

Random Forests, Decision Trees, and Categorical Predictors: The “Absent Levels” Problem

Timothy C. Au
Google Inc.
timau@google.com

Abstract

One of the advantages that decision trees have over many other models is their ability to natively handle categorical predictors without having to first transform them (e.g., by using one-hot encoding). However, in this paper, we show how this capability can also lead to an inherent “absent levels” problem for decision tree based algorithms that, to the best of our knowledge, has never been thoroughly discussed, and whose consequences have never been carefully explored. This predicament occurs whenever there is indeterminacy in how to handle an observation that has reached a categorical split which was determined when the observation’s level was absent during training. Although these incidents may appear to be innocuous, by using Leo Breiman and Adele Cutler’s random forests **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) as motivating case studies, we show how overlooking the absent levels problem can systematically bias a model. Afterwards, we discuss some heuristics that can possibly be used to help mitigate the absent levels problem and, using three real data examples taken from public repositories, we demonstrate the superior performance and reliability of these heuristics over some of the existing approaches that are currently being employed in practice due to oversights in the software implementations of decision tree based algorithms. Given how extensively these algorithms have been used, it is conceivable that a sizable number of these models have been unknowingly and seriously affected by this issue—further emphasizing the need for the development of both theory and software that accounts for the absent levels problem.

1 Introduction

Since its introduction in Breiman (2001), random forests have enjoyed much success as the most widely used decision tree based machine learning algorithm. However, despite their popularity and apparent simplicity, random forests have proven to be very difficult to analyze. Indeed, the basic mathematical properties of the algorithm are still not completely understood, and theoretical investiga-

tions have often had to rely on simplifying assumptions and variations of the standard framework in order to make the analysis more tractable. Examples of recent work along these lines include Biau et al. (2008), Biau (2012), and Denil et al. (2014).

One of the advantages that decision trees have over many other models is their ability to natively handle categorical predictors without having to first transform them (e.g., by using one-hot encoding). However, in this paper, we show how this capability can also lead to an inherent “absent levels” problem for decision tree based algorithms that, to the best of our knowledge, has never been thoroughly discussed, and whose consequences have never been carefully explored. This predicament, which occurs whenever there is indeterminacy in how to handle an observation that has reached a categorical split which was determined when the observation’s level was absent during training, can arise in three different ways:

1. The unordered levels are present in the population but, due to sampling variability, are absent in the training set.
2. The unordered levels are present in the original training set but, due to bagging, are absent in an individual tree’s bootstrapped sample of the training set.
3. The unordered levels are present in an individual tree’s training set but, due to a series of earlier node splits, are absent in certain branches of the tree.

These occurrences subsequently result in situations where the observations with absent levels are unsure of how to proceed further down the tree—an intrinsic problem for decision tree based algorithms that has seemingly been overlooked in both the theoretical literature and in much of the software that implements these models.

Although these incidents may appear to be innocuous, by using Leo Breiman and Adele Cutler’s random forests **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) as motivating case studies, we show how overlooking the absent levels problem can systematically bias a model.¹ Afterwards, we discuss some heuristics that can possibly be used to help mitigate the absent levels problem and, using three real data examples taken from public repositories, we demonstrate the superior performance and reliability of these heuristics over some of the existing approaches that are currently being employed in practice due to oversights in the software implementations of decision tree based algorithms. Given how extensively these algorithms have been used, it is conceivable that a sizable number of these models have been unknowingly and seriously affected by this issue—further emphasizing the need for the development of both theory and software that accounts for the absent levels problem.

¹Breiman and Cutler’s **FORTRAN** code is available online at:
<https://www.stat.berkeley.edu/~breiman/RandomForests/>

This paper is organized as follows. In Section 2, we introduce notation and give an overview of the random forests algorithm, although the more experienced reader may only need to review how splits are determined, which is covered in Sections 2.1.1 and 2.1.2. Then, in Section 3, we investigate the potential consequences that can emerge when the absent levels problem is overlooked, and we motivate our discussions by documenting the systematic biases that absent levels can cause in Breiman and Cutler’s `FORTAN` code and the `randomForest` R package (Liaw and Wiener, 2002). Although a theoretical analysis of potential solutions to the absent levels problem is beyond the scope of this paper, in Section 4, we consider some heuristics that may be able to help to mitigate the issue. Afterwards, in Section 5, we present three real data examples taken from public repositories that demonstrate how the treatment of absent levels can dramatically alter a model’s performance in practice. Based on these results and our own personal experiences, in Section 6, we conclude by making some final recommendations on how to temporarily handle the absent levels problem until a more robust theoretical solution is found.

2 Background

In this section, we give an overview of the random forests algorithm. Here, the more experienced reader may only need to review Sections 2.1.1 and 2.1.2, which cover how splits are determined.

2.1 Classification and Regression Trees (CART)

We begin by discussing the Classification and Regression Trees (CART) methodology since the random forests algorithm uses a slightly modified version of CART to construct the individual decision trees in its ensemble. For a more comprehensive overview of CART, we refer the reader to Breiman et al. (1984) or Hastie et al. (2009).

Suppose that we have a training set of N observations,

$$z_i = (x_i, y_i), \quad i = 1, 2, \dots, N,$$

where $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ is observation i ’s p -dimensional feature vector and y_i is observation i ’s response. Given this training set, CART is a greedy recursive binary partitioning algorithm that recursively splits a larger subset of the training data \mathcal{D}_M (the “mother node”) into two smaller subsets \mathcal{D}_L and \mathcal{D}_R (the “left” and “right” daughter nodes, respectively), with a model being fit to each subset of the training data. Each iteration of this splitting process, which can be referred to as “growing the tree,” is accomplished by using a simple decision rule that is characterized by a “splitting variable” $j \in \{1, 2, \dots, p\}$ and a “splitting criterion” set S_j that defines the subset of j ’s support that gets sent to the left daughter node \mathcal{D}_L . In particular, any splitting variable and splitting criterion pair (j, S_j) divides a mother node \mathcal{D}_M into the two daughter nodes

that are defined as follows:

$$\begin{aligned}\mathcal{D}_L(j, S_j) &= \{i \in \mathcal{D}_M \mid x_{ij} \in S_j\}, \\ \mathcal{D}_R(j, S_j) &= \{i \in \mathcal{D}_M \mid x_{ij} \in S_j^C\}.\end{aligned}\tag{1}$$

This recursive binary partitioning procedure is continued until some stopping rule is reached—a tuning parameter that is typically specified by placing some constraint on the maximum number of training observations allowed in each terminal node. Afterwards, in order to help guard against overfitting, the tree can then be “pruned”—although we will not discuss this further as the trees that are grown in random forests have not traditionally been pruned (Breiman, 2001). Inferences can then be carried out on an observation by first sending it down the tree according to the set of decision rules, and then using the model that was fit in the furthest node of the tree that the observation was able to reach.

In order to grow a tree, the CART algorithm will select the optimal splitting variable and splitting criterion pair $(j^*, S_{j^*}^*)$ that minimizes some measure of “node impurity” in the resulting left and right daughter nodes that are created with respect to the pair as defined in equation (1). Here, the specific node impurity measure that is being minimized will depend on whether the tree is being used for regression or classification.

In a regression tree, the responses in each node \mathcal{D}_n are modeled using a constant c_n which, under a squared error loss, is estimated by the mean of the training responses that are in \mathcal{D}_n :

$$\hat{c}_n = \text{ave}(y_i \mid i \in \mathcal{D}_n).\tag{2}$$

Therefore, when splitting a mother node \mathcal{D}_M in a regression tree, the CART algorithm will minimize the resulting squared error node impurity measure in the two daughter nodes \mathcal{D}_L and \mathcal{D}_R that are created with respect to a (j, S_j) pair:

$$\min_{j, S_j} \left\{ \min_{c_L} \sum_{i \in \mathcal{D}_L(j, S_j)} (y_i - c_L)^2 + \min_{c_R} \sum_{i \in \mathcal{D}_R(j, S_j)} (y_i - c_R)^2 \right\},\tag{3}$$

where the inner minimization for any choice of the pair (j, S_j) is solved by

$$\hat{c}_L = \text{ave}(y_i \mid i \in \mathcal{D}_L(j, S_j)) \quad \text{and} \quad \hat{c}_R = \text{ave}(y_i \mid i \in \mathcal{D}_R(j, S_j)),$$

and where $\mathcal{D}_L(j, S_j)$ and $\mathcal{D}_R(j, S_j)$ are defined as in equation (1).

Meanwhile, in a classification tree where the response is categorical with K possible classes which are indexed by the set $\mathcal{K} = \{1, 2, \dots, K\}$, denote the sample response class proportions of the training observations that are in node \mathcal{D}_n as

$$\hat{p}_{nk} = \frac{1}{N_n} \sum_{i \in \mathcal{D}_n} I(y_i = k), \quad k \in \mathcal{K},\tag{4}$$

where $I(\cdot)$ is the indicator function and

$$N_n = \# \{i \in \mathcal{D}_n\}\tag{5}$$

is the number of training observations in node \mathcal{D}_n . Observations in \mathcal{D}_n will then be classified to the majority class

$$\arg \max_{k \in \mathcal{K}} \hat{p}_{nk}, \quad (6)$$

and the Gini index,

$$G_n = \sum_{k=1}^K \hat{p}_{nk}(1 - \hat{p}_{nk}), \quad (7)$$

provides one way of quantifying the node impurity in \mathcal{D}_n , where \hat{p}_{nk} is as defined in equation (4). Consequently, when splitting a mother node \mathcal{D}_M in a classification tree, the CART algorithm will minimize the resulting weighted Gini index in the two daughter nodes \mathcal{D}_L and \mathcal{D}_R that are created with respect to a (j, S_j) pair as in equation (1):

$$\min_{j, S_j} \left\{ \frac{N_L \cdot G_L + N_R \cdot G_R}{N_L + N_R} \right\}, \quad (8)$$

where N_L and N_R are as defined in equation (5), and where G_L and G_R are as defined in equation (7).

Despite their differences, for both regression and classification trees, the actual optimal splitting variable and splitting criterion pair $(j^*, S_{j^*}^*)$ that is chosen to split the mother node \mathcal{D}_M is the (j, S_j) pair that minimizes the node impurity measure across all possible pairs subject to some greedy search constraints. However, the manner in which the optimal splitting criterion S_j^* is determined for a predictor j will depend on whether j is an ordered or a categorical variable.

2.1.1 Splitting on an Ordered Predictor

An ordered predictor j 's splitting criterion S_j is characterized by a numeric split point s_j that defines the half-line $S_j = \{x \in \mathbb{R} \mid x \leq s_j\}$. Consequently, by equation (1), a (j, S_j) pair will divide the mother node \mathcal{D}_M into the left and right daughter nodes that are defined by

$$\begin{aligned} \mathcal{D}_L(j, S_j) &= \{i \in \mathcal{D}_M \mid x_{ij} \leq s_j\}, \\ \mathcal{D}_R(j, S_j) &= \{i \in \mathcal{D}_M \mid x_{ij} > s_j\}. \end{aligned}$$

Therefore, for an ordered predictor j , minimizing either of the node impurity measures given in equations (3) or (8) is straightforward, and the optimal splitting criterion S_j^* for an ordered predictor j can be greedily found by searching through all of the “midpoints” between any two consecutive training x_{ij} values that are in the mother node \mathcal{D}_M .

2.1.2 Splitting on a Categorical Predictor

Consider a categorical predictor j with Q possible unordered levels which are indexed by the set $\mathcal{Q} = \{1, 2, \dots, Q\}$. Because it is only the levels of j that are

present in the mother node \mathcal{D}_M during training which contribute to the node impurity measures defined in equations (3) or (8), for simplicity of exposition and ease of notation, we assume in this section that all Q levels of j are present in the mother node \mathcal{D}_M during training.

For a categorical predictor j , the splitting criterion $S_j \subset \mathcal{Q}$ defines the subset of the levels that are sent to the left daughter node \mathcal{D}_L , while the complement set S_j^C defines the subset of the levels that are sent to the right daughter node \mathcal{D}_R . Consequently, there are $2^{Q-1} - 1$ non-redundant ways of partitioning the Q unordered levels into the two daughter nodes, making it computationally expensive to evaluate either of the node impurity measures given in equations (3) or (8) for every possible split when Q is large. However, this computation simplifies in certain situations.

In the case of a regression tree with a squared error node impurity measure, the optimal splitting criterion S_j^* can be determined by using results derived in Fisher (1958). Specifically, the mean of the training responses in the mother node \mathcal{D}_M is first calculated within each of j 's levels:

$$\gamma_j(q) = \text{ave}(y_i \mid i \in \mathcal{D}_M \text{ and } x_{ij} = q), \quad q \in \mathcal{Q}.$$

These categorical level means are then be used to define numeric ‘‘pseudo values’’ \tilde{x}_{ij} for every training observation in \mathcal{D}_M , where

$$\tilde{x}_{ij} = \gamma_j(x_{ij}), \quad i \in \mathcal{D}_M. \quad (9)$$

The optimal splitting criterion S_j^* for the categorical predictor j is then determined by doing an ordered split on these numeric pseudo values \tilde{x}_{ij} . That is, we can minimize the squared error node node impurity measure given in equation (3) with respect to the left and right daughter nodes that are defined as

$$\begin{aligned} \mathcal{D}_L(j, \tilde{s}_j) &= \{i \in \mathcal{D}_M \mid \tilde{x}_{ij} \leq \tilde{s}_j\}, \\ \mathcal{D}_R(j, \tilde{s}_j) &= \{i \in \mathcal{D}_M \mid \tilde{x}_{ij} > \tilde{s}_j\}, \end{aligned} \quad (10)$$

and where the optimal numeric pseudo split point \tilde{s}_j^* can be greedily found by searching through the midpoints between any two consecutive pseudo training values \tilde{x}_{ij} that are in the mother node \mathcal{D}_M .

Meanwhile, in the case of a classification tree and a weighted Gini index node impurity measure, whether the computation simplifies or not is dependent on the number of response classes. For the $K > 2$ multiclass classification context, no such simplification is possible, although several approximations have been proposed (Loh and Vanichsetakul, 1988). However, for the $K = 2$ binary classification situation, a similar procedure to the one that was described for regression trees can be employed where, in this case, the proportion of training observations in the mother node \mathcal{D}_M that belong to the $k = 1$ response class is instead first calculated within each of j 's levels:

$$\gamma_j(q) = \frac{\#\{i \in \mathcal{D}_M \mid x_{ij} = q \text{ and } y_i = 1\}}{\#\{i \in \mathcal{D}_M \mid x_{ij} = q\}}, \quad q \in \mathcal{Q}.$$

Then, just as in equation (9), these categorical level $k = 1$ response class proportions are used to define numeric pseudo values \tilde{x}_{ij} for every training observation in \mathcal{D}_M . And once again, the optimal splitting criterion S_j^* for the categorical predictor j is then determined by doing an ordered split on these numeric pseudo values \tilde{x}_{ij} in order to find the optimal pseudo numeric split point \tilde{s}_j^* that minimizes the weighted Gini index node impurity measure given by equation (8) for the daughter nodes that are defined as in equation (10). The proof that this gives the optimal split in a binary classification tree in terms of the weighted Gini index among all possible splits can be found in Breiman et al. (1984) and Ripley (1996).

For both the regression and binary classification situations, after the optimal numeric pseudo split point \tilde{s}_j^* has been determined, the corresponding optimal splitting criterion S_j^* for categorical predictor j can be expressed as follows:

- The unordered levels of j that are being sent *left* have means or $k = 1$ response class proportions that are *less than or equal to* \tilde{s}_j^* :

$$S_j^* = \{q \in \mathcal{Q} \mid \gamma_j(q) \leq \tilde{s}_j^*\}. \quad (11)$$

- The unordered levels of j that are being sent *right* have means or $k = 1$ response class proportions that are *greater than* \tilde{s}_j^* :

$$S_j^{*C} = \{q \in \mathcal{Q} \mid \gamma_j(q) > \tilde{s}_j^*\}. \quad (12)$$

As we shall later discuss in Section 3, equations (11) and (12) lead to systematic differences in the left and right daughter nodes when splitting on a categorical predictor in regression and binary classification trees—differences which can have significant implications for the absent levels problem.

2.2 Random Forests

Introduced in Breiman (2001), random forests are an ensemble learning method that corrects for each individual tree’s propensity to overfit the training set. This is accomplished through the use of bagging and a CART-like tree learning algorithm in order to build a large collection of “de-correlated” decision trees.

2.2.1 Bagging

Proposed in Breiman (1996a), bagging is an ensembling technique for improving the accuracy and stability of models. Given a training set Z with N observations, this is achieved by sampling with replacement from Z in order to generate B new bootstrapped training sets Z_b with N' observations, where oftentimes $N' = N$. A separate model is then trained on each Z_b , with the model’s prediction on an observation x being given by $\hat{f}_b(x)$. Here, showing each model a different bootstrapped sample helps to de-correlate them, and the overall

bagged estimate for an observation x is then obtained by averaging over all of the individual predictions in the case of regression

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x),$$

or by taking the majority vote over the K response classes in the case of classification

$$\hat{f}(x) = \arg \max_{k \in \mathcal{K}} \left\{ \sum_{b=1}^B I(\hat{f}_b(x) = k) \right\}.$$

One important aspect of bagging is the fact that each observation i will only appear “in-bag” in a subset of the bootstrapped training sets Z_b . Therefore, for each observation i , an “out-of-bag” (OOB) prediction can be constructed by only considering the subset of models in which i did not appear in the bootstrapped training set. Afterwards, an OOB error for bagged models can then be estimated by averaging over the OOB prediction errors for each training observation i , which helps to alleviate the need for cross-validation or a separate test set (Breiman, 1996b).

2.2.2 CART-Like Tree Learning Algorithm

In the case of random forests, the model that is being trained on each individual bootstrapped data set Z_b is a decision tree that is constructed using the CART algorithm, but with two key modifications.

First, as mentioned earlier in Section 2, the trees in random forests are generally not pruned (Breiman, 2001). Second, instead of considering all p predictors at a split, only $m \leq p$ predictors selected at random are allowed to be used for the split—a restriction which helps to de-correlate the trees by placing a constraint on how similarly they can be grown. This process, sometimes referred to as “feature bagging,” was also independently proposed by Ho (1998) and Amit and Geman (1997).

3 The Absent Levels Problem

In Section 1, we defined the absent levels problem as the issue which occurs when there is indeterminacy in how to handle an observation that has reached a categorical split which was determined when the observation’s level was absent during training, and we mentioned three different ways in which the absent levels problem can arise in random forests and other decision tree based algorithms. Then, in Section 2.1.2, we reviewed how the levels of a categorical predictor j that were present in a mother node \mathcal{D}_M were used to determine the optimal splitting criterion S_j^* .

In this section, we investigate the potential consequences of overlooking the absent levels problem, where for a categorical predictor j with Q possible unordered levels which are indexed by the set $\mathcal{Q} = \{1, 2, \dots, Q\}$, we now denote the subset of the levels of j that were either present or absent in the mother node \mathcal{D}_M during training as follows:

$$\begin{aligned}\mathcal{Q}_M^P &= \{q \in \mathcal{Q} \mid \#\{i \in \mathcal{D}_M \text{ and } x_{ij} = q\} > 0\}, \\ \mathcal{Q}_M^A &= \{q \in \mathcal{Q} \mid \#\{i \in \mathcal{D}_M \text{ and } x_{ij} = q\} = 0\}.\end{aligned}\tag{13}$$

Specifically, by documenting how absent levels have been handled by Breiman and Cutler’s random forests **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002), we show how failing to account for the absent levels problem can systematically bias decision tree based algorithms in practice. However, although our investigations are motivated by these two particular software implementations of random forests, we emphasize that these discussions should not overshadow the fact that the absent levels problem is, first and foremost, an inherent methodological issue for decision trees.

3.1 Regression

For regression trees, recall from our discussions in Section 2.1.2 and equations (11) and (12), that a categorical predictor j ’s optimal splitting criterion S_j^* can be characterized by an optimal numeric pseudo split point \tilde{s}_j^* where:

- The unordered levels of j being sent *left* have means that are *less than or equal to* \tilde{s}_j^* .
- The unordered levels of j being sent *right* have means that are *greater than* \tilde{s}_j^* .

Furthermore, recall from equation (2) that a node \mathcal{D}_n ’s prediction is given by the mean of the training responses that are in the node. Therefore, because the prediction of each daughter node can be expressed as a weighted average over the means of the unordered levels that are being sent to it, it follows that *the left daughter node \mathcal{D}_L will always give a prediction that is smaller than the right daughter node \mathcal{D}_R when splitting on a categorical predictor in a regression tree.*

In terms of implementation, both the **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) employ the pseudo value simplification when determining the optimal splitting criterion S_j^* for a categorical predictor j in regression. However, the code that computes the means within each unordered level $q \in \mathcal{Q}$ behaves as follows:

$$\gamma_j(q) = \begin{cases} \text{ave}(y_i \mid i \in \mathcal{D}_M \text{ and } x_{ij} = q) & \text{if } q \in \mathcal{Q}_M^P \\ 0 & \text{if } q \in \mathcal{Q}_M^A \end{cases},$$

where \mathcal{Q}_M^P and \mathcal{Q}_M^A are as defined in equation (13).

Although these “zero imputed” means for the absent levels $q \in \mathcal{Q}_M^A$ are inconsequential when determining the optimal numeric pseudo split point \tilde{s}_j^*

since they do not contribute to the node impurity measure during training, they can be highly influential on the subsequent inferences that are made for observations that have absent levels. In particular, by equations (11) and (12), the absent levels $q \in \mathcal{Q}_M^A$ will be sent left if $\tilde{s}_j^* \geq 0$, and they will be sent right if $\tilde{s}_j^* < 0$. But due to the systematic difference that exists between the predictions of the two daughter nodes, this arbitrary decision of sending the absent levels left versus right can significantly alter the inferences that are made on observations that have absent levels. Specifically, even though the final predictions will also depend on the subsequent splits that take place after the absent levels problem occurs, these observations that have absent levels will tend to receive smaller predictions when they are sent to the left daughter node, and they will tend to receive larger predictions when they are sent to the right daughter node.

Furthermore, this also implies that the random forest models that are trained using the `FORTMAN` code and the `randomForest` R package (Liaw and Wiener, 2002) are sensitive to the support of the training responses. For example, consider the following two extreme cases for the subset of training data in a mother node \mathcal{D}_M that is being split on the categorical predictor j :

- If the training responses are all nonnegative, then $\tilde{s}_j^* \geq 0$. In this situation, since $\gamma_j(q) = 0 \leq \tilde{s}_j^*$ for all $q \in \mathcal{Q}_M^A$, the absent levels will always be sent to the left daughter node \mathcal{D}_L which is biased towards smaller predictions.
- If the training responses are all negative, then $\tilde{s}_j^* < 0$. In this situation, since $\gamma_j(q) = 0 > \tilde{s}_j^*$ for all $q \in \mathcal{Q}_M^A$, the absent levels will always be sent to the right daughter node \mathcal{D}_R which is biased towards larger predictions.

And although this sensitivity to the support of the training responses was most easily illustrated using these two extreme situations, the fact that the absent levels problem can also cause substantial issues in the more general case of training responses in \mathcal{D}_M with mixed signs should not be overlooked.

Another consequence of absent levels being handled in this way by the `FORTMAN` code and the `randomForest` R package (Liaw and Wiener, 2002) is that the predictions for observations that have absent levels can be influenced by changing the support of the training responses *prior* to fitting the model, even when conditioning on the randomness in the algorithm.² Specifically, for some real constant C and a fixed random seed, consider training a random forest model on a new data set that differs only by its location shifted responses $y'_i = y_i + C$ and then, after the model is trained, subtracting C from the predictions of each of the nodes.

Conditional on the bagging and the feature bagging, the same set of in-bag training data will appear in the root mother node of each tree, and the same

²This behavior was present in versions 4.6-7 and earlier of `randomForest` R package (Liaw and Wiener, 2002). Beginning in version 4.6-9 of the `randomForest` R package, however, the R package began to internally mean center the response prior to fitting the model which effectively negates any attempts to change the support of the training responses before fitting the model. Nevertheless, such a strategy merely masks the issue and does not actually address the underlying absent levels problem.

set of candidate predictors will be considered for each tree’s initial split. And since the objective function given by equation (3) is invariant to C , the same optimal splitting variable and splitting criterion pair $(j^*, S_{j^*}^*)$, and daughter nodes \mathcal{D}_L and \mathcal{D}_R will always be chosen for the initial split. But then, by this same reasoning, all subsequent splits and daughter nodes in each of the trees will also always remain the same for any C . Furthermore, by equation (2), each node \mathcal{D}_n will give the location shifted prediction $\hat{c}'_n = \hat{c}_n + C$, which implies that the original non-location shifted predictions \hat{c}_n for each node can be recovered by subtracting C from \hat{c}'_n . Therefore, for any real constant C , the random forest models that are trained using this procedure are identical with respect to the in-bag training data.

However, the value of C can alter the behavior of the random forest models trained using the `FORTMAN` code or the `randomForest` R package (Liaw and Wiener, 2002) with respect to observations that have absent levels. For example, if $C > \max_{i=1,2,\dots,N} |y_i|$, then all of the location shifted training responses will be positive, which then guarantees that all of the categorical splits will have numeric pseudo split points $\tilde{s}_j^* > 0$. But since absent levels have their means imputed to 0, observations with these absent levels will always be sent to the left daughter nodes that are biased towards smaller predictions. Conversely, if $C < -\max_{i=1,2,\dots,N} |y_i|$ instead, then all of the categorical splits will have numeric pseudo split points $\tilde{s}_j^* < 0$, and these same observations that have absent levels will always be sent to the right daughter nodes that are biased towards larger predictions.

3.2 Classification

For binary classification trees, recall from our discussions in Section 2.1.2 and equations (11) and (12), that a categorical predictor j ’s optimal splitting criterion S_j^* can be characterized by an optimal numeric pseudo split point \tilde{s}_j^* where:

- The unordered levels of j being sent *left* have $k = 1$ response class proportions that are *less than or equal to* \tilde{s}_j^* .
- The unordered levels of j being sent *right* have $k = 1$ response class proportions that are *greater than* \tilde{s}_j^* .

Furthermore, recall from equation (6), that a node \mathcal{D}_n ’s prediction is given by the response class with the highest sample proportion occurring in the node. Therefore, because the prediction of each daughter node can be expressed as a weighted average over the response class proportions that are being sent to it, it follows that *the left daughter node \mathcal{D}_L is always less likely to classify an observation to the $k = 1$ response class than the right daughter node \mathcal{D}_R when splitting on a categorical predictor in a binary classification tree.*

In terms of implementation, the `randomForest` R package (Liaw and Wiener, 2002) relies on the pseudo value simplification when determining the optimal

splitting criterion S_j^* for a categorical predictor j with a large number of unordered levels in binary classification.³ However, the code that is responsible for computing the $k = 1$ response class proportions within each unordered level $q \in \mathcal{Q}$ executes as follows:

$$\gamma_j(q) = \begin{cases} \frac{\#\{i \in \mathcal{D}_M | x_{ij}=q \text{ and } y_i=1\}}{\#\{i \in \mathcal{D}_M | x_{ij}=q\}} & \text{if } q \in \mathcal{Q}_M^P \\ 0 & \text{if } q \in \mathcal{Q}_M^A \end{cases}.$$

Consequently, the issues that arise here when making inferences for observations that have absent levels are similar to the ones that were described for regression.

Although these zero imputed $k = 1$ response class proportions for the absent levels $q \in \mathcal{Q}_M^A$ are unimportant when determining the optimal numeric pseudo split point \tilde{s}_j^* since they do not contribute to the node impurity measure during training, they can be highly influential on the subsequent inferences that are made for observations that have absent levels. In particular, since $\gamma_j(q) \geq 0$ for all of the present levels $q \in \mathcal{Q}_M^P$, with strict inequality holding for at least one of these q ,⁴ it follows that the optimal numeric pseudo split point $\tilde{s}_j^* > 0$. As a result, even though the final predictions will also depend on the subsequent splits that take place after the absent levels problem occurs, observations that have absent levels will always be sent to the left daughter node \mathcal{D}_L which is biased towards the $k = 2$ response class. Consequently, random forest binary classification models trained using the `randomForest` R package may be sensitive to the actual ordering of the response variable classes—because observations that have absent levels are always sent to the left daughter node \mathcal{D}_L which is biased towards the $k = 2$ response class, the inferences for these observations can be affected by reindexing the response classes.

On the other hand, for cases where the pseudo value simplification is not or cannot be used, the `FORTTRAN` code and the `randomForest` R package (Liaw and Wiener, 2002) will instead adopt a more brute force approach by either exhaustively or randomly searching through the space of possible splits. However, in order to understand the potential problems that absent levels can cause in these situations, we must first briefly digress into a discussion of how categorical splits are internally represented in their code.

Specifically, a split on a categorical predictor j with Q unordered levels is encoded and decoded as an integer whose binary expansion identifies which levels go left (the bits that are “turned on”) and which levels go right (the bits that are “turned off”). As an example, consider the situation where the categorical predictor j has four unordered levels and where the integer representation of the split is 10. In this case, the binary expansion of 10 is (0, 1, 0, 1) since

³The exact condition for using the pseudo value simplification for binary classification in the `randomForest` R package (Liaw and Wiener, 2002) is when a categorical predictor j has $Q > 10$ unordered levels. Meanwhile, although the `FORTTRAN` code for binary classification references the pseudo value simplification, there does not appear to be any functionality for this simplification in the code.

⁴Otherwise there would be no point in splitting on the categorical predictor j since it would not result in daughter nodes that are “purer” than their mother node.

$10 = [0] \cdot 2^0 + [1] \cdot 2^1 + [0] \cdot 2^2 + [1] \cdot 2^3$, so levels $q \in \{2, 4\}$ get sent left while levels $q \in \{1, 3\}$ get sent right.

Now, when executing an exhaustive search to find the optimal splitting criterion S_j^* for a categorical predictor j with Q unordered levels, the **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) will proceed in a systematic fashion.⁵ In particular, *all $2^{Q-1} - 1$ possible integer representations of the non-redundant partitions of predictor j in increasing sequential order starting from 1 and ending at $2^{Q-1} - 1$, with the choice of the optimal split being updated if and only if the weighted Gini index node impurity measure in equation (8) strictly improves.*

But since the absent levels $q \in \mathcal{Q}_M^A$ are not in the mother node \mathcal{D}_M during training, *sending them left or right has no effect on the weighted Gini index.* And because turning on a bit for any individual level q while holding the bits for all of the other levels constant will always result in a larger integer, it follows that *the exhaustive search that is implemented will always prefer splits that send all of the absent levels right since they are always checked before the analogous splits that send any of them left.*

Furthermore, in this exhaustive search, the “last” bit in the binary expansion representations of the splits is always turned off since checking the splits where this last bit is turned on would be redundant—they would amount to just swapping the “left” and “right” daughter node labels for splits that have already been evaluated. Consequently, the last level Q of a categorical predictor j will also always be sent to the right daughter node \mathcal{D}_R and, as a result, the inferences made for observations that have absent levels may be biased towards the responses of the training observations that took on a level of Q for their predictor j . Therefore, although it sounds contradictory, this also implies that random forest multiclass classification models trained using the **FORTRAN** code or the **randomForest** R package (Liaw and Wiener, 2002) may be sensitive to the actual ordering of a categorical predictor’s unordered levels. Specifically, a reordering of the levels could potentially swap the “left” and “right” daughter, which could in turn affect any of the inferences made for observations that have absent levels since they will still always be sent to whichever node ends up being designated as the “right” daughter node.

Finally, when a categorical predictor j has too many levels for an exhaustive search to be computationally feasible, the **FORTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) will resort to approximating the optimal split S_j^* with the best split that was found amongst a large number of randomly generated splits.⁶ This is accomplished by randomly setting all of the bits in the binary representations to a 0 or a 1, so both the present and absent levels will be randomly sent to either the left or right daughter node. Therefore, although

⁵The **FORTRAN** code will use an exhaustive search for both binary and multiclass classification whenever $Q < 25$. In the **randomForest** R package (Liaw and Wiener, 2002), an exhaustive search will be used for both binary and multiclass classification whenever $Q < 10$.

⁶The **FORTRAN** code will use a random search for both binary and multiclass classification whenever $Q \geq 25$. In the **randomForest** R package (Liaw and Wiener, 2002), a random search will be used only for multiclass classification whenever $Q \geq 10$.

the absent levels problem can still occur in these situations, it is difficult to determine whether it results in any systematic bias in their implementations of random forests. However, it still remains an open question as to whether or not such treatment of absent levels is sufficient.

4 Heuristics for Mitigating the Absent Levels Problem

Although a more theoretical analysis of potential solutions to the absent levels problem is beyond the scope of this paper, in this section we discuss some possible heuristics that may be able to help mitigate the issue.

Despite the fact that absent levels are observed and not actually missing, the missing data literature for decision trees is perhaps the area of existing research that is most closely related to the absent levels problem. One potentially straightforward way of mitigating the absent levels problem along these lines would be to stop an observation from going further down the tree whenever the issue occurs and just use the mother node for inference—an approach for missing data that is used by the `rpart` R package for CART (Therneau et al., 2015) and the `gbm` R package for generalized boosted regression models (Ridgeway, 2013). Even with this missing data functionality already in place, however, the `gbm` R package has still had its own issues around how it handles observations that have absent levels—serving as yet another example of a software implementation of a decision tree based algorithm that has overlooked and suffered from the absent levels problem.⁷

A second missing data heuristic that can be applied to the absent levels problem would be to send the observations that have absent levels down both daughter nodes, perhaps by using an approach that is similar to the distribution based missing data imputation method that is used by the C4.5 algorithm for learning decision trees (Quinlan, 1993). Specifically, in the C4.5 algorithm, an OOB or test observation with missing data is split into multiple pseudo-instances, where each instance takes a different value for the missing predictor, and each instance is assigned a different weight that is based on the frequency of their imputed value in the node’s subset of the training data. These pseudo-instances are then sent down their appropriate daughter nodes, with the final prediction being derived from the weighted results of all the terminal nodes that are reached (Saar-Tsechansky and Provost, 2007).

Surrogate splits, which the `rpart` R package (Therneau et al., 2015) also supports, are arguably the most popular way of handling missing data in CART, and they can also possibly be used to help mitigate the absent levels problem. In this situation, if $(j^*, S_{j^*}^*)$ is the optimal splitting variable and splitting criterion pair at a mother node \mathcal{D}_M , then the first surrogate split is the $(j', S_{j'})$ pair where $j' \neq j^*$ that yields the split most closely resembling the original optimal split, the second surrogate split is the $(j'', S_{j''})$ pair where $j'' \neq j^*$ and $j'' \neq j'$

⁷See <https://code.google.com/archive/p/gradientboostedmodels/issues/7>

that gives the second most similar split, and so on. Then, when an observation with a missing value for its j^* predictor reaches \mathcal{D}_M , the surrogate splits are used in the order of decreasing similarity until a non-missing value can be used for the split (Breiman et al., 1984).

However, despite their extensive use in CART, surrogate splits may not be appropriate for random forests. As pointed out in Ishwaran et al. (2008):

Although surrogate splitting works well for trees, the method may not be well suited for forests. Speed is one issue. Finding a surrogate split is computationally intensive and may become infeasible when growing a large number of trees, especially for fully saturated trees used by forests. Further, surrogate splits may not even be meaningful in a forest paradigm. [A random forest model] randomly selects variables when splitting a node and, as such, variables within a node may be uncorrelated, and a reasonable surrogate split may not exist. Another concern is that surrogate splitting alters the interpretation of a variable, which affects measures such as [variable importance].

Nevertheless, the option of using surrogate splits for missing data is available in the `partykit` R package (Hothorn and Zeileis, 2015), which is an implementation of a bagging ensemble of conditional inference trees that attempt to correct for the biased variable selection issues that exist in CART (Hothorn et al., 2006).

Interestingly, the `partykit` R package (Hothorn and Zeileis, 2015) appears to recognize the possibility of absent levels occurring, and handles them as if they were missing—its reference manual states that “Factors in test samples whose levels were empty in the learning sample are treated as missing when computing predictions.” Furthermore, besides surrogate splits (which are not enabled by default), the `partykit` R package offers some additional missing data functionality that may also be viable ways of addressing absent levels. These include their default approach of randomly sending the observations to one of the daughter nodes (following the daughter node distribution) or, alternatively, by simply having the observations go to the daughter node with more training observations. Whether such heuristics adequately address the absent levels problem, however, remains an open question.

Finally, besides missing data methods, one-hot encoding the categorical predictors may provide another alternative for mitigating the effects of absent levels. Specifically, even though unordered levels may still be absent when a categorical split is being determined during training, by recoding the levels of a categorical predictor j into Q dummy predictors, any uncertainty over where to send the observations that have absent levels will disappear.

However, one-hot encoding is not without its own drawbacks. First, one-hot encoding may not always be computationally feasible since the feature space may become computationally unmanageable. Furthermore, by recoding a categorical predictor’s unordered levels into separate dummy predictors, a decision tree forfeits its ability to simultaneously consider all of the predictor’s levels together at a single split. And potentially more troubling, the CART-like tree learning procedure that is used in random forests is known to be biased in favor of

splitting on ordered predictors and categorical predictors with many unordered levels since they offer more potential splitting points from which to choose from (Hothorn et al., 2006)—an issue that one-hot encoding would further exacerbate. Consequently, it is not clear whether one-hot encoding is preferable in a decision tree based setting.

Interestingly, despite these possible shortcomings surrounding one-hot encoding, it is currently required by the random forests implementation in the `scikit-learn` Python module (Pedregosa et al., 2011). There has, however, been some recent discussions about adding support for the native categorical split capabilities of decision trees that is used by the `FORTRAN` code and the `randomForest` R package (Liaw and Wiener, 2002).⁸ Such efforts, needless to say, would also introduce the indeterminacy of the absent levels problem into another popular software implementation of a decision tree based algorithm.

5 Examples

Although the actual severity of the absent levels problem will depend on the underlying data, in this section we present three real data examples taken from public repositories that illustrate how the absent levels problem can dramatically alter the performance of decision tree based algorithms in practice. In particular, we compare how six different heuristics perform when faced with the absent levels problem in random forests where, due to the inherent randomness in the algorithm, we repeat each example 1000 times.

The first set of heuristics that we consider are ones that have been employed by the `FORTRAN` and `randomForest` R package (Liaw and Wiener, 2002) due to having overlooked the absent levels problem:

- **Left:** Sending the observations to the left daughter node.
- **Right:** Sending the observations to the right daughter node.

However, for reasons discussed in Section 3, these two heuristics are unreliable ways of dealing with absent levels due to the systematic biases that they introduce—they’ve been included for comparative purposes only.

Next, we consider some approaches that have been used to handle missing data in decision tree based algorithms:

- **Stop:** Stopping the observations from going further down the tree and using the mother node for inference.
- **Random:** Randomly sending observations to the left or right daughter node (following the daughter node distribution).
- **Both:** Sending the observations down both daughter nodes using a distribution based missing data imputation method.

⁸See, for example, <https://github.com/scikit-learn/scikit-learn/pull/3346>

Compared to the first set of inadvisable heuristics, these “missing data heuristics” are all less systematic in their preference amongst the two daughter nodes.

Finally, we consider a heuristic which transforms the original data set:

- **One-Hot:** Recoding every categorical predictor’s unordered levels into separate dummy predictors.

Under this approach, although unordered levels may still be absent when a categorical split is being determined during training, any uncertainty over where to send the observations that have absent levels will disappear.

Code for implementing the five “non-one-hot heuristics” was implemented on top of version 4.6-12 of the `randomForest` R package (Liaw and Wiener, 2002). Specifically, the `randomForest` R package is used to first train the random forest models that are used in our experiments. Afterwards, our code sends each tree’s in-bag training observations back down in order to record the unordered levels that were present or absent in nodes that were split on a categorical predictor. Finally, when sending OOB or test observations down each tree, our code provides some functionality for carrying out each of the heuristics whenever the absent levels problem occurs.

By structuring our code in this way, we are able to isolate the effects of the absent levels problem with respect to the five non-one-hot heuristics since the underlying random forest models that are being compared within an experimental replication are identical except for their treatment of the absent levels. Consequently, the inferences obtained using the five non-one-hot heuristics will be positively correlated across the 1000 experiments that are repeated for each example—a fact which we exploit in order to improve the precision of our comparisons.

Recall from Section 4, however, that the random forest models that are trained on one-hot encoded data sets are intrinsically different from the random forest models that are trained on the original untransformed data sets. As a result, the inferences for the One-Hot heuristic will be uncorrelated with the other five non-one-hot heuristics across each example’s 1000 experimental replications.

Similar to how they are often employed in practice, each random forest model that we consider is trained “off-the-shelf” by using the default parameter values in the `randomForest` R package (Liaw and Wiener, 2002), some of which can be seen in Table 1.

Parameter	Regression	Classification
Number of Trees	500	500
Max Terminal Node Size	5	1
Feature Bagging m	$\frac{p}{3}$	\sqrt{p}
Bagging Used?	Yes	Yes

Table 1: Default values for some of the parameters in the `randomForest` R package (Liaw and Wiener, 2002).

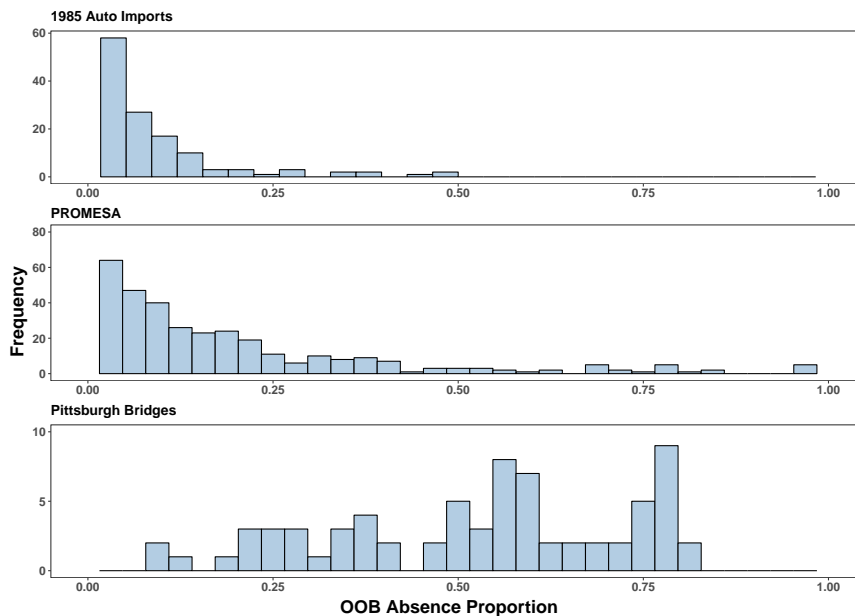


Figure 1: Histograms of the OOB absence proportions for the three examples.

5.1 1985 Auto Imports

For a random forest regression example, we consider the 1985 Auto Imports data set from the UCI Machine Learning repository (Lichman, 2013) which, after discarding observations with missing data, contains 25 features that can be used to predict the prices of 159 cars. Categorical predictors for which the absent levels problem can occur include an automobile’s make (18 levels), body style (5 levels), drive layout (3 levels), engine type (5 levels), and fuel system (6 levels).

In our analysis, random forest models were trained on the log-transformed prices in order to make the responses more symmetric; afterwards, the predictions were transformed back to their original unlogged scale. Because all of log-transformed car prices were positive, we know from Section 3.1, that the `FORTRAN` code and earlier versions of the `randomForest` R package (Liaw and Wiener, 2002) will always employ the Left heuristic when confronted with absent levels, which is biased towards smaller predictions.

The top panel in Figure 1 shows a histogram of this data set’s OOB absence proportions, which we define for each observation as the proportion of its OOB trees across all 1000 experiments which had the absent levels problem occur at least once. Meanwhile, Table 2 lists more detailed summaries for this data set’s distribution of OOB absence proportions, from which we see that the majority of observations had the absent levels problem occur in less than 5% of their OOB trees.

Statistic	1985 Auto Imports	PROMESA	Pittsburgh Bridges
Min	0.003	0.001	0.080
1st Quartile	0.021	0.020	0.368
Median	0.043	0.088	0.564
Mean	0.076	0.162	0.526
3rd Quartile	0.093	0.206	0.702
Max	0.498	0.992	0.820

Table 2: Summary statistics of the OOB absence proportion distributions for the three examples.

Within each experimental replication, pairwise differences between the OOB predictions of the six different heuristics were calculated. Figure 2 shows these pairwise differences as a function of the OOB absence proportions, where each panel plots the mean and middle 95% of the pairwise differences across all 1000 experimental replications when the OOB predictions of the heuristic that is labeled at the right of the panel’s row is subtracted from the OOB predictions of the heuristic that is labeled at the top of the panel’s column. From this figure, we see that both the estimates and the intervals for the pairwise differences tend to increase as the OOB absence proportion increases—indicating that the distinct effects of the six heuristics becomes more pronounced the more often the absent levels problem occurs. Furthermore, relative to the other heuristics and consistent with our discussions in Section 3.1, we see that the Left and Right heuristics tend to underpredict and overpredict, respectively. And although the other four heuristics seem to be more aligned with one another in terms of their OOB predictions, some significant differences do exist between them.

The overall performance of each heuristic’s OOB predictions within each experiment j was evaluated in terms of its root mean squared error (RMSE):

$$RMSE_{h,j} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_{h,i,j})^2} \quad (14)$$

where $\hat{y}_{h,i,j}$ is heuristic h ’s OOB prediction for observation i in experiment j . Boxplots comparing each heuristic’s marginal distribution of RMSEs across all 1000 experimental replications are shown in the left panel of Figure 3. However, these marginal boxplots ignore the positive correlation that exists between the non-one-hot heuristics. Consequently, within every experimental replication j , we also compare these RMSEs relative to the lowest RMSE amongst the Stop, Random, and Both heuristics:

$$RMSE_{h,j}^R = \frac{RMSE_{h,j} - RMSE_j^*}{RMSE_j^*}, \text{ where} \quad (15)$$

$$RMSE_j^* = \min (RMSE_{h,j} \mid h \in \{\text{Stop, Random, Both}\})$$

and where $RMSE_{h,j}$ is as defined in equation (14). Here the Left and Right

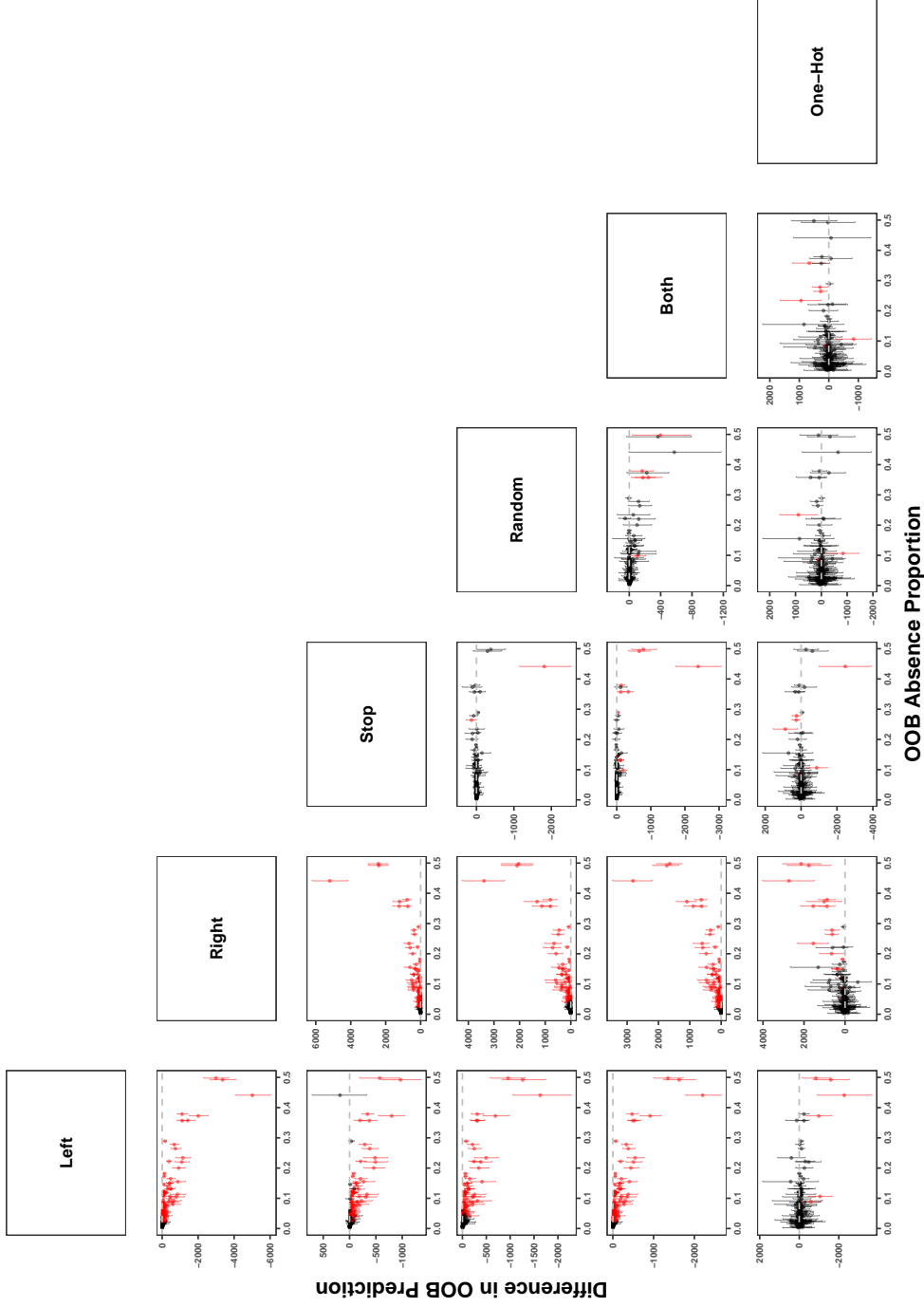


Figure 2: Pairwise differences in the OOB predictions as a function of the OOB absence proportions in the 1985 Auto Imports data set. Each panel plots the mean and middle 95% of the pairwise differences across all 1000 experimental replications when the OOB predictions of the heuristic that is labeled at the right of the panel's row is subtracted from the OOB predictions of the heuristic that is labeled at the top of the panel's column. Pairwise differences were taken within each experimental replication to account for the positive correlation that exists between the non-one-hot heuristics. Intervals containing 0 (the horizontal dashed line) are in black; intervals not containing 0 are in red.

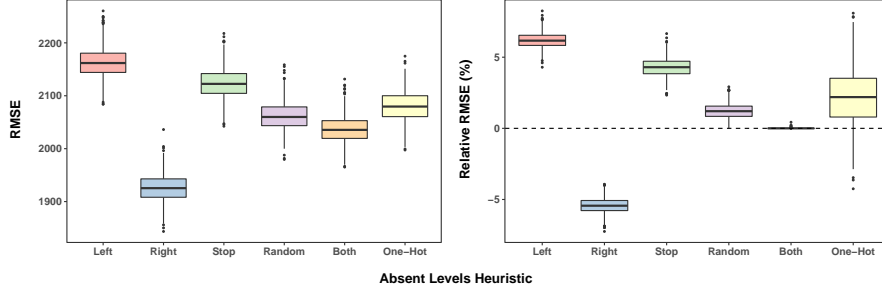


Figure 3: RMSEs of the OOB predictions from the six heuristics in the 1985 Auto Imports data set across all 1000 experiments. The left panel shows boxplots of each heuristic’s marginal distribution of RMSEs, which ignores the positive correlation that exists between the non-one-hot heuristics. The right panel accounts for this positive correlation by comparing the RMSEs of the heuristics relative to the lowest RMSE amongst the Stop, Random, and Both heuristics within each of the 1000 experimental replications as in equation (15). In both panels, lower values are better.

heuristics were not included in the definition of $RMSE_{h,j}^*$ since they are unreliable given the issues discussed in Section 3.1, while the One-Hot heuristic was not considered in definition of $RMSE_{h,j}^*$ since it is uncorrelated with the other heuristics.

Boxplots of these relative RMSEs across all 1000 experimental replications are shown in the right panel of Figure 3. For this particular data set and inference task, we see that the Both heuristic outperforms the Random heuristic, and the two of them significantly outperform the Stop heuristic. Furthermore, the FORTRAN code and `randomForest` R package’s (Liaw and Wiener, 2002) convention of always sending absent levels left for this particular data set substantially underperforms relative to the other five heuristics—giving an RMSE that is, on average, 6.2% worse than the best performing missing data heuristic. Meanwhile, the Right heuristic’s performance is highly deceptive since its tendency to overpredict coincidentally happens to be beneficial in this situation; in general, this will not necessarily be the case. Finally, even though the One-Hot heuristic may sometimes outperform the other heuristics, on average, it yields an RMSE that is 2.2% worse than the best missing data heuristic.

5.2 PROMESA

For a random forest binary classification example, we consider the June 9, 2016 United States House of Representatives vote on the Puerto Rico Oversight, Management, and Economic Stability Act (PROMESA) for addressing the Puerto Rican government’s debt crisis. Data for this vote was obtained by querying

the Voteview database through the `Rvoteview` R package (Lewis, 2015). Ignoring those not voting on the bill, the data set contains four features that can be used to predict the “NO” or “YES” votes of 424 House of Representative members. These features include a representative’s political party (categorical with 2 levels), the representative’s state (categorical with 50 levels), and two ordered “DW-NOMINATE” predictors which quantify the representative’s political ideological position (McCarty et al., 1997).

In our analysis, the “NO” vote is taken to be the $k = 1$ response class, while the “YES” vote is taken to be the $k = 2$ response class. Recall from Section 3.2, that this ordering of the response classes is meaningful in a binary classification setting because the `randomForest` R package (Liaw and Wiener, 2002) will always employ the Left heuristic to handle absent levels, which is biased in favor of the $k = 2$ response class.

From Figure 1 and Table 2, we see that the absent levels problem occurs more frequently in this example than it did in the 1985 Auto Imports example. In particular, seven of the observations had OOB absence proportions that were greater than 0.961—corresponding to the seven people who were the sole representatives from their state. Consequently, the absent levels problem occurred for these seven observations every time they reached an OOB node that split on the state predictor.

For a random forest classification model, predicted response class probabilities for each observation can be estimated by the proportion of trees that classify the observation to each response class.⁹ Figure 4 shows the mean and 95% intervals for the pairwise differences between each heuristic’s OOB predicted probabilities of voting “YES” as a function of the OOB absence proportions. As expected given our discussions in Section 3.2, since a vote of “YES” corresponds to the $k = 2$ response class in our analysis, the Left heuristic yields considerably higher predicted “YES” probabilities than the other heuristics, while the Right heuristic gives “YES” probabilities that are much lower. And similar to what was previously seen in 1985 Auto Imports example, although the other four heuristics are more in agreement with one another, significant differences in their predicted probabilities still exist.

Large discrepancies in the OOB predicted probabilities of the heuristics is particularly concerning since they can lead to observations being classified to different classes. In particular, let $\hat{y}_{h,i,j}$ be heuristic h ’s OOB classification for observation i in experiment j . Cohen’s kappa coefficient provides one way of quantifying the degree of inter-annotator agreement between two classifiers h_1 and h_2 (Cohen, 1960), and is defined for each experimental replication j as

$$\kappa_j = \frac{p_j^o - p_j^e}{1 - p_j^e}, \quad (16)$$

⁹This is the approach that is used by the `randomForest` R package (Liaw and Wiener, 2002). An alternative method of calculating the predicted response class probabilities, which is used by the `scikit-learn` Python module (Pedregosa et al., 2011), is to take the average of the predicted class probabilities over the trees in the random forest, where the predicted probability of response class k in an individual tree is estimated using the proportion of a node’s training samples that belong to class k .

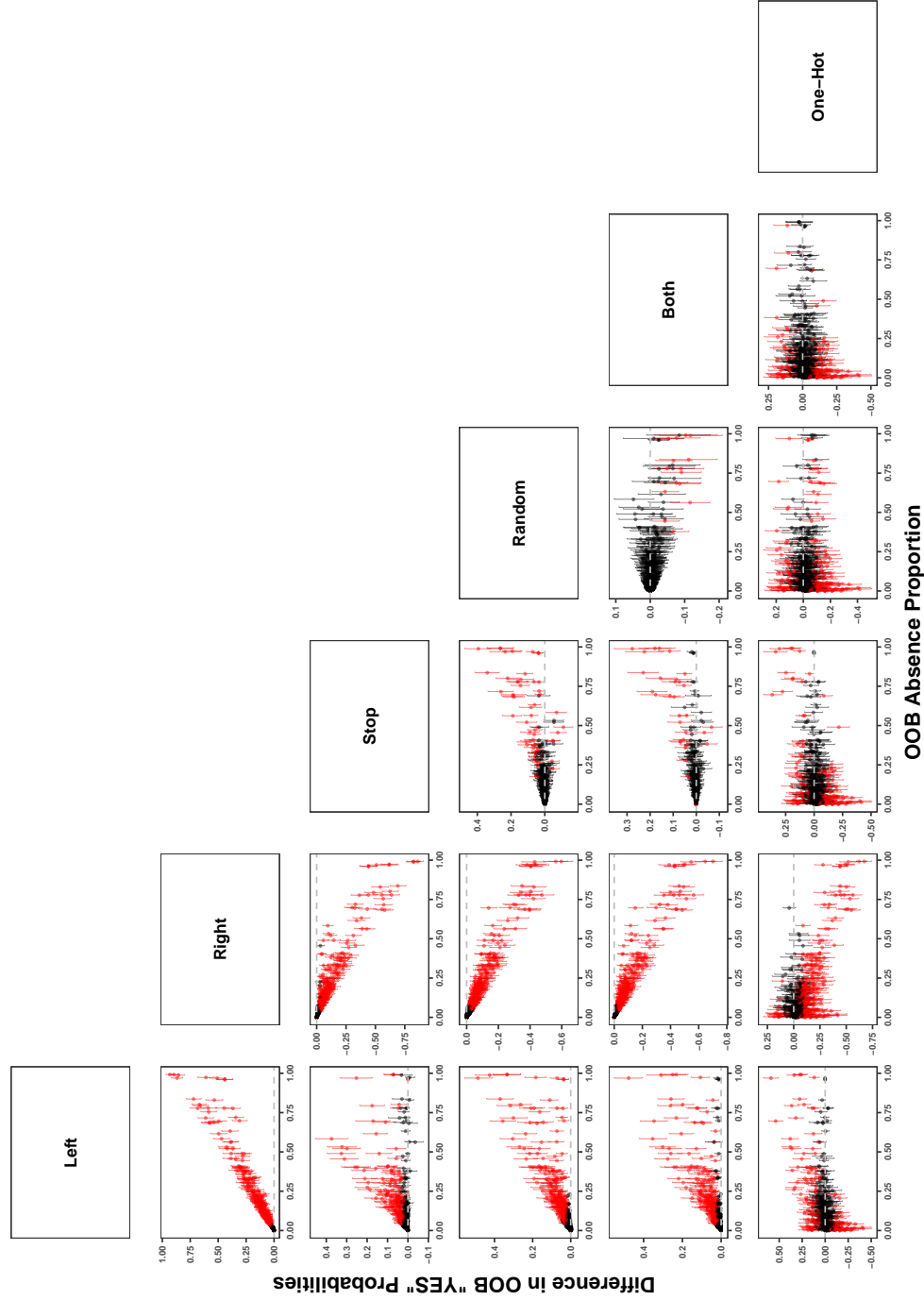


Figure 4: Pairwise differences in the OOB predicted probabilities of voting “YES” as a function of the OOB absence proportion in the PROMESA data set. Each panel plots the mean and middle 95% of the pairwise differences across all 1000 experimental replications when the OOB predictions of the heuristic that is labeled at the right of the panel’s row is subtracted from the OOB predictions of the heuristic that is labeled at the top of the panel’s column. Pairwise differences were taken within each experimental replication to account for the positive correlation that exists between the non-one-hot heuristics. Intervals containing 0 (the horizontal dashed line) are in black; intervals not containing 0 are in red.

where

$$p_j^o = \frac{1}{N} \sum_{i=1}^N I(\hat{y}_{h_1,i,j} = \hat{y}_{h_2,i,j})$$

is the observed probability of agreement between the two classifiers, and

$$p_j^e = \frac{1}{N^2} \sum_{k=1}^K \left[\left(\sum_{i=1}^N I(\hat{y}_{h_1,i,j} = k) \right) \cdot \left(\sum_{i=1}^N I(\hat{y}_{h_2,i,j} = k) \right) \right]$$

is the expected probability of chance agreement. If the classifiers are in complete agreement in experimental replication j , then $\kappa_j = 1$; if there is no agreement among the classifiers other than what would be expected by chance, then $\kappa_j \leq 0$.

Figure 5 shows histograms for each pair of heuristic’s Cohen’s kappa coefficients across all 1000 experimental replications when the random forests algorithm’s default majority vote discrimination probability threshold of 0.5 is used (Liaw and Wiener, 2002). We see from these histograms that, even though the non-one-hot heuristics are positively correlated, they will oftentimes yield very different classifications for the same observations. Furthermore, consistent with what we’ve already seen, the systematically biased Left and Right heuristics tend to exhibit a higher level of disagreement with every other heuristic, while there appears to be much more agreement between the Stop, Random, and Both heuristics. This time, however, the One-Hot heuristic appears to be less aligned with the missing data heuristics.

More generally, the receiver operating characteristic (ROC) and precision-recall (PR) curves can be used to evaluate the overall performance of binary classifiers as the discrimination probability threshold is varied. In particular, as the threshold changes, the ROC curve plots the proportion of positive observations that a classifier correctly labels (true positive rate) as a function of the proportion of negative observations that a classifier incorrectly labels (false positive rate), while the PR curve plots the proportion of a classifier’s positive labels that are truly positive (precision) as a function of the proportion of positive observations that a classifier correctly labels (recall, which is the same as the true positive rate). In our analysis, we take the “YES” vote to be our positive response class, while we take the “NO” vote to be our negative response class.

The ROC and PR curves are commonly summarized by calculating the areas underneath them (Davis and Goadrich, 2006). Boxplots comparing each heuristic’s marginal distribution of these areas across all 1000 experimental replications are shown in the left panels of Figure 6. However, these marginal boxplots ignore the positive correlation that exists between the non-one-hot heuristics. Therefore, similar to what was previously done in the 1985 Auto Imports Example, within every experimental replication j , we also compare these areas

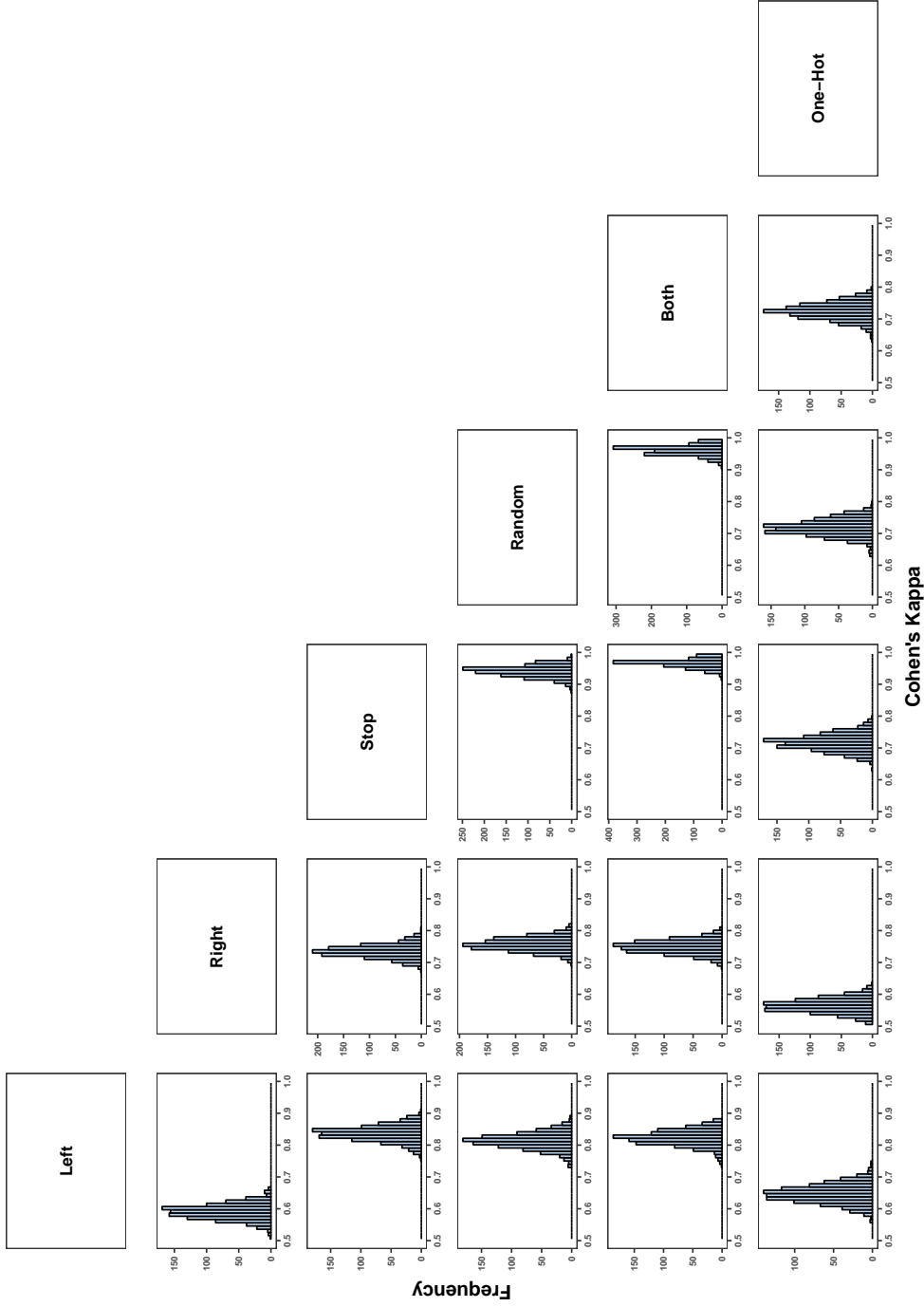


Figure 5: Pairwise Cohen's kappa coefficients for the six different heuristics in the PROMESA data set when the random forests algorithm's default majority vote criterion is used. Each panel plots the histogram of coefficients across all 1000 experimental replications when the heuristic that is labeled at the right of the panel's row is paired with the heuristic that is labeled at the top of the panel's column. Coefficients were calculated within each of the 1000 experimental replications to account for the positive correlation between the non-one-hot heuristics.

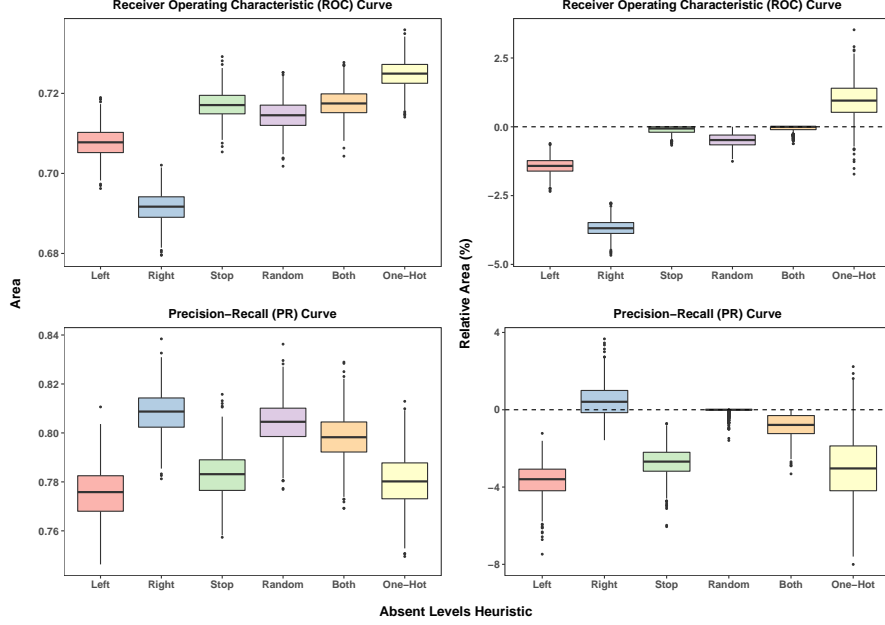


Figure 6: Areas underneath the ROC and PR curves for the six heuristics in the PROMESA data set across all 1000 experiments. The left panels show boxplots of each heuristic’s marginal distribution of areas, which ignores the positive correlation that exists between the non-one-hot heuristics. The right panels account for this positive correlation by comparing the areas of the heuristics relative to the largest area amongst the Stop, Random, and Both heuristics within each of the 1000 experimental replications as in equation (17). In all four panels, higher values are better.

relative to the largest area amongst the Stop, Random, and Both heuristics.

$$Area_{h,j}^R = \frac{Area_{h,j} - Area_j^*}{Area_j^*}, \text{ where } (17)$$

$$Area_j^* = \max (Area_{h,j} \mid h \in \{\text{Stop, Random, Both}\})$$

and where, depending on the context, $Area_{h,j}$ denotes heuristic h ’s area under either the ROC or PR curve for experimental replication j .

Boxplots of these relative areas are shown in the right panels of Figure 6, which show that the relative performances of the heuristics for this particular data set will depend on the inference task. In particular, in terms of the area under the ROC curve, the Both heuristic appears to slightly outperform the Stop heuristic, and the two heuristics seem to outperform the Random heuristic. However, in terms of the area under the PR curve, the relative ranking of these three missing data heuristics reverses. Similarly, although the One-Hot heuristic

performs well in the ROC case, it also performs poorly in the PR case relative to the other heuristics. And even though the performances of the Left and Right heuristics are not generalizable, it is interesting to note that the **randomForest** R package’s (Liaw and Wiener, 2002) practice of always sending absent levels left in binary classification results in significantly worse areas relative to the best performing missing data heuristic—giving areas under the ROC and PR curves that are, on average, 1.4% and 3.6% worse, respectively.

5.3 Pittsburgh Bridges

Finally, for a random forest multiclass classification example, we consider the Pittsburgh Bridges data set from the UCI Machine Learning repository (Lichman, 2013), which, after removing observations with missing data, contains seven features that can be used to classify 72 bridges into one of six different bridge types. Categorical predictors for which the absent levels problem can occur include a bridge’s river (3 levels), purpose (3 levels), and location (46 levels). Consequently, recall from Section 3.2, that the **FORTTRAN** code and **randomForest** R package (Liaw and Wiener, 2002) will employ an exhaustive search that always sends absent levels right when splitting on the river or location features, and they will resort to using a random search when splitting on the location feature since there are too many levels for an exhaustive search to be computationally feasible. The OOB absence proportions for this data set are shown in the bottom panel of Figure 1 and in Table 2.

Let $\hat{p}_{h,i,j,k}$ be heuristic’s h ’s OOB predicted probability that observation i belongs to class k in experiment j . The log loss function,

$$LogLoss_{h,j} = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y_i = k) \log(\hat{p}_{h,i,j,k}), \quad (18)$$

provides one way of measuring a heuristic’s overall performance in experiment j . Meanwhile, comparing the log losses of each heuristic relative to the smallest log loss amongst the Stop, Random, and Both heuristics within each experimental replication j allows us to once again take advantage of the positive correlation that exists amongst the non-one-hot heuristics:

$$LogLoss_{h,j}^R = \frac{LogLoss_{h,j} - LogLoss_j^*}{LogLoss_j^*}, \text{ where} \quad (19)$$

$$LogLoss_j^* = \min (LogLoss_{h,j} \mid h \in \{\text{Stop, Random, Both}\}),$$

and where $LogLoss_{h,j}$ is as defined in equation (18).

The left and right panels of Figure 7 show boxplots of each heuristic’s marginal and relative distribution of log losses, respectively. Here, we see that the Random heuristic significantly outperforms the Both and Stop heuristics. And although the One-Hot heuristic occasionally does better, on average, it performs 4.1% worse than the best missing data heuristic.

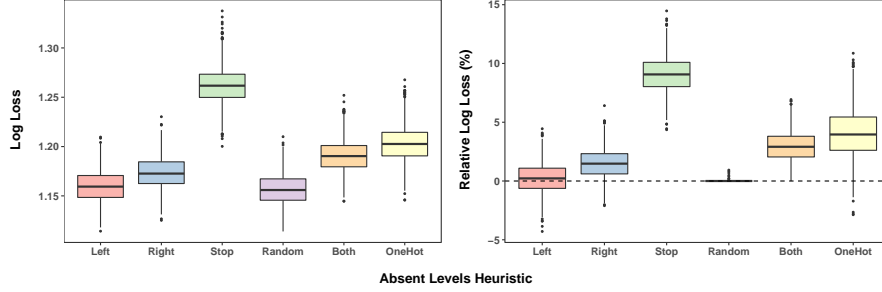


Figure 7: Log losses of the OOB predicted response class probabilities for the six heuristics in the Pittsburgh Bridges data set across all 1000 experiments. The left panel shows boxplots of each heuristic’s marginal distribution of log loss, which ignores the positive correlation that exists between the non-one-hot heuristics. The right panel accounts for this positive correlation by comparing the log losses of the heuristics relative to the smallest log loss amongst the Stop, Random, and Both heuristics within each of the 1000 experimental replications as in equation (19). In both panels, lower values are better.

6 Conclusion

In this paper, we introduced and investigated the absent levels problem for random forests and other decision tree based algorithms. Then, by using Breiman and Cutler’s random forests **FORTTRAN** code and the **randomForest** R package (Liaw and Wiener, 2002) as motivating case studies, we showed how overlooking the absent levels problem could systematically bias a model. Afterwards, we considered some possible heuristics that may be able to help mitigate the issue, and we presented three real data examples taken from public repositories that demonstrated how the treatment of absent levels can significantly alter a model’s performance in practice.

Although a theoretical analysis of potential solutions to the absent levels problem is beyond the scope of this paper, our analysis and results showed that the Stop, Random, Both, and One-Hot heuristics were all more reliable and generalizable than some of the existing biased approaches that are currently being employed due to oversights in the software implementations of decision tree based algorithms. In particular, even though the relative performances of these four heuristics can vary depending on the underlying data set and inference task, we suggest that software packages incorporate the Random heuristic as a provisional measure until a more robust theoretical solution is found. This recommendation is based on both our empirical results, which show that the Random heuristic is able to perform relatively well even when it is not the optimal heuristic, as well as our own personal experiences surrounding the degree of difficulty in implementing the different heuristics on top of the **randomForest**

R package (Liaw and Wiener, 2002). In the meantime, while waiting for these interim and permanent solutions to be implemented, we also urge users who rely on decision tree based algorithms to temporarily one-hot encode their data sets when possible—even though our empirical results suggest that one-hot encoding may lead to a decrease in model performance, we believe this to still be preferable to using the existing biased approaches which do not adequately address absent levels.

Finally, although this paper focused on the absent levels problem for a popular subset of the possible types of analysis in which decision tree based algorithms have been used, it is important to recognize that the issue applies more generally. For example, random forests have also been used for clustering, detecting outliers, imputing missing values in the data, and generating variable importance measures (Breiman, 2001, 2003)—tasks which also depend on the terminal node behavior of the observations. Furthermore, various extensions of random forests—such as quantile regression forests (Meinshausen, 2006, 2012) and the infinitesimal jackknife method for estimating the variance (Wager et al., 2014)—provide software implementations that are built on top of the `randomForest` R package (Liaw and Wiener, 2002) where systematic biases exist. Consequently, given how extensively decision tree based algorithms have been used, it is conceivable that a sizable number of these models have been significantly and unknowingly affected by this issue—further emphasizing the need for the development of both theory and software that accounts for the absent levels problem.

Acknowledgments

The author is extremely grateful to Art Owen for numerous valuable discussions, insightful comments, and helpful suggestions which substantially improved this paper. The author would also like to thank David Chan and Robert Bell for their many useful comments, as well as Jim Koehler, Tim Hesterberg, Iván Díaz, Joseph Kelly, Jingang Miao, and Aiyu Chen for several interesting discussions.

References

- Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.
- G rard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(1):1063–1095, 2012.
- G rard Biau, Luc Devroye, and G bor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(1), 2008.
- Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a.

- Leo Breiman. Out-of-bag estimation. Technical report, Department of Statistics, U.C. Berkeley, 1996b. URL <https://www.stat.berkeley.edu/~breiman/00Bestimation.pdf>.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- Leo Breiman. Manual—setting up, using, and understanding random forest v4.0. 2003. URL https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf.
- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.
- Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.
- Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.
- Misha Denil, David Matheson, and Nando de Freitas. Narrowing the gap: Random forests in theory and in practice. In *Proceedings of the 31th International Conference on Machine Learning*, pages 665–673, 2014.
- Walter D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.
- Trevor J Hastie, Robert J. Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, New York, 2009.
- Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- Torsten Hothorn and Achim Zeileis. partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research*, 16:3905–3909, 2015.
- Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.
- Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. Random survival forests. *Annals of Applied Statistics*, 2(3):841–860, 2008.
- Jeff Lewis. *Rvoterview: Voteview Data in R*, 2015. R package version 0.1. <https://github.com/JeffreyBLewis/Rvoterview>, <http://voteview.polisci.ucla.edu>, <http://voteview.com>.

- Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002.
- Moshe Lichman. UCI machine learning repository, 2013. URL <http://archive.ics.uci.edu/ml>.
- Wei-Yin Loh and Nunta Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83(403):715–725, 1988.
- Nolan M. McCarty, Keith T. Poole, and Howard Rosenthal. *Income Redistribution and the Realignment of American Politics*. AEI Press, publisher for the American Enterprise Institute, 1997.
- Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.
- Nicolai Meinshausen. *quantregForest: Quantile Regression Forests*, 2012. URL <http://CRAN.R-project.org/package=quantregForest>. R package version 0.2-3.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- John R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.
- Greg Ridgeway. *gbm: Generalized Boosted Regression Models*, 2013. URL <http://CRAN.R-project.org/package=gbm>. R package version 2.1.
- Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996. ISBN 0-521-46086-7.
- Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8: 1623–1657, 2007.
- Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. URL <http://CRAN.R-project.org/package=rpart>. R package version 4.1-10.
- Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15(1):1625–1651, 2014.