Available online at www.sciencedirect.com
ScienceDirect
Machine Learning with Applications 00 (2021) 1–15
Machine Learning with Applications

Partial Dependence through Stratification

Terence Parr[a], James D. Wilson[b,*]

[a]Master of Data Science Program, University of San Francisco. San Francisco, CA 94117, USA. Email: parrt@usfca.edu
[b]Department of Mathematics and Statistics, University of San Francisco. San Francisco, CA 94117, USA. Email: jdwilson4@usfca.edu

Abstract

Partial dependence curves are commonly used to explain feature importance once a supervised learning model has been fitted to data. However, it is common for the same partial dependence algorithm to give meaningfully different curves for different supervised models, even when the algorithm is applied to the same data. As a result, it is difficult to distinguish between model artifacts and true relationships in the data. In this paper, we contribute methods for computing partial dependence curves, for both numerical (StratPD) and categorical explanatory variables (CatStratPD), that work directly from training data rather than the predictions of a fitted model. Our methods provide a direct estimate of partial dependence, and rely on approximating the partial derivative of an unknown regression function. We investigate settings where contemporary partial dependence methods — including FPD, ALE, and SHAP methods — give biased results. We demonstrate that our approach works correctly on synthetic data and plausibly on real data sets. This work motivates a new line of inquiry into nonparametric partial dependence that provides robust information about the variables considered in a supervised learning task.

Keywords: decision trees; feature importance; random forests; variable importance

1. Introduction

Partial dependence, the isolated effect of a specific variable or variables on the response variable, y, is important to researchers and practitioners in many disparate fields such as medicine, business, and the social sciences. In medicine, for example, researchers are interested in the relationship between an individual's demographics or clinical features and their susceptibility to illness. Business analysts at a car manufacturer might need to know how changes in their supply chain affect defect rates. Climate scientists are interested in how different atmospheric carbon levels affect temperature.

In the simplest setting when there is only one explanatory variable, x_1, a plot of the y against x_1 visualizes the marginal effect of feature x_1 on y exactly. Given two or more features, one can similarly plot the marginal effects of each feature separately, however, the analysis is complicated by the interactions of the variables (Cox and Wermuth, 2014). Variable interactions and codependencies between features result in marginal plots that do not isolate the specific contribution of a feature of interest to the response. For example, a marginal plot of sex (male/female)

*Corresponding author

against body weight would likely show that, on average, men are heavier than women. While true, men are also taller than women on average, which likely accounts for most of the difference in average weight. It is unlikely that two "identical" people, differing only in sex, would be appreciably different in weight.

In general, strategies for partial dependence seek to characterize the individual and joint effects of the variables in the columns of an explanatory matrix $\mathbf{X} \in \mathcal{R}^{n \times p}$ on a response vector $\mathbf{y} \in \mathcal{R}^n$. Contemporary partial dependence techniques such as Friedman's original partial dependence (Friedman, 2000) (which we will denote FPD), functional ANOVA (Hooker, 2007), Individual Conditional Expectations (ICE) (Goldstein et al., 2015), Accumulated Local Effects (ALE) (Apley and Zhu, 2016), and most recently SHAP (Lundberg and Lee, 2017) interrogate variable importance through fitted models. Model-based techniques like these assess partial dependence by (1) first estimating an unknown model $f : \mathbf{X} \rightarrow \mathbf{y}$ with $\hat{f}$ that characterizes the relationship between $\mathbf{X}$ and $\mathbf{y}$ and (2) subsequently quantifying the effect of each feature x_j by marginalizing $\hat{f}$ (specific details of these methods are provided in Section 3). Model-based techniques dominate the partial dependence research literature because interpreting the output of a fitted model has several advantages. Perhaps the most important consideration is that models generally smooth over noise. Models thereby act like analysis preprocessing steps, potentially reducing the computational burden of the technique; e.g., ALE is $O(n)$ for the n records of $\mathbf{X}$. Model-based techniques are typically model-agnostic, though for efficiency, some provide model-specific optimizations, as SHAP does. Partial dependence techniques that interrogate models also provide insight into the models themselves; i.e., how variables affect model behavior. It is also true that, in some cases, a predictive model is the primary goal so creating a suitable model is not an extra burden.

Despite their wide use, model-based techniques have two significant disadvantages. The first relates to their ability to tease apart the effect of codependent features. It is often the case that models are sometimes required to extrapolate into regions of nonexistent support or even into nonsensical observations (see discussions in Apley and Zhu (2016) and Hooker (2007)). As we demonstrate in our numerical studies in Section 4, model-based techniques can vary in their ability to isolate variable effects in practice. Second, hazards exist in the estimation of the true function that relates $\mathbf{X}$ to $\mathbf{y}$. If a fitted model is unable to accurately capture the relationship between features and y, for whatever reason, then partial dependence does not provide any useful information to the user. To make interpretation more challenging, there is no definition of "accurate enough." A user's understanding of partial dependence comes down to the model that they fit. As there is no "ground-truth" model or algorithm, and partial dependence analyses are often not robust across models, analyses depend almost entirely on the model that a practitioner utilizes. Also, given an accurate fitted model, business analysts and scientists peer at the data through the lens of the model, which can distort partial dependence curves. Separating visual artifacts of the model from real effects present in the data requires expertise in model behavior (and optimally in the implementation of model fitting algorithms).

Consider the combined FPD/ICE plots shown in Figure 1 derived from several models (random forest, gradient boosting, linear regression, deep learning) fitted to the same New York City rent data set from Kaggle (2017). The subplots in Figure 1(b)-(e) present starkly different partial dependence relationships and it is unclear which, if any, is correct. The marginal plot, (a), drawn directly from the data shows a roughly linear growth in price for a rise in the number of bathrooms, but this relationship is biased because of the dependence of bathrooms on other variables, such as the number of bedrooms. (e.g., five bathroom, one bedroom apartments are unlikely.) For real data sets with codependent features, the true relationship is unknown so it is hard to evaluate the correctness of the plots. (Humans are unreliable estimators, which is why we need data analysis algorithms in the first place.) Nonetheless, having the same algorithm, operating on the same data, give meaningfully different partial dependences is undesirable and makes one question their precision.

Experts are often able to quickly recognize model artifacts, such as the stairstep phenomenon in Figure 1(b) and (c) inherent to decision tree-based methods trying unsuccessfully to extrapolate. In this case, though, the stairstep is more accurate than the linear relationship in (d) and (e) because the number of bathrooms is discrete (except for "half baths"). The point is that interpreting model-based partial dependence plots can be misleading, even for experts.

An accurate mechanism to compute partial dependences that did not peer through fitted models would be most welcome. Such partial dependence curves would be accessible to users who lack the expertise to create suitable models. (One can imagine a spreadsheet plug-in that produced partial dependence curves.) A mechanism that did not rely on a user-provided model would also reduce the chance of plot misinterpretation due to model artifacts and could even help machine learning practitioners to choose appropriate models based upon relationships exposed in the data.

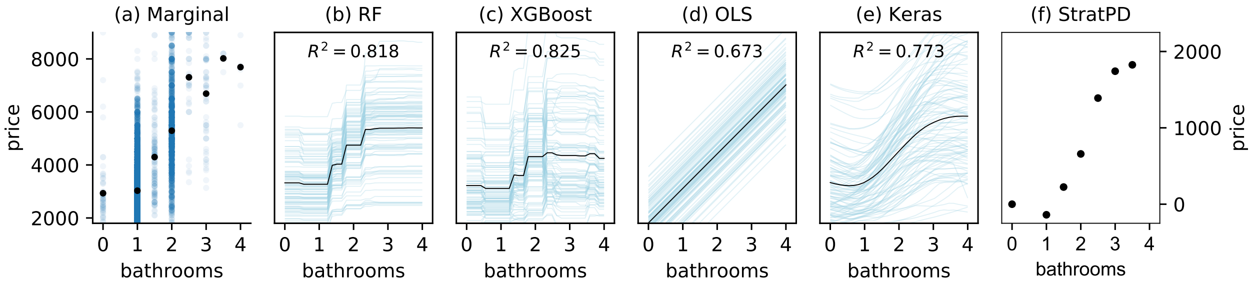

Figure 1. Meaningfully different results from a single partial dependence technique, FPD/ICE, applied to the same data but different models. Plots of number of bathrooms versus rent price using New York City apartment rent data (Kaggle, 2017) with $n = 10,000$ of ˜50k. (a) marginal plot, (b) plot derived from random forest, (c) plot derived from gradient boosted machine, and (d) plot derived from ordinary least squares regression. Hyper parameters were tuned using 5-fold cross validation grid search over several hyper parameters. (e) A Keras model trained by experimentation: single hidden layer of 100 neurons, 500 epochs, batch size of 1000, batch normalization, and 30% dropout. (f) StratPD gives a plausible result that rent goes up linearly with the number of bathrooms. R^2 were computed on 20% validation sets.

In this paper, we propose a strategy, called STRAtified Partial Dependence (StratPD), that computes partial dependences directly from training data $(\mathbf{X}, \mathbf{y})$, rather than through the predictions of a fitted model. Our technique is based upon the notion of an idealized partial dependence: integration over the partial derivative of y with respect to the variable of interest for the smooth function that generated $(\mathbf{X}, \mathbf{y})$. As that function is unknown, we estimate the partial derivatives from the data non-parametrically. Informally, the approach examines changes in y across x_j while holding $x_{\setminus j}$ constant or nearly constant ($x_{\setminus j}$ denotes all variables except x_j). To hold $x_{\setminus j}$ constant, we use a single decision tree to partition feature space, a concept used by Strobl et al. (2008) and Breiman and Cutler (2003) for conditional permutation importance and observation similarity measures, respectively. We furthermore develop CatStratPD, a stratification technique that computes partial dependence curves for categorical variables that, unlike existing techniques, does not assume adjacent category levels are similar. Both StratPD and CatStratPD are quadratic in n, in the worst case (like FPD), though StratPD behaves linearly on real data sets. Both StratPD and CatStratPD are constructed specifically for regression where the the response $\mathbf{y}$ is assumed to be continuous on $\mathbf{R}^n$. Furthermore both strategies effectively handle mutually dependent features. We show the unique result of StratPD when applied to the same example considered in Figure 1 in plot 1(f). In this example, StratPD provides a plausible interpretation of the New York City appartment rent data, namely that rent increases linearly with the number of bathrooms.

We begin by describing the proposed stratification approach in Section 2. We compare StratPD to related (model-based) work in Section 3. In Section 4, we present partial dependence curves generated by StratPD and CatStratPD on synthetic and real data sets, contrast these results with existing methods, and use synthetic data to highlight advantages of StratPD and CatStratPD over contemporary model-based methods. Software and reproducible scripts for the algorithm and all results in this manuscript are available via the Python package `stratx` with source code at `github.com/parrt/stratx`.

2. Partial dependence without model predictions

Assume we are given training data $(\mathbf{X}, \mathbf{y})$ where $\mathbf{X} = [x^{(1)}, \ldots, x^{(n)}]$ is an $n \times p$ matrix whose p columns represent observed features and $\mathbf{y}$ is the $n \times 1$ vector of responses. For any smooth function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ that precisely maps each $x^{(i)}$ row vector to response $y^{(i)}$, $y^{(i)} = f(x^{(i)})$, the partial derivative of y with respect to x_j gives the change in y holding all other variables constant. Integrating the partial derivative then gives the *idealized partial dependence* of y on x_j, the isolated contribution of x_j to y:

Definition 1 The *idealized partial dependence* of y on feature x_j for continuous and smooth generator function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ evaluated at $x_j = z$ is the cumulative sum up to z:

$$PD_j(z) = \int_{min(x_j)}^{z} \frac{\partial f}{\partial x_j} dx_j \tag{1}$$

$PD_j(z)$ is the value contributed to f by x_j at $x_j = z$ and $PD_j(min(x_j)) = 0$. The underlying generator function is unknown, and so other approaches begin by estimating f with a fitted model, $\hat{f}$. Instead, we estimate the partial derivatives of the true function, $\partial f/\partial x_j$, from the raw training data then integrate to obtain PD_j. We note that this definition of partial dependence is local in the sense that the effect of x_j on y is measured for all values across its domain: $z \in Domain(x_j)$. The advantages of this PD_j definition are that it (i) does not require a fitted model or its predictions and (ii) is insensitive to codependent features.

Our definition of partial dependence is the same as that introduced in the recent work of Apley and Zhu (2016), where the ALE procedure was developed for partial dependence. It is important to note, however, that our proposed strategy for estimating PD_j differs from ALE in that ALE first estimates $\hat{f}$ and then calculates PD_j; whereas, our strategies directly estimate $\partial f/\partial x_j$. We show in Section 4 and 5 where our algorithm can have significant advantages in real and simulated data.

2.1. A stratification approach to partial dependence

The aim of StratPD is to estimate PD_j for each feature x_j across its domain. There are two primary steps in the algorithm, described below. Psuedo-code for both StratPD and CatStratPD are provided in the Appendix in Algorithm 1 and Algorithm 2, respectively.

Step 1: First, the $x_{\setminus j}$ feature space is partitioned into disjoint regions of observations where all $x_{\setminus j}$ variables are approximately matched across the observations in that region. Stratification occurs through the use of a decision tree fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$. The leaves of the decision tree aggregate observations with equal or similar $x_{\setminus j}$ features. The $x_{\setminus j}$ features can be numerical variables or label-encoded categorical variables (assigned a unique integer). StratPD only uses the tree for the purpose of partitioning feature space and never uses predictions from any model. This use of decision trees is closely related to the work from Strobl et al. (2008) and Breiman and Cutler (2003) for conditional permutation importance and observation similarity measures, respectively.

Step 2: Once each variable has been partitioned via stratification, StratPD next estimates $\partial f/\partial x_j$ and $PD_j(z)$ through averaging. Within each $x_{\setminus j}$ region, any fluctuations in the response variable are likely due to the variable of interest, x_j. In the ideal case, the values for each $x_{\setminus j}$ variable within a region are identical and, because only x_j is changing, y changes should be attributed to x_j, noise, or irreducible error. Estimates of the partial derivative within a region are computed discretely as the changes in y values between unique and ordered x_j positions: $(\bar{y}^{(i+1)} - \bar{y}^{(i)})/(x_j^{(i+1)} - x_j^{(i)})$ for all i in a region such that $x_j^{(i)}$ is unique and $\bar{y}^{(i)}$ is the average y at unique $x_j^{(i)}$. This amounts to performing piecewise linear regression through the region observations, one model per unique pair of x_j values, and collecting the model β_1 coefficients to estimate partial derivatives. The overall partial derivative at $x_j = z$ is the average of all slopes, found in any region, whose range $[x_j^{(i)}, x_j^{(i+1)})$ spans z. One could apply a nonparametric method to smooth through the discontinuities at x_j points within a leaf, but this has not proven necessary in practice. The partial dependence curve points are often the result of two averaging operations, one within and one across regions, which tends to smooth the curve; e.g., see Figure 3(d) below.

For the stratification approach to work, decision tree leaves must satisfactorily stratify $x_{\setminus j}$. If the $x_{\setminus j}$ observations in each region are not similar enough, the relationship between x_j and y is less accurate. Regions can also become so small that even the x_j values become equal, leaving a single unique x_j observation in a leaf. Without a change in x_j, no partial derivative estimate is possible and these nonsupporting observations must be ignored (e.g., the two leftmost points in Figure 2(a)). A degenerate case occurs when identical or nearly identical x_j and x'_j variables exist. Stratifying x_j as part of $x_{\setminus j}$ would also match up x'_j values, leading to both exhibiting flat curves, as if the decision tree were trained on $(\mathbf{X}, \mathbf{y})$ not $(\mathbf{X}_{\setminus j}, \mathbf{y})$. Our experience is that using the collection of leaves from a random forest, which restricts the number of variables available during node splitting, prevents partitioning from relying too heavily on either x_j or x'_j. Some leaves have observations that vary in x_j or x'_j and partial derivatives can still be estimated.

StratPD uses a hyper parameter, `min_samples_leaf`, to control the minimum number of observations in each decision tree leaf. Generally speaking, smaller values lead to more confidence that fluctuations in y are due solely to x_j, but more observations per leaf prevent StratPD from missing nonlinearities and make it less susceptible to noise. As the leaf size grows, however, one risks introducing contributions from $x_{\setminus j}$ into the relationship between x_j and y. At the extreme, the decision tree would consist of a single leaf node containing all observations, leading to a marginal not partial dependence curve.

StratPD uses another hyper parameter called `min_slopes_per_x` to ignore any partial derivatives estimated with too few observations. Dropping uncertain partial derivatives greatly improves accuracy and stability. Partial dependences computed by integrating over local partial derivatives are highly sensitive to partial derivatives computed at the left edge of any x_j's range because imprecision at the left edge affects the entire curve. This presents a problem when there are few samples with x_j values at the extreme left (see, for example, the x_j histogram of Figure 8(d)). Fortunately, sensible defaults for StratPD (10 observations and 5 slopes) work well in most cases and were used to generate all plots in this paper.

For categorical explanatory variables, CatStratPD uses the same stratification approach, but cannot apply regression of y to non-ordinal, categorical x_j. Instead, CatStratPD groups leaf observations by category and computes the average response per category in each leaf. Consider a single leaf and its p-dimensional average response vector $\bar{\mathbf{y}}$. Then choose a random reference category, *refcat*, and subtract that category's average value from $\bar{\mathbf{y}}$ to get a vector of relative deltas between categories: $\Delta\mathbf{y} = \bar{\mathbf{y}} - \bar{\mathbf{y}}_{refcat}$. The $\Delta\mathbf{y}$ vectors from all leaves are then merged via averaging, weighted by the number of observations per category, to get the overall effect of each category on the response. The delta vectors for two leaves, $\Delta\mathbf{y}$ and $\Delta\mathbf{y}'$, can only be merged if there is at least one category in common. CatStratPD initializes a running average vector to an arbitrary starting leaf's $\Delta\mathbf{y}$ and then makes multiple passes over the remaining vectors, merging any vectors with a category in common with the running average vector. Observations associated with any remaining, unmerged leaves must be ignored. CatStratPD uses a single hyper parameter `min_samples_leaf` to control stratification. Both StratPD and CatStratPD have an optional hyper parameter called `ntrials` (default is 1) that averages the results from multiple bootstrapped samples, which can reduce variance.

Stratification of high-cardinality categorical variables tends to create small groups of category subsets, which complicates the averaging process across groups. (Such $\Delta\mathbf{y}$ vectors are sparse and we use *NaN*s to represent missing values.) If both groups have the same reference category, merging is a simple matter of averaging the two delta vectors, where $mean(z, NaN) = z$. For delta vectors with different reference categories and at least one category in common, one vector is adjusted to use a randomly-selected reference category, c, in common: $\Delta\mathbf{y}' = \Delta\mathbf{y}' - \Delta\mathbf{y}'_c + \Delta\mathbf{y}_c$. That equation adjusts vector $\Delta\mathbf{y}'$ so $\Delta\mathbf{y}'_c = 0$ then adds the corresponding value from $\Delta\mathbf{y}$ so $\Delta\mathbf{y}'_c = \Delta\mathbf{y}_c$, which renders the average of $\Delta\mathbf{y}$ and $\Delta\mathbf{y}'$ meaningful. See Algorithm 2 for more details.

StratPD and CatStratPD both have theoretical worst-case time complexity of $O(n^2)$ for n observations. For StratPD, stratification costs $O(pn\log n)$, computing y deltas for all observations among the leaves has linear cost, and averaging slopes across unique x_j ranges is on the order of $|unique(\mathbf{X}_j)| \times n$ or n^2 when all $\mathbf{X}_j$ are unique in the worst case. StratPD is, thus, $O(n^2)$ in the worst case. CatStratPD also stratifies in $O(pn\log n)$ and computes category deltas linearly in n but must make multiple passes over the $|T|$ leaves to average all possible leaf category delta vectors. In practice, three passes is the max we have seen (for high-cardinality variables), so we can assume the number of passes is some small constant to get a tighter bound. Averaging two vectors costs $|unique(\mathbf{X}_j)|$, so each pass requires $|T| \times |unique(\mathbf{X}_j)|$. The number of leaves is roughly n / `min_samples_leaf` and, worst-case, $|unique(\mathbf{X}_j)| = n$, meaning that merging dominates CatStratPD complexity leading to $O(n^2)$. Experiments show that our prototype is fast enough for practical use (see Section 4).

3. Related work

The mechanisms most closely related to StratPD and CatStratPD are FPD, ICE, SHAP, and ALE, which all define partial dependence in terms of impact on estimated models, $\hat{f}$, rather than the unknown true function f. Let x_S be the subset of features of interest where $S \subset F = \{1, 2, .., p\}$. Friedman (2000) defines the partial dependence as an expectation conditioned on the remaining variables:

$$FPD_S(x_S) = \mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\setminus S})], \tag{2}$$

where the expectation can be estimated by $\frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_{\setminus S}^{(i)})$. The Individual Conditional Expectation (ICE) plot (Goldstein et al., 2015) estimates the partial dependence of the prediction $\hat{f}$ on x_S, or single variable x_j, across individual observations. ICE produces a curve from the fitted model over all values of x_j while holding $x_{\setminus j}$ fixed: $\hat{f}_j^{(i)} = \widehat{f}(\{x_j^{(k)}\}_{k=1}^{n}, x_{\setminus j}^{(i)})$; the FPD curve for x_j is the average over all x_j ICE curves. The motivation for ICE is to identify variable interactions that average out in the FPD curve.

The SHAP method from Lundberg and Lee (2017) has roots in *Shapley regression values* (Lipovetsky and Conklin, 2001) and calculates the average marginal effect of adding x_j to models, $\hat{f}_S$, trained on all possible subsets of features:

$$\phi_j(\hat{f}, x_F) = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[\hat{f}_{S \cup \{j\}}(x_{S \cup \{j\}}) - \hat{f}_S(x_S)\right] \tag{3}$$

To avoid training a combinatorial explosion of models with the various feature subsets, $\hat{f}_S(x_S)$, SHAP reuses a single model fitted to $(\mathbf{X}, \mathbf{y})$ by running simplified feature vectors into the model. As Sundararajan and Najmi (2019) describes, there are many possible implementations for simplified vectors. One is to replace "missing" features with their expected value or some other baseline vector (BShap in Sundararajan and Najmi 2019). SHAP uses a more general approach ("interventional" mode) that approximates $\hat{f}_S(x_S)$ with $\mathbb{E}[\hat{f}(x_S, \mathbf{X}'_{\setminus S}) | \mathbf{X}'_S = x_S]$ where $\mathbf{X}'$ is called the *background set* and users can pass in, for example, a single vector with $\mathbf{X}_{\setminus S}$ column averages or even the entire training set, $\mathbf{X}$, which is what we will assume (and is called CES($\hat{D}$) in Sundararajan and Najmi 2019 where $\hat{D} = \mathbf{X}$). To further reduce computation costs, SHAP users typically explain a small subsample of the data set, but with potentially a commensurate reduction in the explanatory resolution of the underlying population. SHAP has model-type-dependent optimizations for linear regression, deep learning, and decision-tree based models.

For efficiency, SHAP approximates $\mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\setminus S}) | \mathbf{X}_S = x_S]$ with $\mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\setminus S})]$, which assumes feature independence. (Janzing et al., 2019) argues that "*unconditional* [as implemented] rather than *conditional* [as defined] expectations provide the right notion of dropping features." But, using the unconditional expectation makes the inner difference of Equation 3 a function of codependency-sensitive FPDs:

$$\begin{aligned} \hat{f}_{S\cup\{j\}}(x_{S\cup\{j\}}) - \hat{f}_S(x_S) &= \mathbb{E}[\hat{f}(x_{S\cup\{j\}}, \mathbf{X}_{\setminus(S\cup\{j\})})] - \mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\setminus S})] \\ &= FPD_{S\cup\{j\}}(\mathbf{x}) - FPD_S(\mathbf{x}) \end{aligned} \tag{4}$$

If the individual contributions are potentially biased, averaging the contributions of many such feature permutations might not lead to an accurate partial dependence. Even if the conditional expectation is used, Sundararajan and Najmi (2019) points out that SHAP is sensitive to the sparsity of $\mathbf{X}$ because condition $\mathbf{X}_S = x_S$ will find few or no training records with the exact x_S values of some input vector.

The goal of ALE (Apley and Zhu, 2016) is to overcome the bias in previous model-based techniques arising from extrapolations of $\hat{f}$ far outside the support of the training data in the presence of codependent variables. ALE partitions range $[\min(\mathbf{X}_j) .. \max(\mathbf{X}_j)]$ for variable x_j into K bins and estimates the "uncentered main effect" (Equation 15) at $x_j = z$ as the cumulative sum of the partial derivatives for all bins up to the bin containing z. ALE estimates the partial derivative of $\hat{f}$ at $x_j = z$ as $\mathbb{E}[\hat{f}(b_k, \mathbf{X}_{\setminus j}) - \hat{f}(b_{k-1}, \mathbf{X}_{\setminus j}) \,|\, x_j \in (b_{k-1}, b_k]]$ for bin b_k that contains z. They also extend ALE to two variables by partitioning feature space into K^2 rectangular bins and computing the second-order finite difference of $\hat{f}$ with respect to the two variables for each bin.

Another related technique that integrates over partial derivatives to measure x_j effects is called Integrated Gradients (IG) from Sundararajan et al. (2017). Given a single input vector, $\mathbf{x}$, to a deep learning classifier, IG integrates over the gradient of the model output function $\hat{f}$ at points along the path from a baseline vector, $\mathbf{x}'$, to $\mathbf{x}$. IG can be seen as computing the partial dependence of $\hat{f}$ at a single $\mathbf{x}$, but using multiple $\mathbf{x}$ vectors would yield an x_j partial dependence curve (relative to a baseline $\mathbf{x}'$).

StratPD is like a "model-free" version of ALE in that StratPD also defines partial dependence as the cumulative sum of partial derivatives, but we estimate derivatives using response values directly rather than $\hat{f}$ predictions. An advantage to estimating partial dependence via fitted models is that $\hat{f}$ removes variability from the potentially noisy response values, $\mathbf{y}$. However, in practice, this requires expertise to choose and tune an appropriate model for a data set. The fact that different models can lead to meaningfully different curves for the same data can lead to misinterpretation. Also, expertise is often required to distinguish between model artifacts and interesting visual phenomena arising from the data.

ALE partitions x_j into bins then fixes $x_{\setminus j}$ as it shifts x_j to bin edges to compute finite differences, as depicted in Figure 2(b) for $p = 2$. The wedges on the x_1 axis indicate the x_1 points of the computed partial dependence curve.

StratPD partitions $x_{\setminus j}$ into regions of (hopefully) similar observations and computes finite differences between the average y values at unique x_j values in each region, as depicted in Figure 2(a). The leftmost two observations are ignored as there is no change in x_1 in that leaf. The shaded area illustrates that the partial derivative at any $x_j = z$ is the average of all derivatives spanning z across $x_{\setminus j}$ regions. StratPD assumes all points within a region are identical in $x_{\setminus j}$, effectively projecting points in $x_{\setminus j}$ space onto a hyperplane if they are not. ALE shifts x_j values in a small neighborhood and StratPD depends on a suitable `min_samples_leaf` hyper parameter to prevent $x_{\setminus j}$ points in a regions from becoming too dissimilar. StratPD automatically generates more curve points in areas of high x_j density, but ALE is more efficient.

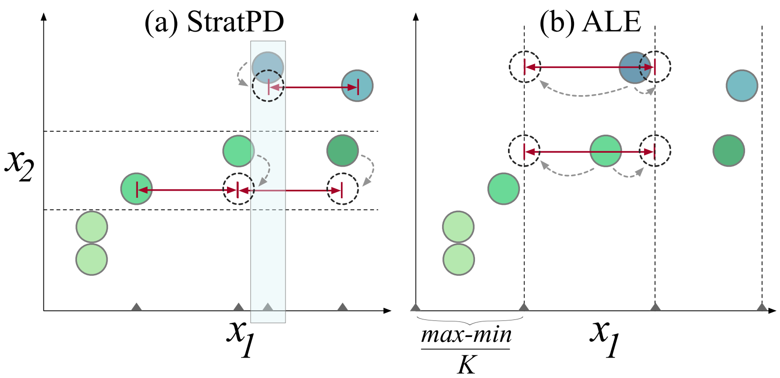

Figure 2. Comparison of StratPD, CatStratPD and ALE for $p = 2$ with continuous x_2 and continuous x_1 in (a), (b) and categorical x_1 in (c), (d). F=false and T=true. Vertical dashed lines are ALE bin edges and horizontal dashed lines are regions partitioned by a decision tree fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$. Whereas ALE shifts x_1 observations to bin edges holding x_2 exactly constant (and asks for predictions at those points), StratPD assumes x_2 values are the same and uses known y values. The small wedges on x_1 axis indicate partial dependence curve points. The shades of green and blue indicate y values. The leftmost two observations in (a) and the lowermost observation in (c) are ignored as finite differences are not defined. There is no curve value for the rightmost x_1 value in (a) due to forward differencing.

Using decision trees for the purpose of partitioning feature space as StratPD does was previously used by Strobl et al. (2008) to improve permutation importance for random forests by permuting x_j only within the observations of a leaf. Earlier, Breiman and Cutler (2003) defined a similarity measure between two observations according to how often they appear in the same leaf in a random forest. Rather than partitioning $x_{\setminus j}$ space like StratPD, those techniques partitioned all of x space.

The model-based techniques under discussion treat boolean and label-encoded categorical variables (encoded as unique integers) as numerical variables, even though there is no defined order, as depicted in Figure 2(d). ALE does, however, take advantage of the lack of order to choose an x_j order that reduces "extrapolation outside the data envelope" by measuring the similarity of $\mathbf{X}_{\setminus j}$ sample values across x_j categories. Adjacent category integers, though, could still represent the most semantically different categories, so any shift of an observation's category to extrapolate is risky. Even the smallest possible extrapolation can conjure up nonsensical observations, such as pregnant males, as we demonstrate in Figure 7 below where FPD, SHAP, and ALE underestimate pregnancy's effect on body weight. (For boolean x_j, ALE behaves like FPD.) See Hooker and Mentch (2019) for more on the dangers of permuting features.

In contrast, CatStratPD uses a different algorithm for categoricals and computes differences between the average y for all categories within the leaf to a random reference category; see Figure 2(c). These leaf delta vectors are then merged across leaves to arrive at an overall delta vector relating the relative effect of each category on y. One could argue that StratPD also extrapolates because $x_{\setminus j}$ could include categorical variables and not all $x_{\setminus j}$ records would be identical. But, our approach only assumes $x_{\setminus j}$ values are similar and uses known training y values for finite differences, rather than asking a model to make prediction for nonsensical records, which could be wildly inaccurate. Also, the decision tree would, by definition, likely partition $x_{\setminus j}$ space into regions that can be treated similarly, thus, grouping semantically similar categorical levels together.

4. Numerical Study

In this section, we demonstrate experimentally that StratPD and CatStratPD compute accurate partial dependence curves for synthetic data and plausible results for a real data set. Experiments also provide evidence that existing model-based techniques can provide meaningfully-biased curves. We begin by comparing the partial dependence curves from popular techniques on synthetic data with complex interactions.[1]

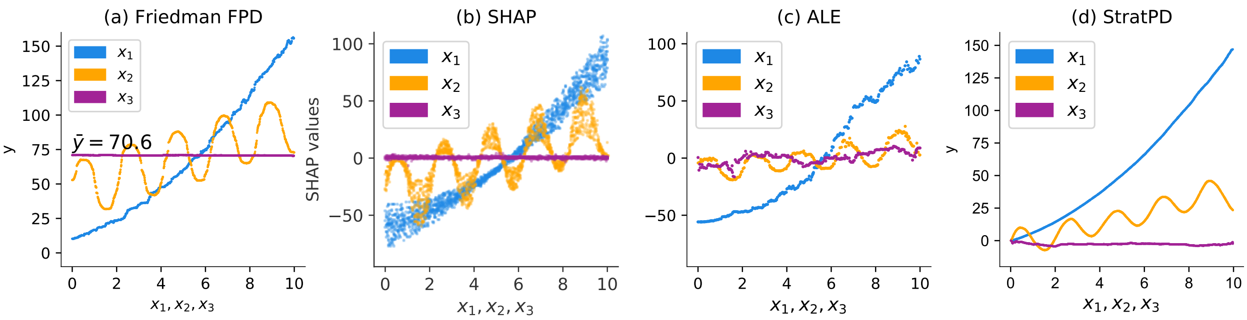

Figure 3. Partial dependence plots of $n = 2000$ data generated from noiseless $y = x_1^2 + x_1x_2 + 5x_1 sin(3x_2) + 10$ where $x_1, x_2, x_3 \sim U(0, 10)$ and x_3 does not affect y. For ALE, SHAP, and FDP, a random forest model is fit with 10 trees trained on all data ($R^2 = 0.997$). SHAP used all $\mathbf{X}$ as background data and ALE used $K = 300$. The curves were generated from same 2000 data points that the model was trained on.

Figure 3 illustrates that FPD, SHAP, ALE, and StratPD all suitably isolate the effect of independent individual variables on the response for noiseless data generated via: $y = x_1^2 + x_1x_2 + 5x_1 sin(3x_2) + 10$ for $x_1, x_2, x_3 \sim U(0, 10)$ where x_3 does not affect y. The shapes of the curves for all techniques look similar except that StratPD starts all curves at $y = 0$ (as could the others). SHAP's curves have the advantage that they indicate the presence of variable interactions. To our eye, StratPD's curves are smoothest despite not having access to model predictions.

Models have a tendency to smooth out noise and a legitimate concern is that, without the benefit of a model, StratPD could be adversely affected. Figure 4 demonstrates StratPD curves for noisy quadratics generated from $y = x_1^2 + x_2 + 10 + \epsilon$ where $\epsilon \sim N(0, \sigma)$ ϵ and, at $\sigma = 2$, 95% of the noise falls within [0,4] (since $2\sigma = 4$), meaning that the signal-to-noise ratio is at best 1-to-1 for x_1^2 in $[-2, 2]$. For zero-centered Gaussian noise and this data set, StratPD appears resilient, though $\sigma = 2$ does show considerable variation across runs. The superfluous noise variable x_3 in Figure 3 also did not confuse StratPD.

Turning to categorical variables, Figure 5 presents partial dependence curves for FPD, ALE, and CatStratPD derived from a noisy synthetic weather data set, where temperature varies in sinusoidal fashion over the year and with different baseline temperatures per state. (The vertical "smear" in the FPD plot shows the complete sine waves but from the side, edge on.) Variable `state` is independent and all plots identify the baseline temperature per state correctly.

[1] All simulations in this section were run on a 4.0 Ghz 32G RAM machine running OS X 10.13.6 with SHAP 0.34, scikit-learn 0.21.3, XGBoost 0.90, TensorFlow 2.1.0, and Python 3.7.4; ALEPlot 1.1 and R 3.6.3. A single random seed was used across simulations for graph reproducibility purposes.

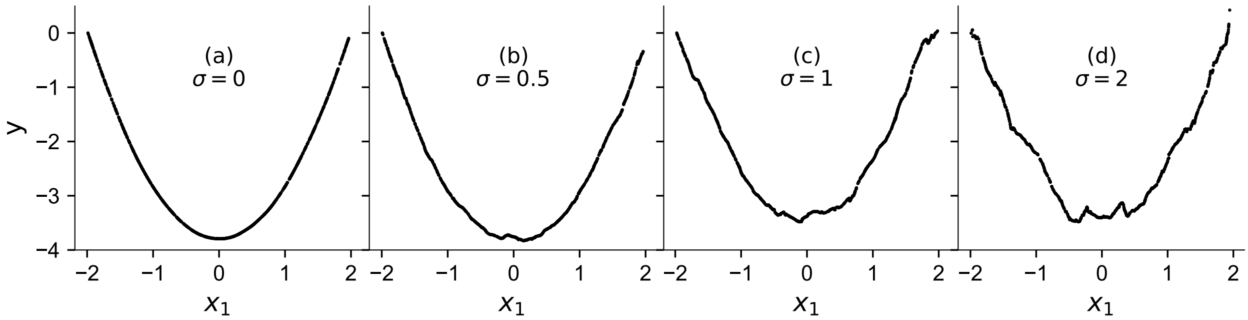

Figure 4. The effective noise on StratPD for $y = x_1^2 + x_1 + 10 + \epsilon$ where $x_1, x_2 \sim U(-2, 2)$, $\epsilon \sim N(0, \sigma)$ with $\sigma \in \{0, 0.5, 1, 2\}$. The 95% interval for amplitude of the noise in (d) for $\sigma = 2$ is the same as the signal.

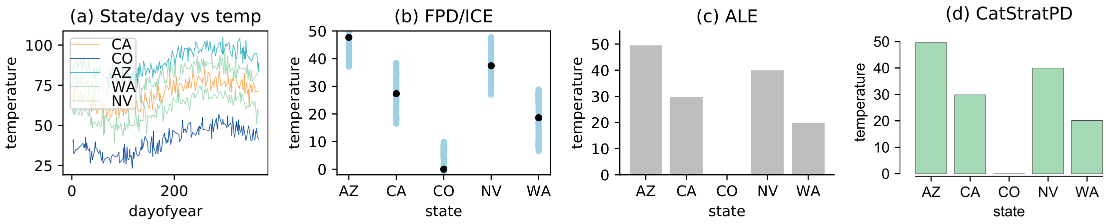

Figure 5. $y = base[x_{state}] + 10sin(\frac{2\pi}{365}x_{dayofyear} + \pi) + \epsilon$ where $\epsilon \sim N(\mu = 0, \sigma = 4)$. The *base* temperature per state is $\{AZ = 90, CA = 70, CO = 40, NV = 80, WA = 60\}$. Sinusoids in (a) are the average of three years' temperature data.

The primary goal of the stratification approach proposed in this paper is to obtain accurate partial dependence curves in the presence of codependent variables. To test StratPD/CatStratPD and discover potential bias in existing techniques, we synthesized a body weight data set generated by the following equation with nontrivial codependencies between variables:

$$
\begin{aligned}
y \quad &= \quad 120 + 10(x_{height} - min(x_{height})) + 40x_{pregnant} - 1.5x_{education} \\
\text{where} \quad & x_{sex} \sim Bernoulli(\{M, F\}, p = 0.5) \\
& x_{pregnant} = \begin{cases} Bernoulli(\{0, 1\}, p = 0.5) & \text{if } x_{sex} = F \\ 0 & \text{if } x_{sex} = M \end{cases} \\
& x_{height} = \begin{cases} 5 * 12 + 5 + \epsilon & \text{if } x_{sex} = F,\ \epsilon \sim U(-4.5, 5) \\ 5 * 12 + 8 + \epsilon & \text{if } x_{sex} = M,\ \epsilon \sim U(-7, 8) \end{cases} \\
& x_{education} = \begin{cases} 12 + \epsilon & \text{if } x_{sex} = F,\ \epsilon \sim U(0, 8) \\ 10 + \epsilon & \text{if } x_{sex} = M,\ \epsilon \sim U(0, 8) \end{cases}
\end{aligned}
\tag{5}
$$

The partial derivative of y with respect to x_{height} is 10 (holding all other variables constant), so the optimal partial dependence curve is a line with slope 10. Figure 6 illustrates the curves for the techniques under consideration, with ALE and StratPD giving the sharpest representation of the linear relationship. (StratPD's curve is drawn on top of the SHAP plots using the righthand scale.) The FPD and both SHAP plots also suggest a linear relationship, albeit with a little less precision. The ICE curves in Figure 6(a) and "fuzzy" SHAP curves have the advantage that they alert users to variable dependencies or interaction terms. On the other hand, the kink in the partial dependence curve and other visual phenomena could confuse less experienced machine learning practitioners and certainly analysts and researchers in other fields (our primary target communities).

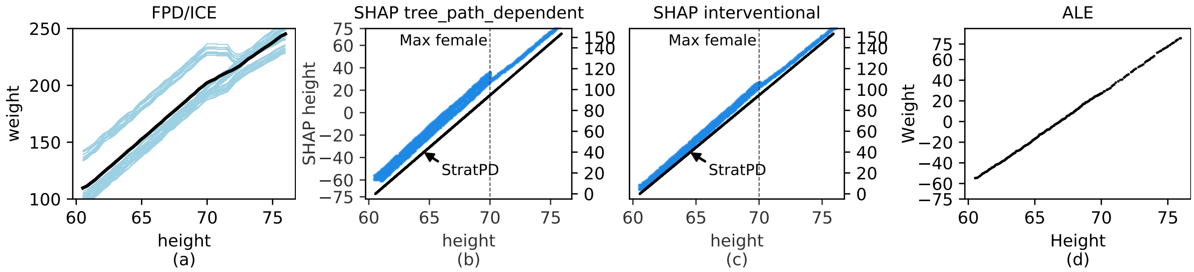

Figure 6. Partial dependence plots of response body weight on feature x_{height} using n=2000 synthetic observations from Equation 5. FPD and SHAP have "kinks" at the maximum female height. SHAP defines feature importance as the average SHAP value magnitude, which overemphasizes importance for heights below 70 inches here. The male/female ratio is 50/50, half of the women are pregnant, and pregnancy contributes 40 pounds. SHAP interrogated an RF tuned via 5-fold cross validation grid search (OOB R^2 0.999) and explained all 2000 samples; the interventional case used 100 observations as background data. ALE used K = 300.

Even for experts, explaining this behavior requires some thought, and one must distinguish between model artifacts and interesting phenomena. The discontinuity at the maximum female height location arises partially from the model having trouble extrapolating for extremely tall pregnant women. Consider one of the upper ICE lines in Figure 6(a) for a pregnant woman. As the ICE line slides x_{height} above the maximum height for a woman, the model leaves the support of the training data and predicts a *lower* weight as height increases (there are no pregnant men in the training data). ALE's curve is straight because it focuses on local effects, demonstrating that the lack of sharp slope-10 lines for FPD and SHAP cannot be attributed simply to a poor choice of model.

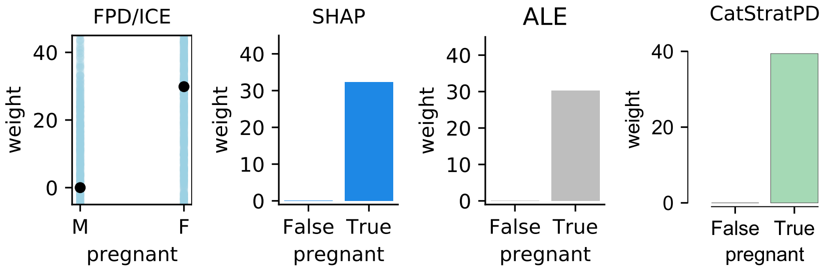

Figure 7. Partial dependence bar charts for boolean $x_{pregnant}$ with the same model from Figure 6. Only CatStratPD gets the correct 40 pound contribution per Equation 5.

Also, SHAP defines x_j feature importance as the average magnitude of the x_j SHAP values, which introduces a paradox. The spread of the SHAP values alerts users to variable interactions, but allows contributions from other variables to leak in, thus, potentially leading to less precise estimates of x_{height}'s importance. The average SHAP magnitude skews upward, in this case, because of the contributions from pregnant women.

It is conceivable that a more sophisticated model (in terms of extrapolation) could sharpen the FPD and SHAP curves for x_{height}. There is a difference, however, between extrapolating to a meaningful but unsupported vector and making predictions for nonsensical vectors arising from variable codependencies. Techniques that rely on such predictions make the implicit assumption of variable independence, introducing the potential for bias. Consider Figure 7 that presents the partial dependence results for categorical variable $x_{pregnant}$ (same data set). The weight gain from pregnancy is 40 pounds per Equation 5, but only CatStratPD identifies that exact relationship; FPD, SHAP, and ALE show a gain of 30 pounds.

CatStratPD stratifies persons with the same or similar sex, education, and height into groups and then examines the relationship between $x_{pregnant}$ and y. If a group contains both pregnant and nonpregnant females, the difference in weight will be 40 pounds in this noiseless data set (if we assume identical $x_{\setminus j}$). FPD and ALE rely on computations that require fitted models to conjure up predictions for nonsensical records representing pregnant males (e.g., $\hat{f}(x_j = pregnant, \mathbf{X}_{\setminus j})$). Not even a human knows how to estimate the weight of a pregnant male. SHAP, per its definition, does not require such predictions, but in practice for efficiency reasons, SHAP approximates $\mathbb{E}[\hat{f}(x_j = pregnant, \mathbf{X}_{\setminus j}) | \mathbf{X}_j = x_j]$ with $\mathbb{E}[\hat{f}(x_j = pregnant, \mathbf{X}_{\setminus j})]$, which does not restrict pregnancy to females. As discussed above, there are advantages to all of these model-based techniques, but this example demonstrates there is potential for partial dependence bias.

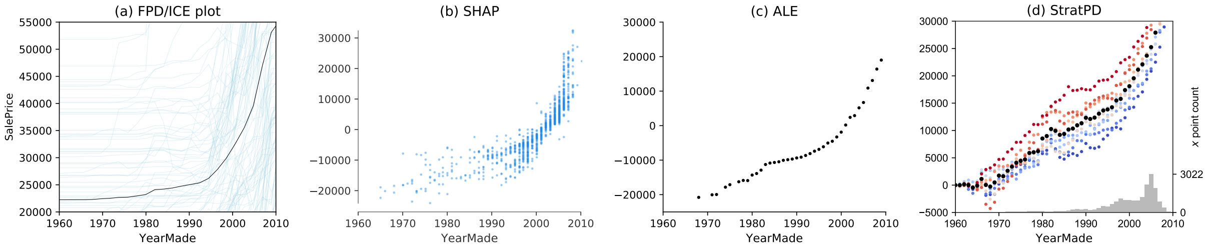

Figure 8. Partial dependence curves of bulldozer `YearMade` versus `SalePrice` for FPD/ SHAP, ALE, and StratPD. n=20,000 observations drawn from ˜362k. SHAP interrogated a random forest (OOB $R^2 = 0.85$) to explain 1000 training observations with 100 observations as background data. ALE used $K = 300$. Hyper parameters were tuned using 5-fold cross validation grid search over several hyper parameters.

The stratification approach also gives plausible results for real data sets, such as the bulldozer auction data from Kaggle (2018). Figure 8 shows the partial dependence curves for the same set of techniques as before on feature `YearMade`, chosen as a representative because it is very predictive of sale price. The shape and magnitude of the FPD, SHAP, ALE, and StratPD curves are similar, indicating that older bulldozers are worth less at auction, which is plausible. The StratPD curve shows 10 bootstrapped trials where the heavy black dots represent the partial dependence curve and the other colored curves describe the variability.

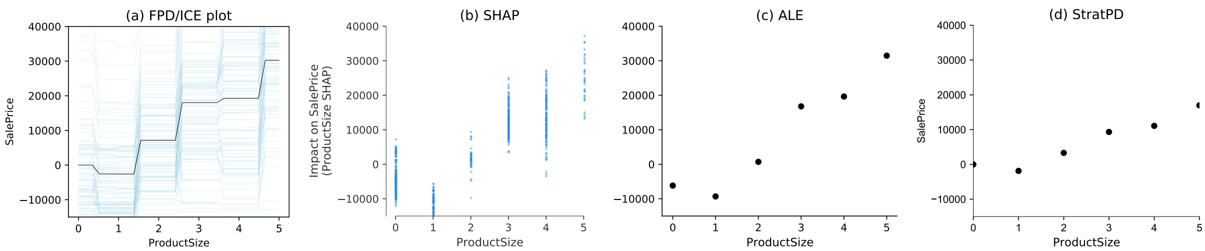

Figure 9. Partial dependence curves of bulldozer `ProductSize` versus `SalePrice` for FPD/ICE, SHAP, ALE, and StratPD. n=20,000 observations drawn from ˜362k. Same model, setup as in Figure 8. The StratPD dots are the average of 10 bootstrapped trials and the partial dependence for `ProductSize` is missing because a forward difference is unavailable at the right edge.

As a second example, consider the curves for feature `ProductSize` in Figure 9. All plots have roughly the same shape but the StratPD plot lacks a dot for `ProductSize`=5 because forward differences are unavailable at the right edge. (We anticipate switching switching to a central difference to avoid this issue with low-cardinality discrete variables.) It also appears that StratPD considers `ProductSize`'s 3 and 4 to be worth less than suggested by the other techniques, though StratPD's might be in line with the average SHAP plot values.

And, finally, an important consideration for any tool is performance, so we plotted execution time versus data size (up to 30,000 observations) for three real Kaggle data sets: rent, bulldozer, and flight arrival delays (Kaggle, 2015). Figure 10 shows growth curves for 40 numerical variables and 11 categorical variables grouped by type of variable. For these data sets, StratPD takes 1.2s or less to process 30,000 records for any numerical x_j, despite the potential for quadratic cost. CatStratPD typically processes categorical variables in less than 2s but takes 13s for the high-cardinality categorical `ModelID` of bulldozer (which looks mildly quadratic). These elapsed times for our prototype show it to be practical and competitive with FPD/ICE, SHAP, and ALE. If the cost to train a model using (cross validated) grid search for hyper parameter tuning is included, StratPD and CatStratPD outperform these existing techniques (as training and tuning is often measured in minutes).

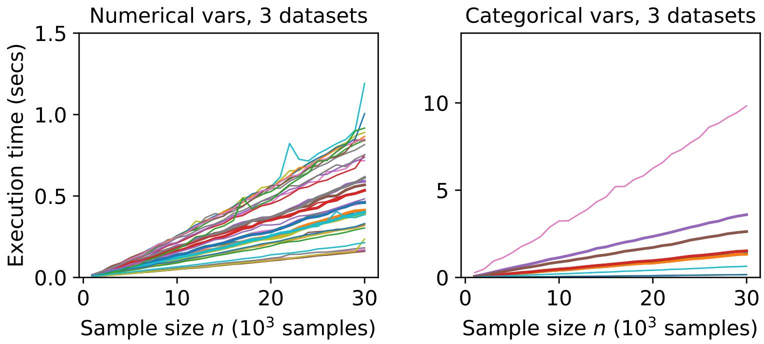

Figure 10. Time to compute partial dependence curve for up to 30,000 observations for 40 numerical and 11 categorical variables. p = 20, bulldozer p = 14, and flight p = 17. The Numba just-in-time compiler is used to improve performance, but timing does not include compilation; users do experience this "warm-up" time.

5. Discussion and future work

In this paper, we contribute a method for computing partial dependence curves, for both numerical and categorical explanatory variables, that does not use predictions from a fitted model. Working directly from the data makes partial dependences accessible to business analysts and scientists not qualified to choose, tune, and assess machine learning models. For experts, it can provide hints about the relationships in the data to help guide their choice of model. Our experiments show that StratPD and CatStratPD are fast enough for practical use and correctly identify partial dependences for synthetic data and give plausible curves on real data sets. StratPD relies on two important hyper parameters (with broadly applicable defaults) but model-based techniques should include the hyper parameters of the required fitted model for a fair comparison. Our goal here is not to argue that model-based techniques are not useful. Rather, we are pointing out potential issues and hoping to open a new line of nonparametric inquiry that experiments have shown to be applicable in situations and accurate in cases where model-based techniques are not. We aim to next generalize this approach for classifiers and to extend the technique to two or more variables.

Declaration of Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT Author Statement

Terence Parr: Conceptualization, Investigation, Methodology, Software, Writing- Original draft preparation. **James D. Wilson**: Conceptualization, Investigation, Validation, Writing- Reviewing and Editing.

Acknowledgments

JDW acknowledges partial support from the National Science Foundation, grant NSF DMS-1830547.

References

Apley, D. W., Zhu, J., 2016. Visualizing the effects of predictor variables in black box supervised learning models. arXiv preprint arXiv:1612.08468.

Breiman, L., Cutler, A., 2003. Random forests website. `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`, accessed: 2019-05-24.

Cox, D. R., Wermuth, N., 2014. Multivariate dependencies: Models, analysis and interpretation. Chapman and Hall/CRC.

Friedman, J. H., 2000. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232.

Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E., 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. Journal of Computational and Graphical Statistics 24 (1), 44–65.
URL `https://doi.org/10.1080/10618600.2014.907095`

Hooker, G., 2007. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. Journal of Computational and Graphical Statistics 16 (3), 709–732.
URL `https://doi.org/10.1198/106186007X237892`

Hooker, G., Mentch, L., 2019. Please stop permuting features: An explanation and alternatives.

Janzing, D., Minorics, L., Blöbaum, P., 2019. Feature relevance quantification in explainable ai: A causal problem.

Kaggle, 2015. 2015 flight delays and cancellations. `https://www.kaggle.com/usdot/flight-delays`, accessed: 2019-12-01.

Kaggle, 2017. Two sigma connect: Rental listing inquiries. `https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries`, accessed: 2019-06-15.

Kaggle, 2018. Blue book for bulldozer. `https://www.kaggle.com/sureshsubramaniam/blue-book-for-bulldozer-kaggle-competition`, accessed: 2019-06-15.

Lipovetsky, S., Conklin, M., 10 2001. Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry 17, 319 – 330.

Lundberg, S. M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 4765–4774.
URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., Zeileis, A., 2008. Conditional variable importance for random forests. BMC Bioinformatics 9 (1), 307.
URL `https://doi.org/10.1186/1471-2105-9-307`

Sundararajan, M., Najmi, A., 2019. The many shapley values for model explanation. ArXiv abs/1908.08474.

Sundararajan, M., Taly, A., Yan, Q., 2017. Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17. JMLR.org, p. 3319–3328.

Appendix A. Pseudocode for StratPD and CatStratPD

Algorithm 1: *StratPD*($\mathbf{X}$, $\mathbf{y}$, j, *min_samples_leaf*, *min_slopes_per_x*)

T := Decision tree regressor fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$ with hyper parameter: *min_samples_leaf*

for *each leaf* $L \in T$ **do**

$\mathbf{x}_L := \{x_j^{(i)}\}_{i \in L}$, the unique and ordered x_j in

$L\bar{\mathbf{y}}_L^{(k)} := \mathbb{E}[y^{(i)} \mid (x_j^{(i)} = x_L^{(k)}, y^{(i)})]$ for $k = 1..|\mathbf{x}_L|, i \in L\delta_L^{(k)} = \frac{\bar{\mathbf{y}}_L^{(k+1)} - \bar{\mathbf{y}}_L^{(k)}}{\mathbf{x}_L^{(k+1)} - \mathbf{x}_L^{(k)}}$Add tuples $(\mathbf{x}_L^{(k)}, \mathbf{x}_L^{(k+1)}, \ \delta_L^{(k)})$ to list $\mathbf{D}$ for $k = 1..|\mathbf{x}_L| - 1$

$\mathbf{ux} := \{x_j^{(i)}\}_{i \in 1..n}$, the unique and ordered x_j in $\mathbf{X}_j$

Let $\mathbf{c}$ and δ be vectors of length $|\mathbf{ux}|$

for *each* $x \in$ **ux do** ▷ *Count slopes and compute average slope per unique x_j value*

$\mathbf{slopes}_x := [slope$ for $(a, b, slope) \in \mathbf{D}$ if $x \geq a$ and $x < b]$

$\mathbf{c}_x := |\mathbf{slopes}_x| \delta_x := \overline{\mathbf{slopes}_x}$

$\delta := \delta[\mathbf{c} \geq min\_slopes\_per\_x]\mathbf{ux} := \mathbf{ux}[\mathbf{c} \geq min\_slopes\_per\_x]\mathbf{pd}_x := \mathbf{ux}^{(k+1)} - \mathbf{ux}^{(k)}$ for $k = 1..|\mathbf{ux}| - 1$

$\mathbf{pd}_y := [0] +$ cumulative_sum$(\delta \times \mathbf{pd}_x)$ **return** $\mathbf{pd}_x, \mathbf{pd}_y$

Algorithm 2: *CatStratPD*($\mathbf{X}$, $\mathbf{y}$, j, *min_samples_leaf*)

$n_{cats} := |\{x_j^{(i)}\}_{i \in 1..n}|$, $n_{leaves} := |T|$

T := Decision tree regressor fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$ with hyper parameter: *min_samples_leaf*

Let ΔY be a $n_{cats} \times n_{leaves}$ matrix whose columns, ΔY_L, are vectors of leaf category deltas

Let C be a $n_{cats} \times n_{leaves}$ matrix whose columns, C_L, are vectors for leaf category counts

for *each leaf* $L \in T$ **do** ▷ *Get average y delta relative to random ref category for obs. in leaves*

$\mathbf{x}_L := \{x_j^{(i)}\}_{i \in L}$, the unique x_j categories in

$L\bar{\mathbf{y}}_L^{(k)} := \mathbb{E}[y^{(i)} \mid (x_j^{(i)} = x_L^{(k)}, y^{(i)})]$ for $k = 1..|\mathbf{x}_L|, i \in LC_L^{(k)} := |i \in L : x_j^{(i)} = \mathbf{x}_L^{(k)}| refcat :=$ random category chosen from $\mathbf{x}_L$

$\Delta Y_L = \bar{\mathbf{y}}_L - \bar{\mathbf{y}}_L^{refcat}$

$\Delta\mathbf{y}, \mathbf{c} := \Delta Y_1, C_1 completed := \{L_1\}; work := \{L_2..L_{n_{leaves}}\};$ **while** $|work| > 0$ *and* $|completed| > 0$ **do** ▷ *2 passes is typical to merge all ΔY_L into $\Delta\mathbf{y}$*

completed := $\emptyset$

for *each leaf L in work* **do**

Let *common* := categories in common between ΔY_L and $\Delta\mathbf{y}$

if $common \neq \emptyset$ **then**

$completed := completed \cup \{L\}$

cat := random category in *common*

$\Delta Y_L := \Delta Y_L - \Delta Y_L^{(cat)} + \Delta\mathbf{y}^{(cat)}\Delta\mathbf{y} := (\mathbf{c} \times \Delta\mathbf{y} + C_L \times \Delta Y_L)/(\mathbf{c} + C_L)$ where $z + NaN{=}z\mathbf{c} := \mathbf{c} + C_L$

$work := work \setminus completed$

return $\Delta\mathbf{y}$