Supplementary Methods

1 The three medical datasets used for experiments

1.1 Mortality dataset

The mortality data was obtained from the National Health and Nutrition Examination Survey (NHANES I) conducted by the U.S. Centers for Disease Control (CDC), as well as the NHANES I Epidemiologic Follow-up Study (NHEFS) [14]. Raw tape-format data files were obtained from the CDC website and converted to a tabular format by custom scripts. This reformatted version of the public data has been released at `http://github.com/suinleelab/treexplainer-study`. NHANES I examined 23,808 individuals in the United States between 1971 and 1974, recording a large number of clinical and laboratory measurements. The NHANES I Epidemiologic Followup Study researched the status of the original NHANES I participants as of 1992 to identify when they had died, or alternatively when they were last known to be alive. After filtering NHANES I to subjects that had both followup mortality data as well as common health measurements (such as systolic blood pressure) we obtained 79 features for 14,407 individuals, of which 4,785 individuals had recorded death dates before 1992. This data was used to train several cox proportional hazard ratio model types (Section). Because NHANES I represents a cross section of the United States population, it is a classic dataset that has been often used to understand the association between standard clinical measurements and long term health outcomes [36, 18].

1.2 Chronic kidney disease dataset

The kidney dataset is from the Chronic Renal Insufficiency Cohort (CRIC) Study which follows individuals with chronic kidney disease recruited from 7 clinical centers [35]. Participants are assessed at an annual visit and study follow-up is ongoing. We joined both visit and person level attributes to create 333 features for 10,745 visits among 3,939 patients. For each visit, we determined if the patient progressed to 'end-stage renal disease' within the following four years. (End-stage renal disease is the last stage of chronic kidney disease when kidneys are functioning at 10-15% of their normal capacity; dialysis or kidney transplantation is necessary to sustain life.) Predicting this progression outcome results in a binary classification task. Understanding what leads some people with chronic kidney disease to progress to end-stage renal disease while others do not is a priority in clinical kidney care that can help doctors and patients better manage and treat their condition [45, 68, 66, 42]. In the United States chronic kidney disease affects 14% of the population, so improving our management and understanding of the disease can have a significant positive impact on public health [31].

1.3 Hospital procedure duration dataset

Our hospital's operating rooms have installed an Anesthesia Information Management System - AIMS (Merge AIMS, Merge Inc., Hartland, WI) that integrates with medical devices and other electronic medical record systems to automatically acquire hemodynamic, ventilation, laboratory, surgery schedule and patient registration data. The automatic capture of data is supplemented by the manual documentation of medications and anesthesia interventions to complete the anesthesia record during a surgical episode. We extracted data from the AIMS database from June 2012 to May 2016. The corresponding medical history data of each patient were also extracted from our electronic medical record data warehouse (Caradigm Inc., Redmond, WA). Patient and procedure specific data available prior to surgery were captured and summarized into 2,185 features over 147,000 procedures. These data consist of diagnosis codes, procedure types, location, free text notes (represented as a bag of words), and various other information recorded in the EMR system. We measured the duration of a procedure as the time spent in the room by the patient. This is an important prediction task since one potential application of machine learning in hospitals is to reduce costs by better anticipating procedure duration (and so improve surgery scheduling). When hospital scheduling systems depend on machine learning models it is important to monitor the ongoing performance of the model. We demonstrate in the main text results how local explanations can significantly improve the process of monitoring this type of model deployment.

2 Model accuracy performance experiments

Modern gradient boosted decision trees often provide state-of-the-art performance on tabular style datasets where features are individually meaningful, as consistently demonstrated by open data science competitions [20, 10]. All three medical datasets we examine here represent tabular-style data, and gradient boosted trees achieve the highest accuracy across all three datasets (Figure 2A). We use 100 random train/test splits followed by retraining to assess the statistical significance of the separation between methods. In the mortality dataset, gradient boosted trees outperform the linear lasso with a P-value < 0.01 and the neural network with a P-value of 0.03. In the chronic kidney disease dataset, gradient boosted trees outperform the linear lasso with a P-value < 0.01 and the neural network with a P-value of 0.08. In the hospital procedure duration dataset, gradient boosted trees outperform both the linear lasso and neural network with a P-value < 0.01. The details of how we train each method and obtain the results in Figure 2A are presented below.

2.1 Mortality dataset

For NHANES I mortality prediction in Figure 2A, we use a cox proportional hazards loss, and a C-statistic [24] to measure performance. For each of the three algorithms we split the $14,407$ samples according to a 64/16/20 split for train/validation/test. The features for the linear and the neural network models are mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models, we pass the original data unnormalized and with missing values.

Gradient Boosted Trees - Gradient boosted trees are implemented using an XGBoost [10] model with the cox proportional hazards objective (we implemented this objective and merged it into XGBoost to support the experiments in this paper). Hyper-parameters are chosen using coordinate descent on the validation set loss. This results in a learning rate of 0.001; $6,765$ trees of max depth 4 chosen using early stopping; ℓ_2 regularization of 5.5; no ℓ_1 regularization; no column sampling during fitting; and bagging sub-sampling of 50%.

Linear Model - The linear model for the mortality dataset is implemented using the lifelines Python package [16]. The ℓ_2 regularization weight is chosen using the validation set and set to 215.44.

Neural Network - The neural network model is implemented using the DeepSurv Python package [29]. Running DeepSurv to convergence for mortality data takes hours on a modern GPU server, so hyper-parameter tuning is done by manual coordinate decent. This results in ℓ_2 regularization of 1.0; the use of batch normalization; a single hidden layer with 20 nodes; a dropout rate of 0.5; a learning rate of 0.001 with no decay; and momentum of 0.9.

2.2 Chronic kidney disease dataset

For CRIC kidney disease prediction, we use logistic loss for binary classification, and measure performance using the area under the precision-recall curve in Figure 2A. For each of the three algorithms we split the $10,745$ samples according to a 64/16/20 split for train/validation/test. The features for the linear and the neural network models are mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models we pass the original data unnormalized and with missing values.

Gradient Boosted Trees - Gradient boosted trees are implemented using an XGBoost [10] model with the binary logistic objective function. Hyper-parameters are chosen using coordinate descent on the validation set loss. This results in a learning rate of 0.003; $2,300$ trees of max depth 5 chosen using early stopping; no ℓ_2 regularization; no ℓ_1 regularization; a column sampling rate during fitting of 15%; and bagging sub-sampling of 30%.

Linear Model - The linear model for the kidney dataset is implemented using scikit-learn [47]. Both ℓ_1 and ℓ_2 regularization are tested and ℓ_1 is selected based on validation set performance, with an optimal penalty of 0.1798.

Neural Network - The neural network model is implemented using Keras [12]. We choose to explore various feed-forward network architectures as well as 1D convolution based kernels (that learn a shared non-linear transform across many features). The best performance on the validation data comes from a 1D convolution based architecture. After coordinate descent hyper-parameter tuning, we choose 15 1D convolution kernels which then go into a layer of 10 hidden units with rectified linear unit activation functions. Dropout of 0.4 is used during training between the convolution kernels and the hidden layer, and between the hidden layer and the output. Because stochastic gradient descent can have varying performance from run to run, we choose the best model based on validation loss from 10 different optimization runs.

2.3 Hospital procedure duration dataset

For hospital procedure duration prediction, we use a squared error loss and measure performance by the coefficient of determination (R^2) in Figure 2A. Two different hospital procedure dataset splits are used in this paper. The first dataset split, used for model comparison in Section (Figure 2A), consists of 147,000 procedures divided according to a random 80/10/5/5 split for train/validation/test1/test2. The test2 set is only used to get final performance numbers and not during method development. The features for the linear and the neural network models are mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models, we pass the original data unnormalized and with missing values. All hyper-parameter tuning is done with the validation dataset.

The second dataset split, used for the monitoring plots, divides procedures from the two hospitals based on time: the first 1 year of data is used for training and the last 3 years is used for test, in a manner intended to simulate actual model deployment. Models for this data are not hyper-parameter tuned, as the goal is not to achieve perfect performance but to demonstrate the value of monitoring methods; instead, they simply use the best hyper-parameter values found on the first (random-splits) dataset.

Gradient Boosted Trees - Gradient boosted trees are implemented using an XGBoost [10] model with a squared loss objective function. Hyper-parameters are chosen using grid search over the validation set loss. This results in a learning rate of 0.1; 530 trees of max depth 10 chosen using early stopping; a column sampling rate during fitting of 50%; and no bagging sub-sampling.

Linear Model - The linear model for the procedure duration dataset is implemented using the `LassoCV` class in scikit-learn [47]. ℓ_1 regularization is tuned based on validation set performance to 0.120.

Neural Network - The neural network model is implemented using Keras [12]. We limit our architecture search to feed-forward networks with up to three hidden layers, and sizes up to 4,096 nodes per layer. The best performance on validation data comes from a single-hidden-layer architecture with a 1,024-node layer followed by a dropout probability of 0.75 before the output.

3 Interpretability comparison of linear models and tree-based models in the presence of non-linearities

Even if linear models appear easier to interpret, their high bias may force the relationships they learn to be farther from the truth than a low-bias tree-based model (Figures 2B-D). We illustrate this using *simulated outcomes* from the NHANES I mortality dataset with varying amounts of non-linearity between certain features and the outcome. We use a random set of 10189 patients from NHANES-I, and 118 features. No rows or columns are dropped out due to missingness or unclean data. The data are standardized and missing values are mean-imputed. Even if the simulated outcome only depends on age and body mass index (BMI), the linear model learns features in the mortality dataset that are non-linearly dependent with age and BMI to try to approximate non-linear relationships with the outcome. This increases the test accuracy of the linear model slightly, as seen by the small increase in performance from the two-feature linear model (dashed red) to the all-feature linear model (solid red) (Figure 2C). However, this comes at the cost of placing much of its weight on features that are not actually used to generate the outcome. In contrast, the gradient boosted tree model correctly uses only age and BMI across all levels of function non-linearity (Figure 2D). Below we describe the experimental setup behind the results in Figure 2B-D.

3.1 Selection of features

We generate a synthetic label with a known relationship to two input features from the mortality dataset, but to the extent possible, we intend this synthetic relationship to be realistic; while still retaining the ability to control the amount of non-linearity in the relationship. Starting with the classifiers trained on the full mortality (NHANES I) dataset, SHAP dependence plots are used to find one feature that has a strong linear relationship with mortality (age) and one that has a "U-shaped" relationship with mortality (BMI). These two features are selected and used as the "true" label-generating features in our synthetic model.

3.2 Label generation

The synthetic label-generating function is constructed using a logistic function applied to the sum of a linear function of age and a quadratic function of BMI. The functional form allows us to smoothly vary the amount of nonlinearity in the label-generating model. The quadratic function of BMI is parameterized as $(1-p)$ times a linear function, plus p times a quadratic function, where both functions have the same minimum and maximum values (the x-location of the minimum for the quadratic is set to the mean BMI). The contribution in logits is set to range from a minimum of -2.0 to a maximum of 2.0 for age, and -1.0 to 1.0 for BMI, so that even when nonlinear contributions are the strongest ($p = 1$) the linear contribution of the main risk factor is still more important (as is true in the original data). The output for each data point is then a probability to be predicted by our models (we do not add noise by binarizing these probabilities to 0-1, so as to avoid the need to average our results over many random replicates). Thus the final label-generating model is

$$y = \sigma\left((1.265(\text{age}) + 0.0233) + (1-p)(0.388(\text{BMI}) - 0.325) + (p)(1.714(\text{BMI})^2 - 1)\right)$$

where $\sigma = \frac{1}{1+e^{-t}}$ is the logistic function. A total of 11 datasets are generated, with $p \in [0.0, 0.1, 0.2...1.0]$, by applying this labeling function to the true data covariates, so that the matrix of predictors X is real data but the labels y are generated synthetically by a known mechanism.

3.3 Model training

We train both gradient boosted trees and linear logistic regression to predict the synthetic labels. For each of the 11 datasets with varying degrees p of nonlinearity, models are tuned on a validation set and evaluated on test data with a 6000/1500/2689 patient train/validation/test split. Logistic regression is unregularized in the experiments shown in the main text; in Supplementary Figures 2 and 3 we show that the same pattern of results hold when an L1 regularization penalty is tuned on validation data over the range $\lambda \in [10^{-10}, 10^{-9}...10^{3}, 10^{4}]$. The tuned gradient boosting model hyper-parameters are optimized over the tree depths of $[1, 2, 4, 8, 10]$ and bagging sub-sampling over the rates $[0.2, 0.5, 0.8, 1.0]$. The learning rate is fixed at 0.01; the minimum loss reduction for splitting is 1.0; the minimum child weight for splitting is 10; and we train for a maximum of $1,000$ rounds with 10-round early stopping based on validation set loss.

3.4 Feature importance

Per-sample importance values are calculated for each feature in each of the 11 datasets using SHAP values for both the logistic (using Linear SHAP values assuming feature independence [39]) and gradient boosted tree (using TreeExplainer's Tree SHAP algorithm) models. At each value of p, the total weight of irrelevant features is calculated by taking the absolute value of all SHAP values for all features other than age and BMI, and summing these values across all samples and features.

4 Convergence experiments for model agnostic Shapley value approximations

In Supplementary Figures 4C-D we generated random datasets of increasing size and then explained (over)fit XGBoost models with 1,000 trees. The runtime and standard deviation of the local explanations are reported for Kernel SHAP [39], IME [64], and TreeExplainer; except that for Kernel SHAP and IME the reported times are only a lower bound. Both the IME and Kernel SHAP model-agnostic methods must evaluate the original model a specific number of times for each explanation, so the time spent evaluating the original model represents a lower bound on the runtime of the methods (note that the QII method [15] is not included in our comparisons since for local feature attribution it is identical to IME). In Supplementary Figure 4C we report this lower bound, but it is important to note that in practice there is also additional overhead associated with the actual execution of the methods that depends on how efficiently they are implemented. We also only used a single background reference sample for the model-agnostic approaches. This allows them to converge faster at the expense of using less accurate estimates of conditional expectations. Increasing the number of background samples would only further reduce the computational performance of these methods. Each method is run ten times, then the standard deviation for each feature is divided by the mean of each feature to get a normalized standard deviation (Supplementary Figure 4D). In order to maintain a constant level of normalized standard deviation, Kernel SHAP and IME are allowed a linearly increasing number of samples as the number of features in a dataset, M, grows. In Supplementary Figure 4C, TreeExplainer is so much faster than the model-agnostic methods that it appears to remain unchanged as we scale M, though in reality there is a small growth in its runtime. In Supplementary Figure 4D there is truly no variability since the TreeExplainer method is exact and not stochastic.

In Supplementary Figures 4E-F, the different explainers are compared in terms of estimation error (absolute deviation from the ground truth) on the chronic kidney disease dataset. We chose this dataset model for our comparisons because it is much smaller than the hospital procedure duration dataset, and has a more common loss function than the mortality dataset model (logistic loss vs. a cox proportional hazards loss). Ground truth is obtained via TreeExplainer's exact Independent Tree SHAP algorithm with the reference set fixed to be the mean of the data. The plots are obtained by increasing the number of samples allowed to each explainer and reporting the max and mean estimation error. For IME, we tune the minimum samples per feature, a hyper-parameter that is utilized to estimate which features' attributions have larger variability. After the minimum samples per feature has been achieved, the rest of the samples are allocated so as to optimally reduce the variance of the sum of the estimated values (by giving more samples to features with high sampling variance). As expected this is beneficial to the max evaluation error, but can potentially lead to bias (as for IME (min 10) in Supplementary Figure 4E). While often helpful, ℓ_1 regularization was not useful to improve Kernel SHAP for this dataset, so we report the results from unregularized regression.

To measure the cost of computing low variance estimates of explanations for the chronic kidney disease dataset we defined "low variance" as 1% of the tenth largest feature impact (out of 333 features), then measured how many samples it took on average to reach a standard deviation below that level (where standard deviation is measured across repeated runs of the explanation method). This was done for both the maximum standard deviation across all features (Supplementary Figure 4E), and the mean standard deviation (Supplementary Figure 4F). Calculating low variance estimates for the experiments presented in this paper on the chronic kidney disease dataset would have taken almost 2 CPU days for basic explanations, and over 3 CPU years for interaction values.

5 Additional properties of SHAP values

In addition to the three properties described in the main text, SHAP values also satisfy several previously proposed properties of good explanation methods:

- *Implementation invariance*, which states that the explanation of a model should only depend on the behavior of the model and not how it is implemented [65]. This is a desirable property since if two models behave identically across all inputs, then it natural that they should get identical explanations of their behavior. SHAP values satisfy this property since Equation 4 only depends on the model's behavior and not on any implementation details.

- *Linearity*, which states that a joint model made by linearly combining two other models on the same inputs should have explanations that are the same linear combination of the explanations of each of the constituent models [64]. SHAP values follow linearity as it is a well-known property of Shapley values [58].

- *Sensitivity-n*, which is a metric for which Shapley values provide a justifiable (in a sense optimal) solution [1]. Sensitivity-n measures how well a set of feature attribution values can represent the effect on the model's output of masking a random set of n features. For complex functions, no method can perfectly represent all high order effects by summing feature attribution effects, but Shapley values provide a principled way to pick a solution. This is because Shapley values are the only solution that is both consistent (Property 2), and locally accurate (meaning the values sum to the model's output when all features are included).
- *Explanation continuity*, which states that for continuous models, small changes in the input should lead to small changes in the explanation [44]. SHAP values satisfy this because they are a linear function of the model's output, and hence if the underlying model is continuous, the SHAP values will be also. Note that for TreeExplainer this property is typically not applicable since tree-based models are not continuous.

6 Evaluation metrics for benchmark

6.1 Runtime

Runtime is reported as the time to explain 1,000 predictions. For the sake of efficiency the runtime for each explanation method was measured using 100 random predictions, and then scaled by 10 to represent the time to explain 1,000 predictions. Both the initialization time of each method and the per-prediction time was measured, and only the per-prediction time was scaled.

6.2 Local accuracy

Local accuracy strictly holds only when the sum of the attribution values exactly sum up from some constant base value to the output of the model for each prediction (Property 1). This means $E_x[(f(x) - \sum_i \phi_i)^2] = 0$. But to also capture how close methods come to achieving local accuracy when they fail, we compute the normalized standard deviation of the difference from the model's output over 100 samples

$$\sigma = \frac{\sqrt{E_x[(f(x) - \sum_i \phi_i)^2]}}{\sqrt{E_x[f(x)^2]}} \tag{1}$$

then define nine cutoff levels of σ for reporting a positive score between 0 and 1:

$$\sigma < 10^{-6} \implies 1.00 \tag{2}$$
$$10^{-6} \leq \sigma < 0.01 \implies 0.90 \tag{3}$$
$$0.01 \leq \sigma < 0.05 \implies 0.75 \tag{4}$$
$$0.05 \leq \sigma < 0.10 \implies 0.60 \tag{5}$$
$$0.10 \leq \sigma < 0.20 \implies 0.40 \tag{6}$$
$$0.20 \leq \sigma < 0.30 \implies 0.30 \tag{7}$$
$$0.30 \leq \sigma < 0.50 \implies 0.20 \tag{8}$$
$$0.50 \leq \sigma < 0.70 \implies 0.10 \tag{9}$$

6.3 Consistency guarantees

Consistency guarantees are a theoretical property of an explanation method that ensure pairs of cases will never be inconsistent (Property 2). We broke agreement with this property into three different categories: an exact guarantee, a guarantee that holds in the case of infinite sampling, and no guarantee. Note that while inconsistency could be tested computationally, it would require enumerating a search space exponential in the number of input features, which is why we chose to directly report the theoretical guarantees provided by different methods.

6.4 Keep positive (mask)

The Keep Positive (mask) metric measures the ability of an explanation method to find the features that increased the output of the model the most. For a single input the most positive input features are kept at their original values, while all the other input features are masked with their mean value. This is done for eleven different fractions of features ordered by how positive an impact they have as estimated by the explanation method we are evaluating (those that have a negative impact are always removed and never kept). Plotting the fraction of features kept vs. the model output produces a curve that measures how well the local explanation method has identified features that increase the model's output for this prediction. Higher valued curves represent better explanation methods. We average this curve over explanations of 100 test samples for 10 different models trained on different train/test splits. To summarize how well an explanation method performed we take the area under this curve. Masking features and observing their impact on a model's output is a common method for assessing local explanation methods [1, 39, 59]. An example plot of this metric for a random forest model of the chronic kidney disease model is available in Supplementary Figure 21.

6.5 Keep positive (resample)

The Keep Positive (resample) metric is similar to the Keep Positive (mask) metric, but instead of replacing hidden features with their mean value, this resample version of the metric replaces them with values from a random training sample. This replacement with values from the training dataset is repeated 100 times and the model output's are averaged to integrate over the background distribution. If the input features are independent then this estimates the expectation of the model output conditioned on the observed features. The mask version of this metric described above can also be viewed as approximating the conditional expectation of the model's output, but only if the model is linear. The resample metric does not make the assumption of model linearity. An example plot of this metric for a random forest model of the chronic kidney disease model is available in Supplementary Figure 22.

6.6 Keep negative (mask)

The Keep Negative (mask) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Positive (mask) metric described above, but keeps the most negative impacting features as computed by the explanation method (Supplementary Figure 23).

6.7 Keep negative (resample)

The Keep Negative (resample) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Positive (resample) metric described above, but keeps the most negative impacting features as computed by the explanation method (Supplementary Figure 24).

6.8 Keep absolute (mask)

The Keep Absolute (mask) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Positive (mask) metric described above, but keeps the most important features as measured by the absolute value of the score given by the explanation method (Supplementary Figure 25). Since removing features by the absolute value of their effect on the model is not designed to push the model's output either higher or lower, we measure not the change in the model's output, but rather the change in the model's accuracy. Good explanations will enable the model to achieve high accuracy with only a few important features.

6.9 Keep absolute (resample)

The Keep Absolute (resample) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (mask) metric described above, but uses resampling instead of mean masking (Supplementary Figure 26).

6.10 Remove positive (mask)

The Remove Positive (mask) metric measures the ability of an explanation method to find the features that increased the output of the model the most. It works just like the Keep Positive (mask) metric described above, but instead of keeping the most positive features, it instead removes them, which should lower the model output (Supplementary Figure 27). This metric is conceptually similar to the sequential masking metric used by Fong and Vedaldi to sequentially hide features that raise the probability of a specific output class [19].

6.11 Remove positive (resample)

The Remove Positive (resample) metric measures the ability of an explanation method to find the features that increased the output of the model the most. It works just like the Keep Positive (resample) metric described above, but instead of keeping the most positive features, it instead removes them, which should lower the model output (Supplementary Figure 28).

6.12 Remove negative (mask)

The Remove Negative (mask) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Negative (mask) metric described above, but instead of keeping the most negative features, it instead removes them, which should raise the model output (Supplementary Figure 29).

6.13 Remove negative (resample)

The Remove Negative (resample) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Negative (resample) metric described above, but instead of keeping the most negative features, it instead removes them, which should raise the model output (Supplementary Figure 30).

6.14 Remove absolute (mask)

The Remove Absolute (mask) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (mask) metric described above, but instead of keeping the most important features, it instead removes them, which should lower the model's performance (Supplementary Figure 31). This metric is equivelant to the perterbation metric used by Samek et al. to sequentially hide important features [55], and the explanation selectivity property described in Montavon, Samek, and Müller [44].

6.15 Remove absolute (resample)

The Remove Absolute (resample) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (resample) metric described above, but instead of keeping the most important features, it instead removes them, which should lower the model's performance (Supplementary Figure 32).

7 User study experiments

Here we explore how consistent different attribution methods are with human intuition. While most complex models cannot be easily explained by humans, very simple decision trees can be explained by people. This means we can run user study experiments that measure how people assign credit to input features for simple models that people understand, then compare people's allocations to allocations given by different local explanation methods. If an explanation method gives the same result as humans, then we say that method is consistent with human intuition.

To test human intuition, we use four simple models and ask 33 english speaking individuals in the U.S. on Amazon Mechanical Turk to explain three different samples for each model. Each model is a simple depth-two decision tree that only depends on two binary features. To make the model more approachable we called the model a "sickness score," and used three binary input features: *fever*, *cough*, and *headache*. The models we used were: *AND*, *OR*, *XOR*, and *SUM* (study participants were not told these names). The headache feature was never used by any of the models. The fever and cough features always each contributed a linear effect of +2 when they were on, but for models other than SUM there were also non-linear effects. For AND +6 was given when both features were true. For OR +6 was given when either feature was true. For XOR +6 was given when either feature was true but not both. For each model we explained three samples: 1) fever false, cough false, headache true; 2) fever false, cough true, headache true; and 3) fever true, cough true, headache true.

Users were asked to allocate blame for a sickness score output value among each of the three input features. No constraints were placed on the input fields used to capture user input, except that the fields were not left blank when there was a non-zero sickness score. This experiment resulted in twelve different consensus credit allocations (Supplementary Figures 33-36). We then took these twelve consensus credit allocations and used them as ground truth for twelve metrics. Each metric is the sum of the absolute differences between the human consensus feature attribution and attribution given by an explanation method (Supplementary Figures 37-39). Note that we used a healthy population with independent input features as the background reference dataset (the background is used for computing conditional expectations by the explanation methods). Python implementations of these study scenarios are available online `https://github.com/suinleelab/treeexplainer-study`. Results are provided in Supplementary Results Section 7.1

7.1 User Study Results

The results of the user study show that all the local explanation methods based on Shapley values agree with the human consensus feature attribution values across all twelve cases. However, the heuristic Saabas method differs significantly from the human consensus in several of the nonlinear cases (Supplementary Figure 8). Not only do Shapley values have attractive theoretical guarantees, and strong quantitative performance (Figure 3), but these experiments show they also match human intuition on small example models (Supplementary Figure 8). Performance plots for all user study results are also available in Supplementary Data 1.

Supplementary Results

1 Prevalence of tree-based machine learning models

Here we seek to quantify how prevalent tree-based machine learning models are in health care, industry, and academia. First, we use the Kaggle "State of ML and Data Science" 2017 survey [28] to understand the relative importance of different methods in health care, industry, academia, and other fields (Supplementary Figure 18). This study surveyed over 16,000 people who used machine learning in their jobs and measured, among other things, the percentage of respondents who used various data science methods. Note that the percentages do not add to 100 percent because respondents may use multiple methods and because some methods are specific sub-types of other methods. We filter to focus on survey results from the 2,039 respondents who self-identified as "data scientists" and had provided information on the methods they used. In particular, we examine model usage across all data scientists as well as in the specific fields of pharmaceuticals, marketing, finance, insurance, and academia. These fields had 31, 107, 265, 74, and 102 data scientist respondents, respectively. Overall, tree-based methods are the most popular non-linear methods, representing four of the five most popular nonlinear models across all fields. No nonlinear model is more popular than random forests and decision trees in any field we examine. The third most popular nonlinear models are gradient boosting machines in finance and insurance, ensemble methods in marketing, Bayesian techniques in academia, and neural networks in pharmaceuticals. Ensemble methods are the third most popular nonlinear methods overall.

The lowest prevalence of trees occurs in academia, where the prevalence of Bayesian techniques and neural networks correspondingly increases.

We also investigate data science methods in an academic health care setting. To do this, we compile all articles published in *npj Digital Medicine* between January 1 and September 1 in 2019 and determine which methods they used. Out of all 58 articles, we select the 28 that used any kind of predictive model, and count the dataset types and model types used in each article. Four articles use multiple dataset types and are included in the counts for each dataset type. Models that are not linear, tree, or deep models are grouped together as "other". We do not filter or rank models by predictive performance. The results are plotted in Supplementary Figure 19. The height of each bar is the total number of articles using a particular dataset type, and the shaded areas within each bar represent the proportion of papers using the given dataset type that also used a particular model type. The shading is normalized such that models from a paper that uses 5 different model types only count one-fifth as much as those from a paper that uses a single model type. Within articles that use a predictive model, images and tabular data are the most popular data type. Both are substantially more popular than time series, the next most popular data type. Both trees and deep models are used on image, tabular, and time series data, but deep models are much more common than trees on image data and trees were much more common than deep models on tabular data. Papers utilizing time series data are equally likely to use tree models and deep models.

Overall, our survey of the field concludes that tree-based methods are widespread in a variety of fields, from health care to academia to industry.

2 Previous Global Explanation Methods for Trees

While local explanation methods for trees have not been extensively studied, interpreting tree-based models by assigning a global importance value to each input feature is a well studied problem and many methods have been proposed [21, 20, 56, 62, 63, 26, 2, 38, 30]. The most basic global approach is to simply count the number of times a feature was used for splitting, but this fails to account for the differing impacts of different splits. A better approach is to attribute the reduction in loss (aka. Gain) provided by each split in each decision tree to the feature that was split on [8, 21]. This "Gain" measure of feature importance was shown to correctly recover the mutual information between the input features and the outcome label in the limit of an infinite ensemble of totally random fully developed trees [38]. However, it becomes biased for finite ensembles of greedily built trees, and so approaches have been designed to account for this bias when using Gain for feature selection [25, 9]. Another popular method for determining feature importance is to permute the data column corresponding to a feature and then observe the change in the model's loss [7]. If the model's loss increases significantly when a feature is permuted then it indicates the model was heavily depending on that feature. This permutation approach can be further extended to account for statistical dependencies between features by permuting only within specified groups of samples [63]. All of these approaches are designed to estimate the global importance of a feature over an entire dataset, so they are not directly applicable to local explanations that are specific to each prediction. If we try to use global methods in place of true local explanations we get significantly worse performance on many benchmark metrics (Figure 3).

While not explicitly designed to be a global feature attribution method, TreeExplainer can be used as a global method by averaging many local explanations. If we do this over all samples in a dataset, then we get a global measure of feature importance that does not suffer from the inconsistencies of the classic Gain method (Supplementary Figure 5), and unlike the permutation method, does not miss high-order interaction effects. Global feature attribution based on TreeExplainer has a higher power to detect important features in the presence of interactions than current state-of-the-art methods (Supplementary Figure 9). This has important implications for the popular task of feature selection based on tree-ensembles.

3 Previous local explanation methods for trees

To our knowledge, there are only two previous tree-specific local explanation methods: reporting the decision path; and an unpublished heuristic difference in expectations method (proposed by Saabas) [54]. Since reporting the decision path is not useful for large tree ensembles we instead focus on the heuristic Saabas method. The Saabas difference in expectations approach explains a prediction by following the decision path and attributing changes in the expected output of the model to each feature along the path. This is efficient since the expected value of every node in the tree can be estimated by averaging the model output over all the training samples that pass through that node. Let f be a decision tree model, x the instance we are going to explain, $f(x)$ the output of the model for the current instance, and $f_x(S) \approx E[f(x) \mid x_S]$ the estimated expectation of the model output conditioned on the set S of feature values (, Algorithm 1), then we can define the *Saabas value* for the i'th feature as

$$\phi_i^s(f, x) = \sum_{j \in D_x^i} f_x(A_j \cup j) - f_x(A_j), \tag{10}$$

where D_x^i is the set of nodes on the decision path from x that split on feature i, and A_j is the set of all features split on by ancestors of j. Equation 10 results in a set of feature attribution values that sum up to the difference between the expected output of the model and the output for the current prediction being explained (Supplementary Figure 15). When explaining an ensemble model made up of a sum of many decision trees, the Saabas values for the ensemble model are defined as the sum of the Saabas values for each tree.

3.1 Unifying previous heuristics with Shapley values

In this section, we aim to unify previous heuristic methods with the Shapley values. For a more in-depth discussion of the Shapley values, please refer to Supplementary Section .

Surprisingly, there is a close parallel between the Saabas values (Equation 10) and the SHAP values (Equation 4). While SHAP values average the importance of introducing a feature into a interventional expectation $E[f(X) \mid do(X_S = x_S)]$ [27] over all possible feature orderings, Saabas values only consider the single ordering defined by a tree's decision path (Supplementary Figure 15). This means that Saabas values satisfy the local accuracy property since they always sum up to the difference between the expected value of the model $E[f(x)]$ and the current output $f(x)$. But since they do not average over all orderings they do not match the SHAP values, and so must violate consistency. Examples of such inconsistencies can be found for even very small trees, where changing the model to depend more on one feature can actually cause the Saabas attribution values for that feature to decrease (Supplementary Figure 5). The difference this makes can be seen by examining trees representing multi-way AND functions. If we let all the input features be independent and identically distributed then no feature in an AND function should have any more credit than another, yet for Saabas values, splits near the root are given much less credit than splits near the leaves (Supplementary Figures 4A). This means that while mathematically all the features play an equal role in the model, the features near the root of the tree get little or no credit. This is particularly troubling since features near the root are likely the most important as they were chosen first by greedy splitting.

There is a close parallel between Saabas values and the classic "Gain" method for global feature attribution (sometimes known as Gini importance) [21]. Just as Gain attributes the change in loss after each split to the feature that was split on, so Saabas attributes the change in conditional expectation after each split to the feature that was split on. Both methods only consider a single order for introducing features into the model, the order defined by paths in the tree. Choosing to use only a single ordering leads to inconsistent allocation of credit (Supplementary Figures 4A and 5). Shapley values guarantee a consistent allocation of credit by averaging the change after each split over all possible orderings. It has been previously shown through a connection with mutual information (which is consistent) that the Gain method becomes consistent in the limit of an infinite ensemble of totally random fully developed trees [38]. This suggests that the Saabas method may also become consistent in the limit of an infinite ensemble of totally random fully developed trees. This is indeed the case:

Theorem 1. *In the limit of an infinite ensemble of totally random fully developed trees on binary features the Saabas values equal the SHAP values [39] (where conditional expectations are computed using path expectations as in Algorithm 1).*

Proof. Assume all features are binary, then the decision path of a single input instance x for a single tree will be a random ordering of all the input features. The Saabas values for that tree will be equivalent to a single permutation from the formula for Shapley values (Equation 4). Since there is an infinite ensemble of trees, all possible feature orderings will be represented in equal proportions. Given a finite output domain, there will furthermore be all possible feature orderings represented for each possible leaf value. Taking the average of the Saabas values over the ensemble of trees then becomes the same as the averaging function in the definition of the Shapley values (Equation 4). □

4 Model agnostic local explanation methods

Many different local explanation methods have been proposed in the literature [4, 64, 51, 15, 39, 50]. The most well known is simply taking the gradient of the model's output with respect to its inputs at the current sample. This is common in the deep learning literature, as is the related approach of multiplying the gradient times the value of the input features. Relying entirely on the gradient of the model at a single point though can often be misleading [59]. To provide a better allocation of credit for deep learning models, various methods have been proposed that either modify the standard gradient back propagation rule to instead propagate attributions [61, 69, 3, 59, 33, 1], or integrate the gradient along a path based on fair allocation rules from economics [65].

In contrast to deep learning methods, model-agnostic methods make no assumptions about the internal structure of the model. These methods rely only on observing the relationship between changes in the model inputs and model outputs. This can be done by training a global mimic model to approximate the original model, then locally explaining the mimic model by either taking its gradient [4], or fitting a local linear model as in MAPLE [49]. Alternatively, the mimic model can be fit to the original model locally for each prediction. For a local linear mimic model the coefficients can be used as an explanation, as in the popular LIME method [51]. For a local decision rule mimic model the rules can be used as the explanation as in Anchors [50]. Another class of approaches do not explicitly fit a mimic model, but instead perturb sets of features to measure their importance, then use methods from game theory to fairly allocate the importance of these sets among the input features, this class includes IME [64] and QII [15].

Perhaps surprisingly, despite the seeming variety of different local explanation methods, two back propagation-style deep learning methods [3, 59], local linear mimic models [51], and several game theoretic methods [37, 64, 15] were recently unified into a single class of *additive feature attribution methods* in our prior study [39]. This class is of particular interest since results from cooperative game theory imply there is a unique optimal explanation approach in the class (the Shapley values) that satisfies several desirable properties [58, 53]. Unfortunately computing the Shapley values is NP-hard in general [41], with a runtime cost exponential in the number of input features. When faced with an NP-hard optimization problem it is typical to build approximate methods, which exactly IME [64] or QII [15] do. However, here we take an alternative approach and restrict our focus specifically to tree-based machine learning models. By doing this we are able to show constructively that solving for the exact Shapley values in trees is not NP-hard, and can be solved by TreeExplainer in low-order polynomial time ().

5 Model summarization experiments

Here, we describe in more detail the model summarization results. One of the most basic ways to understand a model is to display the global importance of each feature, often as a bar chart (Figure 4A left). For tree-based models such as gradient boosted trees a basic operation supported by any implementation is providing the total "Gain" over all splits for a feature as a global measure of feature importance [21]. Computing SHAP values across a whole dataset, we can improve on this basic task of displaying global feature importance in two ways: 1) By averaging the SHAP values across a dataset, we can get a single global measure of feature importance that retains the theoretical guarantees of SHAP values. This avoids the troubling inconsistency problems of the classic Gain method (Supplementary Figure 5), and also provides better global feature selection power than either Gain or permutation testing (Supplementary Figure 9). This is particularly important since tree-based models are often used in practice for feature selection [22, 46]. 2) A limitation of traditional global explanations for trees is that reporting a single number as the measure of a feature's importance conflates two important and distinct concepts: the magnitude of an effect, and the prevalence of an effect. By plotting many local explanations in a beeswarm-style *SHAP summary plot* we can see both the magnitude and prevalence of a feature's effect (Figure 4A right), and by adding color, we can also display the effect's direction.

The value of summary plots based on many local explanations is illustrated in Figure 4A for the NHANES I mortality dataset, where an XGBoost cox proportional hazards model was trained using hyper-parameters optimized on a validation dataset (2), its predictions were explained using TreeExplainer, and then compiled into a summary plot. On the left of Figure 4A is a familiar bar-chart, not based on a typical heuristic global measure of feature importance, but on the average magnitude of the SHAP values. On the right of Figure 4A is a set of beeswarm plots where each dot corresponds to an individual person in the study. Each person has one dot for each feature, where the position of the dot on the x-axis corresponds to the impact that feature has on the model's prediction for that person (as measured by the prediction's SHAP value for that feature). When multiple dots land at the same x position they pile up to show density.

Unsurprisingly, the dominating factor for risk of death in the U.S. in the 1970s (which is when the NHANES I data was collected) is age. By examining the top row of the summary plot we can see that a high value for the age feature (red) corresponds to a large increase in the log hazard ratio (i.e., a large positive SHAP value), while a low value for age (blue) corresponds to a large decrease in the log hazard ratio (i.e., a large negative SHAP value). The next most important feature for mortality prediction is sex, with men having about a 0.6 increase in log-hazards relative to women, which corresponds to about 7 years of change in the age feature. Interestingly, the impact of being a man vs. woman on the model's output is not constant across individuals, as can be seen by the spread of the blue and red dots. The differences of effect within the same sex are due to interactions with other features in the model that modulate the importance of sex for different individuals (we explore this in more detail in 6).

An important pattern in the mortality data revealed by the summary plot, but not by classic global feature importance, is that features with a low global importance can still be some of the most important features for a specific person (Figure 4A). Blood protein, for example, has a low global impact on mortality, as indicated by its small global importance (Figure 4A left). However, for some individuals, high blood protein has a very large impact on their mortality risk, as indicated by the long tail of red dots stretching to the right in the summary plot (Figure 4A right). This trend of rare high magnitude effects is present across many of the features, and always stretches to the right. This reflects the fact that there are many ways to die abnormally early when medical measurements are out of range, but there not many ways to live abnormally longer (since there are no long tails stretching to the left). Summary plots combine many local explanations to provide a more detailed global picture of the model than a traditional list of global feature importance values. In many cases this more detailed view can provide useful insights, as demonstrated in our medical datasets (Figure 4A, Supplementary Figures 16 and 17).

6 Feature dependence experiments

Here we describe in more detail the feature dependence results. Just as summary plots provide richer information than traditional measures of global feature importance (Figure 4A), dependence plots based on SHAP values can provide richer information than traditional partial dependence plots by combining the local importance of a single feature across many samples (Figure 4B-G). Plotting a feature's value on the x-axis vs. the feature's SHAP value on the y-axis produces a *SHAP dependence plot* that shows how much that feature impacted the prediction of every sample in the dataset.

For the mortality model a SHAP dependence plot reproduces the standard risk inflection point known for systolic blood pressure between 120 mmHg and 140 mmHg [23] (Figure 4B). This highlights the value of a flexible model that is able to capture non-linear patterns, and also shows how interaction effects can have a significant impact on a feature. Standard partial dependence plots capture the general trends, but do not provide any information about the heterogeneity present within people that have the same measured systolic blood pressure (Supplementary Figure 10). Each dot in a SHAP dependence plot represents a person, and the vertical dispersion of the dots is driven by interaction effects with other features in the model. Many different individuals have a recorded blood pressure of 180 mmHg in the mortality dataset, but the impact that measurement has on their log-hazard ratio varies from 0.2 to 0.6 because of other factors that differ among these individuals. We can color by another feature to better understand what drives this vertical dispersion. Coloring by age in Figure 4B explains most of the dispersion, meaning that early onset high blood pressure is more concerning to the model than late onset high blood pressure.

For the chronic kidney disease (CKD) model a SHAP dependence plot again clearly reveals the previously documented non-linear inflection point for systolic blood pressure risk, but in this dataset the vertical dispersion from interaction effects appears to be partially driven by differences in blood urea nitrogen (Figure 4E). Correctly modeling blood pressure risk is important, since blood pressure control in select CKD populations may delay progression of kidney disease and reduce the risk of cardiovascular events. Lower blood pressure has been found to slow progression of CKD and decrease overall cardiovascular mortality in some studies [67, 52, 60, 48, 57, 5]. For example, long-term follow-up of the Modification of Diet in Renal Disease (MDRD) study suggested that lower systolic blood pressure led to improved kidney outcomes in patients with CKD [57]. The SPRINT trial, which randomized patients to treatment to systolic blood pressure <120 vs. <140 mmHg found that treatment to lower systolic blood pressure was associated with lower risk of cardiovascular disease; though no difference was seen in rates of CKD progression between the treatment groups [23, 11].

7 Interaction effect experiments

Here we describe in more detail the interaction effect results. Using SHAP interaction values we can decompose the impact of a feature on a specific sample into interaction effects with other features. SHAP interaction values allow pairwise interaction effects to be measured at an individual sample level. By combining many such samples we can then observe patterns of interaction effects across a dataset.

In the mortality dataset, we can compute the SHAP interaction values for every sample and then decompose the systolic blood pressure dependence plot to get the (symmetric) SHAP interaction value of systolic blood pressure and age. Interaction effects are visible in dependence plots through vertical dispersion of the samples, and coloring can often highlight patterns likely to explain this dispersion, but it is necessary to compute the SHAP interaction values to confirm the causes of vertical dispersion. In the systolic blood pressure dependence plot from the mortality model the vertical dispersion is primarily driven by an interaction with age (Figure 4D), as suggested by the original coloring (Figure 4B).

Plotting interaction values can reveal interesting relationships picked up by complex tree-ensemble models that would otherwise be hidden, such as the interaction effect between age and sex described in the main text results. In the chronic kidney disease model an interesting interaction effect is observed between 'white blood cells' and 'blood urea nitrogen' (Figure 4F). This means that high white blood cell counts are more concerning to the model when they are accompanied by high blood urea nitrogen. Recent evidence has suggested that inflammation may be an important contributor to loss of kidney function [6, 17]. While there are numerous markers of inflammation, white blood cell count is one of the most commonly measured clinical tests available, and this interaction effect supports the notion that inflammation may interact with high blood urea nitrogen to contribute to faster kidney function decline.

8 Model monitoring experiments

Here we describe in more detail the model monitoring results. Deploying machine learning models in practice is challenging because they depend on a large range of input features, any of which could change after deployment and lead to degraded performance. Finding problems in deployed models is difficult because the result of bugs is typically not a software crash, but rather a change in an already stochastic measure of prediction performance. It is hard to determine when a change in a model's performance is due to a feature problem, an expected generalization error, or random noise. Because of this, many bugs in machine learning pipelines can go undetected, even in core software at top tech companies [70].

A natural first step when debugging model deployments is to identify which features are causing problems. Computing the SHAP values of a model's loss function directly supports this by decomposing the loss among the model's input features. This has two important advantages over traditional model loss monitoring: First, it assigns blame directly to the problematic features so that instead of looking at global fluctuations of model performance, one can see the impact each feature has on the performance. Second, by focusing on individual features we have higher power to identify problems that would otherwise be hidden in all the other fluctuations of the overall model loss. Improving our ability to monitor deployed models is an important part of enabling the safe use of machine learning in medicine.

To simulate a model deployment we used the hospital procedure duration prediction dataset. It contains four years of data from two large hospitals. We used the first year of data for training and ran the model on the next three years of data to simulate a deployment. This is a simple batch prediction task, and so is far less prone to errors than actual real-time deployments, yet even here we observed dataset issues that had not been previously detected during data cleaning (Figure 5).

Figure 5A shows the smoothed loss of the procedure duration model over time, and represents the type of monitoring used widely in industry today. There is a clear increase in the model's error once we switch to the test set, which was expected; then there are short spikes in the error that are hard to differentiate from random fluctuations. To test the value of using monitoring plots based on local explanations we intentionally swapped the labels of operating rooms 6 and 13 two-thirds of the way through the dataset. This is meant to represent the type of coding change that can often happen during active development of a real-time machine learning pipeline. In the overall loss of the model's predictions, two-thirds of the way through there is no indication that a problem has occurred (Figure 5A), which means this type of monitoring would not be able to catch the issue. In contrast, the *SHAP monitoring plot* for the room 6 feature clearly shows when the room labeling error begins (Figure 5B). The y-axis of the SHAP monitoring plot is the impact of the room 6 feature on the loss. About two-thirds of the way through the data there is a clear shift from negative values (meaning using the feature helps accuracy) to positive values (meaning using the feature hurts accuracy). The impact on accuracy is substantial, but because procedures occurring in room 6 and 13 are just a small fraction of the overall set of procedures, the increased error is invisible when looking at the overall loss (Figure 5A).

In addition to finding our intentional error, SHAP monitoring plots also revealed problems that were already present in the dataset. Two of these are shown Figure 5C and D.

In Figure 5C we can see a spike in error for the general anesthesia feature shortly after the deployment window begins. This spike represents a transient configuration issue in our hospital revealed by a SHAP monitoring plot. It corresponds to a subset of procedures from a single hospital where the anesthesia type data field was left blank. After going back to the hospital system we found that this was due to a temporary electronic medical record configuration issue whereby the link between the general anesthesia induction note and the anesthesia type got broken. This prevented the anesthesia type data field from being auto-populated with "general anesthesia" when the induction note was completed. This is exactly the type of configuration issue that impacts machine learning models deployed in practice, and so needs to be detected during model monitoring.

In Figure 5D there is an example, not of a processing error, but of feature drift over time. The atrial fibrillation feature denotes if a patient is undergoing an atrial fibrillation ablation procedure. During the training period, and for some time into deployment, using the atrial fibrillation feature lowers the loss. But then over the course of time the feature becomes less and less useful until it begins hurting the model by the end of the deployment window. Based on the SHAP monitoring plot for atrial fibrillation we went back to the hospital to determine possible causes. During an atrial fibrillation ablation, the goal is to electrically isolate the pulmonary veins of the heart using long catheters placed from the groin area. Traditionally, the procedure was completed with a radiofrequency ablation catheter delivering point-by-point lesions to burn the left atrium around the pulmonary veins. During the deployment window, the hospital began to use the second generation cryoballoon catheter (Arctic Front Advance, Medtronic Inc., Minneapolis, MN), which freezes the tissue and has been demonstrated to have a shorter procedure duration compared to radiofrequency ablation [34]. At the same time, there were improvements in radiofrequency ablation catheter technology including the use of contact force sensing which allowed the operator to determine how strongly the catheter was touching the left atrial wall. This technology ensures that ablation lesions are delivered with enough force to create significant lesion size. With noncontact force catheters, the operator may think the catheter is touching the atrial wall but it may, in actuality, simply be floating nearby. Contact force sensing is also associated with shorter procedure times [40, 32]. Cryoballoon versus radiofrequency ablation is chosen based on patient characteristics and physician preference. Lastly, during this time there were staffing changes including the use of specialized electrophysiology technologists which decreased procedural preparation time. All of these changes led to a significant decrease in atrial fibrillation ablation procedural and in-room time during the test period (Supplementary Figure 11), which translated into high model error attributable to the atrial fibrillation feature. We quantified the significance of the SHAP monitoring plot trend using an independent t-test between atrial fibrillation ablation procedures that appear in the first 30,000 samples of the test time period vs. those that appear in the last 60,000 samples of the test time period. This lead to a P-value of 5.4×10^{-19}. The complexity of the reasons behind the feature drift in Figure 5D illustrate why it is so difficult to anticipate how assumptions depended on by a model might break in practice. Using SHAP values to monitor the loss of a model allow us to retroactively identify how model assumptions may be changing individually for each input feature, even if we cannot a priori anticipate which features are likely to cause problems.

Explaining the loss of a model is not only useful for monitoring over time, it can also be used in dependence plots to understand how a specific feature helps improve model performance. For hospital procedure duration prediction we can plot the dependence between the time of the day feature and it's impact on the model's output. The time of day feature indicates when a procedure started, and is particularly effective at reducing the model's loss just after 7:30am and 8:30am in the morning (Supplementary Figure 20). These two times are when long-running elective surgeries are scheduled in this hospital system, so the dependence plot reveals that the model is using the time of the day to detect routine long-running surgeries.

Current practice is to monitor the overall loss of the model, and also potentially monitor the statistics of the features for changes over time. This is problematic because just monitoring the overall loss of the model can hide many important problems, while monitoring the changes in the statistics of features is essentially an unsupervised anomaly detection problem that is prone to both false positives and false negatives. Swapping the names of operating rooms that are used equally often would be invisible to such an unsupervised method. By directly attributing the loss to the features we can highlight precisely which features are impacting the loss and by how much. When changes show up in the monitoring plot, those changes are in the units of the model's loss and so we can quantify how much it is impacting our performance. These monitoring plots represent a compelling way that many local explanations can be combined to provide richer and more actionable insights into model behavior than the current state of the art.

9 Local explanation embedding experiments

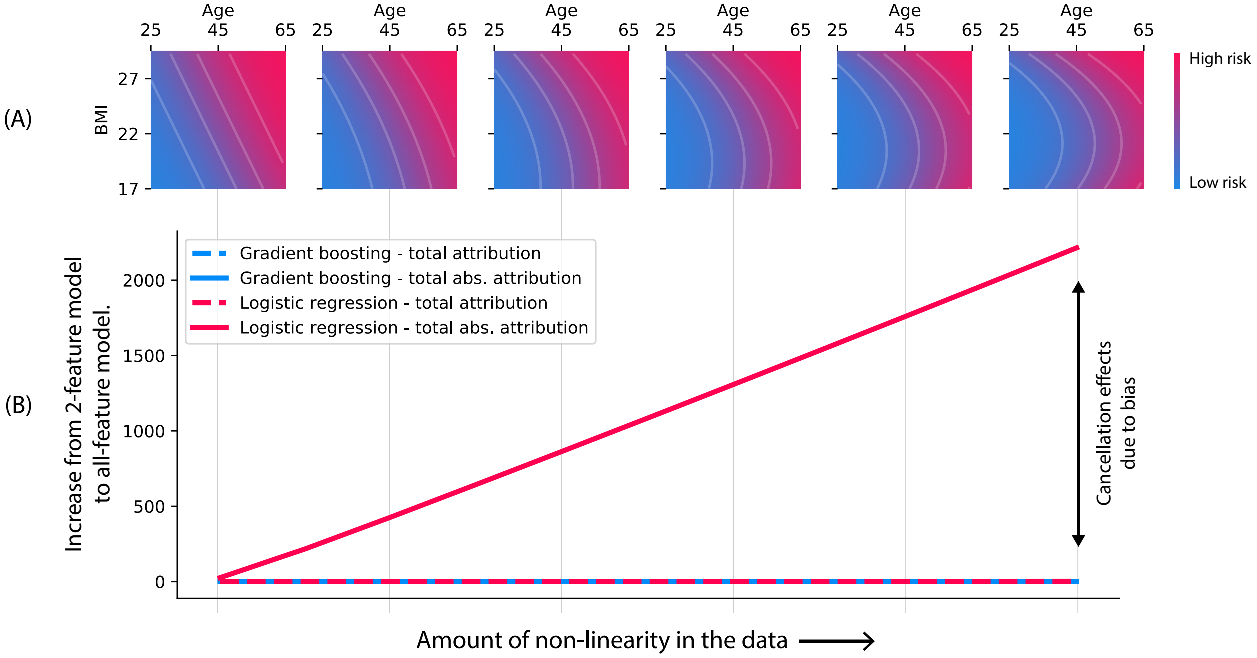
Here we describe in more detail the local explanation embedding results. There are two challenges with standard unsupervised clustering methods: 1) The distance metric does not account for the discrepancies among the units and meaning of features (e.g., units of years vs. units of cholesterol), and simple standardization is no guarantee the resulting numbers are comparable. 2) Even after a distance metric is defined, there is no way for an unsupervised approach to know which features are relevant for an outcome of interest, and so should be weighted more strongly. Some applications might seek to cluster patients by groups relating to kidney disease, and another by groups relating to diabetes. But given the same feature set, unsupervised clustering will give the same results in both cases.

We can address both of the above problems in traditional unsupervised clustering by using local explanation embeddings to embed each sample into a new "explanation space." If we then run clustering in this new space, we will get a *supervised clustering* where samples are grouped together that have the same model output for the same reason. Since SHAP values have the same units as the model's output they are all comparable within a model, even if the original features were not comparable. Supervised clustering naturally accounts for the differing scales of different features, only highlighting changes that are relevant to a particular outcome. Running hierarchical supervised clustering using the mortality model results in groups of people that share a similar mortality risk for similar reasons (Figure 6A). The heatmap in Figure 6A uses the leaf order of a hierarchical agglomerative clustering based on the SHAP values of each sample. On the left are young people, in the middle are middle aged people, and on the right are older people. Within these broad categories many smaller groups are of particular interest, such as people with early onset high-blood pressure, young people with inflammation markers, and underweight older people. Each of these recapitulate known high risk groups of people, which would not have been captured by a simple unsupervised clustering approach (Supplementary Figure 12).

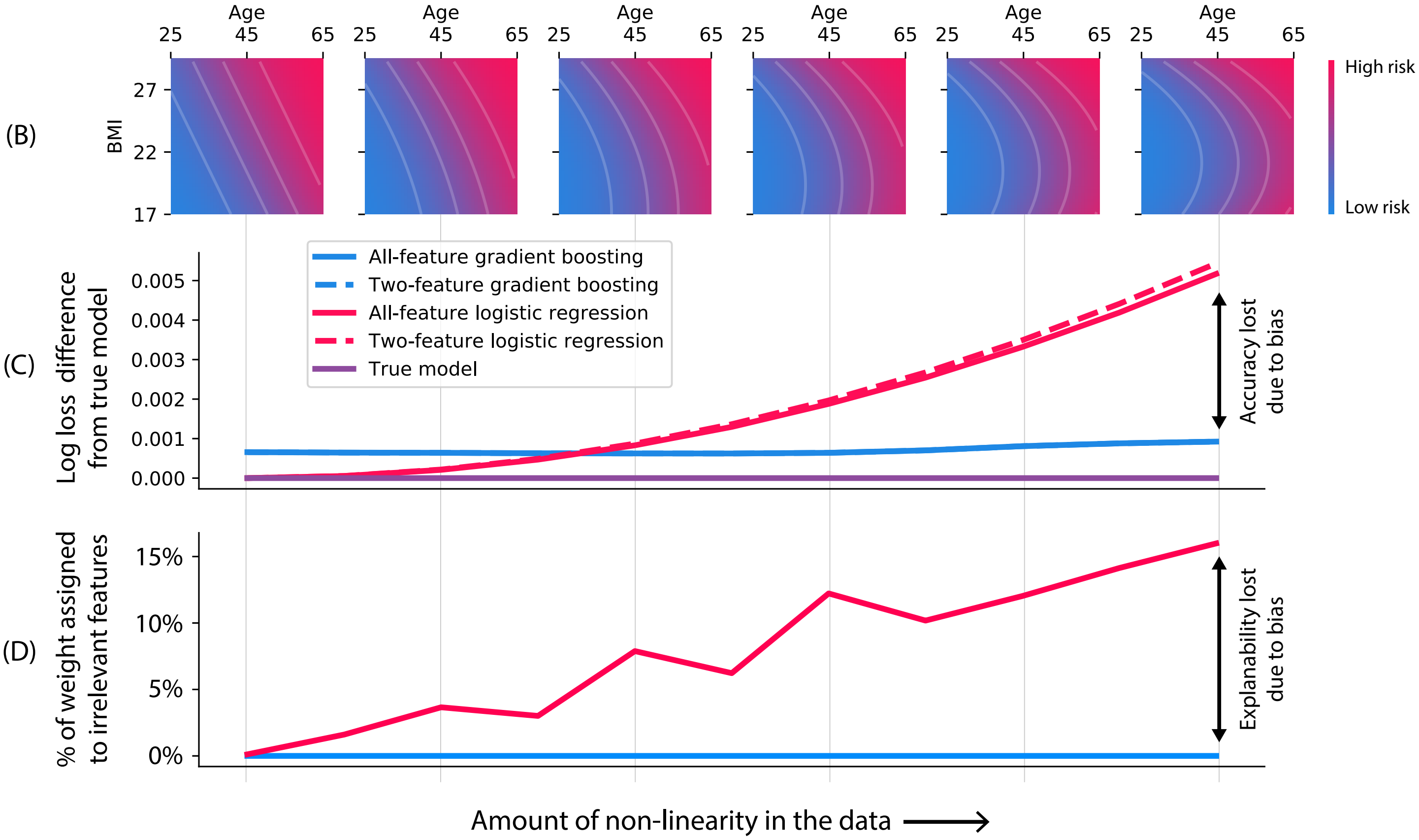
In addition to making explicit clusters, we can also use dimensionality reduction to directly visualize the explanation space embedding produced by the SHAP values. This gives a continuous representation of the primary directions of model output variation in the dataset. For the kidney disease dataset, the top two principal components of the explanation embedding highlight two distinct risk factors for disease progression

(Figures 6B-D). The first principal component aligns with blood creatinine levels, which are used to compute the estimated glomerular filtration rate (eGFR) of the kidneys. High levels of creatinine are a marker of lower eGFR and are the primary test for detection of kidney disease. The second principal component aligns with higher urine protein concentration in the urine. Quantified as the urine protein to urine creatinine ratio (PCR), this is a marker of kidney damage and is used in conjunction with eGFR to quantify levels of kidney disease. If we color the explanation space embedding by the risk of kidney disease progression, there is a roughly continuous increase in risk from left to right (Figure 6B). This is largely explained by a combination of the two orthogonal risk directions described above: One direction follows the blood creatinine level feature (which determine the eGFR) (Figure 6C) and the other direction follows the urine protein feature (Figure 6D). Several of the other top features in the chronic kidney disease model also align with these two orthogonal embedding directions (Supplementary Figure 6B-D). It is well established that eGFR and urine PCR are the strongest predictors of progression to end-stage renal disease among patients with chronic kidney disease [43, 13]. Physiologically, eGFR and PCR are likely complimentary, yet distinct kidney disease markers. Figure 6B-D shows that eGFR and PCR each identify unique individuals at risk of end-stage renal disease; thus confirming that clinically they should be measured in parallel. This type of insight into the overall structure of kidney risk is not at all apparent when just looking at a standard unsupervised embedding (Supplementary Figure 14).

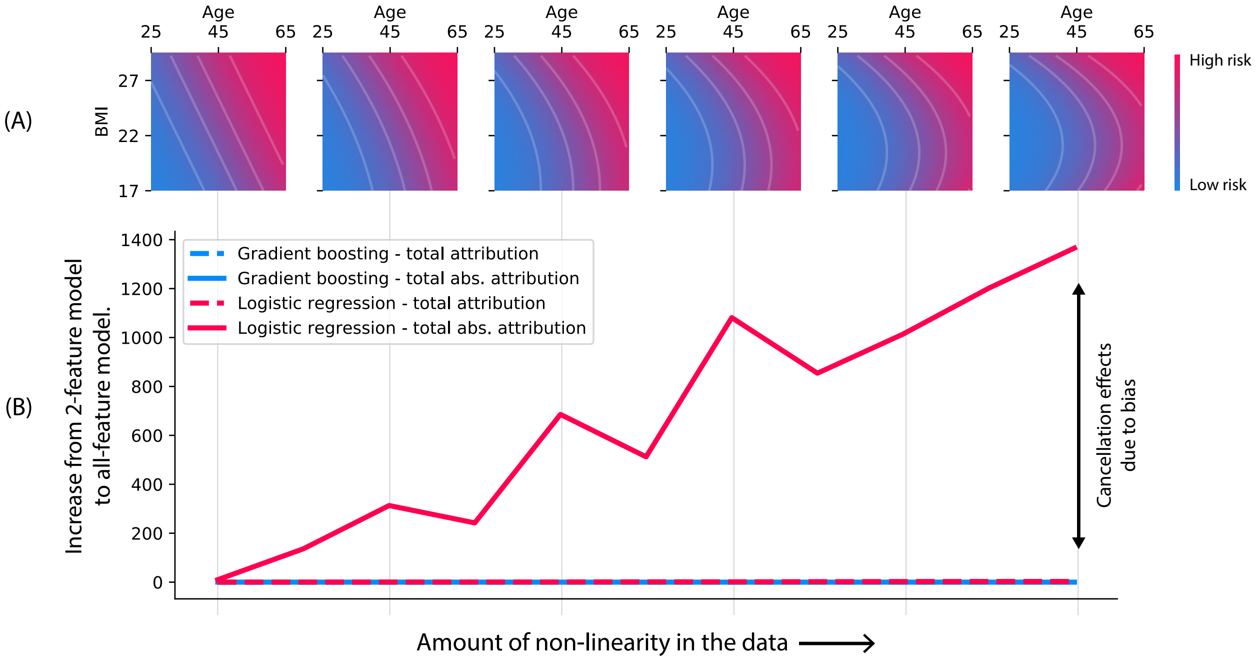
Supplementary Figures



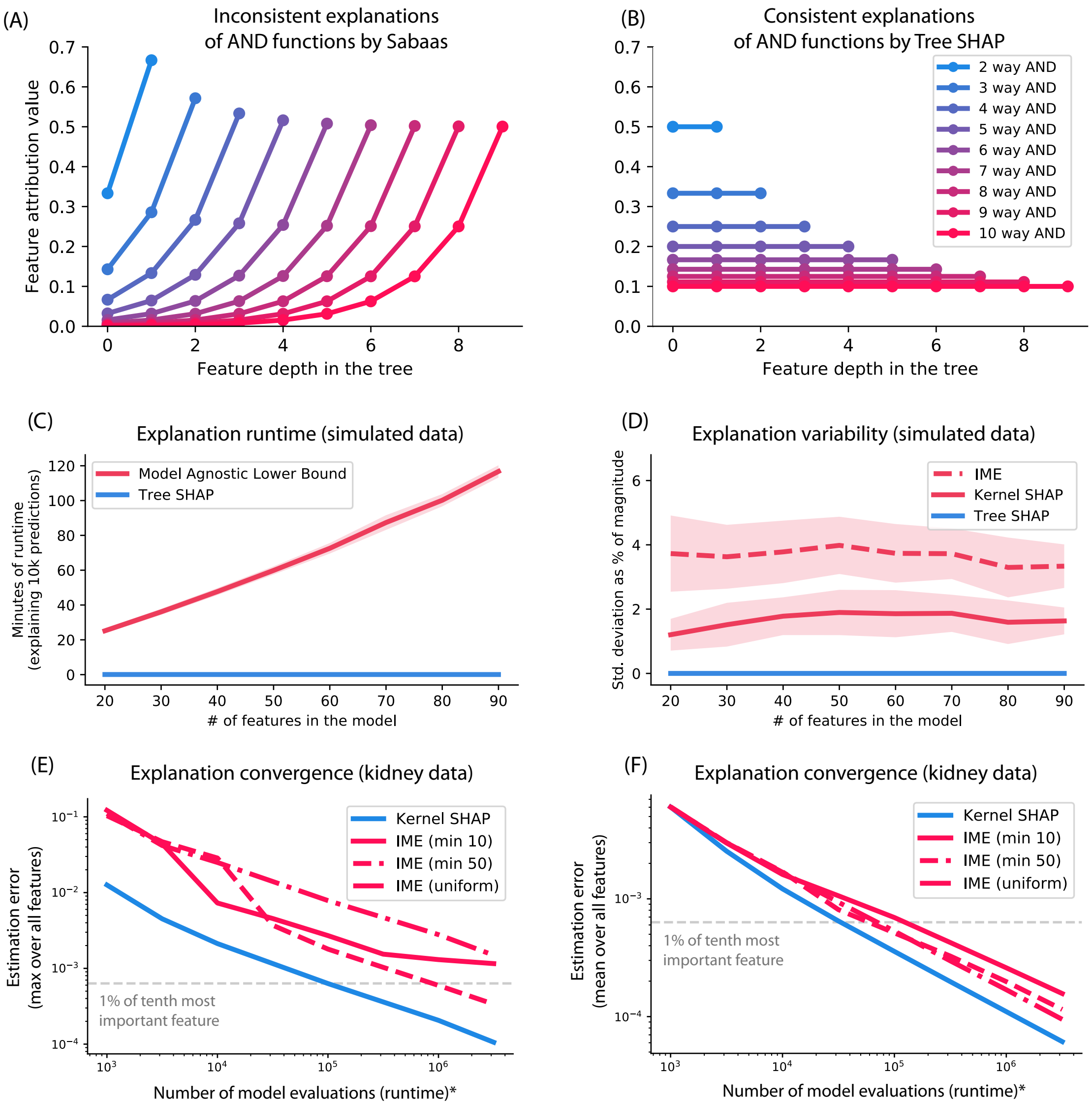
Supplementary Figure 1: **Poor weight allocation by linear models is driven by cancellation effects.** In the main text Figure 2C shows that as non-linearity increases in the true data generating distribution, a linear model begins to put weight on irrelevant features. This figure demonstrates that the weight placed on irrelevant features relies on complex cancellation effects, and is hence difficult to interpret. **(B)** For each model type from Figure 2 we show two things: 1) The increase in the total attribution from the 2-feature model to the all-feature model. If the total attribution changes then it means the output of the model is changing when we include the irrelevant features (because the attributions sum up to the model's output). For both gradient boosting and logistic regression, the model output does not change when we include all features. 2) The increase in the total absolute attribution from the 2-feature model to the all-feature model. If the total absolute attribution changes more than total attribution then that implies there are cancellation effects in the model. For gradient boosting there is no evidence of cancellation, whereas for linear logistic regression there is a significant amount of cancellation as the amount of non-linearity in the function grows.



Supplementary Figure 2: **The improved interpretability of gradient boosted tree models vs. linear models persists under regularization.** In Figure 2 it is shown that gradient boosted tree models can be more interpretable than linear models when the underlying data generating distribution is non-linear. This is because in the presence of non-linearity linear models tend to place a lot of weight on irrelevant features. This happens because these linear models are relying on complex cancellation effects to capture a small amount of the non-linear behavior. One potential way to reduce un-interpretable cancellation effects to use L1 regularization to encourage sparsity. This figure illustrates that even with optimal L1 penalization (chosen by cross validation), significant weight is still placed on irrelevant features, and this is due to complex cancellation effects (Supplementary Figure 3).

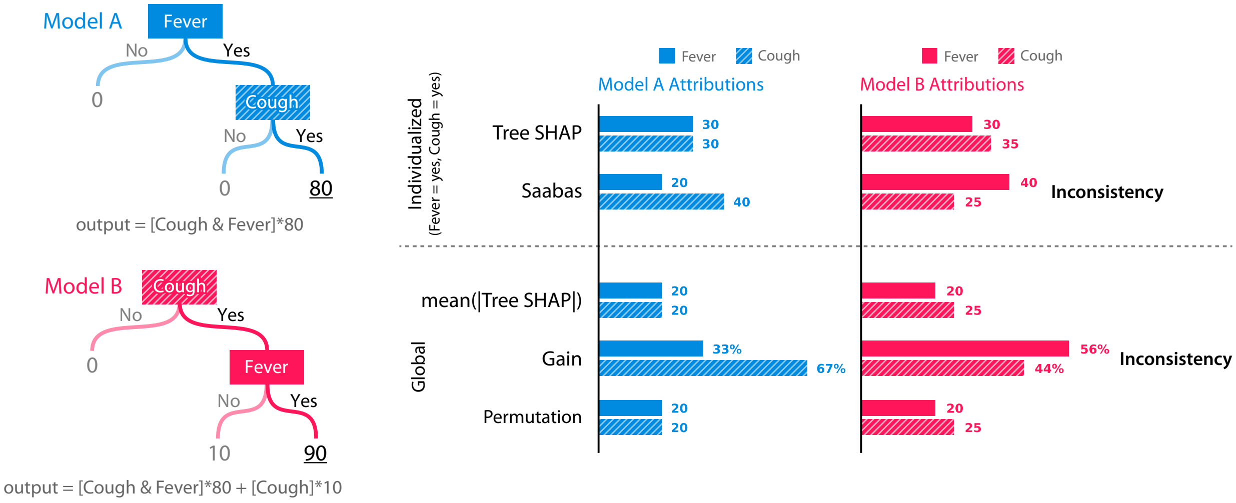


Supplementary Figure 3: **Poor weight allocation by linear models is driven by cancellation effects even under regularization.** This is figure is identical to Supplementary Figure 1, but instead of an unregularized linear model, the model is lasso regularized to encourage sparsity. The prevalence of cancellation effects is only slightly reduced by regularization.

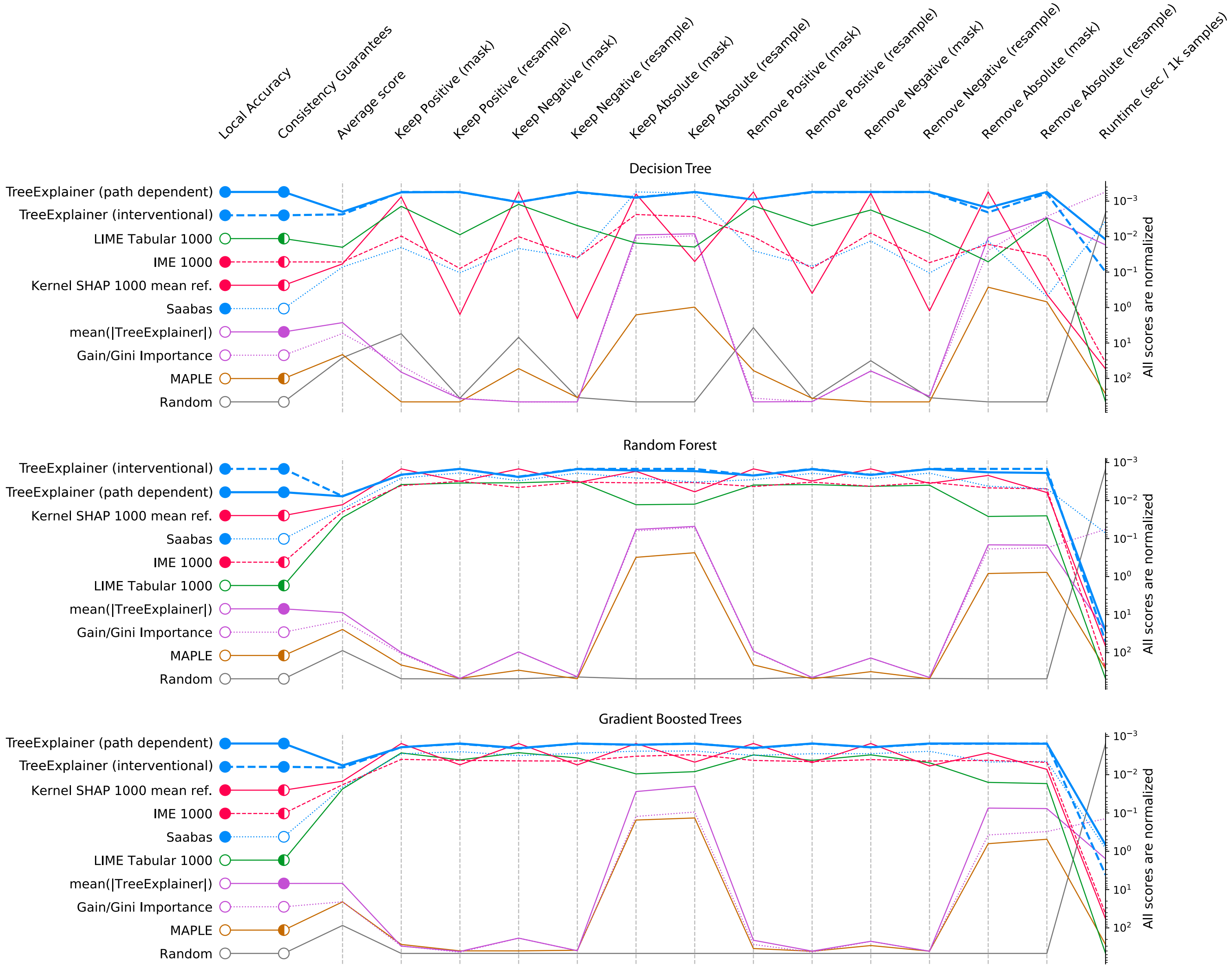


*Using 10^5 evaluations it takes 1.8 CPU days to explain the chronic kidney disease dataset with SHAP values.
With SHAP interaction values it would take 3.3 CPU years.

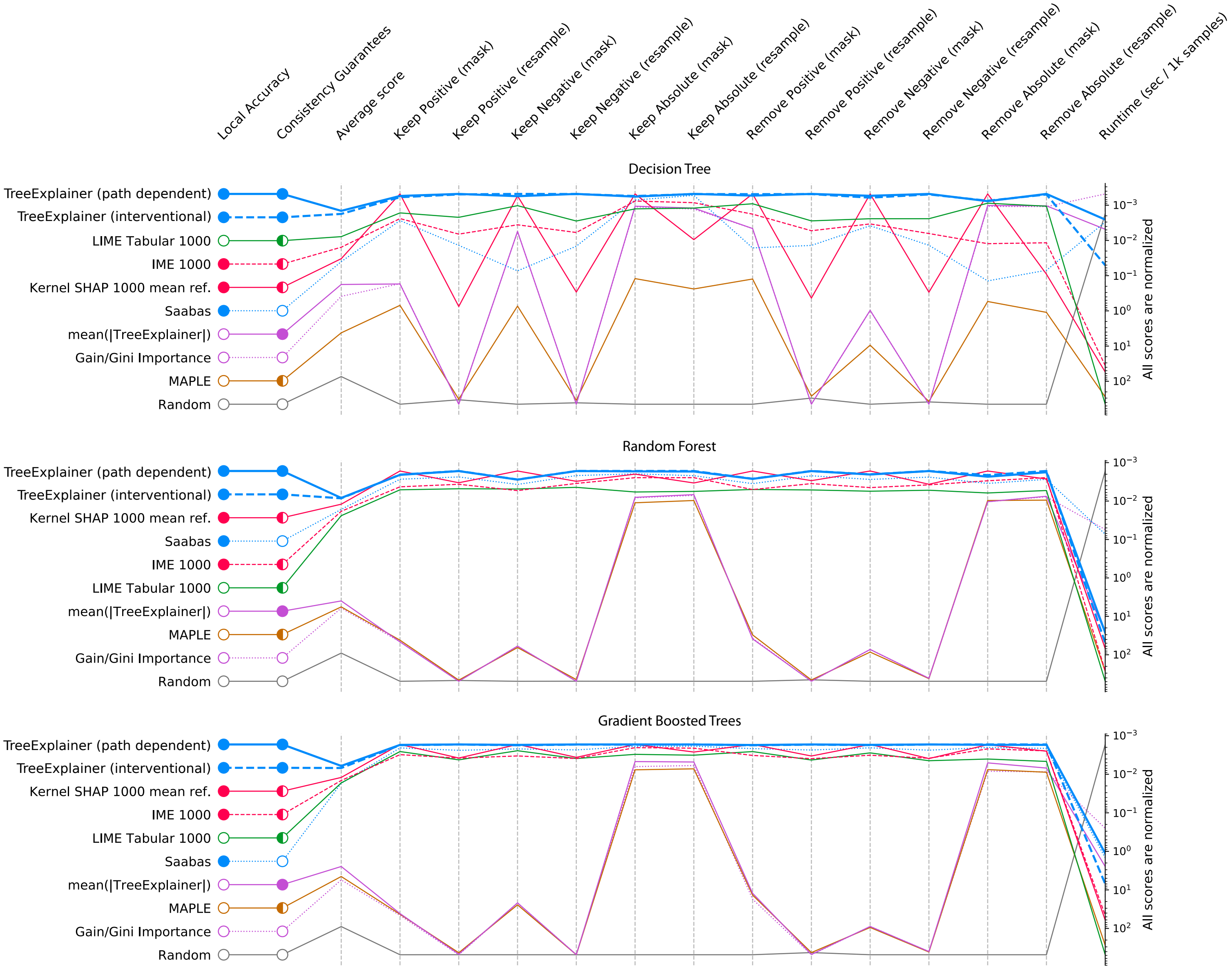
Supplementary Figure 4: **(A-B) TreeExplainer avoids the consistency problems of previous tree-specific approaches.** For tree models that represent a multi-way AND function the Saabas method gives little credit to features near the root, while Tree SHAP evenly distributes credit among all the features involved in the AND function. **(C-F) TreeExplainer represents a dramatic performance improvement over model agnostic approaches.** All model agnostic approaches rely on sampling, so their runtime is lower bounded by the run-time to evaluate the model, and they always have some sampling variability. TreeExplainer's Tree SHAP algorithm runs thousands of times faster, and has no sampling variability since it is exact. We consider both the Kernel SHAP [39] and IME [64] model agnostic estimators for Shapley values. (C-D) Using simulated data we can train XGBoost models over datasets of different sizes and observe that the number of samples required to maintain estimates with constant variance grows linearly with the number of features in the model. The reported runtime is a lower bound since it only accounts for the time to execute the model, not execute the explanation method itself. (E-F) As we increase the number of model evaluations used by model agnostic approaches we converge towards the exact solution. However, achieving low variance estimates requires days or years of CPU time even on the smallest of our medical datasets. We also only used a single background reference sample for computing conditional expectations in the model agnostic methods (instead of an entire dataset) so these represent lower bounds on the runtime. The estimation error on the y-axis represents the difference from the exact solution with a single background reference sample. Details of the experimental setup for C-F are described in 4.



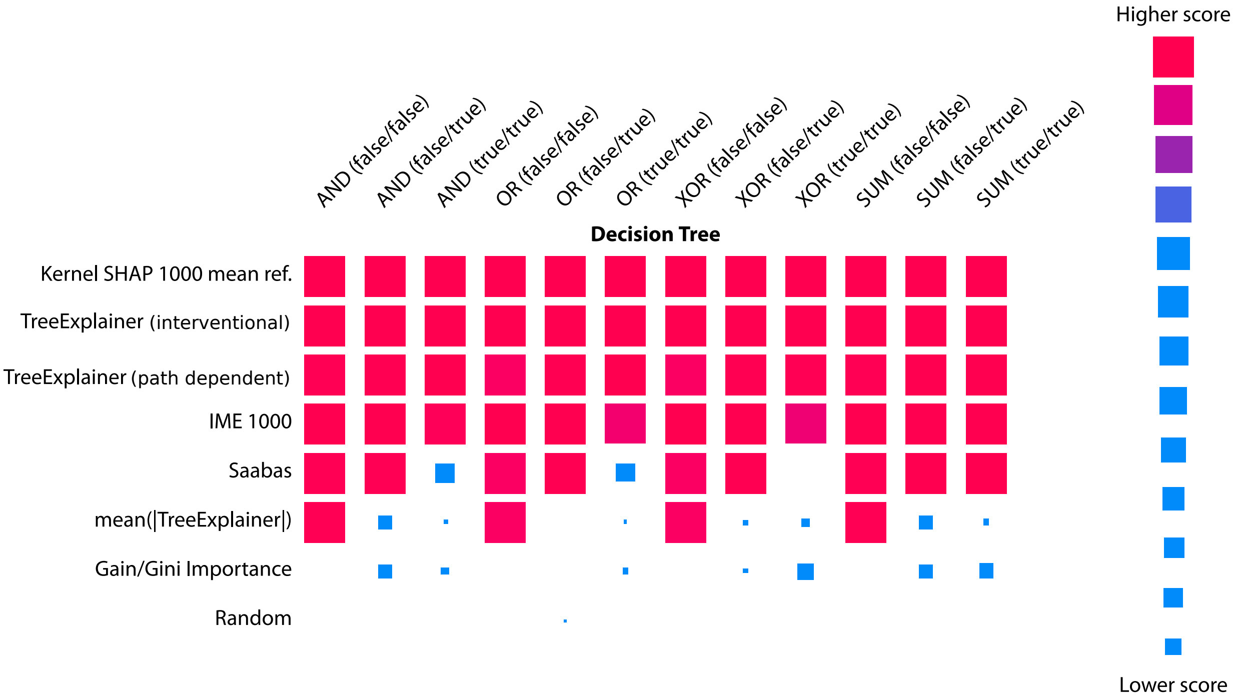
Supplementary Figure 5: **Two simple tree models that demonstrate inconsistencies in the Saabas and gain feature attribution methods.** The Cough feature has a larger impact in Model B than Model A, but is attributed less importance in Model B. Similarly, the Cough feature has a larger impact than Fever in Model B, yet is attributed less importance. The individualized attributions explain a single prediction of the model (when both Cough and Fever are Yes) by allocating the difference between the expected value of the model's output (20 for Model A, 25 for Model B) and the current output (80 for Model A, 90 for Model B). Inconsistency prevents the reliable comparison of feature attribution values. The global attributions represent the overall importance of a feature in the model. "Gain" is the most common way of measuring feature importance in trees and is the sum of the reductions in loss that come from all splits using that feature. "Permutation" is the change in the model's accuracy when a single feature is permuted.



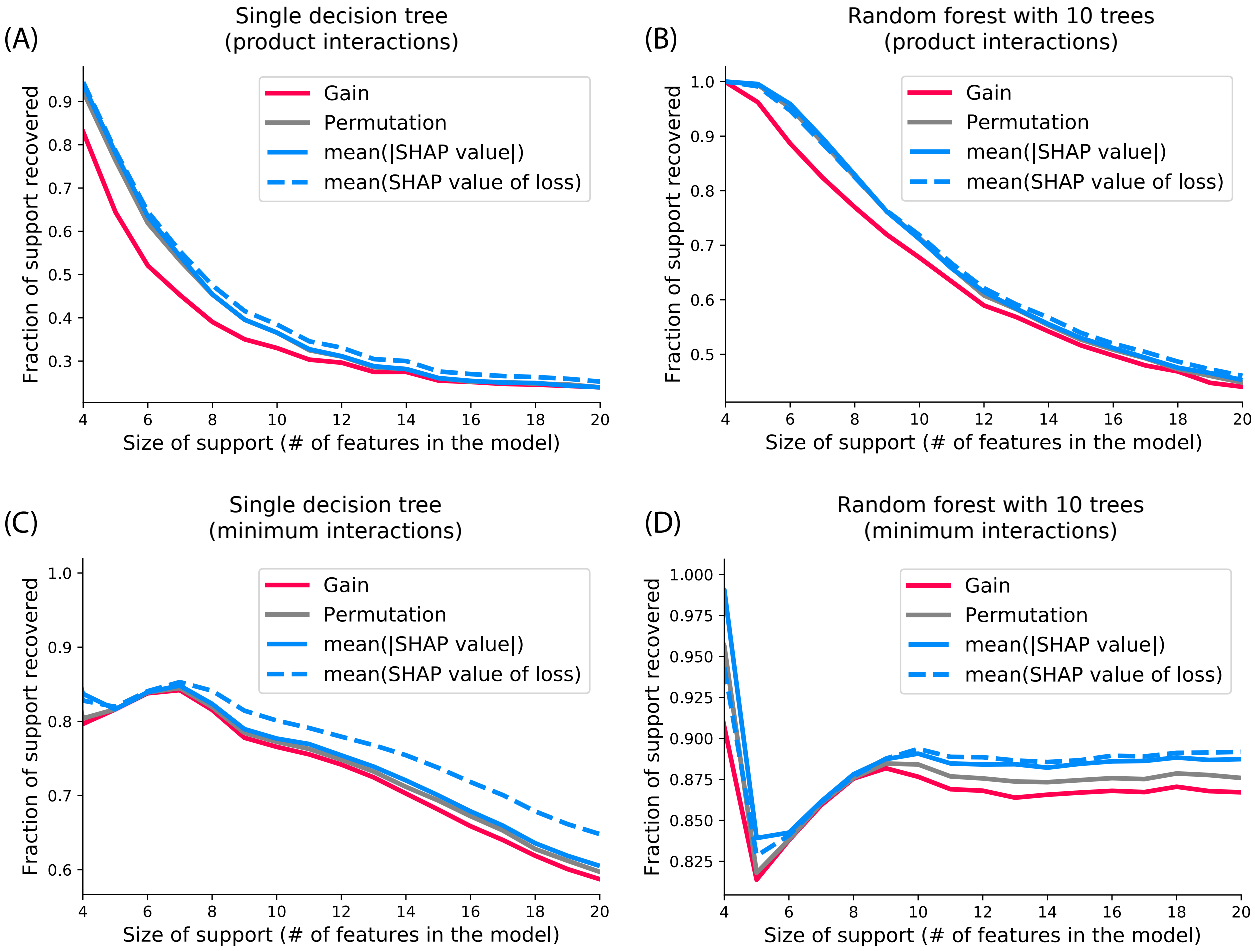
Supplementary Figure 6: **Explanation method performance across thirteen different evaluation metrics and three regression models in a simulated dataset with 60 features divided into tightly correlated groups.** Each tile represents the performance of a feature attribution method on a given metric for a given model. Within each model the columns of tiles are scaled between the minimum and maximum value, and methods are sorted by their overall performance. Some of these metrics have been proposed before and others are new quantitative measures of explanation performance that we are introducing (see Methods).



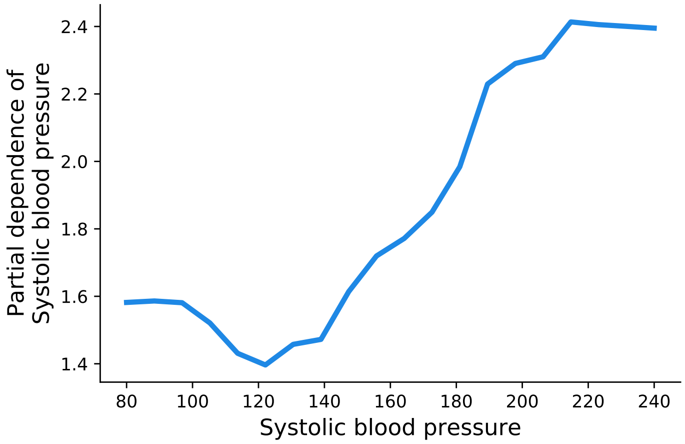
Supplementary Figure 7: **Explanation method performance across thirteen different evaluation metrics and three regression models in a simulated dataset with 60 independent features.** Each tile represents the performance of a feature attribution method on a given metric for a given model. Within each model the columns of tiles are scaled between the minimum and maximum value, and methods are sorted by their overall performance. Some of these metrics have been proposed before and others are new quantitative measures of explanation performance that we are introducing (see Methods).



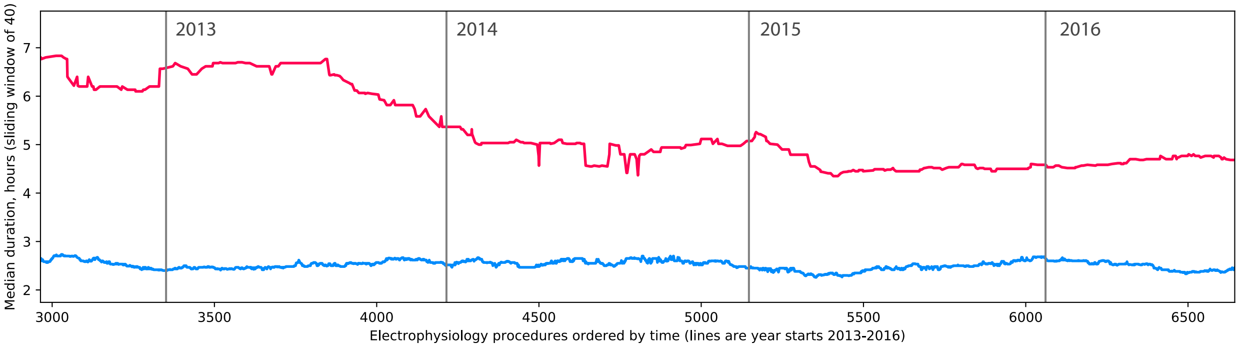
Supplementary Figure 8: **Shapley value based methods agree with human intuition.** We measured the most intuitive way to assign credit among input features by asking users to assign credit for predictions from four simple models (three predictions per model). The consensus allocation observed from a user study was then used as the ground truth and compared with the allocations given by different explanation methods. The sum of absolute differences was used to quantify agreement. All the Shapley value based methods had nearly perfect agreement across all the scenarios. The raw allocations for the cases where Saabas fails are shown in Supplementary Figures 37-39 (7). Note that since these small (human understandable) models have only three features, model agnostic Shapley methods are accurate and so comparable with TreeExplainer.



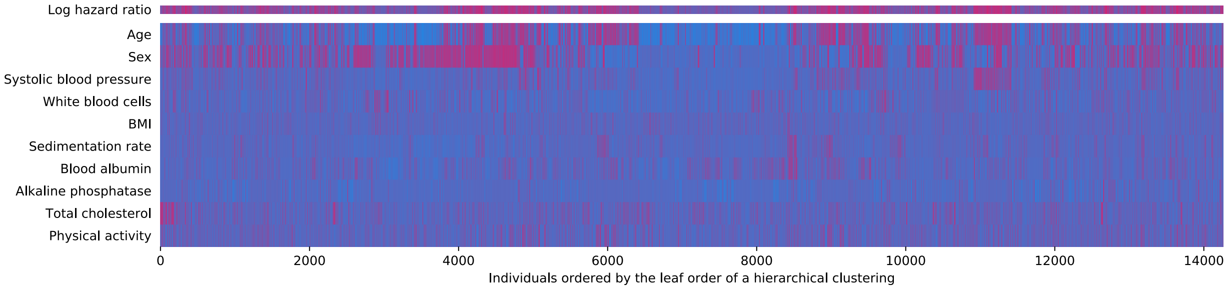
Supplementary Figure 9: **SHAP values can be combined to provide better feature selection power than traditional approaches.** Feature selection methods based on trees or ensembles of trees typically either use the total gain (reduction in train error when splitting) or a permutation test (scramble each feature and observe the change in error) to do feature selection. To compare feature selection power we reproduce the same simulated independent features setting as [30] (but with strong 3rd order interactions) and compare the mean SHAP value of the loss vs. mean SHAP value magnitude vs. gain vs. permutation for feature selection. We also repeat the experiment replacing the product style interactions from [30] with minimum functions. For both a single tree (A,C) and an ensemble of ten trees (B,D) the SHAP values provide a better ranking of features. Perhaps because, unlike gain, they guarantee consistency as defined by Property 2, and unlike permutations, they account for high-order interaction effects. The x-axis of the plots represents the number of features used in the true model (out of 200 total features), while the y-axis represents the fraction of those true features recovered in the set of top ranked features of the same size. Results are averages over performance on 1000 simulated datasets. Both SHAP value based methods outperform gain and permutation testing in every figure, with all paired t-test P-values being $< 10^{-7}$.



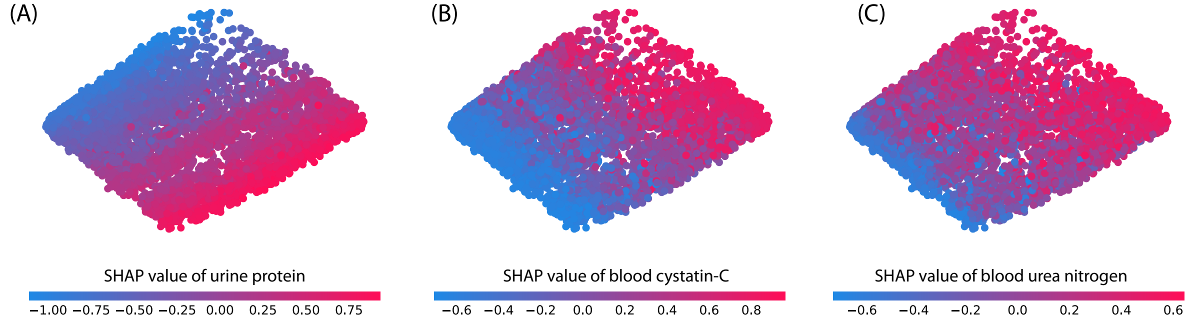
Supplementary Figure 10: **Partial dependence plot of systolic blood pressure in the mortality model.** Unlike the corresponding SHAP dependence plot in Figure 4B, the partial dependence plot gives no indication of the heterogeneity between individuals caused by interaction effects in the model.



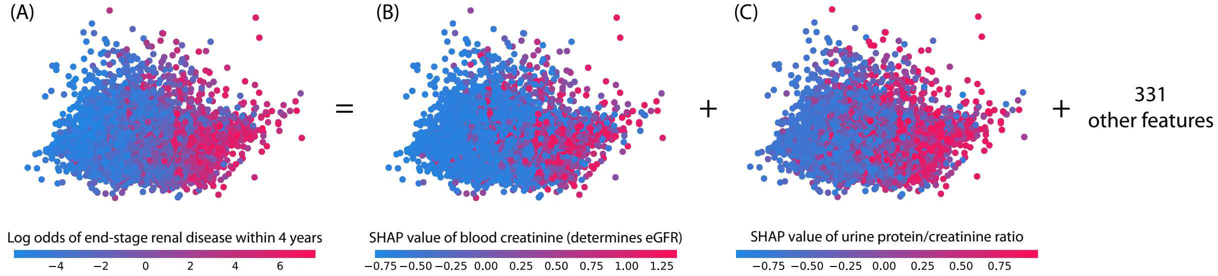
Supplementary Figure 11: **The average duration of ablation procedures for atrial fibrillation dropped significantly during 2014.** This data was obtained directly from the electrophysiology lab to diagnose why the atrial fibrillation feature was found (using a SHAP monitoring plot) to degrade model performance (Figure 5D). The reason is that around 2014 (which was after the simulated model deployment) the duration of these procedures dropped significantly.



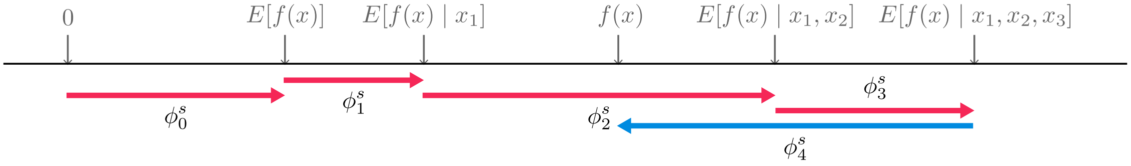
Supplementary Figure 12: **A clustering of 14,264 mortality study individuals by their normalized data values.** Standard complete linkage hierarchical clustering reveals fewer groups consistency with model risk than supervised clustering (Figure 6A). This is because unsupervised clustering has no bias towards clusters that share common risk characteristics. Row-normalized feature SHAP values are used for coloring, as in Figure 6A.



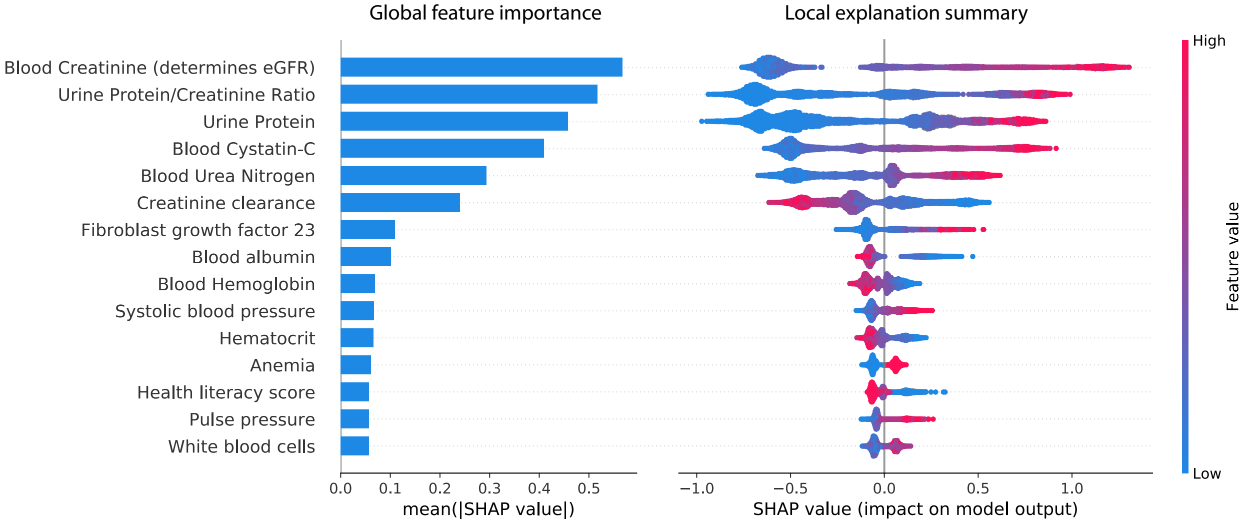
Supplementary Figure 13: **A local explanation embedding of kidney visits projected onto its first two principle components.** This shows the next three top features beyond those shown in Figures 6B-D. The fact that these features also align with the top principal components shows how many of the important features in the data set are capturing information along two largely orthogonal dimensions.



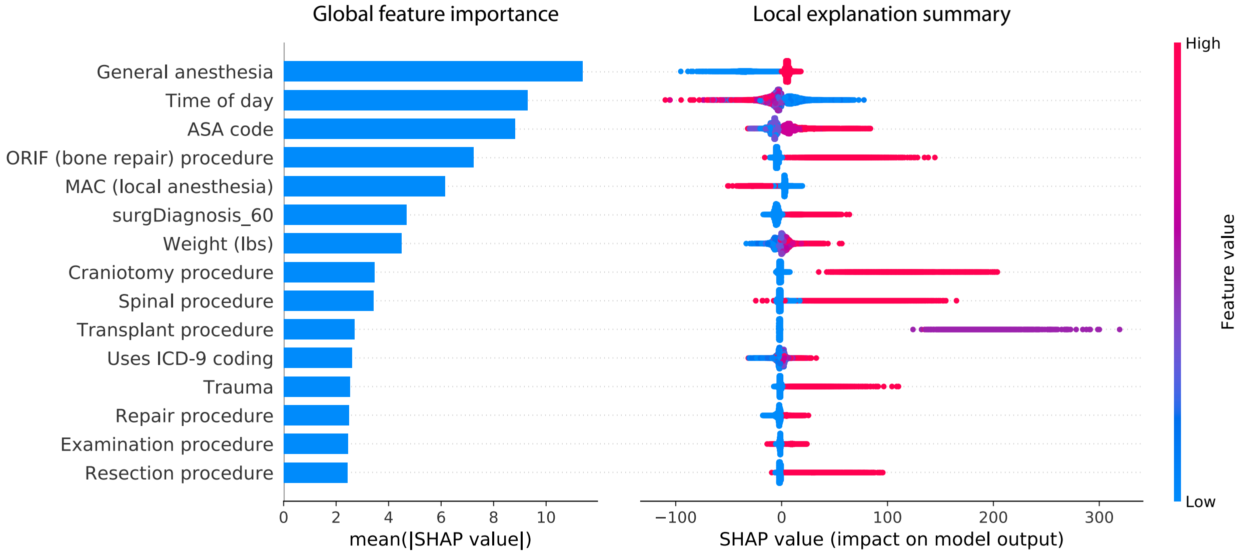
Supplementary Figure 14: **Principle component embedding of the chronic kidney disease dataset.** Unlike an embedding based in the local explanation space (Figures 6B-D), an unsupervised embedding of the data does not necessarily align with the outcome of interest in a dataset.



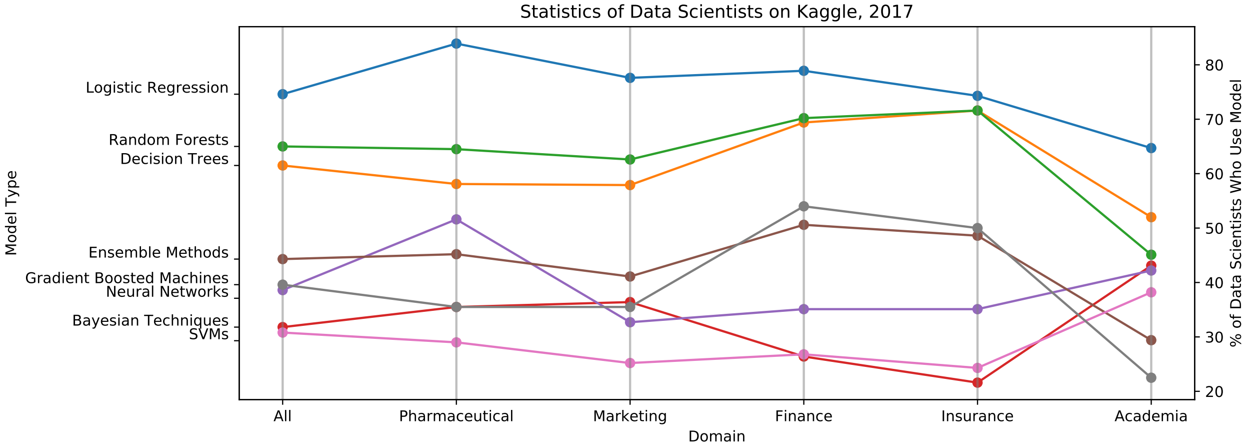
Supplementary Figure 15: The Sabaas values, ϕ_i^s, attribute feature importance by measuring differences in conditional expectations along the order defined by the decision path. This is very similar to SHAP (SHapley Additive exPlanation) values, ϕ_i, except SHAP values result from averaging over all possible orderings and use interventional conditional expectations $E[f(X) \mid do(X_s = x_s)]$ [27]. Averaging over all possible orderings is important since for non-linear functions the order in which features are introduced matters. Proofs from game theory show that averaging over all orderings is the only possible consistent approach where $\sum_{i=0}^{M} \phi_i = f(x)$ ().



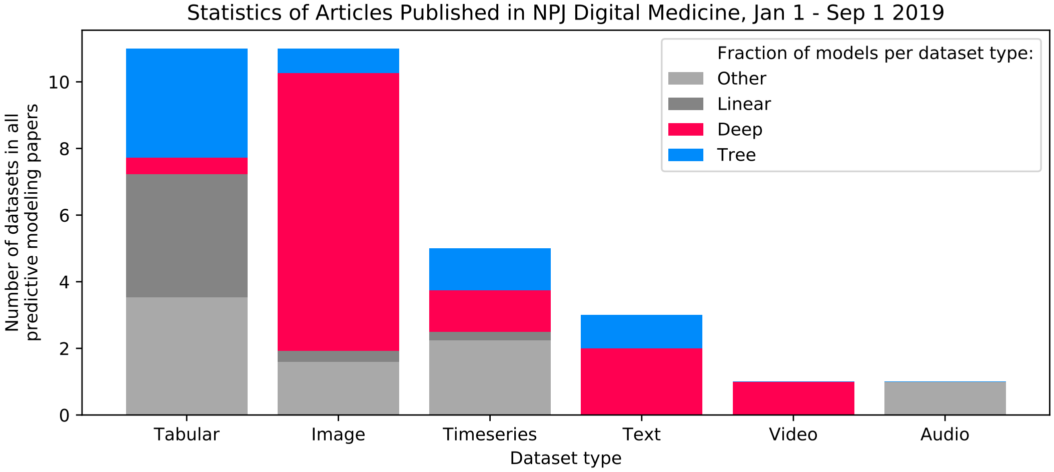
Supplementary Figure 16: **Bar chart (left) and summary plot (right) for a gradient boosted decision tree model trained on the chronic kidney disease data.** For the summary plot red dots indicate a high value of that feature for that individual, while blue dots represent a low feature value. The x-axis is the SHAP value of a feature for each individual's prediction, representing the change in the log-hazard ratio caused by observing that feature. High blood creatinine increases the risk of end-stage kidney disease. Conversely, low creatinine clearance increases the risk of end-stage kidney disease.



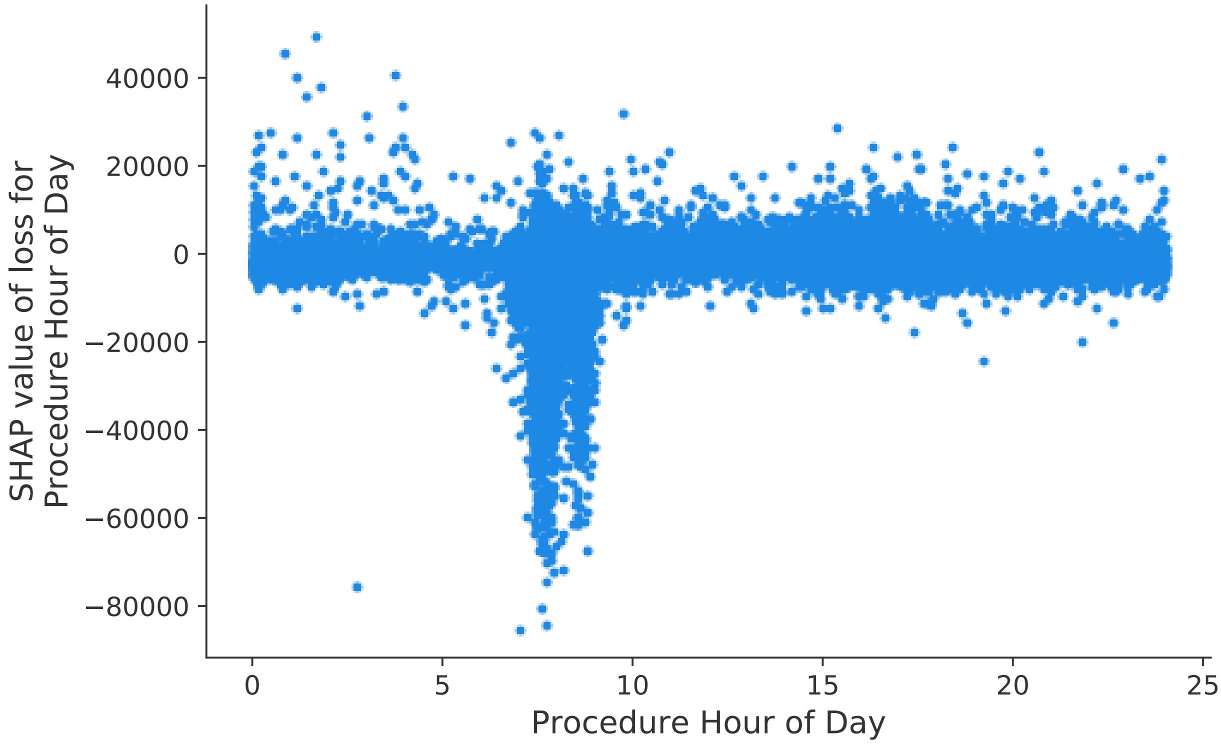
Supplementary Figure 17: **Bar chart (left) and summary plot (right) for a gradient boosted decision tree model trained on the hospital procedure duration data.** For the summary plot red dots indicate a high value of that feature for that individual, while blue dots represent a low feature value. The x-axis is the SHAP value of a feature for each individual's prediction, representing the change in the log-hazard ratio caused by observing that feature. Many of the features are bag-of-words counts that have only a few non-zero values. Because the model is very nonlinear, the impact of a flag being on (such as the trauma flag) can have very different effects for different procedures (as shown for trauma by the horizontal dispersion of the red dots).



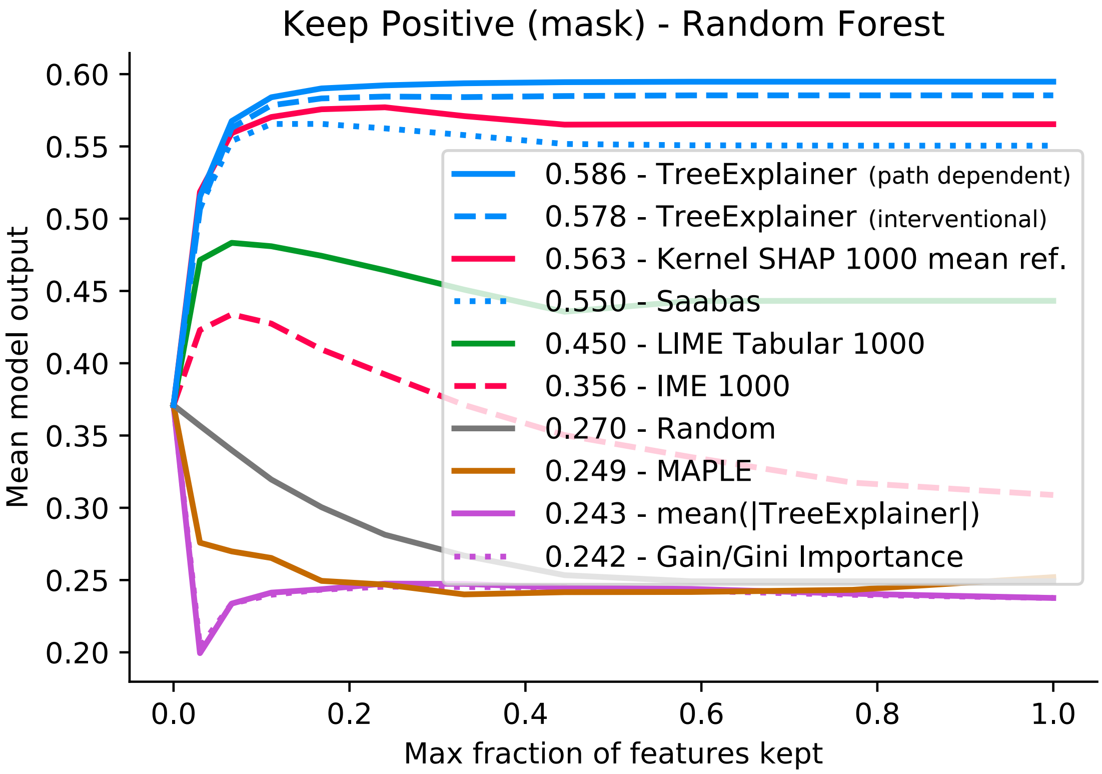
Supplementary Figure 18: **Statistics from Kaggle's "State of ML and Data Science" survey in 2017.** Tree-based models are the most popular non-linear models by a wide margin across many domains, including the pharmaceutical industry. Percentages indicate what percent of self-identified "data scientists" in a given field use a given method. Since data scientists may use many different methods, the percentages do not add to 100.



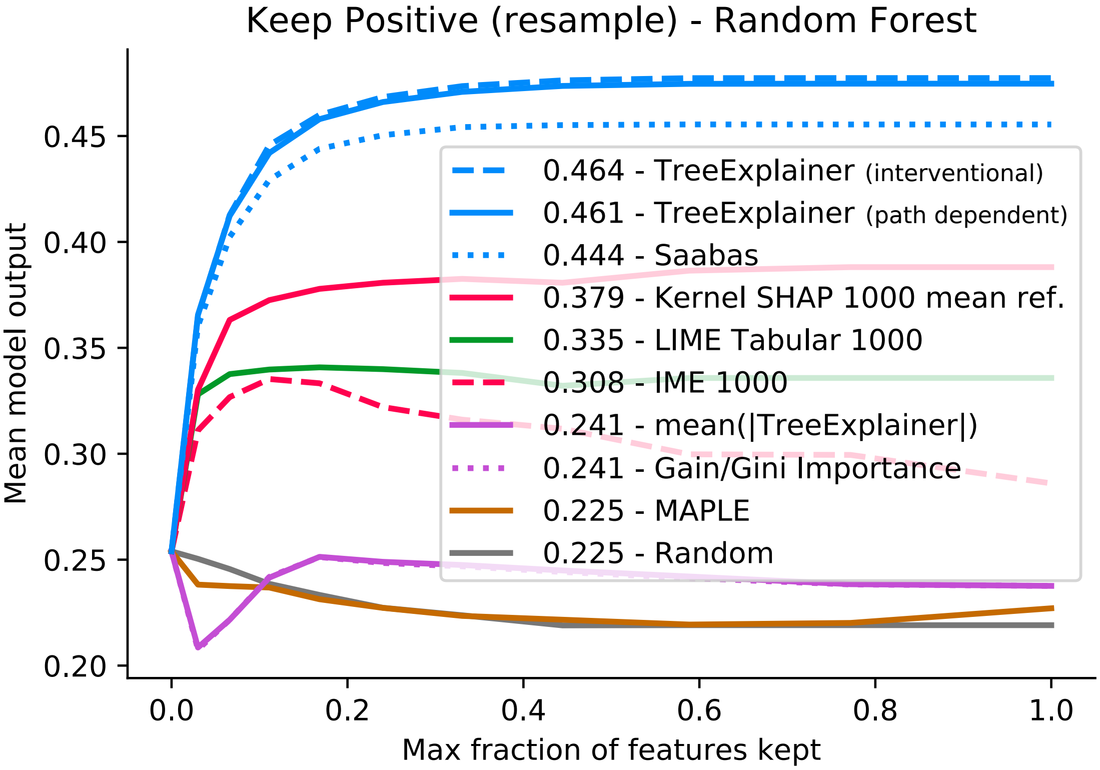
Supplementary Figure 19: **Prevalence of different model types among articles published in *npj Digital Medicine*.** For articles that involve prediction tasks, tabular and image data are most prevalent. Trees are the most popular non-linear models for tabular data, while deep models are the most popular for images. The height of each bar is the total number of papers using a particular dataset type. The shaded area for a particular model type represents the proportion of papers in a category that used that model type, normalized so that; for example, models from a paper using five model types only count as one-fifth as much as models from a paper that uses a single model.



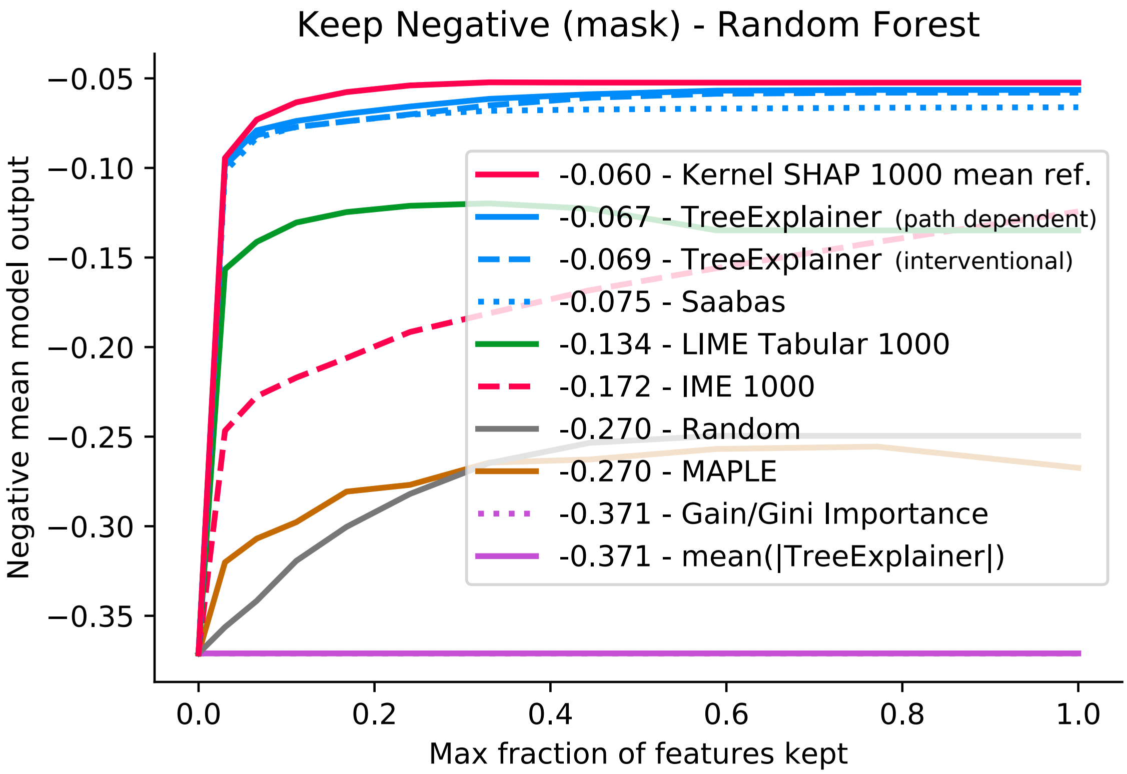
Supplementary Figure 20: **A dependence plot for time of day vs the SHAP value of time of day with respect to the model loss.** In our hospital system long running elective surgeries are scheduled at 7:20-7:30 AM on Monday/Tuesday/Thursday/Friday and at 8:20-8:30 AM on Wednesday. This plot shows that the primary way model is using time of day to reduce the model's loss is by detecting these surgery scheduling times. Each dot is a procedure, the x-axis is the time that procedure was scheduled to begin, and the y-axis is the impact knowing the time of day had on the model's loss for predicting that procedure's duration (lower is better).



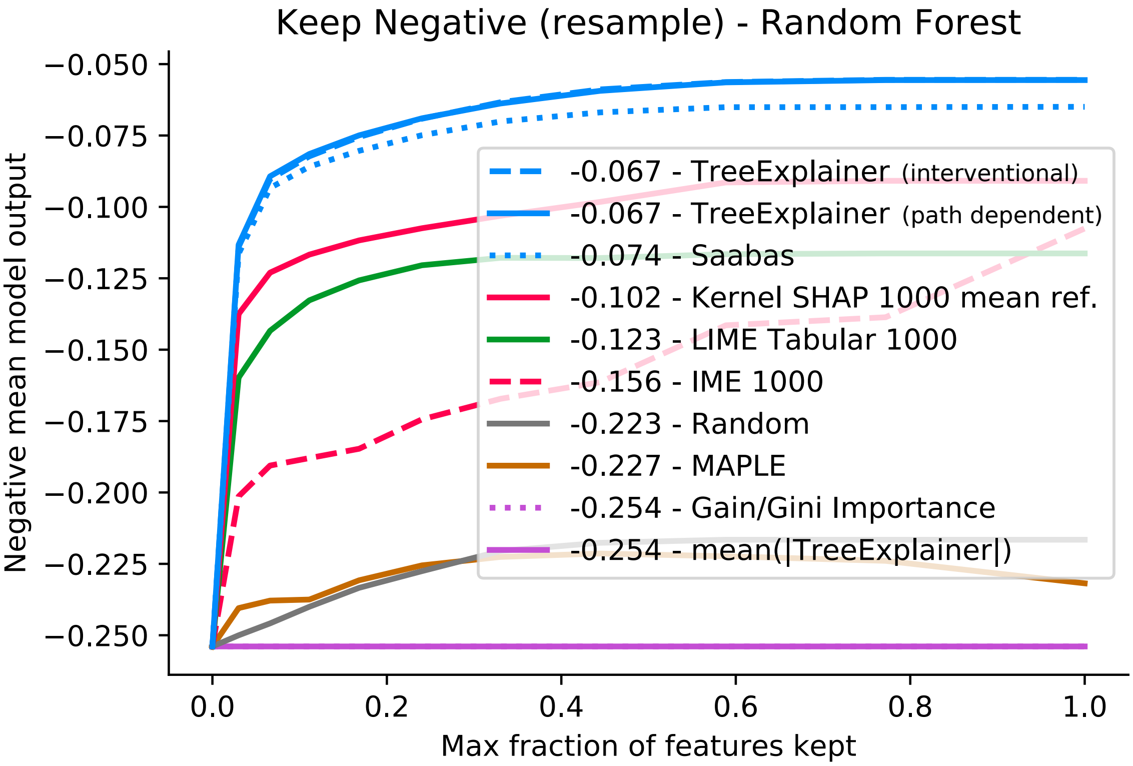
Supplementary Figure 21: **Keep positive (mask) metric for a random forest trained on the chronic kidney disease dataset.** Sorting the attribution values of an explanation method provides an ordering of the features for each prediction made by the model. Here we keep a fraction of the features ordered by how much they increase the model's output. Features that are not kept are masked with their mean value. If the ordering is good, as we include more and more features we push the model's output higher. Note that features with a negative contribution are always masked. The x-axis is the maximum fraction of features kept and the y-axis is the mean increase of the model over 100 predictions (averaged over 10 model's trained on different train/test splits of the dataset). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



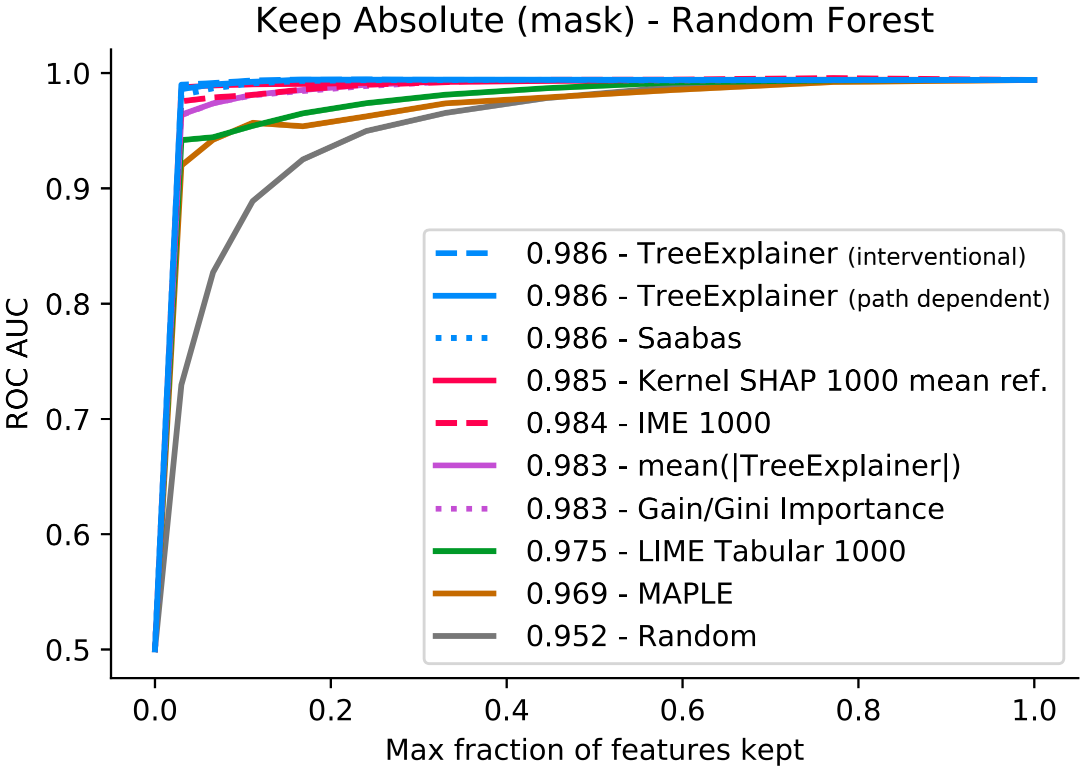
Supplementary Figure 22: **Keep positive (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 21 except that instead of masking the hidden features with their mean value, we instead replace them with a random sample from the training dataset. This resampling process is averaged over 100 times to integrate over the distribution of background samples. If the input features are independent of one another then this effectively computes the conditional expectation of the model output conditioned on only the observed features. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



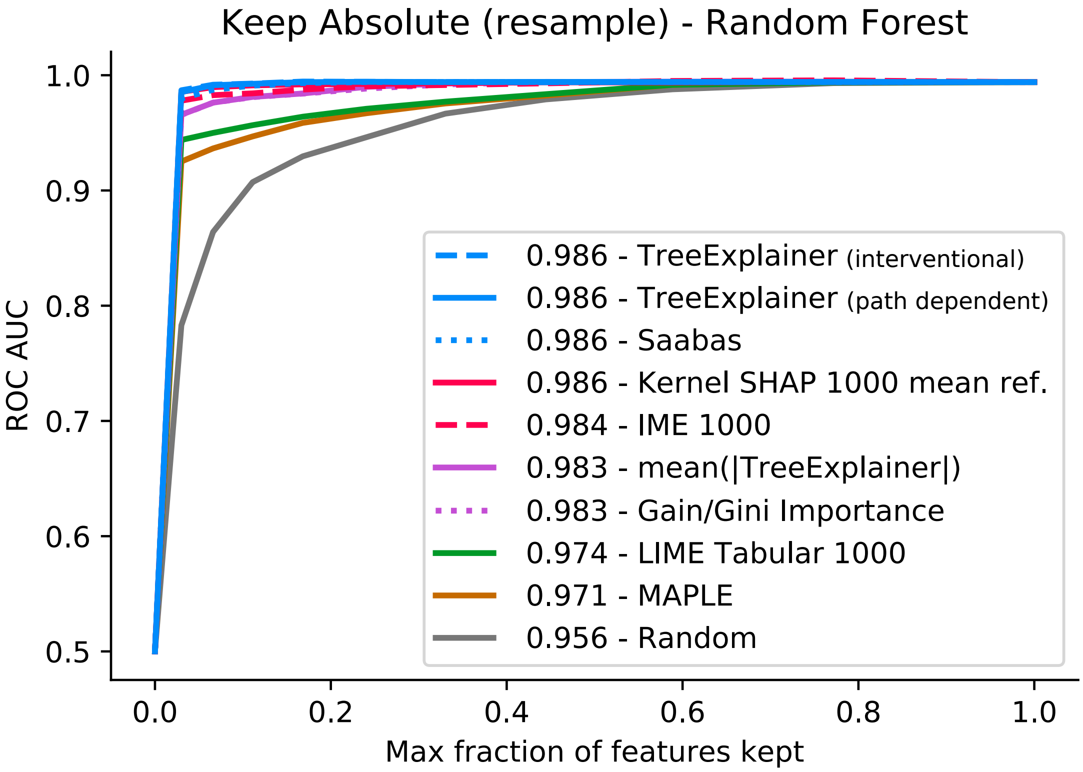
Supplementary Figure 23: **Keep negative (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 21 except that we keep the most negative features instead of the most positive. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



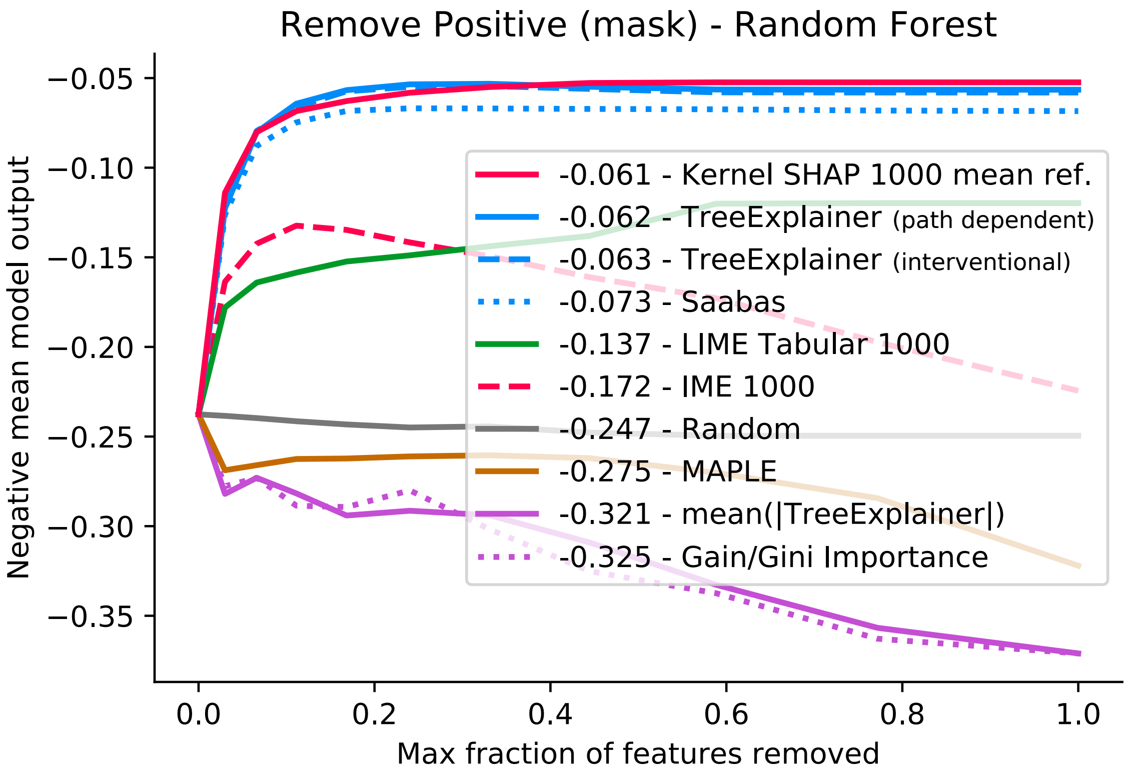
Supplementary Figure 24: **Keep negative (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 22 except that we keep the most negative features instead of the most positive. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



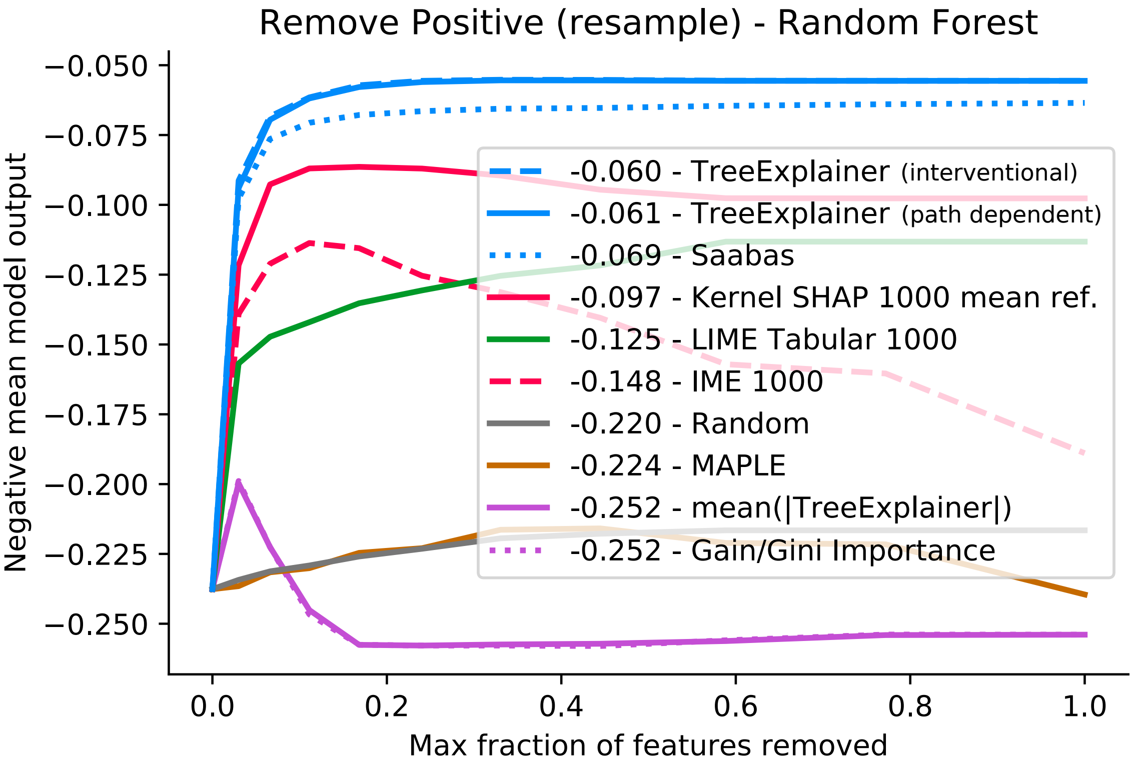
Supplementary Figure 25: **Keep absolute (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures 21 and 23 except that we keep the most important features by absolute value instead of the most positive or negative. Since this no longer specifically pushes the model output higher or lower, we instead measure the accuracy of the model. Good attribution methods will identify important features that when kept will result in better model accuracy, measured in this case by the area under the receiver operating characteristic (ROC) curve. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



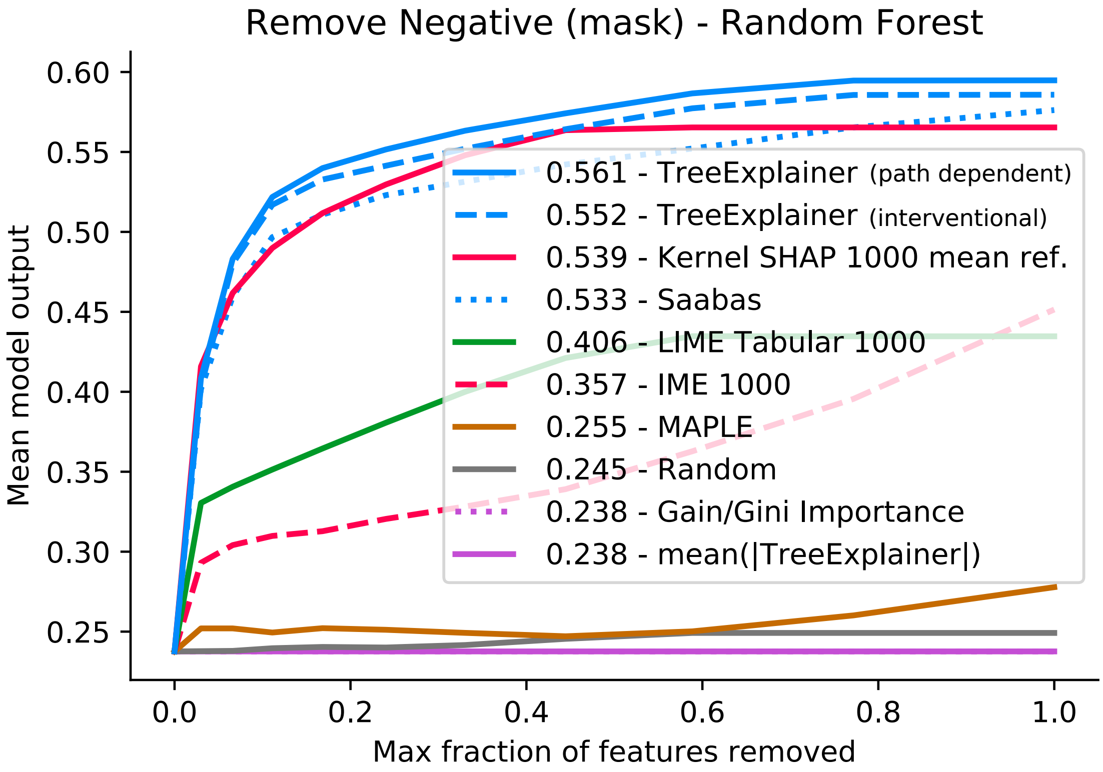
Supplementary Figure 26: **Keep absolute (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures 22 and 24 except that we keep the most important features by absolute value instead of the most positive or negative (as in Figure 25). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



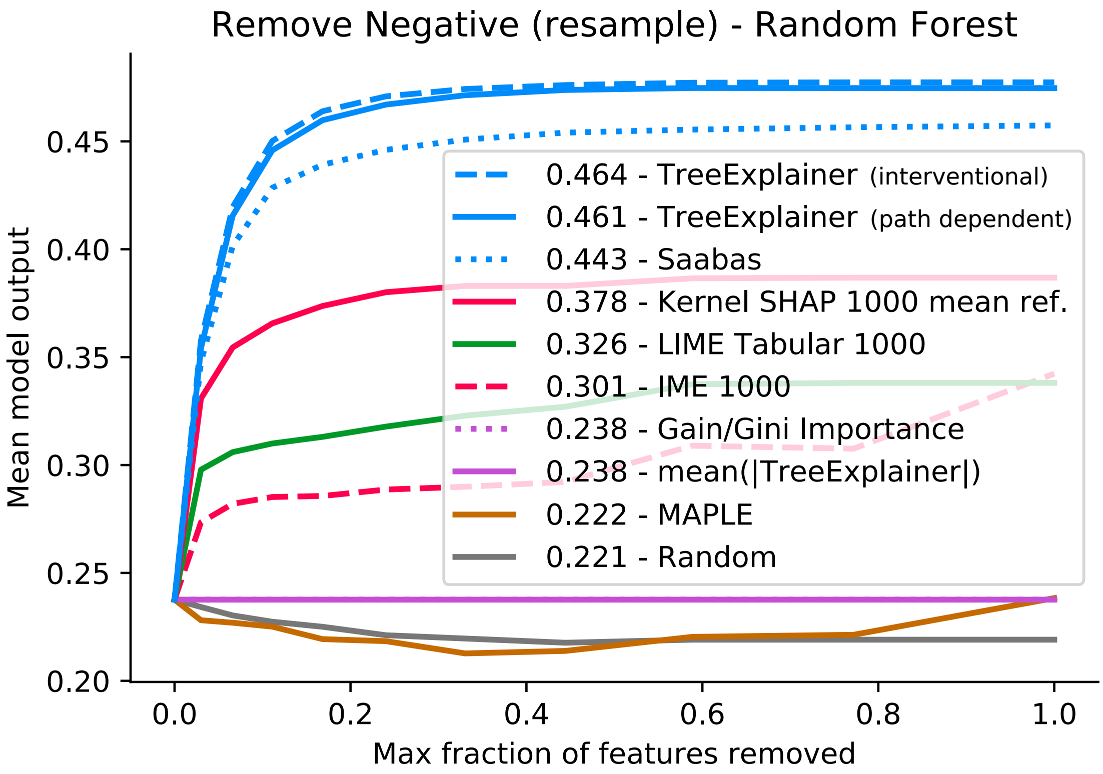
Supplementary Figure 27: **Remove positive (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 21 except that we remove the most positive features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



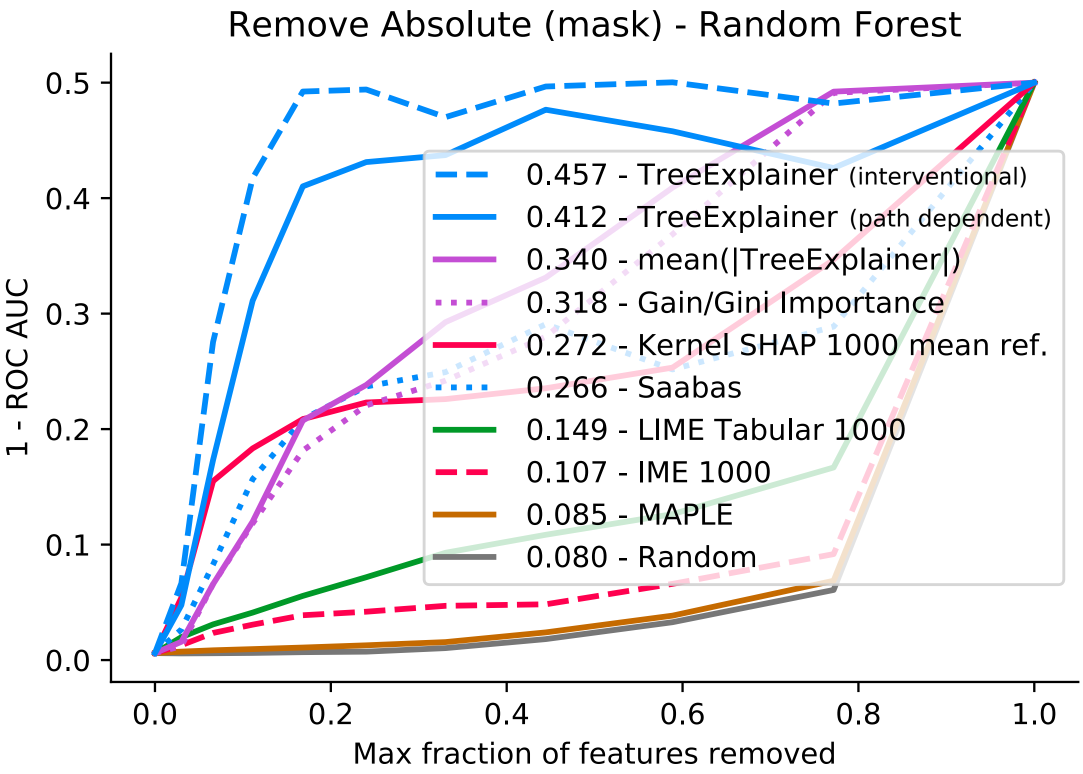
Supplementary Figure 28: **Remove positive (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 22 except that we remove the most positive features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



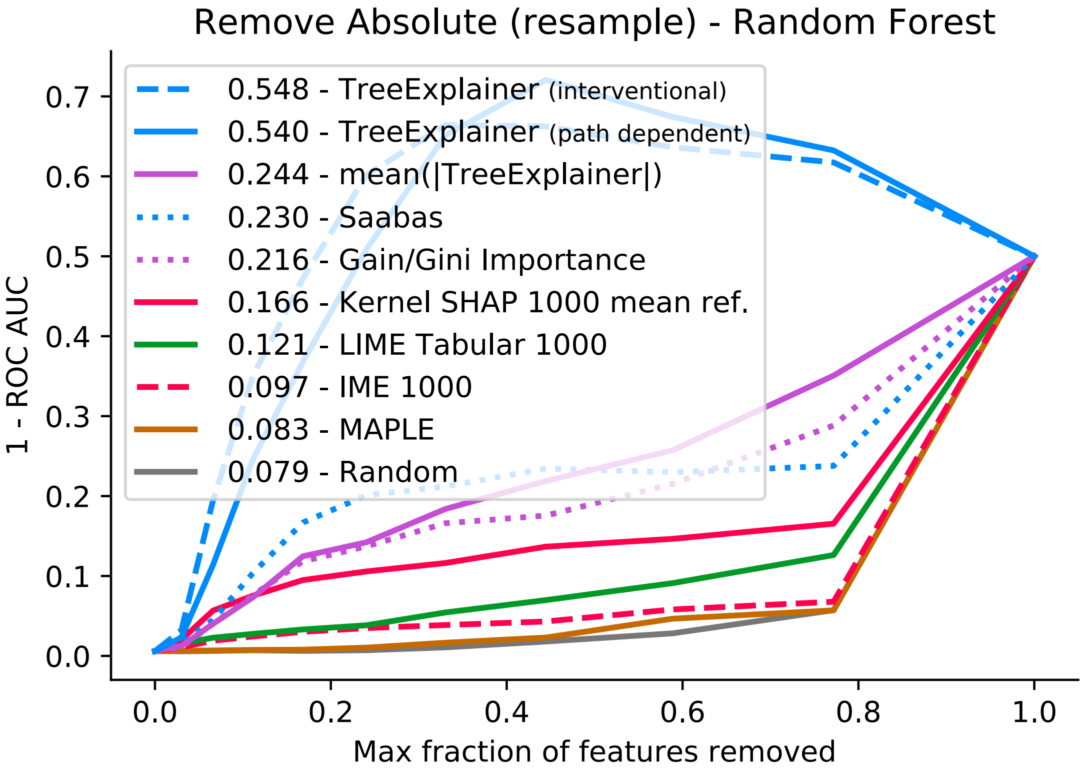
Supplementary Figure 29: **Remove negative (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 23 except that we remove the most negative features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



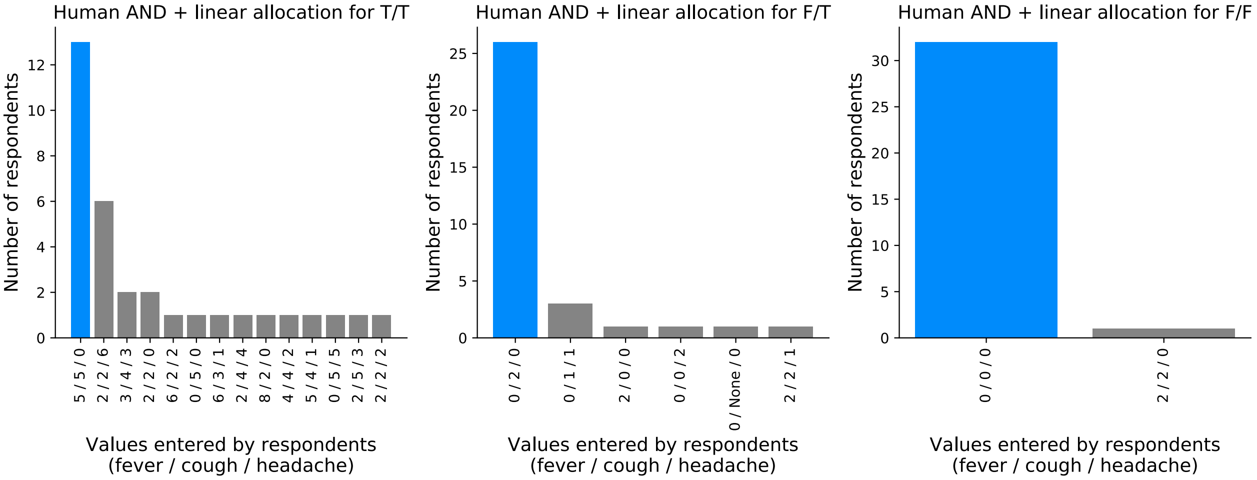
Supplementary Figure 30: **Remove negative (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 24 except that we remove the most negative features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



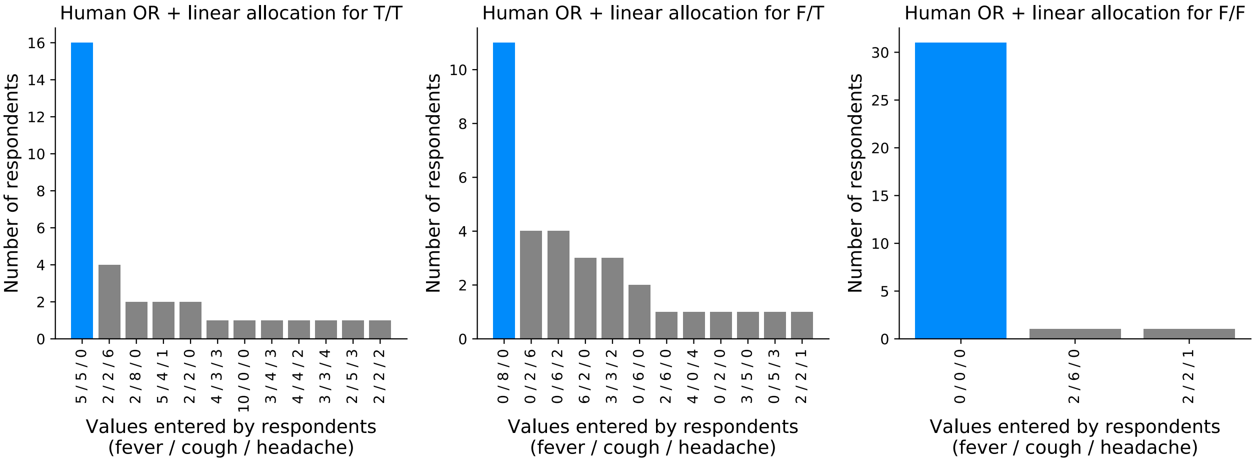
Supplementary Figure 31: **Remove absolute (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 25 except that we remove the most important features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



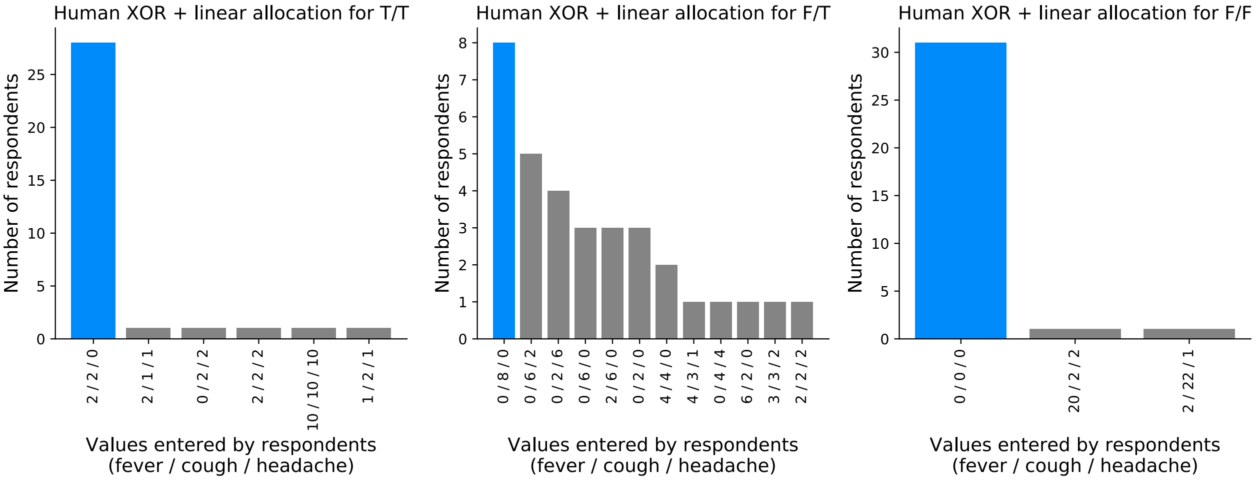
Supplementary Figure 32: **Remove absolute (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure 26 except that we remove the most important features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [64], respectively.



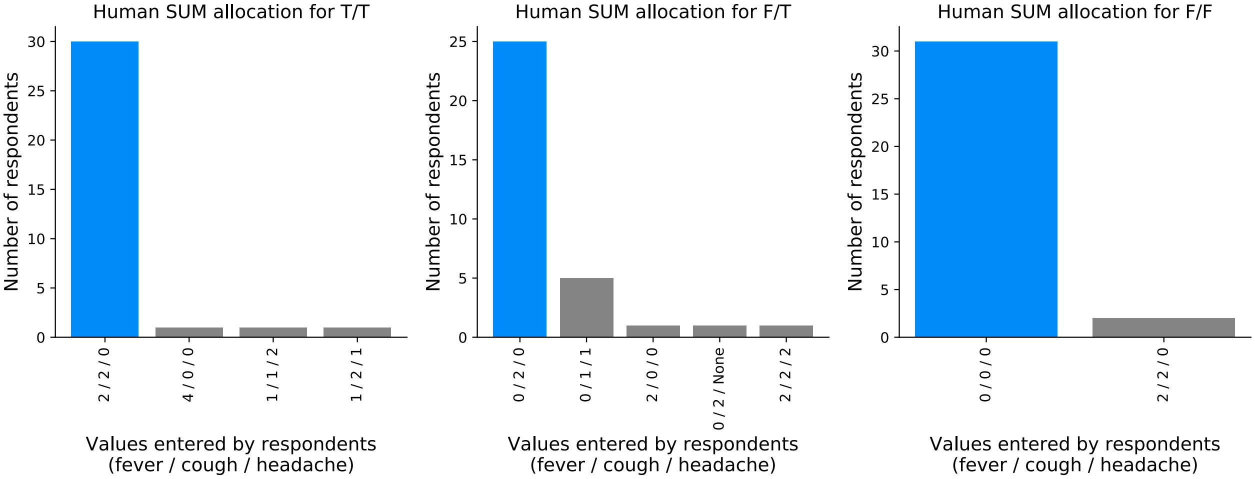
Supplementary Figure 33: **Consensus human intuition values for an AND function.** Human consensus values were measured by a user study over 33 participants for a simple AND-based function. The most popular allocation was chosen as the consensus (7). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



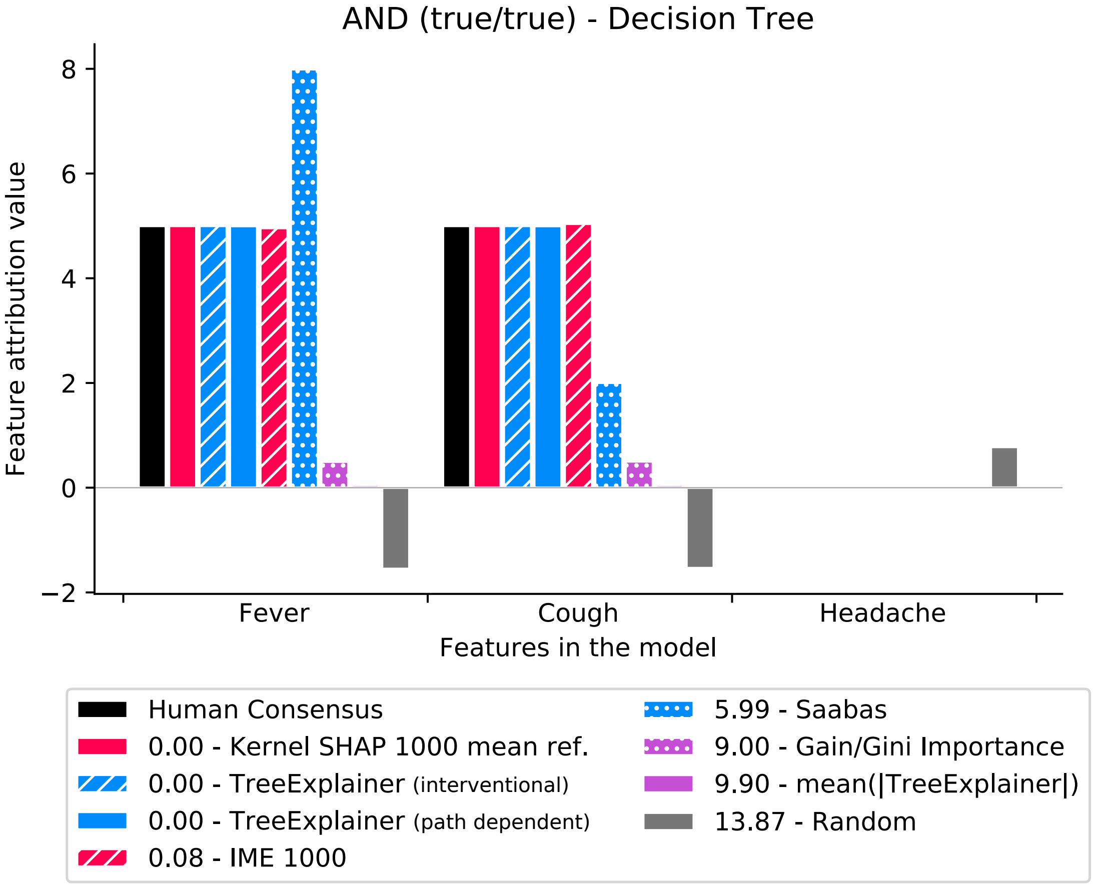
Supplementary Figure 34: **Consensus human intuition values for an OR function.** Human consensus values were measured by a user study over 33 participants for a simple OR-based function. The most popular allocation was chosen as the consensus (7). The labels denote the allocation given by people as “fever / cough / headache”. The title gives the input values for sample being explained as “fever value/cough value”, where ‘T’ is true and ‘F’ is false; note that headache is always set to true.



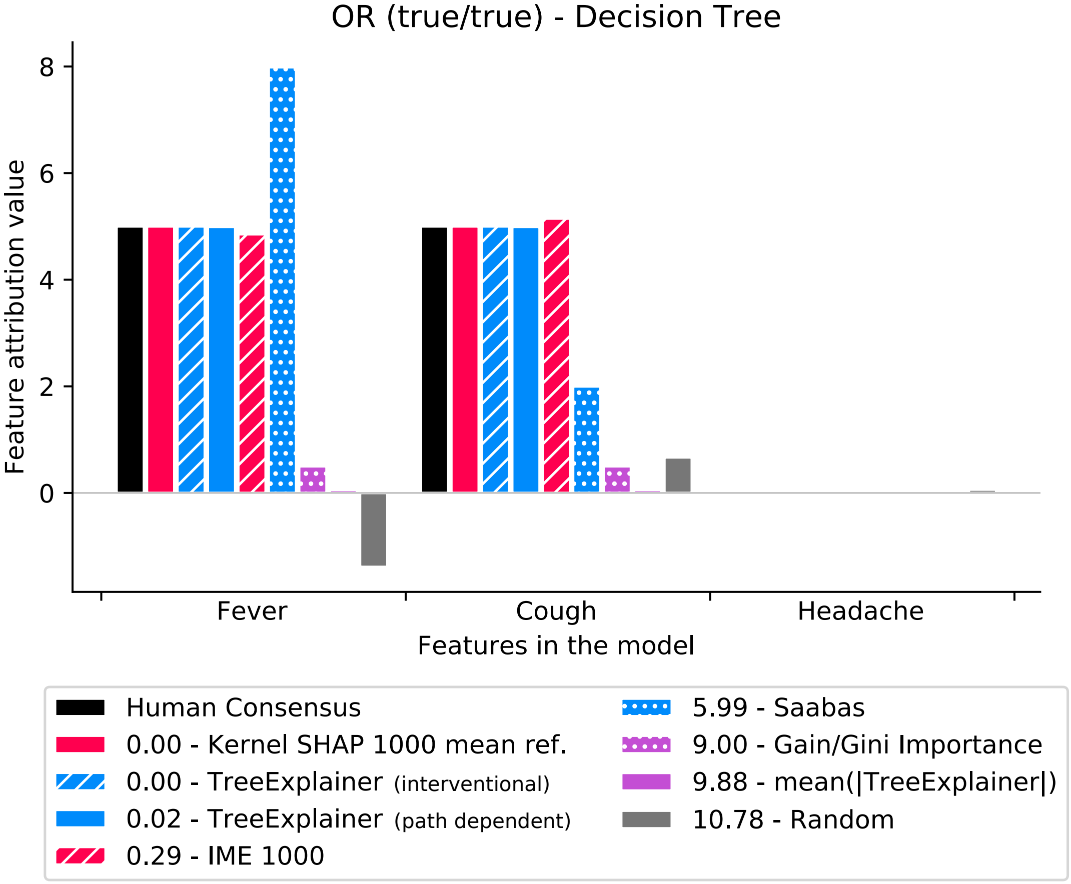
Supplementary Figure 35: **Consensus human intuition values for an eXclusive OR (XOR) function.** Human consensus values were measured by a user study over 33 participants for a simple XOR-based function. The most popular allocation was chosen as the consensus (7). The labels denote the allocation given by people as “fever / cough / headache”. The title gives the input values for sample being explained as “fever value/cough value”, where ‘T’ is true and ‘F’ is false; note that headache is always set to true.



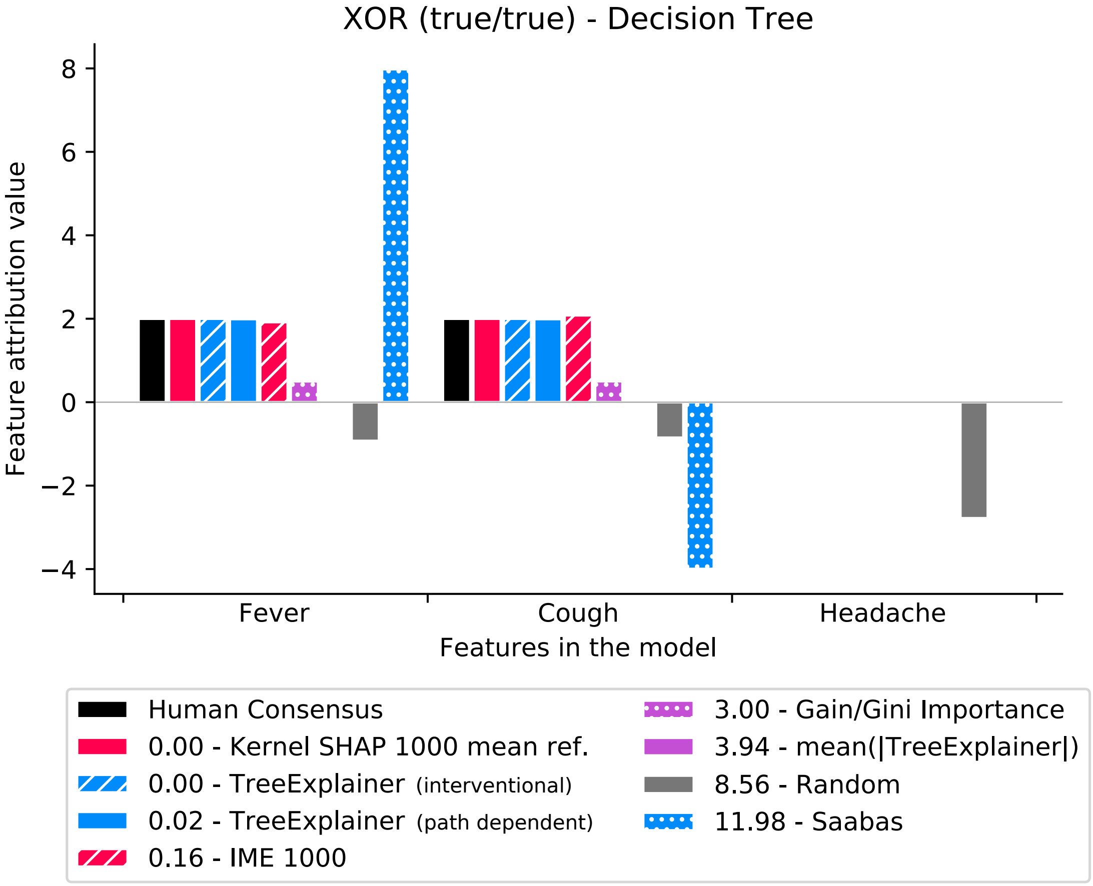
Supplementary Figure 36: **Consensus human intuition values for a SUM function.** Human consensus values were measured by a user study over 33 participants for a simple SUM function. The most popular allocation was chosen as the consensus (7). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



Supplementary Figure 37: **Comparison with human intuition for an AND function.** Human consensus values were measured by a user study for a simple AND-based function evaluated when all inputs were set to 'true' (Supplementary Figure 33). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (7).



Supplementary Figure 38: **Comparison with human intuition for an OR function.** Human consensus values were measured by a user study for a simple OR-based function evaluated when all inputs were set to 'true' (Supplementary Figure 34). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (7).



Supplementary Figure 39: **Comparison with human intuition for an eXclusive OR (XOR) function.** Human consensus values were measured by a user study for a simple XOR-based function evaluated when all inputs were set to ‘true’ (Supplementary Figure 35). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (7).

References

[1] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". In: *6th International Conference on Learning Representations (ICLR 2018)*. 2018.

[2] Lidia Auret and Chris Aldrich. "Empirical comparison of tree ensemble variable importance measures". In: *Chemometrics and Intelligent Laboratory Systems* 105.2 (2011), pp. 157–170.

[3] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[4] David Baehrens et al. "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11.Jun (2010), pp. 1803–1831.

[5] George L Bakris et al. "Effects of blood pressure level on progression of diabetic nephropathy: results from the RENAAL study". In: *Archives of internal medicine* 163.13 (2003), pp. 1555–1565.

[6] Benjamin Bowe et al. "Association between monocyte count and risk of incident CKD and progression to ESRD". In: *Clinical Journal of the American Society of Nephrology* 12.4 (2017), pp. 603–613.

[7] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[8] Leo Breiman et al. *Classification and regression trees.* CRC press, 1984.

[9] S Chebrolu, A Abraham, and J Thomas. "Feature deduction and ensemble design of intrusion detection systems". In: *Computers & security* 24.4 (2005), pp. 295–307.

[10] Tianqi Chen and Carlos Guestrin. "XGBoost: A scalable tree boosting system". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 785–794.

[11] Alfred K Cheung et al. "Effects of intensive BP control in CKD". In: *Journal of the American Society of Nephrology* 28.9 (2017), pp. 2812–2823.

[12] François Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[13] Chronic Kidney Disease Prognosis Consortium et al. "Association of estimated glomerular filtration rate and albuminuria with all-cause and cardiovascular mortality in general population cohorts: a collaborative meta-analysis". In: *The Lancet* 375.9731 (2010), pp. 2073–2081.

[14] Christine S Cox et al. "Plan and operation of the NHANES I Epidemiologic Followup Study, 1992". In: (1997).

[15] Anupam Datta, Shayak Sen, and Yair Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 598–617.

[16] Cameron Davidson-Pilon. *Lifelines*. `https://github.com/camdavidsonpilon/lifelines`. 2016.

[17] Fangfang Fan et al. "White blood cell count predicts the odds of kidney function decline in a Chinese community-based population". In: *BMC nephrology* 18.1 (2017), p. 190.

[18] Jing Fang and Michael H Alderman. "Serum uric acid and cardiovascular mortality: the NHANES I epidemiologic follow-up study, 1971-1992". In: *Jama* 283.18 (2000), pp. 2404–2410.

[19] Ruth C Fong and Andrea Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3429–3437.

[20] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[21] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[22] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. "Variable selection using random forests". In: *Pattern Recognition Letters* 31.14 (2010), pp. 2225–2236.

[23] SPRINT Research Group. "A randomized trial of intensive versus standard blood-pressure control". In: *New England Journal of Medicine* 373.22 (2015), pp. 2103–2116.

[24] Patrick J Heagerty, Thomas Lumley, and Margaret S Pepe. "Time-dependent ROC curves for censored survival data and a diagnostic marker". In: *Biometrics* 56.2 (2000), pp. 337–344.

[25] A Irrthum, L Wehenkel, P Geurts, et al. "Inferring regulatory networks from expression data using tree-based methods". In: *PloS one* 5.9 (2010), e12776.

[26] Hemant Ishwaran et al. "Variable importance in binary regression trees and forests". In: *Electronic Journal of Statistics* 1 (2007), pp. 519–537.

[27] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. "Feature relevance quantification in explainable AI: A causality problem". In: *arXiv preprint arXiv:1910.13413* (2019).

[28] Kaggle. *The State of ML and Data Science 2017*. 2017. URL: `https://www.kaggle.com/surveys/2017`.

[29] Jared L Katzman et al. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network". In: *BMC medical research methodology* 18.1 (2018), p. 24.

[30] Jalil Kazemitabar et al. "Variable Importance using Decision Trees". In: *Advances in Neural Information Processing Systems*. 2017, pp. 426–435.

[31] *Kidney Disease Statistics for the United States*. 2018. URL: `https://www.niddk.nih.gov/health-information/health-statistics/kidney-disease`.

[32] Masaomi Kimura et al. "Comparison of lesion formation between contact force-guided and non-guided circumferential pulmonary vein isolation: a prospective, randomized study." In: *Heart rhythm* 11.6 (June 2014), pp. 984–91. ISSN: 1556-3871. DOI: `10.1016/j.hrthm.2014.03.019`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S1547527114003002%20http://www.ncbi.nlm.nih.gov/pubmed/24657428`.

[33] Pieter-Jan Kindermans et al. "Learning how to explain neural networks: PatternNet and PatternAttribution". In: *arXiv preprint arXiv:1705.05598* (2017).

[34] Karl-Heinz Kuck et al. "Cryoballoon or Radiofrequency Ablation for Paroxysmal Atrial Fibrillation". In: *New England Journal of Medicine* 374.23 (June 2016), pp. 2235–2245. ISSN: 0028-4793. DOI: `10.1056/NEJMoa1602014`. URL: `http://www.nejm.org/doi/10.1056/NEJMoa1602014`.

[35] James P Lash et al. "Chronic Renal Insufficiency Cohort (CRIC) Study: baseline characteristics and associations with kidney function". In: *Clinical Journal of the American Society of Nephrology* 4.8 (2009), pp. 1302–1311.

[36] Lenore J Launer et al. "Body mass index, weight change, and risk of mobility disability in middle-aged and older women: the epidemiologic follow-up study of NHANES I". In: *Jama* 271.14 (1994), pp. 1093–1098.

[37] Stan Lipovetsky and Michael Conklin. "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pp. 319–330.

[38] Gilles Louppe. "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502* (2014).

[39] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 4768–4777. URL: `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

[40] Eloi Marijon et al. "Real-time contact force sensing for pulmonary vein isolation in the setting of paroxysmal atrial fibrillation: procedural and 1-year results." In: *Journal of cardiovascular electrophysiology* 25.2 (Feb. 2014), pp. 130–7. ISSN: 1540-8167. DOI: `10.1111/jce.12303`. URL: `http://doi.wiley.com/10.1111/jce.12303%20http://www.ncbi.nlm.nih.gov/pubmed/24433324`.

[41] Yasuko Matsui and Tomomi Matsui. "NP-completeness for calculating power indices of weighted majority games". In: *Theoretical Computer Science* 263.1-2 (2001), pp. 305–310.

[42] Kunihiro Matsushita et al. "Comparison of risk prediction using the CKD-EPI equation and the MDRD study equation for estimated glomerular filtration rate". In: *Jama* 307.18 (2012), pp. 1941–1951.

[43] Kunihiro Matsushita et al. "Estimated glomerular filtration rate and albuminuria for prediction of cardiovascular outcomes: a collaborative meta-analysis of individual participant data". In: *The lancet Diabetes & endocrinology* 3.7 (2015), pp. 514–525.

[44] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[45] Paula F Orlandi et al. "Hematuria as a risk factor for progression of chronic kidney disease and death: findings from the Chronic Renal Insufficiency Cohort (CRIC) Study". In: *BMC nephrology* 19.1 (2018), p. 150.

[46] Feng Pan et al. "Feature selection for ranking using boosted trees". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pp. 2025–2028.

[47] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[48] H Mitchell Perry Jr et al. "Early predictors of 15-year end-stage renal disease in hypertensive patients". In: *Hypertension* 25.4 (1995), pp. 587–594.

[49] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. "Model agnostic supervised local explanations". In: *Advances in Neural Information Processing Systems*. 2018, pp. 2515–2524.

[50] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-precision model-agnostic explanations". In: *AAAI Conference on Artificial Intelligence*. 2018.

[51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD*. ACM. 2016, pp. 1135–1144.

[52] Steven J Rosansky et al. "The association of blood pressure levels and change in renal function in hypertensive and nonhypertensive subjects". In: *Archives of internal medicine* 150.10 (1990), pp. 2073–2076.

[53] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[54] A. Saabas. *treeinterpreter Python package*. URL: `https://github.com/andosa/treeinterpreter`.

[55] Wojciech Samek et al. "Evaluating the visualization of what a deep neural network has learned". In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), pp. 2660–2673.

[56] Marco Sandri and Paola Zuccolotto. "A bias correction algorithm for the Gini variable importance measure in classification trees". In: *Journal of Computational and Graphical Statistics* 17.3 (2008), pp. 611–628.

[57] Mark J Sarnak et al. "The effect of a lower target blood pressure on the progression of kidney disease: long-term follow-up of the modification of diet in renal disease study". In: *Annals of internal medicine* 142.5 (2005), pp. 342–351.

[58] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[59] Avanti Shrikumar et al. "Not Just a Black Box: Learning Important Features Through Propagating Activation Differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[60] Neil B Shulman et al. "Prognostic value of serum creatinine and effect of treatment of hypertension on renal function. Results from the hypertension detection and follow-up program. The Hypertension Detection and Follow-up Program Cooperative Group." In: *Hypertension* 13.5 Suppl (1989), p. I80.

[61] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[62] Carolin Strobl et al. "Bias in random forest variable importance measures: Illustrations, sources and a solution". In: *BMC bioinformatics* 8.1 (2007), p. 25.

[63] C Strobl et al. "Conditional variable importance for random forests". In: *BMC bioinformatics* 9.1 (2008), p. 307.

[64] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41.3 (2014), pp. 647–665.

[65] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *arXiv preprint arXiv:1703.01365* (2017).

[66] Navdeep Tangri et al. "Multinational assessment of accuracy of equations for predicting risk of kidney failure: a meta-analysis". In: *Jama* 315.2 (2016), pp. 164–174.

[67] W Gordon Walker et al. "Renal function change in hypertensive members of the Multiple Risk Factor Intervention Trial: racial and treatment effects". In: *JAMA* 268.21 (1992), pp. 3085–3091.

[68] F Perry Wilson et al. "Urinary creatinine excretion, bioelectrical impedance analysis, and clinical outcomes in patients with CKD: the CRIC study". In: *Clinical Journal of the American Society of Nephrology* 9.12 (2014), pp. 2095–2103.

[69] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[70] Martin Zinkevich. *Rules of Machine Learning: Best Practices for ML Engineering*. 2017.