Technical Report: Partial Dependence without Model Predictions through Stratification

Terence Parr
University of San Francisco
parrt@cs.usfca.edu

James D. Wilson
University of San Francisco
jdwilson4@usfca.edu

Abstract

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. For example, in medicine, 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 are affecting defect rates. Climate scientists are interested in how different atmospheric carbon levels affect temperature.

For an explanatory matrix, $\mathbf{X}$, with a single 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. Variable interactions, codependencies between features, result in marginal plots that do not isolate the specific contribution of a feature of interest to the target. For example, a marginal plot of sex (male/female) 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.

Rather than looking directly at the data, there are several partial dependence techniques that interrogate fitted models provided by the user: Friedman's original partial dependence (which we will denote FPD) Friedman [2000], 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]. Model-based techniques dominate the partial dependence research literature because interpreting the output of a fitted model has several advantages. Models have a tendency to smooth over noise. Models act like analysis preprocessing steps, potentially reducing the computational burden on model-based partial dependence techniques; 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.

Model-based techniques do have two significant disadvantages, however. The first relates to their ability to tease apart the effect of codependent features because models are required
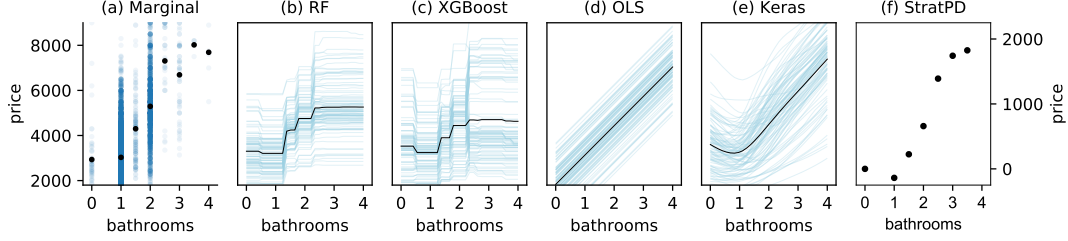
Figure 1: Plots of bathrooms versus rent price using New York City apartment rent data. (a) marginal plot, (b) PD/ICE plot derived from random forest, (c) PD/ICE plot derived from gradient boosted machine, and (d) PD/ICE plot derived from ordinary least squares regression; sample size is 10,000 observations of ~50k. The PD/ICE plots are different for the same data set, depending on the chosen user model. X['bathrooms'].unique() shows (array([0. , 1. , 1.5, 2. , 2.5, 3. , 3.5, 4.]), array([54, 8151, 140, 1539, 39, 67, 3, 7])). STRATPD has missing last value, not enough data. what are R^2 values. how tuned all but last share y with left

to extrapolate into regions of sparse or nonexistent support; e.g., see discussions in Apley and Zhu [2016] and Hooker [2007]. As we demonstrate in Section 4, using synthetic and real data sets, model-based techniques can vary in their ability to isolated variable effects in practice. Second, recall that there are vast armies of business analysts and scientists at work that need to analyze data, in a manner akin to exploratory data analysis (EDA), that have no intention of creating a predictive model. Either they have no need, perhaps needing only partial dependence plots, or they do not have the expertise to choose, tune, and assess models (or write software at all).

Even in the case where a machine learning practitioner is available to create a fitted model for the analyst, hazards exist. First, if a fitted model is unable to accurately capture the relationship between features and y accurately, 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." Second, given an accurate fitted model, business analysts and scientists are still peering 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 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 validity.

Experts are often able to quickly recognize model artifacts, such as the stairstep phenomenon inherent to the decision tree-based methods in Figure 1(b) and (c). 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, like business analysts, who lack the expertise to create suitable models and would also reduce the chance of plot misinterpretation due to model artifacts. The curves could also help machine learning practitioners to choose appropriate models based upon relationships exposed in the data.

In this paper, we propose a strategy, called STRATified Partial Dependence (StratPD), that (*i*) computes partial dependences directly from training data $(\mathbf{X}, \mathbf{y})$, rather than through the predictions of a fitted model, and (*ii*) does not presume mutually-independent features. As an example, Figure 1(f) shows the partial dependence plot computed by StratPD. The technique depends on 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. Colloquially, the approach examines changes in y across x_j while holding $x_{\backslash j}$ constant or nearly constant ($x_{\backslash j}$ denotes all observations with all variables except x_j). A similar stratification approach works for categorical variables (CatStratPD). Both StratPD and CatStratPD are quadratic in n, in the worst case (like FPD), but StratPD behaves linearly on real data sets. Our prototype is currently limited to regression, isolates only single-variable partial dependence, and cannot identify interaction effects (as ICE can). The software is available via Python package `stratx` with source code at `github`, including the code to regenerate images in this paper.

We begin by describing the proposed stratification approach in Section 2 then compare StratPD to related (model-based) work in Section 3. In Section 4, we present partial dependence curves generated by StratPD and CatStratPD on real data sets, contrast the plots with those of existing methods, and use synthetic data to highlight possible bias in some model-based methods.

2 Partial dependence without model predictions

In special circumstances, we know the precise effect of each feature x_j on y. Assume we are given training data pair $(\mathbf{X}, \mathbf{y})$ 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 $f : \mathbb{R}^p \to \mathbb{R}$ that precisely maps each $x^{(i)}$ 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 smooth generator function $f : \mathbb{R}^p \to \mathbb{R}$ evaluated at $x_j = z$ is the cumulative sum up to z **TODO:** do we need to say that the partial derivative is continuous also?:

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

$PD_j(z)$ is the value contributed to f by x_j at $x_j = z$ and $PD_j(min(x_j)) = 0$. The advantages of this definition are that it does not depend on predictions from a fitted model and is insensitive to codependent features. Although the underlining generator function is unknown, we can estimate its partial derivatives from the raw training data. ALE also derives partial dependence by estimating and integrating across partial derivatives (e.g., see Equation 7 in Apley and Zhu [2016]) but does so using local changes in fitted model predictions.

The key idea is to stratify $x_{\backslash j}$ feature space into disjoint regions of observations where all $x_{\backslash j}$ variables are approximately matched across the observations in that region. Within each $x_{\backslash j}$ region, any fluctuations in the response variable are likely due to the variable of interest, x_j. Estimates of the partial derivative within a region are estimated discretely as the changes in y values between unique and ordered x_j positions: $(y^{(i+1)} - y^{(i)})/(x_j^{(i+1)} - x_j^{(i)})$ for all i in a region. This amounts to performing piecewise linear regression through the leaf 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.

Stratification occurs through the use of a decision tree fit to $(\mathbf{X}_{\backslash j}, \mathbf{y})$, whose leaves aggregate observations with equal o I don't click on random links buddyr similar $x_{\backslash j}$ features. The $x_{\backslash j}$ 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. See Algorithm 1 for more details.

Using decision trees for the purpose of partitioning feature space was previously used by Strobl et al. [2008] to improve permutation importance 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.

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. A degenerate case occurs when identical or nearly identical x_j and x'_j variables exist. Flattening x_j as part of $x_{\setminus j}$ would also flatten x'_j, 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 experiments show 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 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 allow STRATPD to capture more 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 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 $\bar{y}$ per category. Then, within each leaf, it chooses a random reference category and subtracts that category's $\bar{y}$ value from the leaf $\bar{\mathbf{y}}$ vector 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 y. 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 the first leaf's $\Delta\mathbf{y}$ and then makes 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.

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 *NaN*s 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. CATSTRATPD uses a single hyper parameter `min_samples_leaf` to control stratification. 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. Averaging two vectors costs $|unique(\mathbf{X}_j)|$, so each pass requires $|T| \times |unique(\mathbf{X}_j)|$. The number of leaves is roughly $n/\texttt{min\_samples\_leaf}$ 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

Friedman [2000] defines the partial dependence of $\hat{f}$ on feature subsets, $S \subset F = \{1, 2, .., p\}$, as an expectation conditioned on the remaining variables:

$$FPD_S(x_S) = \mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\backslash S})] = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_{\backslash S}^{(i)})$$

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 $x_{\backslash j}$ fixed: $\hat{f}_j^{(i)} = \widehat{f}(\{x_j^{(k)}\}_{k=1}^n, x_j^{(i)})$; the FPD curve for x_j is the average over all ICE curves for x_j. The motivation for ICE is to identify variable interactions that average out in the FPD curve.

SHAP 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 trained on all possible subsets of features:

$$\phi_j(\hat{f}, \mathbf{x}) = \sum_{S \subseteq F \backslash \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [\hat{f}_{S \cup \{j\}}(x_{S \cup \{j\}}) - \hat{f}_S(x_S)] \qquad (2)$$

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] points out, 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 $\mathbb{E}[\hat{f}(x_S, \mathbf{X}'_{\backslash S})|\mathbf{X}'_S = x_S]$ where $\mathbf{X}'$ is called the *background set* and users can pass in, for example, a single vector with $\mathbf{X}_{\backslash S}$ 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 very 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, again, SHAP approximates $\mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\backslash S})|\mathbf{X}_S = x_S]$ with $\mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\backslash S})]$, 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 2 a function of codependency-sensitive FPDs:

$$\hat{f}_{S\cup\{j\}}(x_{S\cup\{j\}}) - \hat{f}_S(x_S) = \mathbb{E}[\hat{f}(x_{S\cup\{j\}}, \mathbf{X}_{\backslash(S\cup\{j\})})] - \mathbb{E}[\hat{f}(x_S, \mathbf{X}_{\backslash S})]$$
$$= FPD_{S\cup\{j\}}(\mathbf{x}) - FPD_S(\mathbf{x})$$

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 $\mathbb{E}[\hat{f}(b_k, \mathbf{X}_{\backslash j}) - \hat{f}(b_{k-1}, \mathbf{X}_{\backslash j}) \,|\, x_j \in (b_{k-1}, b_k]]$ 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 network, IG integrates over the loss function gradient at points along the path from a baseline vector, $\mathbf{x}'$, to $\mathbf{x}$. IG can be seen as computing the partial dependence of the loss function, rather than model output, $\hat{f}$, on the x_j values in $\mathbf{x}$.
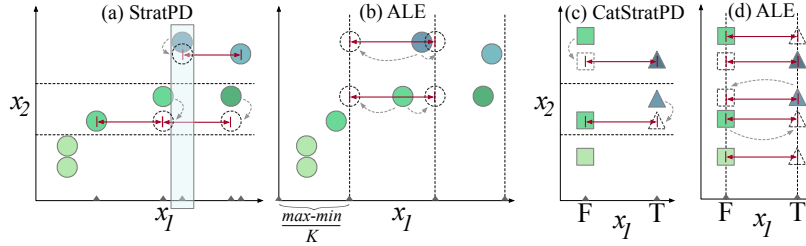

Figure 2: x_2 continuous and partitioned by decision tree fit to $(x_{\backslash j}, \mathbf{y})$, x_1 continuous (a), (b) and categorical (c), (d). Small wedges on x_1 axis indicate PD curve points

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 $\mathbf{y}$ values rather than $\hat{f}$ predictions. ALE partitions x_j into bins then fixes $x_{\backslash j}$ 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 $x_{\backslash j}$ 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 $x_{\backslash j}$ regions. StratPD assumes all points within a region are identical in $x_{\backslash j}$, effectively projecting points in $x_{\backslash j}$ space onto a hyperplane if they are not. ***TODO:*** sound right James? ALE shifts x_j values in a small neighborhood and StratPD depends on a suitable `min_samples_leaf` hyper parameter to prevent $x_{\backslash j}$ 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.

The model-based techniques under discussion treat label-encoded categorical variables (encoded as unique integers) as numerical variables, even though there is no defined order. 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 $\mathbf{X}_{\backslash j}$ sample values

across x_j categories. Adjacent category integers, though, could still represent the most semantically different categories, so shifting 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.) In contrast, CatStratPD uses a different algorithm for categoricals and differences the average y for all categories within the leaf to a random reference category, which are then merged across leaves. One could argue that StratPD also extrapolates because $x_{\setminus j}$ could include categorical variables, but the decision tree would, by definition, likely partition $x_{\setminus j}$ space into regions that can be treated similarly.

4 Experimental results

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.

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: $y = x_1^2 + x_1 x_2 + 5x_1 sin(3x_2) + 10$ 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.
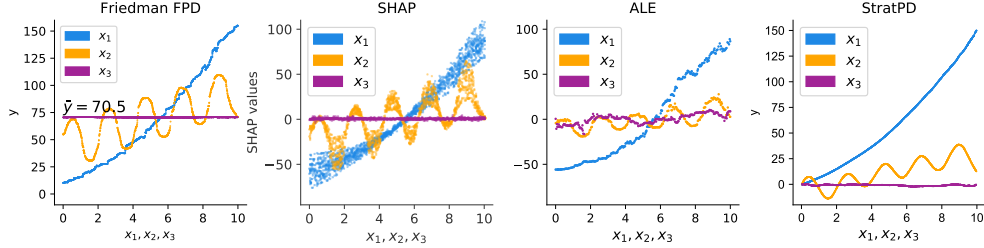

Figure 3: $y = x_1^2 + x_1 x_2 + 5x_1 sin(3x_2) + 10$ where $x_1, x_2, x_3 \sim U(0, 10)$ and x_3 does not affect y. No noise added. Random forest with 10 trees. R^2 0.9973

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 $y = x_1^2 + x_1 + 10 + \epsilon$ 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 accuracy is stable. The superfluous noise variable x_3 in Figure 3 also did not confuse StratPD.
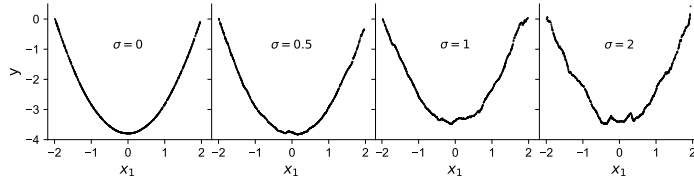

Figure 4: $y = x_1^2 + x_1 + 10 + \epsilon$ where $x_1, x_2 \sim U(-2, 2)$, $\epsilon \sim N(0, \sigma)$ with $\sigma \in [0, 0.5, 1, 2]$.

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.
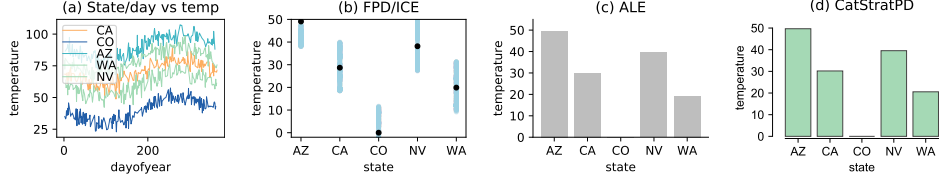

Figure 5: $y = base[x_{state}] + 10 sin(\frac{2\pi}{365} x_{dayofyear} + \pi) + \epsilon$ where $\epsilon \sim N(0, 4)$. The baseline temperature per state is $\{AZ = 90, CA = 70, CO = 40, NV = 80, WA = 60\}$. Sinusoids in (a) are the average of three years' temperature data.

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:

$$
\begin{aligned}
y &= 120 + 10(x_{height} - min(x_{height})) + 40x_{pregnant} - 1.5x_{education} \\
\text{where} \quad & x_{sex} \sim Bernoulli(\{M, F\}, p = 0.5) \\
& x_{pregnant} = \begin{cases} Bernoulli(\{0, 1\}, p = 0.5) & \text{if } x_{sex} = F \\ 0 & \text{if } x_{sex} = M \end{cases} \\
& x_{height} = \begin{cases} 5 * 12 + 5 + \epsilon & \text{if } x_{sex} = F, \ \epsilon \sim U(-4.5, 5) \\ 5 * 12 + 8 + \epsilon & \text{if } x_{sex} = M, \ \epsilon \sim U(-7, 8) \end{cases} \\
& x_{education} = \begin{cases} 12 + \epsilon & \text{if } x_{sex} = F, \ \epsilon \sim U(0, 8) \\ 10 + \epsilon & \text{if } x_{sex} = M, \ \epsilon \sim U(0, 8) \end{cases}
\end{aligned}
\tag{3}
$$

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).
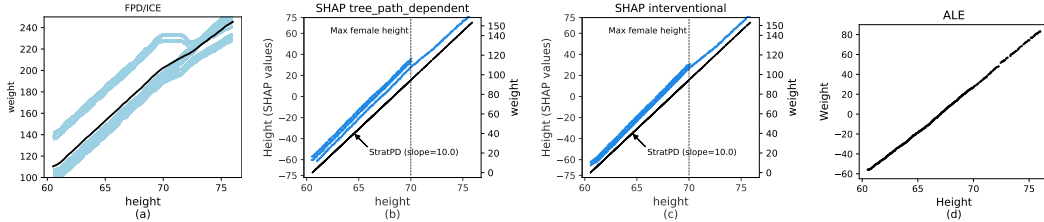

Figure 6: SHAP partial dependence plots of response body weight on feature `height` using 2000 synthetic observations from Equation 3. 50/50 male/female, half of the women are pregnant, and pregnancy contributes 40 pounds. SHAP interrogated an RF with 40 trees and explained 2000 samples; the interventional case used 100 observations as background data. RF OOB R^2 0.999

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
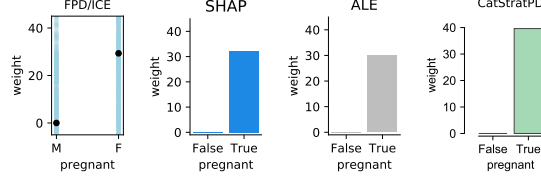
Figure 7: foo.

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. 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.

SHAP also 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.

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 3, 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, such as 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.
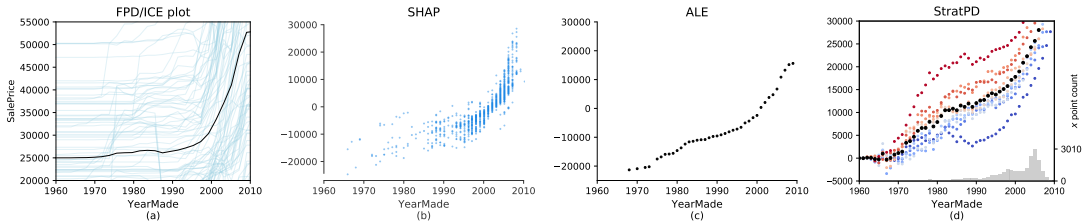

Figure 8: (a) Marginal plot of bulldozer `YearMade` versus `SalePrice` using subsample of 20k observations, (b) partial dependence drawn by SHAP interrogating an RF with 40 trees and explaining 1000 values with 100 observations as background data, (c) STRATPD partial dependence.

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 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.

9

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.

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 9 shows growth curves for 40 numerical variables and 11 categorical variables grouped by type of variable. For these data sets, StratPD takes 1.1s or less to process 30,000 records for any 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 outperforms these existing techniques (as training and tuning is often measured in minutes).
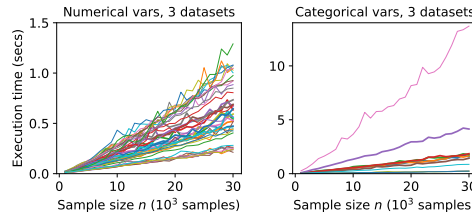

Figure 9: timing blah. $p = 20$, bulldozer $p = 14$, and flight $p = 17$. Numba JIT warmup Numba just-in-time compiler (`numba.pydata.org`)

what if X,y relationship is very weak? models would get low accuracy. what happens to us? I think we would simply show low partial dependence curves.

TODO: unsupervised?

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, and Python 3.7.4.

5   Discussion and future work

we need more data, more susceptible to noise

can parallelize code or do in C++.

similar to ALE but much simpler for categorical variables.

we can handle partial derivatives for $x_{j,l}$ using 2D finite difference to get gradients then using double integral instead of single cumulative sum. I believe that gets us an estimate for plot value at point (x_j, x_l). actually looks like the ALE guy is calling it a second order effect and is subtracting the two one-dimensional finite differences. Hmm... he's just holding x_j, x_l constant and integrating over the other variables to find the value at (x_j, x_l).

A word on hyper parameters. All techniques have hyper parameters that can affect results, particularly if one includes hyper parameters in the model itself for model-based techniques. For example, a SHAP plot like Figure 8(c) with fewer explained vectors looks much more like a line but then does not alert the user about variable interactions. It is sound practice to try a variety of hyper parameters with any partial dependence technique, including ours.

Our goal here is not to say these other techniques are not useful. Rather, we are pointing out potential issues and hoping to open a new line of inquiry that experiments have shown to be accurate in cases where previous techniques are not.

10

6 Appendix

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**
    $(\mathbf{x}_L, \mathbf{y}_L) := \{(x_j^{(i)}, y^{(i)})\}_{i \in L}$ ▷ *Get leaf observations*
    $\mathbf{ux}, \bar{\mathbf{y}} :=$ Group and sort $(\mathbf{x}_L, \mathbf{y}_L)$ by x_j value, computing $\bar{y}$ per unique x_j value
    $\mathbf{dx} := \mathbf{ux}^{(i+1)} - \mathbf{ux}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$ ▷ *Discrete difference between adjacent*
    $\mathbf{dy} := \bar{\mathbf{y}}^{(i+1)} - \bar{\mathbf{y}}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$
    Add tuples $(\mathbf{ux}^{(i)}, \mathbf{ux}^{(i+1)}, \mathbf{dy}^{(i)}/\mathbf{dx}^{(i)})$ to list $\mathbf{D}$ for $i = 1..|\mathbf{ux}| - 1$
$\mathbf{ux} := unique(\mathbf{X}_j)$
for *each* $x \in \mathbf{ux}$ **do**   ▷ *Count slopes and compute average slope per unique x_j value*
    $\mathbf{slopes} := [slope$ for $(a, b, slope) \in \mathbf{D}$ if $x \geq a$ and $x < b]$
    $\mathbf{c}_x := |\mathbf{slopes}|$
    $\mathbf{dydx}_x := \overline{\mathbf{slopes}}$
$\mathbf{dydx} := \mathbf{dydx}[\mathbf{c} \geq min\_slopes\_per\_x]$ ▷ *Drop missing slopes, those computed with too few*
$\mathbf{ux} := \mathbf{ux}[\mathbf{c} \geq min\_slopes\_per\_x]$
$\mathbf{pdx} := \mathbf{ux}^{(i+1)} - \mathbf{ux}^{(i)}$ for $i = 1..|\mathbf{ux}| - 1$
$\mathbf{pdy} := [0] + $ cumulative_sum$(\mathbf{dydx} * \mathbf{pdx})$ ▷ *integrate, inserting 0 for leftmost x_j*
return $\mathbf{pdx}, \mathbf{pdy}$

Algorithm 2: $CatStratPD(\mathbf{X}, \mathbf{y}, j, min\_samples\_leaf)$

$T :=$ Decision tree regressor fit to $(\mathbf{X}_{\setminus j}, \mathbf{y})$ with hyper parameter: $min\_samples\_leaf$
Let ΔY be a matrix whose columns are vectors of leaf category deltas
Let C be a matrix whose columns 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*
    $(\mathbf{x}_L, \mathbf{y}_L) := \{(x_j^{(i)}, y^{(i)})\}_{i \in L}$ ▷ *Get leaf observations*
    $\mathbf{ucats}, C_L := unique(\mathbf{x}_L)$ ▷ *Get unique categories, counts from leaf observations*
    $\bar{\mathbf{y}} :=$ Group leaf records $(\mathbf{x}_L, \mathbf{y}_L)$ by category, computing $\bar{y}$ per unique category
    $refcat :=$ random category from $\mathbf{x}_L$
    $\Delta Y_L = \bar{\mathbf{y}} - \bar{\mathbf{y}}_{refcat}$
$\Delta \mathbf{y}, \mathbf{c} := \Delta Y_1, C_1$ ▷ *$\Delta \mathbf{y}$ is running average vector mapping category to average y delta*
$completed := \{1\}; work := \{2..|leaves|\};$ ▷ *set of leaf indexes*
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**
        if ΔY_L *has category in common with* $\Delta \mathbf{y}$ **then**
            $completed := completed \cup \{L\}$
            $c =$ random category in intersection
            $\Delta \mathbf{y}_L := \Delta Y_L - \Delta Y_{L,c} + \Delta \mathbf{y}_c$ ▷ *Adjust so $\Delta Y_{L,c} = 0$, add corresponding $\Delta \mathbf{y}_c$ value*
            $\Delta \mathbf{y} := (\mathbf{c} \times \Delta \mathbf{y} + C_L \times \Delta \mathbf{y}_L)/(\mathbf{c} + C_L)$ where $z + NaN = z$ ▷ *weighted average*
            $\mathbf{c} := \mathbf{c} + C_L$ ▷ *update running weight*
    $work := work \setminus completed$
return $\Delta \mathbf{y}$

References

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

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

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

Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv preprint arXiv:1612.08468*, 2016.

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

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

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307, 2008. doi: 10.1186/1471-2105-9-307. URL `https://doi.org/10.1186/1471-2105-9-307`.

Leo Breiman and Adele Cutler. Random forests website. `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#unsup`, 2003. Accessed: 2019-05-24.

Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17:319 – 330, 10 2001. doi: 10.1002/asmb.446.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *ArXiv*, abs/1908.08474, 2019.

Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

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

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