

Black Box Model Explainability

Data Sciences Institute
Topics in Deep Learning

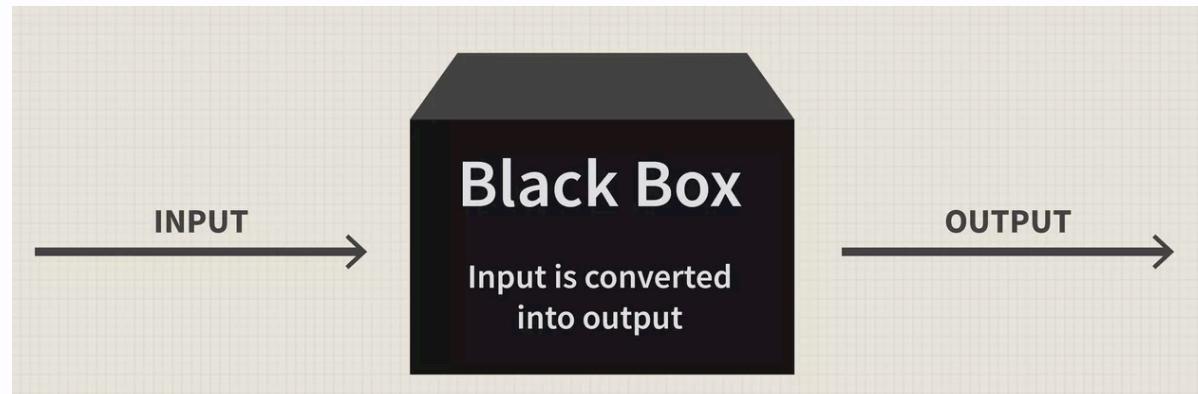
Outline

- Black box models
- White box models
- Local methods
 - ICE, SHAP, & LIME
- Global methods
 - Global surrogates, Selective Inference, Permutation, Knockoffs, HRT, DML
- Architecture specific methods
 - Tree-based, GradCAM, Attention

Black Box Models

What is a black box?

- In general parlance, a "black box" refers to some function whose internal workings are unknown or opaque
- Like a machine learning model, we assume a black box maps an input to an output: $f(x) = y$, where $f : \mathbb{R}^p \rightarrow \mathbb{R}^k$

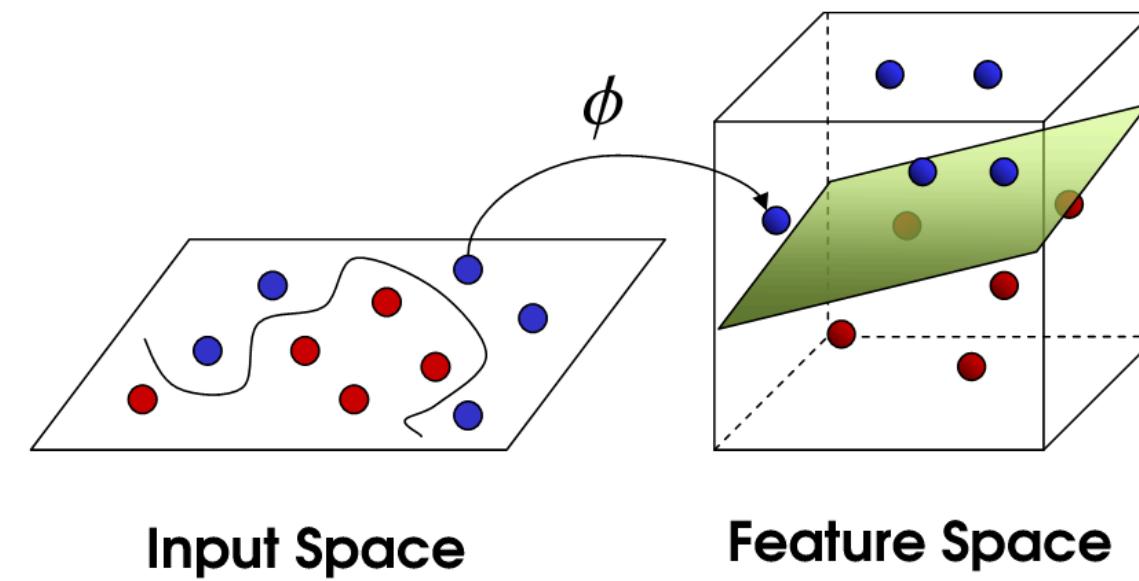


What is a black box in ML?

- In machine learning, we use "black box" to refer to:
 - Algorithm classes "whose internal workings are unknown or opaque"
 - Methods that work for any arbitrary function (i.e. as long as you can perform inference or possibly take a gradient from $f_\theta(x)$)
- Questions:
 - What are some ML algorithms you would consider "black boxes"?
 - What are some methods that work for arbitrary functions?

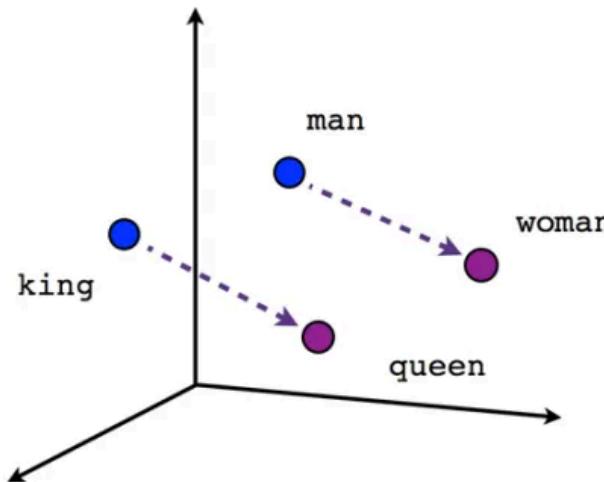
Why use a black box model?

- Flexibly model arbitrary functions

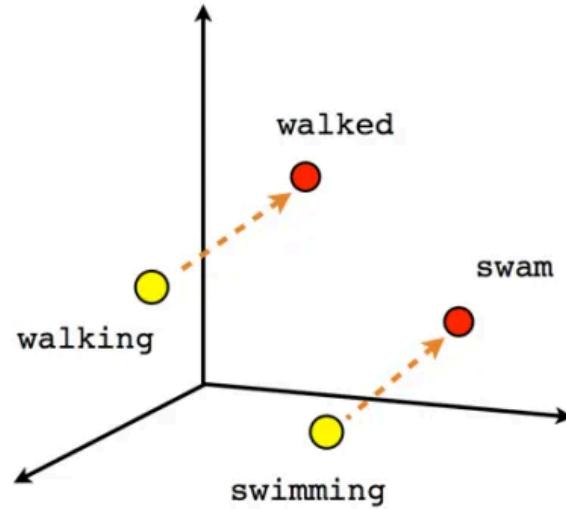


Why use a black box model?

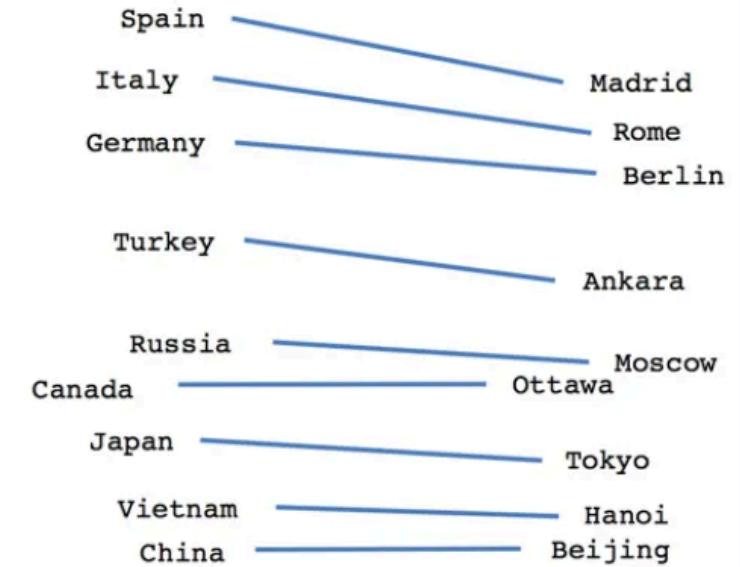
- Learned features >> hand-crafted features (usually)



Male-Female



Verb tense

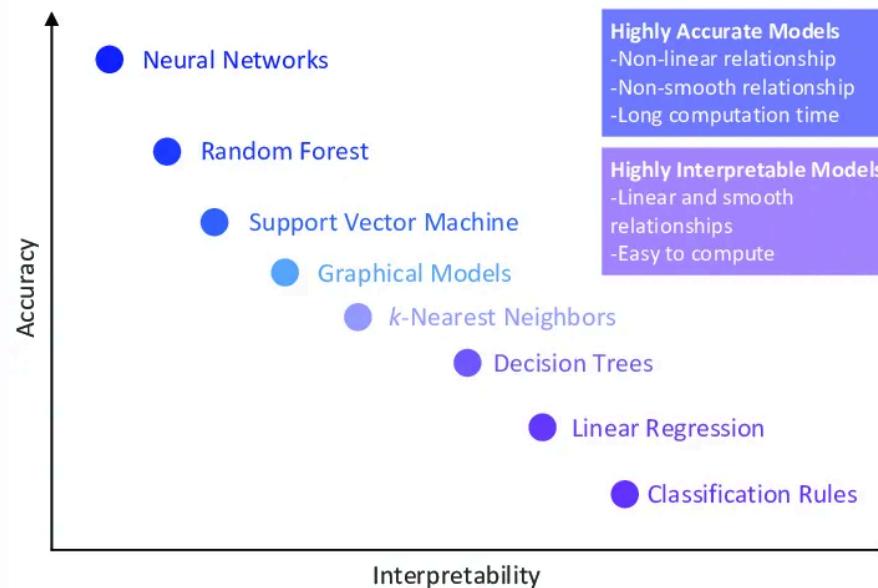


Country-Capital

See: [Word2vec](#)

Complexity vs interpretability

- Many ML models (e.g. DL models) are capable of modeling highly complex, non-linear relationships
- There is usually (but not always) a trade-off between model complexity and performance



Source: Morocho-Cayamcela et. al (2019)

Challenges of black box models

- **Lack of transparency:** how do models generate their predictions?
- **Poor interpretability:** given a certain input, why was a particular prediction made?
- **Trust issues:** stakeholders may find it challenging to trust a well-performing model that they cannot understand
- **Accountability issues:** if a black box model's behaviour results in serious issues such as death, who should be held accountable?

Importance of explainability

- Explainability is crucial for models deployed in high-stakes environments such as healthcare
- The more we understand a model, the more we can:
 - Build trust among stake-holders
 - Foster ethical AI practices
 - Ensure regulatory compliance
 - Facilitate and contextualise model debugging and improvement

Lesson objective

- Explore different methods for elucidating understanding how complex, non-linear models work

White box models

White box models

- Several important classes of machine learning models are "naturally interpretable" to humans and do not require black box explainers
 - Note: A natural sanity check for a black box explainer is to compare its interpretations to a linear model
- We'll discuss two classes of explainable models
 - Linear models*
 - Decision trees*

**Even these methods can become uninterpretable from a human's perspective if there are too many covariates or too much depth*

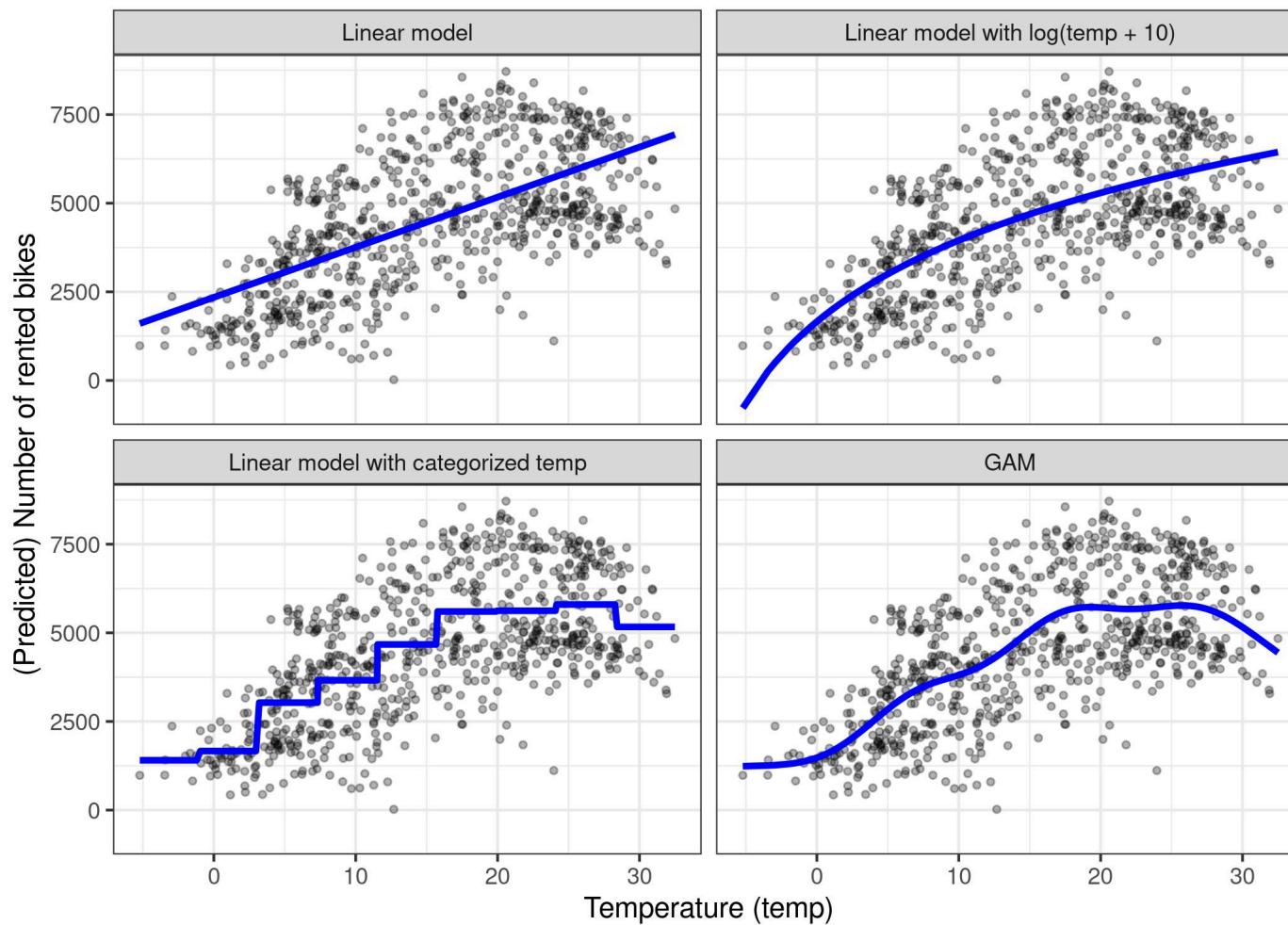
Linear models

- We'll consider any model class that can be written as (1) to be a linear model

$$g(E[y|x]) = f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \text{interactions} \quad (1)$$

- Where $g(E[y|x])$ is the inverse link function (e.g. log-odds)
- f_k is some function of a single covariate (e.g. identity, square, etc)
- Where "interactions" are a combination of two features (e.g. $x_1 \cdot x_4$)

Linear models



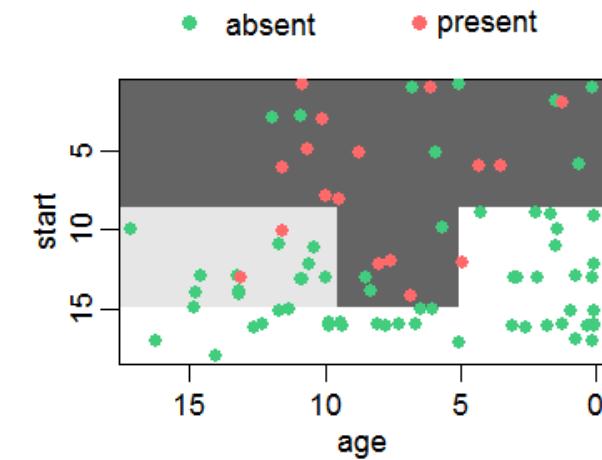
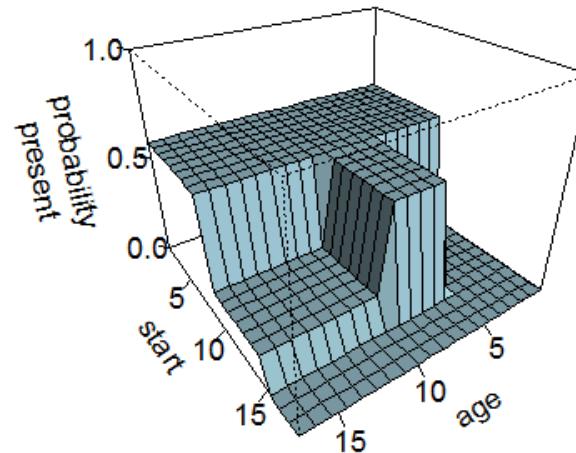
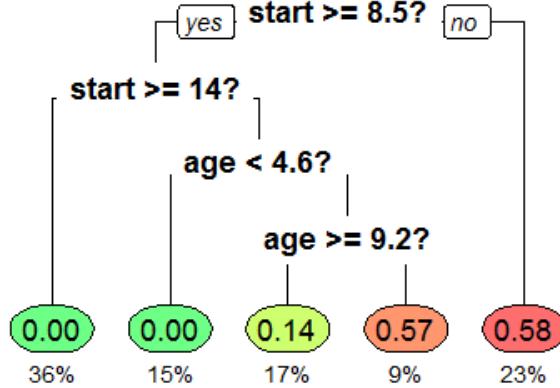
Source: Molnar (2023)

Breakout #1

What is the difference between $f(x_1, \dots, x_p)$ and $f_1(x_1) + \dots + f_p(x_p)$? Which one does a 1-layer NNet describe?

Decision trees

- Are a class of supervised ML which recursively partitions the feature space so that the terminal leaves minimizes entropy (classification) or variance (regression)
 - The CART algorithm which "learns the tree" with data is inherently greedy which is why there are various rules for early stopping



Local Methods



Why is this person at high risk of suffering from cardiovascular disease?

Understanding individual predictions

- **Local explainability methods** (aka "instance level explanations") offer insights into individual predictions made by black box models
 - They focus on explaining why a particular prediction was made for a specific instance or region of the input space

Methodological approaches

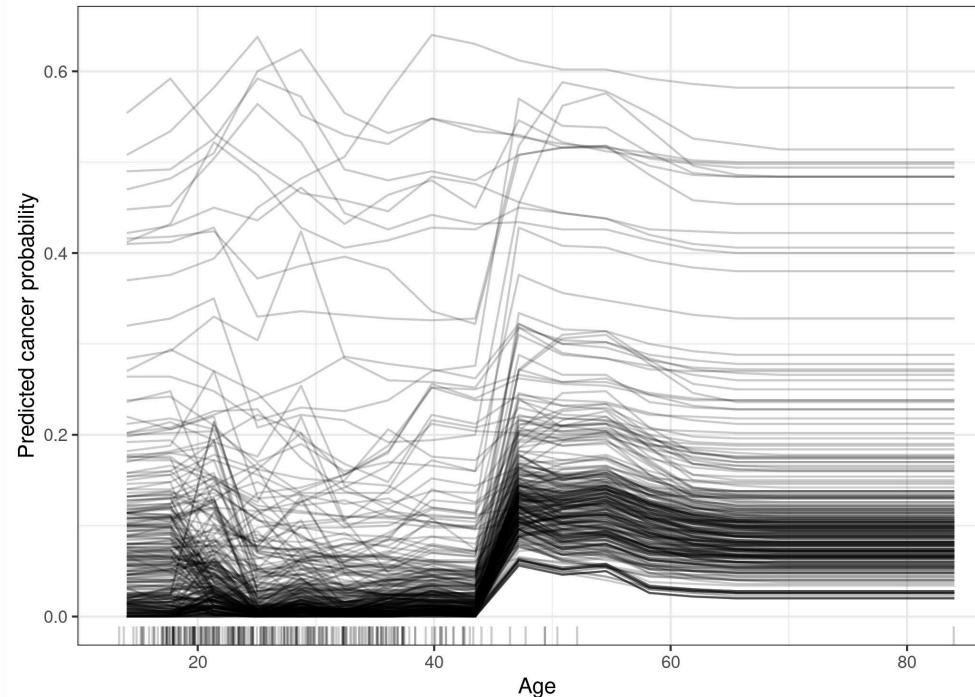
In this lesson, we will go over three different approaches for local explainability:

- 1. Individual Conditional Expectation (ICE):** How does an individual instance's prediction change when a single variable changes at a time?
- 2. Variable attribution (SHAP):** how can differences in the prediction level be attributed among the input variables?
- 3. Surrogate models of behaviour (LIME):** can we model black box behaviour locally using an easy to interpret white box model?

Variable Attribution: ICE

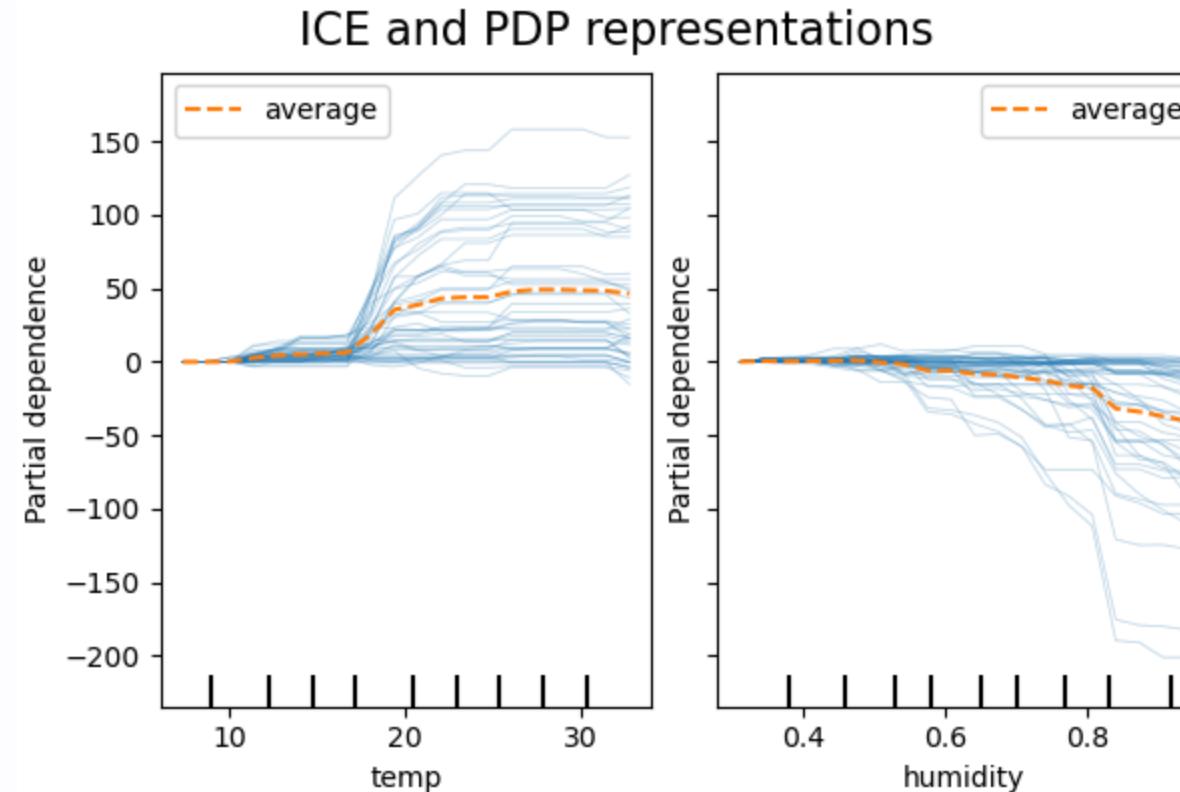
Individual Conditional Expectation (ICE)

- ICE shows how an instance's prediction changes when a feature changes:
- If $x \in \mathbb{R}^p$, $x'_i = (x_{i1}, x'_2, x_{i3}, \dots, x_{ip})$ and we compare $f_\theta(x'_i)$ for $x'_2 \in \mathbb{R}$ to see how the prediction changes for instance i when x_2 changes



Individual Conditional Expectation (ICE)

- ICE can be turned into a "global" explainer by averaging over the individual curves (this is known as a partial dependency plot (PDP))
- ICE works for all black box models that we can run inference on (no gradients needed)
- It is a form of "mechanistic interpretability" (example below: [sklearn](#))



Variable Attribution: SHAP

SHapley Additive exPlanations (SHAP)

- SHAP is a method for explaining the variation in the output of machine learning model's predictions
- This is based on the concept of Shapley values from cooperative game theory: given a set of players (features), how do we distribute the payout (prediction) resulting from a collaborative game (prediction task)

Calculating variable contribution

- For a given feature j and instance i , that SHAP value (ϕ_{ij}) is:

$$\phi_{ij} = \phi_j(f, x_i) = \sum_{z' \subseteq x'_i} \frac{|z'|(|M| - |z'| - 1)}{M!} [f_x(z') - f_x(z' \setminus j)]$$

- This represents a subset of the 2^p calculations needed to do an exact calculation
- Contributions are anchored to:

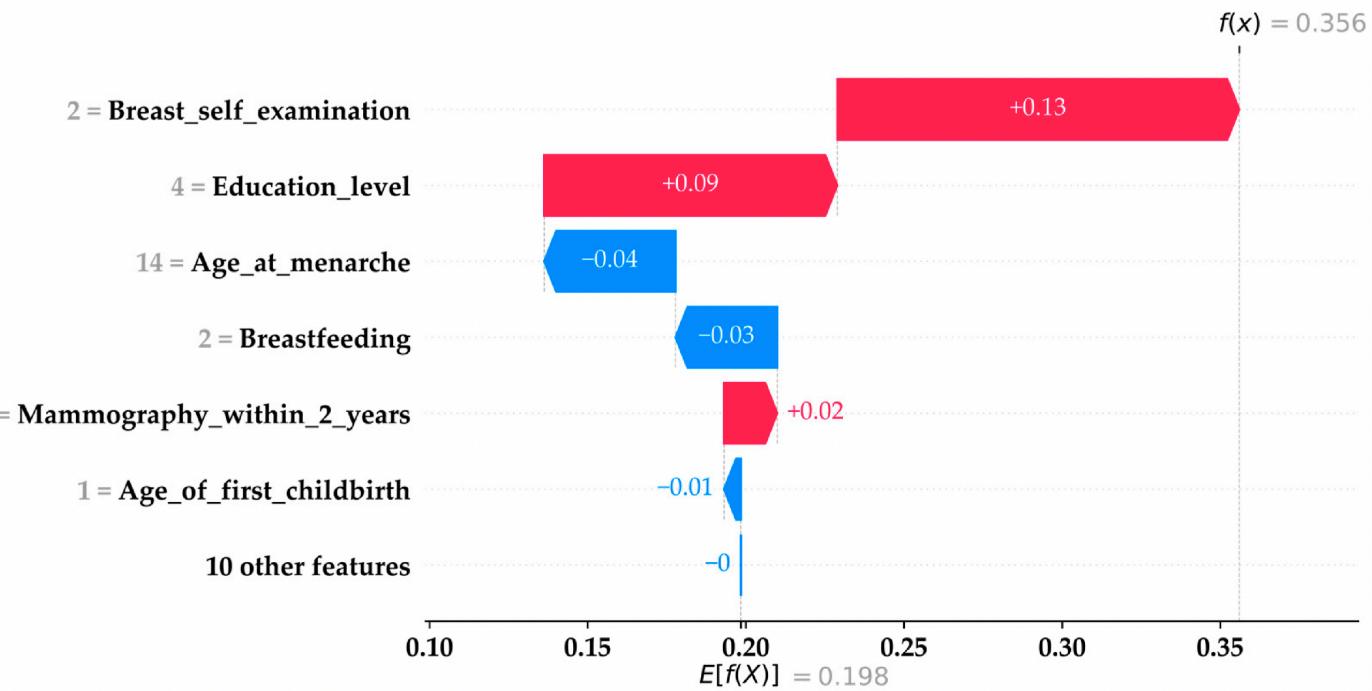
$$f(x_i) = E[f(x)] + \sum_{j=1}^p \phi_j(f, x_i)$$

Source: Lundberg & Lee (2017)

Calculating variable contribution

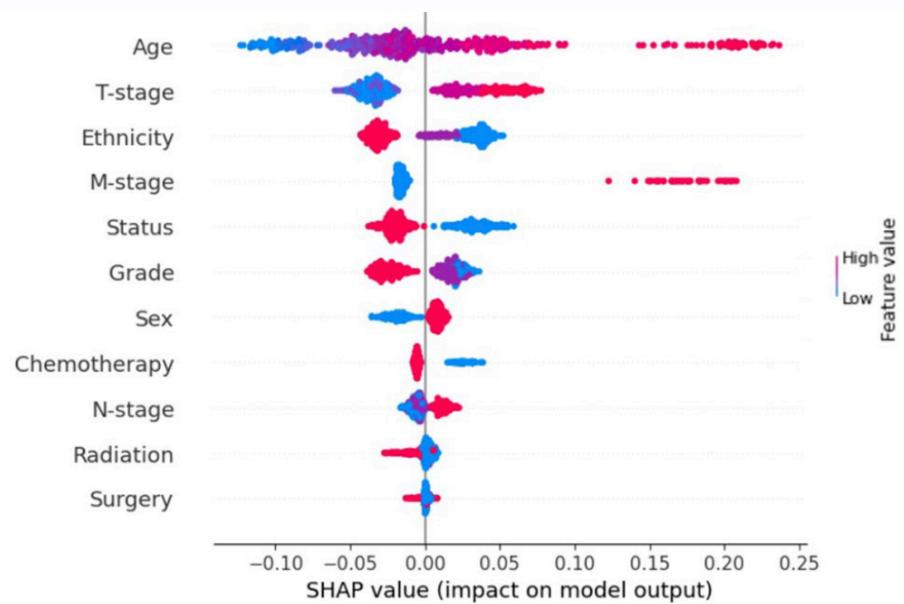
- In practice, we never calculate the 2^p combinations
- Instead we use sampling methods
- While there are algorithm-specific SHAP approaches, the default is KernelSHAP
 - Fit a linear model to the samples inference where \hat{y} is the predicted value, the variables (Z) are binary indicators (feature present or not) and the estimated coefficients will be the SHAP values.
 - i.e. regression of $\hat{y} = Z\Phi$
- But how do we run: $f_x(z')$ if $z \in \mathbb{R}^q$ where $q < p$?

SHAP waterfall plot: mammography prediction



Source: Sun et. al (2023)

SHAP bee swam plot: Cancer survival



Source: Alabi et. al (2023)

Interpreting SHAP values

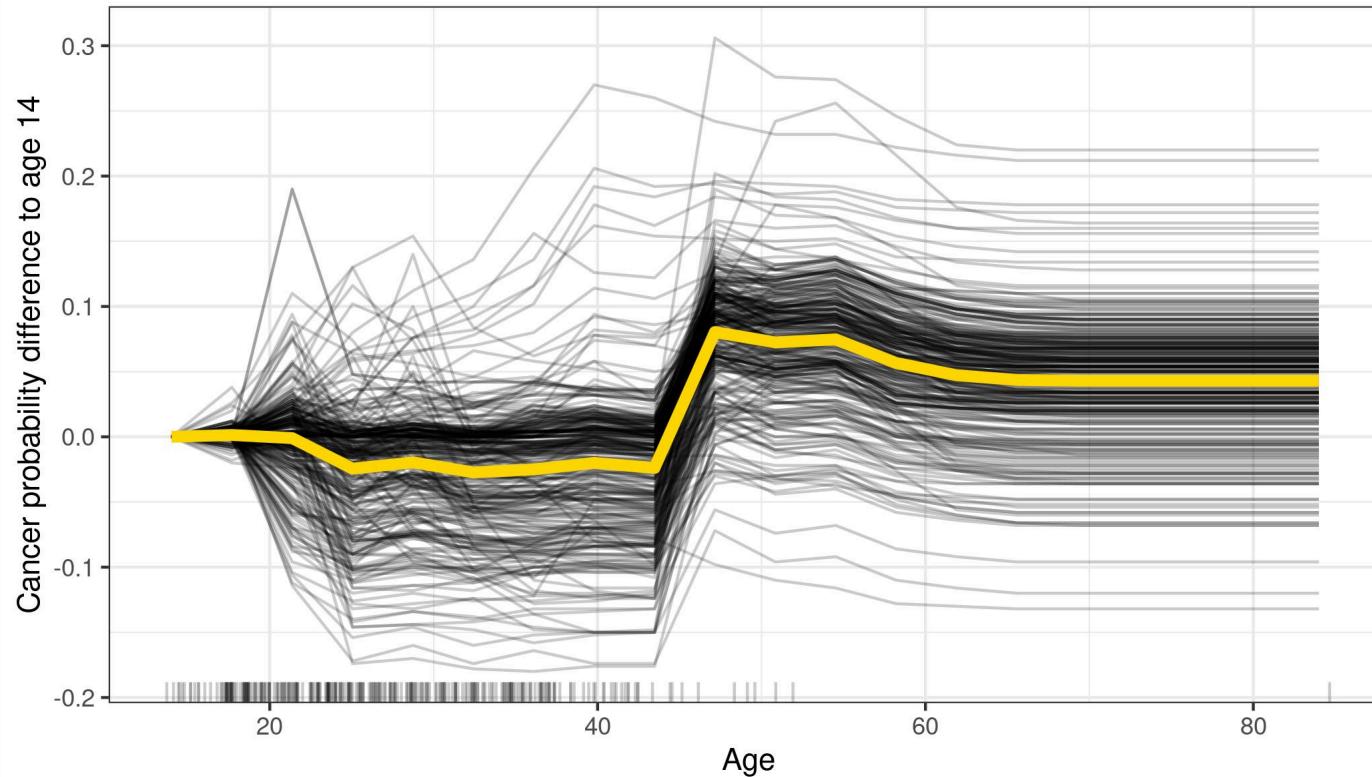
- **Sign:** positive SHAP values indicate that a feature value increases the prediction, while negative values indicate a decrease
- **Magnitude:** the magnitude of the SHAP value represents the importance or impact of that feature value on the prediction
- **Additive property:** the sum of SHAP values across all features equals the difference between instance and average predictions
- **Visual interpretation:** SHAP values can be visualized using various plots, such as the waterfall plot, which displays how individual feature values push the prediction of an instance away from the average value

Limitations of SHAP

- **Computationally expensive:** considering all coalitions can be computationally intensive, especially in complex contexts
- **Assumption of independence:** considering all possible coalitions equally may does not reflect feature interdependence, which indicate that certain coalitions are more likely than others in real life
- **Potential misinterpretation:** Users may sometimes misinterpret SHAP values, assuming causality, leading to false conclusions

Question

- Why would the SHAP value ϕ_{ij} be different than the difference between the ICE and the PDP?



Plot: ICE curves (black lines), PDP curve (yellow line)

Source: Molnar (2023)

Breakout #2

How would you create a "variable importance" based on the SHAP values? How would you explain this metric to a data science expert and non-expert?

Surrogate models: LIME

Local Interpretable Model-agnostic Explanations (LIME)

- LIME is a technique for explaining individual predictions of black box machine learning models at a local level
- It approximates the behavior of the black box model by training interpretable surrogate models on perturbed instances around the prediction of interest
- LIME provides insights into why a specific prediction was made by highlighting the contribution of different features for that instance

LIME's philosophy

- The LIME model is based on three "Desired Characteristics for Explainers":
 - Interpretable
 - Locally faithful
 - Model-agnostic
- An "interpretable" model class $g(x')$ is used to approximate $f(x)$ (where x' is an interpretable feature space of x), where $\ell(\cdot)$ is the error between them and π_x defines what is "local"

$$\xi(x) = \ell(f, g, \pi_x) + \Omega(g)$$

Mechanistic overview

Given an original instance of interest, LIME does the following:

1. Sample multiple new instances around the original neighbourhood
2. Weight each new instance according to their proximity to the original
3. Train a simple, interpretable model (such as linear regression) on the neighbourhood data
4. Explain the original instance's prediction by interpreting the surrogate model

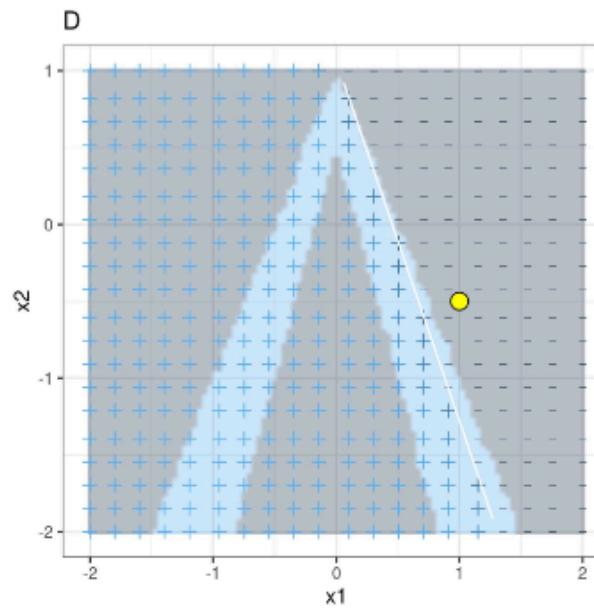
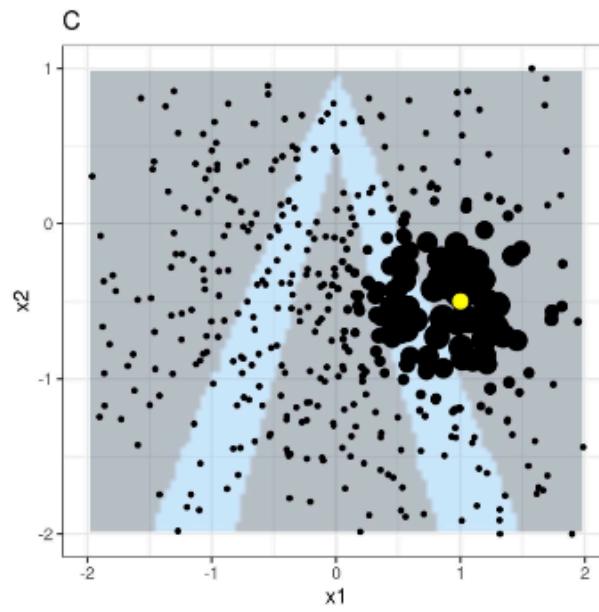
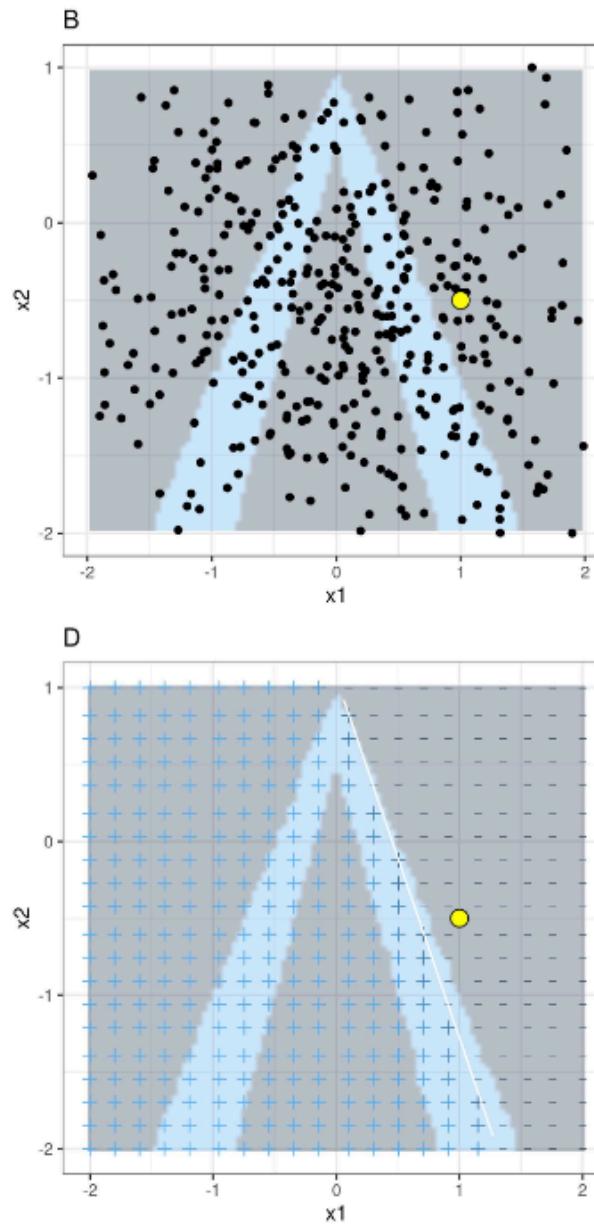
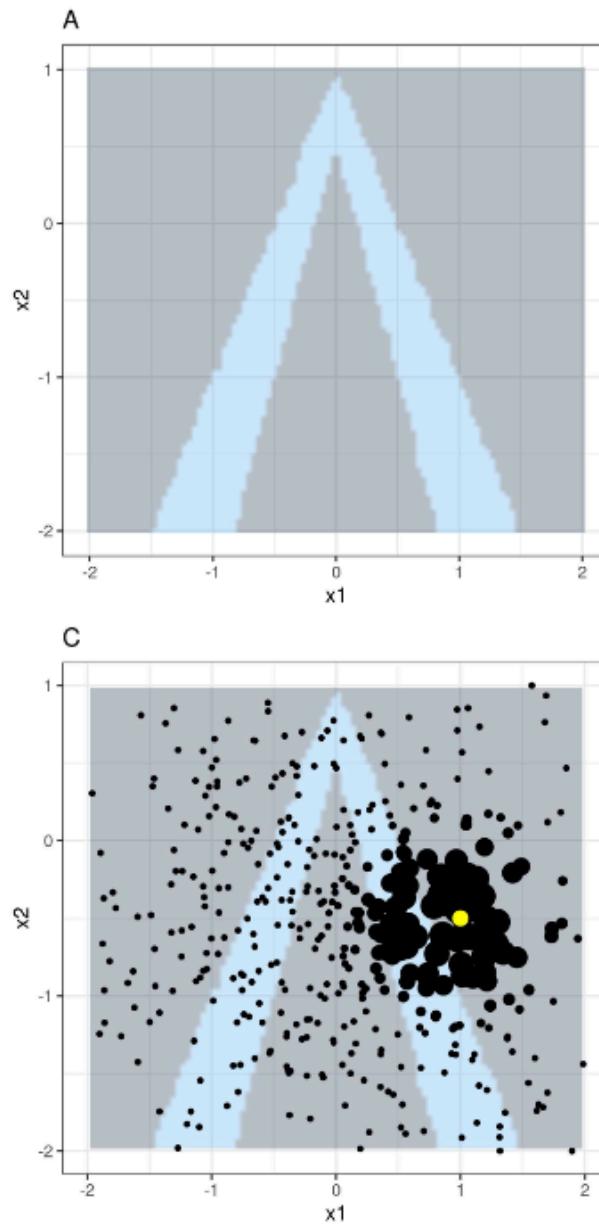
Mechanistic overview

- Loss functions could include:

$$\ell(f, g, \pi_x) = \sum_{z \in \Pi_X} \pi_x(z) [f(z) - g(z)]^2$$

- Or if we were willing to "classify" our points ($\hat{y} = I[f(z) > t]$):

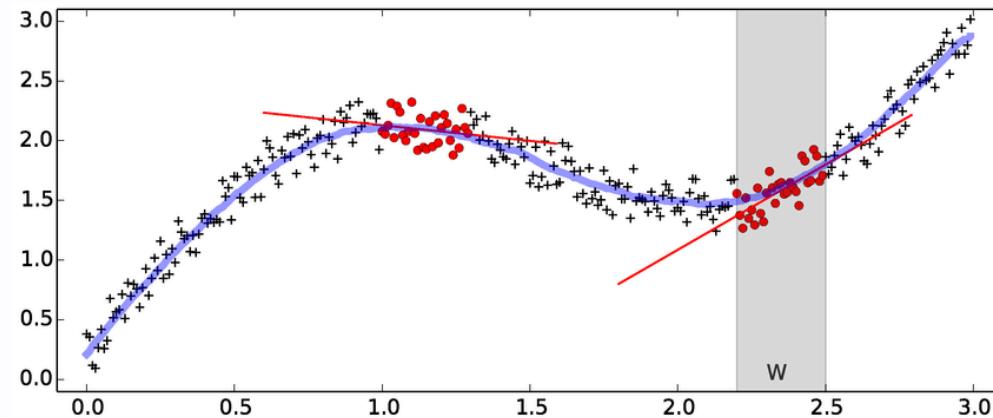
$$\ell(f, g, \pi_x) = - \sum_{z \in \Pi_X} \pi_x(z) [\hat{y} \log(g(z)) + (1 - \hat{y}) \log(1 - g(z))]$$



- A. Black box model predictions given features x_1 and x_2
- B. Instance of interest (big yellow dot) and data sampled from a normal distribution (small dots)
- C. Assign higher weight to points near the instance of interest.
- D. Signs of the grid show the classifications of the locally learned model from the weighted samples. The white line marks the decision boundary ($P(\text{class}=1) = 0.5$)

Some similarity to a local linear regression (LOESS)

- Local linear regression fits a linear model for every new instance
 - $\arg \min_{\alpha, \beta} \sum_{i=1}^n K_\lambda(x, x_i)[y_i - \alpha - x_i^T \beta]^2$



Limitations of LIME

- **No single correct way of defining a neighbourhood:** the reweighing function used for new sampled instances based on their distance from the original is variable and can have important impacts in downstream results
- **Generation of unlikely samples:** sampling a neighbourhood by using a normal distribution around the instance of interest may generate samples that wouldn't exist in real data, leading to surrogate models that do not adequately represent the real underlying data distribution
- **Model dependence:** interpretability results heavily depend on the choice of both the black box model and the surrogate model

Global Methods



Which risk factors are generally associated with higher risk of suffering from cardiovascular disease?

Understanding overall model behaviour

- **Global explainability methods** offer insights into what features drive model performance and prediction

Why does it matter?

- Global explainability enhances our general understanding of a model's decision-making process across an entire dataset, enhancing methodological transparency and increasing trust amongst stakeholders
- It also facilitates model debugging and improvement by identifying unexpected behaviours and potential areas of improvement, such as feature selection

Methodological approaches

In this lesson we will go over two different global explainability approaches:

- **Global surrogates**
- **Selective Inference**
- **Knockoffs**
- **Permutation tests**
- **Holdout Randomization Test (HRT)**
- **Double machine learning (DML)**

Global Surrogates

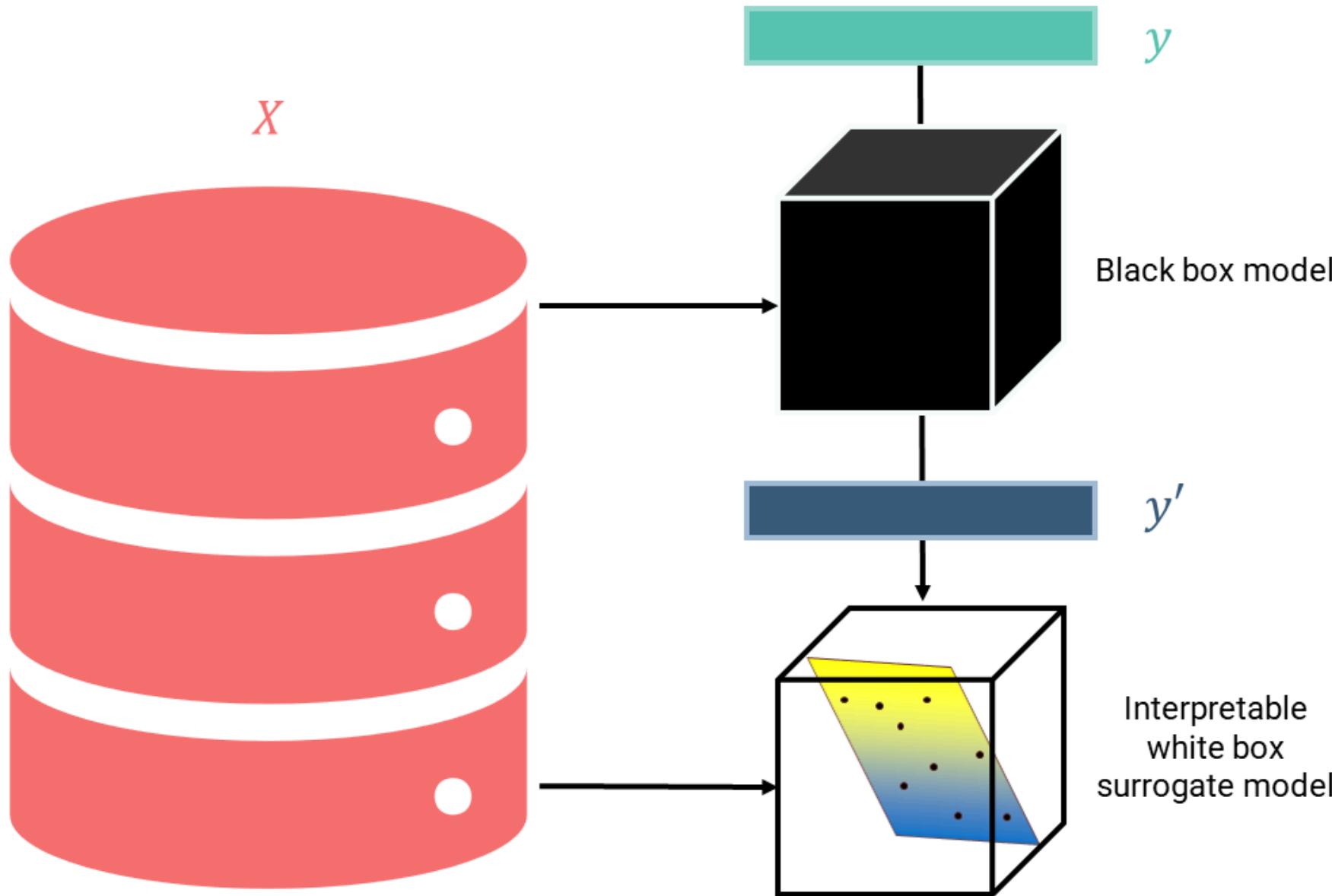
Global surrogate models

- A global surrogate is a simple, interpretable model (e.g., linear regression or decision tree) trained to approximate the predictions of a black box model
 - We are using simple machine learning to model the behaviour of more complex DL algorithms

Basic principle

A global surrogate model can be obtained and interpreted as follows:

1. Define a dataset X
2. Obtain prediction outputs of X using the black box model
3. Select and train an interpretable model using X as input and the black box predictions as output
4. Measure how closely the predictions of both models align
5. Interpret the surrogate model (e.g., which features have the most important coefficients in linear regression)



Interpretable
white box
surrogate model

y

y'

X

Black box model

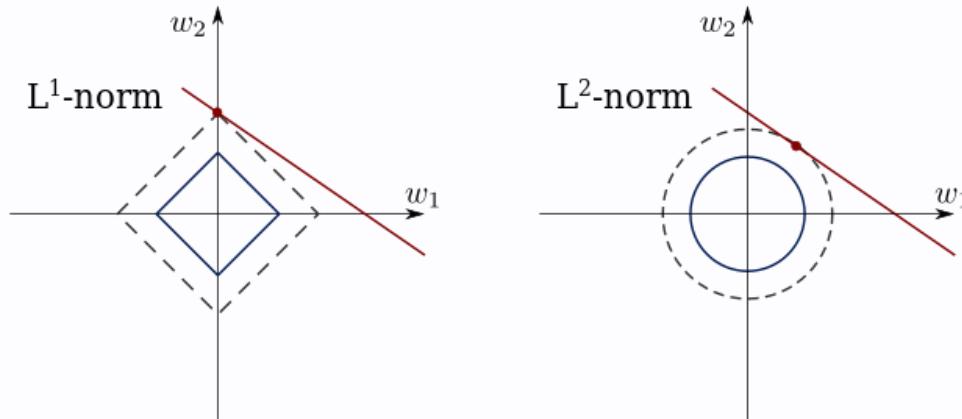
Limitations

- **Misinterpretation:** the insights gained from global surrogate models are related to model behaviour, **NOT** to the characteristics of the data itself
- **Susceptibility to choice of training data for surrogate model:** the surrogate model can be trained in any data of similar distribution to that used by the black box model in training. It can happen that surrogate models can model black box behaviour better for some data subsets than others
- **How good is good enough?:** there are no clear rules to determine how similar the surrogate model predictions have to be to its black box counterpart to be considered an acceptable approximation of behaviour

Selective inference (for high dimensional linear models)

Selective inference (overview)

- For high dimensional datasets, there is a reasonable chance that only a small number of variables matter ("sparsity hypothesis")
- We use algorithms like the Lasso, ElasticNet, and stepwise regression to select a subset of covariates



Source: [Wikipedia](#)

Selective inference (overview)

- But can do hypothesis testing on the results of these algorithms?
 - This does not align with the "classical" framework of statistical hypothesis testing

Textbook practice

- (1) Select hypotheses/model/question
- (2) Collect data
- (3) Perform inference

Modern practice

- (1) Collect data
- (2) Select hypotheses/model/questions
- (3) Perform inference

Source: Candes (2017)

Selective inference (overview)

- The short answer is NO!
 - P-values will not be uniform (or even conservative) when the null is true
 - There will be massively inflated type-I error rate

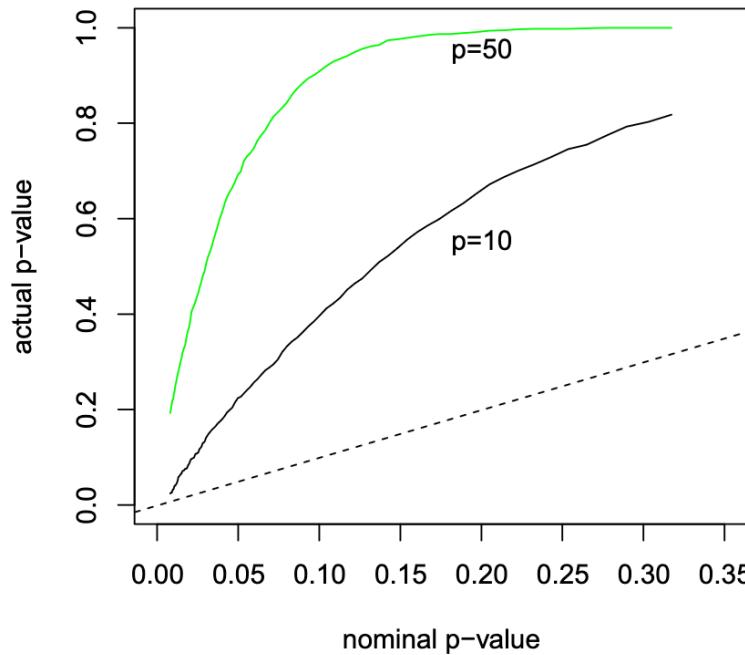


Fig. 1. A simulation example with $N=100$ observations and $p=10$ or 50 predictors and null effects ($\beta=0$). Shown are the nominal and actual P values for entering the strongest predictor at the first step; the 45° dashed line is included for reference.

Source: Taylor and Tibshirani (2015)

Selective inference (overview)

- This problem of HARK'ing (hypothesizing after results are known), is part of the reason why we have the reproducibility crisis

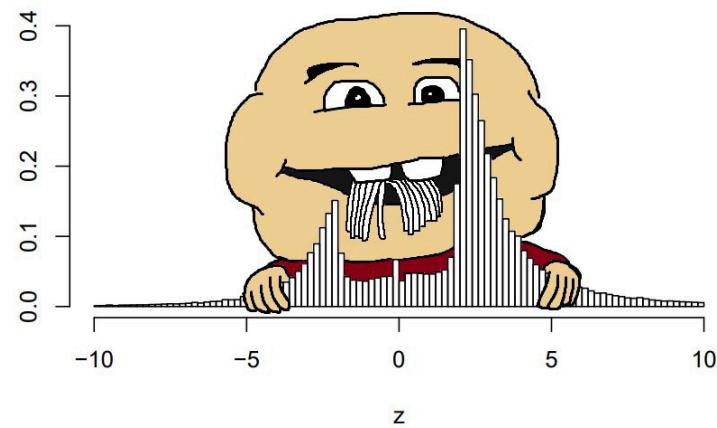
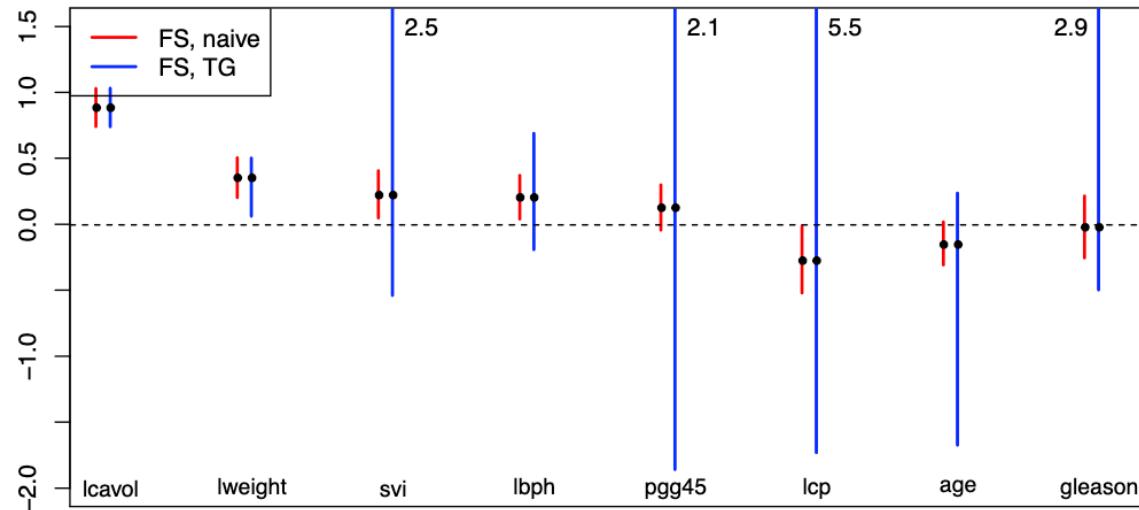


Figure 1: The distribution of more than one million z -values from Medline (1976–2019).

Source: [Ranking Fields by p-Value Suspiciousness](#)

Selective inference (overview)

- Instead, in selective inference we wish to test:
 - $P(\text{parameters}|\text{data, selection event})$
- We can do this for many classes of algorithms and get "corrected" p-values and confidence intervals
 - Lasso, ElasticNet, Stepwise Regression



Source: Prostate cancer dataset, from [Recent Advances in Selective Inference](#)

Selective inference (example)

- Suppose we have a high-dimensional dataset (e.g. GWAS), and we're modelling a response a linear combination of variables:
 - $g(E[y_i|x_i]) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$
 - $i \in \{1, \dots, n\}$, where $p \gg n$
- We use some sort of "sparsity inducing algorithm" (e.g. Lasso)
- We get back a new model: $g(E[y_i|x_i]) = \beta_0 + \beta_2 x_{i1} + \beta_{12} x_{i12} + \beta_{39} x_{i39}$
- We now want to test: $H_0 : \beta_{12} = 0$
- We get back an "explainable" linear model with associated p-values and CIs

Permutation Tests

Permutation Tests

- Given a trained model and a held out test set, permutation testing repeatedly evaluates model performance on the test set following individual feature perturbations
 - Measuring the impact of these perturbations on model predictions serves as a proxy of overall feature importance
- These measures provide insights into feature interactions and overall model behaviour

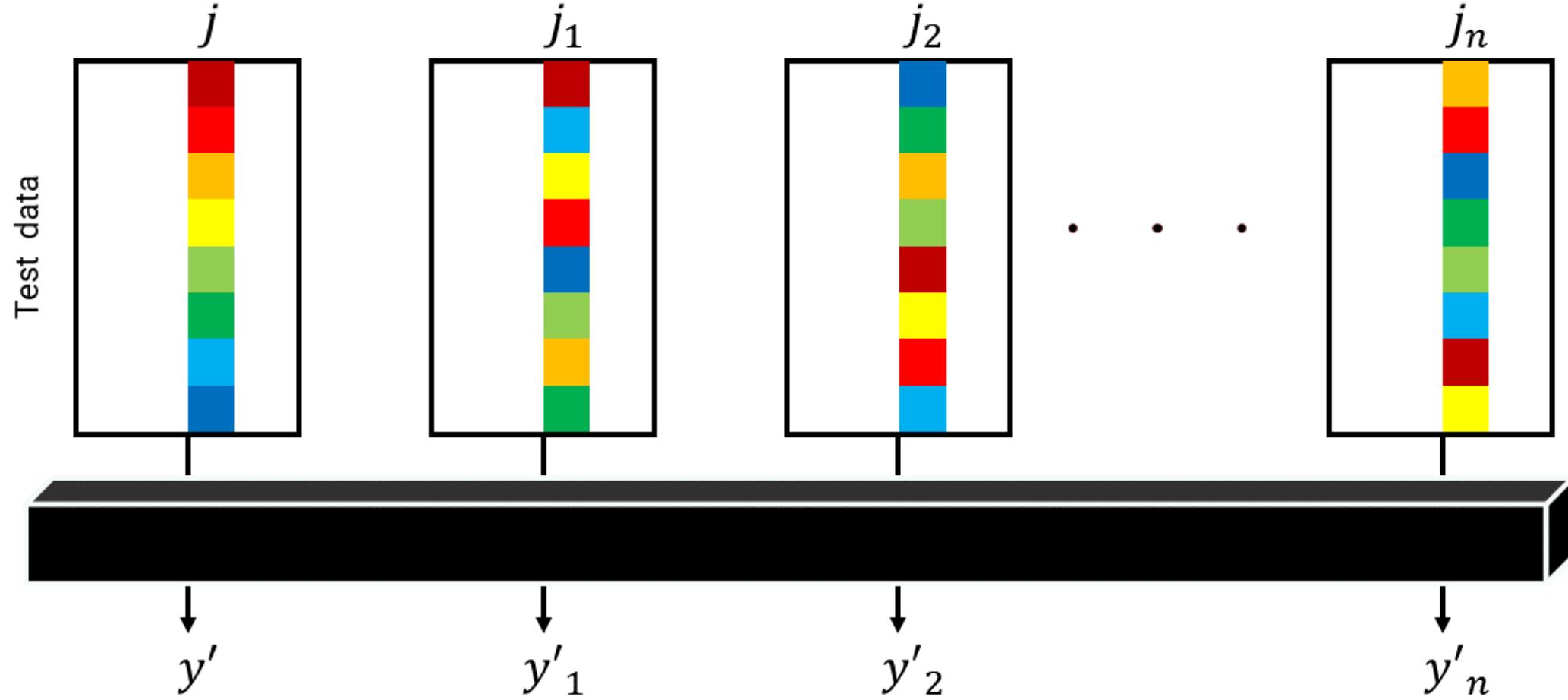
Permutation algorithm

Given a trained model and a test set permutation testing can be implemented as follows:

1. Compute baseline performance on the test set
2. For each feature in the test set:
 - Shuffle the feature of interest
 - Evaluate test set performance following this shuffle
 - Repeat this process multiple times to generate a distribution of performance given the shuffled feature
 - Compute a test statistic to determine whether or not the disturbance of this feature led to worse test set performance

Interpreting Permutation results

- At a high level, permutation testing conducts a conditional independence test for each feature X_j , with the null hypothesis stating that an outcome y is independent of feature X_j given all other features
- Intuitively, if X_j is predictive of y , perturbing this feature in isolation will break down its relationship to y and lead to drops in performance



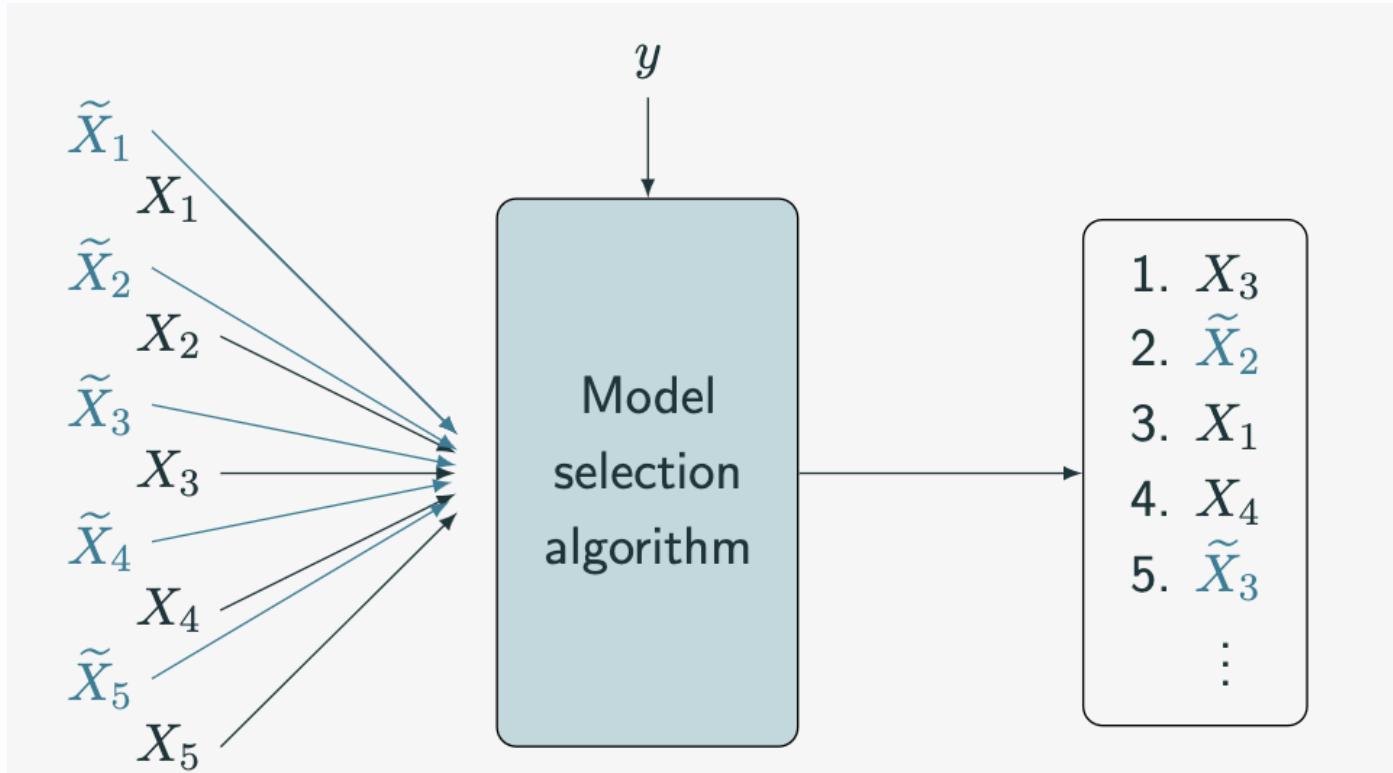
$$H_0: loss(y', y) = \frac{1}{n} \sum_{i=1}^n loss(y'_i, y)$$

Limitations of Permutation testing

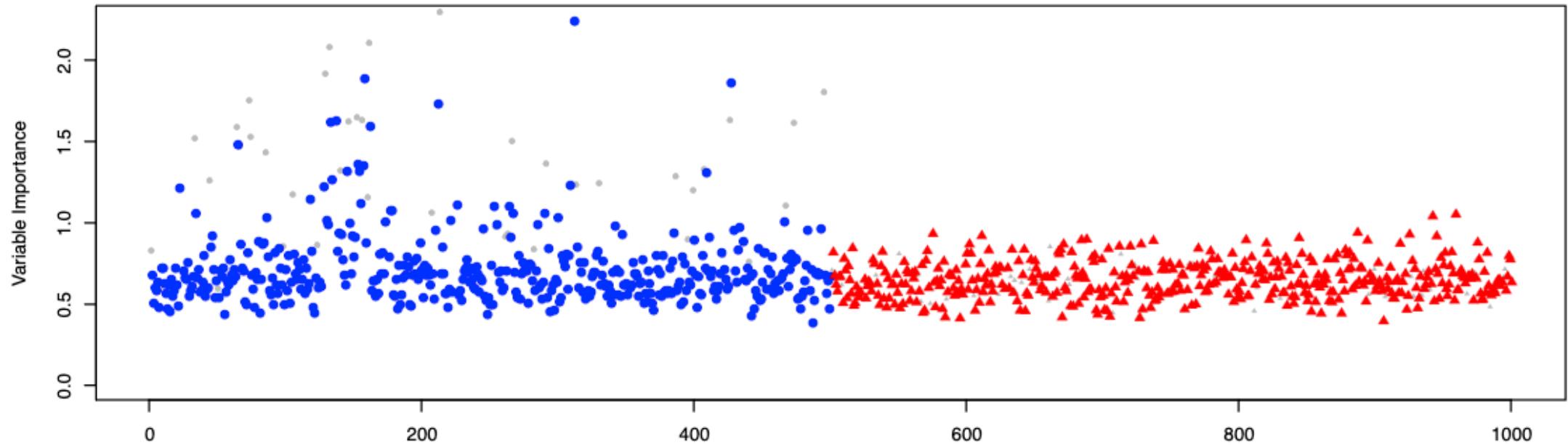
- **Sensitivity to test set size:** effectiveness of Permutation testing may vary depending on the size of the holdout set
- **Limited interpretability:** Does not account for dependencies between columns of the data
- **Assumption of exchangeability:** The act of shuffling features in isolation may introduce unrealistic data upon which feature importance is calculated

Knockoffs

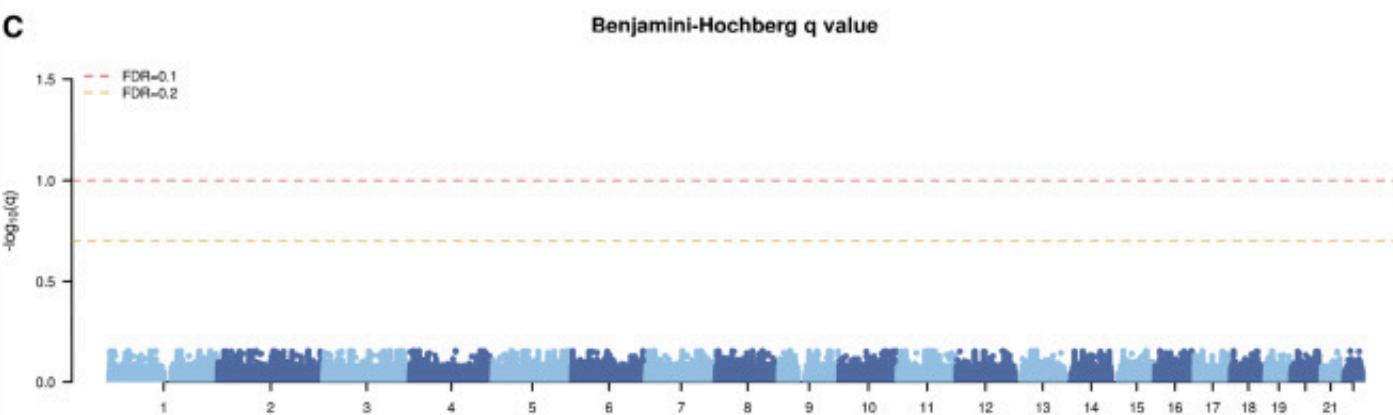
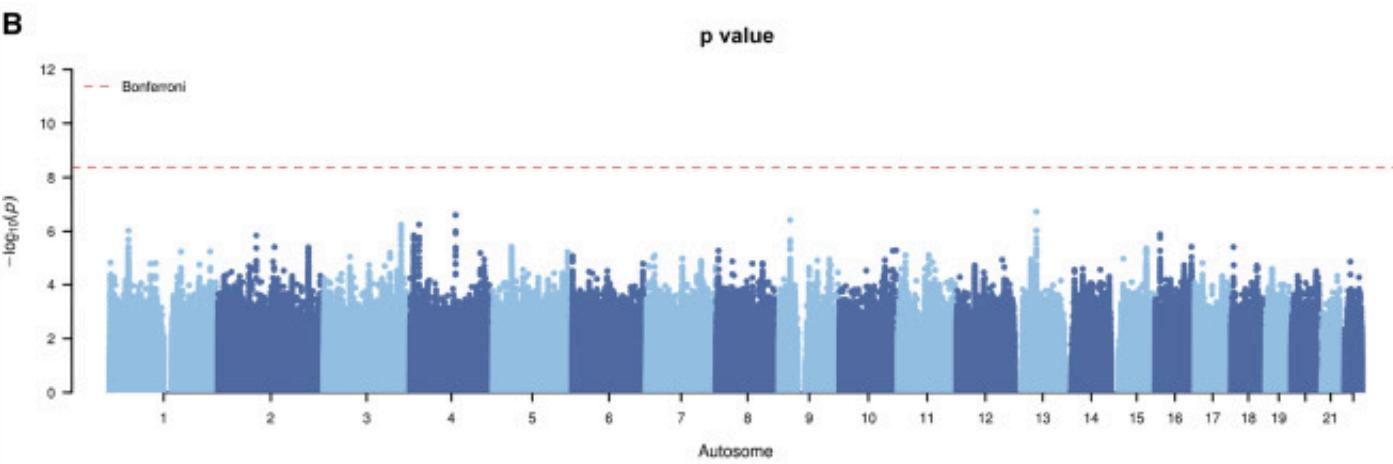
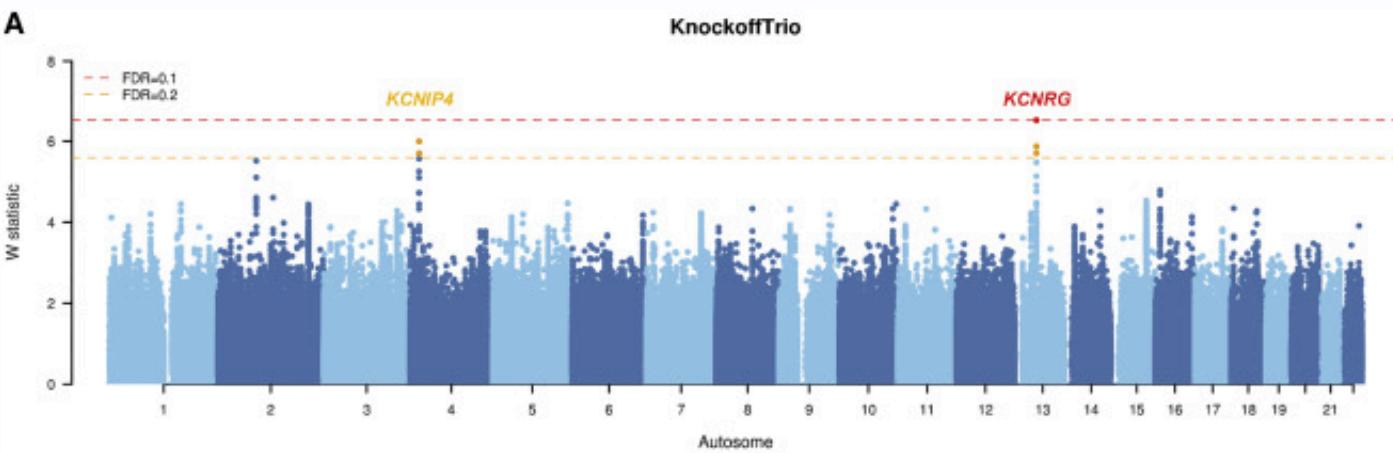
- Suppose we want to model $y = f_\theta(X)$
- Create a "knockoff" version of $\tilde{X} = X$, then model $y = f_\theta([X, \tilde{X}])$
 - Match correlation structure (except for knockoff pair): $X^T X = \Sigma$, $\tilde{X}^T \tilde{X} = \Sigma$, and $X^T \tilde{X} = \Sigma - \text{diag}(s)$
- Calculate a feature importance score for the original and knockoffs
- Select a subset of features based on empirical distribution of scores which guarantees a certain false discovery rate (FDR) proportion



Knockoffs (overview)



Source: Using Knockoffs to Find Important Variables with Statistical Guarantees



Knockoffs (summary)

- Advantages
 - Works with any model able to generate a feature importance score
 - Provides robust statistical guarantees (false discovery)
- Disadvantages
 - Requires special optimization procedure to construct the knockoff
 - Errors in knockoff construction will destroy statistical guarantees
 - Higher computational cost (i.e. need to fit model with $2 \cdot p$ features!)
 - Different feature importance metrics will lead to different subsets of features

Holdout randomization test (HRT)

- A feature is "uninformative" if, conditional on all other features, it does not impact model performance
 - $y \perp X_j | X_{-j}$
- Suppose we train our model to learn θ : $\arg \min_{\theta} \ell(y, f_{\theta}(x))$
- We can get an unbiased estimate of the loss function $\ell(\cdot)$ if we evaluate it on a new set of data (e.g. a "test set")
- The HRT shows how we can determine whether a given feature (x_j) has any impact on this representative "test set"
 - And therefore we can get back a p-value for $H_0 : y \perp X_j | X_{-j}$

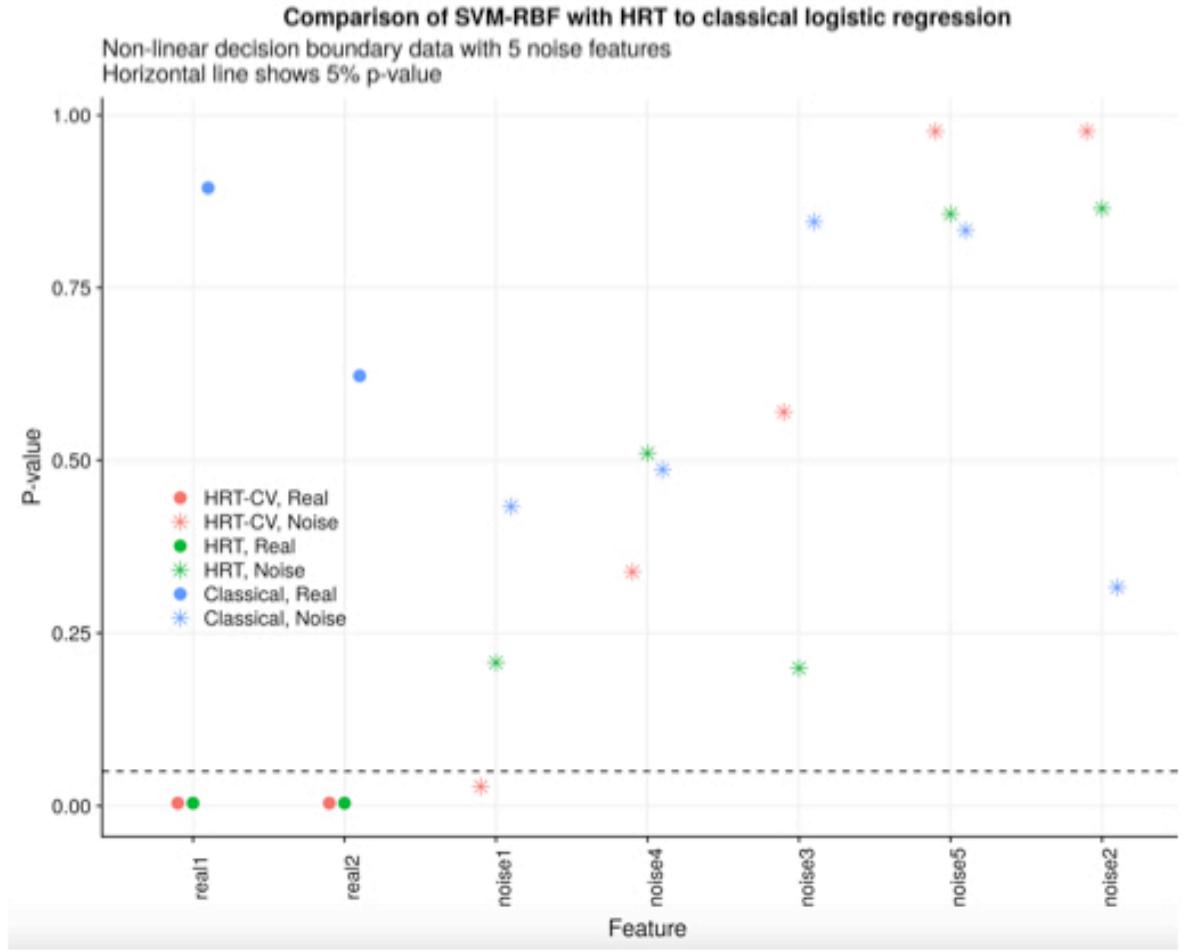
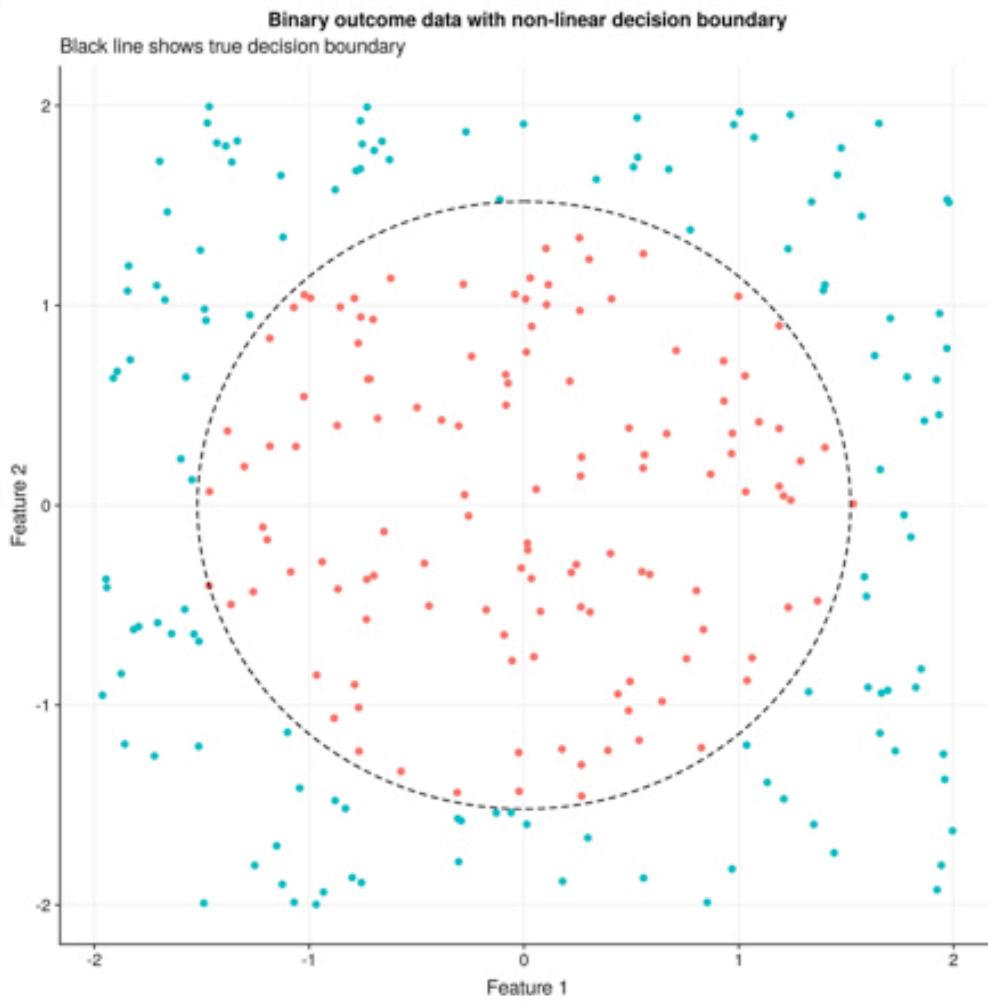
HRT algorithm

- Four step process for **any ML model** and **any loss function**
 - Step 1: Obtain a fitted model, \hat{f}_θ , using training/validation data (y_R, X_R)
 - Step 2: Compute the empirical risk on a test set (y_T, x_T) :
 $\hat{R}(\hat{f}_\theta) = \ell(y_T, \hat{f}_\theta(x_T))$, where $E[\hat{R}(\hat{f}_\theta)] = R(\hat{f}_\theta)$
 - Step 3: Draw S samples of feature j , $\tilde{x}_j^s \sim F(x_j|x_{-j})$, and calculate the test set risk: $\hat{R}^s = \ell(y_T, \hat{f}_\theta(\tilde{x}_T))$, swapping only column j
 - Step 4: Calculate a one-sided p-value as the proportion of times the "sampled" covariate led to a lower loss (i.e. risk):
 $p_j = \frac{1}{1+S} \left(1 + \sum_{s=1}^S I[\hat{R}^s \leq \hat{R}] \right)$

HRT sampling

- But how do we draw $\tilde{x}_j^s \sim F(x_j|x_{-j})$?
- Similar to parametric data imputation, we need to have a new model for every column we want to apply the HRT algorithm towards
- For example, if x_j is a binary indicator, we could model it with a logistic regression model: $\log \frac{E[x_j|x_{-j}]}{1-E[x_j|x_{-j}]} = \gamma_0 + \sum_{k \neq j} \gamma_k x_k$
 - And then draw samples from a Bernoulli distribution:
$$\tilde{x}_{ij} = \text{Bern}(\sigma(\hat{\gamma}_0 + \sum_{k \neq j} \hat{\gamma}_k x_{ik}))$$

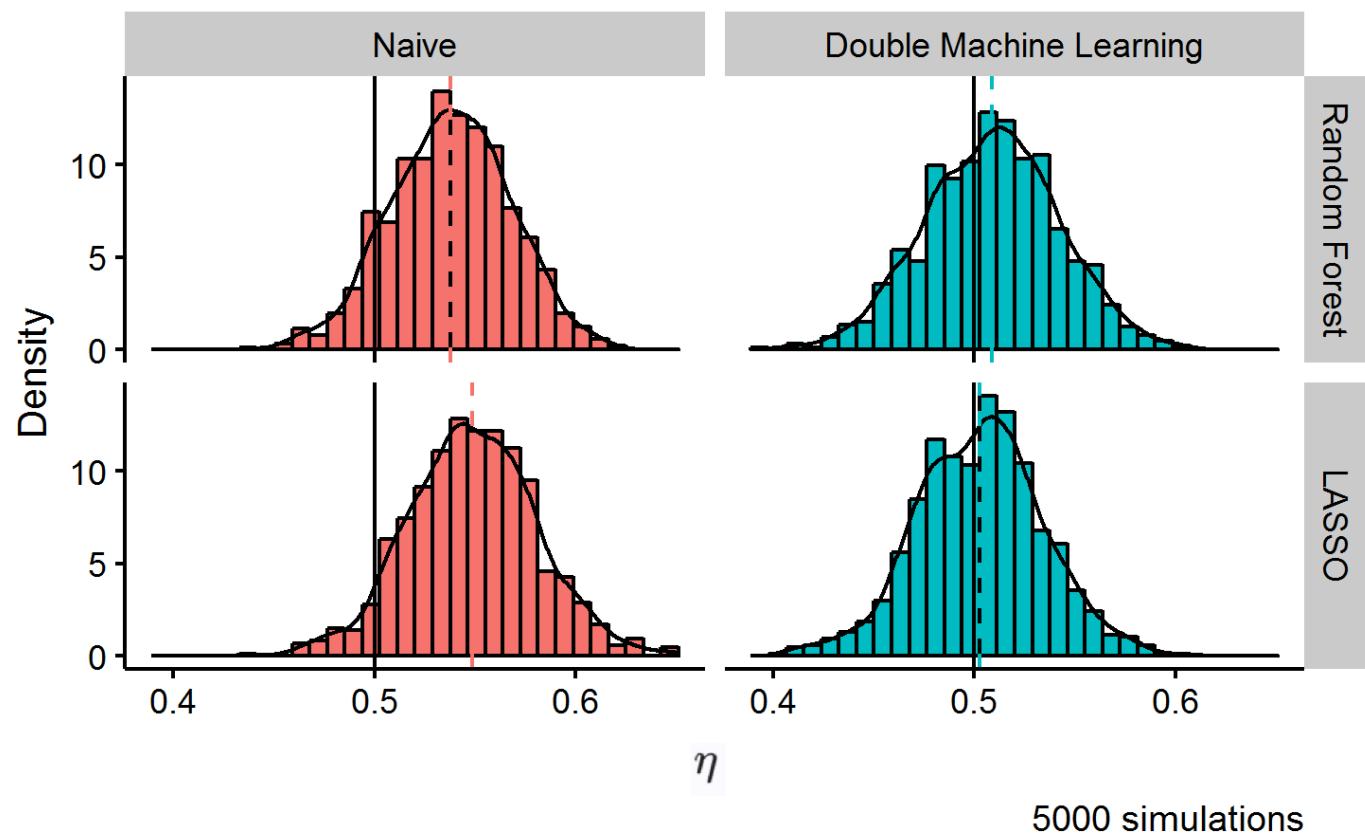
HRT example



Double machine learning (DML)

- If we assume that our data is "partially linear":
 - $y = d^T \eta + f_\theta(x)$
 - d is either a vector or matrix with a small number of columns
 - η is the parameters we are interested in doing statistical testing on
 - $f_\theta(x)$ is some functional form we will approximate with ML
- Example:
 - disease = smoking $\cdot \eta + f_\theta(\text{genetics})$
- Question:
 - What will happen if we learn $y = \hat{f}_\theta(x)$ first, and then "plug in" these predictions and run a simple regression ($y = d^T \eta + \hat{f}_\theta(x)$)?

Double machine learning (DML)



Double machine learning (DML)

- The problem is that $y - \hat{f}_\theta(x)$ creates biased "residuals" (i.e. still lots of confounding)
- The solution is to learn two sets of models so that that residual bias cancels out at a very fast rate so we get a consistent estimator of η
 - Step 1: learn $y = \hat{f}_\theta(x)$
 - Step 2: learn $d = \hat{g}_\phi(x)$
 - Step 3: Determine if there's any variation in y attributable to d that isn't explained by x !
 - $y - \hat{f}_\theta(x) = d - \hat{g}_\phi(x)$

DML intuition

- Why does this work?
 - The errors in the first stage (estimating f_θ and g_ϕ) do not systematically bias the second stage
 - Cross-fitting ensures the procedure is asymptotically unbiased



θ_0 is identified from $\mathbb{E}[\psi(W, \theta_0, \eta_0)]$

by solving $\frac{1}{n} \sum_{i=1}^n \psi(W_i, \theta, \hat{\eta}_0) = 0$

$$\psi_{\text{naive}}(W, \theta_0, \eta_0) = (Y - D\theta_0 - g_0(X))D$$

where $\eta = g(X)$, $\eta_0 = g_0(X)$

$$\psi_{\text{dml}}(W, \theta_0, \eta_0) = ((Y - l_0(X)) - (D - m_0(X))\theta_0) \cdot (D - m_0(X))$$

where $\eta = (l(X), m(X))$, $\eta_0 = (\mathbb{E}[Y|X], \mathbb{E}[D|X])$

$$D = \partial_\eta \mathbb{E}[\psi(W, \theta_0, \eta)] \Big|_{\eta=\eta_0} = 0 \quad (\text{Neyman Orthogonality})$$

$$J\sqrt{n}(\hat{\theta} - \theta_0) = A_n + D\sqrt{n}(\hat{\eta} - \eta_0) + C\sqrt{n}\mathcal{O}(\|\hat{\eta} - \eta_0\|^2) + \mathcal{O}_P(1)$$

$A_n \rightsquigarrow N(0, \Sigma_a)$

$\|\hat{\eta} - \eta_0\| \rightarrow 0$ if sample-splitting is used

when $D \neq 0$, since $\|\hat{\eta} - \eta_0\| = \mathcal{O}_P(n^{-\phi})$, $0 < \phi < 1/2$,

$\Rightarrow D\sqrt{n}(\hat{\eta} - \eta_0)$ is of order $\sqrt{n}n^{-\phi}$

when $D = 0$, $D\sqrt{n}(\hat{\eta} - \eta_0) = 0$

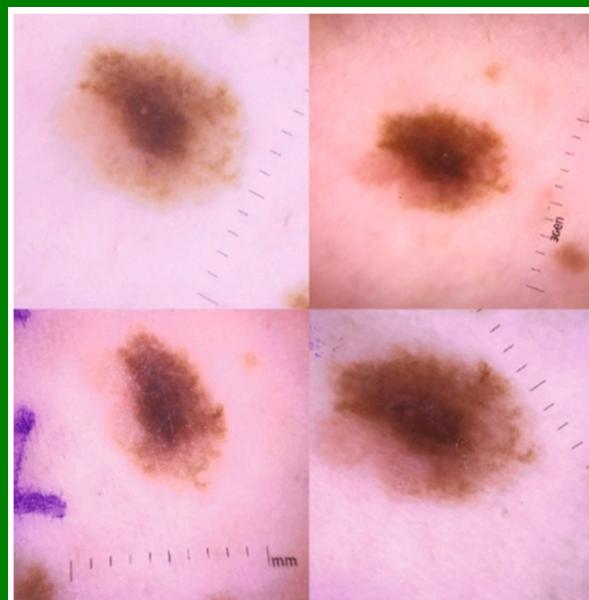
if $\|\hat{\eta} - \eta_0\| = \mathcal{O}_P(n^{-1/4}) \Rightarrow C\sqrt{n}\mathcal{O}(\|\hat{\eta} - \eta_0\|^2) \rightarrow 0$

Breakout #3

Come up with examples when you would use a local explainability method but not a global one, a global one but not a local one, and when you would use both?

Breakout #4

Suppose there is a melanoma classifier that uses a CNN.
As a potential future patient, how would you want this classifier to explain its "prediction" about whether you had melanoma or not from your picture?



References

- (1) Molnar, C. (2022). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2nd ed.). [Available online](#)
- (2) A Data Odyssey. (2023, March 20). SHAP with Python (Code and Explanations) [Video]. YouTube. https://www.youtube.com/watch?v=L8_sVRhBDLU
- (3) Tansey, W., Veitch, V., Zhang, H., Rabadian, R., & Blei, D. M. (2018, November 1). The Holdout randomization test for feature selection in black box models. arXiv.org. [Available online](#)
- (4) Spector, A., & Janson, L. (2020, November 30). Powerful knockoffs via minimizing reconstructability. arXiv.org. [Available online](#)