Advanced Machine Learning

Paper: *From local explanations to global understanding with explainable AI for trees*
by Lundberg et al. (2020)

Antonio Sutera
a.sutera@uliege.be

LIÈGE université

March 12, 2020

Motivation – trees

"Machine learning models based on trees are the ***most popular nonlinear models in use today****. Random Forests, gradient boosted trees and other-based models are used in finance, medicine, biology, customer retention, advertising, supply chain management, manufacturing, public health and other* ***to make predictions based on sets of input features****."*

Performance of tree-based models

"Tree-based models can be ***more accurate*** *than neural networks in many applications. While deep learning models are more appropriate in fields such as image recognition, speech recognition and natural language processing, tree-based models consistently outperform standard deep models on tabular-style datasets, where* ***features are individually meaningful*** *and lack strong multiscale temporal or spatial structures."*

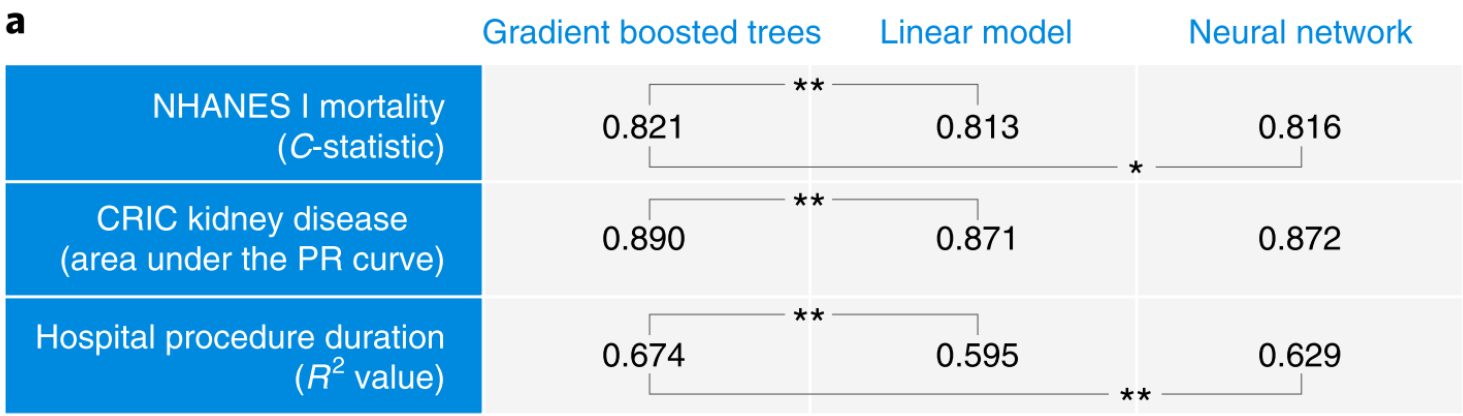


Fig. 2 | Gradient boosted tree models can be more accurate than neural networks and more interpretable than linear models. a, Gradient boosted tree models outperform both linear models and neural networks on all our medical datasets, where ** represents a bootstrap retrain $P < 0.01$, and * represents $P = 0.03$. The C-statistic is the area under the true positive versus false positive curve, PR stands for precision recall, and R^2 is the coefficient of determination.

Motivation – applications need interpretation

"For these applications, models often must be both accurate and interpretable, where ***interpretability means that we can understand how the model uses input features to make predictions****."*

Motivation – interpretations of trees

*"**Explaining predictions from tree models** is particularly importance in medical applications, where the pattern a model uncovers can be more important than the model's prediction performance."*

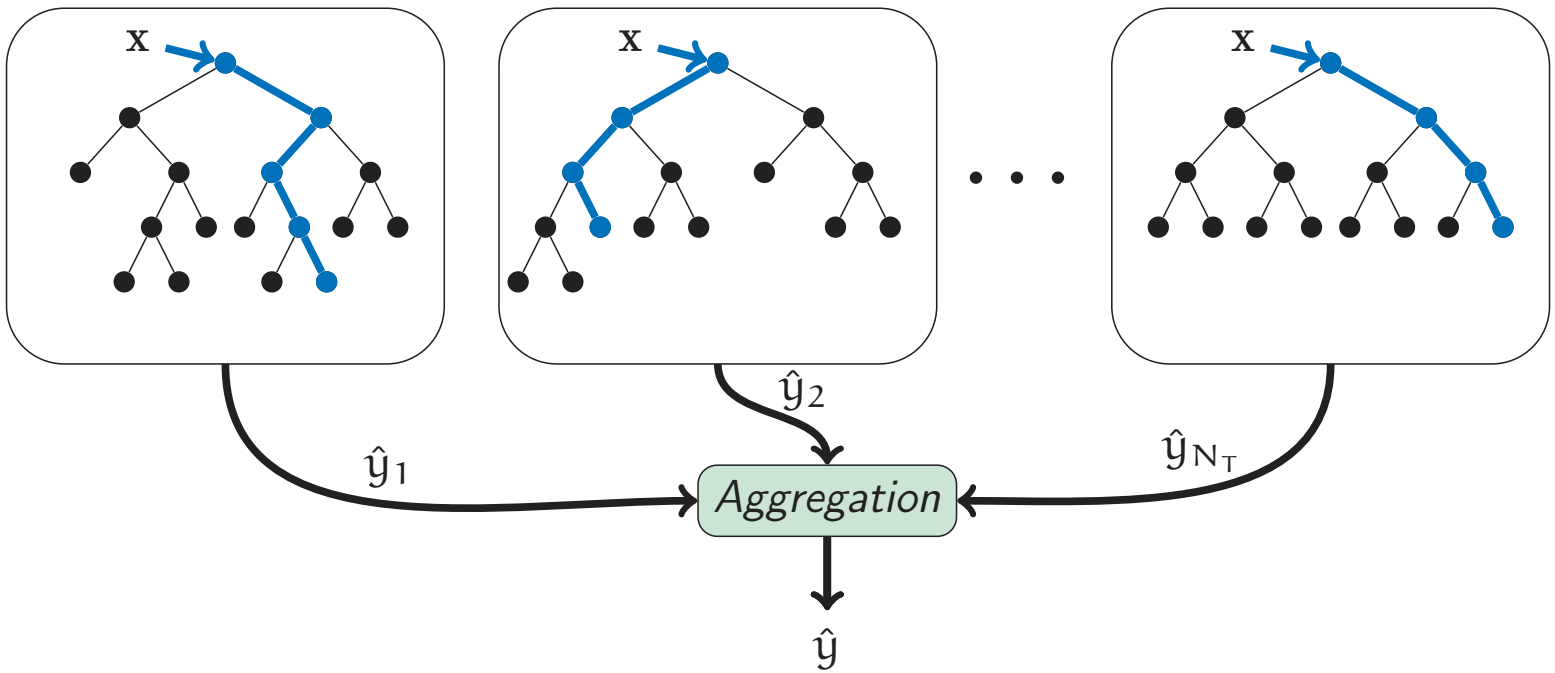
*"Tree-based machine learning models such as random forests, decision trees and gradient boosted trees are popular nonlinear models, **yet comparatively little attention has been paid to explaining their predictions**"*

Supervised learning setting

Goal: From a learning set of N samples, find a function f of the inputs $V = \{X_1, X_2, \ldots, X_p\}$ that approximates at best the output Y.

Random forest

Idea: Combining several *randomised* trees is better than a single one.



Techniques to build diverse trees

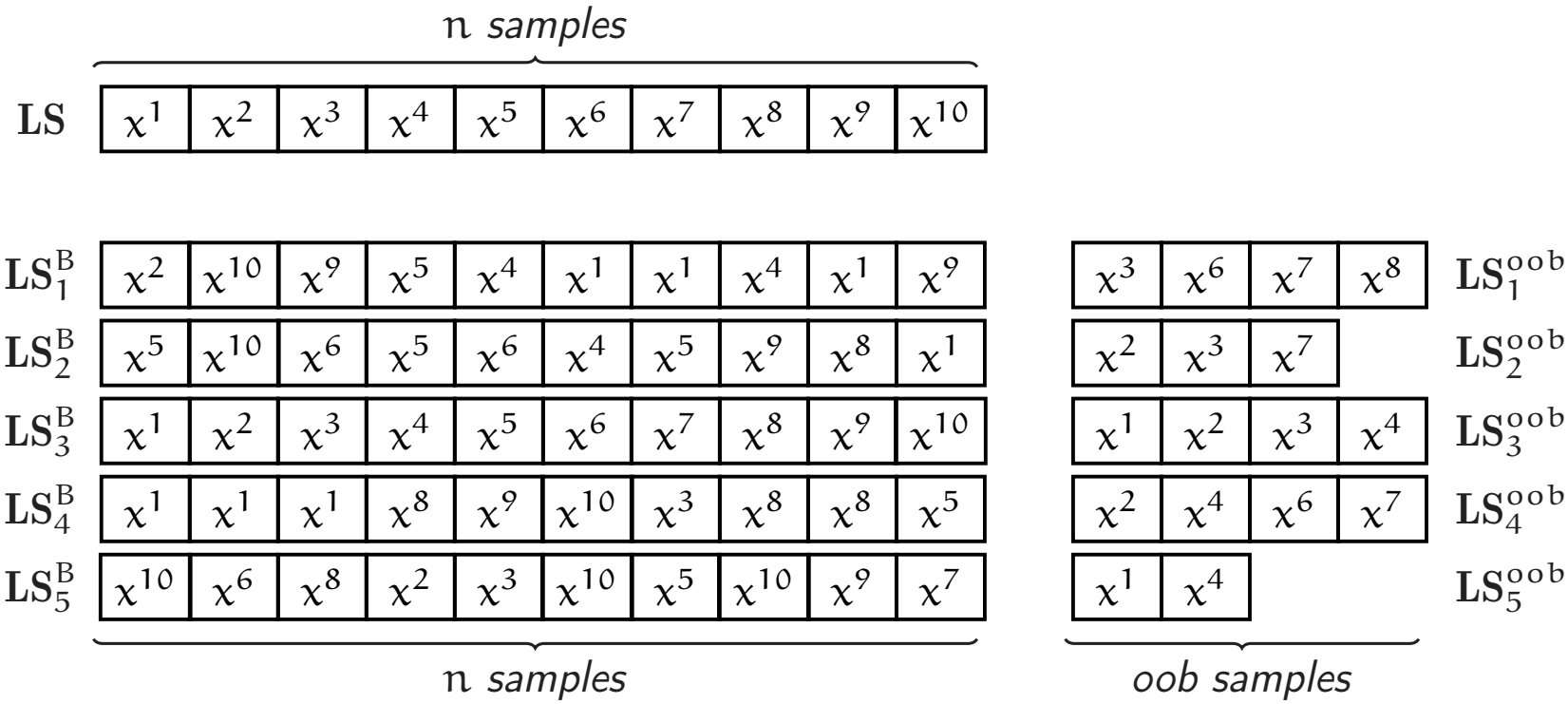
Ways of introduction randomization in the tree growing procedure

- Tree-wise learning set randomization: e.g. bootstraps or feature (or sample) subspaces
- Node-wise variable randomization
- Node-wise split randomization

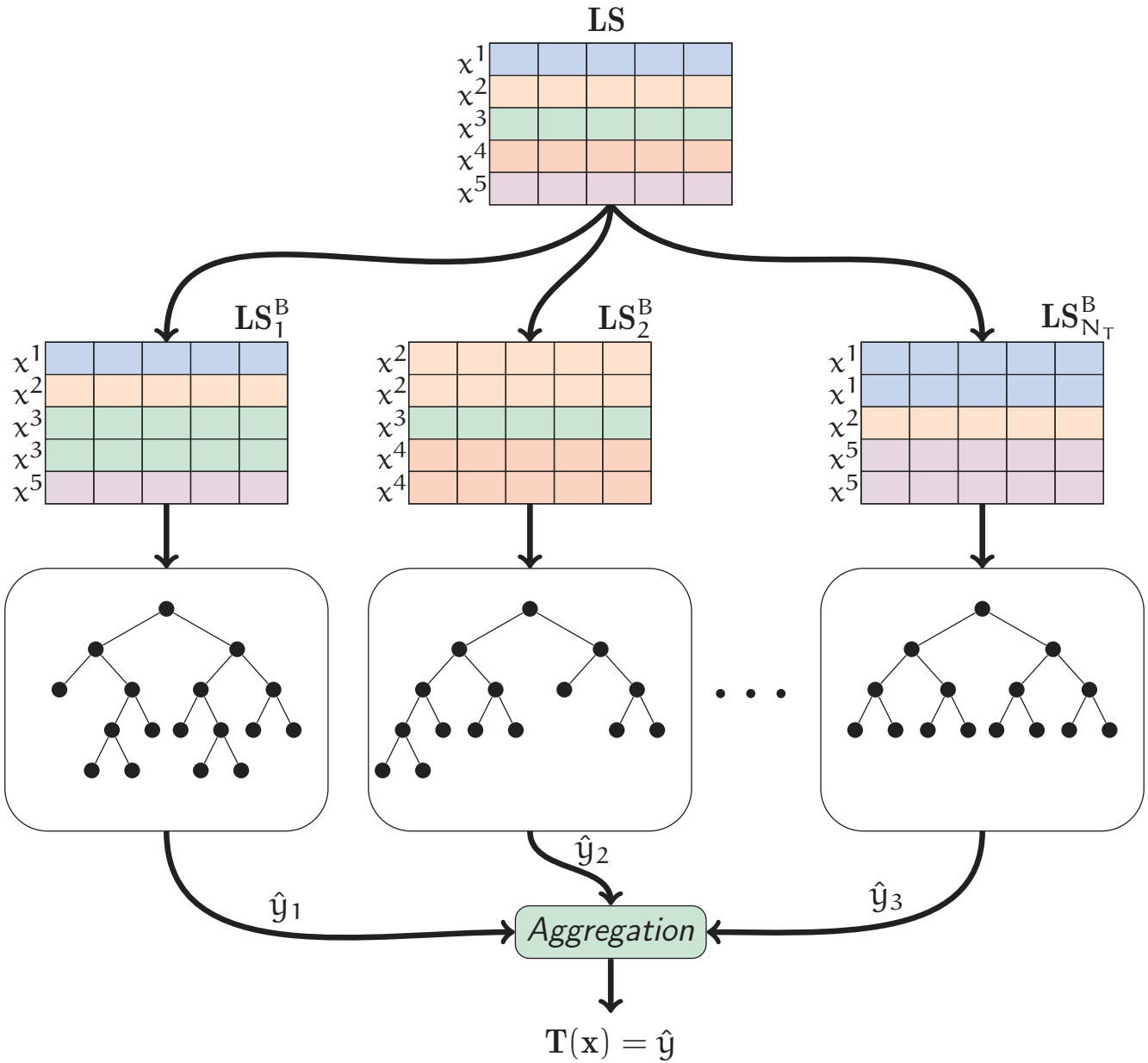
Random forests vs randomization techniques

	Boot.	Subspace	Feat. rand.	Split rand.
Bagging	✓			
Random Subspace		✓	(✓)	
Random Forest	✓		✓	
Extra-Trees			✓	✓

How to make bootstrap samples?



Bagging


LS
x^1
x^2
x^3
x^4
x^5
LS_1^B
LS_2^B
$LS_{N_T}^B$
x^1
x^2
x^3
x^3
x^5
x^2
x^2
x^3
x^4
x^4
x^1
x^1
x^2
x^5
x^5
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
Aggregation
$\mathbf{T}(\mathbf{x}) = \hat{y}$

Node-wise randomisation

Idea: At each node, choose the best split variable.

Split variable	Δi
X_1	$\Delta i_1 = 0.2$
X_2	$\Delta i_2 = 1.2$
X_3	$\Delta i_3 = 2.6$
X_4	$\Delta i_4 = 0.7$
X_5	$\Delta i_5 = 1.8$
X_6	$\Delta i_6 = 1.3$
$\vdots$	$\vdots$

The split variable yielding the highest impurity reduction is X_3.

Node-wise randomisation

Idea: At each node, choose the best split variable among K randomly selected variables.

Split variable	Δi
X_1	$\Delta i_1 = 0.2$
X_2	$\Delta i_2 = 1.2$
X_3	$\Delta i_3 = 2.6$
X_4	$\Delta i_4 = 0.7$
X_5	$\Delta i_5 = 1.8$
X_6	$\Delta i_6 = 1.3$
$\vdots$	$\vdots$

The split variable yielding the highest impurity reduction is X_5.

Is random forest a black-box model?

It **works well** in practice **but** the model is **not interpretable**.

Understanding a random forest model

- Identifying the variables that are involved in the random forest model gives insight on its mechanism.
- Therefore, we need to **measure variable importances in random forest models**.

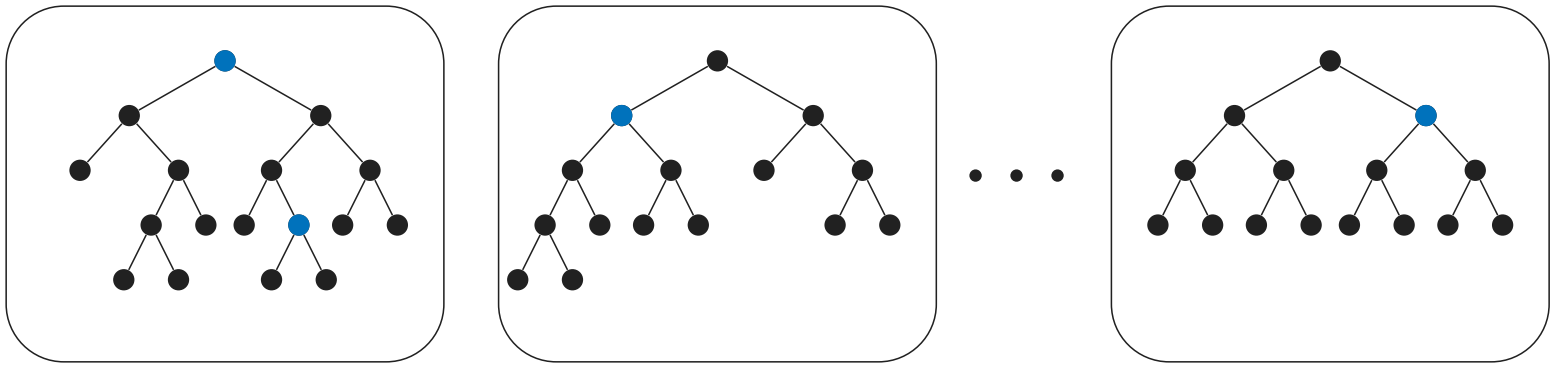
Two importance measures

From a random forest, we can evaluate the importance of variables in the model based on their:

- Mean Decrease of Accuracy (MDA)
- Mean Decrease of Impurity (MDI)

Mean Decrease of Impurity (MDI)

Nodes splitting on X_m are in blue.

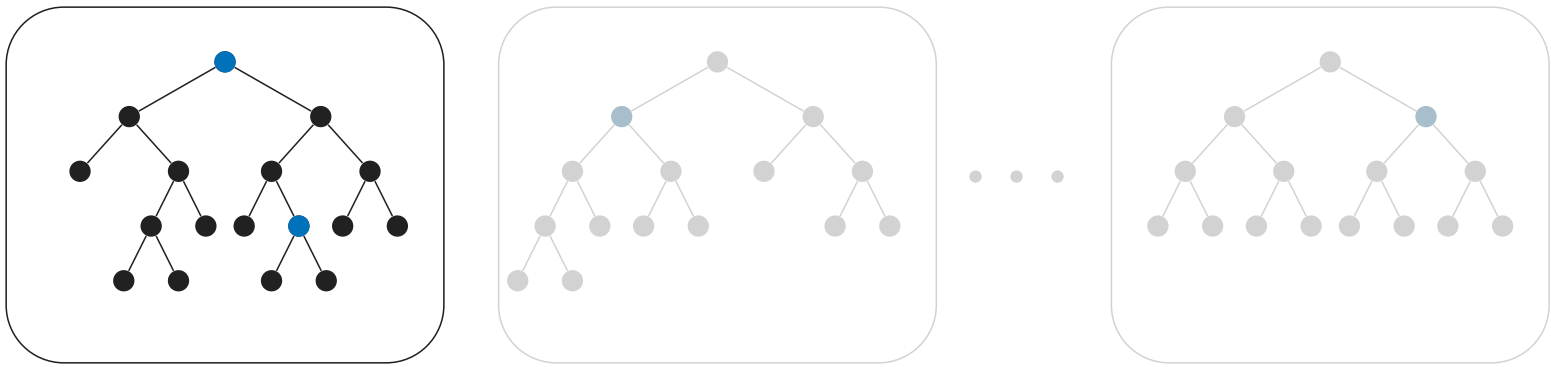


MDI importance of X_m for an ensemble of N_T trees:

$$\mathrm{Imp}^{mdi}(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t) \Delta i(s, t)$$

Mean Decrease of Impurity (MDI)

Nodes splitting on X_m are in blue.

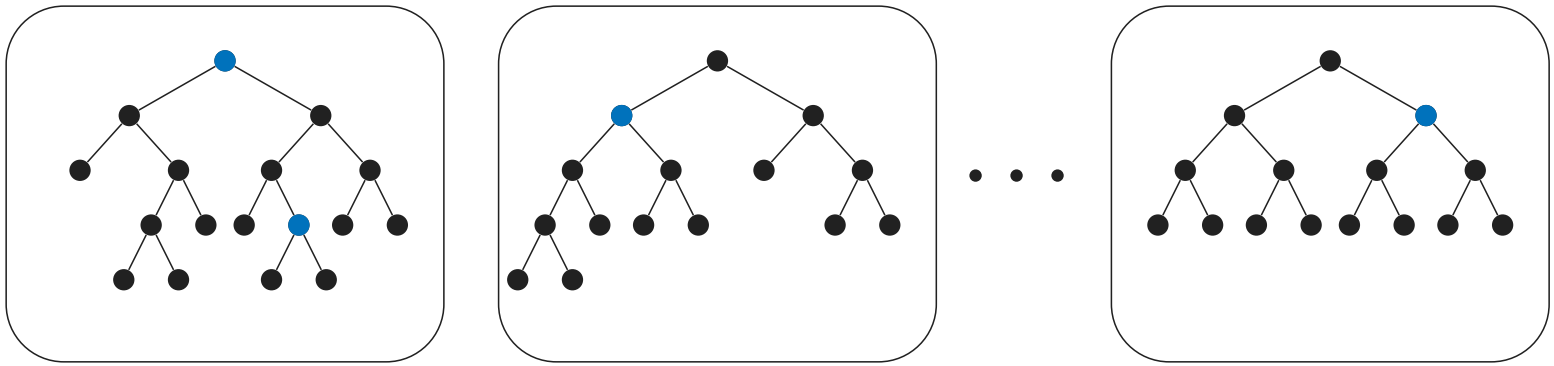


MDI importance of X_m for *one tree*:

$$\mathrm{Imp}^{mdi}(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t) \Delta i(s, t)$$

Mean Decrease of Impurity (MDI)

Nodes splitting on X_m are in blue.



MDI importance of X_m for an ensemble of N_T trees:

$$\mathrm{Imp}^{mdi}(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t) \Delta i(s, t)$$

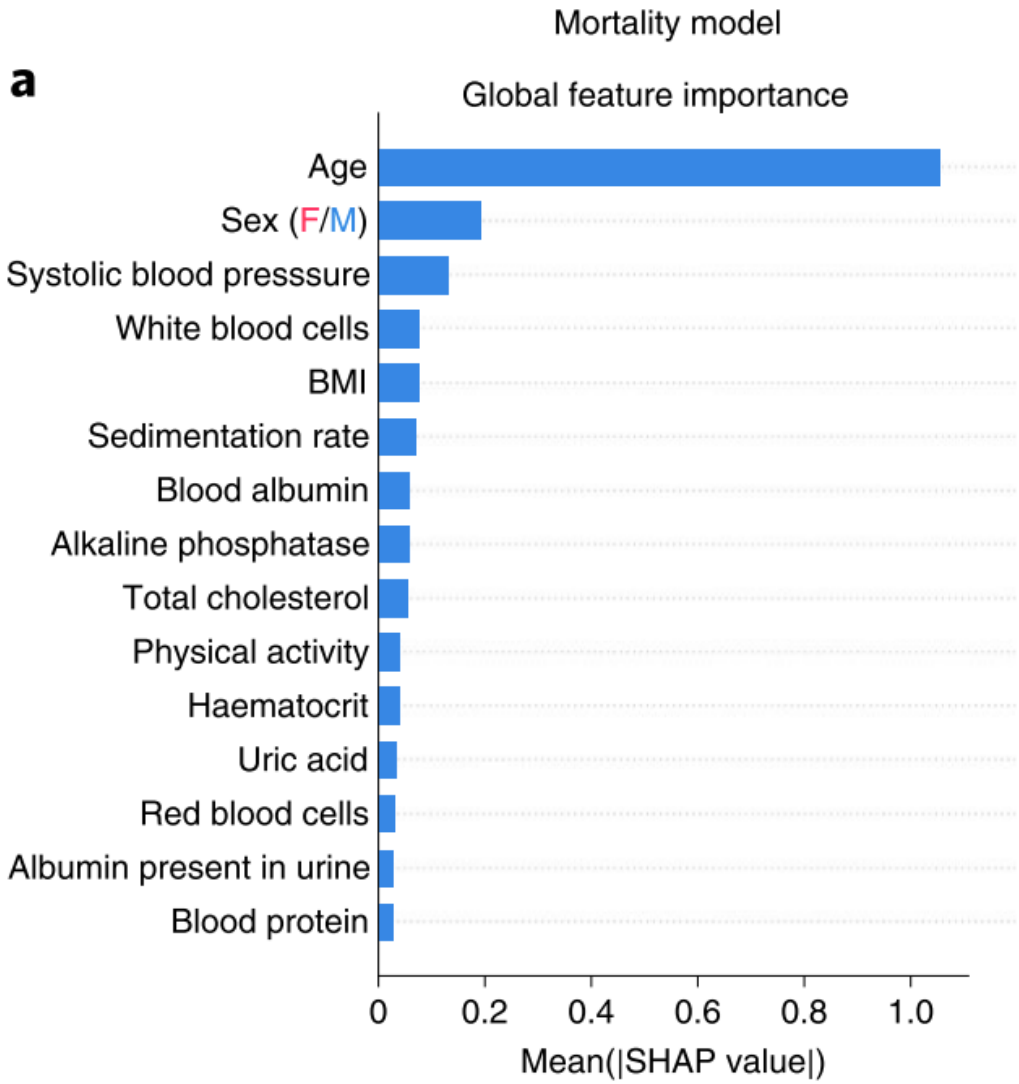
Mean Decrease of Accuracy (MDA)

$$\mathrm{Imp}^{\mathrm{mda}}(X_m) = \frac{1}{N_T} \sum_T (\widetilde{\mathrm{errOOB}}_T^m - \mathrm{errOOB}_T)$$

where

- N_T is the number of trees,
- errOOB_T is the error (MSE, $0-1$, ...) of a single tree T on its OOB_T sample,
- $\widetilde{\mathrm{errOOB}}_T^m$ is the error of a single tree T on its OOB_T sample where the values of X_m have been permuted.

Global interpretation



Local explanations: motivation

*"**Explaining predictions** from tree models is particularly importance in medical applications, where the pattern a model uncovers can be more important than the model's prediction performance."*

*"Tree-based machine learning models such as random forests, decision trees and gradient boosted trees are popular nonlinear models, **yet comparatively little attention has been paid to explaining their predictions**"*

Tree-based local explanation methods

Current methods:

- reporting the decision path,
- using a heuristic approach that assigns credit to each input feature and,
- applying various model-agnostic approaches that require repeatedly executing the model for each explanation.

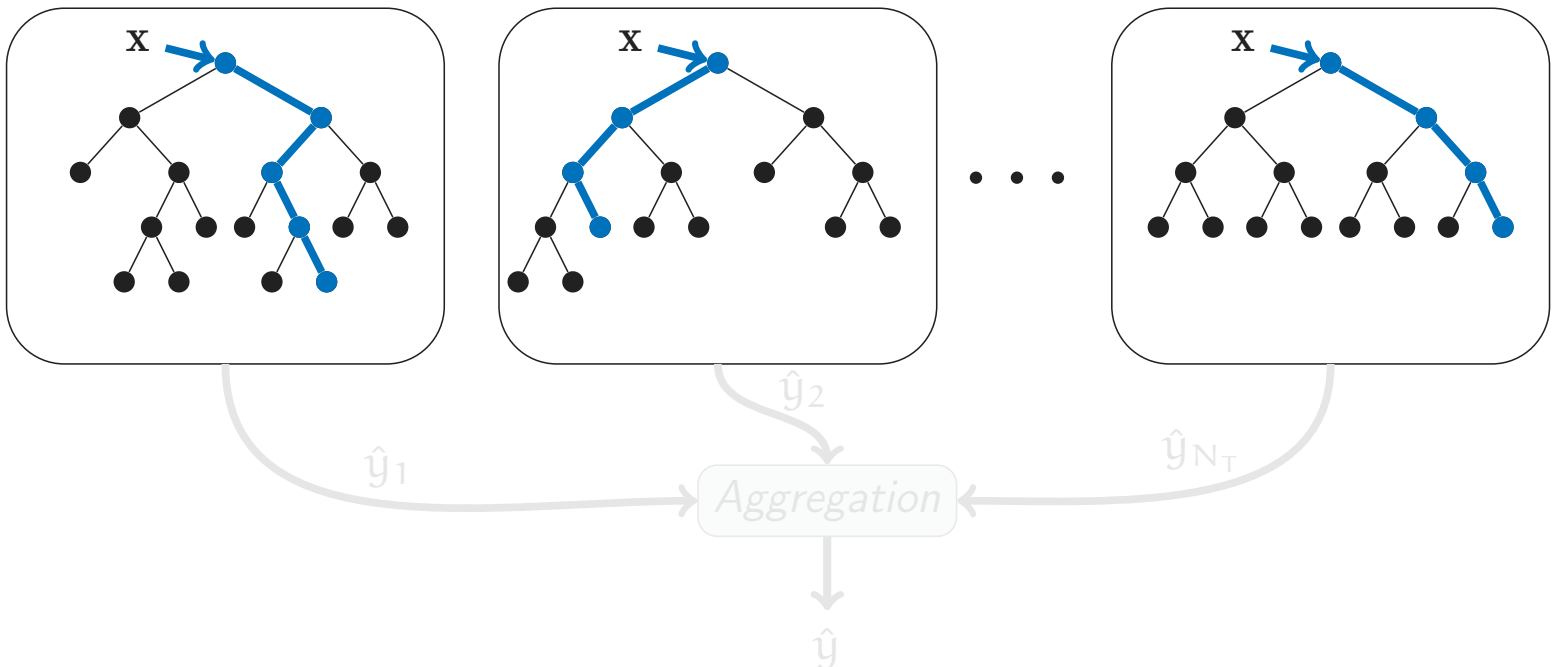
Tree-based local explanation methods

Current methods have limitations:

- reporting the decision path,
  Unhelpful for most models, particularly those based on multiple trees
- using a heuristic approach that assigns credit to each input feature and,
  - (applying various model-agnostic approaches that require repeatedly executing the model for each explanation.)

Random forest

Idea: Combining several *randomised* trees is better than a single one.



Tree-based local explanation methods

Current methods have limitations:

- reporting the decision path,
- using a heuristic approach that assigns credit to each input feature and,
  The behaviour of the heuristic credit allocation has to be carefully analysed
  - (applying various model-agnostic approaches that require repeatedly executing the model for each explanation.)

Tree-based local explanation methods

Current methods have limitations:

- reporting the decision path,
- using a heuristic approach that assigns credit to each input feature and,
  - (applying various model-agnostic approaches that require repeatedly executing the model for each explanation). *Rely on post hoc modelling of an arbitrary function, they can be slow and suffer from sampling variability*

Difference in expectation (proposed by Saabas)

Principle: Heuristic that explains a prediction by following the decision path and attributing changes in the model's expected output to each feature along the path.

Sabaas value

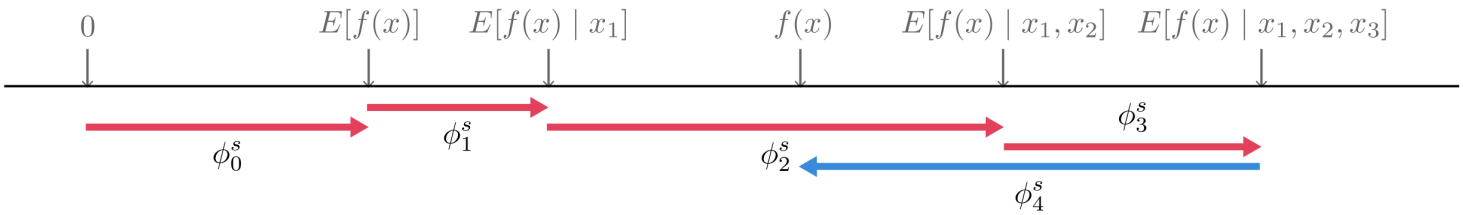
Let f be a decision tree model, x the instance we are going to explain, $f(x)$ the output of the model for the current instance, $f_x(S) \approx \mathbb{E}[f(x)|x_S]$ the estimated expectation of the model output conditioned on the set S of feature values,

the **Sabaas value** for the i'th feature as

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

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.

Sum of Saabas values



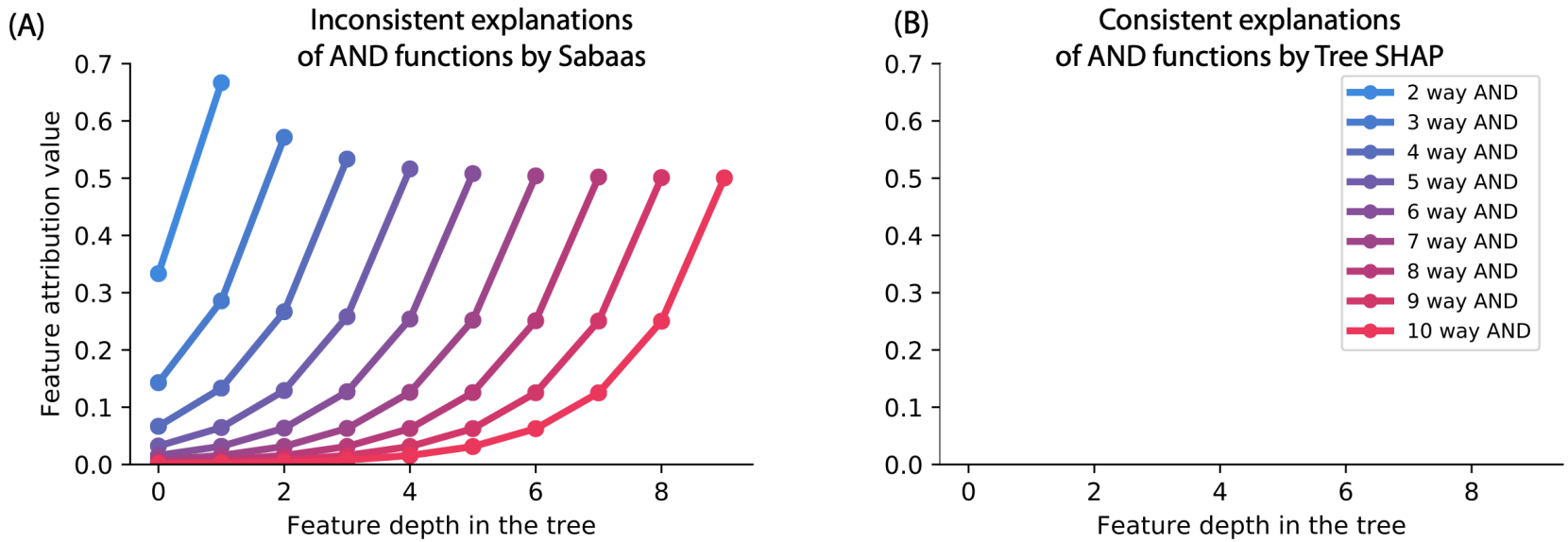
Supplementary Figure 15: The Sabaas values, ϕ_i^s, attribute feature importance by measuring differences in conditional expectations along the order defined by the decision path. This is very similar to SHAP (SHapley Additive exPlanation) values, ϕ_i, except SHAP values result from averaging over all 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 $\sum_{i=0}^{M} \phi_i = f(x)$ ().

Sabaas value: remarks

- This 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.
- When explaining an ensemble model made of a sum of many decision trees, the Saabas values for the ensemble model are defined as the sum fo the Saabas values for each tree.

Bias of Saabas value

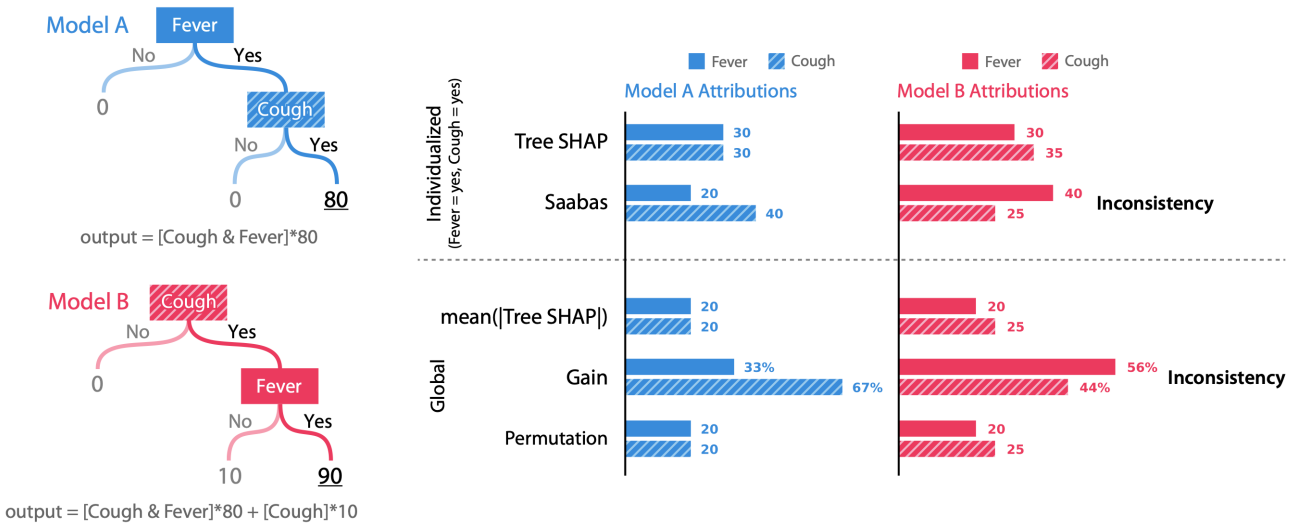
"The Saabas method is biased to alter the impact of features based on their distance from a tree's root."



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.

Inconsistency

"This bias makes Saabas values inconsistent, where increasing a model's dependence on a feature may actually decrease that feature's Saabas value."



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.

TreeExplainer

Three improvements:

- Exact computation of Shapley value explanations for tree-based models in polynomial time.
- Extending local explanations to directly capture feature interactions.
- Tools for interpreting global model structure based on many local explanations.

Global structure representation

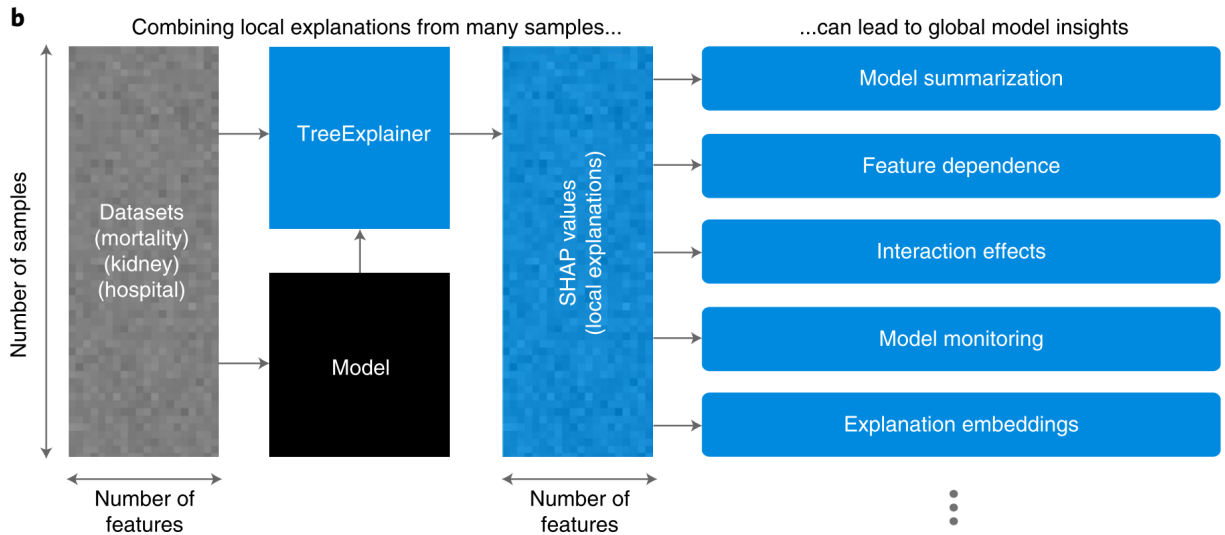


Fig. 1 | Local explanations based on TreeExplainer enable a wide variety of new ways to understand global model structure. **b**, By combining many local explanations, we can represent global structure while retaining local faithfulness to the original model. We demonstrate this by using three medical datasets to train gradient boosted decision trees and then compute local explanations based on SHAP values[3]. Computing local explanations across all samples in a dataset enables development of many tools for understanding global model structure.

Shapley values (from game theory)

Let V be a set $\{X_1, \ldots, X_p\}$ of p players and a characteristic function ν ($\nu : 2^V \to \mathbb{R}$), with $\nu(\emptyset) = 0$.

The **Shapley value of X_m** is (given ν):

$$\Phi_\nu(X_m) = \sum_{S \subseteq V^{-m}} \frac{|S|!(p - |S| - 1)!}{p!} (\nu(S \cup \{X_m\} - \nu(S))$$

where

- $V^{-m} = V \setminus \{X_m\}$,
- $\nu(S)$ (called the worth of coalition S) is the total expected sum of payoffs the members of S can obtain by cooperation.

Source : Wikipedia

Understanding Shapley values

$$\Phi_\nu(X_m) = \sum_{S \subseteq V^{-m}} \frac{|S|!(p - |S| - 1)!}{p!}(\nu(S \cup \{X_m\} - \nu(S))$$

Given a coalition of players S, $\nu(S \cup \{X_m\}) - \nu(S))$ is the marginal contribution of X_m to the coalition and $\Phi_\nu(X_m)$ is the average of its contribution over all possible coalitions.

Shapley values properties (1/2)

- **Efficiency**:

$$\sum_{m=1}^{p} \Phi_\nu(X_m) = \nu(V)$$

The sum of the Shapley values equals the worth of the coalition made of all players $\nu(V)$.

- **Symmetry**: If X_i and X_j are two players who are such that

$$\nu(S \cup \{X_i\}) = \nu(S \cup \{X_j\})$$

for every subset S of $V^{-i,j}$, then $\Phi_\nu(X_i) = \Phi_\nu(X_j)$.

If two players are equivalent, they are treated equally

Shapley values properties (2/2)

- **Consistency / monotonicity**: Let us denote by v and w two characteristic functions. If, for all $S \subseteq V^{-m}$, we have

$$(v(S \cup \{X_m\}) - v(S)) \geqslant (w(S \cup \{X_m\}) - w(S)),$$

then we have $\Phi_v(X_m) \geqslant \Phi_w(X_m)$.

If a game changes (v to w) so that some player's contribution increases or stays the same regardless of the other players, that player's attribution should not decrease.

- **Missingness / Null player**: If, for all $S \subseteq V^{-i}$

$$v(S \cup \{X_i\}) = v(S)$$

then $\Phi_v(X_i) = 0$.

A null player has a zero Shapley value.

Theorem

*"The **only way** to simultaneously satisfy these properties is to use the **classic Shapley values**."*

Shapley values applied to local explanations

"Classic Shapley values can be considered ***'optimal'*** *since, within a large class of approaches, they are the only way to measure feature importance while* ***maintaining several natural properties*** *from cooperative game theory."*

Computing Shapley values

"Unfortunately, in general, these values can only be approximated since computing them exactly is ***NP-hard****, requiring summation over all feature subsets."*

Shapley values applied to local explanations

Principle: Shapley values are computed by introducing each feature, one at a time, into a conditional expectation function of the model's output, $f_x(S) = E[f(X)|do(X_S = x_S)]$, and attributing the change produced at each step to the feature that was introduced; then averaging this process over all possible feature orderings.

Properties

- **Efficiency**:

$$f(x) = \Phi_f(\emptyset) + \sum_{i=1}^{p} \Phi_f(X_i, x)$$

- **Consistency**: For any two models f and f'

$$f'_x(S \cup \{X_i\}) - f'_x(S) \geqslant f_x(S \cup \{X_i\}) - f_x(S)$$

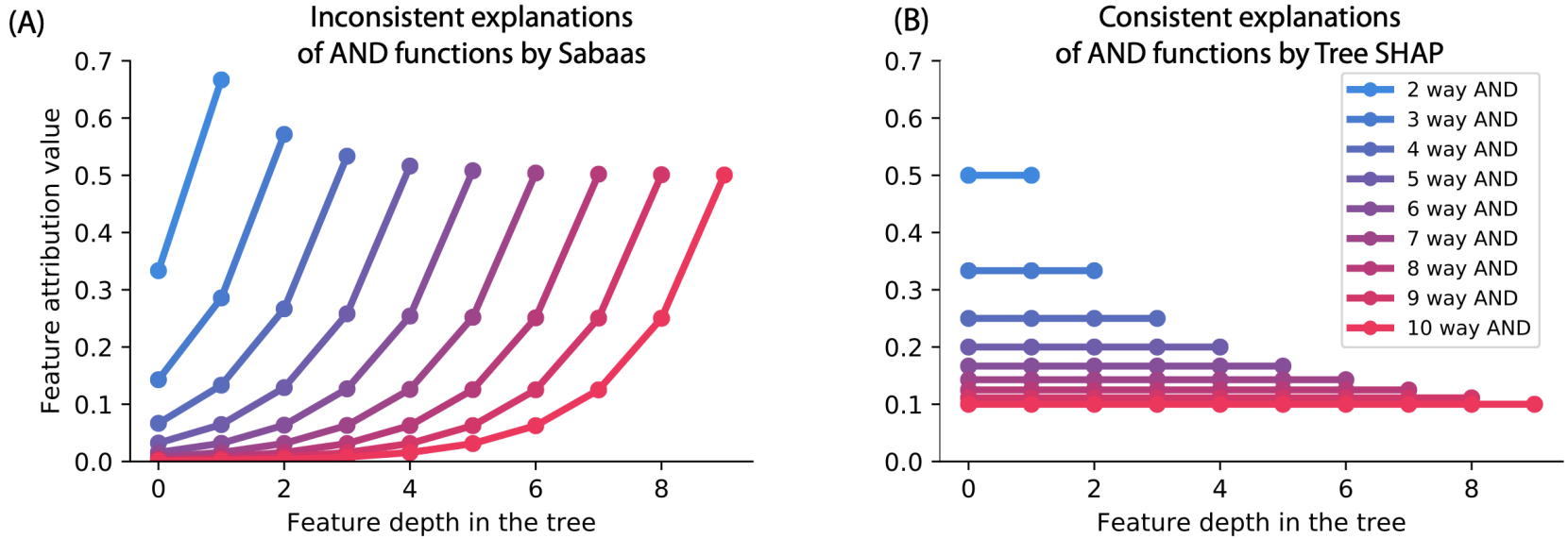
- **Missingness**: If

$$f_x(S \cup \{X_i\}) = f_x(S)$$

for all $S \in V^{-i}$, then $\Phi_f(X_i, x) = 0$.

Bias of Saabas value (bis)

"The Saabas method is biased to alter the impact of features based on their distance from a tree's root."



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.

Theorem remains valid

Theorem: Only one possible feature attribution method based on f_x satisfies properties (efficiency, consistency and missingness) :

$$\Phi_f(X_i, x) = \sum_{R \in \mathcal{R}} \frac{1}{p!} \left[f_x(P_i^R \cup \{X_i\}) - f_x(P_i^R) \right]$$

where

- $\mathcal{R}$ is the set of all features ordering,
- P_i^R is the set of all features that come before feature X_i in ordering R,
- p is the number of input features for the model.

Computing SHAP

"By focusing specifically on tree, we developed an algorithm that ***computes local explanations based on exact Shapley values in polynomial time****."*

Three algorithms

By restricting their attention to trees, it is possible to find "exact" solutions in polynomial time...

- Algorithm 1 : Easy to understand (but slow) using path-dependent feature perturbation.
- Algorithm 2: Complex polynomial time using path dependence.
- Algorithm 3: Using interventional (marginal) feature perturbation.

Note that Algorithms 1 and 2 use $f_x(S)$ approximating $E[f(X)|do(X_S = x_S)]$ while Algorithm 3 uses $f_x(S)$ that exactly equals $E[f(X)|do(X_S = x_S)]$.

Algorithm 1

Algorithm 1 Estimating $E[f(X)|do(X_S = x_S)]$

```
procedure EXPVALUE (x, S, tree = {v, a, b, t, r, d})
    procedure G(j)                                  ▷ Define the G procedure which we will call on line 1
        if v_j ≠ internal then                      ▷ Check if node j is a leaf
            return v_j                              ▷ Return the leaf's value
        else
            if d_j ∈ S then                         ▷ Check if we are conditioning on this feature
                return G(a_j) if x_{d_j} = t_j else G(b_j)   ▷ Use the child on the decision path
            else
                return [G(a_j) · r_{a_j} + G(b_j) · r_{b_j}] / r_j   ▷ Weight children by their coverage
    return G(1)                                     ▷ Start at the root node
```

- It uses the coverage information form the model about which training samples went down which path in a tree.
- Do not need to supply a background dataset
- It directly parallels the traversal used by the classic 'gain' style of feature importance (MDI).

Unifiying Saabas and Shapley values

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

Algorithm 2

Algorithm 2 Tree SHAP with path dependent feature perturbation

procedure TREESHAP_PATH (x, $tree = \{v, a, b, t, r, d\}$)
  ϕ = array of $len(x)$ zeros
  procedure RECURSE(j, m, p_z, p_o, p_i)
    m = EXTEND (m, p_z, p_o, p_i) ▷ Extend subset path with a fraction of zeros and ones
    if $v_j \neq internal$ **then** ▷ Check if we are at a leaf node
      for $i \leftarrow$ 2to $len(m)$ **do** ▷ Calculate the contributions from every feature in our path
        $w = sum(\text{UNWIND}(m, i).\ w)$ ▷ Undo the weight extension for this feature
        $\phi_{m_i} = \phi_{m_i} + w(m_i.\ o - m_i.\ z)v_j$ ▷ Contribution from subsets matching this leaf
    else
      $h, c = (a_j, b_j)$ **if** $x_{d_j} \leq t_j$ **else** (b_j, a_j) ▷ Determine hot and cold children
      $i_z = i_o = 1$
      k = FINDFIRST($m.\ d, d_j$)
      if $k \neq$ nothing **then** ▷ Undo previous extension if we have already seen this feature
        $i_z, i_o = (m_k.\ z, m_k.\ o)$
        m = UNWIND(m, k)
      RECURSE($h, m, i_z r_h / r_j, i_o, d_j$) ▷ Send both zero and one weights to the hot child
      RECURSE($c, m, i_z r_c / r_j, 0, d_j$) ▷ Send just zero weights to the cold child
  procedure EXTEND(m, p_z, p_o, p_i)
    $l, m = len(m), copy(m)$
    $m_{l+1}.\ (d, z, o, w) = (p_i, p_z, p_o, (1\ \textbf{if}\ l = 0\ \textbf{else}\ 0))$ ▷ Init subsets of size l
    for $i \leftarrow l$ to 1 **do** ▷ Grow subsets using p_z and p_o
      $m_{i+1}.\ w = m_{i+1}.\ w + p_o \cdot m_i.\ w \cdot (i/l)$ ▷ Subsets that grow by one
      $m_i.\ w = p_z \cdot m_i.\ w \cdot (l - i)/l$ ▷ Subsets that stay the same size
    return m ▷ Return the new extended subset path
  procedure UNWIND (m, i) ▷ The inverse of the ith call to EXTEND(m, …)
    $l, n, m = len(m), m_l.\ w, copy(m_{1\ldots l-1})$
    for $j \leftarrow l - 1$ to 1 **do** ▷ Shrink subsets using $m_i.\ z$ and $m_i.\ o$
      if $m_i.\ o \neq 0$ **then**
        $t = m_j.\ w$
        $m_j.\ w = n \cdot l/(j \cdot m_i.\ o)$
        $n = t - m_j.\ w \cdot m_i.\ z \cdot (l - j)/l$
      else
        $m_j.\ w = (m_j.\ w \cdot l)/(m_i.\ z(l - j))$
    for $j \leftarrow i$ to $l - 1$ **do**
      $m_j.\ (d, z, o) = m_{j+1}.\ (d, z, o)$
    return m
  RECURSE(1, [], 1, 1, 0) ▷ Start at first node with all zero and one extensions
  return ϕ

Algorithm 3

Algorithm 3 Tree SHAP with interventional feature perturbation

procedure TREESHAP_INT (x, *refset*, *tree* = {v, a, b, t, r, d})
  ϕ = array of *len(x)* zeros
  procedure CALCWEIGHT(U, V) ▷ Shapley value weight for a set size and number of features
    return $\frac{U!(V-U-1)!}{V!}$
  procedure RECURSE (j, U, V, *xlist*, *clist*)
    if $v_j \neq$ *internal* **then** ▷ Calculate possible contributions at leaf
      $pos = neg = 0$
      if $U == 0$ **then return** (*pos*, *neg*)
      if $U \neq 0$ **then** $pos = calcweight(V, U-1)^* v_j$
      if $U \neq V$ **then** $neg = -calcweight(V, U)^* v_j$
      return (*pos*, *neg*)
    $k = None$ ▷ Represents the next node
    if $(x_{d_j} > t_j)$ and $(c_{d_j} > t_j)$ **then** $k = b_j$ ▷ Both x and c go right
    if $!(x_{d_j} > t_j)$ and $!(c_{d_j} > t_j)$ **then** $k = a_j$ ▷ Both x and c go left
    if $xlist_{d_j} > 0$ **then** ▷ Feature was previously x
      if $x_{d_j} > t_j$ **then** $k = b_j$
      else $k = a_j$
    if $clist_{d_j} > 0$ **then** ▷ Feature was previously c
      if $c_{d_j} > t_j$ **then** $k = b_j$
      else $k = a_j$
    if $k \neq None$ **then** ▷ Recurse down a single path if next node is set
      return RECURSE(k, U, V, *xlist*, *clist*)
    if $(x_{d_j} > t_j)$ and $!(c_{d_j} > t_j)$ **then** ▷ Recurse x right and c left
      $xlist_{d_j} = xlist_{d_j} + 1$
      (*posx*,*negx*) = RECURSE(b_j, $U+1$, $V+1$, *xlist*, *clist*)
      $xlist_{d_j} = xlist_{d_j} - 1$
      $clist_{d_j} = clist_{d_j} + 1$
      (*posc*,*negc*) = RECURSE(a_j, U, $V+1$, *xlist*, *clist*)
      $clist_{d_j} = clist_{d_j} - 1$
    if $!(x_{d_j} > t_j)$ and $(c_{d_j} > t_j)$ **then** ▷ Recurse x left and c right
      $xlist_{d_j} = xlist_{d_j} + 1$
      (*posx*,*negx*) = RECURSE(a_j, $U+1$, $V+1$, *xlist*, *clist*)
      $xlist_{d_j} = xlist_{d_j} - 1$
      $clist_{d_j} = clist_{d_j} + 1$
      (*posc*,*negc*) = RECURSE(b_j, U, $V+1$, *xlist*, *clist*)
      $clist_{d_j} = clist_{d_j} - 1$
    $\phi_{d_j} = \phi_{d_j} + posx + negc$ ▷ Save contributions for d_j
    return (*posx* + *posc*, *negx* + *negc*) ▷ Pass up both contributions
for c **in** *refset* **do**
  RECURSE(0, 0, 0, array of *len*(x) zeros, array of *len*(x) zeros)
return ϕ/len(*refset*)

SHAP interaction values

Using Shapley interaction index, SHAP interaction values are defined as follows

$$\Phi_f(\{X_i, X_j\}, x) = \sum_{S \subseteq V^{-i,j}} \frac{|S|!(p - |S| - 2)!}{2(p-1)!} \nabla_f(\{X_i, X_j\}, x, S)$$

when $i \neq j$, and

$$\begin{aligned} \nabla_f(\{X_i, X_j\}, x, S) &= f_x(S \cup \{X_i, X_j\}) - f_x(S \cup \{X_i\}) - f_x(S \cup \{X_j\}) + f_x(S) \\ &= [f_x(s \cup \{X_i, X_j\}) - f_x(S \cup \{X_j\})] - [f_x(S \cup \{X_i\} - f_x(S)] \end{aligned}$$

- SHAP interaction value between features X_i and X_j is split equally between each feature so

$$\Phi_f(\{X_i, X_j\}, x) = \Phi_f(\{X_j, X_i\}, x),$$

- the total interaction effect is

$$\Phi_f(\{X_i, X_j\}, x) + \Phi_f(\{X_j, X_i\}, x).$$

Experiments

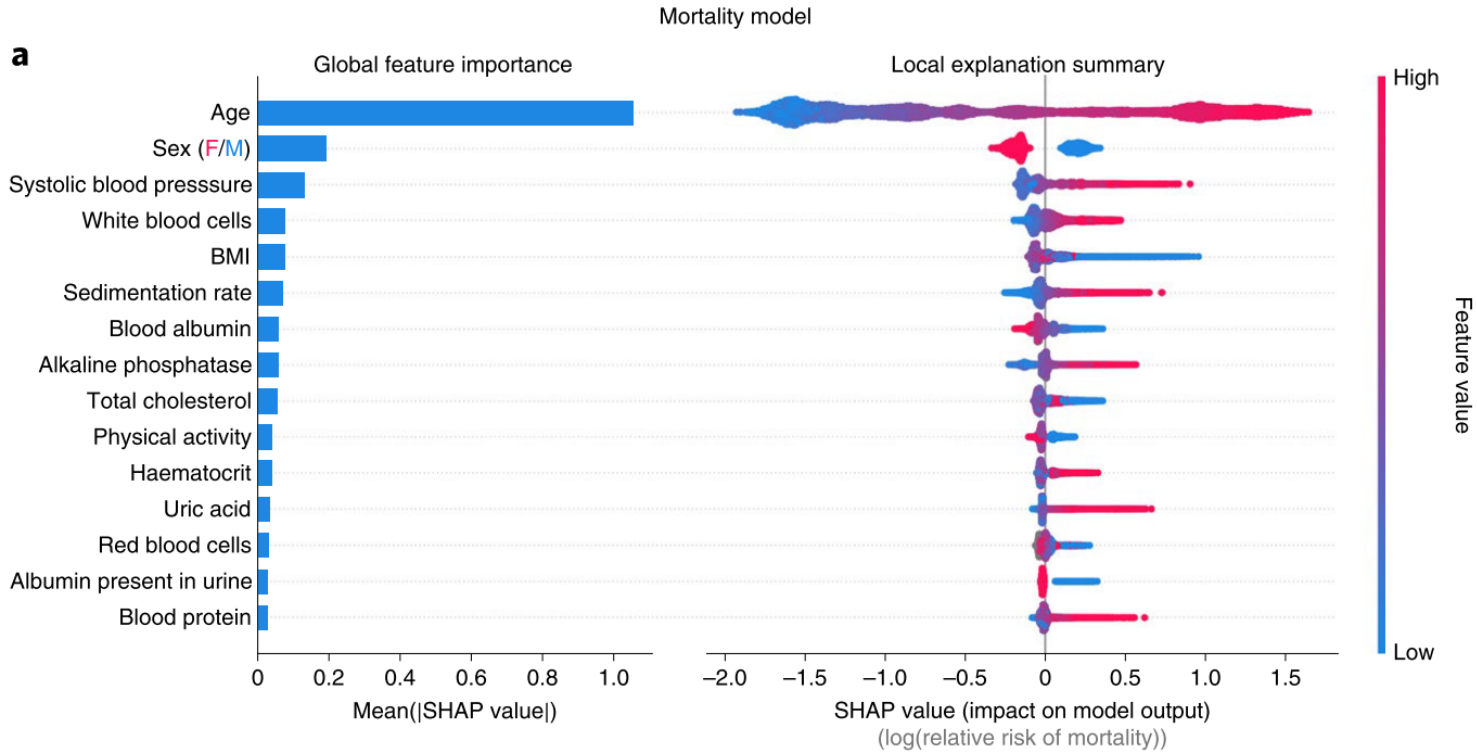


Figure: Left: bar chart of the average SHAP value magnitude. Right: a set of beeswarm plots, where each dot corresponds to an individual person in the study. The dot's position on the x axis shows the impact that feature has on the model's prediction for that person. When multiple dots land at the same x position, they pile up to show density.

Experiments

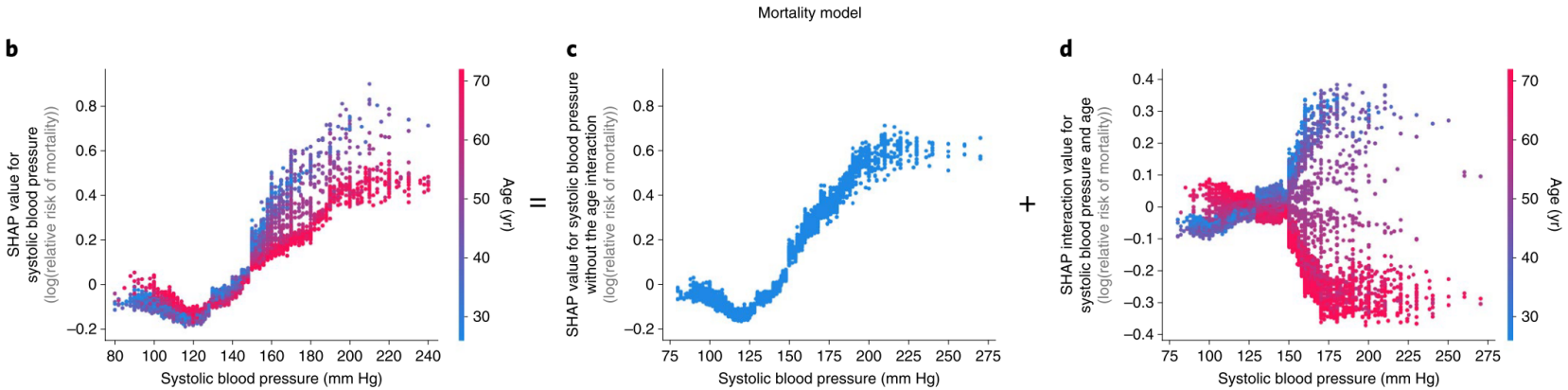


Figure: **b**, SHAP dependence plot of systolic blood pressure versus its SHAP value in the mortality model. A clear interaction effect with age is visible, which increases the impact of early onset high blood pressure. **c**, The residual after subtracting the SHAP interaction values for age from the model. **d**, Plot of just the interaction effect of systolic blood pressure with age shows how the effect of systolic blood pressure on mortality risk varies with age. Adding the y values of **c** and **d** produces **b**.

Experiments

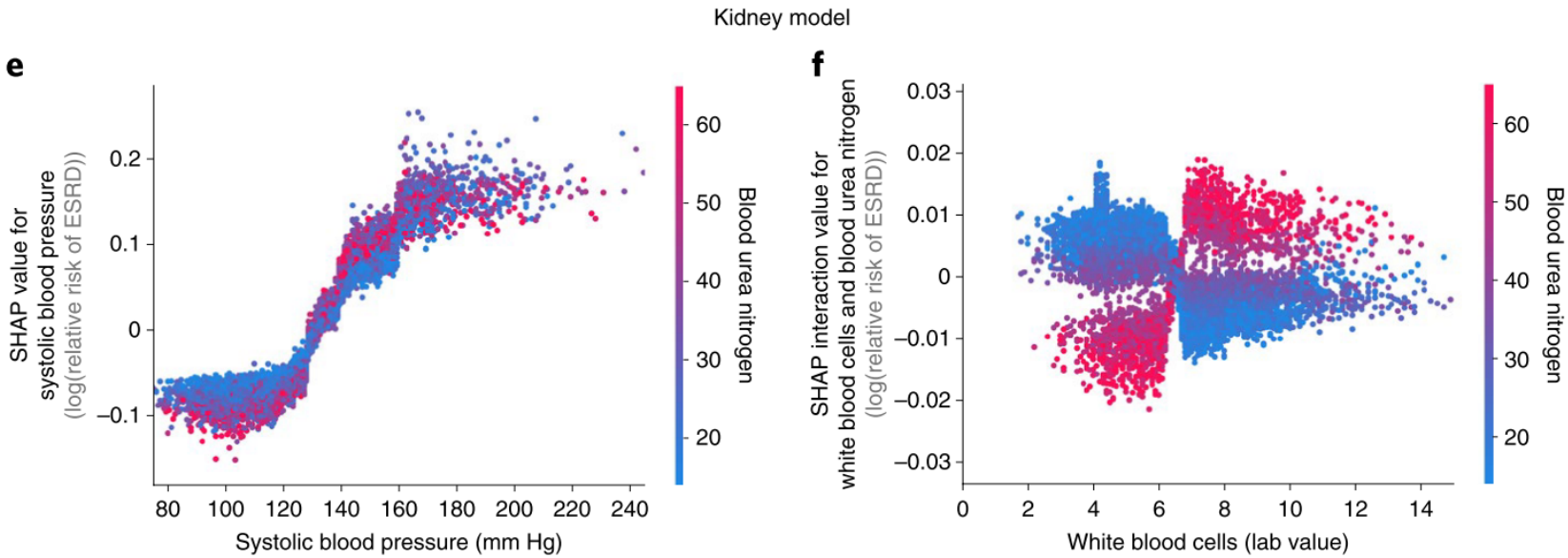


Figure: **e**, A dependence plot of systolic blood pressure versus its SHAP value in the kidney model shows an increase in kidney disease risk at a systolic blood pressure of 125 (which parallels the increase in mortality risk). **f**, Plot of the SHAP interaction value of 'white blood cells' with 'blood urea nitrogen' shows that high white blood cell counts increase the negative risk conferred by high blood urea nitrogen for progression to end stage renal disease (ESRD).

Experiments

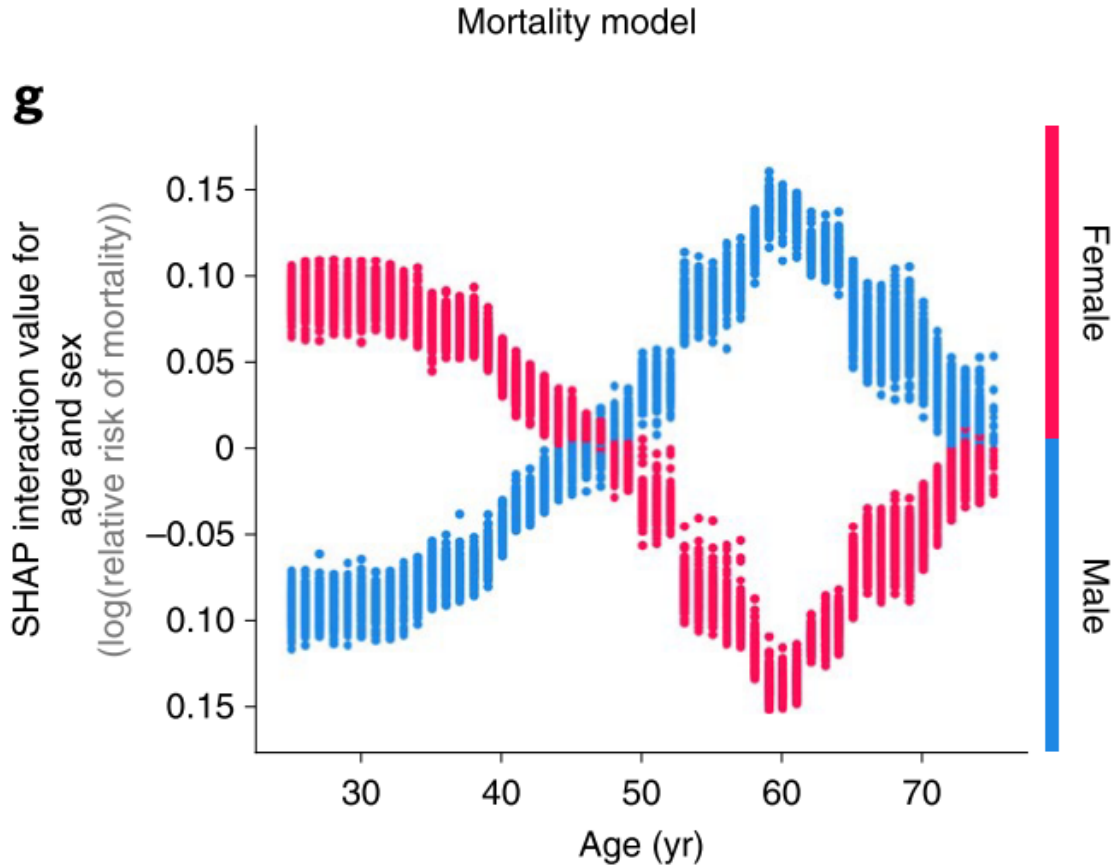
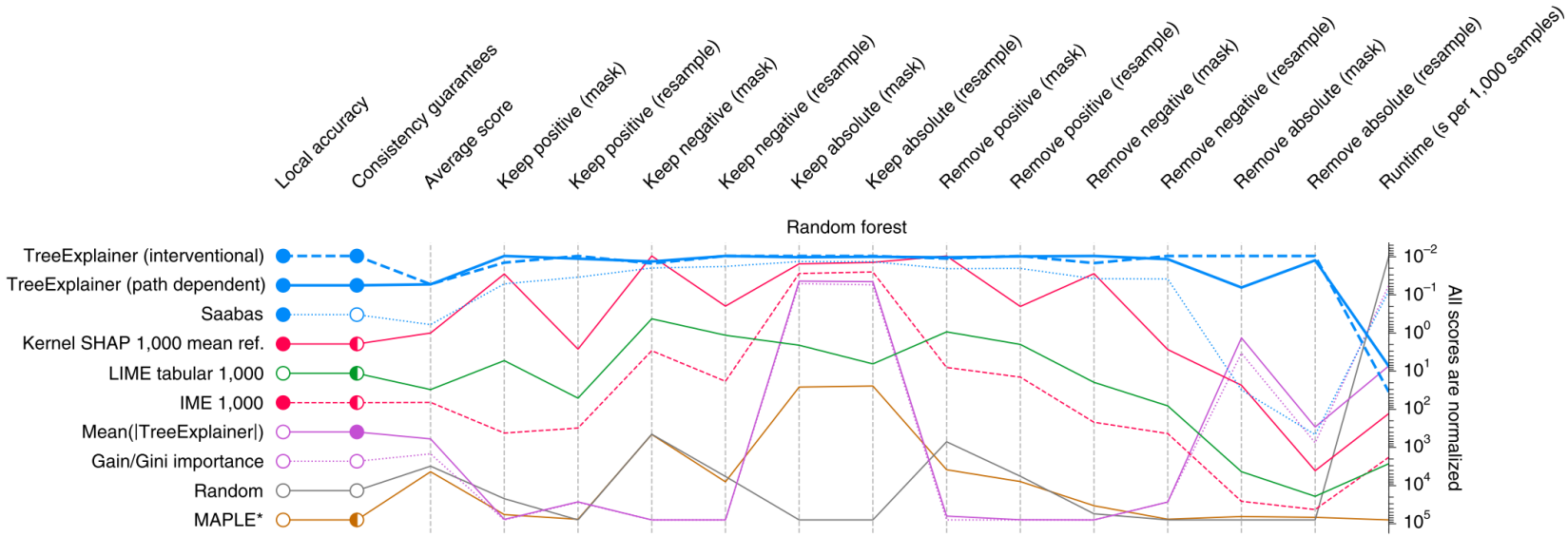


Figure: **g**, Plot of the SHAP interaction value of sex versus age in the mortality model shows how the differential risk for men and women changes over a lifetime.

Comparison with other methods



Discussion

Problems with Shapley-value-based explanations as feature importance measures

I. Elizabeth Kumar[u], *Suresh Venkatasubramanian*[u], *Carlos Scheidegger*[a], *and Sorelle Friedler*[h]

[u]{*kumari, suresh*}*@cs.utah.edu, University of Utah*
[a]*cscheid@cs.arizona.edu, University of Arizona*
[h]*sorelle@cs.haverford.edu, Haverford College*

References I

Genuer, R., Poggi, J.-M., and Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

References II

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Young, H. P. (1985). Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72.

Also thanks to Pierre Geurts and Vân Anh Huynh-Thu.