A Simple and Effective Model-Based Variable Importance Measure

Brandon M. Greenwell[*]
Ascend Innovations
and
Andrew J. McCarthy
The Perduco Group
and
Bradley C. Boehmke
Ascend Innovations

December 7, 2017

Abstract

In the era of "big data", it is becoming more of a challenge to not only build state-of-the-art predictive models, but also gain an understanding of what's really going on in the data. For example, it is often of interest to know which, if any, of the predictors in a fitted model are relatively influential on the predicted outcome. Some modern algorithms—like random forests and gradient boosted decision trees—have a natural way of quantifying the importance or relative influence of each feature. Other algorithms—like naive Bayes classifiers and support vector machines—are not capable of doing so and model-free approaches are generally used to measure each predictor's importance. In this paper, we propose a standardized, model-based approach to measuring predictor importance across the growing spectrum of supervised learning algorithms. Our proposed method is illustrated through both simulated and real data examples. The R code to reproduce all of the figures in this paper is available in the supplementary materials.

Keywords: Relative influence, Interaction effect, Partial dependence function, Partial dependence plot, PDP.

[*]The authors gratefully acknowledge *please remember to list all relevant funding sources in the unblinded version*

1 Introduction

TODO:

- Discuss PDPs for variables not included in the model. For these variables, the partial dependence function in a constant; hence, the standard deviation of the partial dependence values is exactly zero. For the Ames housing example, the predictors `Street`, `Utilities`, `PoolArea`, and `PoolQC` were not used as splitters in the GBM ensemble; hence, there PDPs would be flat and there variable importance score should be zero.

- Display all variable importance scores for the Ames housing example?

Complex supervised learning algorithms, such as neural networks (NNs) and support vector machines (SVMs), are more common than ever in predictive analytics, especially when dealing with large observational databases that don't adhere to the strict assumptions imposed by traditional statistical techniques (e.g., multiple linear regression which typically assumes linearity, homoscedasticity, and normality). However, it can be challenging to understand the results of such complex models and explain them to management. Graphical displays such as variable importance plots (when available) and partial dependence plots (PDPs) (Friedman, 2001) offer a simple solution (see, for example, (Hastie et al., 2009, pp. 367–380)). PDPs are low-dimensional graphical renderings of the prediction function $\widehat{f}(\boldsymbol{x})$ that allow analysts to more easily understand the estimated relationship between the outcome and predictors of interest. These plots are especially useful in interpreting the output from "black box" models. While PDPs can be constructed for any predictor in a fitted model, variable importance scores are more difficult to define, and when available, their interpretation often depends on the model fitting algorithm used.

In this paper, we consider a standardized method to computing variable importance scores using PDPs. There are a number of advantages to using our approach. First, it offers a standardized procedure to quantifying variable importance across the growing spectrum of supervised learning algorithms. For example, while popular statistical learning algorithms like random forests (RFs) and gradient boosted decision trees (GBMs) have there own natural way of measuring variable importance, each is interpreted differently (these are briefly described in Section 2.1). Secondly, our method is suitable for use with any trained supervised learning algorithm, provided predictions on new data can be obtained. For example, it is often beneficial (from an accuracy standpoint) to train and tune multiple state-of-the art predictive models (e.g., multiple RFs, GBMs, and deep learning NNs (DNNs)) and then combine them into an ensemble called a *super learner* through a process called *model stacking*. Even if the base learners can provide there own measures of variable importance, there is no logical way to combine them to form an overall score for the super learner. However, since new prediction can be obtained from the super learner, our proposed variable importance measure is still applicable (examples are given in Sections 5–6). Thirdly, as shown in Section 3.2, our proposed method can be modified to quantify the strength of potential interaction effects. Finally, since our approach is based on constructing PDPs for all the main effects, the analyst is forced to also look at the estimated functional relationship between each feature and the target—which should be done in tandem with studying the importance of each feature.

2 Background

We are often confronted with the task of extracting knowledge from large databases. For this task we turn to various statistical learning algorithms which, when tuned correctly, can have state-of-the-art predictive performance. However, having a model that predicts well is only solving part of the problem. It is also desirable to extract information about the relationships uncovered by the learning algorithm. For instance, we often want to know which predictors, if any, are important by assigning some type of variable importance score to each feature. Once a set of influential features has been identified, the next step is summarizing the functional relationship between each feature, or subset thereof, and the outcome of interest. However, since most statistical learning algorithms are "black box" models, extracting this information is not always straightforward. Luckily, some learning algorithms have a natural way of defining variable importance.

2.1 Model-based approaches to variable importance

Decision trees probably offer the most natural model-based approach to quantifying the importance of each feature. In a binary decision tree, at each node t, a single predictor is used to partition the data into two homogeneous groups. The chosen predictor is the one that maximizes some measure of improvement $\widehat{i}_t$. The relative importance of predictor x is the sum of the squared improvements over all internal nodes of the tree for which x was chosen as the partitioning variable; see Breiman et al. (1984) for details. This idea also extends to ensembles of decision trees, such as RFs and GBMs. In ensembles, the improvement score for each predictor is averaged across all the trees in the ensemble. Fortunately, due to the stabilizing effect of averaging, the improvement-based variable importance metric is often more reliable in large ensembles (Hastie et al., 2009, pg. 368). RFs offer an additional method for computing variable importance scores. The idea is to use the leftover out-of-bag (OOB) data to construct validation-set errors for each tree. Then, each predictor is randomly shuffled in the OOB data and the error is computed again. The idea is that if variable x is important, then the validation error will go up when x is perturbed in the OOB data. The difference in the two errors is recorded for the OOB data then averaged across all trees in the forest.

In multiple linear regression, the absolute value of the t-statistic is commonly used as a measure of variable importance. The same idea also extends to generalized linear models (GLMs). Multivariate adaptive regression splines (MARS), which were introduced in Friedman (1991a), is an automatic regression technique which can be seen as a generalization of multiple linear regression and generalized linear models. In the MARS algorithm, the contribution (or variable importance score) for each predictor is determined using a generalized cross-validation (GCV) statistic.

For NNs, two popular methods for constructing variable importance scores are the Garson algorithm (Garson, 1991), later modified by Goh (1995), and the Olden algorithm (Olden et al., 2004). For both algorithms, the basis of these importance scores is the network's connection weights. The Garson algorithm determines variable importance by identifying all weighted connections between the nodes of interest. Olden's algorithm, on the other hand, uses the product of the raw connection weights between each input and output neuron and sums the product across all hidden neurons. This has been shown to outperform the Garson method in various simulations. For DNNs, a similar method due to Gedeon (1997) considers the weights connecting the input features to the first two hidden layers (for simplicity and speed); but this method can be slow for large networks.

2.2 Filter-based approaches to variable importance

Filter-based approaches, which are described in Kuhn and Johnson (2013, chap. 18), do not make use of the fitted model to measure variable importance. They also do not take into account the other predictors in the model.

For regression problems, a popular approach to measuring the variable importance of a numeric predictor x is to first fit a flexible nonparametric model between x and the target Y; for example, the locally-weighted polynomial regression (LOWESS) method developed by Cleveland (1979). From this fit, a pseudo-R^2 measure can be obtained from the resulting residuals and used as a measure of variable importance. For categorical predictors, a different method based on standard statistical tests (e.g., t-tests and ANOVAs) is employed; see Kuhn and Johnson (2013, chap. 18) for details.

For classification problems, an area under the ROC curve (AUC) statistic can be used to quantify predictor importance. The AUC statistic is computed by using the predictor x as input to the ROC curve. If x can reasonably separate the classes of Y, that is a clear indicator that x is an important predictor (in terms of class separation) and this is captured in the corresponding AUC statistic. For problems with more than two classes, extensions of the ROC curve or a one-vs-all approach can be used.

2.3 Partial dependence plots

Harrison and Rubinfeld (1978) analyzed a data set containing suburban Boston housing data from the 1970 census. They sought a housing value equation using an assortment of features; see Harrison and Rubinfeld (1978, Table IV) for a description of each variable. The usual regression assumptions, such as normality, linearity, and constant variance, were clearly violated, but through an exhausting series of transformations, significance testing, and grid searches, they were able to build a model which fit the data reasonably well

($R^2 = 0.81$). Their prediction equation is given in Equation (1). This equation makes interpreting the model easier. For example, the average number of rooms per dwelling (RM) is included in the model as a quadratic term with a positive coefficient. This means that there is a monotonic increasing relationship between RM and the predicted median home value, but larger values of RM have a greater impact.

$$\begin{aligned}\log\widehat{(MV)} = {} & 9.76 + 0.0063RM^2 + 8.98\times10^{-5}AGE - 0.19\log(DIS) + 0.096\log(RAD) \\ & - 4.20\times10^{-4}TAX - 0.031PTRATIO + 0.36\,(B-0.63)^2 - 0.37\log(LSTAT) \\ & - 0.012CRIM + 8.03\times10^{-5}ZN + 2.41\times10^{-4}INDUS + 0.088CHAS \\ & - 0.0064NOX^2.\end{aligned} \tag{1}$$

However, classical regression and model building is rather ill-suited for more contemporary big data sets, like the Ames housing data described in Cock (2011) which has a total of 79 predictors (and many more that can be created through feature engineering). Fortunately, using modern computing power, many supervised learning algorithms can fit such data sets in seconds, producing powerful, highly accurate models. The downfall of many of these machine learning algorithms, however, is decreased interpretability. For example, fitting a well-tuned RF to the Boston housing data will likely produce a model with more accurate predictions, but no interpretable prediction formula such as the one in Equation (1).

To help understand the estimated functional relationship between each predictor and the outcome of interest in a fitted model, we can construct PDPs. PDPs are particularly effective at helping to explain the output from "black box" models, such as RFs and SVMs. Not only do PDPs visually convey the relationship between low cardinality subsets of the feature set (usually 1-3) and the response (while accounting for the average effect of the other predictors in the model), they can also be used to rank and score the predictors in terms of their relative influence on the predicted outcome, as will be demonstrated in this paper.

Let $\boldsymbol{x} = \{x_1, x_2, \dots, x_p\}$ represent the predictors in a model whose prediction function is $\widehat{f}(\boldsymbol{x})$. If we partition $\boldsymbol{x}$ into an interest set, $\boldsymbol{z}_s$, and its complement, $\boldsymbol{z}_c = \boldsymbol{x} \setminus \boldsymbol{z}_s$, then the "partial dependence" of the response on $\boldsymbol{z}_s$ is defined as

$$f_s\left(\boldsymbol{z}_s\right) = E_{\boldsymbol{z}_c}\left[\widehat{f}\left(\boldsymbol{z}_s, \boldsymbol{z}_c\right)\right] = \int \widehat{f}\left(\boldsymbol{z}_s, \boldsymbol{z}_c\right) p_c\left(\boldsymbol{z}_c\right) d\boldsymbol{z}_c, \tag{2}$$

where $p_c\left(\boldsymbol{z}_c\right)$ is the marginal probability density of $\boldsymbol{z}_c$: $p_c\left(\boldsymbol{z}_c\right) = \int p\left(\boldsymbol{x}\right) d\boldsymbol{z}_s$. Equation (2) can be estimated from a set of training data by

$$\bar{f}_s\left(\boldsymbol{z}_s\right) = \frac{1}{n}\sum_{i=1}^{n} \widehat{f}\left(\boldsymbol{z}_s, \boldsymbol{z}_{i,c}\right), \tag{3}$$

where $\boldsymbol{z}_{i,c}$ $(i = 1, 2, \dots, n)$ are the values of $\boldsymbol{z}_c$ that occur in the training sample; that is, we average out the effects of all the other predictors in the model.

Constructing a PDP (3) in practice is rather straightforward. To simplify, let $\boldsymbol{z}_s = x_1$ be the predictor variable of interest with unique values $\{x_{11}, x_{12}, \dots, x_{1k}\}$. The partial dependence of the response on x_1 can be constructed as follows:

```
Input: the unique predictor values $x_{11}, x_{12}, \dots, x_{1k}$;
Output: the estimated partial dependence values $\bar{f}_1(x_{11}), \bar{f}_1(x_{12}), \dots, \bar{f}_1(x_{1k})$.
for $i \in \{1, 2, \dots, k\}$ do
    (1) copy the training data and replace the original values of $x_1$ with the
        constant $x_{1i}$;
    (2) compute the vector of predicted values from the modified copy of the
        training data;
    (3) compute the average prediction to obtain $\bar{f}_1(x_{1i})$.
end
```

Algorithm 1: A simple algorithm for constructing the partial dependence of the response on a single predictor x_1.

The PDP for x_1 is obtained by plotting the pairs $\left\{x_{1i}, \bar{f}_1\left(x_{1i}\right)\right\}$ for $i = 1, 2, \dots, k$.

Algorithm 1 can be computationally expensive since it involves k passes over the training records. Fortunately, it is embarrassingly parallel and computing partial dependence functions for each predictor can be done rather quickly on a machine with a multi-core processor. For large data sets, it may be worthwhile to reduce the grid size by using specific quantiles for each predictor, rather than all the unique values. For example, the partial dependence function can be approximated very quickly by using the deciles of the unique predictor values. The exception is classification and regression trees based on single-variable splits which can make use of the efficient weighted tree traversal method described in Friedman (2001).

While PDPs are an invaluable tool in understanding the relationships uncovered by complex nonparametric models, they can be misleading in the presence of substantial interaction effects (Goldstein et al., 2015). To overcome this issue, Goldstein et al. developed the concept of individual conditional expectation (ICE) curves. ICE curves display the estimated relationship between the response and a predictor of interest for each observation; in other words, skipping step 1 (c) in Algorithm 1. Consequently, the PDP for a predictor of interest can be obtained by averaging the corresponding ICE curves across all observations. Although ICE curves provide a refinement over traditional PDPs in the presence of substantial interaction effects, in Section 3.2, we show how to use partial dependence functions to evaluate the strength of potential interaction effects.

2.4 The Ames housing data set

For illustration, we will use the Ames housing data set—a modernized and expanded version of the often cited Boston Housing data set. These data are available from Kaggle: `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data`. Using the R package `h2o` (The H2O.ai team, 2017), we trained and tuned a GBM using 10-fold cross-validation. The model fit is reasonable, with a cross-validated (pseudo) R^2 of 90.06%. Like other tree-based ensembles, GBMs have a natural way of defining variable importance which was described in Section 2.1. The variable importance scores for these data are displayed in the left side of Figure 2. This plot indicates that the overall quality of the material and finish of the house (`OverallQual`), physical location within the Ames city limits (`Neighborhood`), and the above grade (ground) living area (`GrLivArea`) are highly associated with the logarithm of the sales price (`LogSalePrice`). The variable importance scores also indicate that the slope of the property (`LandSlope`), the dollar value of miscellaneous features (`MiscVal`), and the presence of miscellaneous features (`MiscFeature`) have little association with `LogSalePrice`. Note that the bottom four features in Figure 2—`Street`, `Utilities`, `PoolArea`, and `PoolQC`—have a variable importance score of zero (i.e., they were never used to partion the data at any point in the GBM ensemble).

The PDPs for these six variables are displayed in Figure 1. These plots indicate that `OverallQual`, `Neighborhood`, and `GrLivArea` have a strong nonlinear relationship with the predicted outcome. For instance, it seems that `GrLivArea` has a monotonically increasing relationship with `LogSalePrice` until about 12 sq ft, after which the relationship flattens out. Notice how the PDPs for `LandSlope`, `MiscVal`, and `MiscFeature` are relatively flat in comparison. It is this notion of "flatness" which we will use as a basis to define our variable importance measure.

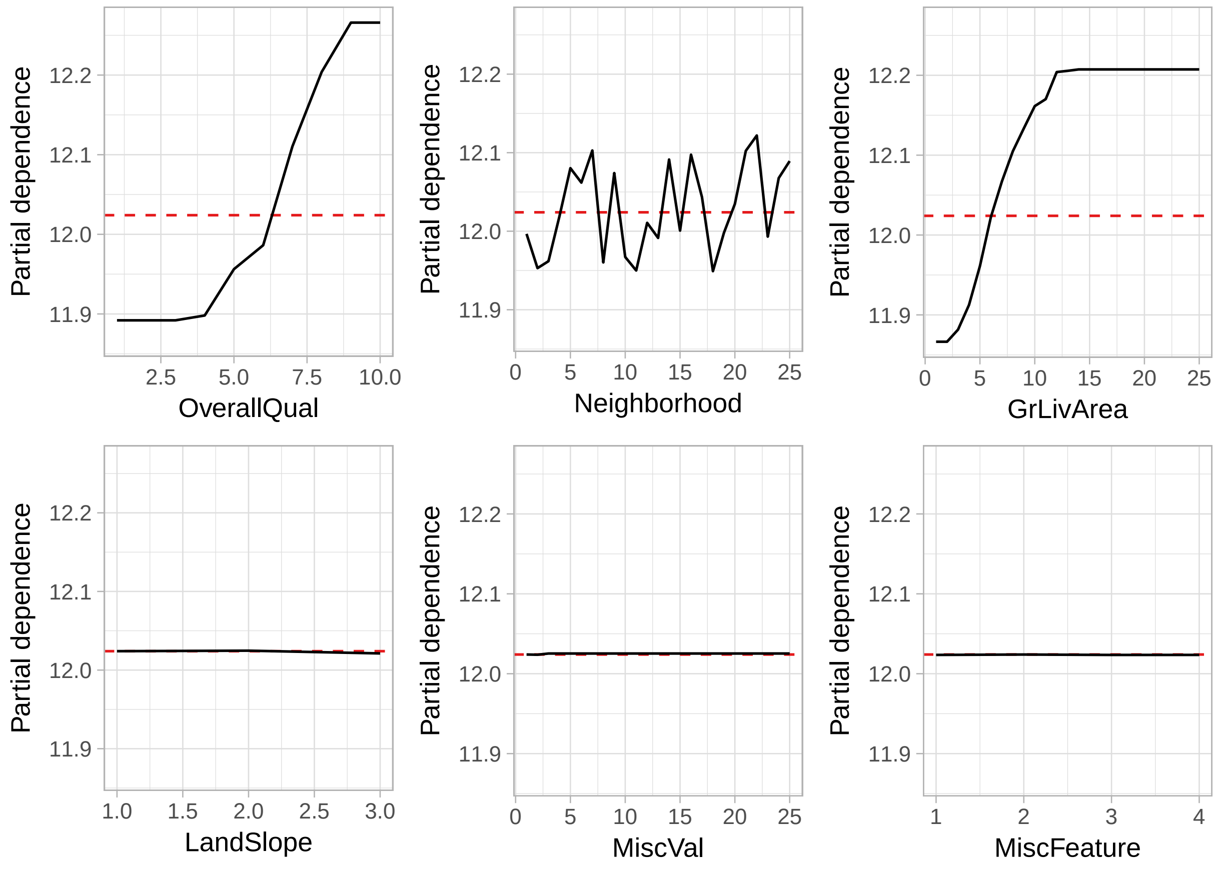

Figure 1: Partial dependence of log selling price on the three highest (top row) and lowest (bottom row) ranked predictors. The dashed red line in each plot represents the mean of the $n = 1460$ log selling prices.

3 A partial dependence-based variable importance measure

The PDPs for `LandSlope`, `MiscVal`, and `MiscFeature` in Figure 1 are relatively flat (i.e., near-zero slope), indicating that they do not have much influence on the predicted value of `LogSalePrice`. In other words, the partial dependence values $\bar{f}_i\left(x_{ij}\right)$ $(j = 1, 2, \dots, k_i)$ display little variability. One might conclude that any variable for which the PDP is "flat" is likely to be less important than those predictors whose PDP varies across a wider range of the response.

Our notion of variable importance is based on any measure of the "flatness" of the partial dependence function. In general, we define

$$\text{imp}\left(x\right) = \text{flatness}\left(\bar{f}_s\left(\boldsymbol{z}_s\right)\right),$$

where flatness is any measure of "flatness" of the curve. A simple and effective measure to use is the sample standard deviation for continuous predictors and the range statistic (divided by four[1]) for factors with K levels. Based on Algorithm 1, our importance measure for predictor x_1 is simply

$$\text{imp}\left(x_1\right) = \begin{cases} \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}\left[\bar{f}_1\left(x_{1i}\right) - \frac{1}{k}\sum_{i=1}^{k}\bar{f}_1\left(x_{1i}\right)\right]^2} & \text{if } x_1 \text{ is continuous} \\ \left[\max_i\left(\bar{f}_1\left(x_{1i}\right)\right) - \min_i\left(\bar{f}_1\left(x_{1i}\right)\right)\right]/4 & \text{if } x_1 \text{ is categorical} \end{cases}. \tag{4}$$

Note that our variable importance metric relies on the fitted model; hence, it is crucial to properly tune and train the model to attain the best performance possible.

To illustrate, we applied Algorithm 1 to all of the predictors in the Ames GBM model and computed (4). The results are displayed in right side of Figure 2. In this case, our partial dependence-based algorithm matches closely with the results from the GBM. In particular, Figure **??** indicates that `OverallQual`, `Neighborhood`, and `GrLivArea` are still the most important variables in predicting `LogSalePrice`; though, `Neighborhood` and `GrLivArea` have swapped places. The overall condition of the house (`OverallCond`) also appears to be relatively important according to Equation (4).

[1]The range divided by four provides an estimate of the standard deviation for small to moderate sample sizes.

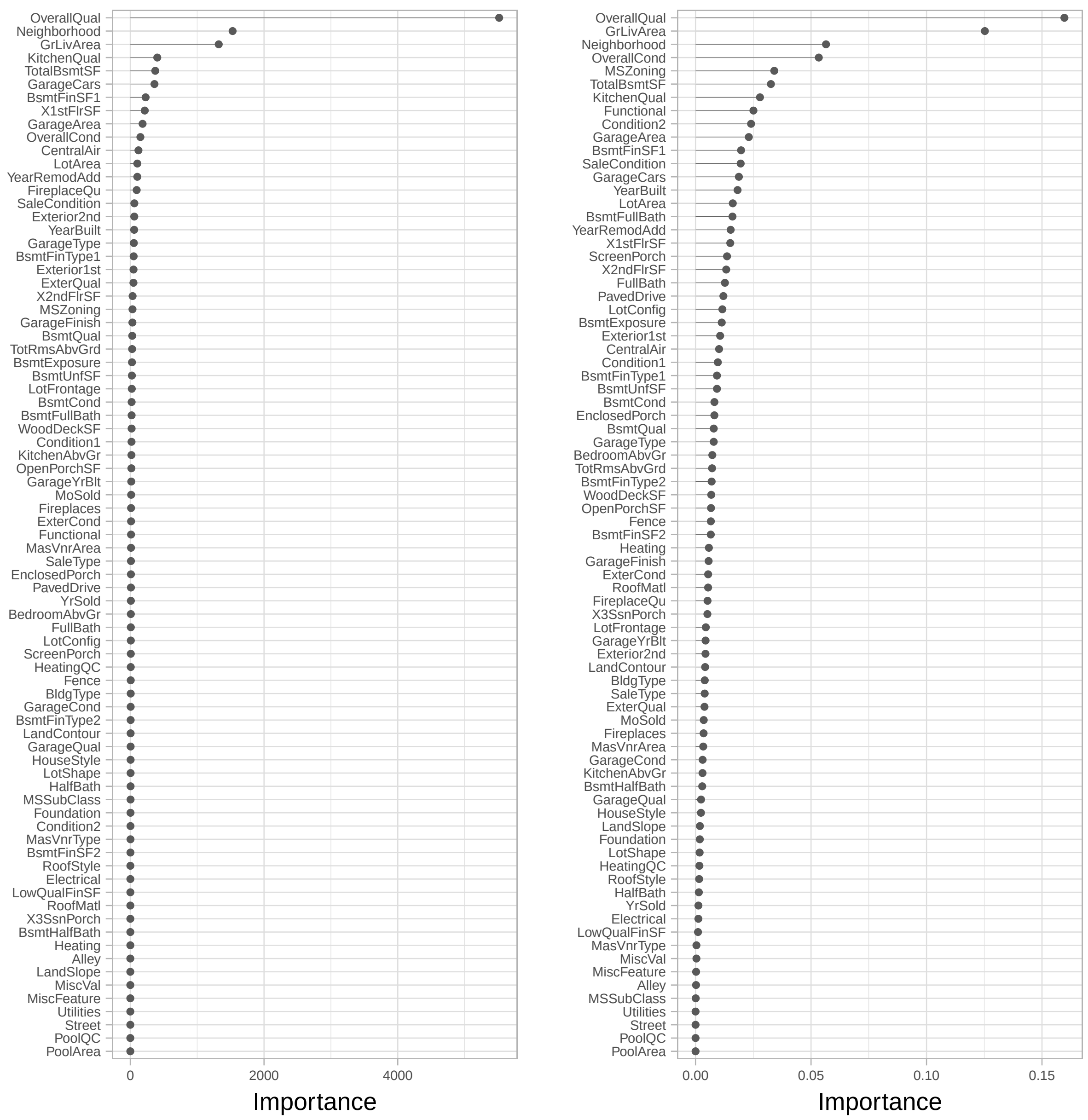

Figure 2: Variable importance scores from a GBM fit to the Ames housing data. *Left*: Standard GBM method. *Right*: partial dependence-based method.

3.1 Linear models

As mentioned earlier, a natural choice for measuring the importance of each term in a linear model is to use the absolute value of the corresponding coefficient divided by its estimated standard error (i.e., the absolute value of the t-statistic). This turns out to be equivalent to the partial dependence-based metric (4) when the predictors are independently and uniformly distributed over the same range.

For example, suppose we have a linear model of the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where β_i $(i = 1, 2)$ is a constant, X_1 and X_2 are both independent $\mathcal{U}(0, 1)$ random variables, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Since we know the distribution of X_1 and X_2, we can easily find $f_1(X_1)$ and $f_2(X_2)$. For instance,

$$f_1(X_1) = \int_0^1 E[Y|X_1, X_2] \, p(X_2) \, dX_2,$$

where $p(X_2) = 1$. Simple calculus then leads to

$$f_1(X_1) = \beta_0 + \beta_2/2 + \beta_1 X_1 \quad and \quad f_2(X_2) = \beta_0 + \beta_1/2 + \beta_2 X_2..$$

Because $E[Y|X_1, X_2] = f(X_1, X_2)$ is additive, the true partial dependence functions are just simple linear regressions in each predictor with their original coefficient and an adjusted intercept. Taking the variance of each gives

$$Var[f_1(X_1)] = \beta_1^2/12 \quad and \quad Var[f_2(X_2)] = \beta_2^2/12.$$

Hence, the standard deviations are just the absolute values of the original coefficients (scaled by the same constant).

To illustrate, we simulated $n = 1000$ observations from the following linear model

$$Y = 1 + 3X_1 - 5X_2 + \epsilon,$$

where X_1 and X_2 are both independent $\mathcal{U}(0, 1)$ random variables, and $\epsilon \sim \mathcal{N}(0, 0.01^2)$. For this example, we have

$$f_1(X_1) = -\frac{3}{2} + 3X_1 \quad and \quad f_2(X_1) = \frac{5}{2} - 5X_2.$$

These are plotted as red lines in Figure 3. Additionally, the black lines in Figure 3 correspond to the estimated partial dependence functions using Algorithm 1.

Based on these plots, X_2 is more influential than X_1. Taking the absolute value of the ratio of the slopes in $f_2(X_2)$ and $f_1(X_1)$ gives $5/3 \approx 1.67$. In other words, X_2 is

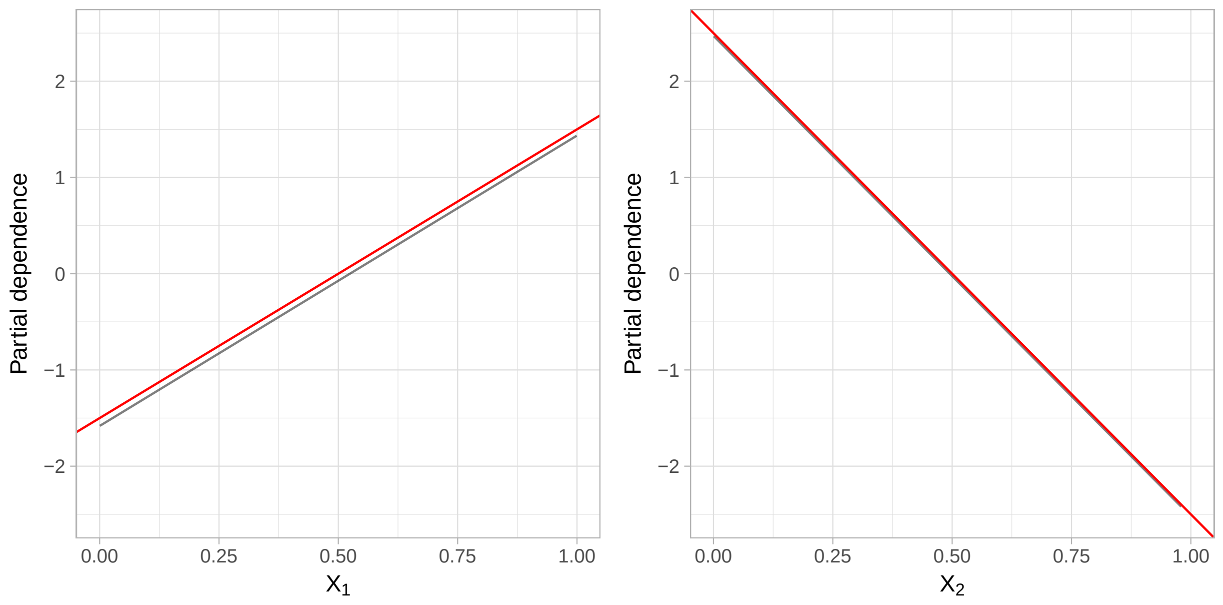

Figure 3: Estimated and true partial dependence plots.

roughly 1.67 times more influential on $\widehat{Y}$ than X_1. Using the partial-dependence-based variable importance metric, we obtain $\text{imp}\left(X_1\right) = 1.4828203$ and $\text{imp}\left(X_2\right) = 0.8961719$ which gives the ratio $\text{imp}\left(X_2\right) / \text{imp}\left(X_1\right) \approx 1.65$. In fact, we can compute the ratio of the true variances of $f_1\left(X_1\right)$ and $f_2\left(X_1\right)$:

$$Var\left[f_2\left(X_2\right)\right] / Var\left[f_1\left(X_1\right)\right] = \left(5^2/12\right) / \left(3^2/12\right) = \left(5/3\right)^2 .$$

Taking the square root gives $5/3 \approx 1.67$.

Using the absolute value of the t-statistic becomes less useful in linear models when, for example, a predictor appears in multiple terms (e.g., interaction effects and polynomial terms). The partial dependence approach, on the other hand, does not suffer from such drawbacks.

3.2 Detecting interaction effects

As it turns out, our partial dependence-based variable importance measure (4) can also be used to quantify the strength of potential interaction effects. Let $\text{imp}\left(x_i, x_j\right)$ $(i \neq j)$ be the standard deviation of the joint partial dependence values $\bar{f}_{ij}\left(x_{ii'}, x_{jj'}\right)$ for $i' = 1, 2, \dots, k_i$ and $j' = 1, 2, \dots, k_j$. Essentially, a weak interaction effect of x_i and x_j on Y would suggest that $\text{imp}\left(x_i, x_j\right)$ has little variation when either x_i or x_j is held constant while the other varies.

Let $\boldsymbol{z}_s = \left(x_i, x_j\right)$, $i \neq j$, be any two predictors in the feature space $\boldsymbol{x}$. Construct the partial dependence function $\bar{f}_s\left(x_i, x_j\right)$ and compute $\text{imp}\left(x_i\right)$ for each unique value of x_j, denoted $\text{imp}\left(x_i | x_j\right)$, and take the standard deviation of the resulting importance scores. The same can be done for x_j and the results are averaged together. Large values (relative to each other) would be indicative of possible interaction effects.

4 Friedman's regression problem

To further illustrate, we will use one of the regression problems described in Friedman (1991b) and Breiman (1996). The feature space consists of ten independent $\mathcal{U}\left(0, 1\right)$ random variables; however, only five out of these ten actually appear in the true model. The response is related to the features according to the formula

$$Y = 10 \sin\left(\pi x_1 x_2\right) + 20\left(x_3 - 0.5\right)^2 + 10 x_4 + 5 x_5 + \epsilon,$$

where $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$. Using the R package `nnet` (Venables and Ripley, 2002), we fit a NN with one hidden layer containing eight units and a weight decay of 0.01 (these parameters were chosen using 5-fold cross-validation) to 500 observations simulated from the above model with $\sigma = 1$. The cross-validated R^2 value was 0.94.

Variable importance plots are displayed in Figure 4. Notice how the Garson and Olden algorithms incorrectly label some of the features not in the true model as "important". For example, the Garson algorithm incorrectly labels x_8 (which is not included in the true model) as more important than x_5 (which is in the true model). Similarly, Oden's method incorrectly labels x_{10} as being more important than x_2. Our method, on the other hand, clearly labels all five of the predictors in the true model as the most important features in the fitted NN.

We also constructed the partial dependence functions for all pairwise interactions and computed the interaction statistic discussed in Section 3.2. The top ten interaction statistics are displayed in Figure 5. There is a clear indication of an interaction effect between features x_1 and x_2, the only interaction effect present in the true model.

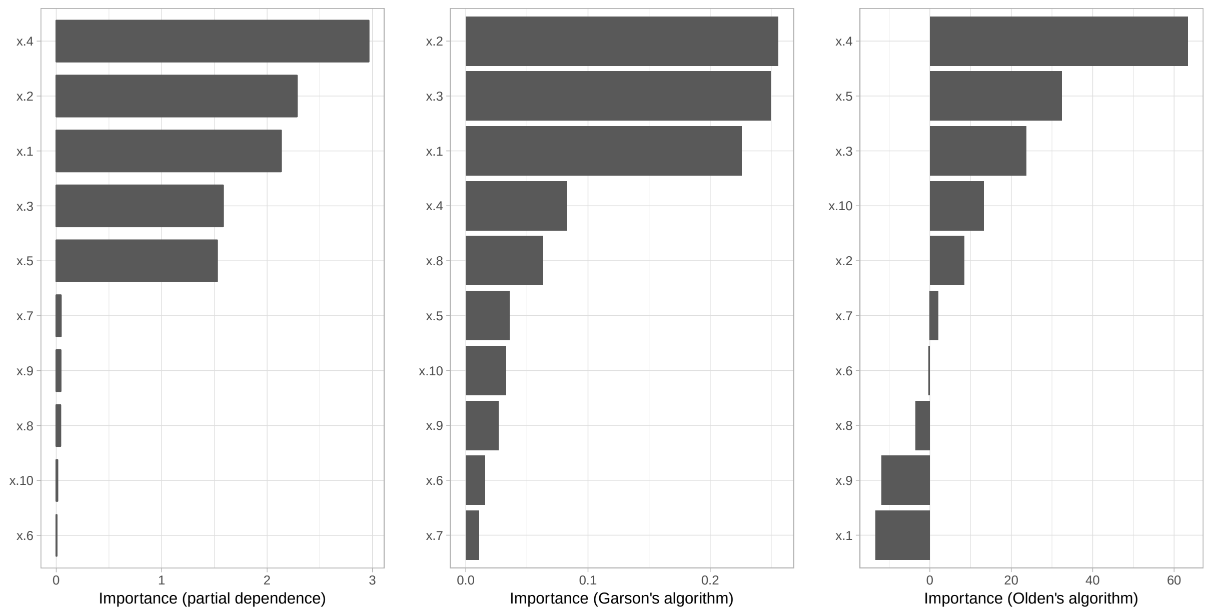

Figure 4: Variable importance plots for the NN fit to the Friedman regression data. *Left*: partial dependence-based method. *Middle*: Garson's method. *Right*: Olden's method.

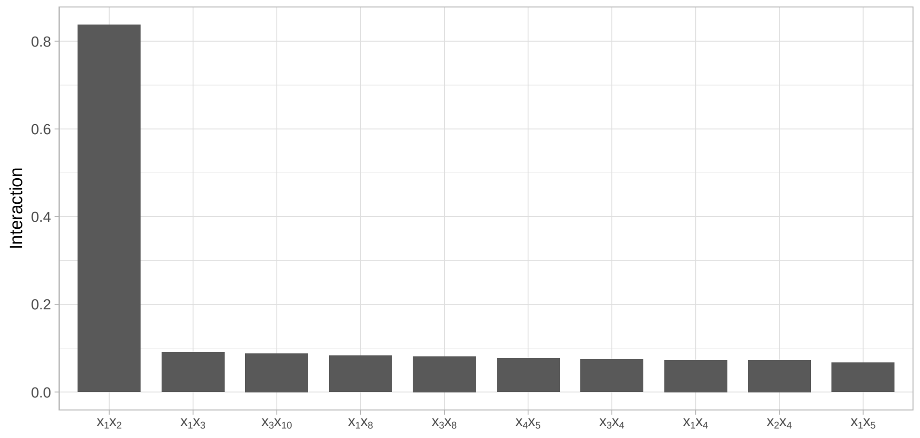

Figure 5: Variable importance-based interaction statistics from the NN fit to the Friedman regression data set.

In fact, since we know the distributions of the predictors in the true model, we can work out the true partial dependence functions. For example, for the pairs (x_1, x_2) and (x_1, x_4), we have

$$f\left(x_1, x_2\right) = 10 \sin\left(\pi x_1 x_2\right) + 55/6,$$

and

$$f(x_1, x_4) = \frac{5\pi x_1(12x_4 + 5) - 12\cos(\pi x_1) + 12}{6\pi x_1}.$$

Next, we simulated the standard deviation of $f(x_1, x_2)$ for a wide range of fixed values of x_2; this is what $\text{imp}(x_1|x_2)$ is trying to estimate. The results from doing this for both predictors in each model are displayed in Figure 6. The top row of Figure 6 illustrates that the importance of x_1 (i.e., the strength of its relationship to the predicted outcome) heavily depends on the value of x_2 and vice versa (i.e., an interaction effect between x_1 and x_2). In the bottom row, on the other hand, we see that the importance of x_1 does not depend on the value of x_4 and vice versa (i.e., no interaction effect between x_1 and x_4).

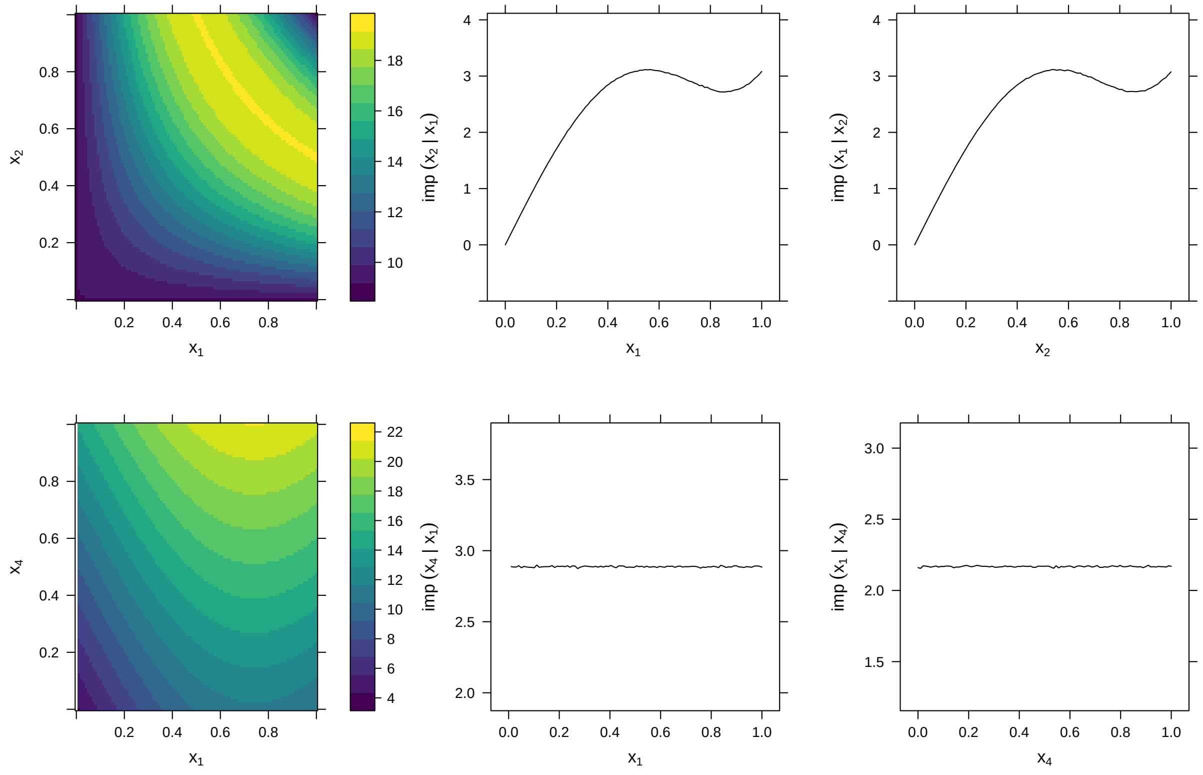

Figure 6: Results from a small Monte Carlo simulation on the interaction effects between x_1 and x_2 (top row), and x_1 and x_4 (bottom row).

4.1 Friedman's H-statistic

An alternative measure for the strength of interaction effects is known as Friedman's H-statistic (Friedman and Popescu, 2008). Coincidentally, this method is also based on the estimated partial dependence functions of the corresponding predictors, but uses a different approach.

For comparison, we fit a GBM to the Friedman regression data from the previous section. The parameters were chosen using 5-fold cross-validation. We used the R package `gbm` (Ridgeway, 2017) which has built-in support for computing Friedman's H-statistic for any combination of predictors. The results are displayed in Figure 7. To our surprise, the H-statistic did not seem to catch the true interaction between x_1 and x_2. Instead, the H-statistic ranked the pairs (x_8, x_9) and (x_7, x_{10}) as having the strongest interaction effects, even though these predictors do not appear in the true model. Our variable importance-based interaction statistic, on the other hand, clearly suggests the pair (x_1, x_2) as having the strongest interaction effect.

5 Application to model stacking

In the Ames housing example, we used a GBM to illustrate the idea of using the partial dependence function to quantify the importance of each predictor on the predicted outcome (in this case, `LogSalePrice`). While the GBM model achieved a decent cross-validated R^2 of 90.06%, better predictive performance can often be attained by creating an *ensemble* of learning algorithms. While GBMs are themselves ensembles, we can use a method called *stacking* to combine the GBM with other cutting edge learning algorithms to form a *super learner* (Wolpert, 1992). Such stacked ensembles tend to outperform any of the individual base learners (e.g., a single RF or GBM) and have been shown to represent an asymptotically optimal system for learning (van der Laan et al., 2003).

For the Ames data set, in addition to the GBM, we trained and tuned a RF using 10-fold cross-validation. The cross-validated predicted values from each of these models were combined to form a $n \times 2$ matrix, where $n = 1460$ is the number of training records. This matrix, together with the observed values of `LogSalePrice`, formed the "level-one" data.

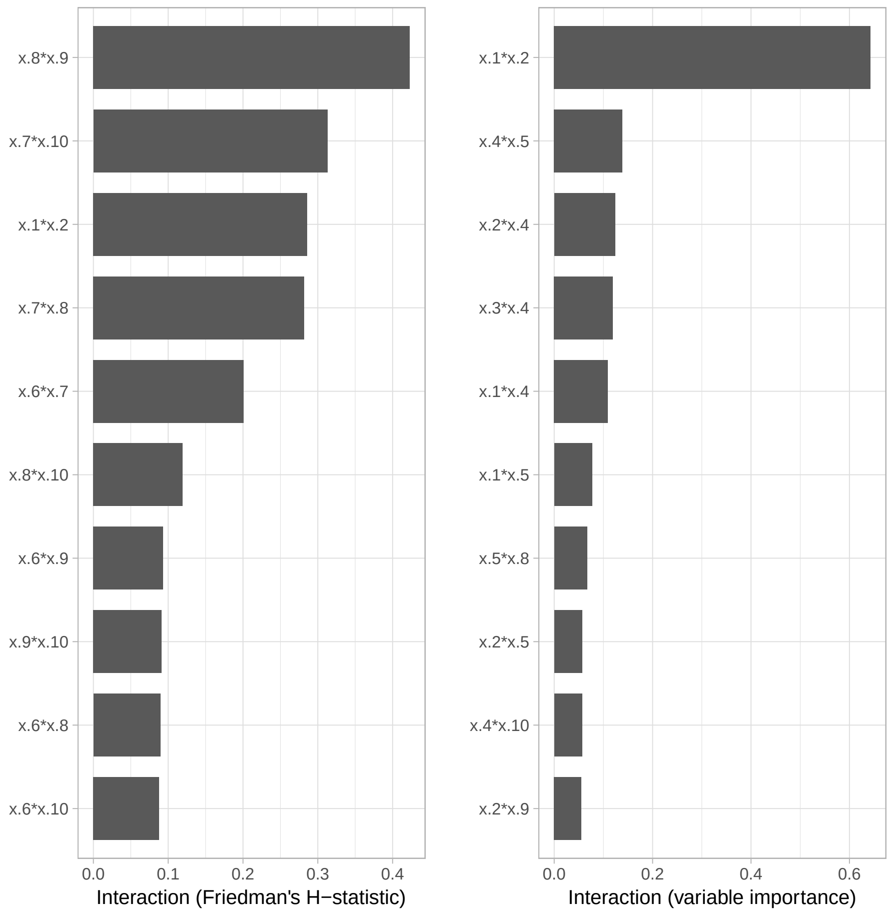

Figure 7: Interaction statistics for the GBM model fit to the Friedman regression data. *Left*: Friedman's H-statistic. *Right*: Our variable importance-based interaction statistic.

Next, we trained a *metalearning* algorithm—in this case a GLM—on the level-one data to create a stacked ensemble. To generate new predictions, we first generate predictions from the RF and GBM learners, then feed those into the GLM metalearner to generate the ensemble prediction.

Even though the base learners in this example have the built-in capability to compute variable importance scores, there is no way of constructing them from the super learner. However, since we can generate predictions from the super learner, we can easily construct partial dependence functions for each predictor and use Equation (4). The left and middle plots in Figure 8 display the top 15 variable importance scores from the individual base learners. The right plot displays the top 15 variable importance scores constructed using

Equation (4) for the combined super learner. All models seem to agree on the top three predictors of `LogSalePrice`; namely, `OverallQual`, `Neighborhood`, and `GrLivArea`.

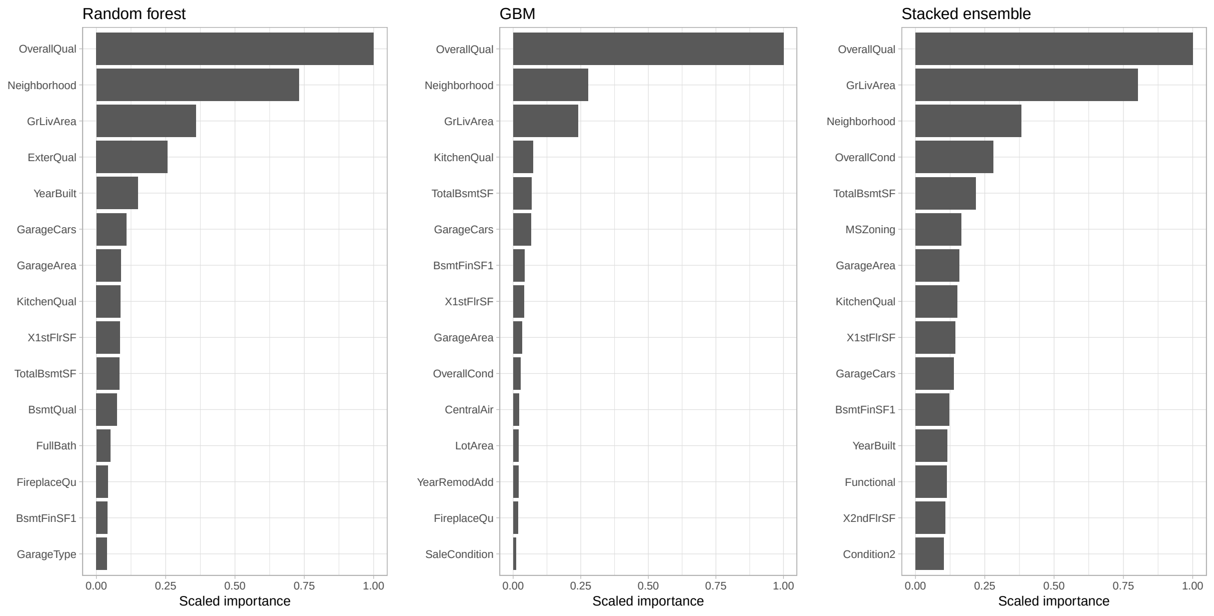

Figure 8: Variable importance plots for the Ames data set. *Left*: RF. *Middle*: GBM. *Right*: Stacked ensemble (i.e., super learner).

6 Application to automatic machine learning

The data science field has seen in explosion in interest over the last decade. However, the supply of data scientists and machine learning experts has not caught up with the demand. Consequently, many organizations are turning to automated machine learning (AutoML) approaches to predictive modelling. AutoML has been a topic of increasing interest over the last couple of years and open source implementations—like those in H2O and auto-sklearn (Feurer et al., 2015)—have made it a simple and viable options for real supervised learning problems.

Current AutoML algorithms leverage recent advantages in Bayesian optimization, meta-learning, and ensemble construction (i.e., model stacking). The benefit, as compared to the stacked ensemble formed in Section 5, is that AutoML frees the analyst from having to do algorithm selection (e.g., "Do I fit an RF or GBM to my data?") and hyperparameter tuning (e.g., how many hidden layers to use in a DNN). While AutoML has great practical potential, it does not automatically provide any useful interpretations—AutoML is **not** automated data science (Mayo, 2017). For instance, which variables are the most important in making accurate predictions? How do these variables functionally related to the outcome of interest? Fortunately, our approach can still be used to answer these two questions simultaneously.

To illustrate, we used the R implementation of H2O's AutoML algorithm to model the airfoil self-noise data set (Lichman, 2013) which are available from the University of California Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise`. These data are from a NASA experiment investigating different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. The objective is to accurately predict the scaled sound pressure level (dB) using frequency (Hz), angle of attack (degrees), chord length (m), free-stream velocity (m/s), and suction side displacement thickness (m). H2O's current AutoML implementation trains and cross-validates an RF, an extremely-randomized forest (XRF) (Geurts et al., 2006), a random grid of GBMs, a random grid of DNNs, and then trains a stacked ensemble using the approach outlined in Section 5. We used the default setting of 5-fold cross-validation and RMSE for the validation metric. Due to hardware constraints, we used a max run time of 15 minutes (the default is one hour). The final model consisted of one RF, one XRT, a random grid of 376 GBMs, and a random grid of four DNNs. The final stacked ensemble achieved a five-fold cross-validated RMSE and R-squared of 1.7131835 and 0.93870246, respectively.

Since predictions can be obtained from the automated stacked ensemble, we can easily apply Algorithm 1 and construct PDPs for all the features; these are displayed in Figure 9. It seems frequency (Hz) and suction side displacement thickness (m) have a strong monotonically decreasing relationship with scaled sound pressure level (dB). They also appear to be more influential than the other three. Using Equation (4), we computed the variable importance of each of the five predictors which are displayed in Table 1.

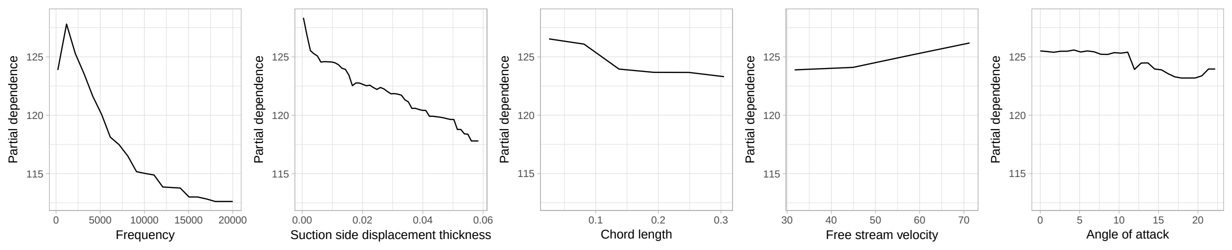

Figure 9: PDPs for the five predictors in the airfoil self-noise data set based on an AutoML algorithm consisting of RFs, XRFs, GBMs, and DNNs.

Variable	Importance
Angle of attack (degrees)	0.9185269
Free stream velocity (m/s)	1.0616875
Chord length (m)	1.3995290
Suction side displacement thickness (m)	2.4407981
Frequency (Hz)	4.7842556

Table 1: Variable importance scores for the five predictors in the airfoil self-noise data set based on an AutoML algorithm consisting of RFs, XRFs, GBMs, and DNNs.

7 Discussion

We have discussed a new variable importance measure that is (i) suitable for use with any supervised learning algorithm, provided new predictions can be obtained, (ii) model-based and takes into account the effect of all the features in the model, (iii) consistent and has the same interpretation regardless of the learning algorithm employed, and (iv) has the potential to help identify possible interaction effects. Since our algorithm is model-based it requires that the model be properly trained and tuned to achieve optimum performance. While this new approach appears to have high utility, more research is needed to determine where its deficiencies may lie. For example, outliers in the feature space can cause abnormally large fluctuations in the partial dependence values $\bar{f}(x_i)$ $(i = 1, 2, \dots, k)$. Therefore, it may be advantageous to use more robust measures of spread to describe the variability in the estimated partial dependence values; a reasonable choice would be the median absolute deviation which has a finite sample breakdown point of $\lfloor k/2 \rfloor / k$. It is also possible to replace the mean in step (3) of Algorithm 1 with a more robust estimate such as the median or trimmed mean. Another drawback is the computational burden imposed by Algorithm 1 on large data sets, but this can be mitigated using the methods discussed in Greenwell (2017).

All the examples in this article were produced using R version 3.4.0 (R Core Team, 2017); a software environment for statistical computing. With the exception of Figure 6, all graphics were produced using the R package `ggplot2` (Wickham, 2009); Figure 6 was produced using the R package `lattice` (Sarkar, 2008).

SUPPLEMENTARY MATERIAL

R package: The R package `vip`, hosted on GitHub at `https://github.com/AFIT-R/vip`, contains functions for computing variable importance scores and constructing variable importance plots for various types of fitted models in R using the partial dependence-based approach discussed in this paper.

R code: The R script 'greenwell-vi-2017.R' contains the R code to reproduce all of the results and figures in this paper.

References

Breiman, L. (1996). Bagging predictors. *Machine Learning 8*(2), 209–218.

Breiman, L., J. Friedman, and R. A. O. Charles J. Stone (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association 74*(368), 829–836.

Cock, D. D. (2011). Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education 19*(3), 1–15.

Feurer, M., A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter (2015). Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 2962–2970. Curran Associates, Inc.

Friedman, J. H. (1991a). Multivariate adaptive regression splines. *The Annals of Statistics 19*(1), 1–67.

Friedman, J. H. (1991b). Multivariate adaptive regression splines. *The Annals of Statistics 19*(1), 1–67.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics 29*, 1189–1232.

Friedman, J. H. and B. E. Popescu (2008). Predictive learning via rule ensembles. *Annals of Applied Statistics 2*(3), 916–954.

Garson, D. G. (1991). Interpreting neural-network connection weights. *Artificial Intelligence Expert 6*(4), 46–51.

Gedeon, T. (1997). Data mining of inputs: Analysing magnitude and functional measures. *International Journal of Neural Systems 24*(2), 123–140.

Geurts, P., D. Ernst, and L. Wehenkel (2006). Extremely randomized trees. *Machine Learning 63*(1), 3–42.

Goh, A. (1995). Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering 9*(3), 143–151.

Goldstein, A., A. Kapelner, J. Bleich, and E. Pitkin (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.

Greenwell, B. M. (2017). pdp: An r package for constructing partial dependence plots. *The R Journal 9*(1), 421–436.

Harrison, D. and D. L. Rubinfeld (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management 5*(1), 81–102.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer Series in Statistics. Springer-Verlag.

Kuhn, M. and K. Johnson (2013). *Applied Predictive Modeling.* SpringerLink : Bücher. Springer New York.

Lichman, M. (2013). UCI machine learning repository.

Mayo, M. (2017). The current state of automated machine learning.

Olden, J. D., M. K. Joy, and R. G. Death (2004). An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling 178*(3), 389–397.

R Core Team (2017). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing.

Ridgeway, G. (2017). *gbm: Generalized Boosted Regression Models.* R package version 2.1.3.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R.* New York: Springer. ISBN 978-0-387-75968-5.

The H2O.ai team (2017). *h2o: R Interface for H2O.* R package version 3.14.0.3.

van der Laan, M. J., E. C. Polley, and A. E. Hubbard (2003). Super learner. *Statistical Applications in Genetics and Molecular Biology 6*(1).

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S* (4th ed.). New York: Springer-Verlag.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks 5*, 241–259.