SN Computer Science (2021) 2:141
https://doi.org/10.1007/s42979-021-00560-5

ORIGINAL RESEARCH

totalvis: A Principal Components Approach to Visualizing Total Effects in Black Box Models

Nicholas Seedorff[1] · **Grant Brown**[1]

Received: 24 September 2020 / Accepted: 3 March 2021 / Published online: 14 March 2021

Abstract

While a wide variety of machine-learning techniques have been productively applied to diverse prediction tasks, characterizing the nature of patterns and rules learned by these techniques has remained a difficult problem. Often called 'black-box' models for this reason, visualization has become a prominent area of research in understanding their behavior. One powerful tool for summarizing complex models, partial dependence plots (PDPs), offers a low-dimensional graphical interpretation by evaluating the effect of modifying individual predictors on fitted/predicted values. Nevertheless, in high-dimensional settings, PDPs may not capture more complex associations between groups of related variables and the outcome of interest. We propose an extension of PDPs based on the idea of grouping covariates, and interpreting the total effects of the groups. The method utilizes principal components analysis to explore the structure of the covariates, and offers several plots for assessing the approximation function. In conjunction with our diagnostic plot, totalvis gives insight into the total effect a group of covariates has on the prediction and can be used in situations where PDPs may not be appropriate. These tools provide a useful approach for pattern exploration, as well as a natural mechanism to reason about potential causal effects embedded in black-box models.

Keywords Data visualization · Model interpretation · Feature exploration · High dimensional · Principal components analysis

Introduction

First introduced by Friedman [10], partial dependence plots (PDPs) are a model agnostic tool that can help summarize the relationships in black-box models. Due to their straightforward implementation and concise view of the association between the response and covariates of interest, PDPs have become an invaluable tool for model interpretation. The procedure is further described in Greenwell [14] and can be implemented in R [27] with the associated package pdp [15]. Despite being applicable in a variety of settings, PDPs are limited to low dimensions and perform best when the dependence between variables is not too strong [10].

✉ Nicholas Seedorff
nicholas-seedorff@uiowa.edu

Grant Brown
grant-brown@uiowa.edu

[1] Department of Biostatistics, University of Iowa College of Public Health, 145 N Riverside Dr, Iowa City, IA 52242, USA

To formally define PDPs, we adopt the notation used in Hastie et al. [17]. First, define $X^T = \{X_1, \ldots, X_p\}$ as a vector of predictor variables in the model. In addition, let $X_s \subset X$ and denote X_c as the compliment of X_s, i.e. $X_s \cup X_c = X$ and $X_s \cap X_c = \emptyset$. The partial dependence of $f(X)$, the approximation function, on X_s is then defined as

$$f_s(X_s) = E_{X_c}\left[f(X_s, X_c)\right]. \tag{1}$$

This can be estimated by averaging out the other predictors in the model as follows:

$$\bar{f}_s(X_s) = \frac{1}{n}\sum_{i=1}^{n} f(X_s, x_{ic}). \tag{2}$$

Using the above estimator, we can assess the average marginal effect of X_s on the predictions. Although this approach offers simplicity and generality, PDPs have several important limitations. First, when constructing PDPs there is an assumption of low dependence between the covariates of interest and the compliment set. This assumption is often violated, and the procedure creates observations that are

outside of the joint distribution of the training data. In essence, PDPs visualize an isolated variable(s) across its range while holding all others constant, but this may not reflect associations in real datasets, especially with correlated predictors. In addition, PDPs are restricted to low-dimensional views and may not identify complex relationships between multiple covariates and the outcome.

Another well-known agnostic visualization tool, individual conditional expectation (ICE) curves [12], offers an alternative to PDPs when the effect of a covariate is heterogeneous between observations. Similar to PDPs, they plot predictions across the range of a covariate while holding the others constant. However, instead of averaging all predictions ICE plots generate a separate line for each individual. In this way the plots allow for interactions between the feature of interest and the other input variables in the model. A common attribute for these figures is to force all the lines to begin from the same starting place to highlight differences in trajectory. Similar to PDPs, the procedure to create ICE curves gives only unidimensional views into association patterns, and may give unrealistic trajectories in the presence of predictors which co-vary in real samples.

Given the straightforward implementation of the PDP estimator, our approach looks to expand its applicability to address the limitations described above. This is done by providing a related framework where PDPs are constructed for the principal components of a matrix transformed by principal components analysis (PCA). Instead of looking at the main effect of a single covariate, we group similar covariates and offer a more direct look at their total effect. Through the loadings associated with the PCA, our approach displays interpretable insight into the structure of these groups. In addition to the total effect interpretation, this approach mitigates some of the issues that arise when dealing with highly correlated input variables. Conjointly with the PDP extension, we provide a "partial_effects" plot, which can be used as a diagnostic tool. To present this framework we construct the aforementioned plots in R [27] and deliver them through the package totalvis (available at https://github.com/nickseedorff/totalvis).

The outline of the paper is as follows. We first provide a summary of related model agnostic interpretation methods and lay out the package components of totalvis. Following this, we display several simulated examples to motivate both the total effect interpretation and the usefulness of the diagnostic tool. Lastly, we present an application using a dataset from the UCI Machine Learning Repository [9].

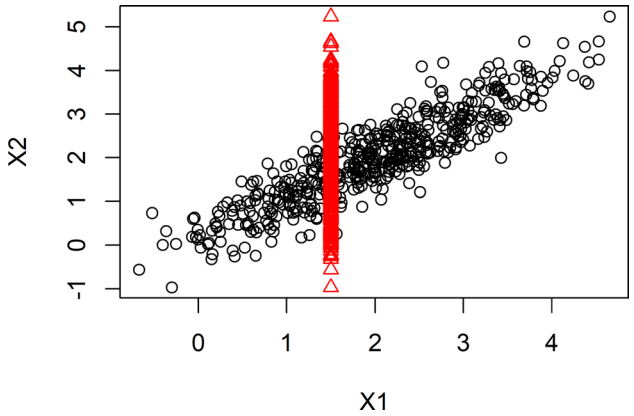


Fig. 1 Extrapolation of the input space may occur while generating PDPs when features are correlated. Calculating the average prediction for a given value of X1 uses points (red triangles) which fall well outside the joint distribution of the training data

Related Work

PDPs and ICE Curves

PDPs serve as the starting point to our approach. These were outlined in the introduction and are referenced throughout the paper. ICE curves were also summarized and further details can be found in [12], who produced an associated package ICEbox [13]. For the sake of completeness, we include the ability to plot simple ice curves in totalvis, but direct the interested reader to ICEbox for increased functionality. In the rest of this section, we summarize other prominent model agnostic interpretation methods.

M Plots

Marginal plots, known as M Plots, offer an approach to avoiding extrapolation in the feature space. When dealing with correlated covariates, Partial dependence and ICE plot procedures generate predictions using observations outside of the joint distribution of the training data. An example of this can be seen in Fig. 1, where the red triangles indicate the data used to calculate $\bar{f}_1(X_1 = 1.5)$.

M plots avoid this issue by integrating over the conditional density instead of the marginal density. The dependence functions can be described as

$$f_{s,M}(X_s) = E_{X_c|X_s}\left[f(X_s, X_c)\middle|X_s = x_s\right]. \tag{3}$$

These can be locally approximated with Eq. 4, although more complex kernel smoothing methods are often used [2]. In the equation, $N(X_s)$ denotes the indices for an appropriately selected neighborhood containing an observed value x_{is} while $n(X_s)$ is the number of observations in the neighborhood:

$$\bar{f}_{s,M}(X_s) = \frac{1}{n(X_s)} \sum_{i \in N(X_s)}^{n} f(X_s, x_{ic}). \quad (4)$$

[2, 10], and [17] cite significant drawbacks of this approach. The most prominent issue is that the method is equivalent to evaluating the effect of X_s while ignoring the effects of X_c [17]. The result is that the main effect of the covariate of interest, X_s, cannot be distinguished from the effects of correlated variables (omitted variable bias) [2].

Accumulated Local Effects

Accumulated Local Effects (ALE) plots address several shortcomings of M plots and PDPs, but interpretation is intended for low-dimensional effects [2]. Specifically, ALE plots avoid the issue of extrapolation far outside of the feature space (PDPs) without biased or misleading results (M plots) and can be implemented in R [27] through ALEPlot [1]. The ALE main effect of X_s is defined as

$$f_{s,ALE}(X_s) = \int_{z_{0s}}^{X_s} E_{X_c}\left[f^s(X_s, X_c)\middle|X_s = z_s\right]\mathrm{d}z_s - \text{constant}, \quad (5)$$

where $f^s(X_s, X_c) = \frac{\partial f(X_s, X_c)}{\partial X_s}$, which is replaced in the estimation procedure by differences in predictions, and z_{0s} is chosen to be just below $\min\{x_{is} : x = 1, 2, \ldots, N\}$. Apley [2] develops the method and estimation procedure, paired with applied examples comparing PDPs, M plots, and ALE plots. Molnar [24] gives an overview of ALE plots and advocates their use over PDPs in most situations due to correlation between features.

Feature Importance

Feature (variable) importance measures can be used to identify predominant features in a fitted model. Available measures and implementation details can vary by algorithm, but we summarize the commonly used permutation approach, which was highlighted by Breiman [4] while introducing Random Forests. For Random Forests, we first calculate the out of bag (OOB) prediction error for each tree. We then permute one of the variables (X_i), destroying its relationship with the outcome, and once again calculate the OOB prediction errors. Finally, the variable importance for X_i is calculated as the difference in prediction error between the permuted and non-permutated OOB estimates, averaged over each tree [25]. The package randomForest [19] normalizes these values by the standard deviation of the differences and provides another measure of feature importance based on node impurities. Calculating the difference in prediction error for the entire model, rather than individual trees, while using either the training or testing dataset gives a model agnostic approach to feature importance [24].

Recent work from Covert et al. [6] based on Shapley values, a concept from coalitional game theory, presents an alternative model agnostic approach to feature importance. Shapley values can additionally be used to explain predictions for specific instances ([20, 24, 32]) and can be implemented through the iml package [23].

Other Model Agnostic Interpretation Methods

A straight forward approach to interpreting a black-box model is to fit a simpler model, such as a generalized linear model, using the original features and the black-box model's predictions. Traditional metrics such as R^2 can then be used to assess how well the simple surrogate model represents the original model [24]. Since the surrogate model is trained on predictions from the black-box model, conclusions should be about the model and not inference on the data. In addition, using surrogate methods, Local Interpretable Model Agnostic Explanations (LIME) present explainable predictions by constructing local surrogate models that focus on individual predictions [28] and can be used with the lime package [26]. For assessing the importance of feature interactions in their model, we direct readers to the iml package [23], which estimates the strength of feature interactions through the H-statistic as defined in [11]

Principal Components Analysis

Principal components analysis (PCA) is a dimensionality reduction technique that can increase interpretability in large datasets by making the structure clearer [3]. PCA has widespread application in a number of domains including image compression, facial recognition, and analyses for massive genomic datasets, and is a key component of our method. Dimensionality is reduced by finding orthogonal linear combinations of variables that successively maximize variance ([3, 18]), with the hope that a relatively small number of principal components explain a large portion of the variability. The optimal solution can be found either through an eigendecomposition of the covariance matrix [18], or as is implemented in totalvis through the prcomp() function in the stats package [27], by a singular value decomposition of the centered and scaled design matrix. Once solved, structure can be explored by interpreting the coefficients (loadings) of the linear combinations (principal components).

Novel Contributions

The primary contribution of this manuscript is a PCA-based model agnostic tool that groups related features and allows for visual interpretation of their combined effect. The method offers a promising avenue for interpretation in high-dimensional and high-dependence situations, and is paired with a diagnostic plot to help assess when its use may or may not be appropriate.

Package Components

The most important functions in totalvis are

- totalvis()
- partial_effects()
- plot() (methods added to the generic function)

Total Effect Plot

The underlying principle behind totalvis() is to work with groups of covariates that have been formed based on a transformation of the design matrix using PCA. The matrix of principal components is altered in a specified way and transformed back to the original scale for prediction. To be explicit, PCA is done to explore the structure among the covariates, but all predictions are done on the original scale used with whatever model the user employs. PCA is not required or intended to be a part of the actual black-box model under investigation, which may be a random forest, series of gradient boosted trees, neural network, or similar. The implementation of totalvis() allows users to visualize the average prediction for any principal component with a procedure that can be summarized as follows

Algorithm 1: totalvis procedure for a specificed principal component, j

1) Transform the design matrix $\mathbf{X}$ to $\mathbf{U}$ using PCA
2) Define a sequence $\mathbf{v}$ from $\min(\mathbf{u}_j)$ to $\max(\mathbf{u}_j)$
3) **for** *i in 1:length of* $\mathbf{v}$ **do**
   set every element in $\mathbf{u}_j = v_i$ and transform $\mathbf{U}$ back to the original scale := $\mathbf{Z}$
   obtain and average a vector of predictions := $\bar{f}_j(\mathbf{Z}(v_i))$
end
4) plot the pairs of $\{v_i, \bar{f}_j(\mathbf{Z}(v_i))\}$

In the standard implementation of PDPs, there is a direct mapping between the value every element of $\mathbf{x}_s$ is set to and the value of the covariate of interest. Algorithm 1 reflects this direct mapping by plotting a mean prediction against a specific value for a principal component. However, totalvis() also provides an option to visualize the average prediction against the mean of a specified covariate, which may be advantageous when a user is interested in understanding the typical behavior associated with a chosen feature.

In addition to the predictions, an object returned by totalvis() contains the components of the PCA. We incorporate details of the transformation using the magnitudes of the loadings as a measure of covariate importance to the principal component. To summarize this information, variables with the largest loadings are presented when plotting a "totalvis" object with arrows indicating the direction of their relationship with the principal component.

Total Effect Plots with a Pinned Feature

Given the limited interpretability of the x-axis when plotting against the range of a principal component, totalvis() allows the user to plot the prediction against a covariate of interest. We refer to this as 'pinning' a covariate. To do this, after the matrix has been mapped back to the original space, we average over the variable of interest and plot this value against the mean prediction. More formally this can be seen as plotting all pairs of

$$\left\{ \frac{1}{n} \sum_{i=1}^{n} z_{ij}(v_k), \bar{f}_l(\mathbf{Z}(v_k)) \right\}, \tag{6}$$

where $\mathbf{z_j}(v_k)$ is a specified vector obtained from transforming $\mathbf{U}$ back to the original space. For clarity, v_k is a value in the sequence specified for the lth principal component. In addition, note that the values of $z_{ij}(v_k)$ are implicitly associated with the chosen principal component.

Partial Effects Plot

totalvis() works by defining orthogonal linear combinations of the input variables and summarizing their effect on the outcome. Thus, it is important to ensure that the grouped covariates act in unison. Since PCA does not consider the relationships with the response, we need to identify when correlated variables behave differently than expected on the predictions. To address this concern, we provide partial_effects() as a way to check assumptions and provide further insight into the dependence structure. The process for generating 'partial_effects' plots is described in detail in algorithm 2, while the general procedure and interpretation are summarized in the following paragraphs.

Table 1 Available model types for classification and regression using the totalvis package

Type of model/package	Package	Class
Boosted trees	gbm [16]	"gbm"
	xgboost [5]	"xgb.Booster"
Generalized linear models	stats [27]	"glm", "lm"
Random forest	randomForest [19]	"randomForest"
Support Vector Machine	e1071 [22]	"svm"
Meta-packages	caret	"train"
	MachineShop	"MLModelFit"

Algorithm 2: partial_effects plot for principal component j

1) Transform the design matrix $\mathbf{X}$ to $\mathbf{U}$ using PCA
2) Define a sequence $\mathbf{v}$ from min($\mathbf{u}_j$) to max ($\mathbf{u}_j$)
3) **for** *i in 1:(length of* $\mathbf{v}$ *- 1)* **do**
   set every element in $\mathbf{u}_j = v_i$ and transform $\mathbf{U}$ back to the original scale := $\mathbf{Z}$
   obtain $\bar{f}_j(\mathbf{Z}(v_i))$
   set every element in $\mathbf{u}_j = v_{i+1}$ and transform $\mathbf{U}$ back to the original scale := $\mathbf{Z}^*$
   for *k in covariates of interest* **do**
      create a copy of $\mathbf{Z}^*$ as $\mathbf{A}$
      set $\mathbf{A}_{/k} = \mathbf{Z}_{/k}$ and obtain the mean prediction := $\tilde{f}_{jk}(\mathbf{A}(v_{i+1}, v_i))$
      calculate differences $d_{ik} := \tilde{f}_{jk}(\mathbf{A}(v_{i+1}, v_i))$ - $\bar{f}_j(\mathbf{Z}(v_i))$
   end
end
4) plot all combinations of $\{v_i, d_{ik}\}$

At a high level, for a specified principal component we give every observation the same value, v_n, and obtain the mean prediction. Denote the matrix that was used for prediction as $\mathbf{Z}(v_n)$ and observe that this matrix is the result of a transformation back to the original scale. The procedure then increases the value of the principal component to v_{n+1} and generates a new prediction matrix, $\mathbf{Z}(v_{n+1})$. Following this, the algorithm takes all the columns except one from $\mathbf{Z}(v_{n+1})$ and replaces them with columns from $\mathbf{Z}(v_n)$, creating a matrix with only a single modified covariate (relative to $\mathbf{Z}(v_n)$). In general, a mean prediction is obtained, as well as the difference between this value and the average prediction resulting from $\mathbf{Z}(v_n)$. We also provide the option to plot the average "shifted" predictions alongside the total effect curve, as this may help contextualize the diagnostic lines; an example of this can be referenced in Fig. 11.

The algorithm implements the above procedure using a sequence of equally spaced points from the minimum to the maximum of the principal component. The process is repeated for a select number of inputs that were main contributors to the principal component. In summary, we plot the differences in mean predictions caused by altering a single covariate in the appropriate direction and magnitude (based on the loadings). In the resulting plot, concordant signs (magnitudes may vary) across all input variables indicates that features are working in unison on the prediction.

Package Specifications and Defaults

totalvis is applicable in both the regression and binary classification settings. In regression, totalvis() and partial_effects() work with any model whose predict method uses a "data.frame" object. Note that the predict method for the trained model needs to be available in the local environment. totalvis functions also work with models fit using the caret [21] and MachineShop [31] meta-packages, as well as a few specific modeling types listed in Table 1.

Implementation of totalvis() relies on a transformation of a matrix of explanatory variables. For this reason, the package requires input of a numeric matrix or data frame with all numeric columns. In addition, the input matrix should exclude the response. To work with data that has categorical variables, the data should be properly encoded prior to training the model. Points are presented below to summarize this information, as well as provide additional details about the package.

- totalvis() requires input of numeric matrix or a data frame with all numeric columns (excludes the response)
- Categorical variables should be appropriately encoded before training the model
- totalvis() plots mean predictions against a sequence of values whose percentiles are equally spaced; the default is 50 points
- partial_effects() plots mean predictions against a sequence of equally spaced values; the default is 20 points
- prcomp() from the stats package is used for all PCA based transformations
- partial_effects() plots use a color blind friendly palette with 8 colors; colors are recycled if needed
- totalvis() and partial_effects() assume regression by default, classification must be specified through the "type" argument
- "Rugs" are included when plotting to avoid interpreting predictions from areas of low density
- totalvis() and partial_effects() plots can be generated for any principal component with the "pc_num" argument

Motivating Examples

To illustrate the attributes of totalvis, and to display the use and interpretation of the total and partial effect plots, we construct several simulated examples.

Total Effect

To demonstrate the notion of a 'total effect' of a group of covariates, we generate an example in which two correlated

```
# Packages and model training
library(kknn); library(MachineShop)
knn_model <- fit(ModelFrame(X, y), model = KNNModel)

# Figures 3 (PDP X1), 4 (PDP X2), and 5 (Total Effect)
plot(totalvis(knn_model, X, feature = "X1", pc_num = NULL))
plot(totalvis(knn_model, X, feature = "X2", pc_num = NULL))
plot(totalvis(knn_mod, X), legend_cex = 0.9)
```

Fig. 2 R code for partial dependence and total effect plots for the first simulation

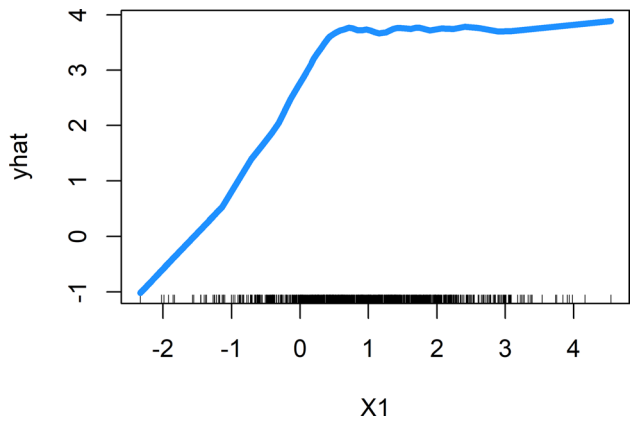


Fig. 3 PDP of X1 where features are correlated and have similar effects on the outcome

covariates affect the outcome in piecewise linear fashions. Further, the covariates impact the outcome in different regions of their space and the goal is to capture their net effect. Data were generated according to Eqs. 7 and 8, while exact specification and replication of the simulation can be found in Sect. "PDPs and ICE Curves" of the "Introduction" vignette. Instructions for downloading totalvis and accessing the vignette can be found in the associated GitHub repository.[1] The example showcases the ability of totalvis() to group related covariates and express the combined impact they have on the response. Note that the PCA transformation is ideally suited to situations in which some correlation exists between covariates.

$$\begin{pmatrix} X_{i1} \\ X_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} 1 \\ 1.2 \end{pmatrix}, \begin{pmatrix} 1 & 0.2 \\ 0.2 & 1 \end{pmatrix} \right) \tag{7}$$

$$y_i = \mathbb{1}_{[X_{i1}>0.5]} + 2X_{i1}\mathbb{1}_{[X_{i1}<=0.5]} + \mathbb{1}_{[X_{i2}<0.5]} + 2X_{i1}\mathbb{1}_{[X_{i2}>=0.5]} + \epsilon_i$$
$$\epsilon_i \sim N(0, 0.2) \tag{8}$$

The PDPs for the covariates (Figs. 3 and 4) capture the intended trends. As simulated, the covariates act in different locations on the real line, and the first principal component

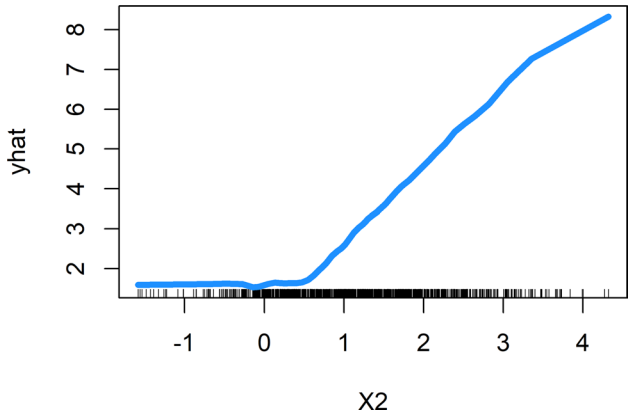


Fig. 4 PDP of X2 where features are correlated and have similar effects on the outcome

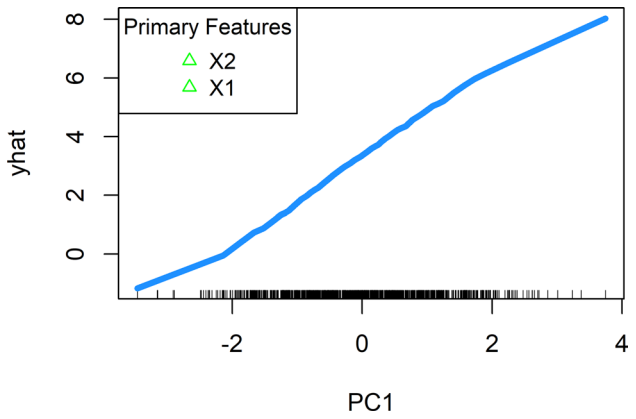


Fig. 5 Total effect plot of the first principal component where features are correlated and have similar effects on the outcome. Directions of the associated loadings are reported in the legend

captures the combined effect across the range (Fig. 5). This example 2 used a k-nearest neighbors model from the kknn package [30] which was fit using the meta-package MachineShop [31].

A practical example of comparable piecewise trends is related to medical expenses and quality of life. When dealing with an extended hospital stay or illness, future quality of life may be immediately impacted by treatment costs. However, the affect medical expenses have on the quality of life may taper off once an individual reaches their deductible or out of pocket maximum. At the same time, missing a couple days of work may not have drastic consequences on future quality of life. Yet, as the time missed increases, patients lose out on potential earnings and opportunities for career advancement. Combined, these two measures may better quantify the net financial burden caused by a medical issue and could be interpreted through their total effect.

[1] https://github.com/nickseedorff/totalvis.

```
# Packages and model training
library(gbm)
gbm_mod <- gbm(y ~ ., data = df, ntrees = 200,
               distribution = "gaussian")

# Figures 7 (total effect) and 8 (partial effects)
plot(totalvis(gbm_mod, X), legend_loc = "topleft")
plot(partial_effects(gbm_mod, X), legend_cex = 0.88)
```

Fig. 6 R code for total effect and partial effect plots for the second simulation

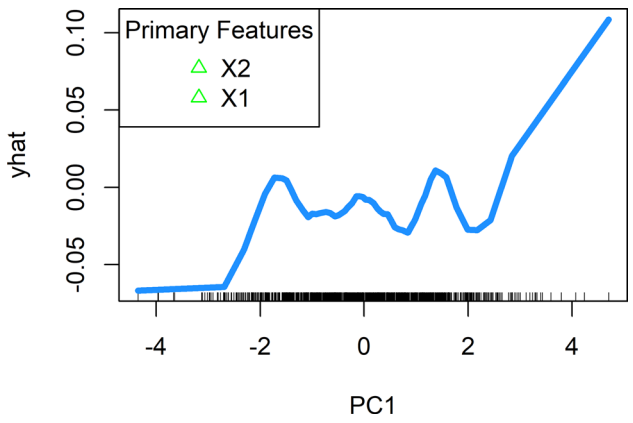


Fig. 7 Total effect of the first principal component where features are correlated and have contrasting effects on the outcome. Directions of the associated loadings are reported in the legend

Correlated Features with Different Effects

As previously indicated, totalvis() plots the effects of linear combinations of correlated covariates. However, the PCA transformation does not take the outcome into account, and the principal components should, therefore, be assessed to see if the covariates contribute constructively to the outcome. With this goal in mind, the next example (Eqs. 9 and 10) introduces a situation in which variables which are correlated nevertheless have opposite effects on the outcome of interest. Details of the simulation can be found in Sect. "M Plots" of the vignette. This demonstrates an important use of partial_effects(), not only to check assumptions, but also to illustrate interesting and important patterns of dependence. The model 6 in question was fit using gradient boosted trees, via the gbm() function in the gbm package [16].

$$\begin{pmatrix} X_{i1} \\ X_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix} \right) \tag{9}$$

$$y_i = X_{i1} - X_{i2} + \epsilon_i \epsilon_i \sim N(0, 0.1) \tag{10}$$

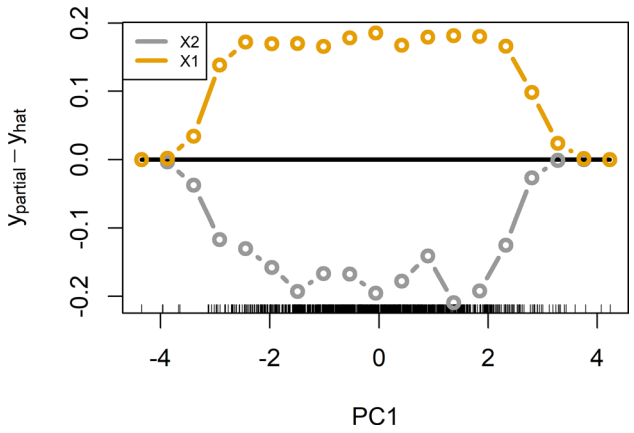


Fig. 8 Partial effects of the first principal component where features are correlated and have contrasting effects on the outcome

```
# Packages and model training
library(nnet); library(caret)
nnet_mod <- train(X, y, method = "nnet", trace = FALSE)

# Figure 10 pin plot
plot(totalvis(nnet_mod, X, pin = "X2", type = "classification"))

# Figure 11 partial effects
diag_obj <- partial_effects(mod, X, type = "classification")
plot(diag_obj, differenced = FALSE, legend_loc = "topleft")
```

Fig. 9 R code for pin and partial effect plots for the third simulation

The average prediction for the totalvis() plot in Fig. 7 jumps around zero, which was expected given the nullifying effects of the covariates. Used as a reference for interpretation, the black line at 0 in the partial_effects() plot (Fig. 8) indicates the average prediction when the principal component is set to the corresponding value of the x-axis. Across the range of the principal component, shifting the input variables separately shows opposing influence on the predictions, thus capturing their counteractive behavior. This application represents a situation in which the lines of the partial_effects() plot do not exhibit similar patterns; another example of this is presented in Fig. 11 while partial effect lines of a consistent nature are seen in Fig. 20.

Correlated Feature with No Effect

In the final example, we assess the ability of totalvis() to correctly identify which inputs are relevant to the response. To accomplish this, we generate two highly correlated covariates, but only one acts on the outcome (Eqs. 11 and 12). Full specification of the example is located in Sect. "Accumulated Local Effects" of the vignette. A suitable method for interpreting a model trained with this data should avoid major extrapolation outside the feature space (a drawback of PDPs and ICE plots) while correctly capturing the relevant inputs. In addition to motivating

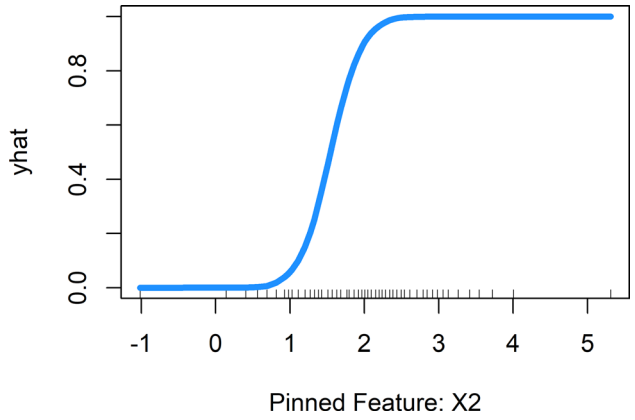


Fig. 10 totalvis pin plot for the first principal component and feature X2. In this example, features are correlated yet only X1 affects the outcome

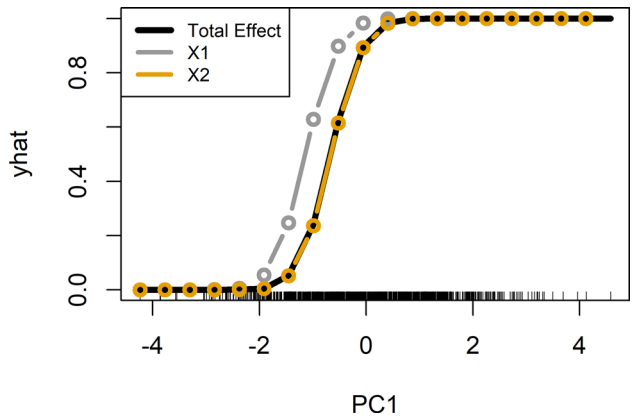


Fig. 11 Non-differenced partial effects for the first principal component where features are correlated yet only X1 affects the outcome

insights to be gleaned from partial_effects(), this also demonstrates the use of totalvis in the classification setting. To evaluate the simulation we trained a single-layer neural network 9 from nnet [29] using the caret package [21].

$$\begin{pmatrix} X_{i1} \\ X_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix} \right) \tag{11}$$

$$\begin{aligned} y_i &= \mathbb{1}_{[(X_{i1}+\epsilon_i)>1.5]} \\ \epsilon_i &\sim N(0, 0.1) \end{aligned} \tag{12}$$

To visually summarize the model while avoiding observations drastically outside of the input space, ALE or M plots could be implemented. ALE plots are a suitable replacement for determining which features affect the outcome, however, they have limited applicability when the goal is interpretation of groups of features. On the other hand, M plots cannot discern the effects between the two covariates

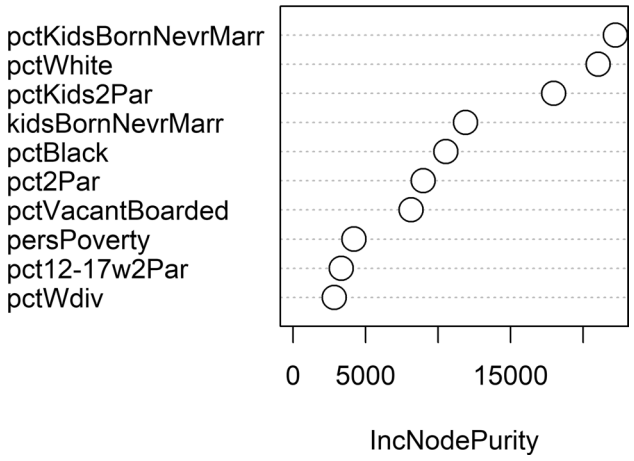


Fig. 12 Feature importance for the "randomForest" fit with the community and crimes data

and, in isolation, the totalvis pin plot approach has a similar result (Fig. 10). To address this issue, we make use of the non-differenced partial_effects() plot. When the principal component is between -2 and 0.5 (Fig. 11), shifting X1 produces substantially higher predicted values. On the other hand, shifting X2 alone produces a curve that is nearly identical to the total effect (black line), thus suggesting that X2 has little influence on the predictions.

An Application of totalvis

We present a motivating application of totalvis using a subset of the "Communities and Crime Unnormalized Data Set" located in the UCI repository [9] and compiled from several sources [7, 8, 33, 34]. The selected subset contains all numeric input variables with no missing values; the analysis is fully replicated in Sect. "Package components" of the "Introduction" vignette. We chose to use the number of murders per 100k population as our response variable (murdPerPop). The data used has 101 covariates, with more than 12% of the absolute correlations above 0.5.

To begin we fit a model using the randomForest() function from randomForest package [19]. randomForest offers a measure of feature importance which can be referenced in Fig. 12. PDPs are often created for the most important input variables; we take a detailed look at "pctKids2Par". In Fig. 13 we see a handful of covariates that are strongly correlated with "pctKids2Par", presenting an issue with PDPs. Note that several of the features listed in the importance plot could be viewed as indicators of neighborhood financial status.

Next we implement the totalvis() function 14 and visualize the output

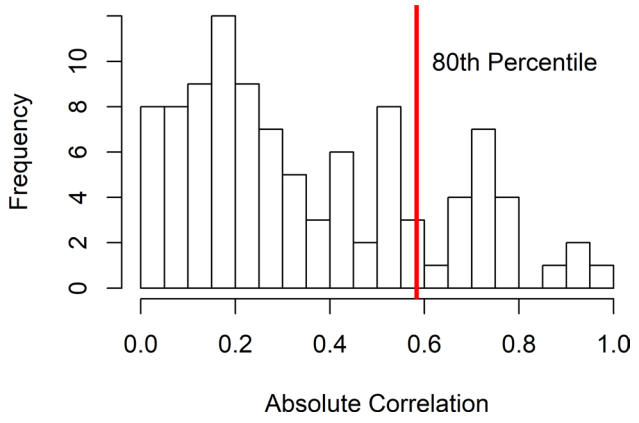
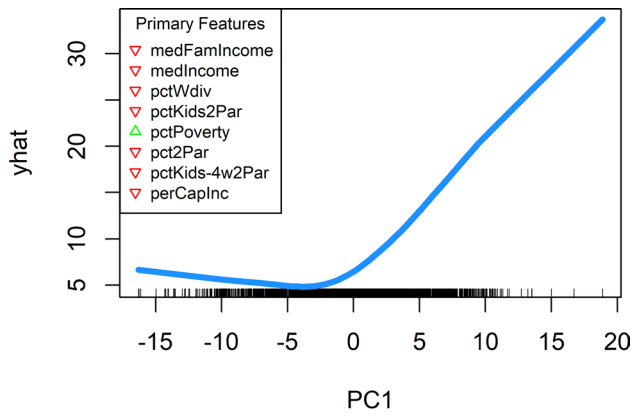


Fig. 13 Absolute correlations between pctKids2Par and the other features in the community and crimes data

```
# Packages and model training
library(randomForest)
rf_mod <- randomForest(X, outcome)

# Figure 15 total effect plot
vis_object <- totalvis(rf_mod, X)
plot(vis_object, num_load = 8)
```

Fig. 14 R code for the total effect plot for the Community and Crimes application



feature_name	loading_value
medFamIncome	-0.187
medIncome	-0.185
pctWdiv	-0.177
pctKids2Par	-0.175
pctPoverty	0.173
pct2Par	-0.173
pctKids-4w2Par	-0.172
perCapInc	-0.169

Fig. 15 totalvis() plot using the communities and crime data along with the associated totalvis() default output table. Directions of the loadings can be referenced in the plot, while directions and magnitudes of the loadings of the most important features are given in the Table

```
# Figure 17 (Pin Plot) and 18 (PDP)
plot(totalvis(rf_model, X, pin = "pctKids2Par"))
plot(totalvis(rf_model, X, feature = "pctKids2Par", pc_num = NULL))
```

Fig. 16 R code for the pin and partial dependence plots for the Community and Crimes application

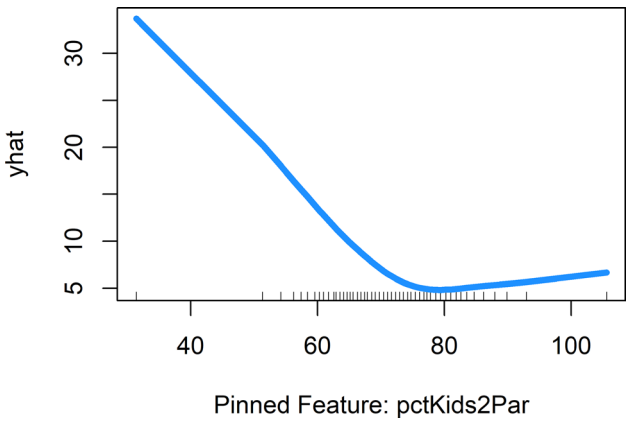


Fig. 17 Pin plot of variable 'pctKids2Par' and the first principal component using the communities and crimes data

The covariates with the strongest influence on the first principal component are presented with arrows in Fig. 15 and on the associated table. The listed covariates could all be interpreted as indicators of the financial status of a neighborhood (descriptions can be found in Table 2 in the appendix). In addition, worth noting is that the signs of the loadings are as expected; covariates whose relationship with income differ enter the linear combination with opposite signs (example: medFamIncome vs pctPoverty). Seen at a group level, an initial decrease in income in a neighborhood is associated with a decrease in crime. However, after a principal component value of roughly − 3.5 we see drastic a increase in crime as the principal component increases (covariates positively related to income decrease).

Note: When plotting a "totalvis" object the default is to return a dataframe with the information from the table in Fig. 15. This can be turned off by setting the return "return_res" argument in plot() to FALSE.

Total Effect Interpretation

Continuing with the crimes dataset and the first principal component, we compare a plot of totalvis() pinned to a specific feature (Fig. 17) to a PDP 16 for the same covariate (Fig. 18).

In the resulting "Pin Plot", related covariates increase/decrease in relation to the covariate of interest. This allows us to explore what typically happens across the range, i.e., correlated features are allowed to behave as expected.

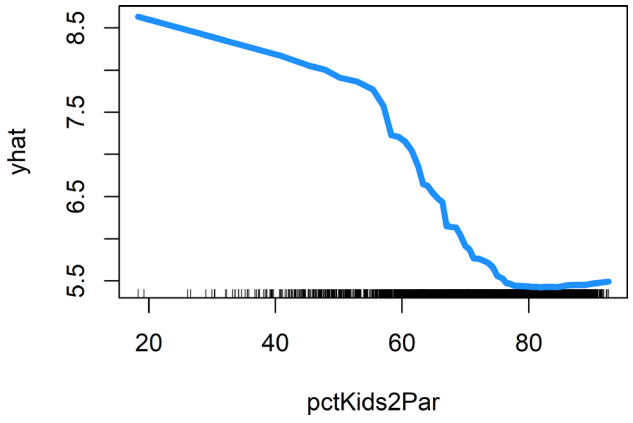


Fig. 18 PDP of variable 'pctKids2Par' using the communities and crimes data

```
# Create object and figure 20 (partial effects plot)
diag_obj <- partial_effects(rf_mod, X, num_load = 8)
plot(diag_obj, legend_loc = "topleft", legend_cex = 0.84)
```

Fig. 19 R code for the partial effects plot for the Community and Crimes application

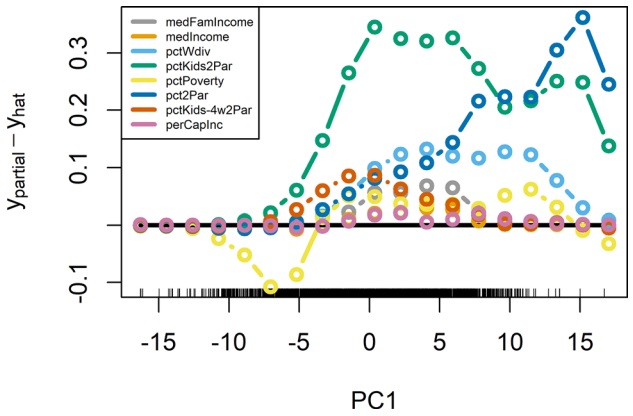


Fig. 20 Diagnostic plot for the first principal component using the community and crimes data. In general, shifting an individual feature increased the averaged predicted outcome indicating the grouped covariates largely act as expected

Comparing this to the PDP in Fig. 18, we see a large difference in the scale of the y-axis. Given the dependence between the covariates in the PDP, the main takeaway is the shape of the curve and not the scale of the y-axis. Due to this, we feel our approach offers more realistic changes to the average predictions and is more in line with a total effect interpretation. Note that the PDP could alternatively have been constructed using partialPlot() from the randomForest package [19].

Diagnostic Plot

While the total effect plots illustrated thus far are useful, they implicitly assume that the principal components structure of the covariates is meaningfully reflected in the data generating mechanism. To assess the reasonableness of this assumption, as well as to further explore the underlying relationships between covariates and the outcome, we utilize the differenced (default) partial_effects() plot 19. In Fig. 20, we see that the partial effects of the top loading variables in first principal component are generally positive, although the magnitudes differ due to their individual importance to the model. This is an example of a situation in which the grouped covariates primarily act as expected, jointly contributing to the outcome. In such a setting, the underlying assumptions of the total effect plot would seem to be reasonable.

Discussion

As presented, the totalvis method has applicability as a model agnostic visualization tool in a variety of settings, including high-dimensional and high-dependence situations, as well as offering a natural approach to reason about potential casual effects. The method provides a natural companion to partial dependence and ICE plots to enable users of black-box models to understand and investigate the patterns learned by their models.

This area remains ripe for further investigation. In particular, the requirement to perform PCA imposes an additional computational burden on the user, above and beyond that required to fit the original models. Investigation of scalable approaches to totalvis-like visualization methods would increase the scope of application. In addition, more work to formally investigate the relationship between these techniques and causal models could potentially provide diagnostic tools as well as inspiring new quantitative approaches to effect estimation in causal machine learning models.

Nevertheless, these techniques are already readily applicable to many analytical problems and numerous model types, and are readily extensible to new modeling approaches. All materials and methods to reproduce these results are available on https://github.com/nickseedorff/totalvis, which includes a vignette that reproduces all tables and figures presented here.

Appendix

See Table 2 for a summary of the relevant variables explored in the community and crimes application.

Table 2 Descriptions of relevant variables from the communities and crimes data

Variable_name	Variable_type	Variable_description
murdPerPop	Outcome	Number of murders per 100k population
medFamIncome	Feature	Median family income
medIncome	Feature	Median household income
pctWdiv	Feature	% of households with investment/rent income
pctKids2Par	Feature	% of kids in family housing with two parents
pctPoverty	Feature	% of people under the poverty level
pct2Par	Feature	% of families with two parents
pctKids-4w2Par	Feature	% of kids 4 and under in two parent households
perCapInc	Feature	Per capita income

Declarations

Conflict of interest On behalf of all the authors, the corresponding author states that there is no conflict of interest.

Funding None.

Availability of data and material All data are publicly available at https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized.

Code availability Code open source available at https://github.com/nickseedorff/totalvis.

References

1. Apley D. ALEPlot: Accumulated local effects (ALE) plots and partial dependence (PD) plots. https://CRAN.R-project.org/package=ALEPlot, r package version 1.1 2018.
2. Apley, D. Visualizing the effects of predictor variables in black box supervised learning models. arXiv preprint (2016). https://arxiv.org/pdf/1612.08468.pdf.
3. Bartholomew D. Principal components analysis. In: Peterson P, Baker E, McGaw B, editors. International encyclopedia of education. 3rd ed. Oxford: Elsevier; 2010. p. 374–7. https://doi.org/10.1016/B978-0-08-044894-7.01358-0.
4. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32. https://doi.org/10.1023/A:1010933404324.
5. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K, Mitchell R, Cano I, Zhou T, Li M, Xie J, Lin M, Geng Y, Li Y. xgboost: Extreme Gradient Boosting. 2019. https://CRAN.R-project.org/package=xgboost, r package version 0.90.0.2.
6. Covert I, Lundberg S, Lee SI. Understanding global feature contributions with additive importance measures. 2020. arXiv:2004.00668.
7. DOJ, BJS. Crime in the United States (computer file). 1995.
8. DOJ, BJS. Law Enforcement Management And Administrative Statistics (Computer File) U.S. Department Of Commerce, Bureau Of The Census Producer, Washington, DC and Inter-university Consortium for Political and Social Research. 1992.
9. Dua D, Graff C. UCI machine learning repository. 2017. http://archive.ics.uci.edu/ml.
10. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29(5):1189–232. https://doi.org/10.1214/aos/1013203451.
11. Friedman J, Popescu B. Predictive learning via rule ensembles. Ann Appl Stat. 2008;2(3):916–54. https://doi.org/10.1214/07-AOAS148.
12. Goldstein A, Kapelner A, Bleich J, Pitkin E. Peeking inside the black box: visualizing statistical learning with plots of individual conditional expectation. J Comput Gr Stat. 2015;24(1):44–65. https://doi.org/10.1080/10618600.2014.907095.
13. Goldstein A, Kapelner A, Bleich J. ICEbox: Individual conditional expectation plot toolbox. 2017. https://CRAN.R-project.org/package=ICEbox, r package version 1.1.2.
14. Greenwell BM. pdp: an R package for constructing partial dependence plots. R J. 2017;9(1):421–36. https://doi.org/10.32614/RJ-2017-016.
15. Greenwell B. pdp: Partial dependence plots. 2018. https://CRAN.R-project.org/package=pdp, r package version 0.7.0.
16. Greenwell B, Boehmke B, Cunningham J, Developers G. gbm: Generalized boosted regression models. 2019. https://CRAN.R-project.org/package=gbm, r package version 2.1.5.
17. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. Berlin: Springer; 2009.
18. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philos Trans A Math Phys Eng Sci. 2016;374(2065):20150202.
19. Liaw A, Wiener M. Classification and regression by randomforest. R News. 2002;2(3):18–22. https://CRAN.R-project.org/doc/Rnews/.
20. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., pp. 4765–4774, 2017. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.
21. M Kuhn Contributions from J Wing, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, Kenkel B, the R Core Team, Benesty M, Lescarbeau R, Ziem A, Scrucca L, Tang Y, Candan C, Hunt T. caret: Classification and regression training. 2019. https://CRAN.R-project.org/package=caret, r package version 6.0-84.
22. Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F. e1071: Misc functions of the department of statistics, probability theory group (Formerly: E1071), TU Wien. 2019. https://CRAN.R-project.org/package=e1071, r package version 1.7-2.
23. Molnar C. iml: Interpretable Machine Learning. 2019a. https://CRAN.R-project.org/package=iml, r package version 0.9.0.
24. Molnar C. Interpretable machine learning. 2019b. https://christophm.github.io/interpretable-ml-book/.
25. Nembrini S, König IR, Wright MN. The revival of the Gini importance? Bioinformatics. 2018;34(21):3711–3718. https://doi.org/10.1093/bioinformatics/bty373, https://academic.oup.

com/bioinformatics/article-pdf/34/21/3711/26146978/bty373.pdf
26. Pedersen TL, Benesty M. lime: Local Interpretable Model-Agnostic Explanations. 2019. https://CRAN.R-project.org/package=lime, r package version 0.5.1.
27. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2019. https://www.R-project.org/.
28. Riberio M, Singh S, Guestrin C. "Why should I trust you?": explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. 2016. https://doi.org/10.1145/2939672.2939778.
29. Ripley B. nnet: feed-forward neural networks and multinomial log-linear models. 2016. https://CRAN.R-project.org/package=nnet, r package version 7.3-12.
30. Schliep K, Hechenbichler K. kknn: Weighted k-nearest neighbors. 2016. https://CRAN.R-project.org/package=kknn, r package version 1.3.1.
31. Smith BJ. MachineShop: machine learning models and tools. 2019. https://cran.r-project.org/package=MachineShop, r package version 1.6.0.
32. Štrumbelj E, Kononenko I. Explaining prediction models and individual predictions with feature contributions. Knowl Inf Syst. 2014;41(3):647–65. https://doi.org/10.1007/s10115-013-0679-x.
33. USDOC, USCB. Census of population and housing 1990 United States: summary tape file 1a & 3a (computer files). 1990.
34. USDOC, USCB. Washington, DC and Inter-university Consortium for Political and Social Research. 1992.