The Use of Binary Choice Forests to Model and Estimate Discrete Choices

Ningyuan Chen*[1], Guillermo Gallego†[2], and Zhuodong Tang‡[2]

[1]Department of Management, University of Toronto Mississauga
[2]Department of Industrial Engineering & Decision Analytics
Hong Kong University of Science and Technology

Abstract

We show the equivalence of discrete choice models and the class of binary choice forests, which are random forests based on binary choice trees. This suggests that standard machine learning techniques based on random forests can serve to estimate discrete choice models with an interpretable output. This is confirmed by our data-driven theoretical results which show that random forests can predict the choice probability of any discrete choice model consistently, with its splitting criterion capable of recovering preference rank lists. The framework has unique advantages: it can capture behavioral patterns such as irrationality or sequential searches; it handles nonstandard formats of training data that result from aggregation; it can measure product importance based on how frequently a random customer would make decisions depending on the presence of the product; it can also incorporate price information and customer features. Our numerical results show that using random forests to estimate customer choices represented by binary choice forests can outperform the best parametric models in synthetic and real datasets.

*ningyuan.chen@utoronto.ca
†ggallego@ust.hk
‡ztangai@connect.ust.hk

1. Introduction

Being able to understand consumers' choice behavior when they are offered an assortment of products provides firms with unique advantages. It is particularly important in the modern era: online retailers that predict consumers' choice behavior more accurately can implement more effective strategies and earn higher profits. In turn, they can afford to invest in advanced technologies and infrastructure, and sharpen their prediction of consumers' behavior. The unstoppable cycle has created a few unprecedented market juggernauts such as Amazon. Firms unwilling or incapable of getting inside the mind of their consumers are left behind. Not surprisingly, discrete choice models (DCM) have become one of the central topics in revenue management and pricing analytics.

To understand and predict consumers' choice behavior, academics and practitioners have proposed several frameworks, some of which are widely adopted in industry, One ubiquitous framework is **model-then-estimate**. In this framework, a parametric DCM is proposed to explain how a customer chooses a product when offered an assortment. The parameters are then estimated using historical data. Once the model has been estimated properly, it can then be used as a workhorse to predict the choice behavior of future consumers.

In the model-then-estimate framework, there is a trade-off between the flexibility and accuracy. A flexible DCM incorporates a wide range of patterns of consumers' behavior, but it may be difficult to estimate and may overfit training data. A parsimonious model, may fail to capture consumers behavior, and even if estimated correctly it would be misspecified. The goal is to reach a delicate balance between flexibility and predictability even relative to assortments never seen before. Not surprisingly, it is not straightforward to find the "sweet spot" when selecting among the large class of parametric DCMs.

Another framework favored by data scientists is **estimate-without-models**. Advanced machine learning algorithms are applied to historical sales data, and used to predict future choice behavior. The framework skips "modeling" entirely and does not attempt to understand the rationality (or irrationality) hidden behind the patterns observed in the training data. With engineering tweaks, the algorithms can be implemented efficiently and capture a wide range of choice behavior. For example, neural networks are known to be able to approximate any continuous functions. This approach

may sound appealing: if the algorithm achieves impressive accuracy when predicting the choice behavior of new consumers, why do we care about the actual rationale behind consumers when they make choices? There are two reasons to care. First, the firm may be interested in not only making accurate predictions, but also other goals such as finding the optimal assortment that maximizes the expected revenue. Without a proper model, it is unclear if the goal can be formulated as an optimization problem. Second, when the market environment or customer preferences change systematically over time, having a reasonable model provide a certain degree of generalizability while black-box algorithms may fail to capture an obvious pattern just because the pattern has not appeared frequently in the past.

In this paper, we introduce a data-driven framework which we call **estimate-and-model** that combines machine learning with DCMs, and thus retains the strengths of both frameworks mentioned previously. The model we propose, binary choice forests, is a mixture of binary trees, each of which mimics the internal decision-making process of a customer. We show that the binary choice forest can be used to approximate *any* DCM, and is thus sufficiently flexible, but still identifiable with training data of reasonable size. Moreover, it can be efficiently estimated using random forests (Breiman, 2001), a popular machine learning technique that has stood the test of time. Random forests are easy to implement using R or Python (Pedregosa et al., 2011; Liaw and Wiener, 2002) and have been shown to have extraordinary predictive power in practice. We provide theoretical guarantees: as the sample size increases, random forests can successfully recover the binary choice forest, and thus *any* DCM. Moreover, the splitting criterion used by the random forests is intrinsically connected to the preference rank list of customers.

As a contribution to the literature, the framework we propose has the following practical advantages:

- It can capture various patterns of customer behavior that cannot be easily captured by other models, such as irregularity and sequential searches (Weitzman, 1979). See Section 5.1 for more details.
- It can deal with nonstandard formats of historical data, which is a major challenge in practice. See Section 5.2 for more details.
- It can return an importance index for all products, based on how frequently

a random customer would make decisions depending on the presence of the product.

- It can incorporate the prices of the products and reflect the information in the decision-making of consumers.
- It can naturally incorporate customer features and is compatible with personalized online retailing.

1.1. Literature Review

We first review DCMs proposed in the literature following the model-then-estimate framework, in the order of increasing flexibility and difficulty in terms of estimation. The independent demand model and the MNL model (McFadden, 1973) have very few parameters (one per product), which are easy to estimate (Train, 2009). Although the MNL model is still widely used, its inherent property of independence of irrelevant alternatives (IIA) has been criticized for being unrealistic (see Anderson et al. (1992) for more details). The mixed logit model, the nested logit model, the Markov chain DCM, and the rank-based DCM (see, e.g., Williams (1977); Train (2009); Farias et al. (2013); Blanchet et al. (2016)) are able to capture much more complex choice behavior than the MNL model. In fact, the mixed logit model and the rank-based DCM can approximate any random utility model (RUM), encompassing a very general class of DCMs. The estimation of these models is challenging, but there has been exciting progress in recent years (Farias et al., 2013; van Ryzin and Vulcano, 2014, 2017; Şimşek and Topaloglu, 2018; Jagabathula et al., 2019). However, the computational feasibility and the susceptibility to overfitting remain a challenge in practice. Even the general class of RUM cannot capture certain choice behavioral. A RUM possesses the so-called regularity property: the probability of choosing an alternative cannot increase if the offered set is enlarged. There are a few experimental studies showing strong evidence that regularity may be violated (Simonson and Tversky, 1992). Several models are proposed to capture even more general behavior than RUM (Natarajan et al., 2009; Flores et al., 2017; Berbeglia, 2019; Feng et al., 2017). It is unclear if the estimation for such models can be done efficiently.

The specifications of random forests used in this paper are introduced by Breiman

(2001), although many of the ideas were discovered even earlier. The readers may refer to Hastie et al. (2009) for a general introduction. Although random forests have been very successful in practice, little is known about their theoretical properties. To date, most studies are focused on isolated setups or simplified versions of the procedure. In a recent study, Scornet et al. (2015) establish the consistency of random forests in regression problems, under less restrictive assumptions. Biau and Scornet (2016) provide an excellent survey of the recent theoretical and methodological developments in the field.

A recent paper by Chen and Mišić (2019) proposes a similar tree-based DCM. They show that their "decision forest" can approximate any DCMs with arbitrary precision; a similar result is proved with a different approach in this paper. Our studies differ substantially in the estimation step: we focus on random forests, while Chen and Mišić (2019) follow an optimization approach based on column generation ideas for estimation. Moreover, we establish the consistency of random forests, and show that the estimation can accommodate the price information and aggregate choice data. In our numerical study, we find that random forests are quite robust and have a good performance even compared with the Markov chain model estimated using the expectation-maximization (EM) algorithm, which has been shown to have outstanding empirical performance compared to MNL, the nested logit, the mixed logit and rank-based DCM (Berbeglia et al., 2018), especially when the training data is large. Our algorithm runs 17 times faster than the EM algorithm. In contrast, the computational study in Chen and Mišić (2019) is limited to the rank-based model estimated by column generation (van Ryzin and Vulcano, 2014), which is shown to be outperformed by the Markov chain model (Berbeglia et al., 2018).

2. Choice Models and Mixture of Binary Trees

Consider a set $[N] \triangleq \{1, \dots, N\}$ of N products and define $[N]_+ \triangleq [N] \cup \{0\}$ where 0 represents the no-purchase option. Let $\boldsymbol{x} \in \{0,1\}^N$ be a binary vector representing an assortment of products, where $\boldsymbol{x}(i) = 1$ indicates product i is in the assortment and $\boldsymbol{x}(i) = 0$ otherwise. A **discrete choice model** (DCM) is a non-negative mapping

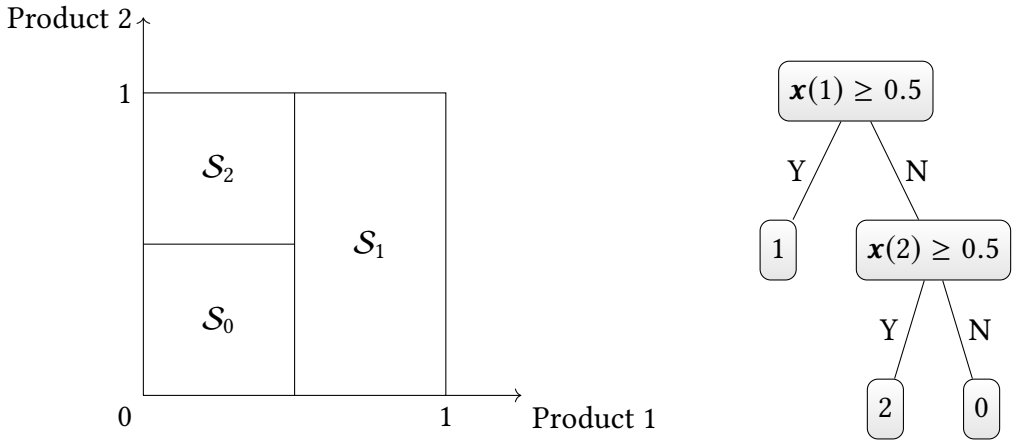

Figure 1: A binary tree representation of the partition.

$p(i, \boldsymbol{x}) \triangleq [N]_+ \times \{0,1\}^N \mapsto [0,1]$ such that

$$\sum_{i \in [N]_+} p(i, \boldsymbol{x}) = 1, \quad p(i, \boldsymbol{x}) = 0 \quad \text{if} \quad \boldsymbol{x}(i) = 0.$$

It is clear that $p(i, \boldsymbol{x})$ represents the probability of a random customer choosing product i when presented the assortment $\boldsymbol{x}$. We refer to a subset S of $[N]$ as an assortment associated with $\boldsymbol{x} \in \{0,1\}^N$, i.e., $i \in S$ if and only if $\boldsymbol{x}(i) = 1$. Without ambiguity, we will write $p(i, S)$ instead of $p(i, \boldsymbol{x})$.

A **binary decision tree** $t(\boldsymbol{x})$ maps $\boldsymbol{x} \in \{0,1\}^N$ into $[N]_+$. More precisely, it specifies a partition of the space $\{0,1\}^N$, $\{\mathcal{S}_i, i \in [N]_+\}$, and assigns label $i \in [N]_+$ to region $\mathcal{S}_i$, so $t(\boldsymbol{x}) = \sum_{i \in [N]_+} i \cdot \mathbb{I}_{\{\boldsymbol{x} \in \mathcal{S}_i\}}$. Some of the regions in the partition may be empty. We refer to the partition as a binary decision tree because any partition of $\{0,1\}^N$ can be obtained by sequentially splitting the space along N dimensions. For example, a decision tree representation of a partition when $N = 2$ is demonstrated in Figure 1.

A **binary decision forest** is defined as a convex combination of multiple binary decision trees. More precisely, a binary decision forest can be written as

$$f(i, \boldsymbol{x}) = \sum_{b=1}^{B} w_b \mathbb{I}_{\{t_b(\boldsymbol{x}) = i\}}$$

where the $t_b(\boldsymbol{x})$ and w_b are, respectively decision trees, and non-negative weights

summing up to one. Notice that a decision forest maps $[N]_+ \times \{0,1\}^N \mapsto [0,1]$ just like DCMs do. Yet decision forest are not necessarily DCMs because $t_b(\boldsymbol{x})$ may be equal to i even if $\boldsymbol{x}(i) = 0$.

A binary decision tree $t(\boldsymbol{x})$ is a **binary choice tree** if $t(\boldsymbol{x}) = i$ only if $\boldsymbol{x}(i) = 1$. A binary decision forest is a **binary choice forest** (BCF) if it is a convex combination of binary choice trees. A BCF can be interpreted as decisions made by B consumer types, with consumers of type b having weight w_b and making decisions based on binary choice tree $t_b(\boldsymbol{x})$. If $f(i, \boldsymbol{x})$ is a BCF, then f is also a DCM. This is because f is non-negative, $\sum_{i \in [N]_+} f(i, \boldsymbol{x}) = 1$ and $f(i, \boldsymbol{x}) = 0$ if $\boldsymbol{x}(i) = 0$. To see that the converse is also true, we will first show that DCMs are closed under convex combinations and that any DCM is in the convex hull of extreme DCMs. We next argue that the extreme DCMs are the deterministic DCMs that assign S to a particular choice $i(S) \in S_+$ with probability one for every $S \subset [N]$. The next step is to show that each extreme DCM can be represented by a binary choice tree concluding that every DCM is a convex combination of choice trees and is thus a BCF.

Theorem 1. *Every BCF is a DCM, and every DCM can be represented as a BCF.*

One way to interpret this result is that for each DCM there exists a set of weights $w_e, e \in E$ adding to one, such that $p(i, S) = \sum_{e \in E} w_e p_e(i, S)$ for all $i \in S_+$, $S \subset N$, where the p_e's are the extreme deterministic DCMs.

Although we can represent every DCM as a BCF, it will be difficult to estimate if we have too many extreme points. The number of extreme points is $\prod_{k=1}^{N} (k+1)^{\binom{N}{k}}$ for N products, which increases tremendously as N increases, with more than 6.7×10^{173} extreme points for $N = 8$. In the next theorem, we will show that any DCM can be represented as a convex combination of much fewer binary choice trees.

Theorem 2. *Every DCM can be represented as a convex combination of a BCF containing at most $N \cdot 2^{N-1} + 1$ trees.*

Proof. Carathéodory's theorem states that if a point x of $\mathbb{R}^d$ lies in the convex hull of a set P, then x can be written as the convex combination of at most $d + 1$ points in P. To apply Carathéodory's theorem to DCM, notice that since the choice probabilities sum to 1, each assortment with cardinality k has dimension of k. We have $d = \sum_{k=1}^{N} k\binom{N}{k} = \sum_{k=1}^{N} k \cdot \frac{N!}{k!(N-k)!} = N \cdot \sum_{k=1}^{N} \frac{(N-1)!}{(k-1)!(N-k)!} = N \cdot 2^{N-1}$. Therefore, Every DCM can be represented as a convex combination of a BCF containing at most $N \cdot 2^{N-1} + 1$. ∎

As an example, for $N = 8$, $d = 1024$, so any DCM with $N = 8$ can be represented by a convex combination of 1025 trees.

A recent working paper by Chen and Mišić (2019) has independently shown, by construction, that any choice model can be represented by a decision forest where each of the trees has depth $N + 1$. While their proof has the virtue of being constructive, our proof is more succinct and insightful as it shows that DCMs and BCFs are equivalent, and the existence of a solution of much lower dimension. Our result implies that choice forests are capable of explaining some of the pathological cases that do not exhibit regularity and are outside the RUM, including the decoy effect (Ariely, 2008) and the comparison-based choice (Huber et al., 1982). Note also that all RUMs can be modelled as convex combinations of permutation lists, which are special cases of decision trees.

3. Data and Estimation

The main goal of this paper is to provide a practical method to estimate DCMs using random forests, which are shown to be able to approximate all BCFs. The numerical recipe for random forests is widely available and implementable. Before proceeding we remark that an alternative approach would be to use column generation starting with a collection of trees and adding additional trees to improve the fit to data. This approach has been taken, for example by van Ryzin and Vulcano (2014); Mišić (2016); Jagabathula and Rusmevichientong (2016) to estimate RUMs by weighted preference rank lists, and a similar approach has been pursued by Chen and Mišić (2019) for trees. We remark that the output of our model can be fed into a column generation algorithm to seek further improvements although we have not pursued this in our paper.

We will assume that arriving consumers make selections independently based on an unknown DCM $p(i, \boldsymbol{x})$, and that a firm collects data of the form $(i_t, \boldsymbol{x}_t)$ (or equivalently (i_t, S_t)) where $\boldsymbol{x}_t$ was the assortment offered to the tth consumer and $i_t \in S_t \cup \{0\}$ is the choice made by consumer $t = 1, \ldots, T$. Our goal is to use the data to construct a family of binary choice trees as a means to estimate the underlying DCM $p(i, \boldsymbol{x})$ represented by a BCF. We view the problem as a classification problem: given the predictor $\boldsymbol{x}$, we would like to provide a classifier that maps the predictor to a class label $i \in [N]_+$, or the class probabilities.

To this end we will use a random forest as a classifier. The output of a random forest

is B individual binary decision trees (CART), $\{t_b(\boldsymbol{x})\}_{b=1}^{B}$, where B is a tunable parameter. Although a single tree only outputs a class label in each region, the aggregation of the trees, i.e., the forest, is naturally equipped with the class probabilities. Then the choice probability of item i in the assortment $\boldsymbol{x}$ is estimated as

$$\sum_{b=1}^{B} \frac{1}{B} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}}, \tag{1}$$

which is a special form of BCF. The next result shows that the random forest can still approximate any DCM.

Theorem 3. *If B is sufficiently large, then a binary choice forest of the form*

$$f(i, \boldsymbol{x}) = \sum_{b=1}^{B} \frac{1}{B} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}},$$

can approximate any DCM.

The implication of this result is that we don't have to worry about generating all of the extreme points, or deterministic DCMs, and then finding a set of weights w_b for each such tree $t_b(\boldsymbol{x})$. Intuitively, if B is sufficiently large, then we need approximately Bw_b type b customers associated with tree t_b with positive weight $w_b > 0$ in the convex combination.

We explain how the random forest can be estimated from the historical data by first reviewing the basic mechanism of CART which preforms recursive binary splitting of the predictor space $[0, 1]^N$. In each iteration, it selects a dimension $i \in [N]$ and a split point to split the predictor space. More precisely, the split (i, s_i) divides the observations to $\{(i_t, \boldsymbol{x}_t) : \boldsymbol{x}_t(i) \leq s_i\}$ and $\{(i_t, \boldsymbol{x}_t) : \boldsymbol{x}_t(i) > s_i\}$. In our problem, because $\boldsymbol{x}_t \in \{0, 1\}^N$ is at the corner of the hypercube, all split points between 0 and 1 create the same partition of the observations and thus we simply set $s_i \equiv 0.5$. To select the dimension, usually an empirical criterion is optimized to favor splits that create "purer" regions. That is, the resulting region should contain data points that mostly belong to the same class. We use a common measure called *Gini index*: $\sum_{R_j} \frac{t_j}{T} \sum_{k=0}^{N} \hat{p}_{jk}(1 - \hat{p}_{jk})$ where t_j is the number of observations in region R_j of the partition and $\hat{p}_{jk}$ is the empirical frequency of class k in R_j. It is not hard to see that the Gini index takes smaller values when the regions

contain predominantly observations from a single class. In this case, a dimension is selected that minimizes the measures and the partition is further refined by a binary split. This splitting operation is conducted recursively for the regions in the resulting partition until a stopping rule is met.

The main drawback of CART is its tendency to overfitting the training data. If a deep decision tree is built (having a large number of splits), then it may fit the training data well but introduce large variances when applied to test data. If the tree is pruned and only has a few leaves (or regions in the predictor space), then it loses the predictive accuracy. Random forests, by creating a number of decision trees and then aggregating them, significantly improve the power of single trees and moves the bias-variance trade-off toward the favorable direction. The basically idea behind random forests is to "shake" the original training data in various ways in order to create decision trees that are as uncorrelated as possible. Because the decision trees are deliberately "decorrelated", they can afford to be deep, as the large variances are remedied by aggregating the "almost independent" trees.

Next we explain the details of random forests. To create B randomized trees, for each $b = 1, \ldots, B$, we randomly choose z samples with replacement from the T observations (a bootstrap sample). Only the sub-sample of z observations is used to train the bth decision tree. Splits are performed only on a random subset of $[N]$ of size m according to one of the criterion of Gini index. The random sub-sample of training data and random directions to split are two key ingredients in creating less correlated decision trees in the random forest. The depth of the tree is controlled by the minimal number of observations, say l, in a region for the tree to keep splitting.

These ideas are subsumed in Algorithm 1. We first remark on the procedure in Algorithm 1 that can be applied to a generic classification problem and then comment on the special properties in our problem. (1) Many machine learning algorithms such as neural networks have numerous parameters to tune and the performance crucially depends on a suitable choice of parameters. Random forests, on the other hand, have only a few interpretable parameters. Even so, in the numerical studies in this paper, we simply choose a set of parameters that are commonly used for classification problems, without cross-validation or tuning, in order to demonstrate the robustness of the algorithm. In particularly, we mostly use $z = T$, $m = \sqrt{N}$ and $l = 50$. There are other alternative options when constructing random forests, such as using a bootstrap

Algorithm 1 Random forests for DCM estimation

Data: $\{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}$
Tunable parameters: number of trees B, sub-sample size $z \in \{1, \ldots, T\}$, number of dimensions to split $m \in \{1, \ldots, N\}$, terminal leaf size $l \in \{1, \ldots, z\}$
for $b = 1$ to B **do**
    Select z observations from the training data with replacement, denoted by Z
    Initialize the tree $t_b(\boldsymbol{x}) \equiv 0$ with a single root node
    while some leaf has greater than or equal to l observations belonging to Z and can be split **do**
        Select m variables without replacement among $\{1, \ldots, N\}$
        Select the optimal one to split among the m dimensions that minimizes the Gini index
        Split the leaf node into two
    end while
    Denote the partition corresponding to the leaves of the tree by $\{R_1, \ldots, R_M\}$; let c_i be the class label of a randomly chosen observation in R_i
    Define $t_b(\boldsymbol{x}) = \sum_{i=1}^{M} c_i \mathbb{I}_{\{\boldsymbol{x} \in R_i\}}$
end for
The trees $\{t_b(\cdot)\}_{b=1}^{B}$ are used to estimate the class probabilities as (1)

sample Step 4. For the ease of exposition, we stick to the canonical version presented in Algorithm 1. (2) The numerical recipe for the algorithm is implemented in many programming languages such as R and Python and ready to use. In Section B, we provide a demonstration using scikit-learn, a popular machine learning package in Python that implements random forests, to estimate customer choice. As one can see, it takes less than 20 lines to implement the procedure.

Because of the structure of the problem, there are three specific observations. (1) Because the entries of $\boldsymbol{x}$ are binary $\{0, 1\}$, the split position of decision trees is always 0.5. Therefore, along a branch of a decision tree, there can be at most one split on a particular dimension, and the depth of a decision tree is at most N. (2) The random forest is a binary decision forest instead of a BCF. In particular, the probability of class i, or the choice probability of product i given assortment $\boldsymbol{x}$, may be positive even when $\boldsymbol{x}(i) = 0$, i.e., product i is not included in the assortment. To fix the issue, we adjust the

probability of class i by conditioning on the trees that output reasonable class labels:

$$\sum_{b=1}^{B} \frac{1}{\sum_{j:\boldsymbol{x}(j)=1} \sum_{b=1}^{B} \mathbb{I}_{\{t_b(\boldsymbol{x})=j\}}} \mathbb{I}_{\{t_b(\boldsymbol{x})=i, \boldsymbol{x}(i)=1\}}$$

(3) When returning the class label of a leaf note in a decision tree, we use a randomly chosen observation instead of taking a majority vote (Step 11 in Algorithm 1). While not being a typical choice, it seems crucial in deriving our consistency result (Theorem 4). Intuitively, unlike other classification problems in which the predictor has a continuous support, in our problem $\boldsymbol{x}_t$ are overlapping when an assortment is offered to multiple consumers in the data. A majority vote would favor the choice of product that most consumers make and ignore less attractive products. To correctly recover the choice probability from the data, we randomly choose an observation in the leaf (equivalently, randomly pick a customer t in the data who has been offered the same assortment), which is at least an unbiased estimator for the choice probability.

4. Why Do Random Forests Work Well?

Many machine learning algorithms have superb performances in practice, while very few theories can be spelt out on why it is the case. For example, for random forests, even consistency, one of the most fundamental properties a statistician would demand for any classic estimators, was only established recently for regression problems under restrictive assumptions (Scornet et al., 2015). The lack of theoretical understandings can worry practitioners when stakes are high and the failure may have harmful consequences. In this section, we attempt to answer the "why" question for our setting from two angles. We show that random forests are consistent for any DCM, and the way that random forests split (Gini index) can naturally help to recover the choice model when it can be represented by a tree.

4.1. Random Forests are Consistent for Any Choice Model

We now show that with enough data, random forests can recover the choice probability of *any* DCM. To obtain our theoretical results, we impose mild assumptions on how the data is generated.

Assumption 1. There is an underlying ground truth DCM from which all T consumers independently make selections from the offered assortments, generating data (i_t, S_t), $t = 1, \dots T$.

Note that the assumption only requires consumers to make choices independently. On the other hand, we focus on a fixed-design experiment, and the sequence of assortment offered $\boldsymbol{x}_t$ can be arbitrary. This is different from most consistency results of random forests in which random design is used (see (Biau and Scornet, 2016) for references), i.e., $\boldsymbol{x}_t$ are i.i.d. In our setting, the assortment is unlikely to be generated randomly, but chosen by the firm, either to maximize the revenue or explore customer preferences by A/B testing. Therefore, a fixed design probably reflects the reality more than a random design.

Since the consistency result requires the sample size $T \to \infty$, we use the subscript T to emphasize the fact that the parameters may be chosen based on T. For a given assortment $\boldsymbol{x}$, let $k_T(\boldsymbol{x}) \triangleq \sum_{t=1}^{T} \mathbb{I}_{\{\boldsymbol{x}_t = \boldsymbol{x}\}}$ be the number of consumers who see assortment $\boldsymbol{x}$. We are now ready to establish the consistency of random forests.

Theorem 4. *Suppose Assumption 1 holds, then for any $\boldsymbol{x}$ and i, if $\lim_{T\to\infty} k_T(\boldsymbol{x})/T > 0$, l_T is fixed, $z_T \to \infty$, $B_T \to \infty$, then the random forest is consistent:*

$$\lim_{T\to\infty} \mathsf{P}\left(\left|\sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} - p(i, \boldsymbol{x})\right| > \epsilon\right) = 0$$

for all $\epsilon > 0$.

According to Theorem 4, the random forest can accurately predict the choice probability of any DCM, given that the firm offers the assortment for many times. Practically, the result can guide us about the choice of parameters. In fact, we just need to generate many trees in the forest ($B_T \to \infty$), re-sample many observations in a decision tree ($z_T \to \infty$), and keep the terminal leaf small (l_T is fixed). The requirement is easily met by the choice of parameters in the remarks following Algorithm 1, i.e., $z = T$, $m = \sqrt{N}$ and $l = 50$. Theorem 4 guarantees a good performance of the random forest when the seller has collected a large dataset. This is a typical case in online retailing, especially in the era of "big data".

Random forests thus provide a novel *data-driven* approach to model customer choices. In particular, the model is first trained from data, and then used to interpret

the inherent thought process of consumers when they make purchases. By Theorem 4, when the historical data has a large sample size, the model can accurately predict how consumers make decisions in reality. This reflects the universality of the model. In this section, we provide concrete examples demonstrating several practical considerations that can hardly be captured by other DCMs and handled well by random forests.

4.2. Gini Index Recovers the Rank List

In Section 2, we have shown that any DCM can be represented by a combination of binary decision trees. Moreover, through numerous experiments, we have found out that random forests perform particularly well when the data is generated by DCMs that can be represented by a few binary decision trees. In this section, we further explore this connection by studying a concrete setting where the DCM is represented by a single regular decision tree. Without loss of generality, we assume that customers always prefer product i to $i+1$, for $i = 1, \ldots, N-1$, and product N to the no-purchase option. Equivalently, the DCM is a single rank list (the preferences of all customers form an ordered set). The following finite-sample result demonstrates that the rank list can be recovered from the random forest with high probability.

Theorem 5. *Suppose the actual DCM is a rank list and the assortments in the training data are sampled uniformly. The random forest algorithm with sub-sample size $z = T$ (without replacement), $m = N$, terminal leaf size $l = 1$ and $B = 1$ accurately predicts the choices of at least a fraction $1-\epsilon$ of all 2^N assortments with probability at least*

$$1 - \sum_{i=1}^{k} \left[13 \exp\left(-\frac{T}{164 \cdot 2^{i-1}} \right) + 10(N-i-1) \exp\left(-\frac{T}{113 \cdot 2^{i-1}} \right) \right]$$

where $k = \lceil \log_2 \frac{1}{\epsilon} \rceil$.

Since the bound scales exponentially in T, the predictive accuracy increases tremendously with size.

The proof of the theorem reveals an intrinsic connection between the Gini index and the recovery of the rank list. Consider the first split of the random forest in Step 8. We can show that, in expectation, if the first split is on product i, then the resulting

Gini index is

$$\frac{2}{3} - \frac{1}{3 \cdot 2^{2i-2}} - \frac{1}{3 \cdot 2^{2N-2}} + O(1/T).$$

In other words, if the data is generated without randomness (centered at the mean), then the first split would occur on product 1 because of the ordering of the Gini index when T is large. Therefore, for the data points falling into the right branch of the first split (having product 1 in the assortment), no more splits are needed as all customers would choose product 1 according to the rank list. Such a split correctly identifies roughly half of the assortments. In the proof, we control the randomness by concentration inequalities, and conduct similar computations for the second, third splits and so on. The proof reveals the following insight into why random forests may work well in practice: The Gini index criterion tends to find the products that are ranked high in the rank lists, because they create "purer" splits that lower Gini index. As a result, the topological structure of the decision trees trained in the random forest is likely to resemble that of the binary choice trees underlying the DCM generating the data.

5. Flexibility and Benefits of Random Forests

In this section we demonstrate the flexibility and benefit of using random forests to estimate the choice forest.

5.1. Behavioral Issues

Because of Theorem 3 and Theorem 4, random forests can be used to estimated any DCMs. For example, there is empirical evidence showing that behavioral considerations of consumers may distort their choice and thus violate regularity, e.g., the decoy effect (Ariely, 2008) and the comparison-based DCM (Huber et al., 1982; Russo and Dosher, 1983). It is already documented in Chen and Mišić (2019) that the decision forest can capture the decoy effect. In this section, we use the choice forest to model consumer search.

How consumers search to obtain new information when making purchases, is an important behavioral issue that is not monitored, or "unsupervised" in statistical terms, and hard to estimate by most models (for a few exceptions, see e.g. Wang and Sahin (2017)). Therefore, most DCMs abstract away those thought processes and only capture

the aggregate effect. Weitzman (1979) proposes a sequential search model with search costs. Prior to initiating the search consumers know only the distribution, say V_j of the net utility of product $j \in [N]$ and the cost c_j to learn the realization of V_j. Let z_j be the root of the equation $E[(V_j - z_j)^+] = c_j$ and sort the products in descending order of z_j. Weitzman shows that it is optimal to walk away without making any observations if the realized value of the no-purchase alternative, say $W_0 = V_0$ exceeds z_1. Otherwise c_1 is paid to observe V_1 is observed and $W_1 = \max(V_1, W_0)$ is computed. The process stops if W_1 exceeds z_2 and continued otherwise, stopping the first time, if ever, that $W_i > z_{i+1}$.

We next show that this search process can be represented by decision trees. Consider three products ($N = 3$). Suppose that the products are sorted so that $z_1 > z_2 > z_3 > 0$, and that the valuations of an arriving customer satisfy $v_2 > v_1 > v_3$. Hence the customer always searches in the order of product one $\longrightarrow$ product two $\longrightarrow$ product three. If in addition we suppose $v_2 > z_3 > v_1$, then the decision tree can be illustrated in Figure 2. For example, suppose products one and tree are offered. The customer first searches product one, because the reservation price of product one z_1 is the highest. The realized valuation of product one is, however, not satisfactory ($v_1 < z_3$). Hence the customer keeps on searching the product with the second highest reservation price in the assortment, which is product three (product two is skipped because it is not in the assortment). However, the search process results in an even lower valuation of product three $v_3 < v_1$. As a result, the customer recalls and chooses product one. Clearly, a customer with different realized valuations would conduct a different search process, and leads to a different decision tree.

5.2. Aggregated Choice Data

One of the most pressing practical challenges in data analytics is the quality of data. In Section 2, the historical data $\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T$ is probably the most structured and granular form of data one can hope to acquire. While most academic papers studying the estimation of DCMs assume this level of granularity, in practice it is frequent to see data in a more aggregate format. As an example, consider an airline offering three service classes E, T and Q of a flight where data is aggregated over a time window during which there may be changes to the assortment, and compiled from different sales channels. The company records information at certain time clicks as in Table 1. For each class, the

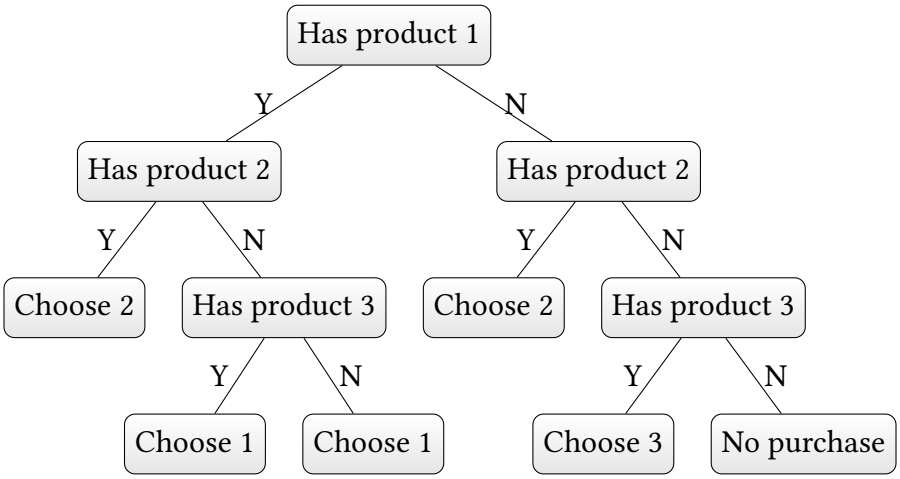

Figure 2: The sequential search process when $N = 3$ and the realized valuations and reservation prices satisfy $v_2 > v_1 > v_3$, $z_1 > z_2 > z_3 > 0$ and $v_2 > z_3 > v_1$.

Class	Closure percentage	# Booking
E	20%	2
T	0%	5
Q	90%	1

Table 1: A sample daily data of offered service classes and number of bookings.

closure percentage reflects the fraction of time that the class is not open for booking, i.e., included in the assortment. Thus, 100% would imply that the corresponding class is not offered during that the time window. In a retail setting, this helps to deal with products that sell-out between review periods. The number of bookings for each class is also recorded. There may be various reasons behind the aggregation of data. The managers may not realize the value of high-quality data or are unwilling to invest in the infrastructure and human resources to reform the data collection process. One of the author has encountered this situation in practice with aggregate datasets as in Table 1.

Fortunately, random forests can deal with aggregated choice data naturally, a feat that may be quite difficult to deal with with the column generation approach. Suppose the presented aggregated data has the form $\{(\boldsymbol{p}_s, \boldsymbol{b}_s)\}_{s=1}^{S}$, where $\boldsymbol{p}_s \in [0,1]^N$ denotes the closure percentage of the N products in day s, $\boldsymbol{b}_s \in \mathbb{Z}_+^{N+1}$ denotes the number of bookings[1], and the data spans S time windows. We transform the data into the desired form as follows: for each time window s, we create $D_s \triangleq \sum_{k=0}^{N} \boldsymbol{b}_s(k)$ observations, $\left\{(i_{s,j}, \boldsymbol{x}_{s,j})\right\}_{j=1}^{D_s}$. The predictor $\boldsymbol{x}_{s,j} \equiv \mathbf{1} - \boldsymbol{p}_s \in [0,1]^N$ and let $\boldsymbol{b}_s(k)$ of $i_{s,j}$ be valued k, for $k = 0, \ldots, N$.

To explain the intuition behind the data transformation, notice that we cannot tell from the data which assortment a customer faced when she made the booking. We simply take an *average* assortment that the customer may have faced, represented by $\mathbf{1} - \boldsymbol{p}_s$. In other words, if $1 - \boldsymbol{p}_s(j) \in [0,1]$ is large, then it implies that product j is offered most of the time during the day, and the transformation leads to the interpretation that consumers see a larger "fraction" of product j. As the closure percentage has a continuous impact on the eventual choice, it is reasonable to transform the predictors into a Euclidean space $[0,1]^N$, and build a smooth transition between the two ends $\boldsymbol{p}_s(j) = 0$ (the product is always offered) and $\boldsymbol{p}_s(j) = 1$ (the product is never offered).

The transformation creates a training dataset for classification with continuous predictors. The random forest can accommodate the data with minimal adaptation. In particular, all the steps in Algorithm 1 can be performed. The tree may have different structures: because the predictor $\boldsymbol{x}$ may not be at the corner of the unit hypercube any more, the split points may no longer be at 0.5.

[1]Again, we do not deal with demand censoring in this paper and assume that $\boldsymbol{b}_s$ has an additional dimension to record the number of consumers who do not book any class.

5.3. Product Importance

Random forests can be used to assign scores to each product and rank the importance of products. A common score, mean decrease impurity (MDI), is based on the total decrease in node impurity from splitting on the variable (product), averaged over all trees (Biau and Scornet, 2016). The score for product m is defined as

$$\begin{aligned}\mathrm{MDI}(m) = \frac{1}{B}\sum_{b=1}^{B} \sum_{\substack{\text{all splits } s \\ \text{in the } b\text{th tree}}} &(\text{fraction of data in the parent node of } s) \\ &\times (\text{reduction in the Gini index caused by } s) \times \mathbb{I}_{\{s \text{ splits on } m\}}.\end{aligned}$$

In other words, if consumers make decisions frequently based on the presence of product m (a lot of splits occur on product m), or their decisions are more consistent after observing the presence of product m (the Gini index is reduced significantly after splitting on m), then the product gains more score in MDI and regarded as important.

The identification of important products provides simple yet powerful insights into the behavioral patterns of consumers. Consider the following use cases: (1) An online retailer wants to promote its "flagship" products that significantly increase the conversion rate. By computing the MDI from the historical data, important products can be identified without extensive A/B testing. (2) Due to limited capacity, a firm plans to reduce the available types of products in order to cut costs. It could simply remove the products that have low sales according to the historical data. However, some products, while not looking attractive themselves, serve as decoys or references and boost the demand of other products. Removing these products would distort the choice behavior of consumers and may lead to unfavorable consequences. The importance score provides an ideal solution: if a product is ranked low based on MDI, then it does not strongly influence the decision making of consumers. It is therefore safe to leave them out. (3) When designing a new product, a firm attempts to decode the impact of various product features on customer choices. Which product feature is drawing most attentions? What do attractive products have in common? To conduct successful product engineering, first it needs to use the historical data to nail down a set of attractive products. Moreover, to quantify and separate out the contribution of various features, a numerical score of product importance is necessary. The importance

score is a more reasonable criterion than sales volume, because the latter cannot capture the synergy created between the products.

5.4. Incorporating Price Information

Besides the ease of estimation, the other benefit of a parametric DCM, such as the MNL or nested logit model, is the ability to account for covariates. For example, in the MNL model, the firm can estimate the price sensitivity of each product, and extrapolate/predict the choice probability when the product is charged a new price that has never been observed in the historical data. Many nonparametric DCMs cannot easily be extended to new prices. In this section, we show that while enjoying the benefit of a nonparametric formulation, random forests can also accommodate the price information.

Consider the data of the following format: $\{(i_t, \boldsymbol{p}_t)\}_{t=1}^{T}$, where $\boldsymbol{p}_t \in [0, +\infty]^N$ represent the prices of all products. For product j that is not included in the assortment offered to customer t, we set $\boldsymbol{p}_t(j) = +\infty$. This is because when a product is priced at $+\infty$, no customer would be willing to purchase it, and it is equivalent to the scenario that the product is not offered at all. Such view of equivalence is commonly adopted in the literature.[2] Therefore, compared to the binary vector $\boldsymbol{x}_t$ that only records whether a product is offered, the price vector $\boldsymbol{p}_t$ provides more information.

However, the predictor $\boldsymbol{p}$ can not be readily used in random forests. The predictor space $[0, +\infty]^N$ is unbounded, and the value $+\infty$ added to the extended real number line is not implementable in practice. To apply Algorithm 1, we introduce link functions that map the predictors into a compact set.

Definition 1. A function $g(\cdot) : [0, +\infty) \mapsto (0, 1]$ is referred to as a link function, if (1) $g(x)$ is strictly decreasing, (2) $g(0) = 1$, and (3) $\lim_{x\to+\infty} g(x) = 0$.

The link function can be used to transform a price $p \geq 0$ into $(0, 1]$. Moreover, because of property (3), we can naturally define $g(+\infty) = 0$. Thus, if product j is not included in assortment $\boldsymbol{x}_t$, then $g(\boldsymbol{p}_t(j)) = g(+\infty) = 0 = \boldsymbol{x}_t(j)$. If product j is offered at a very low price, then $g(\boldsymbol{p}_t(j)) \approx g(0) = 1$. After the transformation of predictors,

[2] One may argue that an assortment with a product having an artificially high price is not equivalent to the one without such a product, as the product may induce reference effects. We do not consider such behaviors here.

$\boldsymbol{p}_t \to g(\boldsymbol{p}_t)$[3], we introduce a continuous scale to the problem in Section 2. Instead of binary status (included or not), each product now has a spectrum of presence, depending on the price of the product. Now we can directly apply Algorithm 1 to the training data $\{(i_t, g(\boldsymbol{p}_t))\}_{t=1}^T$. As a result, we need to modify Step 7, because the algorithm needs to find not only the optimal dimension to split, but also the optimal split location. The slightly modified random forests are demonstrated in Algorithm 2. Because of the

Algorithm 2 Random forests for DCM estimation with price information

1: Data: $\{(i_t, \boldsymbol{p}_t)\}_{t=1}^T$
2: Tunable parameters: number of trees B, sub-sample size $z \in \{1, \ldots, T\}$, number of dimensions to split $m \in \{1, \ldots, N\}$, terminal leaf size $l \in \{1, \ldots, z\}$, a link function $g(\cdot)$
3: Transform the training data to $\{(i_t, g(\boldsymbol{p}_t))\}_{t=1}^T$
4: **for** $b = 1$ to B **do**
5: Select z observations from the training data with replacement, denoted by Z
6: Initialize the tree $t_b(g(\boldsymbol{p})) \equiv 0$ with a single root node
7: **while** some leaf has greater than or equal to l observations belonging to Z and can be split **do**
8: Select m variables without replacement among $\{1, \ldots, N\}$
9: Select the optimal one among the m dimensions and the optimal position to split that minimize the Gini index
10: Split the leaf node into two
11: **end while**
12: Denote the partition corresponding to the leaves of the tree by $\{R_1, \ldots, R_M\}$; let c_i be the class label of a randomly chosen observation in R_i
13: Define $t_b(g(\boldsymbol{p})) = \sum_{i=1}^M c_i \mathbb{I}_{\{g(\boldsymbol{p}) \in R_i\}}$
14: **end for**
15: The choice probability of product i given price vector $\boldsymbol{p}$ is $\sum_{b=1}^B \frac{1}{B} \mathbb{I}_{\{t_b(g(\boldsymbol{p}))=i\}}$

nature of the decision trees, the impact of prices on the choice behaviors is piecewise linear. For example, Figure 3 illustrates a possible decision tree with $N = 3$.

It is not surprising that there are numerous link functions to choose from. We give two examples below:

- $g(x) = e^{-x}$
- $g(x) = 1 - \frac{2}{\pi} \arctan(x)$

[3]When $g(\cdot)$ is applied to a vector $\boldsymbol{p}$, it is interpreted as applied to each component of the vector.

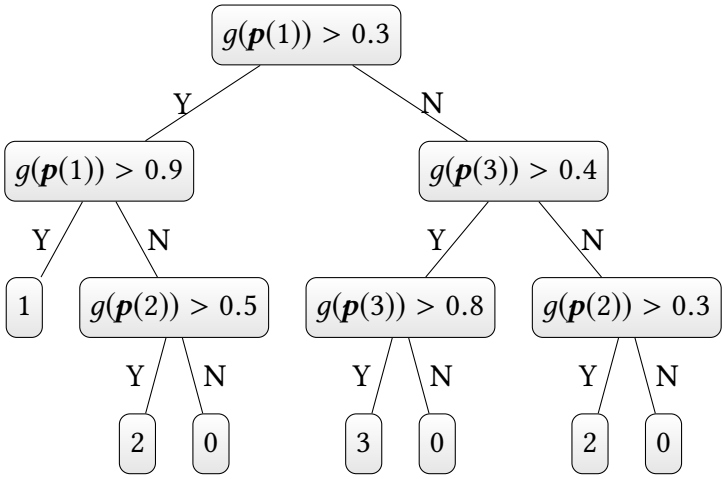

Figure 3: A possible decision tree when the price information is incorporated for $N = 3$. $g(\boldsymbol{p}(i)) > a$ is equivalent to $\boldsymbol{p}(i) < g^{-1}(a)$, i.e., product i is included in the assortment and its price is less than $g^{-1}(a)$.

In fact, the survival function of any non-negative random variables with positive PDF is a candidate for the link function. This extra degree of freedom may concern some academics and practitioners: How sensitive is the estimated DCM to the choice of link functions? What criteria may be used to pick a "good" link function? Our next result guarantees that the choice of link functions does not affect the estimated DCM. For any two link functions $g_1(x)$ and $g_2(x)$, we can run Algorithm 2 for training data $\{(i_t, g_1(\boldsymbol{p}_t))\}_{t=1}^T$ and $\{(i_t, g_2(\boldsymbol{p}_t))\}_{t=1}^T$. We use $t_b^{(j)}(\boldsymbol{x})$ to denote the returned bth tree of the algorithm for link function $g_j(x)$, $j = 1, 2$.

Proposition 1. *If we equalize*

- *the choice of parameters in Step 2 except for the link function*
- *the internal randomizers in Step 5, 8, and 12*

in Algorithm 2, then the trees of both link functions return the same class label for an observation in the training data: $t_b^{(1)}(g_1(\boldsymbol{p}_t)) = t_b^{(2)}(g_2(\boldsymbol{p}_t))$ *for all* $t = 1, \ldots, T$ *and* $b = 1, \ldots, B$.

It is worth pointing out that although the random forests using two link functions output identical class labels for $\boldsymbol{p}_t$ in the training data, they may differ for when predicting a new price vector $\boldsymbol{p}$. This is because the splitting operation that minimizes

the Gini index in Step 8 is not unique. Any split between two consecutive observations[4] results in an identical class composition in the new leaves and thus the same Gini index. Usually the algorithm picks the middle between two consecutive observations to split, which may differ for different link functions if they are not locally linear. Nevertheless, these cases are rare and Algorithm 2 is not sensitive to the choice of link functions.

The theoretical guarantee in the pricing setting, however, is far more involved than Section 2. The state-of-art theoretical guarantee of random forests is given by Scornet et al. (2015). The authors prove that random forests are consistent for the regression problem, under some mild assumptions. Their setup is the closest to the original algorithm proposed in Breiman (2001), while other papers have proved the consistency for random forests with simplified or special implementations. Our setup differs from Scornet et al. (2015) in that we are focusing on a classification problem. We can recast it into a regression problem by analyzing the class probability of a particular class. However, instead of the Gini index, the sum of squared errors is typically used in regression problems, and the analysis has to be modified substantially. We thus leave the theoretical guarantee for future research.

5.5. Incorporating Customer Features

A growing trend in online retailing and Ecommerce is personalization. Due to the increasing access to personal information and computation power, the retailer is able to device specific policies, including pricing or recommendation, for different customers based on his/her observed features. Personalization turns out to be hugely successful. Imagine an arriving customer being labeled as a college student. Then for a fashion retailer, it is a strong indicator that she/he may be interested in affordable brands. Leveraging personal information can greatly increase the garnered revenue of the firm.

To offer personalized assortment, the very first step is to incorporate the feature information into the choice model. One possible model, The mixed logit model assumes that the customers are categorized into discrete types, and customers of the same type behave homogeneously according to an independent MNL model. As one of the main drawbacks, it is not straightforward to connect continuous customers features

[4]If the algorithm splits on dimension m, then $\boldsymbol{p}_{t_1}$ and $\boldsymbol{p}_{t_2}$ are consecutive if there does not exist $\boldsymbol{p}_{t_3}$ in the same leaf node such that $(\boldsymbol{p}_{t_1}(m) - \boldsymbol{p}_{t_3}(m))(\boldsymbol{p}_{t_2}(m) - \boldsymbol{p}_{t_3}(m)) < 0$.

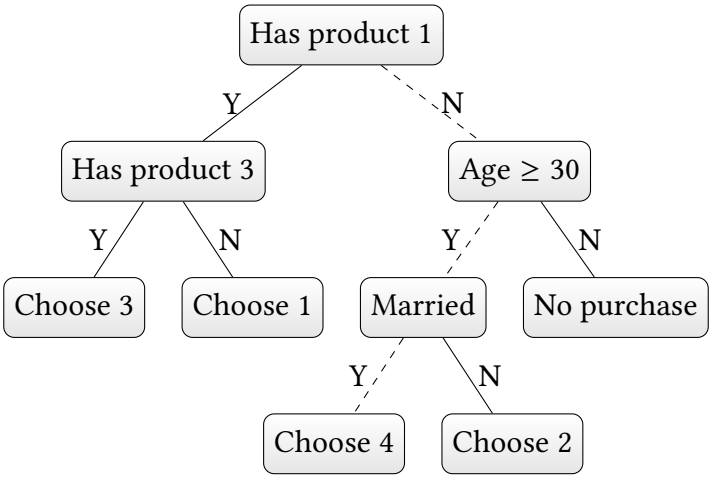

Figure 4: A possible binary choice tree after incorporating customer features.

to discrete types, and unsupervised learning algorithms may be needed. Recently, Bernstein et al. (2018) propose a dynamic Bayesian algorithm to address the issue. Another typical approach is built on the MNL model, while replacing the deterministic utility of a product by a linear function of the customer feature. For example, see Cheung and Simchi-Levi (2017) and references therein. In such personalized MNL models, the critics of the MNL model (such as IIA) persist.

In this section, we demonstrate that it is natural for random forests to capture customer features and return a binary choice forest that is aware of such information. Suppose the collected data of the firm have the form $(i_t, \boldsymbol{x}_t, \boldsymbol{f}_t)$ for customer t, where in addition to $(i_t, \boldsymbol{x}_t)$, the choice made and the offered set, the normalized customer feature $\boldsymbol{f}_t \in [0,1]^M$ is also recorded. The procedure in Section 3 can be extended naturally. In particular, we may append $\boldsymbol{f}_t$ to $\boldsymbol{x}_t$, so that the predictor $(\boldsymbol{x}, \boldsymbol{f}) \in [0,1]^{M+N}$. Algorithm 1 can be modified accordingly.

The resulting binary choice forest consists of B binary choice trees. The splits of the binary choice tree now encode not only whether a product is offered, but also predictive feature information of the customer. For example, a possible binary choice tree illustrated in Figure 4 may result from the algorithm. Compared with the current personalized choice models, the framework introduced in this paper has the following benefits:

- The estimation is straightforward (same as the algorithm without customer

features) and can be implemented efficiently.

- The nonparametric nature of the model allows to capture complex interaction between products and customer features, and among customer features. For example, "offering a high-end handbag" may become a strong predictor when the combination of features "female" and "age$\geq$ 30" are activated. In a binary choice tree, the effect is captured by three splits (one for the product and two for the customer features) along a branch. It is almost impossible to capture in a parametric (linear) model.

- The framework can be combined with aforementioned adjustments, such as pricing and product importance. For example, the measure MDI introduced in Section 5.3 can be used to identify predictive customer features.

6. Numerical Experiments

In this section, we conduct a comprehensive numerical study based on both synthetic and real datasets. We find that (1) random forests are quite robust and the performance does not vary much for underlying DCMs with different levels of complexity. In particular, random forests only underperform the correctly specified parametric models by a small margin and do not overfit; (2) the standard error of random forests are small compared to other estimation procedures; (3) random forests benefit tremendously from increasing sample size compared to other DCMs; (4) the computation time of random forests almost does not scale with the size of the training data; (5) random forests perform well even if the training set only includes 1/100 of all available assortments; (6) random forests handle training data with nonstandard format reasonably well, such as aggregated data and price information (see Section 5.2 and 5.4 for more details) which cannot be handled easily by other frameworks.

We will compare the estimation results of random forests with the MNL model (Train, 2009) and the Markov chain model (Blanchet et al., 2016)[5] for both synthetic

[5]The MNL model is estimated using MLE. The Markov chain model is estimated using the EM algorithm, the same as the implementation in Şimşek and Topaloglu (2018). The random forest is estimated using the Python package "scikit-learn". The implementation is slightly different in that scikit-learn outputs the empirical class probability rather than a random sample in Step 11. The difference is negligible when B is large.

and real data sets. We choose the MNL and the Markov Chain models as benchmarks because the MNL model is one of the most widely used DCM and the Markov chain model can flexibly approximate RUM ($O(N^2)$) and has been shown (Berbeglia et al., 2018) to have an outstanding empirical performance compared to MNL, the nested logit, the mixed logit, and rank-based DCM. Note that the actual DCM generating the training data is not necessarily one of the three models mentioned above.

When conducting numerical experiments, we set the hyper-parameters of the random forest as follows: $B = 1000$, $z = T$, $m = \sqrt{N}$, $l = 50$. Choosing the parameters optimally using cross validation would further improve the performance of random forest.

6.1. The Random Utility Model

We first investigate the performance of random forests when the training data is generated by RUM. The RUM includes a large class of DCMs. Consider $N = 10$ products. We generate the training set using the MNL model as the ground truth, where the expected utility of each product is generated from a standard normal distribution. Our training data consists of $\tilde{T} \in \{30, 75, 150, 300, 600\}$ periods. Each period contains a single assortment and 10 transactions so the total number of data points is $T = 10\tilde{T}$. This is following the setup of Berbeglia et al. (2018). We randomly generate an assortment in each period uniformly randomly among all assortments.

The performance is evaluated by root mean squared error, which is also used in Berbeglia et al. (2018):

$$RMSE\left(\mathsf{P}, \hat{\mathsf{P}}\right) = \sqrt{\frac{\sum_{S\subseteq[N]} \sum_{j\in S\cup\{0\}} \left(\mathsf{P}(j|S) - \hat{\mathsf{P}}(j|S)\right)^2}{\sum_{S\subseteq[N]}(|S|+1)}}, \tag{2}$$

where P denotes the actual choice probability and $\hat{\mathsf{P}}$ denotes the estimated choice probability. The RMSE tests all the assortments and there is no need to generate a test set. For each setting, we generate 100 independent training data sets and compute the average and standard deviation of the RMSEs. The result is shown in Table 2. Not surprisingly, MNL model performs the best among the three because it has very few parameters and correctly specifies the ground truth. With such a simple DCM, the

T	RF	MNL	Markov
300	0.084 (0.014)	0.030 (0.007)	0.062 (0.009)
750	0.061 (0.006)	0.019 (0.005)	0.042 (0.005)
1500	0.048 (0.005)	0.014 (0.003)	0.031 (0.004)
3000	0.041 (0.004)	0.009 (0.002)	0.023 (0.003)
6000	0.037 (0.002)	0.006 (0.002)	0.017 (0.002)

Table 2: The average and standard deviation of RMSE using random forests, the MNL and the Markov chain model when the training data is generated by the MNL model.

random forest does not overfit and only slightly underperforms the Markov chain model. As the data size increases, the RMSE of random forest converges to zero.

Next we use the rank-based model to generate the training data, which is shown to be equivalent to RUM (Block et al., 1959). Consider $N = 10$ products. Consumers are divided into $k = 4$ or $k = 10$ different types, each with a random preference permutation of all the products and the no-purchase alternative. For a given assortment of products, each type of consumer will purchase the product ranked the highest in her preference rank. If the no-purchase option is ranked higher than all the products in the assortment, then the customer does not purchase anything. We also randomly generate the fractions of customer types as follows: draw uniform random variables u_i between zero and one for $i = 1, ..., k$, and then set $\frac{u_i}{\sum_{j=1}^{k} u_j}$ to be the proportion of type i, $i = 1, ..., k$. The result is shown in Table 3. We can see that the MNL model underperforms and does not improve significantly as the data size increases, because of the misspecification error. The Markov chain model performs the best among the three. The performance of the random forest is quite robust, judged from the low standard deviation. Moreover, the performance improves dramatically as T increases; for $T = 20000$, the RMSE is smaller than the Markov chain model, which is shown in Berbeglia et al. (2018) to outperform other DCM estimators. Predicted by Theorem 4, the RMSE tends to zero when the training set is large.

We run our algorithm on iMac with 2.7GHz quad-core Inter Core i5 and 8GB RAM installed. The running time is shown in Table 4. In terms of computation time, both the MNL model and the random forest can be implemented efficiently, while the EM algorithm used to estimate the Markov chain model takes much longer. When $T = 20000$, the random forest spends 1/17 of the computation time of the Markov chain model.

T	$k = 4$		
	RF	MNL	Markov
300	0.115 (0.031)	0.121 (0.034)	0.078 (0.032)
750	0.090 (0.021)	0.118 (0.025)	0.058 (0.024)
1500	0.069 (0.016)	0.114 (0.029)	0.047 (0.020)
3000	0.056 (0.009)	0.118 (0.018)	0.044 (0.017)
6000	0.045 (0.006)	0.116 (0.021)	0.040 (0.017)
20000	0.034 (0.004)	0.115 (0.020)	0.037 (0.017)
	$k = 10$		
	RF	MNL	Markov
300	0.104 (0.013)	0.097 (0.016)	0.077 (0.016)
750	0.079 (0.009)	0.093 (0.012)	0.057 (0.009)
1500	0.065 (0.008)	0.091 (0.014)	0.048 (0.009)
3000	0.053 (0.005)	0.088 (0.013)	0.042 (0.008)
6000	0.046 (0.004)	0.088 (0.013)	0.040 (0.008)
20000	0.038 (0.003)	0.087 (0.014)	0.037 (0.009)

Table 3: The average and standard deviation of RMSE of random forests, the MNL and the Markov chain model when the training data is generated using the rank-based model.

T	RF	MNL	Markov
300	72.3s	0.7s	25.7s
750	72.5s	1.4s	36.1s
1500	72.3s	3.2s	113.7s
3000	74.0s	6.8s	203.0s
6000	74.5s	17.4s	445.2s
20000	81.8s	55.5s	1460.6s

Table 4: The average running time of Random forest, MNL and the Markov chain Model

Note that the running time of random forest almost does not increase for larger training set. This makes it useful when dealing with big data.

6.2. Generalizability to Unseen Assortments

One of the major challenges in the estimation of the DCM, compared to other statistical estimation problems, is the limited coverage of the training data, which strongly violates the i.i.d. assumption. In particular, the seller tends to offer a few assortments that they believe are profitable. As a result, in the training data $\{\boldsymbol{x}_t\}_{t=1}^{T}$ only makes up a small fraction of the total 2^N available assortments. Any estimation procedure needs to address the following issue: can the DCM estimated from a few assortments generalize to the assortments that have never been offered in the training data?

Next we show that random forests perform this task well: theoretically, random forests adaptively choose nearest neighbors, and the choice probability of an assortment can be generalized to "neighboring" assortments (those with one more or one less product), as long as the underlying DCM possesses a certain degree of continuity in terms of the offered set $\boldsymbol{x}$. Consider $N = 10$ products and $T = 6000$. We randomly choose $\tilde{T}$ assortments to offer in the training set and thus there are $6000/\tilde{T}$ transactions for each assortment. "Large" assortments refer to those with many products ($7 \leq |S| \leq 10$). The result is shown in Table 5. Note that there are $2^N - 1 = 1023$ possible available assortments. Therefore, for example, $\tilde{T} = 10$ implies that only $1/100$ of the total assortments have been offered in the training data. The RMSE is only two to three times larger than the case where most assortments have been offered $\tilde{T} = 600$. Moreover, a larger assortment helps the estimation of the DCM. When the actual DCM is the MNL

$\tilde{T}$	Rank-based $k = 4$	Rank-based $k = 10$	MNL
5	0.193 (0.064)	0.156 (0.034)	0.133 (0.041)
10	0.158 (0.034)	0.128 (0.026)	0.111 (0.035)
5 (large)	0.181 (0.056)	0.124 (0.028)	0.038 (0.017)
10 (large)	0.150 (0.047)	0.109 (0.027)	0.034 (0.014)
50	0.087 (0.025)	0.073 (0.014)	0.054 (0.008)
100	0.068 (0.014)	0.060 (0.007)	0.042 (0.004)
600	0.045 (0.006)	0.046 (0.004)	0.037 (0.002)

Table 5: The average and standard deviation of RMSE using random forests when there are a few assortments in the training data. The column represents different ground-truth models.

model, training random forests with 10 large assortments performs better than training with 600 randomly chosen assortments.

We also remark that the generalizability of random forests does not only depend on the estimator, but also the actual DCM. Some DCMs are more accessible to generalization to unseen assortments. It remains an exciting future research to formalize the statement and theoretically quantify the generalizability of a DCM to unseen data in the framework of random forests.

6.3. Behavioral Choice Models

When the DCM is outside the scope of RUM and the regularity is violated, the Markov chain and MNL model may fail to specify the choice behavior correctly. In this section, we generate choice data using the comparison-based DCM (Huber et al., 1982), described below. Consumers implicitly score various attributes of the products in the assortment. Then they undergo an internal round-robin tournament of all the products. When comparing two products from the assortment, the customer checks their attributes and count the number of preferable attributes of both products. Eventually, the customer count the total number of wins (preferable attributes) in the pairwise comparisons. Here we assume that customers choose with equal probability if there is a tie.

In the experiment, we consider $N = 10$ products. Consumers are divided into $k = 2$ different types, whose proportions are randomly generated between 0 and 1. Each type assigns uniform random variables between 0 and 1 to the five attributes of all the

T	RF	MNL	Markov
300	0.157 (0.031)	0.160 (0.033)	0.146 (0.038)
750	0.133 (0.025)	0.156 (0.030)	0.132 (0.036)
1500	0.112 (0.022)	0.152 (0.030)	0.123 (0.033)
3000	0.094 (0.021)	0.155 (0.030)	0.120 (0.037)
6000	0.079 (0.018)	0.152 (0.032)	0.120 (0.036)

Table 6: The average and standard deviation of RMSE using Random Forest, MNL and Markov chain Model under the comparison-based DCM

products (including the no-purchase option). Again we use the RMSE in (2) to compare the predictive accuracy. Like in the previous experiment, each setting is simulated 100 times. The result is shown in Table 6. Because of the irregularity, both the MNL and the Markov chain DCM are outperformed by the random forest, especially when the data size increases. Note that as $T \to \infty$, the random forest is able to achieve diminishing RMSE, while the other two models do not improve because of the misspecification error. Like the previous experiment, the random forest achieves stable performances with small standard deviations.

6.4. Aggregated Choice Data

In this section, we investigate the performance of random forests when the training data is aggregated as in Section 5.2. To generate the aggregated training data, we first generate T observations using the rank-based model for $N = 10$ products and $k = 10$ customer types, as in Section 6.1. The only difference is that we only simulate one instead of ten transactions for each offered assortment. Then, we let a be aggregation levels, i.e., we aggregate a data points together. For example, $a = 1$ is equivalent to the original data. For $a = 5$, Table 7 illustrates five observations in the original data set for $n = 5$. Upon aggregation, the five transactions are replaced by five new observations with $\boldsymbol{x}_t \equiv [0.6, 0.4, 0.8, 0.4, 0.6]$ and $i_t = 1, 0, 4, 3, 1$ for $t = 1, 2, 3, 4, 5$.

We test the performance for different sizes of the training set $T \in \{500, 5000, 50000\}$ and different aggregate levels $a \in \{1, 5, 10, 50, 100\}$. The performance is measured in RMSE. We simulate 100 instances for each setting to evaluate the average and standard deviation, shown in Table 8. From the results, random forests handle aggregate data

Product 1	Product 2	Product 3	Product 4	Product 5	Choices
1	1	1	1	1	1
0	1	0	0	1	0
1	0	1	1	1	4
0	0	1	0	0	3
1	0	1	0	0	1

Table 7: Five observations in the unaggregated original data. Upon aggregation, they are replaced by five new observations with $\boldsymbol{x}_t \equiv [0.6, 0.4, 0.8, 0.4, 0.6]$ and $i_t = 1, 0, 4, 3, 1$ for $t = 1, 2, 3, 4, 5$.

T	$a = 1$	$a = 5$	$a = 10$	$a = 50$	$a = 100$
500	0.082 (0.009)	0.109 (0.016)	0.114 (0.016)	0.119 (0.015)	0.120 (0.015)
5000	0.047 (0.004)	0.085 (0.010)	0.097 (0.012)	0.111 (0.013)	0.114 (0.013)
50000	0.039 (0.002)	0.068 (0.009)	0.082 (0.011)	0.103 (0.013)	0.108 (0.013)

Table 8: Random Forest performance for different aggregate levels

relatively well. Even with aggregation level $a = 100$, the RMSE does not seem to deteriorate significantly. Note that no other DCMs can handle aggregate data to the best of our knowledge, so no benchmark can be provided in this case.

6.5. Incorporating Pricing Information

In this section, we test the performance of random forests when the price information is incorporated. This is a unique feature of random forests as most DCMs can't estimate the choice probability efficiently with prices.

We use the MNL model to generate the choice data. Let $\boldsymbol{u}$ denote the expected utility of the products and $\boldsymbol{p}$ their prices. Therefore, for given assortment S, the choice probabilities of product $i \in S$ and the no-purchase option are:

$$\mathsf{P}_i = \frac{\exp(u_i - p_i)}{1 + \sum_{j \in S} \exp(u_j - p_j)}, \quad \mathsf{P}_0 = \frac{1}{1 + \sum_{j \in S} \exp(u_j - p_j)}. \tag{3}$$

Consider $N = 10$ products. We generate u_i as uniform random variables between 0 and 1 for each product. For each observation, we first randomly generate an assortment as Section 6.1. Then we generate a price for each product in the assortment as the

T	RMSE
500	0.067 (0.008)
5000	0.040 (0.002)
50000	0.035 (0.002)

Table 9: The RMSE of random forests with price information

absolute value of a standard normal random variable. As explained in Section 5.4, we use the link function $g(x) = \exp(-x)$. The customer's choice then follows the choice probability (3).

The RMSE in (2) is no longer applicable because the assortments and prices cannot be exhausted. To evaluate the performance, we randomly generate $N = 1000$ assortments and prices according to the same distribution as the training data. Then we evaluate the RMSE as follows:

$$RMSE\left(\mathsf{P}, \hat{\mathsf{P}}\right) = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j \in S_i \cup \{0\}} \left(\mathsf{P}(j|S_i) - \hat{\mathsf{P}}(j|S_i)\right)^2}{\sum_{i=1}^{N} (|S_i| + 1)}}, \tag{4}$$

where P denotes the actual choice probability, and $\hat{\mathsf{P}}$ denotes the estimation. We investigate the performance of the random forest for different sizes of training data $T \in \{500, 5000, 50000\}$. The result is shown in Table 9.

The result confirms that random forests can tackle price information well. Although we do not have benchmarks, the RMSE is comparable to the previous experiments, e.g., Table 8.

6.6. Real Data: Hotel

In this section we apply the random forest algorithm to a public dataset based on Bodea et al. (2009). The dataset includes transient customers (mostly from business travelers) who stayed in one of five continental U.S. hotels between March 12, 2007, and April 15, 2007. The minimum booking horizon for each check-in date is four weeks. Rate and room type availability and reservation information are collected via the hotel and/or customer relationship officers (CROs), the hotel's websites, and offline travel agencies. Since there is no direct competition among these five hotels, we will process the data

separately. A product is uniquely defined by the room type (e.g. Suite 1, 2 Double Beds Room 1, etc). For each transaction, the purchased room type and the assortment offered are recorded.

When processing the dataset, we have removed the product that has less than 10 transactions. We also removed the transactions whose offered assortments are not available due to technical reasons. For the transactions that none of the products in the available sets are purchased by the customer, we assume customers choose the no-purchase alternative. We do not add dummy transactions with no-purchases to uncensor the data like van Ryzin and Vulcano (2014), Şimşek and Topaloglu (2018) and Berbeglia et al. (2018).

To compare different estimation procedures, we use five-fold cross validation to examine the out-of-sample performance. Because we no longer know the actual choice model that generates the data, after estimating the model in the training set, we follow Berbeglia et al. (2018) and evaluate the "empirical" version of the RMSE in the validation set. That is, letting $\mathcal{T}$ be the validation set, we define

$$RMSE\left(\hat{\mathrm{P}}, \mathcal{T}\right) = \sqrt{\frac{\sum_{(i,S)\in\mathcal{T}} \sum_{j\in S\cup\{0\}} \left(\mathbb{I}_{\{j=i\}} - \hat{\mathrm{P}}(j|S)\right)^2}{\sum_{(i,S)\in\mathcal{T}} (|S|+1)}}. \tag{5}$$

In Table 10 we show the scale of the five datasets after preprocessing. We show the out-of-sample RMSE data for each hotel (average and standard deviation). In addition, we also show the performance of the independent demand model (ID), which does not incorporate the substitution effect and is expected to perform poorly, in order to provide a lower bound of the performance.

Consistent with the insights drawn from the synthetic data, random forest outperforms the parametric methods for larger dataset (Hotel 1, 2 and 3). For smaller data size (Hotel 4 and 5), random forest is on a par with the best parametric estimation procedure (Markov) according to Berbeglia et al. (2018).

6.7. Real Data: IRI Academic Dataset

In this section we compare several algorithms on the IRI Academic Dataset (Bronnenberg et al., 2008). The IRI Academic Dataset collects weekly transaction data from 47 U.S.

	# products	# in-sample	# out-of-sample
Hotel 1	10	1271	318
Hotel 2	6	347	87
Hotel 3	7	1073	268
Hotel 4	4	240	60
Hotel 5	6	215	54

	RF	MNL	Markov	ID
Hotel 1	**0.3040** (0.0046)	0.3098 (0.0031)	0.3047 (0.0039)	0.3224 (0.0043)
Hotel 2	**0.3034** (0.0120)	0.3120 (0.0148)	0.3101 (0.0124)	0.3135 (0.0178)
Hotel 3	**0.2842** (0.0051)	0.2854 (0.0065)	0.2842 (0.0064)	0.2971 (0.0035)
Hotel 4	0.3484 (0.0129)	**0.3458** (0.0134)	0.3471 (0.0125)	0.3584 (0.0047)
Hotel 5	0.3219 (0.0041)	0.3222 (0.0069)	**0.3203** (0.0046)	0.3259 (0.0058)

Table 10: Summary statistics of the five datasets and the average and standard deviation of the out-of-sample RMSE.

markets from 2001 to 2012, covering more than 30 product categories. Each transaction includes the week and store of purchase, the universal product code (UPC) of the purchased item, number of units purchased and total paid dollars.

The pre-processing steps taken follow those in Jagabathula and Rusmevichientong (2018) and Chen and Mišić (2019). In particular, we conduct the analysis for 31 categories separately using the data for the first two weeks in 2007. Each product is uniquely defined by the vendor code. Each assortment is defined as the set of products that are available in the same store in that week. We only focus on the top nine purchased products from all stores during the two weeks in each category and treat all other products as the no-purchase alternative.

However, the sales data for most categories is still too large for the EM algorithm to estimate the Markov chain model. For example, carbonated beverages, milk, soup and yogurt have more than 10 million transactions. For computational efficiency, we uniformly sample 1/200 of original data size without replacement. This does not significantly increase the sampling variability as most transactions in the original data are repeated entries.

We use five-fold cross-validation and RMSE defined in (5) to examine the out-of-sample performance. The result is shown in Table 11. Random forests outperform the

other two in 24 of 31 categories, especially for larger data size. According to Berbeglia et al. (2018), the Markov chain choice model has already been shown to have superb performance in synthetic and real-world studies. Table 11 fully demonstrates the potential of random forests as a framework to model and estimate consumer behaviors in practice.

7. Concluding Remarks

We hope that this study will encourage more scholars to pursue BRF as a research topic. We believe that addressing the following questions would help us decode the empirical success of random forests and understand the pitfalls:

- What type of DCMs can be estimated well by random forests and have higher generalizability to unseen assortments?
- As we use the choice forest to approximate DCMs, how can we translate the properties of a DCM to the topological structure of decision trees?
- Can we provide finite-sample error bounds for the performance of random forests, with or without the price information?
- What properties does the product importance index MDI have?
- Given a binary choice forest, possibly estimated by random forests, can we compute the optimal assortment efficiently?

References

Anderson, S. P., A. De Palma, and J.-F. Thisse (1992). *Discrete choice theory of product differentiation*. MIT press.

Ariely, D. (2008). *Predictably irrational*. Harper Audio.

Berbeglia, G. (2019). The generalized stochastic preference choice model. *Working paper*.

	Product Category	# data	RF	MNL	Markov
1	Beer	10,440	**0.2717** (0.0006)	0.2722 (0.0008)	0.2721 (0.0007)
2	Blades	1,085	0.3106 (0.0037)	**0.3092** (0.0034)	0.3096 (0.0036)
3	Carbonated Beverages	71,114	**0.3279** (0.0004)	0.3299 (0.0004)	0.3295 (0.0004)
4	Cigarettes	6,760	**0.2620** (0.0028)	0.2626 (0.0030)	0.2626 (0.0030)
5	Coffee	8,135	**0.2904** (0.0010)	0.2934 (0.0009)	0.2925 (0.0010)
6	Cold Cereal	30,369	**0.2785** (0.0003)	0.2788 (0.0003)	0.2787 (0.0003)
7	Deodorant	2,775	0.2827 (0.0005)	0.2826 (0.0005)	**0.2826** (0.0005)
8	Diapers	1,528	**0.3581** (0.0024)	0.3583 (0.0020)	0.3583 (0.0022)
9	Facial Tissue	8,956	**0.3334** (0.0007)	0.3379 (0.0010)	0.3375 (0.0007)
10	Frozen Dinners/Entrees	48,349	**0.2733** (0.0003)	0.2757 (0.0003)	0.2750 (0.0003)
11	Frozen Pizza	16,263	**0.3183** (0.0001)	0.3226 (0.0001)	0.3210 (0.0001)
12	Household Cleaners	6,403	0.2799 (0.0010)	0.2798 (0.0009)	**0.2798** (0.0009)
13	Hotdogs	7,281	**0.3123** (0.0011)	0.3183 (0.0005)	0.3170 (0.0007)
14	Laundry Detergent	7,854	**0.2738** (0.0017)	0.2875 (0.0017)	0.2853 (0.0016)
15	Margarine/Butter	9,534	**0.2985** (0.0004)	0.2995 (0.0004)	0.2990 (0.0003)
16	Mayonnaise	4,380	**0.3212** (0.0024)	0.3242 (0.0010)	0.3230 (0.0006)
17	Milk	56,849	**0.2467** (0.0007)	0.2501 (0.0005)	0.2538 (0.0012)
18	Mustard	5,354	**0.2844** (0.0008)	0.2856 (0.0006)	0.2852 (0.0006)
19	Paper Towels	9,520	**0.2939** (0.0009)	0.2964 (0.0008)	0.2959 (0.0008)
20	Peanut Butter	4,985	**0.3113** (0.0017)	0.3160 (0.0006)	0.3146 (0.0009)
21	Photography supplies	189	0.3456 (0.0081)	**0.3399** (0.0081)	0.3456 (0.0088)
22	Razors	111	**0.3269** (0.0300)	0.3294 (0.0225)	0.3323 (0.0195)
23	Salt Snacks	44,975	**0.2830** (0.0006)	0.2844 (0.0007)	0.2840 (0.0007)
24	Shampoo	3,354	0.2859 (0.0006)	**0.2855** (0.0071)	0.2856 (0.0009)
25	Soup	68,049	**0.2709** (0.0007)	0.2738 (0.0005)	0.2729 (0.0005)
26	Spaghetti/Italian Sauce	12,377	**0.2901** (0.0003)	0.2919 (0.0006)	0.2914 (0.0006)
27	Sugar Substitutes	1,269	0.3080 (0.0036)	**0.3067** (0.0035)	0.3072 (0.0034)
28	Toilet Tissue	11,154	**0.3084** (0.0005)	0.3126 (0.0004)	0.3132 (0.0014)
29	Toothbrushes	2,562	0.2860 (0.0009)	0.2859 (0.0004)	**0.2858** (0.0006)
30	Toothpaste	4,258	**0.2704** (0.0008)	0.2708 (0.0011)	0.2708 (0.0011)
31	Yogurt	61,671	**0.2924** (0.0011)	0.2976 (0.0008)	0.2960 (0.0008)

Table 11: The data size after pre-processing and the average and standard deviation of the out-of-sample RMSE (5) for each category.

Berbeglia, G., A. Garassino, and G. Vulcano (2018). A comparative empirical study of discrete choice models in retail operations. *Working paper*.

Bernstein, F., S. Modaresi, and D. Sauré (2018). A dynamic clustering approach to data-driven assortment personalization. *Management Science 65*(5), 2095–2115.

Biau, G. and E. Scornet (2016). A random forest guided tour. *Test 25*(2), 197–227.

Blanchet, J., G. Gallego, and V. Goyal (2016). A markov chain approximation to choice modeling. *Operations Research 64*(4), 886–905.

Block, H. D., J. Marschak, et al. (1959). Random orderings and stochastic theories of response. Technical report, Cowles Foundation for Research in Economics, Yale University.

Bodea, T., M. Ferguson, and L. Garrow (2009). Data set—choice-based revenue management: Data from a major hotel chain. *Manufacturing & Service Operations Management 11*(2), 356–361.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Bronnenberg, B. J., M. W. Kruger, and C. F. Mela (2008). Database paper—the iri marketing data set. *Marketing science 27*(4), 745–748.

Chen, Y.-C. and V. V. Mišić (2019). Decision forest: A nonparametric approach to modeling irrational choice. *Working paper*.

Cheung, W. C. and D. Simchi-Levi (2017). Thompson sampling for online personalized assortment optimization problems with multinomial logit choice models. *Working paper*.

Farias, V. F., S. Jagabathula, and D. Shah (2013). A nonparametric approach to modeling choice with limited data. *Management science 59*(2), 305–322.

Feng, G., X. Li, and Z. Wang (2017). On the relation between several discrete choice models. *Operations research 65*(6), 1516–1525.

Flores, A., G. Berbeglia, and P. Van Hentenryck (2017). Assortment and price optimization under the two-stage luce model. *Working paper*.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

Huber, J., J. W. Payne, and C. Puto (1982). Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research 9*(1), 90–98.

Jagabathula, S. and P. Rusmevichientong (2016). A nonparametric joint assortment and price choice model. *Management Science 63*(9), 3128–3145.

Jagabathula, S. and P. Rusmevichientong (2018). The limit of rationality in choice modeling: Formulation, computation, and implications. *Management Science 65*(5), 2196–2215.

Jagabathula, S., L. Subramanian, and A. Venkataraman (2019). A conditional gradient approach for nonparametric estimation of mixing distributions. *Management Science*.

Liaw, A. and M. Wiener (2002). Classification and regression by randomforest. *R News 2*(3), 18–22.

McFadden, D. (1973). Conditional logit analysis of qualitative choice behaviour. In P. Zarembka (Ed.), *Frontiers in Econometrics*, pp. 105–142. New York, NY, USA: Academic Press New York.

Mišić, V. V. (2016). *Data, models and decisions for large-scale stochastic optimization problems*. Ph. D. thesis, Massachusetts Institute of Technology.

Natarajan, K., M. Song, and C.-P. Teo (2009). Persistency model and its applications in choice modeling. *Management Science 55*(3), 453–469.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research 12*, 2825–2830.

Russo, J. E. and B. A. Dosher (1983). Strategies for multiattribute binary choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition 9*(4), 676.

Scornet, E., G. Biau, J.-P. Vert, et al. (2015). Consistency of random forests. *The Annals of Statistics 43*(4), 1716–1741.

Simonson, I. and A. Tversky (1992). Choice in context: Tradeoff contrast and extremeness aversion. *Journal of marketing research 29*(3), 281–295.

Şimşek, A. S. and H. Topaloglu (2018). An expectation-maximization algorithm to estimate the parameters of the markov chain choice model. *Operations Research 66*(3), 748–760.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

van Ryzin, G. and G. Vulcano (2014). A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science 61*(2), 281–300.

van Ryzin, G. and G. Vulcano (2017). An expectation-maximization method to estimate a rank-based choice model of demand. *Operations Research 65*(2), 396–407.

Wang, R. and O. Sahin (2017). The impact of consumer search cost on assortment planning and pricing. *Management Science 64*(8), 3649–3666.

Weitzman, M. L. (1979). Optimal search for the best alternative. *Econometrica*, 641–654.

Williams, H. C. (1977). On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and planning A 9*(3), 285–344.

A. Proofs

Proof of Theorem 1: It is easy to see that a BCF is a DCM. To show the converse, consider a collection of DCMs $p_c(i,S), c \in C$. Let $\alpha_c \geq 0$ with $\sum_{c\in C} \alpha_c = 1$, Then $p(i,S) = \sum_{c\in C} \alpha_c p_c(i,S)$ is clearly a DCM, so a convex combination of DCM is a DCM and thus all DCMs form a convex set.

Consider the extreme points, i.e., DCMs that cannot be written as a non-trivial convex combination of two or more DCMs. Let E be the collection of all extreme DCMs, $p_e(i,S), e \in E$. Then any DCM $p(i,S), i \in S_+, S \subset N$ is in the convex hull of $p_e, e \in E$. A deterministic DCM is a DCM such that $p(i,S) \in \{0,1\}^n$ for every $i \in S_+$ and every

$S \subset N$. Next we show that a DCM is an extreme point if and only if it is deterministic. Given a deterministic DCM, say d, let $i_d(S) \in S_+$ be the choice made by d, so that $p_d(i, S) = 1$ only if $i = i_d(S)$. It is clear that a deterministic DCM is an extreme point. Conversely, for an extreme DCM, if it is not deterministic, then we can always split the probability between 0 and 1 and makes it a convex combination of two different DCMs. Therefore, extreme points are equivalent to deterministic DCMs.

It is sufficient to show that all deterministic DCMs can be represented as a BCF. This follows directly because every deterministic DCM is the binary choice tree which can be explicitly constructed $t_b(S) \triangleq i_d(S)$ for all $S \subset N$. We can now formally state the connection between DCMs and BCFs. ■

Proof of Theorem 3. Let $p(i, \boldsymbol{x})$ be an arbitrary DCM. Construct B trees t_b, $b = 1, \ldots, B$, with 2^n leaves associated with each of the 2^n possible subsets of $[N]$. For any given $\boldsymbol{x} \in \{0, 1\}^N$, we let $\sum_{b=1}^{B} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} = \lfloor p(i, \boldsymbol{x})B \rfloor$ for $i = 1, \ldots, N$. It is easy to see that

$$|f(0, \boldsymbol{x}) - p(0, \boldsymbol{x})| \leq \frac{N}{B} \quad |f(i, \boldsymbol{x}) - p(i, \boldsymbol{x})| \leq \frac{1}{B}, \quad i = 1, \ldots, N.$$

Since the error bound holds for all $\boldsymbol{x}$ and i, the choice forest can approximate any DCM for a sufficiently large B.

■

Proof of Theorem 4: We first prove that for a single decision tree, there is a high probability that the number of observations chosen in Step 4 in which $\boldsymbol{x}$ is offered is large. More precisely, let $X_t = \mathbb{I}_{\{\boldsymbol{x}_t=\boldsymbol{x}\}}$. It is easy to see that $\sum_{t=1}^{T} X_t = k_T$. Step 4 randomly selects z_T observations out of the T with replacement. Denote the bootstrap sample of $\{X_1, \ldots, X_T\}$ by $\{Y_1, \ldots, Y_{z_T}\}$. By Hoeffding's inequality, we have the following concentration inequality

$$\mathsf{P}\left(\left|\frac{\sum_{j=1}^{z_T} Y_j}{z_T} - \frac{k_T}{T}\right| \leq \epsilon\right) \leq 2\exp\left(-2z_T\epsilon^2\right) \tag{6}$$

for any $\epsilon > 0$. In other words, the bootstrap sample in Step 4 does not deviate too far from the population as long as z_T is large. As we choose $\epsilon < \lim_{T\to\infty} k_T/T$, it implies

that $\sum_{j=1}^{z_T} Y_j \to \infty$ and in particular

$$\lim_{T\to\infty} \mathsf{P}(\sum_{j=1}^{z_T} Y_j > l_T) = 1. \tag{7}$$

Next we show that given $\sum_{j=1}^{z_T} Y_j > l_T$ for a decision tree, the leaf node that contains $\boldsymbol{x}$ only contains observations with $Y_j = 1$. That is, the terminal leaf containing $\boldsymbol{x}$ is a single corner of the unit hypercube. If the terminal leaf node containing an observation with predictor $\boldsymbol{x}$, then it has no less than $\sum_{j=1}^{z_T} Y_j$ observations, because all the $\sum_{j=1}^{z_T} Y_j$ samples used to train the tree fall on the same corner in the predictor space. If another observation with a different predictor is in the same leaf node, then it contradicts Step 6 in the algorithm, because it would imply that another split could be performed. Suppose $\{R_1, \ldots, R_M\}$ is the final partition corresponding to the decision tree. As a result, in the region R_j such that $\boldsymbol{x} \in R_j$, we must have that $t_b(\boldsymbol{x})$ is a random sample from the $\sum_{j=1}^{z_T} Y_j$ customer choices, according to Step 11.

Now consider the estimated choice probability from the random forest: $\sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}}$. Note that $\{t_b(\boldsymbol{x})\}_{b=1}^{B_T}$ are i.i.d. given the training set. By Hoeffding's inequality, conditional on $\{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}$,

$$\mathsf{P}\left(\left|\sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} - \mathsf{P}(t_b(\boldsymbol{x}) = i | \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T})\right| > \epsilon_1 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}\right) \leq 2e^{-2B_T\epsilon_1^2}, \tag{8}$$

for all $\epsilon_1 > 0$. Next we analyze the probability $\mathsf{P}(t_b(\boldsymbol{x}) = i | \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T})$ for a single decision tree. By the previous paragraph, conditional $\sum_{j=1}^{z_T} Y_j > l_T$, the output of a single tree $t_b(\boldsymbol{x})$ is randomly chosen from the class labels of $\sum_{j=1}^{z_T} Y_j$ observations whose predictor is $\boldsymbol{x}$. Let Z_j be the class label of the jth chosen observation in Step 4. Therefore, conditional on the event $\sum_{j=1}^{z_T} Y_j > l_T$ and the training data, we have

$$\mathsf{P}(t_b(\boldsymbol{x}) = i | \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}, \sum_{j=1}^{z_T} Y_j > l_T) = \sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j}. \tag{9}$$

Because $\{Y_j \mathbb{I}_{\{Z_j=i\}}\}_{j=1}^{z_T}$ is a bootstrap sample, having i.i.d. distribution

$$\mathsf{P}(Y_j \mathbb{I}_{\{Z_j=i\}} = 1) = \frac{\sum_{t=1}^{T} \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{T}$$

given the training data, we apply Hoeffding's inequality again

$$\mathsf{P}\left(\left| \frac{\sum_{j=1}^{z_T} Y_j \mathbb{I}_{\{Z_j=i\}}}{z_T} - \frac{\sum_{t=1}^{T} \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{T} \right| > \epsilon_2 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T} \right) \leq 2\exp(-2z_T\epsilon_2^2) \tag{10}$$

for all $\epsilon_2 > 0$. Now applying Hoeffding's inequality to $\sum_{t=1}^{T} \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}$ again, and because of Assumption 1, we have that

$$\mathsf{P}\left(\left| \frac{\sum_{t=1}^{T} \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T} - \mathsf{P}(i|\boldsymbol{x}) \right| > \epsilon_3 \right) \leq 2\exp(-2k_T\epsilon_3^2) \tag{11}$$

for all $\epsilon_3 > 0$.

With the above results, we can bound the target quantity

$$\begin{aligned}
&\mathsf{P}\left(\left| \sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} - \mathsf{P}(i|\boldsymbol{x}) \right| > \epsilon \right) \\
&= \mathsf{E}\left[\mathsf{P}\left(\left| \sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} - \mathsf{P}(i|\boldsymbol{x}) \right| > \epsilon \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T} \right) \right] \\
&\leq \mathsf{E}\left[\mathsf{P}\left(\left| \sum_{b=1}^{B_T} \frac{1}{B_T} \mathbb{I}_{\{t_b(\boldsymbol{x})=i\}} - \mathsf{P}(t_b(\boldsymbol{x}) = i | \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}) \right| > \epsilon/2 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T} \right) \right] \\
&\quad + \mathsf{E}\left[\mathsf{P}\left(\left| \mathsf{P}(i|\boldsymbol{x}) - \mathsf{P}(t_b(\boldsymbol{x}) = i | \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T}) \right| > \epsilon/2 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^{T} \right) \right]
\end{aligned}$$

By (8), the first term is bounded by $2\exp(-B_T\epsilon^2/2)$ which converges to zero as $B_T \to \infty$.

To bound the second term, note that

$$
\begin{aligned}
&\mathsf{P}\left(\left|\mathsf{P}(i|\boldsymbol{x}) - \mathsf{P}(t_b(\boldsymbol{x}) = i|\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T)\right| > \epsilon/2 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\leq \mathsf{P}\left(\left|\mathsf{P}(i|\boldsymbol{x}) - \frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T}\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\quad + \mathsf{P}\left(\left|\frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T} - \sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\quad + \mathsf{P}\left(\left|\sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j} - \mathsf{P}(t_b(\boldsymbol{x}) = i|\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T)\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \qquad (12)
\end{aligned}
$$

The expected value of the first term in (12) is bounded by $2\exp(-k_T\epsilon^2/18)$ by (11), which converges to zero as $k_T \to \infty$. For the second term of (12), we have that

$$
\begin{aligned}
&\mathsf{P}\left(\left|\frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T} - \sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\leq \mathsf{P}\left(\left|\frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T} - \sum_{j=1}^{z_T} \frac{T Y_j \mathbb{I}_{\{Z_j=i\}}}{z_T k_T}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\quad + \mathsf{P}\left(\left|\sum_{j=1}^{z_T} \frac{T Y_j \mathbb{I}_{\{Z_j=i\}}}{z_T k_T} - \sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \qquad (13)
\end{aligned}
$$

For the first term in (13), note that by (9)

$$
\begin{aligned}
&\mathsf{P}\left(\left|\frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{k_T} - \sum_{j=1}^{z_T} \frac{T Y_j \mathbb{I}_{\{Z_j=i\}}}{z_T k_T}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&= \mathsf{P}\left(\left|\frac{\sum_{t=1}^T \mathbb{I}_{\{i_t=i, \boldsymbol{x}_t=\boldsymbol{x}\}}}{T} - \sum_{j=1}^{z_T} \frac{Y_j \mathbb{I}_{\{Z_j=i\}}}{z_T}\right| > k_T\epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \leq 2\exp(-z_T k_T^2 \epsilon^2/72) \to 0
\end{aligned}
$$

as $T \to \infty$. For the second term in (13), we have

$$
\begin{aligned}
&\mathsf{P}\left(\left|\sum_{j=1}^{z_T} \frac{TY_j\mathbb{I}_{\{Z_j=i\}}}{z_T k_T} - \sum_{j=1}^{z_T} \frac{Y_j\mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\leq \mathsf{P}\left(\frac{\sum_{j=1}^{z_T} Y_j\mathbb{I}_{\{Z_j=i\}}}{z_T}\left|\frac{T}{k_T} - \frac{z_T}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\leq \mathsf{P}\left(\left|\frac{T}{k_T} - \frac{z_T}{\sum_{j=1}^{z_T} Y_j}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&= \mathsf{P}\left(\frac{Tz_T}{k_T \sum_{j=1}^{z_T} Y_j}\left|\frac{k_T}{T} - \frac{\sum_{j=1}^{z_T} Y_j}{z_T}\right| > \epsilon/12 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right)
\end{aligned}
$$

It is easy to see that $\frac{Tz_T}{k_T \sum_{j=1}^{z_T} Y_j}$ converges almost surely to a constant as $T \to \infty$. Therefore, by (6) the last term converges to zero. Finally we move on to the third term of (12). By (9), we have

$$
\begin{aligned}
&\mathsf{P}\left(\left|\sum_{j=1}^{z_T} \frac{Y_j\mathbb{I}_{\{Z_j=i\}}}{\sum_{j=1}^{z_T} Y_j} - \mathsf{P}(t_b(\boldsymbol{x}) = i|\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T)\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&= \mathsf{P}\left(\left|\mathsf{P}(t_b(\boldsymbol{x}) = i|\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T, \sum_{j=1}^{z_T} Y_j > l_T) - \mathsf{P}(t_b(\boldsymbol{x}) = i|\{(i_t, \boldsymbol{x}_t)\}_{t=1}^T)\right| > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) \\
&\leq \mathsf{P}\left(2\mathsf{P}\left(\sum_{j=1}^{z_T} Y_j \leq l_T \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) > \epsilon/6 \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right).
\end{aligned}
$$

Note that we are focusing on a fixed-design case, and $\{Y_j\}$ and $\{i_t\}$ are independent given Assumption 1. Therefore,

$$
\mathsf{P}\left(\sum_{j=1}^{z_T} Y_j \leq l_T \middle| \{(i_t, \boldsymbol{x}_t)\}_{t=1}^T\right) = \mathsf{P}\left(\sum_{j=1}^{z_T} Y_j \leq l_T\right) \to 0
$$

by (7). This completes the proof. ■

Proof of Proposition 1: We need to show that the bth tree constructed by the algorithm of both link functions returns the same partition (in the sense that each region contains the same set of observations in the training data) of the predictor space $[0, 1]^N$ and the

same class labels in each region/leaf. The class labels are guaranteed to be the same because we control the internal randomizer in Step 12. To show the partitions are the same, it suffices to show that each split creates regions that are identical for the two link functions in the sense that the resulting regions contain the same set of observations. We will use induction to prove this claim.

Before the construction of the bth tree, because the internal randomizers in Step 5 are equalized, the root node $[0,1]^N$ for both link functions contains the same set of observations. Now focusing on a leaf node in the middle of constructing the bth tree for both link functions. We use $[l_1^{(j)}, u_1^{(j)}] \times \cdots \times [l_N^{(j)}, u_N^{(j)}] \subset [0,1]^N$ to denote the region of the leaf node for link functions $j = 1, 2$. By the inductive hypothesis, both regions contain the same set of observations. Without loss of generality, we assume that the regions contain $\{g_1(\boldsymbol{p}_t)\}_{t=1}^{T_1}$ and $\{g_2(\boldsymbol{p}_t)\}_{t=1}^{T_1}$, respectively. After Step 8, the same set of candidate splitting dimensions are selected. To show that Step 9 results in the same split in the two regions, consider a given split direction m and split point x^j for $j = 1, 2$. If $[l_1^{(1)}, u_1^{(1)}] \times \ldots [l_m^{(1)}, x^{(1)}] \times \cdots \times [l_N^{(1)}, u_N^{(1)}]$ and $[l_1^{(2)}, u_1^{(2)}] \times \ldots [l_m^{(2)}, x^{(2)}] \times \cdots \times [l_N^{(2)}, u_N^{(2)}]$ contain the same set of observations, i.e., for $t = 1, \ldots, T_1$

$$\begin{aligned} & g_1(\boldsymbol{p}_t) \in [l_1^{(1)}, u_1^{(1)}] \times \ldots [l_m^{(1)}, x^{(1)}] \times \cdots \times [l_N^{(1)}, u_N^{(1)}] \\ \iff & g_2(\boldsymbol{p}_t) \in [l_1^{(2)}, u_1^{(2)}] \times \ldots [l_m^{(2)}, x^{(2)}] \times \cdots \times [l_N^{(2)}, u_N^{(2)}], \end{aligned}$$

then the Gini indices resulting from the splits are equal for the two link functions. This is because the Gini index only depends on the class composition in a region instead of the locations of the predictors, and the splits above lead to the same class composition in the sub-regions. This implies that in Step 8, both trees are going to find the optimal splits that lead to the same division of training data in the sub-regions. By induction and the recursive nature of the tree construction, Algorithm 2 outputs the same partition in the bth tree for both link functions, i.e., the training data is partitioned equally. This completes the proof. ■

Proof of Theorem 5: To simplify the notation, we use product $N + 1$ to denote the no-purchase option. Let n^j denote the number of assortments in the training data where product j is in the assortment. Clearly n^j has binomial distribution $B(T, 1/2)$. Let n_k^j denote the number of assortments in the training set where product j is in the

assortment and product k is chosen. Let n_k^{-j} denote the number of assortments in the training set where product j is not in the assortment and product k is chosen. The variable we defined above follows the following binomial distribution when $1 \le j \le N$:

$$n_k^j \begin{cases} \sim B(T, 1/2^{k+1}) & 1 \le k < j \\ \sim B(T, 1/2^j) & k = j \\ = 0 & j < k \le N+1 \end{cases}, \quad n_k^{-j} \begin{cases} \sim B(T, 1/2^{k+1}) & 1 \le k < j \\ = 0 & k = j \\ \sim B(T, 1/2^k) & j < k \le N \\ \sim B(T, 1/2^N) & k = N+1 \end{cases}.$$

For example, a data point counts as n_k^j when products 1 to $k-1$ are not in the assortment while product k and j are in the assortment. So the probability is $1/2^{k+1}$. Note that we have $\sum_{k=1}^{j} n_k^j = n^j$ and $\sum_{k=1}^{j-1} n_k^{-j} + \sum_{k=j+1}^{N+1} n_k^{-j} = T - n^j$. Next we compute the Gini index G_j if the first split is on product j. Recall the definition of the Gini index: $\sum_{R_j} \frac{t_j}{T} \sum_{k=0}^{N} \hat{p}_{jk}(1 - \hat{p}_{jk})$. If the split is on product j, then the left node (assortments without product j) has n^j data points while the right node has $T - n^j$. In the left node, the empirical frequency of label k, i.e., the fraction of assortments resulting in a purchase of product k, is $n_k^{-j}/(T - n^j)$. Similarly, in the right node, the empirical frequency is n_k^j/n^j for label $k \le j$ and 0 for label $k > j$. Therefore, we have that

$$\begin{aligned} G_j &= \frac{n^j}{T} \sum_{k=1}^{j} \frac{n_k^j}{n^j}\left(1 - \frac{n_k^j}{n^j}\right) + \frac{T - n^j}{T} \sum_{k=1}^{j-1} \frac{n_k^{-j}}{T - n^j}\left(1 - \frac{n_k^{-j}}{T - n^j}\right) + \frac{T - n^j}{T} \sum_{k=j+1}^{N+1} \frac{n_k^{-j}}{T - n^j}\left(1 - \frac{n_k^{-j}}{T - n^j}\right) \\ &= \frac{1}{T}\left(n^j - \frac{\sum_{k=1}^{j}(n_k^j)^2}{n^j} + T - n^j - \frac{\sum_{k=1}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2}{T - n^j}\right) \\ &= 1 - \frac{\sum_{k=1}^{j}(n_k^j)^2}{Tn^j} - \frac{\sum_{k=1}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2}{T(T - n^j)} \end{aligned} \tag{14}$$

Note that The second item of above equation only depends on assortments with product j, and the third item of above equation only depends on assortments without product j. Next we will show that $G_1 < G_j$ with high probability. We define H_j for the quantity in

(14) for simplicity:

$$H_j \triangleq \frac{\sum_{k=1}^{j}(n_k^j)^2}{n^j} + \frac{\sum_{k=1}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2}{T-n^j}.$$

It's easy to see that $G_1 < G_j$ if and only if $H_1 > H_j$. To bound its probability, we introduce the Chernoff inequality.

The Chernoff inequality: Let $X_1, X_2, ..., X_n$ be independent Bernoulli random variables. Denote $X = \sum_{i=1}^{n} X_i$ with $\mu = E[X]$. For all $0 < \delta < 1$, we have that $\mathrm{P}(X < (1-\delta)\mu) < \exp(-\frac{\mu\delta^2}{2})$.

Applying the Chernoff inequality to the Binomial random variable $n^j \sim B(T, 1/2)$ for $1 \le j \le N$, we have $\mathrm{P}(n^j < (1-\delta)T/2) = \mathrm{P}(n^j > (1+\delta)T/2) < \exp(-T\delta^2/4)$. Therefore,

$$\mathrm{P}\left(\left|n^j - \frac{1}{2}T\right| > \frac{\delta}{2}T\right) < 2\exp\left(-\frac{\delta^2 T}{4}\right) \tag{15}$$

Next we bound the probability that H_1 is small compared to its mean. We can rewrite H_1 as $H_1 = n^1 + \frac{\sum_{k=2}^{N+1}(n_k^{-1})^2}{T-n^1}$. Conditional on n^1, $n_2^{-1} \sim B(T-n^1, 1/2)$ and $n_3^{-1} \sim B(T-n^1, 1/4)$. Define $\delta_1 \triangleq \delta/\sqrt{1-\delta}$. By the Chernoff inequality we have:

$$\mathrm{P}\left((n_2^{-1})^2 < \frac{1}{4}(1-\sqrt{2}\delta_1)^2(T-n^1)^2 \middle| n^1\right) = \mathrm{P}\left(n_2^{-1} < \frac{1}{2}(1-\sqrt{2}\delta_1)(T-n^1)\middle| n^1\right) < \exp\left(-\frac{\delta_1^2(T-n^1)}{2}\right)$$

$$\mathrm{P}\left((n_3^{-1})^2 < \frac{1}{16}(1-2\delta_1)^2(T-n^1)^2 \middle| n^1\right) = \mathrm{P}\left(n_3^{-1} < \frac{1}{4}(1-2\delta_1)(T-n^1)\middle| n^1\right) < \exp\left(-\frac{\delta_1^2(T-n^1)}{2}\right)$$

Since $(n_k^{-1})^2 \ge 0$ for all $k \ge 4$, we have:

$$\begin{aligned}
&\mathrm{P}\left(\sum_{k=2}^{N+1}(n_k^{-1})^2 < \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right](T-n^1)^2 \middle| n^1\right) \\
&< \mathrm{P}\left((n_2^{-1})^2 + (n_3^{-1})^2 < \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right](T-n^1)^2 \middle| n^1\right) \\
&< \mathrm{P}\left((n_2^{-1})^2 < \frac{1}{4}(1-\sqrt{2}\delta_1)^2(T-n^1)^2 \middle| n^1\right) + \mathrm{P}\left((n_3^{-1})^2 < \frac{1}{16}(1-2\delta_1)^2(T-n^1)^2 \middle| n^1\right) \\
&< 2\exp\left(-\frac{\delta_1^2(T-n^1)}{2}\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathrm{P}\left(H_1 < n^1 + \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right](T-n^1)\middle| n^1\right) \\
&= \mathrm{P}\left(\sum_{k=2}^{N+1}(n_k^{-1})^2 < \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right](T-n^1)^2\middle| n^1\right) \\
&< 2\exp\left(-\frac{\delta_1^2(T-n^1)}{2}\right) \qquad (16)
\end{aligned}$$

Combining with (15), we have that the unconditional probability can also be bounded:

$$\begin{aligned}
&\mathrm{P}\left(H_1 < \frac{(1-\delta)T}{2} + \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right]\frac{(1+\delta)T}{2}\right) \\
&< \mathrm{P}\left(\left|n^1 - \frac{1}{2}T\right| > \frac{\delta}{2}T\right) \\
&\quad + \sum_{k:|k-T/2|<\delta T/2} \mathrm{P}(n^1=k)\mathrm{P}\left(H_1 < \frac{(1-\delta)T}{2} + \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right]\frac{(1+\delta)T}{2}\middle| n^1=k\right) \\
&< 2\exp\left(-\frac{\delta^2 T}{4}\right) + \sum_{k:|k-T/2|<\delta T/2} \mathrm{P}(n^1=k)\times 2\exp\left(-\frac{\delta_1^2(T-k)}{2}\right) \\
&< 2\exp\left(-\frac{\delta^2 T}{4}\right) + 2\exp\left(-\frac{\delta_1^2(1-\delta)T}{4}\right) = 4\exp\left(-\frac{\delta^2 T}{4}\right). \qquad (17)
\end{aligned}$$

Next we will bound the probability that H_2 is large compared to its mean. Recall that

$$H_2 = \frac{(n_1^2)^2 + (n_2^2)^2}{n^2} + \frac{(n_1^{-2})^2 + \sum_{k=3}^{N+1}(n_k^{-2})^2}{T-n^2} \qquad (18)$$

For the first item in (18), conditional on n^2, both n_1^2 and $n_2^2 = n^2 - n_1^2$ follows $B(n^2, 1/2)$. Therefore, by the Chernoff bound, we have

$$\mathrm{P}\left(\left|n_1^2 - \frac{1}{2}n^2\right| > \frac{\sqrt{2}}{2}\delta_1 n^2\middle| n^2\right) < 2\exp\left(-\frac{\delta_1^2 n^2}{2}\right)$$

Moreover, we have

$$\begin{aligned}
&\mathsf{P}\left((n_1^2)^2 + (n^2 - n_1^2)^2 > \frac{1}{2}(1+2\delta_1^2)(n^2)^2 \middle| n^2\right) \\
&= \mathsf{P}\left(\frac{1}{2}(n^2)^2 + 2(n_1^2 - \frac{1}{2}n^2)^2 > \frac{1}{2}(1+2\delta_1^2)(n^2)^2 \middle| n^2\right) \\
&= \mathsf{P}\left(\left|n_1^2 - \frac{1}{2}n^2\right| > \frac{\sqrt{2}}{2}\delta_1 n^2 \middle| n^2\right) < 2\exp\left(-\frac{\delta_1^2 n^2}{2}\right). \qquad (19)
\end{aligned}$$

where the first equality above follows from $(n_1^2)^2 + (n^2 - n_1^2)^2 = \frac{1}{2}(n^2)^2 + 2(n_1^2 - \frac{1}{2}n^2)^2$.

For the second item in (18), conditional on n^2, we have $n_1^{-2} \sim B(T - n^2, 1/2)$, $T - n^2 - n_1^{-2} \sim B(T - n^2, 1/2)$, $n_3^{-2} \sim B(T - n^2, 1/4)$ and $T - n^2 - n_1^{-2} - n_3^{-2} \sim B(T - n^2, 1/4)$. By the Chernoff bound we have:

$$\begin{aligned}
\mathsf{P}\left(n_3^{-2} < \frac{1}{4}(1-2\delta_1)(T-n^2) \middle| n^2\right) &< \exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right) \\
\mathsf{P}\left(T - n^2 - n_1^{-2} - n_3^{-2} < \frac{1}{4}(1-2\delta_1)(T-n^2) \middle| n^2\right) &< \exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)
\end{aligned}$$

From the above two equations we have

$$\mathsf{P}\left(n_3^{-2}(T - n^2 - n_1^{-2} - n_3^{-2}) < \frac{1}{16}(1-2\delta_1)^2(T-n^2)^2 \middle| n^2\right) < 2\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)$$

Similar to (19) we also have

$$\mathsf{P}\left((n_1^{-2})^2 + (T - n^2 - n_1^{-2})^2 > \frac{1}{2}(1+2\delta_1^2)(T-n^2)^2 \middle| n^2\right) < 2\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)$$

Combining the above two inequalities we have

$$
\begin{aligned}
&\mathrm{P}\left((n_1^{-2})^2+(n_3^{-2})^2+\sum_{k=4}^{N+1}(n_k^{-2})^2>\left[\frac{1}{2}(1+2\delta_1^2)-\frac{1}{8}(1-2\delta_1)^2\right](T-n^2)^2\middle|n^2\right)\\
<&\mathrm{P}\left((n_1^{-2})^2+(n_3^{-2})^2+(T-n^2-n_1^{-2}-n_3^{-2})^2>\left[\frac{1}{2}(1+2\delta_1^2)-\frac{1}{8}(1-2\delta_1)^2\right](T-n^2)^2\middle|n^2\right)\\
=&\mathrm{P}\left((n_1^{-2})^2+(T-n^2-n_1^{-2})^2-2n_3^{-2}(T-n^2-n_1^{-2}-n_3^{-2})>\left[\frac{1}{2}(1+2\delta_1^2)-\frac{1}{8}(1-2\delta_1)^2\right](T-n^2)^2\middle|n^2\right)\\
<&\mathrm{P}\left((n_1^{-2})^2+(T-n^2-n_1^{-2})^2>\frac{1}{2}(1+2\delta_1^2)(T-n^2)^2\middle|n^2\right)\\
&+\mathrm{P}\left(n_3^{-2}(T-n^2-n_1^{-2}-n_3^{-2})<\frac{1}{16}(1-2\delta_1)^2(T-n^2)^2\middle|n^2\right)\\
<&\,2\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)+2\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)\\
=&\,4\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right)
\end{aligned}
\tag{20}
$$

The first inequality follows from $\sum_{k=4}^{N+1} n_k^{-2}=T-n^2-n_1^{-2}-n_3^{-2}$ and thus $\sum_{k=4}^{N+1}(n_k^{-2})^2\le(T-n^2-n_1^{-2}-n_3^{-2})^2$. The first equality follows from $(n_3^{-2})^2+(T-n^2-n_1^{-2}-n_3^{-2})^2=(T-n^2-n_1^{-2})^2-2n_3^{-2}(T-n^2-n_1^{-2}-n_3^{-2})$.

Combine (19) and (20) we have

$$
\begin{aligned}
&\mathrm{P}\left(H_2>\frac{1}{2}(1+2\delta_1^2)T-\frac{1}{8}(1-2\delta_1)^2(T-n^2)\middle|n^2\right)\\
<&\mathrm{P}\left(\frac{(n_1^2)^2+(n^2-n_1^2)^2}{n^2}>\frac{1}{2}(1+2\delta_1^2)n^2\middle|n^2\right)\\
&+\mathrm{P}\left(\frac{(n_1^{-2})^2+(n_3^{-2})^2+\sum_{k=4}^{N+1}(n_k^{-2})^2}{T-n^2}>\left[\frac{1}{2}(1+2\delta_1^2)-\frac{1}{8}(1-2\delta_1)^2\right](T-n^2)\middle|n^2\right)\\
=&\mathrm{P}\left((n_1^2)^2+(n^2-n_1^2)^2>\frac{1}{2}(1+2\delta_1^2)(n^2)^2\middle|n^2\right)\\
&+\mathrm{P}\left((n_1^{-2})^2+(n_3^{-2})^2+\sum_{k=4}^{N+1}(n_k^{-2})^2>\left[\frac{1}{2}(1+2\delta_1^2)-\frac{1}{8}(1-2\delta_1)^2\right](T-n^2)^2\middle|n^2\right)\\
<&\,2\exp\left(-\frac{\delta_1^2n^2}{2}\right)+4\exp\left(-\frac{\delta_1^2(T-n^2)}{2}\right).
\end{aligned}
\tag{21}
$$

Next we bound the unconditional probability based on the conditional probability. We have

$$
\begin{aligned}
&\mathrm{P}\left(H_2 > \frac{1}{2}(1+2\delta_1^2)T - \frac{1}{8}(1-2\delta_1)^2\frac{(1-\delta)T}{2}\right) \\
<\ &\mathrm{P}\left(\left|n^2 - \frac{1}{2}T\right| > \frac{\delta}{2}T\right) \\
&+ \sum_{k:|k-T/2|\le\delta T/2} \mathrm{P}(n^2=k)\mathrm{P}\left(H_2 > \frac{1}{2}(1+2\delta_1^2)T - \frac{1}{8}(1-2\delta_1)^2\frac{(1-\delta)T}{2}\middle| n^2=k\right) \\
<\ &2\exp\left(-\frac{\delta^2 T}{4}\right) + \sum_{k:|k-T/2|\le\delta T/2} \mathrm{P}(n^2=k)\left(2\exp\left(-\frac{\delta_1^2 k}{2}\right) + 4\exp\left(-\frac{\delta_1^2(T-k)}{2}\right)\right) \\
\le\ &2\exp\left(-\frac{\delta^2 T}{4}\right) + \sum_{k:|k-T/2|\le\delta T/2} \mathrm{P}(n^2=k)\times 6\exp\left(-\frac{\delta_1^2(1-\delta)T}{4}\right) \\
=\ &2\exp\left(-\frac{\delta^2 T}{4}\right) + 6\exp\left(-\frac{\delta_1^2(1-\delta)T}{4}\right) = 8\exp\left(-\frac{\delta^2 T}{4}\right). \qquad (22)
\end{aligned}
$$

Next we choose a proper value for δ. By inequality (17) and (22), we want to find δ such that with high probability, we have

$$
\begin{aligned}
H_1 &\ge \frac{(1-\delta)T}{2} + \left[\frac{1}{4}(1-\sqrt{2}\delta_1)^2 + \frac{1}{16}(1-2\delta_1)^2\right]\frac{(1+\delta)T}{2} \\
&> \frac{1}{2}(1+2\delta_1^2)T - \frac{1}{8}(1-2\delta_1)^2\frac{(1-\delta)T}{2} \ge H_2
\end{aligned}
$$

where $\delta_1 = \delta/\sqrt{1-\delta}$. We also have the constraint that $0 < 2\delta_1 < 1$, which is equivalent to $0 < \delta < \frac{\sqrt{17}-1}{2} \approx 0.39$. Solving the above inequality for $0 < \delta < 0.39$ we have $0 < \delta \le 0.166185$. Let $\delta = 0.166185$. Then $4/\delta^2 \approx 145$. Plugging into (17) and (22), we have

$$
\mathrm{P}\left(H_1 < 0.512041T\right) < 4\exp\left(-\frac{T}{145}\right) \qquad (23)
$$

$$
\mathrm{P}\left(H_2 > 0.512041T\right) < 8\exp\left(-\frac{T}{145}\right) \qquad (24)
$$

Therefore

$$\mathrm{P}\left(H_1 < H_2\right) < 12\exp(-\frac{T}{145}).$$

This implies that $G_1 < G_2$ with high probability. Note that the probability bound in above equation doesn't depend on N. Next we consider $j \geq 3$. Recall that

$$H_j = \frac{\sum_{k=1}^{j}(n_k^j)^2}{n^j} + \frac{\sum_{k=1}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2}{T-n^j} \tag{25}$$

Consider some $\delta_2 > 0$. From (15) we have

$$\mathrm{P}\left(\left|n^j - \frac{1}{2}T\right| > \frac{\delta_2}{2}T\right) < 2\exp\left(-\frac{\delta_2^2 T}{4}\right) \tag{26}$$

We investigate the second item of (25). Conditional n^j, we have $n_1^{-j} \sim B(T-n^j, 1/2)$, $T - n^j - n_1^{-j} \sim B(T-n^j, 1/2)$, $n_2^{-j} \sim B(T-n^j, 1/4)$ and $T - n^j - n_1^{-j} - n_2^{-j} \sim B(T-n^j, 1/4)$. Define $\delta_3 \triangleq \delta_2/\sqrt{1-\delta_2}$. Then similar to (20), we have

$$\begin{aligned}
&\mathrm{P}\left((n_1^{-j})^2 + (n_2^{-j})^2 + \sum_{k=3}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2 > \left[\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2\right](T-n^j)^2 \middle| n^j\right) \\
&< \mathrm{P}\left((n_1^{-j})^2 + (n_2^{-j})^2 + (T-n^j-n_1^{-j}-n_2^{-j})^2 > \left[\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2\right](T-n^j)^2 \middle| n^j\right) \\
&< 4\exp\left(-\frac{\delta_3^2(T-n^j)}{2}\right).
\end{aligned} \tag{27}$$

Then similarly we can bound the first term of (25) since $n_1^j \sim B(n^j, 1/2)$, $n^j - n_1^j \sim B(n^j, 1/2)$, $n_2^j \sim B(n^j, 1/4)$ and $n^j - n_1^j - n_2^j \sim B(T-n^j, 1/4)$.

$$\begin{aligned}
&\mathrm{P}\left(\sum_{k=1}^{j}(n_k^j)^2 > \left[\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2\right](n^j)^2 \middle| n^j\right) \\
&< \mathrm{P}\left((n_1^j)^2 + (n_2^j)^2 + (n^j-n_1^j-n_2^j)^2 > \left[\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2\right](n^j)^2 \middle| n^j\right) \\
&< 4\exp\left(-\frac{\delta_3^2 n^j}{2}\right).
\end{aligned} \tag{28}$$

Combining (27) and (28), we have

$$\begin{aligned}
&\mathrm{P}\left(H_j > [\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2]T \middle| n^j\right) \\
&< \mathrm{P}\left(\sum_{k=1}^{j}(n_k^j)^2 > [\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2](n^j)^2 \middle| n^j\right) \\
&\quad + \mathrm{P}\left(\sum_{k=1}^{j-1}(n_k^{-j})^2 + \sum_{k=j+1}^{N+1}(n_k^{-j})^2 > [\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2](T-n^j)^2 \middle| n^j\right) \\
&< 4\exp\left(-\frac{\delta_3^2 n^j}{2}\right) + 4\exp\left(-\frac{\delta_3^2(T-n^j)}{2}\right) \qquad (29)
\end{aligned}$$

Using a similar argument, we can bound the unconditional probability:

$$\mathrm{P}\left(H_j > [\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2]T\right) < 10\exp\left(-\frac{\delta_2^2 T}{4}\right). \qquad (30)$$

Similarly we can calculate δ_2. By (30) and (23), we have the following condition for δ_2

$$\frac{1}{2}(1+2\delta_3^2) - \frac{1}{8}(1-2\delta_3)^2 < 0.512041, \qquad (31)$$

where $\delta_3 = \delta_2/\sqrt{1-\delta_2}$. Again when we consider $0 < \delta_2 < 0.39$, the above inequality is equivalent to $0 < \delta_2 < 0.200261$. Let $\delta_2 = 0.200261$, then $4/\delta_2^2 \approx 99.74$, so we have for all $j \geq 3$

$$\mathrm{P}\left(H_j > 0.512041T\right) < 10\exp\left(-\frac{T}{100}\right) \qquad (32)$$

Now note that

$$\begin{aligned}
&\mathrm{P}\,(\text{first split not on product 1}) = \mathrm{P}\left(G_1 > \min\{G_j | 2 \leq j \leq N\}\right) \\
&= \mathrm{P}\left(H_1 < \max\{H_j | 2 \leq j \leq N\}\right) < \mathrm{P}\left(H_1 < 0.512041T\right) + \sum_{j=2}^{N}\mathrm{P}\left(H_j > 0.512041T\right) \\
&< 4\exp\left(-\frac{\delta^2 T}{4}\right) + 8\exp\left(-\frac{\delta^2 T}{4}\right) + (N-2)\times 10\exp\left(-\frac{\delta_2^2 T}{4}\right) \\
&= 12\exp\left(-\frac{T}{145}\right) + 10(N-2)\exp\left(-\frac{T}{100}\right) \qquad (33)
\end{aligned}$$

Next we are going to show that the probability of the second split on product 2, third split on product 3 and so on, can be bounded by union bound.

Let $\mathcal{T}$ denote the training set. Define a sequence of subsets of $\mathcal{T}$ as follows: $\mathcal{T}_i \triangleq \{S \in \mathcal{T} | 1, 2, ..., i-1 \notin S\}$. That is, $\mathcal{T}_i$ only contains assortments that include a subset of $\{i, i+1, \ldots, N\}$. Let $T_i \triangleq |\mathcal{T}_i|$ denote the cardinality of set $\mathcal{T}_i$. Note that $\mathcal{T}_1 = \mathcal{T}$ and $T_1 = T$. for $2 \leq i \leq N$, since $T_i \sim B(T, 1/2^{i-1})$, by the Chernoff inequality we have

$$\mathsf{P}\left(T_i < (1-\delta_0)\frac{1}{2^{i-1}}T\right) < \exp(-\frac{\delta_0^2 T}{2^i}), \tag{34}$$

where $0 < \delta_0 < 1$.

Define the event $A_i \triangleq \{\text{product } i \text{ is the best split for training set } \mathcal{T}_i\}$, and let $\bar{A}_i$ denote the complement of event A_i. Then conditional on T_i, we can bound the probability of event $\bar{A}_i$ from (33) (here we only have $N - i + 1$ products):

$$\mathsf{P}\left(\bar{A}_i \middle| T_i\right) < 12\exp\left(-\frac{T_i}{145}\right) + 10(N-i-1)\exp\left(-\frac{T_i}{100}\right) \tag{35}$$

The unconditional probability can be bounded by combining (34) and (35):

$$\begin{aligned}
\mathsf{P}\left(\bar{A}_i\right) &< \mathsf{P}\left(T_i > (1-\delta_0)\frac{1}{2^{i-1}}T\right) + \sum_{k:k\leq(1-\delta_0)T/2^{i-1}} \mathsf{P}(T_i = k)\mathsf{P}\left(\bar{A}_i | T_i = k\right) \\
&< \exp(-\frac{\delta_0^2 T}{2^i}) + \sum_{k:k<(1-\delta_0)T/2^{i-1}} \mathsf{P}(T_i = k)\left(12\exp\left(-\frac{k}{145}\right) + 10(N-i-1)\exp\left(-\frac{k}{100}\right)\right) \\
&\leq \exp(-\frac{\delta_0^2 T}{2^i}) + 12\exp\left(-\frac{(1-\delta_0)T}{145\cdot 2^{i-1}}\right) + 10(N-i-1)\exp\left(-\frac{(1-\delta_0)T}{100\cdot 2^{i-1}}\right)
\end{aligned} \tag{36}$$

Solving the equation $\frac{\delta_0^2 T}{2^i} = \frac{(1-\delta_0)T}{145\cdot 2^{i-1}}$, we get $\delta_0 \approx 0.1107$. Then we have

$$\mathsf{P}\left(\bar{A}_i\right) \leq 13\exp\left(-\frac{T}{164\cdot 2^{i-1}}\right) + 10(N-i-1)\exp\left(-\frac{T}{113\cdot 2^{i-1}}\right)$$

If all the events $A_1, A_2, ..., A_m$ happen, we can get right split for the first m step. That is, the first split is on product 1, the second split is on product 2, …, the mth split is on

product m. We can bound the probability by the union bound:

$$\begin{aligned}
\mathrm{P}\left(\cap_{i=1}^{m} A_i\right) &= 1 - \mathrm{P}\left(\cup_{i=1}^{m} \bar{A}_i\right) \\
&\geq 1 - \sum_{i=1}^{m} \mathrm{P}(\bar{A}_i) \\
&\geq 1 - \sum_{i=1}^{m} \left[13 \exp\left(-\frac{T}{164 \cdot 2^{i-1}} \right) + 10(N-i-1) \exp\left(-\frac{T}{113 \cdot 2^{i-1}} \right) \right] \qquad (37)
\end{aligned}$$

If the first m splits match the products, then the assortments including at least one product among $\{1, \ldots, m\}$ can be correctly classified. Therefore, with probability at least $\mathrm{P}\left(\cap_{i=1}^{m} A_i\right)$, we can accurately predict at least a fraction $1 - 1/2^m$ of assortments. Given $\epsilon > 0$, letting $m = \lceil \log_2 \frac{1}{\epsilon} \rceil$ completes the proof. ■

B. Sample Code

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier

n = 10 # number of products
T = 500 # size of the training set

offered_set_binary = np.random.randint(2,size = (T,n))  #training assortments
choices = np.random.randint(n,size = T)  #randomly generate choices

for i in range(T):
    if offered_set_binary[i][choices[i]] == 0:
        choices[i] = -1  #no-purchase if not in assortment

classifier = RandomForestClassifier(n_estimators=1000, max_features='auto',
                                    random_state=50, min_samples_split=50)
classifier.fit(offered_set_binary, choices)  #fit data by random forest

new_set_binary = np.random.randint(2,size = n)  #testing assortment
rf_test = classifier.predict_proba([new_set_binary])  #predict probability
rf_test = rf_test.flatten()
#remove the products not in the assortment
predict_prob = np.append(1,new_set_binary)*rf_test
predict_prob = predict_prob/np.sum(predict_prob)
```