	In-sample Risk	Out-of-sample Risk
50	Big Data Model	1.2631	1.0880
	Big Data Regularized	1.3147	0.0314
	Tree-based Model	0.1118	0.8917
	Tree-based Regularized	0.1192	0.7403

100	Big Data Model	1.1719	1.2130
	Big Data Regularized	1.1732	1.2173
	Tree-based Model	0.1488	0.4882
	Tree-based Regularized	0.1517	0.4612
150	Big Data Model	1.1803	1.1728
	Big Data Regularized	1.1628	1.1575
	Tree-based Model	0.1806	0.5327
	Tree-based Regularized	0.1781	0.5203
200	Big Data Model	0.7128	0.0.8878
	Big Data Regularized	0.7105	0.8856
	Tree-based Model	0.2044	0.3183
	Tree-based Regularized	0.2077	0.3412
250	Big Data Model	0.6617	0.7995
	Big Data Regularized	0.6617	0.7995
	Tree-based Model	0.2203	0.3378
	Tree-based Regularized	0.2220	0.3477
300	Big Data Model	0.6763	0.7821
	Big Data Regularized	0.6763	0.7821
	Tree-based Model	0.2327	0.3611
	Tree-based Regularized	0.2380	0.3425
350	Big Data Model	0.6614	0.7412
	Big Data Regularized	0.6614	0.7412
	Tree-based Model	0.2652	0.3289
	Tree-based Regularized	0.2653	0.3538
400	Big Data Model	0.4524	0.6146
	Big Data Regularized	0.4524	0.6146
	Tree-based Model	0.2435	0.3188
	Tree-based Regularized	0.2466	0.3185

Table A.3: Comparison of Risk-Averse Single-Product Models' Performances with $\beta = 0.95$

n	Method	In-sample Risk	Out-of-sample Risk
50	Big Data Model	1.2519	1.3005
	Big Data Regularized	1.3097	0.0769
	Tree-based Model	0.1313	0.1018
	Tree-based Regularized	0.1394	1.0235
100	Big Data Model	1.2841	1.6688
	Big Data Regularized	1.3480	1.5990
	Tree-based Model	0.1644	1.2840
	Tree-based Regularized	0.1713	1.3149
150	Big Data Model	1.3794	1.3339
	Big Data Regularized	1.2871	1.2544
	Tree-based Model	0.2242	0.5236
	Tree-based Regularized	0.2245	0.4727
200	Big Data Model	0.1.1921	1.2159
	Big Data Regularized	1.1917	1.2138
	Tree-based Model	0.2633	0.4375
	Tree-based Regularized	0.2696	0.4316
250	Big Data Model	1.1566	1.1534
	Big Data Regularized	1.1566	1.1534
	Tree-based Model	0.2991	0.4241
	Tree-based Regularized	0.3074	0.4342
300	Big Data Model	1.1232	1.1484
	Big Data Regularized	1.1232	1.1484
	Tree-based Model	0.3314	0.4278
	Tree-based Regularized	0.3384	0.4307
350	Big Data Model	0.9150	0.9352
	Big Data Regularized	0.9150	0.9352
	Tree-based Model	0.3611	0.4064

	Tree-based Regularized	0.3646	0.3921
400	Big Data Model	0.8851	0.9076
	Big Data Regularized	0.8851	0.9076
	Tree-based Model	0.3820	0.3996
	Tree-based Regularized	0.3864	0.4040

Bibliography

[1] E. L. Porteus, "The newsvendor problem," in *Building Intuition*, pp. 115–134, Springer, 2008.

[2] R. Levi, G. Perakis, and J. Uichanco, "The data-driven newsvendor problem: new bounds and insights," *Operations Research*, vol. 63, no. 6, pp. 1294–1306, 2015.

[3] A. Oroojlooyjadid, L. V. Snyder, and M. Takáč, "Applying deep learning to the newsvendor problem," *IISE Transactions*, vol. 52, no. 4, pp. 444–463, 2020.

[4] G.-Y. Ban and C. Rudin, "The big data newsvendor: Practical insights from machine learning," *Operations Research*, vol. 67, no. 1, pp. 90–108, 2019.

[5] V. N. Vapnik and V. Vapnik, "Statistical learning theory. vol. 1 wiley," *New York*, 1998.

[6] I. Bratko, "Machine learning: Between accuracy and interpretability," in *Learning, networks and statistics*, pp. 163–177, Springer, 1997.

[7] W. Silva, K. Fernandes, M. J. Cardoso, and J. S. Cardoso, "Towards complementary explanations using deep neural networks," in *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, pp. 133–140, Springer, 2018.

[8] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[9] C. Molnar, *Interpretable Machine Learning*. Lulu. com, 2020.

[10] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[11] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1721–1730, 2015.

[12] F. Y. Edgeworth, "The mathematical theory of banking," *Journal of the Royal Statistical Society*, vol. 51, no. 1, pp. 113–127, 1888.

[13] P. M. Morse and G. E. Kimball, "Methods of operations i research, john wiley and sons," *Inc., New York*, 1951.

[14] "Novel advances in applications of the newsvendor model," *Decision Sciences*, vol. 47, no. 1, pp. 8–10, 2016.

[15] K. J. Arrow, T. Harris, and J. Marschak, "Optimal inventory policy," *Econometrica: Journal of the Econometric Society*, pp. 250–272, 1951.

[16] K. J. Arrow, S. Karlin, and H. E. Scarf, "Studies in the mathematical theory of inventory and production," 1958.

[17] H. Scarf, "The optimality of (5, 5) policies in the dynamic inventory problem," *Optimal pricing, inflation, and the cost of price adjustment*, 1959.

[18] K. S. Azoury, "Bayes solution to dynamic inventory models under unknown demand distribution," *Management science*, vol. 31, no. 9, pp. 1150–1160, 1985.

[19] W. S. Lovejoy, "Myopic policies for some inventory models with uncertain demand distributions," *Management Science*, vol. 36, no. 6, pp. 724–738, 1990.

[20] H. Scarf, "Bayes solutions of the statistical inventory problem," *The annals of mathematical statistics*, vol. 30, no. 2, pp. 490–508, 1959.

[21] L. H. Liyanage and J. G. Shanthikumar, "A practical inventory control policy using operational statistics," *Operations Research Letters*, vol. 33, no. 4, pp. 341–348, 2005.

[22] G. Gallego and I. Moon, "The distribution free newsboy problem: review and extensions," *Journal of the Operational Research Society*, vol. 44, no. 8, pp. 825–834, 1993.

[23] H. Scarf, "A min-max solution of an inventory problem," *Studies in the mathematical theory of inventory and production*, 1958.

[24] G. Perakis and G. Roels, "Regret in the newsvendor model with partial information," *Operations Research*, vol. 56, no. 1, pp. 188–203, 2008.

[25] X. Chen, M. Sim, and P. Sun, "A robust optimization perspective on stochastic programming," *Operations research*, vol. 55, no. 6, pp. 1058–1071, 2007.

[26] T. Homem-De-Mello, "Monte carlo methods for discrete stochastic optimization. uryasev s, pardalos pm, eds. stochastic optimization: Algorithms and applications," 2000.

[27] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[28] R. Levi, R. O. Roundy, and D. B. Shmoys, "Provably near-optimal sampling-based policies for stochastic inventory control models," *Mathematics of Operations Research*, vol. 32, no. 4, pp. 821–839, 2007.

[29] A. N. Burnetas and C. E. Smith, "Adaptive ordering and pricing for perishable products," *Operations Research*, vol. 48, no. 3, pp. 436–443, 2000.

[30] S. Kunnumkal and H. Topaloglu, "Using stochastic approximation methods to compute optimal base-stock levels in inventory control problems," *Operations Research*, vol. 56, no. 3, pp. 646–664, 2008.

[31] W. T. Huh and P. Rusmevichientong, "A nonparametric asymptotic analysis of inventory planning with censored demand," *Mathematics of Operations Research*, vol. 34, no. 1, pp. 103–123, 2009.

[32] G. A. Godfrey and W. B. Powell, "An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution," *Management Science*, vol. 47, no. 8, pp. 1101–1112, 2001.

[33] W. Powell, A. Ruszczyński, and H. Topaloglu, "Learning algorithms for separable approximations of discrete stochastic optimization problems," *Mathematics of Operations Research*, vol. 29, no. 4, pp. 814–836, 2004.

[34] D. Bertsimas and A. Thiele, "A data-driven approach to newsvendor problems," *Working Papere, Massachusetts Institute of Technology*, 2005.

[35] B. He, F. Dexter, A. Macario, and S. Zenios, "The timing of staffing decisions in hospital operating rooms: incorporating workload heterogeneity into the newsvendor problem," *Manufacturing & Service Operations Management*, vol. 14, no. 1, pp. 99–114, 2012.

[36] L. Hannah, W. Powell, and D. Blei, "Nonparametric density estimation for stochastic optimization with an observable state variable," *Advances in Neural Information Processing Systems*, vol. 23, pp. 820–828, 2010.

[37] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[38] G. S. Watson, "Smooth regression analysis," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[39] J. Huber, S. Müller, M. Fleischmann, and H. Stuckenschmidt, "A data-driven newsvendor problem: From data to decision," *European Journal of Operational Research*, vol. 278, no. 3, pp. 904–915, 2019.

[40] S. Punia, S. P. Singh, and J. K. Madaan, "From predictive to prescriptive analytics: A data-driven multi-item newsvendor model," *Decision Support Systems*, vol. 136, p. 113340, 2020.

[41] G.-Y. Ban, J. Gallien, and A. J. Mersereau, "Dynamic procurement of new products with covariate information: The residual tree method," *Manufacturing & Service Operations Management*, vol. 21, no. 4, pp. 798–815, 2019.

[42] H. Heitsch and W. Römisch, "Scenario tree modeling for multistage stochastic programs," *Mathematical Programming*, vol. 118, no. 2, pp. 371–406, 2009.

[43] D. Bertsimas and N. Kallus, "From predictive to prescriptive analytics," *arXiv preprint arXiv:1402.5481*, 2014.

[44] X. Yu, Z. Qi, and Y. Zhao, "Support vector regression for newspaper/magazine sales forecasting," *Procedia Computer Science*, vol. 17, pp. 1055–1062, 2013.

[45] R. Carbonneau, K. Laframboise, and R. Vahidov, "Application of machine learning techniques for supply chain demand forecasting," *European Journal of Operational Research*, vol. 184, no. 3, pp. 1140–1154, 2008.

[46] Ö. G. Ali and K. Yaman, "Selecting rows and columns for training support vector regression models with large retail datasets," *European Journal of Operational Research*, vol. 226, no. 3, pp. 471–480, 2013.

[47] M. E. Schweitzer and G. P. Cachon, "Decision bias in the newsvendor problem with a known demand distribution: Experimental evidence," *Management Science*, vol. 46, no. 3, pp. 404–420, 2000.

[48] R. T. Rockafellar, S. Uryasev, *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[49] J.-y. Gotoh and Y. Takano, "Newsvendor solutions via conditional value-at-risk minimization," *European Journal of Operational Research*, vol. 179, no. 1, pp. 80–96, 2007.

[50] Y.-j. Zhou, X.-h. Chen, and Z.-r. Wang, "Optimal ordering quantities for multi-products with stochastic demand: Return-cvar model," *International Journal of Production Economics*, vol. 112, no. 2, pp. 782–795, 2008.

[51] A. P. Katariya, S. Cetinkaya, and E. Tekin, "On the comparison of risk-neutral and risk-averse newsvendor problems," *Journal of the Operational Research Society*, vol. 65, no. 7, pp. 1090–1107, 2014.

[52] X. Xu, C. K. Chan, and A. Langevin, "Coping with risk management and fill rate in the loss-averse newsvendor model," *International Journal of Production Economics*, vol. 195, pp. 296–310, 2018.

[53] U. Murarka, V. Sinha, L. S. Thakur, and M. K. Tiwari, "Multiple criteria risk averse model for multi-product newsvendor problem using conditional value at risk constraints," *Information Sciences*, vol. 478, pp. 595–605, 2019.

[54] B. Sawik, "Multiobjective newsvendor models with cvar for flower industry," in *Applications of Management Science*, Emerald Publishing Limited, 2020.

[55] D. Bertsimas and J. Dunn, "Optimal classification trees," *Machine Learning*, vol. 106, no. 7, pp. 1039–1082, 2017.

[56] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013.

[57] J. Quinlan, "Program for machine learning," *C4. 5*, 1993.

[58] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[59] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[60] D. Dua and C. Graff, "Uci machine learning repository," 2017.

[61] A. N. Elmachtoub, J. C. N. Liang, and R. McNellis, "Decision trees for decision-making under the predict-then-optimize framework," *arXiv preprint arXiv:2003.00360*, 2020.

[62] S. J. Erlebacher, "Optimal and heuristic solutions for the multi-item newsvendor problem with a single capacity constraint," *Production and Operations Management*, vol. 9, no. 3, pp. 303–318, 2000.

[63] L. L. Abdel-Malek and R. Montanari, "An analysis of the multi-product newsboy problem with a budget constraint," *International Journal of Production Economics*, vol. 97, no. 3, pp. 296–307, 2005.

[64] D. Chen, S. L. Sain, and K. Guo, "Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining," *Journal of Database Marketing & Customer Strategy Management*, vol. 19, no. 3, pp. 197–208, 2012.

[65] D. Chen, S. L. Sain, and K. Guo, "UCI machine learning repository," 2012.

[66] D. Bertsimas and R. Weismantel, *Optimization over integers*, vol. 13. Dynamic Ideas Belmont, 2005.