Estimating Binary Choice Models with Random Forests

Jihuan Zhang

April 13, 2024

Abstract

We propose a modified random forests estimator for binary choice models, which uses the negative log-likelihood function instead of the impurity measures in the original random forests. The parameters in the latent utility function are estimated using the recursive partitioning maximum likelihood estimation. Simulation studies suggest that the modified random forests estimator works well in finite samples, and it performs better than the kernelized probit approach in estimating discontinuous or nearly discontinuous functions.

1 Introduction

Random forests, developed by Breiman (2001), are tree-based ensemble learning methods for regression and classification. Random forests rapidly gained popularity due to their notable accuracy, significant flexibility, and relatively straightforward tuning process (Genuer et al., 2008; Lechner and Okasa, 2019). In addition, they can handle a large number of variables without overfitting (Biau, 2012). Therefore, random forests have become serious competitors to many other popular machine learning techniques such as boosting (Freund and Schapire, 1997) and support vector machine (Cortes and Vapnik, 1995).

The Binary choice model (BCM) is an econometric model with theoretical implications of individual utility maximization and decision-making, which makes it very popular in many economic fields such as health economics and labor economics (Bhattacharya, 2021). In a typical BCM, the latent utility is a sum of a systematic component as a function of covariates, and a random component that represents the idiosyncratic error. The observable binary outcomes are driven by the latent utility as an indicator function. Like many other machine learning methods, random forests relate the outcome variable directly to the covariates without making any assumptions about the data-generating process. In the classification problem, the underlying error term follows a Bernoulli distribution, with the probability of success given by $p(x)$, the conditional choice probability, which is quite restrictive. The BCM provides an additional structure of $p(x)$ using the information in the systematic component and the distribution function of a more general error component. Without any modification, random forests cannot capture such a structure and thus, they cannot be applied directly to estimate the BCM effectively. Then, it is interesting to investigate how random forests can be adapted to estimate and analyze the BCM. In this paper, we propose a modified random forests probit estimator, which makes use of the negative log-likelihood function instead of the impurity measures that are used in the traditional random forests, such as the Gini index and cross-entropy. The parameters in the systematic component are estimated using the recursive partitioning maximum likelihood estimation.

There is a growing literature on the use of machine learning methods in economic and econometric analysis. For example, Chernozhukov et al. (2018) introduced a de-biased estimator for treatment and structural parameters using machine learning techniques such as LASSO. Wager and Athey (2018) applied random forests on estimation and inference of heterogeneous treatment effects. Farrell et al. (2021) studied the theoretical properties of deep neural networks and their application in causal inference as first-stage estimators. For a more comprehensive discussion, see, for instance, Varian (2014), Mullainathan and Spiess (2017), Athey and Imbens (2019), and references therein. The paper contributes to this expanding literature on estimation and inference in econometric models using machine learning methods, in a high-dimensional and big-data environment.

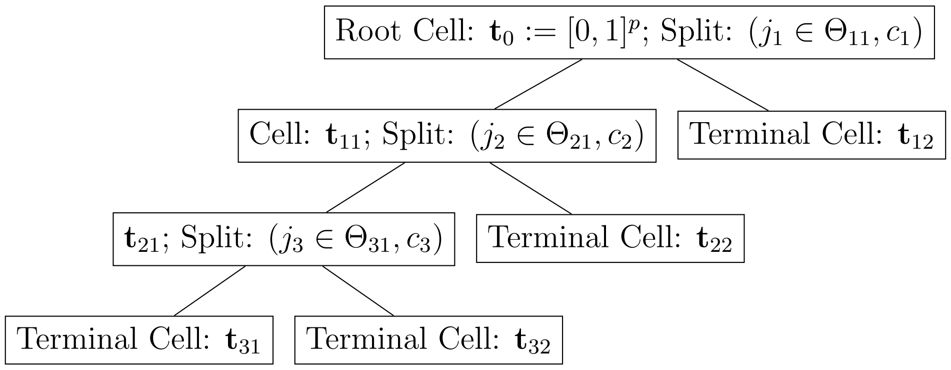
We also enrich the literature on the nonparametric estimation of BCMs. The majority of the studies focus on regression analysis, with relatively less emphasis placed on estimating discrete choice models. Yan (2023) proposes a kernelized nonparametric estimator (KNP) for the BCM using kernel tricks. The method leverages the concept of reproducing kernel Hilbert space, a foundation for many machine learning techniques such as support vector machine. The KNP is able to estimate any function in the space of a continuous function equipped with the supremum norm defined in a compact Euclidean space. However, in econometric modeling (e.g., regime-switching models and regression discontinuity designs), sometimes certain types of jumps may appear in the models inherently. The nature of random forests allows us to estimate the systematic component nonparametrically by simple functions, which can capture a wide class of Borel measurable functions, including discontinuous ones. This is a significant departure from traditional linear models like probit and logistic regression. Moreover, our approach is capable of managing a very large number of covariates without succumbing to overfitting. This capability sets it apart from other nonparametric techniques like local constant/linear estimators, which might struggle with high-dimensional data. They only use data points locally, while as tree-based approaches, random forests look at global information within the entire dataset. Our simulation results demonstrate the effectiveness and robustness of our approach for both continuous and discontinuous model specifications.

The rest of the discussion is organized as follows. Section 2 briefly introduces Breiman's original random forests algorithm. Section 3 describes the methodology for the estimation of BCM using modified random forests in detail. Section 4 presents the Monte Carlo simulation results, and section 5 concludes the paper.

2 Random Forests

The ideas of random forests were influenced by early works of Amit and Geman (1997), Ho (1998), and Dietterich (2000). Bootstrap aggregation (or bagging), column subsampling, and the Classification And Regression Trees (CART) split criterion (Breiman et al., 1984), are key components of forest methodologies. As a computationally intensive nonparametric approach, random forests work especially well for high-variance, low-bias procedures, particularly with extensive, high-dimensional datasets where the complexity of the problem makes it impossible to derive an optimal model in a single attempt (Bühlmann and Yu, 2002; Hastie et al., 2009; Biau and Scornet, 2016).

This subsection aims to briefly introduce the original Breiman's random forests estimation procedure. The method involves a technique commonly referred to as recursive partitioning. Recursive partitioning begins with the root cell containing the entire dataset. The root node is partitioned into two daughter cells based on the optimal split determined by a specific criterion. Subsequently, these subsidiary nodes are each split further into two more nodes. The process will continue until a predefined stopping condition is satisfied.



Root Cell: $\mathbf{t}_0 := [0,1]^p$; Split: $(j_1 \in \Theta_{11}, c_1)$
Cell: $\mathbf{t}_{11}$; Split: $(j_2 \in \Theta_{21}, c_2)$
Terminal Cell: $\mathbf{t}_{12}$
$\mathbf{t}_{21}$; Split: $(j_3 \in \Theta_{31}, c_3)$
Terminal Cell: $\mathbf{t}_{22}$
Terminal Cell: $\mathbf{t}_{31}$
Terminal Cell: $\mathbf{t}_{32}$

Figure 1: A tree example of depth 3

Let us focus on the task of binary classification. Like most machine learning methods for classification, random forests do not place any assumptions on how the binary outcomes are generated. For simplicity, let a p-dimensional covariate vector be $\mathbf{X} \in [0,1]^p$, and the outcome be $Y \in \{0,1\}$[1]. Let the data be $\mathcal{D} = \{y_i, \mathbf{x}_i\}_{i=1}^n$ as n independent copies of $\{Y, \mathbf{X}\}$. From the data, we create B bootstrap samples $\{\mathcal{D}_1, ..., \mathcal{D}_B\}$. A typical tree structure is illustrated in Figure 1. Starting from the root cell $\mathbf{t}_0 := [0,1]^p$ that contains the entire bootstrap sample $\mathcal{D}_b$, we uniformly draw $\Theta_{11} \subset \{1, ..., p\}$ of cardinality $m < p$, with replacement. The random selection of covariates is called column subsampling, one of the essential ingredients of random forests. The integer m is the column-subsampling parameter. It is a tuning parameter and is usually set to be $m = \lfloor \sqrt{p} \rfloor$ for classification[2]. Then, we construct the set of all splitting rule (j_1, c_1), where $j_1 \in \Theta_{11}$ and $c_1 \in \{x_{ij_1} : \mathbf{x}_i \in \mathbf{t}_0\} \subset [0,1]$ is a cutting point corresponding to covariate j_1. Each split will yield two daughter cells denoted by $\mathbf{t}_{11}$ and $\mathbf{t}_{12}$, where $\mathbf{t}_{11} := [0,1]^{j_1-1} \times (c_1, 1] \times [0,1]^{p-j_1}$ and $\mathbf{t}_{12} := [0,1]^{j_1-1} \times [0, c_1] \times [0,1]^{p-j_1}$. The best split is determined by the so-called sample CART-split criterion,

$$(\hat{j}_1, \hat{c}_1) := \arg \max_{j_1 \in \Theta_{11}, c_1} \{ I(p(\mathbf{t}_0)) - p_L I(p(\mathbf{t}_{11})) - p_R I(p(\mathbf{t}_{12})) \} \tag{1}$$

where $p_L := \frac{\#\{i:\mathbf{x}_i \in \mathbf{t}_{11}\}}{\#\{i:\mathbf{x}_i \in \mathbf{t}_0\}}$ and $p_R := \frac{\#\{i:\mathbf{x}_i \in \mathbf{t}_{12}\}}{\#\{i:\mathbf{x}_i \in \mathbf{t}_0\}}$ are proportion of examples in $\mathbf{t}_0$ that fall in $\mathbf{t}_{11}$ and $\mathbf{t}_{12}$, respectively[3]. $p(\mathbf{t}) := \frac{\#(\{i:\mathbf{x}_i \in \mathbf{t}\} \cap \{i:y_i=1\})}{\#\{i:\mathbf{x}_i \in \mathbf{t}\}}$ is the proportion of class 1 in cell $\mathbf{t}$

[1] In fact, there is no need to standardizing the covariates as the sample CART-split criterion described in (1) is scale-invariant.

[2] For regression, m is usually set to be $\lfloor p/3 \rfloor$. In practical applications, m is context-dependent and thus, it requires careful tuning by researchers.

[3] $\#$ denotes the number of elements in a set.

and $I(p)$ is called the impurity measure. The criterion breaks ties randomly. As formally discussed in Breiman et al. (1984), an impurity measure should satisfy the following three properties: (i) $I(0) = I(1) = 0$; (ii) $I(p) = I(1-p)$; and (iii) $I''(p) < 0$, for all $p \in (0,1)$[4]. Commonly used impurity measures for classification include Gini index $I(p) = p(1-p)$ and cross-entropy $I(p) = -p \ln p - (1-p)\ln(1-p)$. Intuitively, these measures are designed so that impurity is highest when p is near one-half, reflecting a high level of uncertainty or mixed classification within a node; thus, $I(p)$ reaches its maximum at $p = 1/2$. The sample CART-split criterion selects the optimal split that maximizes the decrease in impurity. The same process is repeated on the resulting two descendant cells until certain predetermined stopping criteria are met[5].

Let us formally define a final tree $T_b(\Theta_{1:k})$. We use the letter k to denote the depth of the final tree. For example, Figure 1 is a tree of depth $k = 3$, where cell $\mathbf{t}_{11}$ is in level 1 and cell $\mathbf{t}_{21}$ is in level 2. There are four terminal nodes (or leaves) in the example, each of which is associated with a unique path originating from the root node. In particular, they form a set of branches and leaves $T_b(\Theta_{1:k}) = \{(\mathbf{t}_{11}, \mathbf{t}_{21}, \mathbf{t}_{31}), (\mathbf{t}_{11}, \mathbf{t}_{21}, \mathbf{t}_{32}), (\mathbf{t}_{11}, \mathbf{t}_{22}), (\mathbf{t}_{12})\}$, where $\Theta_{1:k} := \{\Theta_1, ..., \Theta_k\}$ is one realization of column subsampling, which is a discrete random variable. In this example, $\Theta_1 = \Theta_{11}$, $\Theta_2 = \Theta_{21}$, $\Theta_3 = \Theta_{31}$. Note that a tree with depth k has at most 2^k terminal nodes, in which case $\Theta_l = (\Theta_{l,1}, ..., \Theta_{l,2^{l-1}})$ for all $1 \leqslant l \leqslant k$ and $T_b(\Theta_{1:k}) = \{\mathbf{t}_{1:k} := (\mathbf{t}_1, ..., \mathbf{t}_k) : \ \mathbf{t}_k \text{ is a terminal node}\}$. A similar definition can be found in Chi et al. (2022).

Given a new input $\mathbf{x}$, the output of a single tree is an estimate

$$\hat{C}_b(\mathbf{x}) := \sum_{\mathbf{t}_{1:l} \in T_b(\Theta_{1:k})} 1\{\mathbf{x} \in \mathbf{t}_l\} 1 \left\{ \sum_{i \in \{i : \mathbf{x}_i \in \mathbf{t}_l\}} y_i > \frac{\#\{i : \mathbf{x}_i \in \mathbf{t}_l\}}{2} \right\}$$

Each leaf is associated with a predicted class which is the majority class within the leaf. The output of the entire random forests classification is an estimate for the classifier $C : [0,1]^p \mapsto \{0,1\}$ such that

$$\hat{C}(\mathbf{x}) = 1 \left\{ \frac{1}{B} \sum_{b=1}^{B} \hat{C}_b(\mathbf{x}) > \frac{1}{2} \right\} \tag{2}$$

[4]See, for example, chapter 4.2, proposition 4.4, and theorem 4.5.

[5]Various frequently applied stopping rules are discussed in section 3.2.

which is the so-called majority vote.

Growing a single tree only will cause overfitting issues, even with a proper pruning process. Random forests effectively reduce variance and significantly mitigate overfitting by aggregating a large ensemble of trees, which are made less correlated through column subsampling. To see this, note that the average of B identically distributed estimators with positive pairwise correlation ρ and individual variance σ^2, has variance $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$ (see Hastie et al., 2009). The variance reduction can be achieved by reducing ρ through column subsampling as B increases. This variance reduction mechanism is one of the reasons why random forests have gained popularity and have been implemented in various software packages.

The next section of the discussion will explore how random forests can be effectively adapted to estimate the BCM.

3 Methodology

3.1 The Model

The BCM is given by

$$\begin{aligned} Y^* &= G_0(\mathbf{X}) - U \\ Y &= 1\{Y^* > 0\} \end{aligned} \tag{3}$$

where Y^* represents the latent utility, G_0 is the systematic component (or indirect utility) and U denotes the idiosyncratic error, assumed to be independent of $\mathbf{X}$. For a binary outcome $Y \in \{0, 1\}$ and covariates $\mathbf{X} \in \mathcal{X}$, the conditional choice probability (CCP) is given by

$$p_0(\mathbf{x}) = \mathbb{P}\{Y = 1|\mathbf{X} = \mathbf{x}\} = \mathbb{E}(Y|\mathbf{X} = \mathbf{x}) = F_0(G_0(\mathbf{x})) \tag{4}$$

where F_0 is the distribution function of U.

Random forests, as a popular machine learning method, are able to estimate the CCP directly by minimizing $\text{Var}(Y|\mathbf{X}) = p_0(\mathbf{X})(1-p_0(\mathbf{X}))$. However, it lacks the ability to discern the underlying structural form present in the CCP. Therefore, we are driven to investigate how random forests can be adapted or refined to estimate discrete choice models as they capture the structural interpretation in the CCP and align with economic theories, such as the random utility theory.

For the BCM, (see Matzkin, 1992), we say $(G_0, F_0) \in (\mathcal{G}, \mathcal{F})$ is identified in $(\mathcal{G}, \mathcal{F})$ if for all $(G, F) \in (\mathcal{G}, \mathcal{F})$ such that $F(G(\mathbf{X})) = F_0(G_0(\mathbf{X}))$ for $\mathbf{X}$ almost surely, it holds that $G = G_0$ and $F = F_0$ almost everywhere with respect to the probability measure induced by $\mathbf{X}$ and the Lebesgue measure, respectively.

In this study, due to the computational intensity of the proposed modified method as well as our limited expertise in programming, we simplify the BCM by assuming that the error term follows an i.i.d. standard normal distribution (so that $F = F_0$). This simplification is standard, as with linear probit regression, and can always serve as a benchmark. Because of the simplification, the model is identified as long as the covariates are independent of the error term[6]. To see this, define $A_G := \{\mathbf{x} \in \mathcal{X} : F(G(\mathbf{x})) = F(G_0(\mathbf{x}))\}$ and $B_G := \{\mathbf{x} \in \mathcal{X} : G(\mathbf{x}) = G_0(\mathbf{x})\}$. For all $\mathbf{x} \in A_G$, we have $G(\mathbf{x}) = G_0(\mathbf{x})$ since the distribution function of standard normal F is strictly increasing. Therefore, A_G is a subset of B_G so that $\mathbb{P}(A_G) = 1$ implies $\mathbb{P}(B_G) = 1$, from which identification of G_0 is established.

3.2 Estimation

The estimation of a binary choice model with random forests can be achieved by using the negative log-likelihood function instead of the impurity measure. Similar to the CART methodology, recursive partitioning can be applied to estimate the parameters of the systematic component.

In a cell $\mathbf{t}$, the binary choice model to be evaluated can be represented by

$$\begin{aligned} Y^* &= \beta_L 1\{\mathbf{X} \in \mathbf{t}_L\} + \beta_R 1\{\mathbf{X} \in \mathbf{t}_R\} - U \\ Y &= 1\{U < \beta_L\}1\{\mathbf{X} \in \mathbf{t}_L\} + \beta_R 1\{\mathbf{X} \in \mathbf{t}_R\}\} \end{aligned} \tag{5}$$

where $\mathbf{t}_L$ and $\mathbf{t}_R$ denote the left and right daughter cells of $\mathbf{t}$ such that $\mathbf{t} = \mathbf{t}_L \cup \mathbf{t}_R$. Here we view the systematic component in the current cell as a linear combination of two indicator functions. This is due to the fact that within the current cell, the indicator functions associated with other cells will take on the value of zero. Different from the CART-split criterion, we must also optimize our objective function over the two coefficients, β_L and β_R.

[6] For the identification results with unknown F_0, see, for example, Yan (2023)

The optimization problem is thus defined by

$$\max_{j\in\Theta,c,\beta_L,\beta_R}\left\{\sum_{X_i\in\mathbf{t}}[y_i\ln(F(G_{\mathbf{t}}(X_i)))+(1-y_i)\ln(1-F(G_{\mathbf{t}}(X_i)))]\right\} \tag{6}$$

where $G_{\mathbf{t}}(X_i):=\beta_L 1\{X_{ij}>c\}+\beta_R 1\{X_{ij}\leqslant c\}$. It can be further simplified to

$$\max_{j\in\Theta,c}\left\{\max_{\beta_L,\beta_R}\left\{\sum_{X_{ij}>c}[y_i\ln(F(\beta_L))+(1-y_i)\ln(1-F(\beta_L))]+\sum_{X_{ij}\leqslant c}[y_i\ln(F(\beta_R))+(1-y_i)\ln(1-F(\beta_R))]\right\}\right\} \tag{7}$$

Note that if either one of the two daughter nodes contains only one class, then $|\hat{\beta}_L|$ or $|\hat{\beta}_R|$ will explode to ∞, because of the fact that $\ln(F(\beta_L))$ and $\ln(1-F(\beta_L))$ are strictly increasing and decreasing in β_L, respectively. The issue is more likely to appear as the depth of the tree increases. Some regularization techniques are needed to make the optimization problem well-defined. In this paper, we consider L1 and L2 penalty terms. The one we choose in the paper is the L2 regularization, and we demonstrate how to include the L1 penalty in the appendix. Finally, the best split is determined by solving the following optimization problem

$$\max_{j\in\Theta,c}\left\{\max_{\beta_L,\beta_R}\{n_{1L}\ln(F(\beta_L))+n_{0L}\ln(1-F(\beta_L))+n_{1R}\ln(F(\beta_R))+n_{0R}\ln(1-F(\beta_R))-\lambda_n(\beta_L^2+\beta_R^2)\}\right\} \tag{8}$$

where n_{1L} denotes the number of observations in cell $\mathbf{t}$ with $y_i=1$ and $X_{ij}>c$, n_{0R} denotes the number of observations with $y_i=0$ and $X_{ij}\leqslant c$, and n_{0L} and n_{1R} are defined similarly. Here the dependence of n_{1L}, n_{1R}, n_{0L}, and n_{0R}, on $(j\in\Theta,c)$ are suppressed for simplicity. The problem given in (8) not only provides the best split for the current cell but also determines the heights of the two indicator functions. However, those heights will not be useful unless the cell is a terminal one.

The estimate for the systematic component given by a single tree corresponding to bootstrap sample $\mathcal{D}_b$ is

$$\hat{G}_b(\mathbf{x})=\sum_{\mathbf{t}_{1:l}\in T_b(\Theta_{1:k_b})}\hat{\beta}_{\mathbf{t}_l}1\{\mathbf{x}\in\mathbf{t}_l\}. \tag{9}$$

And the random forests estimate as a bagging of $\{\hat{G}_b\}_{b=1}^B$, is defined as

$$\hat{G}(\mathbf{x})=\frac{1}{B}\sum_{b=1}^{B}\hat{G}_b(\mathbf{x}). \tag{10}$$

Algorithm 1 below provides the modified version of the original Breiman's random forests algorithm with the sample CART-split criterion replaced by (8), and the majority vote replaced by the estimate for the systematic component as output. `minsize`, `maxdepth`, and `maxnode` characterize the stopping rules, which serve as further tuning parameters to avoid overfitting. In particular, if a cell contains less than `minsize` number of observations, or if it is already cut `maxdepth` times, or if a tree has already `maxnode` number of cells, then the tree will stop growing.

3.3 Model Selection

An important characteristic of random forests involves the utilization of out-of-bag (OOB) samples, which are the data points from the training set that are not included in each bootstrap sample. We will describe how OOB samples can be used for validation under the framework of BCM. Suppose B trees are already grown. For each observation $(y_i, \mathbf{x}_i)$ in the training set, calculate the systematic component predictor by averaging the predictions made by the trees for which $(y_i, \mathbf{x}_i)$ was not part of their corresponding bootstrap samples. More formally, for each $(y_i, \mathbf{x}_i) \in \mathcal{D}$, we calculate

$$\tilde{G}(\mathbf{x}_i) = \frac{1}{\#(\{b : (y_i, \mathbf{x}_i) \notin \mathcal{D}_b\} \cap \{1, ..., B\})} \sum_{\substack{b \in \{b : (y_i, \mathbf{x}_i) \notin \mathcal{D}_b\} \\ b \in \{1, ..., B\}}} \hat{G}_b(\mathbf{x}_i) \tag{11}$$

where $\hat{G}_b(\mathbf{x}_i)$ is defined in (9).

The generalization error is then

$$\text{err}(\gamma) := -\sum_{i=1}^{n} \Big[y_i \ln F(\tilde{G}(\mathbf{x}_i)) + (1 - y_i) \ln(1 - F(\tilde{G}(\mathbf{x}_i))) \Big] \tag{12}$$

where γ represents the set of hyperparameters to be tuned. Therefore, unlike many other nonlinear estimators, validation can be performed in random forests during the training process. As usual, it is done by choosing γ that minimizes the OOB generalization error over a set of different γ's. An OOB error estimate yields similar results to K-fold cross-validation (Hastie et al., 2009; Ljumović and Klar, 2015). These aspects suggest that OOB error can serve as an effective alternative for model selection.

Algorithm 1: Random Forests for BCM

```
Input: Data $\mathcal{D} = (\mathcal{Y}, \mathcal{X}) = \{y_i, \mathbf{x}_i\}_{i=1}^n$, number of bootstrap samples $B$, column
       subsampling size $m$, minsize, maxdepth, maxnode.
for $b = 1, ..., B$ do
    Draw $n$ points uniformly in $\mathcal{D}$, with replacement. Denote the bootstrap sample
     by $\mathcal{D}_b = (\mathcal{Y}_b, \mathcal{X}_b)$.
    Initialize $\mathcal{P} \leftarrow (\mathcal{D}_b)$, the list of root cells containing the entire bootstrap sample.
    Initialize $\mathcal{P}_{final} \leftarrow \emptyset$.
    while $\mathcal{P} \neq \emptyset$ do
        $cell \leftarrow$ the first element of $\mathcal{P}$.
        if cell contains less than maxsize observations, or if the depth of cell equals
         maxdepth, or if all $\boldsymbol{x}_i$ or all $y_i$ in cell are identical then
            Remove $cell$ from the list $\mathcal{P}$.
            $\mathcal{P}_{final} \leftarrow Concatenate(\mathcal{P}_{final}, cell)$.
        else
            Draw $m$ out of $p$ covariates uniformly, without replacement.
            Create a set $\mathcal{S}$ of all possible splitting rules based on the selected $m$
             covariates only.
            Select the best split by solving the problem given in (8).
            Cut $cell$ according to the best split. Denote the resulting cells as $cell_L$
             and $cell_R$. Store the corresponding $\hat{\beta}_L$ and $\hat{\beta}_R$.
            Remove $cell$ from the list $\mathcal{P}$.
            $\mathcal{P} \leftarrow Concatenate(\mathcal{P}, cell_L, cell_R)$.
        end
        if cardinality of $\mathcal{P} \cup \mathcal{P}_{final} \geqslant$ maxnode then
            $\mathcal{P}_{final} \leftarrow Concatenate(\mathcal{P}_{final}, \mathcal{P})$.
            $\mathcal{P} \leftarrow \emptyset$.
        end
    end
    Get the estimate $\hat{G}_b(\mathbf{x}) = \sum_{cell \in \mathcal{P}_{final}} \hat{\beta}_{cell} 1\{\mathbf{x} \in cell\}$.
end
Output: The estimate for the systematic component is $\hat{G} = \frac{1}{B} \sum_{b=1}^{B} \hat{G}_b$.
```

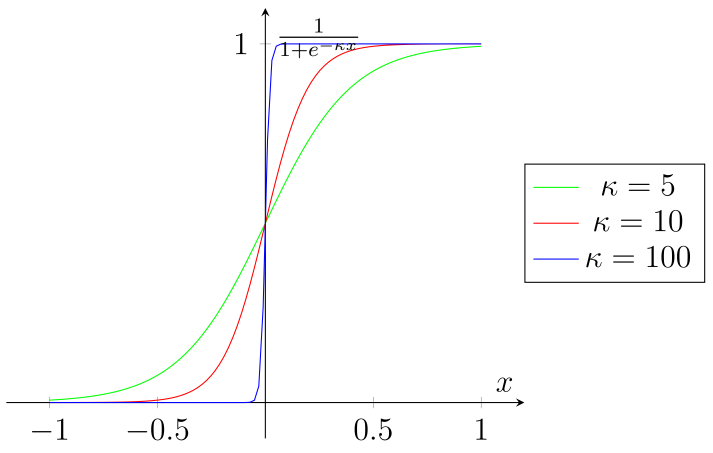


Figure 2: Sigmoid functions

4 Monte Carlo Experiments

To test the performance of the proposed method, we consider three high-dimensional specifications and compare the results with linear probit and the kernelized probit proposed by Yan (2023). We also compare the results with the original random forests using the Gini index for classification.

The BCM model is as given in (3) in section 3.1, in which $U \sim_{iid} \mathbb{N}(0,1)$ and is independent of $\mathbf{X}$. The covariates for all specifications are $\mathbf{X} = (X_1, ..., X_{10})'$ with each $X_j \sim_{iid} \text{Unif}[-1,1]$. The three specifications are given by

$$(1): \quad G_0(\mathbf{X}) = \sum_{j=1}^{5} \beta_j \sin(\pi X_j X_{j+5})$$

$$(2): \quad G_0(\mathbf{X}) = \sum_{j=1}^{5} \beta_j \frac{1}{(1+e^{-\kappa X_j})(1+e^{-\kappa X_{j+5}})}$$

$$(3): \quad G_0(\mathbf{X}) = \sum_{j=1}^{30} \beta_j 1\{\mathbf{X} \in \mathbf{t}_j\}$$

$$(4): \quad G_0(\mathbf{X}) = \sum_{j=1}^{5} \beta_j 1\{X_j > \tau_j\} + \sum_{j=6}^{10} \beta_j X_j$$

where $(\beta_1, ..., \beta_5) = (0.81, 0.91, 0.13, 0.91, 0.63)$ and $(1, -1.75, 0.5, -1.25, 1.5)$ for the first and second specification, respectively. In (4), $\beta = (0.05, -0.1, 0.85, 0.55, 0.06, -0.44, 0.28, 0.99, 0.12, -0.02)$ and $\tau = (-0.16, 0.32, 0.9, -0.34, -0.06)$. For the second specification, we consider a linear combination of interactions of sigmoid functions $f(x) = \frac{1}{1+e^{-\kappa x}}$ whose steepness increases

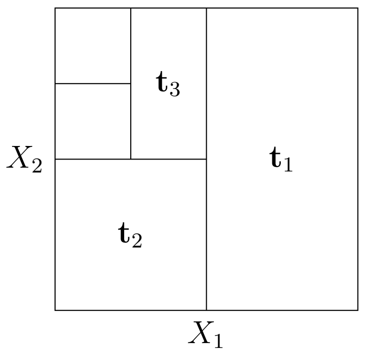


Figure 3: An example of binary partitioning with two covariates

with κ (see Figure 2 for a visual illustration). It is expected that both sin waves and sigmoid functions can be consistently estimated by the kernelized probit approach with Gaussian kernel given by $k(v, w) = \exp(-\|v - w\|^2/2\sigma^2)$ since the reproducing kernel Hilbert space with Gaussian kernel is dense in the space of continuous function equipped with supremum norm defined on a compact Euclidean space. We expect that the modified random forests method is robust against discontinuous functions, and thus will perform well for the third specification. For the second specification, random forests probit is expected to have better performance as κ increases.

In the third specification, β is drawn from Unif$[-2, 2]$ and is fixed across all experiments. Figure 3 demonstrates how the indicator functions are generated. In the j-th cut, a cell and a variable are selected at random and the cell is divided into two cells in the middle along the chosen variable. In Figure 3, $X_1, X_2 \in [-1, 1]$, $\mathbf{t}_0 = [-1, 1]^2$, $\mathbf{t}_1 = (0, 1] \times [-1, 1]$, $\mathbf{t}_2 = [-1, 0]^2$, $\mathbf{t}_3 = (-1/2, 0] \times (0, 1]$, and the sequence continues in this manner. The outcome of this procedure is a highly discontinuous function. Consequently, we expect that neither linear nor kernelized methods can consistently estimate this systematic component.

For each specification, we generate a test set with size 10000, and a training set with size `ntrain`$\in \{200, 500, 1000, 2000, 5000, 10000\}$. As for now, each Monte Carlo simulation has 10 replications. Based on our experiments, we find that growing 200 the number of trees ($B = 200$) would be sufficient for the OOB generalization error to stabilize. As an illustration, Figure 4 plots the OOB generalization error for the second specification with $\kappa = 10$ and `ntrain`=2000, with respect to B. The validation error stabilizes after about 200 trees. For this reason as well as for computational consideration, we set $B = 200$ throughout

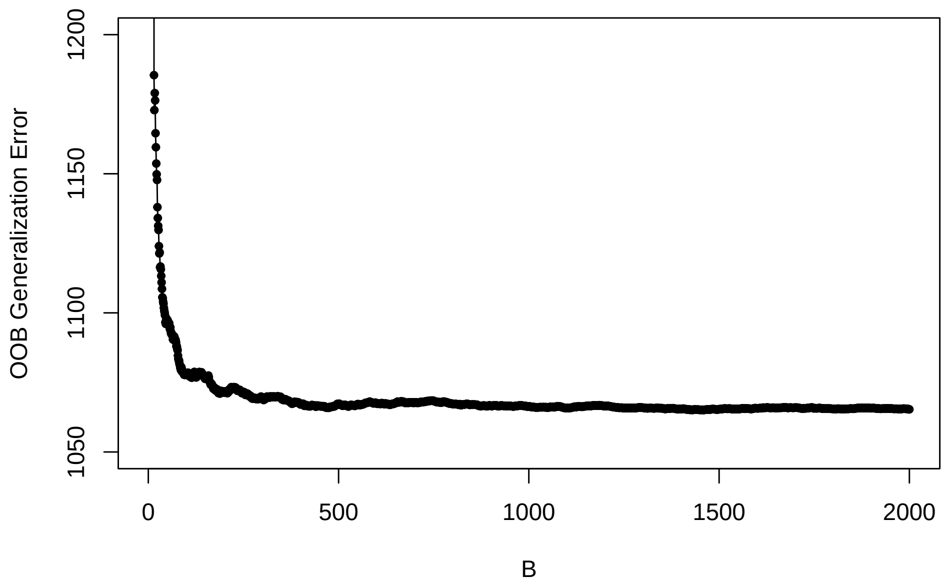


Figure 4: OOB generalization error with respect to number of bootstrap samples

this section.

The metrics for methods evaluation are chosen to be the sample analog of the mean error of the systematic component $\mathbb{E}|\hat{G}(\mathbf{X}) - G_0(\mathbf{X})|$, the mean error of the CCP $\mathbb{E}|\hat{p}(\mathbf{X}) - p_0(\mathbf{X})|$, and the misclassification error rate $\mathbb{P}\{Y \neq 1\{\hat{p}(\mathbf{X}) > 1/2\}\}$.

The outcomes of the simulations are detailed in Table 1, 2, 3, and 4, aligning with our initial expectations. For the first specification, the kernelized probit outperforms the random forests probit. Conversely, for the other specifications, the random forests probit is competitive, and it performs better when dealing with more discontinuous underlying functions. Although the original random forests model is quite effective for classification tasks, it falls short of accurately estimating the CCP.

Table 1: A Comparison of performance across four methods

	$\mathbb{E}\lvert\hat{G}(\mathbf{X}) - G_0(\mathbf{X})\rvert$			$\mathbb{E}\lvert\hat{p}(\mathbf{X}) - p_0(\mathbf{X})\rvert$				$\mathbb{P}\{Y \neq 1\{\hat{p}(\mathbf{X}) > 1/2\}\}$				
ntrain	Probit	KPB	RFP	Probit	KPB	RFP	RF	Probit	KPB	RFP	RF	True
Specification 1												
200	0.8304	0.8708	0.8071	0.2590	0.2732	0.2490	0.2749	0.4815	0.4931	0.4348	0.4243	0.2449
500	0.8304	0.7834	0.7423	0.2590	0.2403	0.2253	0.2711	0.4815	0.4248	0.4042	0.3894	0.2449
1000	0.8304	0.6435	0.6752	0.2590	0.1886	0.1997	0.2688	0.4815	0.3526	0.3423	0.3323	0.2449
2000	0.8304	0.5241	0.6406	0.2590	0.1489	0.1865	0.2668	0.4815	0.3101	0.3183	0.3158	0.2449
5000	0.8304	0.3467	0.6127	0.2590	0.0917	0.1757	0.2645	0.4815	0.2694	0.2917	0.2918	0.2449
10000	0.8304	0.3307	0.6062	0.2590	0.0869	0.1728	0.2629	0.4815	0.2665	0.2770	0.2769	0.2449

Notes: KPB, RFP, and RF represent the kernelized probit, the proposed random forests probit, and the original Brieman's random forests with the Gini index, respectively. True denotes the misclassification error rate using the true CCP.

Table 2: A Comparison of performance across four methods (cont.)

	$\mathbb{E}\lvert\hat{G}(\mathbf{X}) - G_0(\mathbf{X})\rvert$			$\mathbb{E}\lvert\hat{p}(\mathbf{X}) - p_0(\mathbf{X})\rvert$				$\mathbb{P}\{Y \neq 1\{\hat{p}(\mathbf{X}) > 1/2\}\}$				
ntrain	Probit	KPB	RFP	Probit	KPB	RFP	RF	Probit	KPB	RFP	RF	True
					Specification 2 ($\kappa = 10$)							
200	0.5359	0.6098	0.6024	0.1493	0.1710	0.1690	0.2767	0.3015	0.3206	0.3203	0.3175	0.2467
500	0.5359	0.5584	0.5283	0.1493	0.1553	0.1439	0.2757	0.3015	0.3057	0.2917	0.2933	0.2467
1000	0.5359	0.4912	0.4484	0.1493	0.1351	0.1195	0.2734	0.3015	0.2899	0.2764	0.2768	0.2467
2000	0.5359	0.4253	0.3965	0.1493	0.1150	0.1029	0.2713	0.3015	0.2766	0.2685	0.2672	0.2467
5000	0.5359	0.3790	0.3253	0.1493	0.1011	0.0824	0.2697	0.3015	0.2678	0.2619	0.2644	0.2467
10000	0.5359	0.3658	0.3120	0.1493	0.0969	0.0776	0.2684	0.3015	0.2633	0.2573	0.2591	0.2467
					Specification 2 ($\kappa = 100$)							
200	0.7036	0.7570	0.7418	0.1851	0.2021	0.1958	0.2952	0.3137	0.3334	0.3205	0.3206	0.2380
500	0.7036	0.7234	0.6582	0.1851	0.1919	0.1686	0.2931	0.3137	0.3210	0.2898	0.2897	0.2380
1000	0.7036	0.6712	0.5625	0.1851	0.1759	0.1382	0.2904	0.3137	0.3033	0.2656	0.2666	0.2380
2000	0.7036	0.5972	0.4861	0.1851	0.1535	0.1151	0.2878	0.3137	0.2849	0.2574	0.2581	0.2380
5000	0.7036	0.5647	0.4050	0.1851	0.1437	0.0925	0.2861	0.3137	0.2786	0.2501	0.2510	0.2380
10000	0.7036	0.5557	0.3581	0.1851	0.1406	0.0794	0.2838	0.3137	0.2765	0.2473	0.2488	0.2380

Notes: KPB, RFP, and RF represent the kernelized probit, the proposed random forests probit, and the original Brieman's random forests with the Gini index, respectively. True denotes the misclassification error rate using the true CCP.

Table 3: A Comparison of performance across four methods (cont.)

	$\mathbb{E}\lvert\hat{G}(\mathbf{X}) - G_0(\mathbf{X})\rvert$			$\mathbb{E}\lvert\hat{p}(\mathbf{X}) - p_0(\mathbf{X})\rvert$				$\mathbb{P}\{Y \neq 1\{\hat{p}(\mathbf{X}) > 1/2\}\}$				
ntrain	Probit	KPB	RFP	Probit	KPB	RFP	RF	Probit	KPB	RFP	RF	True
					Specification 3							
200	1.0577	0.9971	0.9575	0.3009	0.2794	0.2656	0.3410	0.3975	0.3725	0.3521	0.3447	0.1761
500	1.0577	0.9828	0.8749	0.3009	0.2743	0.2371	0.3425	0.3975	0.3725	0.3153	0.3120	0.1761
1000	1.0577	0.9711	0.8048	0.3009	0.2703	0.2127	0.3436	0.3975	0.3589	0.2778	0.2782	0.1761
2000	1.0577	0.9489	0.6699	0.3009	0.2615	0.1708	0.3428	0.3975	0.3408	0.2428	0.2384	0.1761
5000	1.0577	0.9085	0.5415	0.3009	0.2477	0.1314	0.3419	0.3975	0.3273	0.2076	0.2046	0.1761
10000	1.0577	0.8861	0.4045	0.3009	0.2396	0.0933	0.3428	0.3975	0.3169	0.1892	0.1895	0.1761

Notes: KPB, RFP, and RF represent the kernelized probit, the proposed random forests probit, and the original Brieman's random forests with the Gini index, respectively. True denotes the misclassification error rate using the true CCP.

Table 4: A Comparison of performance across four methods (cont.)

	$\mathbb{E}\lvert\hat{G}(\mathbf{X}) - G_0(\mathbf{X})\rvert$			$\mathbb{E}\lvert\hat{p}(\mathbf{X}) - p_0(\mathbf{X})\rvert$				$\mathbb{P}\{Y \neq 1\{\hat{p}(\mathbf{X}) > 1/2\}\}$				
ntrain	Probit	KPB	RFP	Probit	KPB	RFP	RF	Probit	KPB	RFP	RF	True
Specification 4												
200	0.4373	0.4126	0.4112	0.1369	0.1182	0.1237	0.2641	0.3155	0.3074	0.3135	0.3131	0.2718
500	0.4373	0.3213	0.3199	0.1369	0.0904	0.0944	0.2568	0.3155	0.2909	0.2969	0.3018	0.2718
1000	0.4373	0.2966	0.2751	0.1369	0.0838	0.0810	0.2543	0.3155	0.2876	0.2937	0.2956	0.2718
2000	0.4373	0.2825	0.2464	0.1369	0.0792	0.0711	0.2525	0.3155	0.2840	0.2849	0.2885	0.2718
5000	0.4373	0.2677	0.2147	0.1369	0.0765	0.0617	0.2511	0.3155	0.2868	0.2831	0.2858	0.2718
10000	0.4373	0.2494	0.2152	0.1369	0.0708	0.0628	0.2494	0.3155	0.2855	0.2813	0.2835	0.2718

Notes: KPB, RFP, and RF represent the kernelized probit, the proposed random forests probit, and the original Brieman's random forests with the Gini index, respectively. True denotes the misclassification error rate using the true CCP.

5 Conclusion

In this paper, we propose a new estimation procedure for nonparametric BCMs using random forests, which makes use of the negative log-likelihood function instead of the impurity measures that are used in the traditional random forests, such as the Gini index and cross-entropy. The parameters in the latent utility function are estimated using the recursive partitioning maximum likelihood estimation. Simulation studies suggest that the modified random forests estimator works well in finite samples, and it performs better than the kernelized probit approach in estimating discontinuous or nearly discontinuous functions. A natural extension in future research is to allow for a fully nonparametric model, without the assumption that the distribution function of the error component is known. Another way to go is to estimate BCMs with boosting trees, another popular tree-based machine-learning method. A deeper understanding of fundamental programming languages such as C and C++ is essential to reduce computational burden.

References

Amit, Y. and D. Geman (1997). Shape quantization and recognition with randomized trees. *Neural computation 9*(7), 1545–1588.

Athey, S. and G. W. Imbens (2019). Machine learning methods that economists should know about. *Annual Review of Economics 11*, 685–725.

Bhattacharya, D. (2021). The empirical content of binary choice models. *Econometrica 89*(1), 457–474.

Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research 13*(1), 1063–1095.

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

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

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). Classification and regression trees. *Pacific Grove, Wadsworth.*

Bühlmann, P. and B. Yu (2002). Analyzing bagging. *The annals of Statistics 30*(4), 927–961.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins (2018). Double/debiased machine learning for treatment and structural parameters.

Chi, C.-M., P. Vossler, Y. Fan, and J. Lv (2022). Asymptotic properties of high-dimensional random forests. *The Annals of Statistics 50*(6), 3415–3438.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine learning 20*, 273–297.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning 40*, 139–157.

Farrell, M. H., T. Liang, and S. Misra (2021). Deep neural networks for estimation and inference. *Econometrica 89*(1), 181–213.

Freund, Y. and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*(1), 119–139.

Genuer, R., J.-M. Poggi, and C. Tuleau (2008). Random forests: some methodological insights. *arXiv preprint arXiv:0811.3619*.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*, Volume 2. Springer.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence 20*(8), 832–844.

Lechner, M. and G. Okasa (2019). Random forest estimation of the ordered choice model. *arXiv preprint arXiv:1907.02436*.

Ljumović, M. and M. Klar (2015). Estimating expected error rates of random forest classifiers: A comparison of cross-validation and bootstrap. In *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, pp. 212–215. IEEE.

Matzkin, R. L. (1992). Nonparametric and distribution-free estimation of the binary threshold crossing and the binary choice models. *Econometrica: Journal of the Econometric Society*, 239–270.

Mullainathan, S. and J. Spiess (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives 31*(2), 87–106.

Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of economic perspectives 28*(2), 3–28.

Wager, S. and S. Athey (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association 113*(523), 1228–1242.

Yan, G. (2023). *Three Essays on the Econometrics of Machine Learning*. Indiana University.

Appendix

A. Implementation of L1 Regularization

In this section, we propose a simple and automatic way of implementing L1 regularization in problem (8), which does not require the coordinate descent algorithm. In particular, we deal with

$$\max_{\beta_L}\{n_{1L}\ln(F(\beta_L))+n_{0L}\ln(1-F(\beta_L))-\lambda_n|\beta_L|\}+\max_{\beta_R}\{n_{1R}\ln(F(\beta_R))+n_{0R}\ln(1-F(\beta_R))-\lambda_n|\beta_R|\} \tag{13}$$

Note that if $n_{1L} > n_{0L}$, then $\ln F(\hat{\beta}_L)$ should be larger than $\ln(1-F(\hat{\beta}_L))$, which in turn requires $F(\hat{\beta}_L)$ to be greater than $\frac{1}{2}$ and thus $\hat{\beta}_L > 0$. To see this, suppose $n_{1L} > n_{0L}$ but $\ln F(\tilde{\beta}_L) < \ln(1-F(\tilde{\beta}_L))$ and $\tilde{\beta}_L < 0$. Then, we can simply select $\hat{\beta}_L = -\tilde{\beta}_L$ to improve $n_{1L}\ln(F(\beta_L))+n_{0L}\ln(1-F(\beta_L))$ while keeping the penalty term unchanged, indicating that such $\tilde{\beta}_L$ cannot be optimal. Similarly, if $n_{1L} > n_{0L}$, then $\hat{\beta}_L < 0$. Finally, for the case when $n_{1L} = n_{0L}$, $\hat{\beta}_L = 0$ since $\ln(F(\beta_L))+\ln(1-F(\beta_L))$ attain its global maximum at $F(\beta_L) = \frac{1}{2}$, or $\beta_L = 0$.

The reasoning for β_R follows the same logic as outlined above. Consequently, $|\beta_L| + |\beta_R|$ can be replaced by

$$\operatorname{sgn}\left(\frac{1}{2} - p_L\right)\beta_L + \operatorname{sgn}\left(\frac{1}{2} - p_R\right)\beta_R \tag{14}$$

where $p_L := \frac{n_{1L}}{n_L}$ and $p_L := \frac{n_{1R}}{n_R}$. The problem now can be solved by computing the gradient and applying the first-order conditions as usual. Our simulations indicate that both L1 and L2 regularizations produce comparable outcomes, with the execution speeds of the algorithms being nearly identical.

B. Relation to Cross-Entropy

It should be noted that if the regularization term is omitted from (8), then the problem can be solved analytically, yielding solutions given by $\hat{\beta}_L = F^{-1}(p_L)$ and $\hat{\beta}_R = F^{-1}(p_R)$, where $p_L := n_{1L}/n_L$ and $p_R := n_{1R}/n_R$ represent the proportions of label 1 in the left and right child nodes, respectively. A straightforward computation reveals that after the substitution of optimal solutions in the inner maximization problem, the criterion for splitting simplifies to

$$\max_{j\in\Theta,c}\left\{n_L(p_L\ln(p_L) + (1-p_L)\ln(1-p_L)) + n_R(p_R\ln(p_R) + (1-p_R)\ln(1-p_R))\right\} \tag{15}$$

which essentially is the sample CART-split criterion employing cross-entropy as the impurity measure, as given in (1). However, to effectively estimate the coefficients in the systematic component G under the framework of BCM, it is crucial to introduce a regularization term into the split criterion. Primarily, this helps prevent the estimates $\hat{\beta}_L$ and $\hat{\beta}_R$ from becoming infinite when a node consists solely of one class, as p_L or p_R would be either 1 or 0. Secondly, it ensures that the estimates do not become exceedingly large in almost pure nodes. Beyond bootstrap aggregation, adding a penalty term serves as another mechanism for stabilizing the estimates. Our simulations suggest that applying this regularization technique consistently throughout the entire process of recursive partitioning yields better outcomes than applying it only to completely pure child nodes.

With the introduction of the regularization term, the problem no longer possesses closed-form solutions, and (8) diverges from the penalized sample CART-split criterion with cross-entropy impurity measure.

C. Variable Importance and Average Partial Effects

As tree-based approaches, random forests have their interpretability in the relative importance of the covariates. There are many ways of measuring variable importance and we defer the exploration of their comparisons to future research questions. We present the one that is introduced in Hastie et al. (2009), which makes use of the OOB samples to measure the prediction strength of each covariate. For covariate j, when the b-th tree is grown, the prediction accuracy of the OOB samples is recorded. Then the values for the j-th covariate are randomly permuted in the OOB samples, and the prediction accuracy is computed again. Then the decrease in accuracy for the b-th tree can be calculated. The importance of the j-th covariate is the average decrease in accuracy computed in this way, over all B trees. Intuitively, the variables with higher predicting power will have larger variable importance since a random permutation will cause a greater loss in prediction accuracy.

Another way of interpretation that is more interesting to economists is the analysis of average partial effects of the conditional choice probability $p_0(X) = F(G(X))$. Since the estimates using random forests are simple functions, the average partial effect with respect to j-th covariate is simply given by $\mathbb{E}\Delta F(G(X_j, X_{-j}))$, where Δ denotes the difference. Its sample analog is given by $\frac{1}{n}\sum_{i=1}^{n}(F(\hat{G}(X_j^1, X_{i,-j})) - F(\hat{G}(X_j^0, X_{i,-j})))$, where X_j^0 and X_j^1 denote the values before and after the change, respectively.