# Explainability

Magnus Nielsen, SODAS, UCPH

# Agenda

- Explainability
- Model specific
  - Linear regression
  - Decision trees
- Model agnostic
  - Permutation importance
  - Partial dependencies
- SHAP values

# What is explainability?

# Question

In your opinion, what is explainability in the context of machine learning?

# Explainability

What is an explanation?

- The answer to a why question

In the context of today:

- Why did our model predict what it did

Today we will look at

- What constitutes good answers
- How we can create these answers
- The strengths and weaknesses of different answers

Explainability and interpretability are used interchangeably

# Why explainability?

Understanding our models can help in many scenarios

- Model building
  - Debugging
  - Improving accuracy
- Fairness
  - Detect if models use protected attributes heavily
- Acceptance of models
  - (Some) people don’t trust predictions made by black box models

# When is an explanation good?

Important to remember that explanations are social and should be tailored to the situation

Some other general pointers to think about when explaining your models:

- We like to contrast explanations to some (perhaps imaginary) other observation
  - Contrast to average or other observations
- We prefer selective explanations, even though they may be less truthful
  - Focus on the biggest factors
- We prefer explanations that focus on the abnormal, but if no abnormal events occur, we prefer explanations to be general
  - Focus on abnormal values in observations

# Scope of methods

Global and local methods

Global methods describe the average behaviour of a machine learning model

- Explaining the broader strokes
- Debugging and improving by discovering unexpected and weird behaviour

Local methods explain individual predictions (or model in the vicinity of an individual prediction)

- Explaining a specific instance
- Debugging and improving by examining a few bad predictions

# Comprehensibility above all

The most important property is how comprehensible the explanations are

- Can the audience actually understand the explanation?

The social context of explanations is especially important here

- You would all probably be comfortable with interpretation of linear or logistic regression models

# Other properties of explanations

There are many other properties that one can consider, i.e.:

- Fidelity: How well do the explanation explains the prediction
  - Some methods offer only individual fidelity
- Accuracy: How well do explanations explain unseen values
  - Especially important if using explanations in place of predictions
- Stability: Are explanations similar for similar observations (in regards to both input and prediction)
  - Unless small perturbations drastically change output, explanations should be similar

A longer list can be seen here

# Properties of explanation methods

Expressive Power

- What is the structure of explanations, e.g. weights or flowcharts

Translucency

- How much does the explanation rely on the model which made the prediction

Portability

- How portable is the explanation method between different models
- Translucency and portability have an inverse relationship

Algorithmic Complexity

- How much time does it take to compute the explanations

# How to achieve interpretability

Two methods:

- Use models that are intrinsically interpretable
- Use post-hoc methods for explainability

# Which to choose?

The most accepted within social sciences and legislation are intrinsically interpretable models

- Linear regression, logistic regression or decision trees
- Can obtain full global fidelity
  - Can explain exactly why a prediction was what it was
- Generally entails a performance trade off

This is not always necessary and model-agnostic post-hoc methods are most often used

- It seems that post-hoc methods are becoming the default (at least within data science)

# Linear regression

# Weights

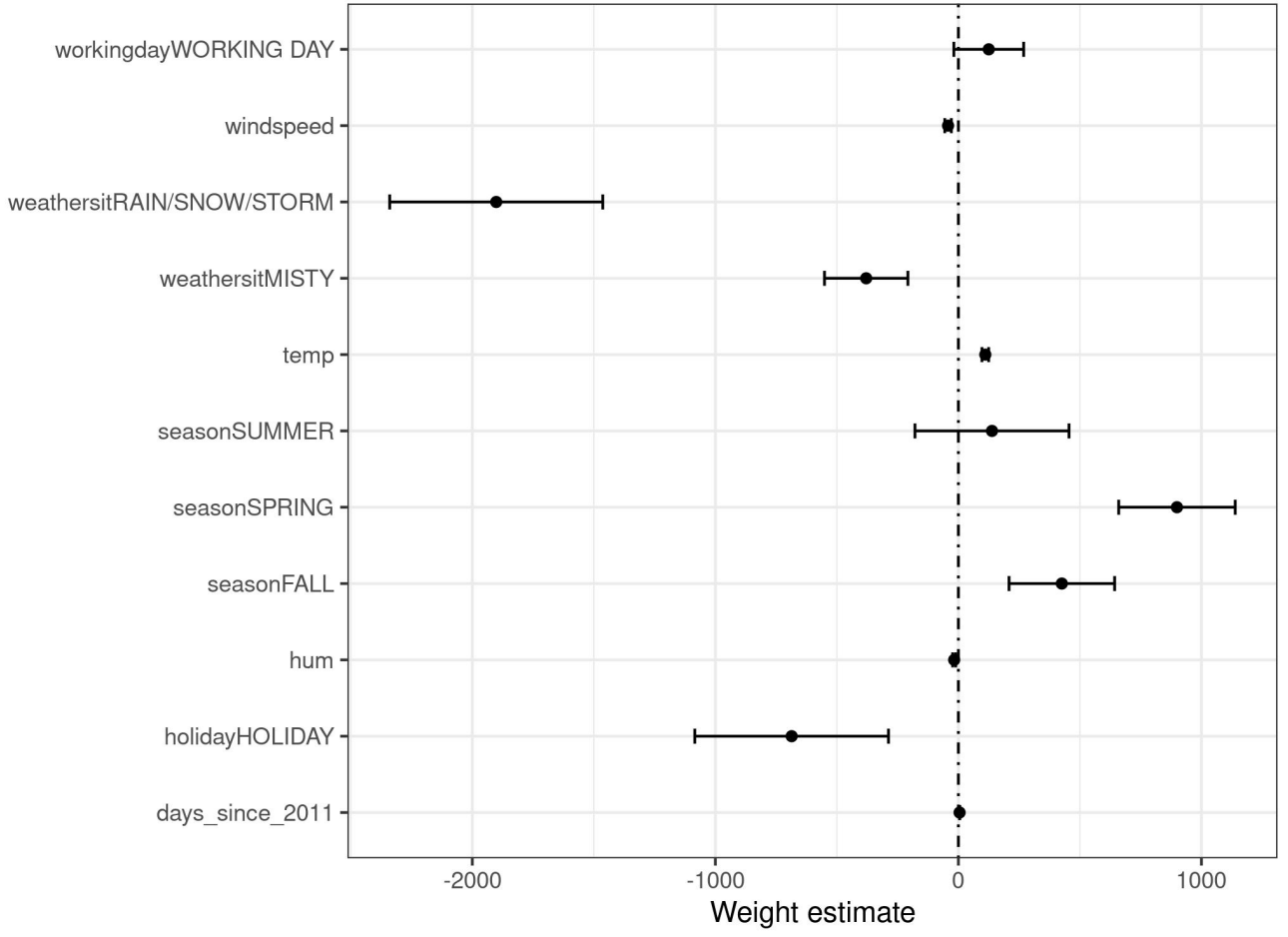
In linear models, we can interpret the weights

- All else equal

Lots of collective experience

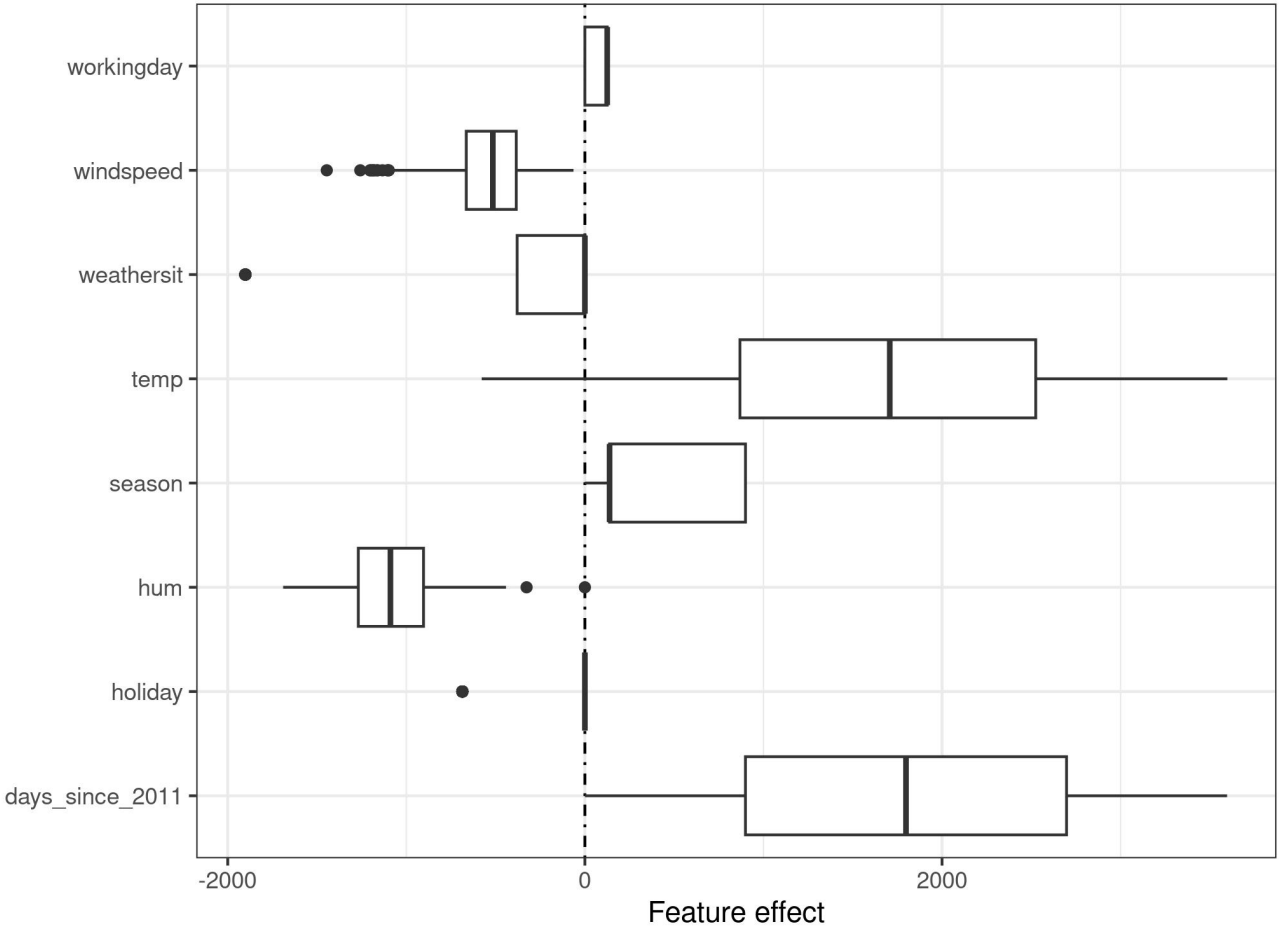
- Both among social scientists, but also other fields
- Going to focus on some visuals

# Weight plots



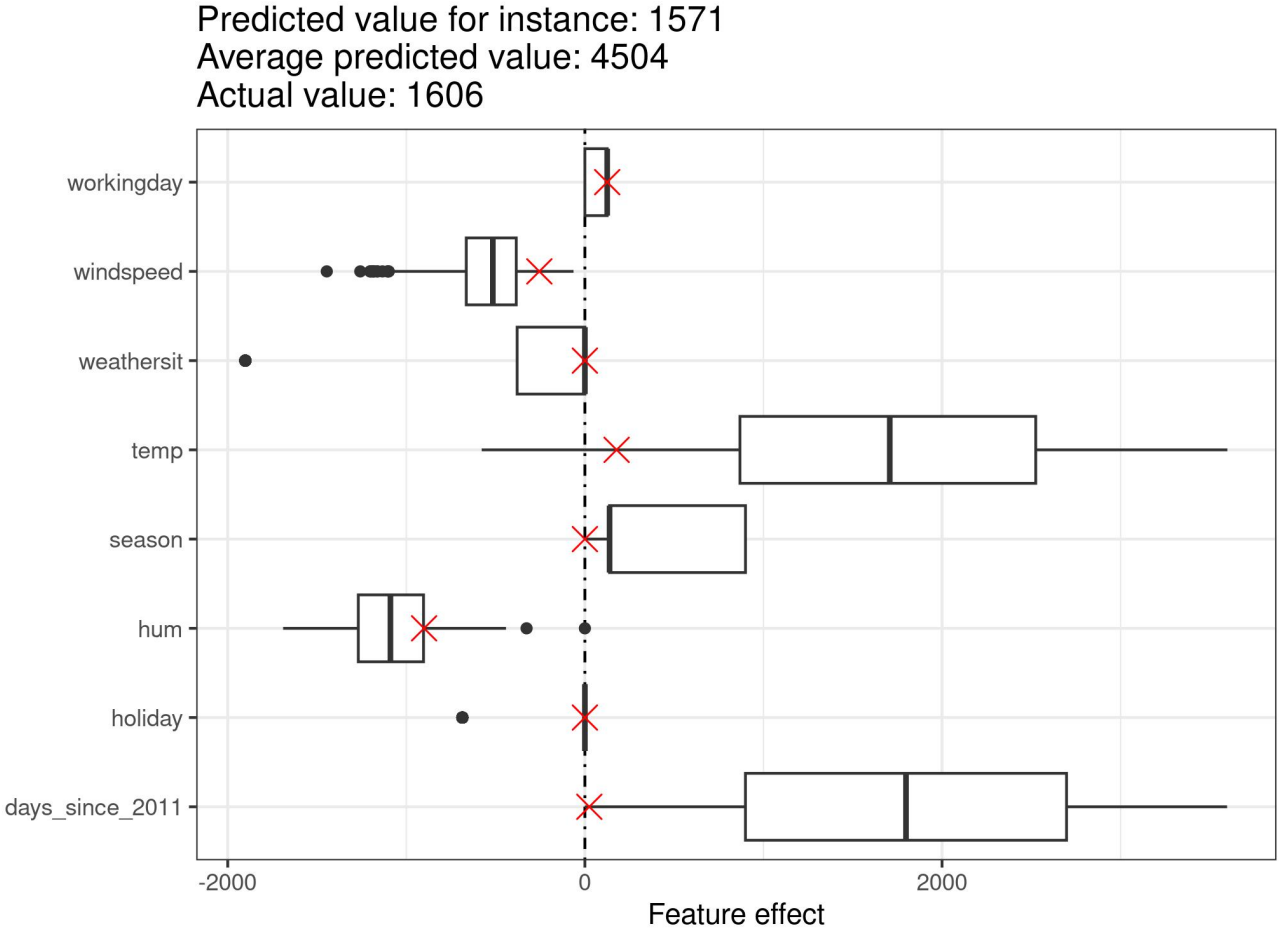

Source: Molnar, 2022

# Weight feature plot




Source: Molnar, 2022

# Individual explanations




Source: Molnar, 2022

# Selective explanations

LASSO models create sparse models due to the $L_2$ norm

Can be done to either

- Increase performance
- Reduce complexity

$\alpha$ can be tuned such that a set amount of weights are non-zero

- Utilize same methods as before

# Drawbacks

- No interactions unless you include them
- Linear models do not perform well with non-linear data
- All else equal interpretation
  - If features covary, this can create weird interpretations
- If features covary, model may arbitrarily select one

# Decision trees

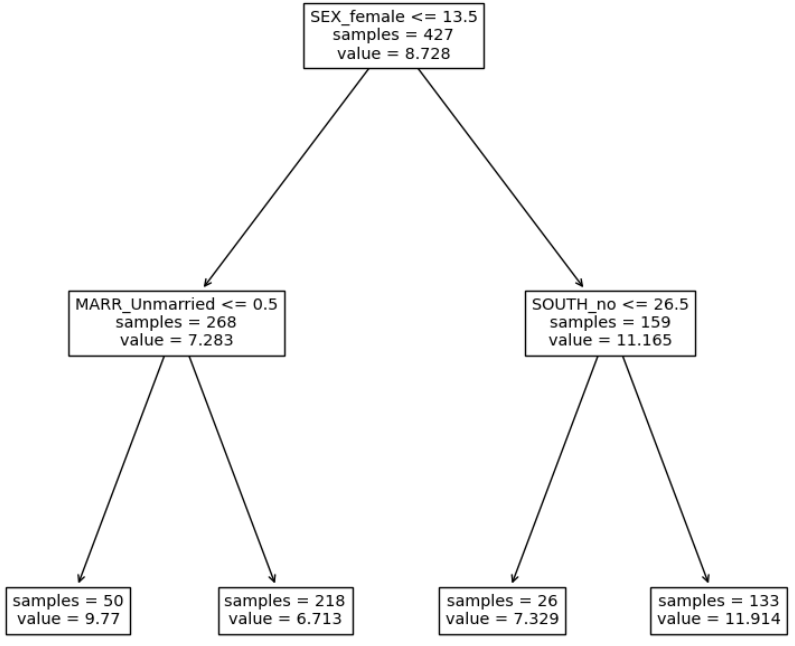
# Decision trees as flowcharts

Decision trees can be plotted

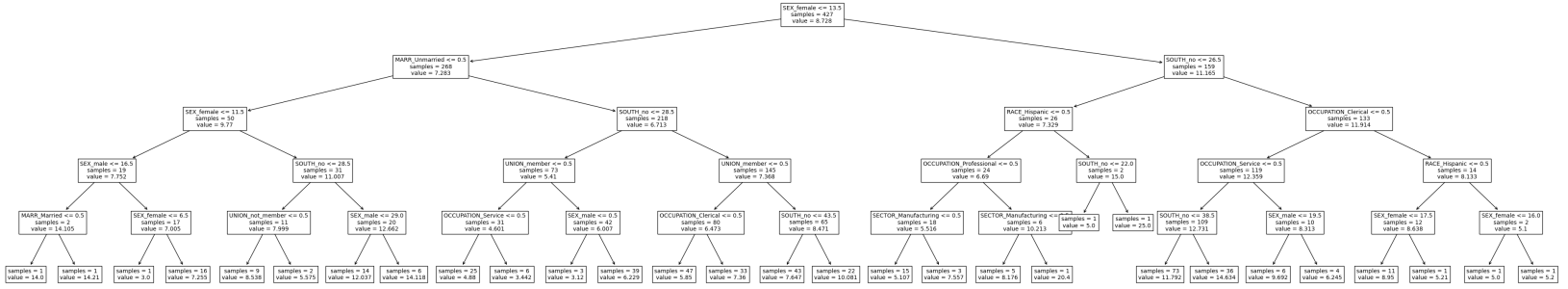
- Easy to understand
- A collection of if-else statements and splits
- Easy to imagine counterfactuals
- They always go left in `sklearn`

Maximum amount of leafs is maximum depth to the power of two

# A shallow flowchart



# A deep flowchart



# Question

Do you consider linear regression or decision trees to be most explainable?

- Does it depend on the amount of features or complexity of the model?

# The social aspect of explainability

I wager that most people here would default to linear regression

You would have no problem interpreting coefficients

- But what about the receiver of the explanation?

I posit that decision trees are more easily understood for a layman

- Especially for a single given observation, e.g. the layman's own observation
- Cutoffs readily available for counterfactuals

# Mean decrease in impurity

Another method for trees, which calculates global feature importance

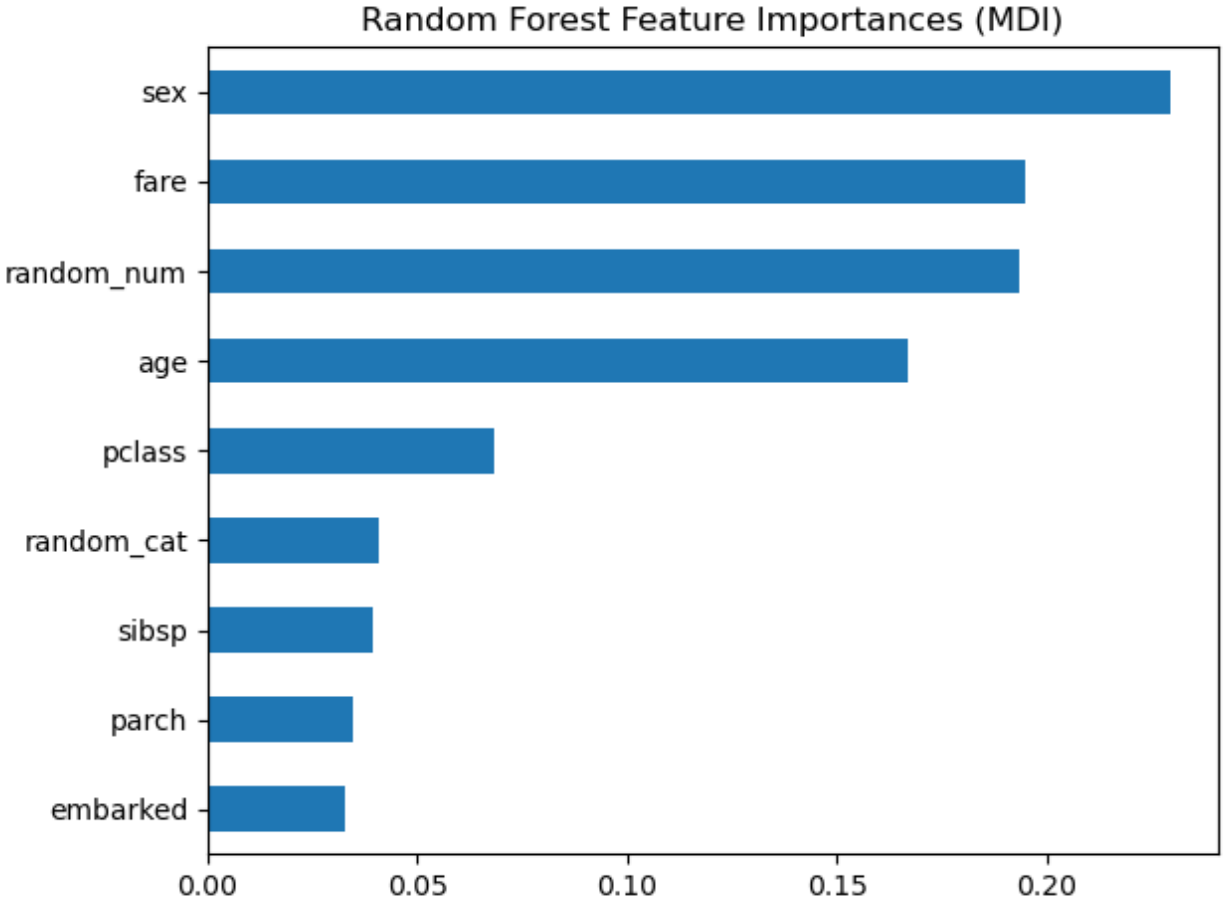
For all features:

- Go through all the splits for which the feature was used
- Measure how much it has reduced the MSE or Gini index compared to the parent node

Scale sum of importances to 100

- Interpretation: Importance as a fraction of total decrease in impurity

# Titanic with two random features




Source: sklearn

# Question

This method is generally biased and favors high cardinality features.

Why?

# Drawbacks

The method favors high cardinality features because these features are more often split on

- E.g. once a decision tree has split on a binary feature, it won't split on it again
- See this example from sklearn

Can only say something about the model in relation to the training data

As a result, not often used

- Permutation feature importance used instead

# Model agnostic

# Shedding light on the black boxes

Some models are so complex that we cannot understand them or their components

Here we can use model agnostic explanation methods

- Translucency is gone

# Permutation feature importance

Permute a feature

- This breaks the dependence between $X$ and $y$
- Also breaks any interactions

How much does this loss of information change the score (e.g. MSE)

- Nice because this is the object of main interest!

Can be computed for both train and test

# What data to use

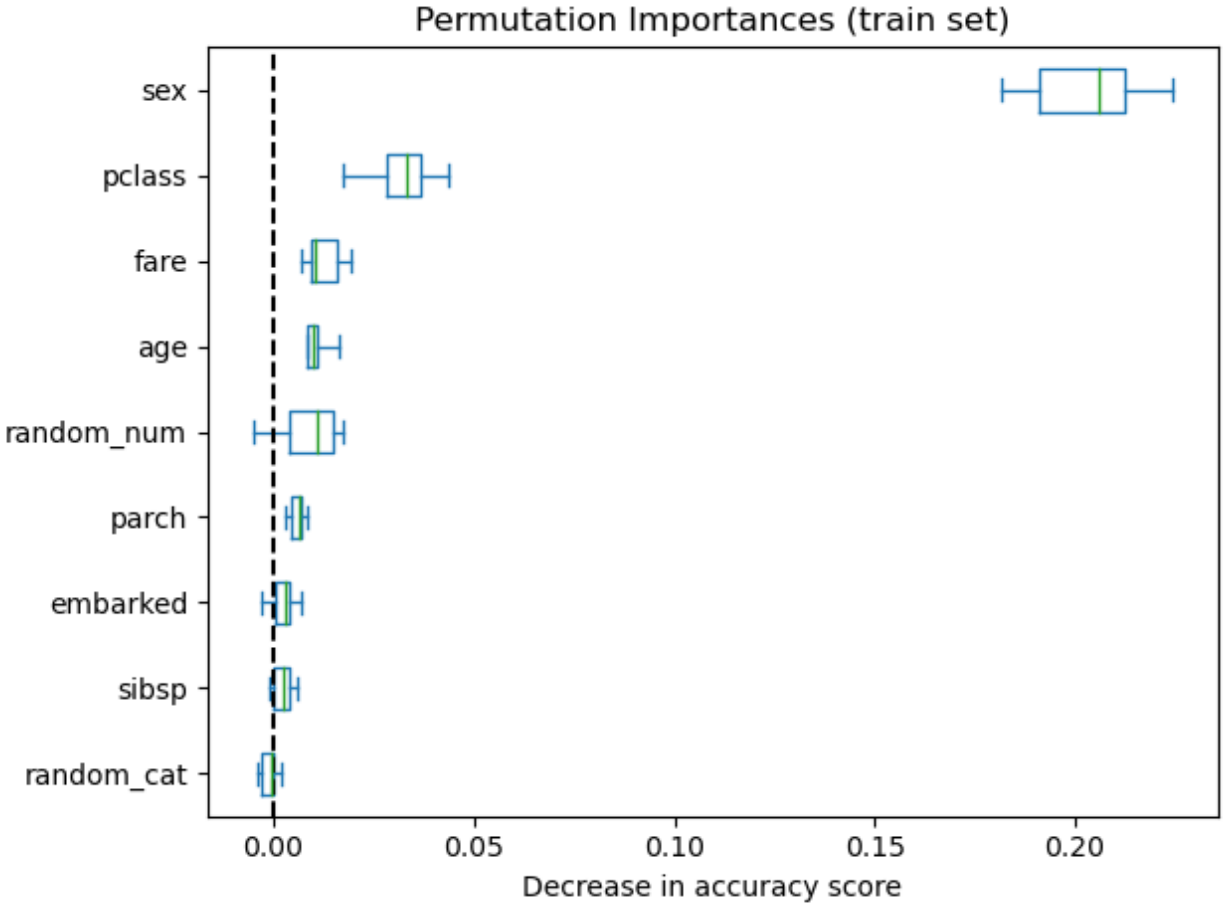
Most commonly done on test data

- Realistic error estimates
- The object of interest

However, training data will show what features the model uses

- Can be used

# Permutation plots




Source: sklearn

# Pros and cons

Pros:

- Nice interpretation
- Highly compressed
- No retraining
- All interactions

Cons:

- Permutations can create unrealistic datapoints
- If features covary, importance is split between them
  - Could do some form of clustering based on covariation, see here
- Interpretation not related to prediction itself

# Partial dependencies

# Average dependence

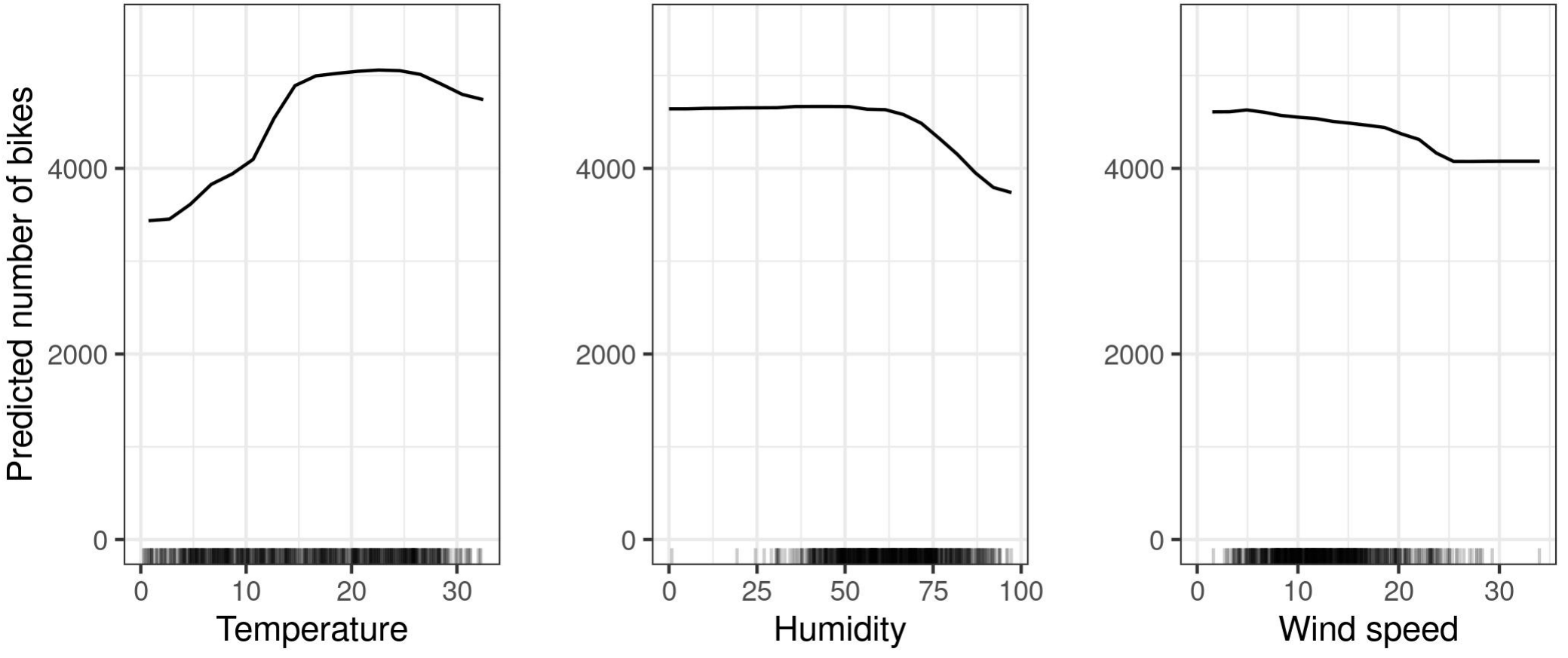
What happens when we set a feature $X$ with $x$?

- In essence, hold all values fixed except one (or two)

Plot how the average prediction changes as a function of $X$

- This is called a partial dependence plot
- Due to the averaging, this is a global method

# Partial dependence plot




Source: Molnar, 2022

# Pros and cons

Pros

- Easy to interpret
- Easy to compute

Cons

- Limited to one or two features due to perception
  - “Due to the limits of human perception, the size of the set of input features of interest must be small (usually, one or two)”, sklearn User Guide
- Only considers averages
  - Can mask heterogeneous effects
- Dependencies are all else equal
  - This is something we’re used to from OLS
- Can create absurd datapoints

# Uncovering the heterogeneity

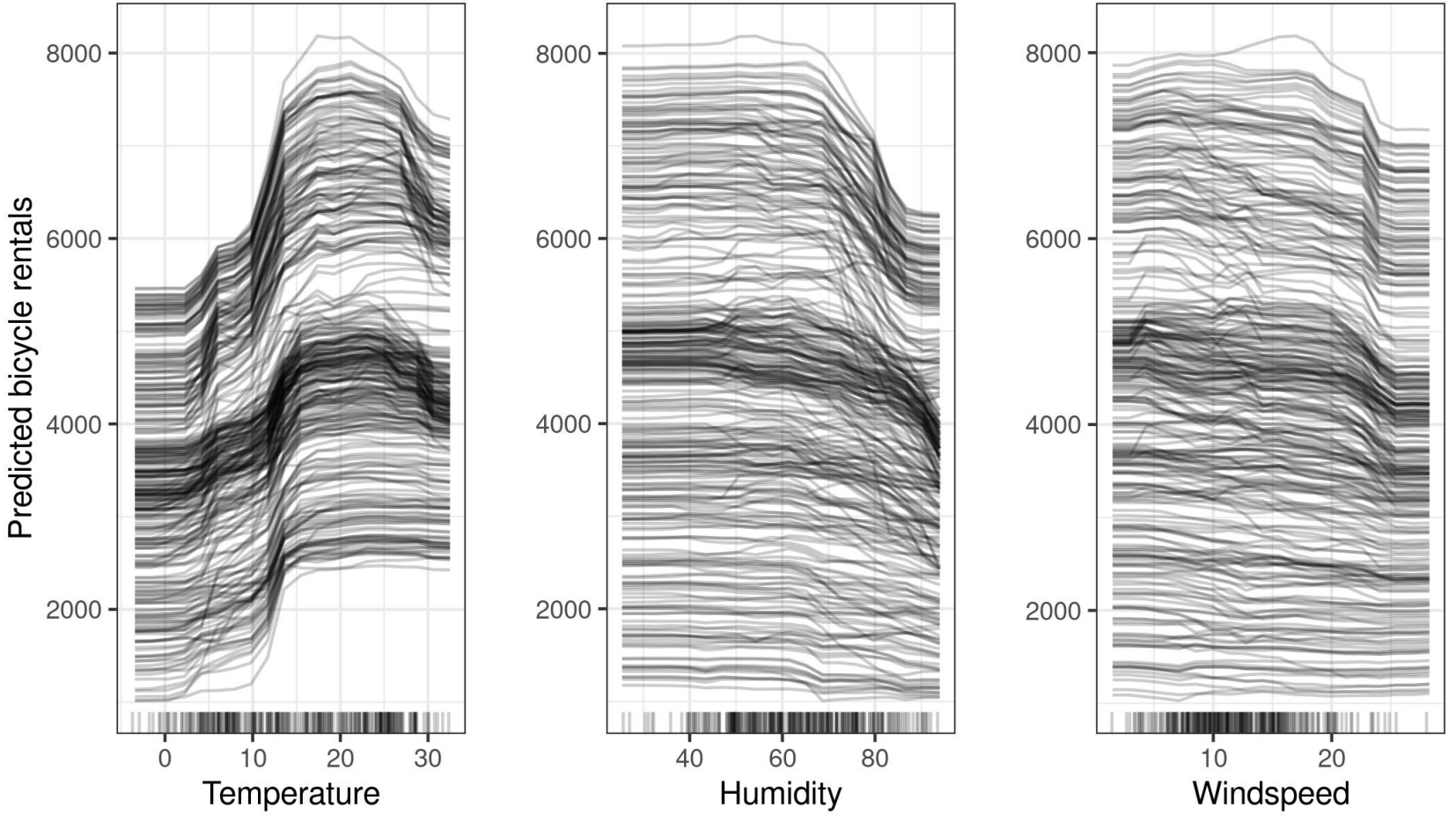
How to avoid hiding heterogeneous effects?

Plot a partial dependence for all observations!

- This is called an individual conditional expectation (ICE) plot

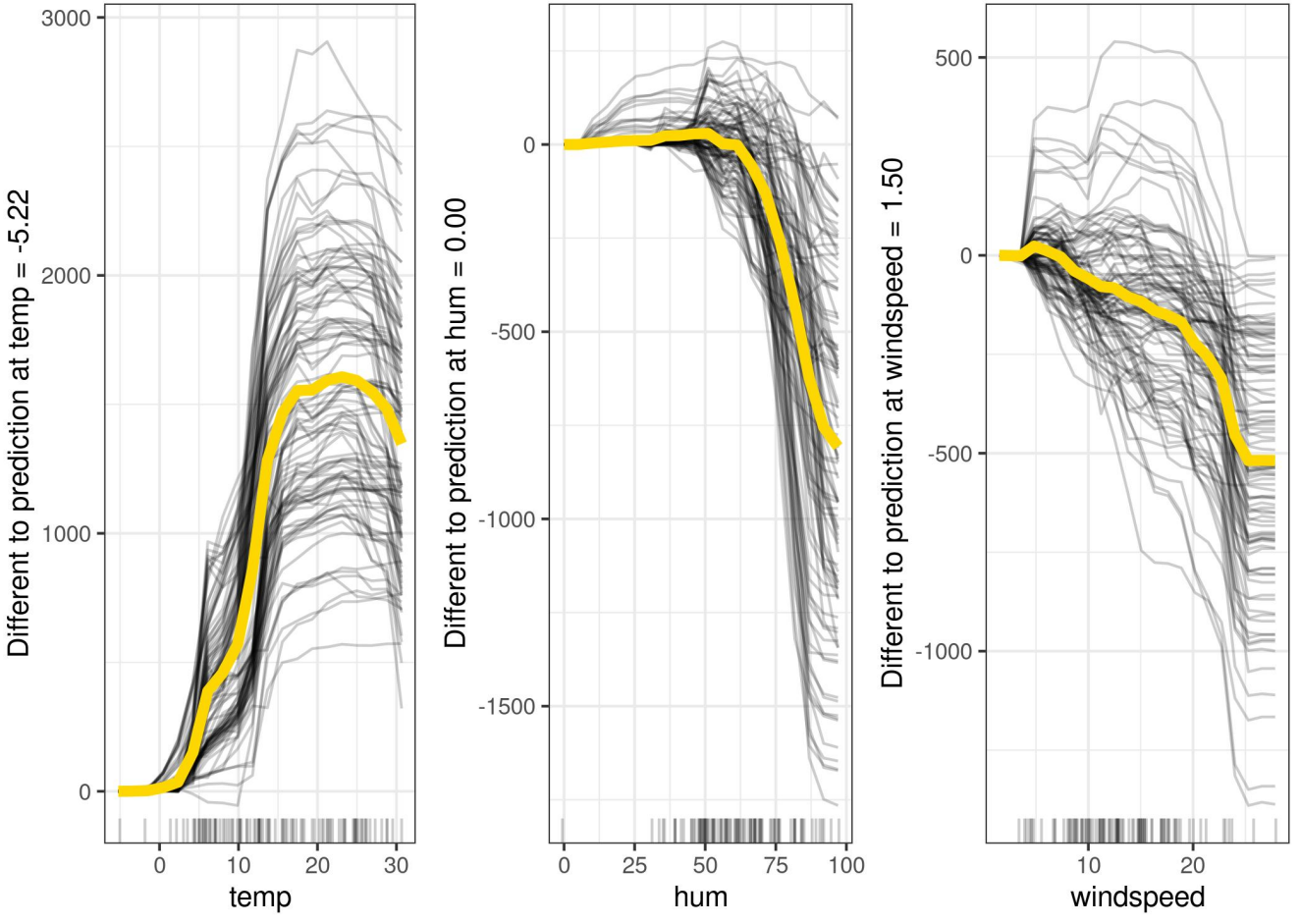
As we no longer average over all observations, it is local

# ICE plots




Source: Molnar, 2022

# Centered ICE plots




Source: Molnar, 2022

# Pros and cons

Pros

- Easy to interpret
- Can display heterogeneity

Cons

- Now limited to just one feature
- Still all else equal dependencies
- Can be hard to visualize for large datasets

# SHAP

# Roadmap

Shapley Additive Explanations allocate prediction outputs as if a game (Shapley values) using a local interpretable model (LIME)

- As such, we will (quickly) cover these

There’s a lot of math and pseudo-code in the papers and book regarding SHAP and it subcomponents

- This I have omitted

# LIME

Local interpretable model-agnostic explanations introduced by Ribeiro et al. (2016)

- Only focus on local fidelity

Create a low-complexity (intrinsically interpretable) model for each observation

- E.g. LASSO

Available stand-alone in the package LIME (not covered in exercises)

# LIME in a nutshell

Select observation to explain

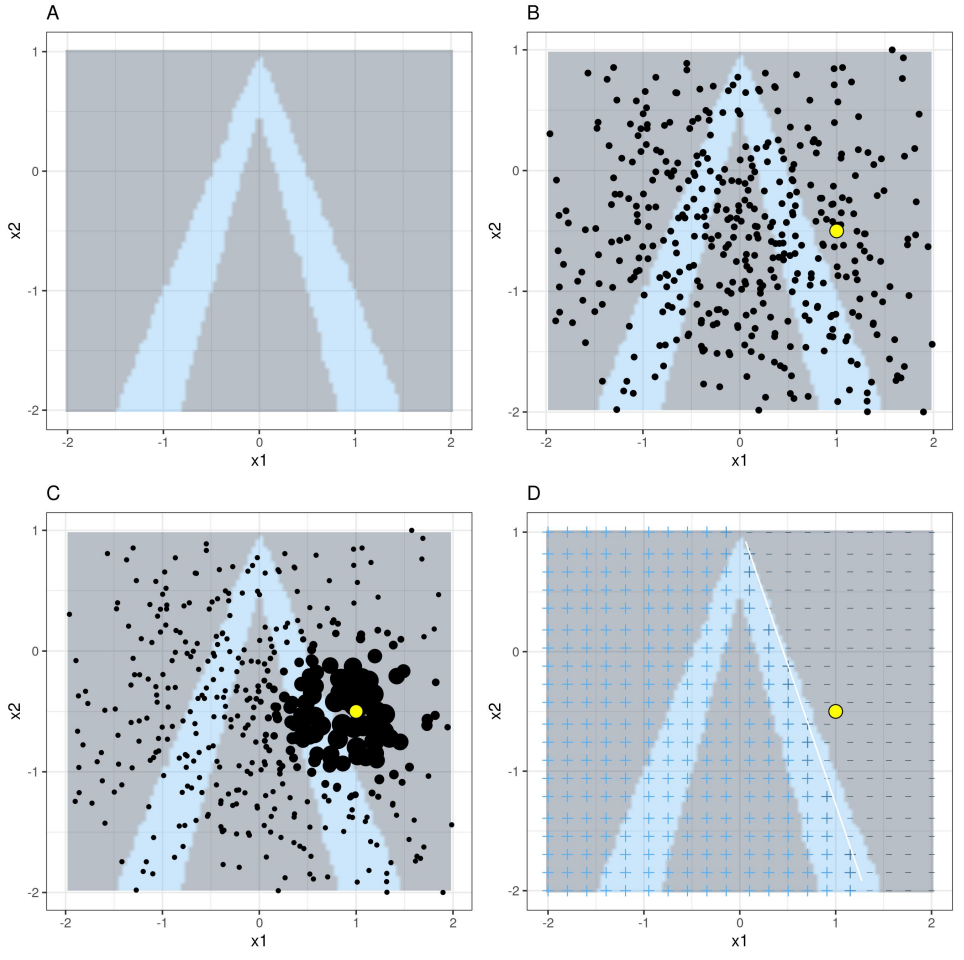
- Perturb the observation and compute black box predictions
- Weight the new samples according to proximity (using some kernel)
- Train a weighted, interpretable model on the dataset with the variations
- Explain the prediction by interpreting the local model

# Devil in the details

Big question: What kernel bandwidth to use?

- LIME implementation in Python has a fixed kernel with a fixed bandwidth

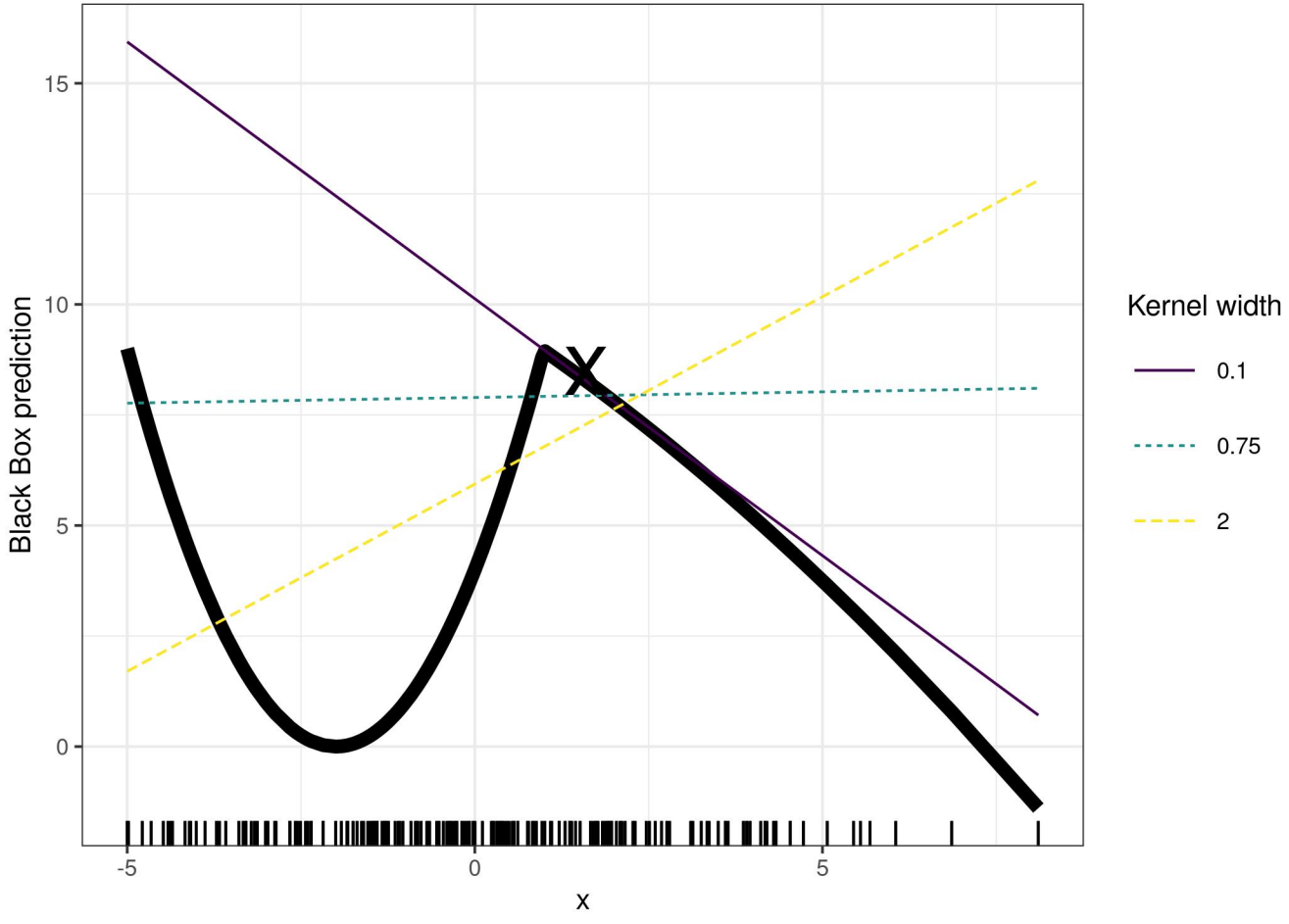
# LIME visually




LIME fitting procedure

Source: Molnar, 2022

# Different kernels




LIME with different kernels and true dependence

Source: Molnar, 2022

# Shapley values

Allocate payout using Shapley values from cooperative game theory

- Features are players, prediction is payout

Theoretically grounded

# Allocation of payouts

Groups of players (coalitions) can either participate in the game or not

- If a player enters the game with a coalition, this can change the payout

This change in payoff is distributed to the players in the coalition

Game is played for each instance

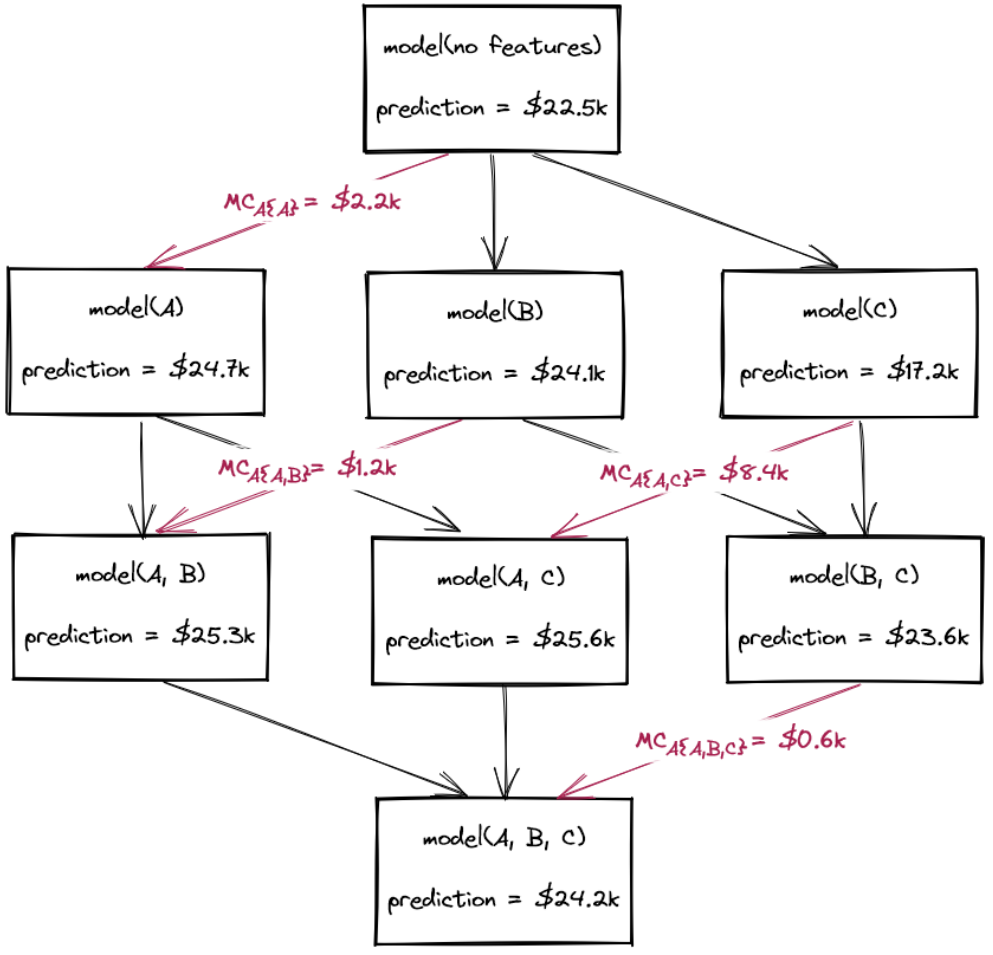
- A local method

# Why?

Nice theoretical guarantees

- Efficiency
  - Full local fidelity
- Symmetry
  - If two features attribute equally, the Shapley values are equal
- Dummy
  - Features which never change the payout have a Shapley value of zero
- Additivity
  - Payout attributions from multiple games are additive

# An example




Marginal contributions in feature sets

Source: aidancooper.co.uk

# One problem

Need to calculate models with the power set of all features

- Retraining a huge amount of models

This quickly becomes computationally expensive

# SHAP

To reduce compute, we return to SHAP (Lundberg & Lee, 2017)

- SHAP and Shapley values are not the same

Essentially: Change kernel in LIME with a weighing scheme based on Shapley values!

- Based not on distance, but how many features are ‘present’

Why?

- Retains the nice guarantees from Shapley values!

# Non-participation

To ‘simulate not participating’, we permute the features

- We break the dependency, just like in permutation feature importance

By doing this, we retain an (uninformative) input (which the model requires) and do not need to retrain!

- Much cheaper to compute

# How to permute?

Generally, we sample from a given dataset (subset of the training data or prototypical observations)

- SHAP values still local and for every observation
- Just a question of what we replace the values with when permuting

This can induce nonsensical feature combinations

- Same tale as with other methods where we change an input value

Called interventional feature perturbation, and is default in SHAP

# An alternative sampling strategy

There exists an alternative SHAP implementations for tree-based models, introduced in Lundberg et al. (2020)

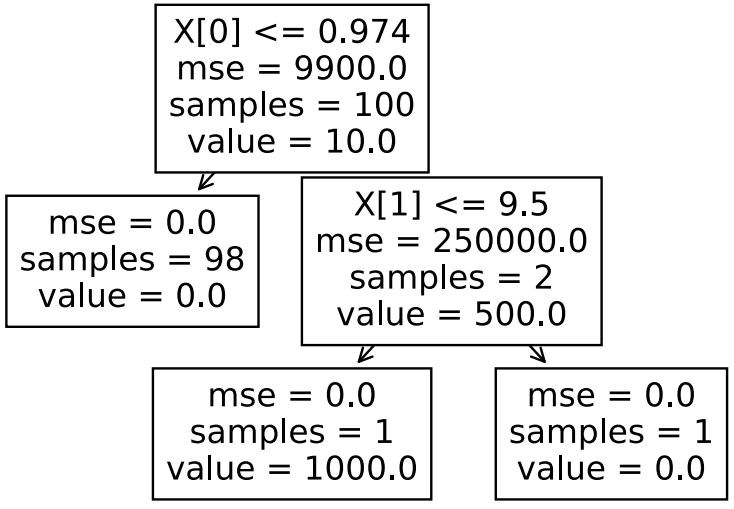
- Faster than fully model agnostic SHAP
- Very common way to explain e.g. SOTA XGBoost models

Here we can do path dependent sampling

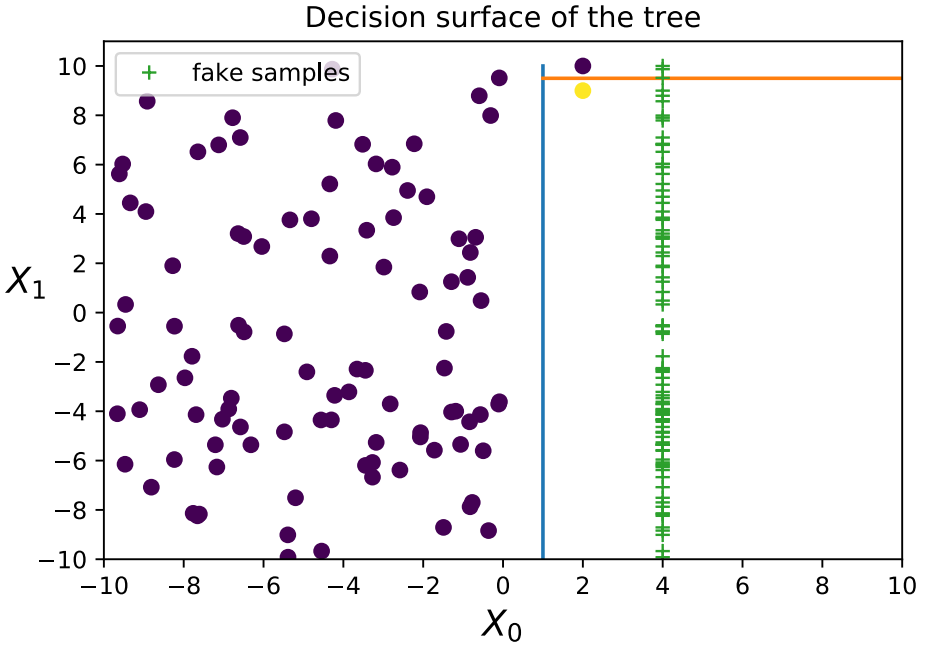
Essentially, we use information about how many samples were in each node during training

- Traverse the tree and give weight to each node as seen in training data
- See this blog for an easy to understand example

# A simpler example




Decision tree




Decision plot

Source: nicolas-hug.com

# Which one to choose?

True to the model: Interventional

- Useful when explaining models
- Don’t think of independence, but SCM and do-operations
  - See e.g. this or this GitHub discussion or Janzing et al. (2020) for SCM math
  - ‘Causal’ for the model
- Requires data (use a subset of training data)

True to the data: Path-dependent

- Useful when trying to replace black-box
- Avoids unrealistic datapoints
- Does not require data

# Enough theory

Lets talk plots

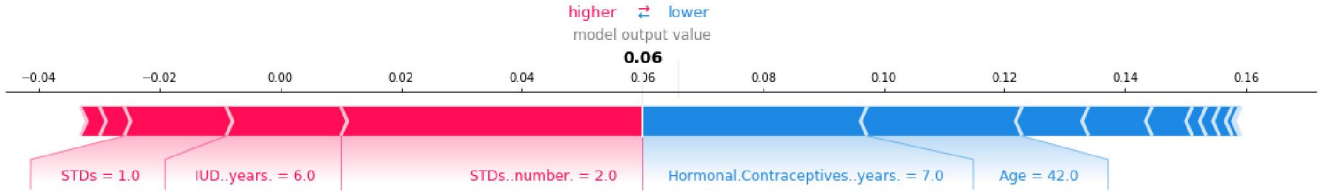
We remember that SHAP values are all individual

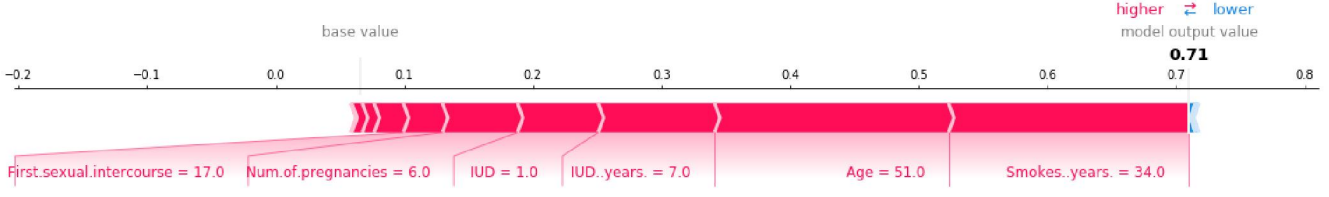
- Thus based on a collection of observations, i.e. a single, train set, test set or a group of interest

Through clever plots and summary statistics, we can still obtain global insights

All easy to compute and plot through SHAP

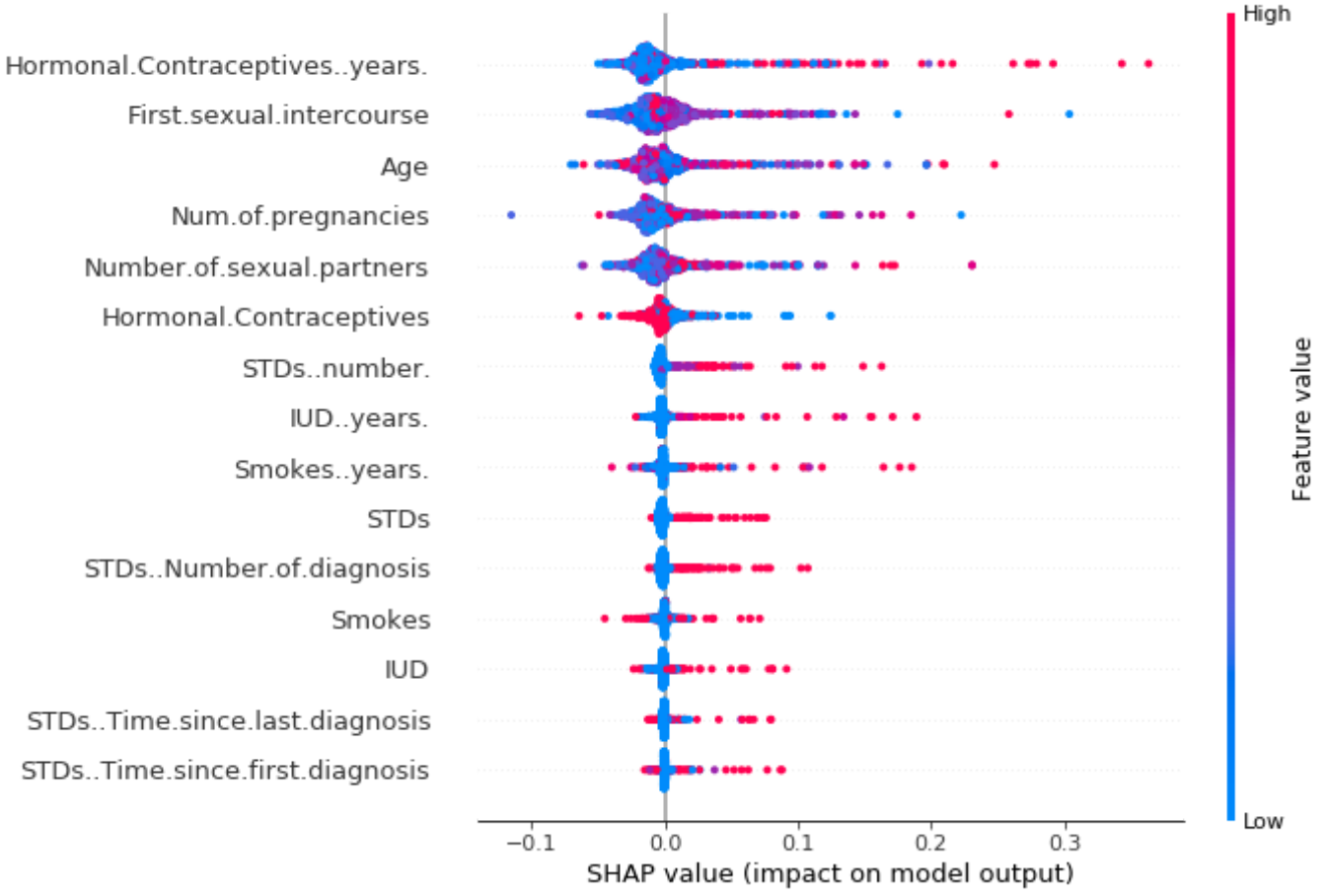
# Force plot



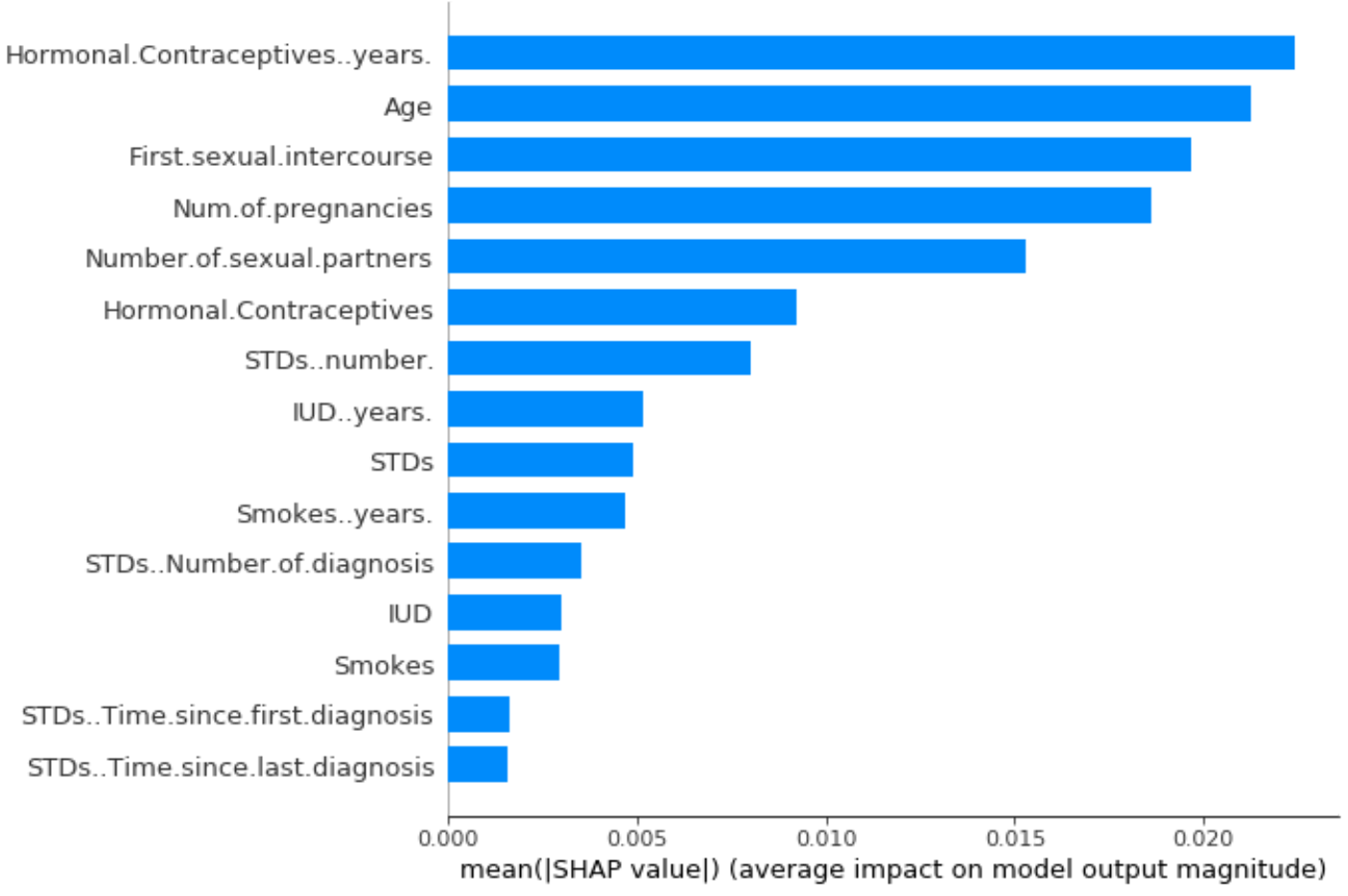

Source: Molnar, 2022

# Beeswarm plots



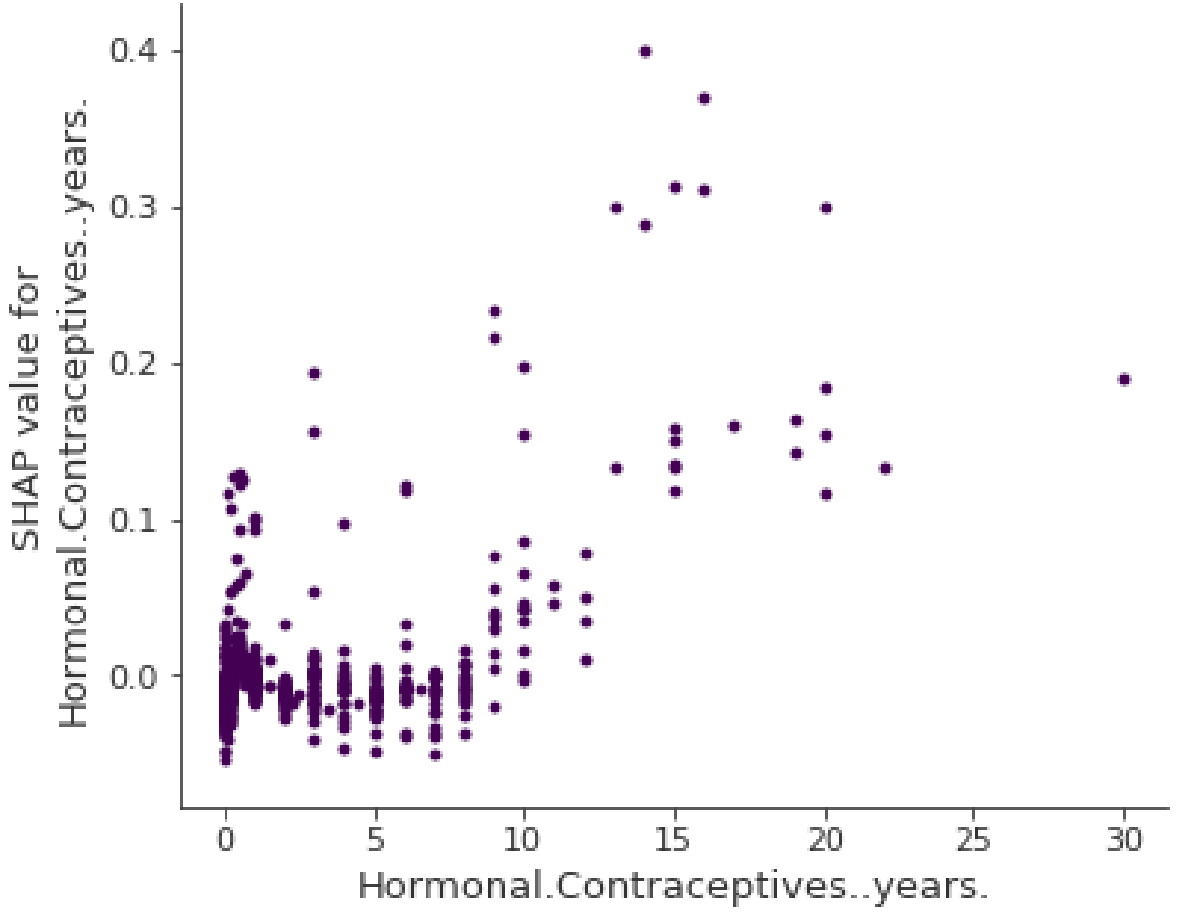

Source: Molnar, 2022

# Feature importance plots



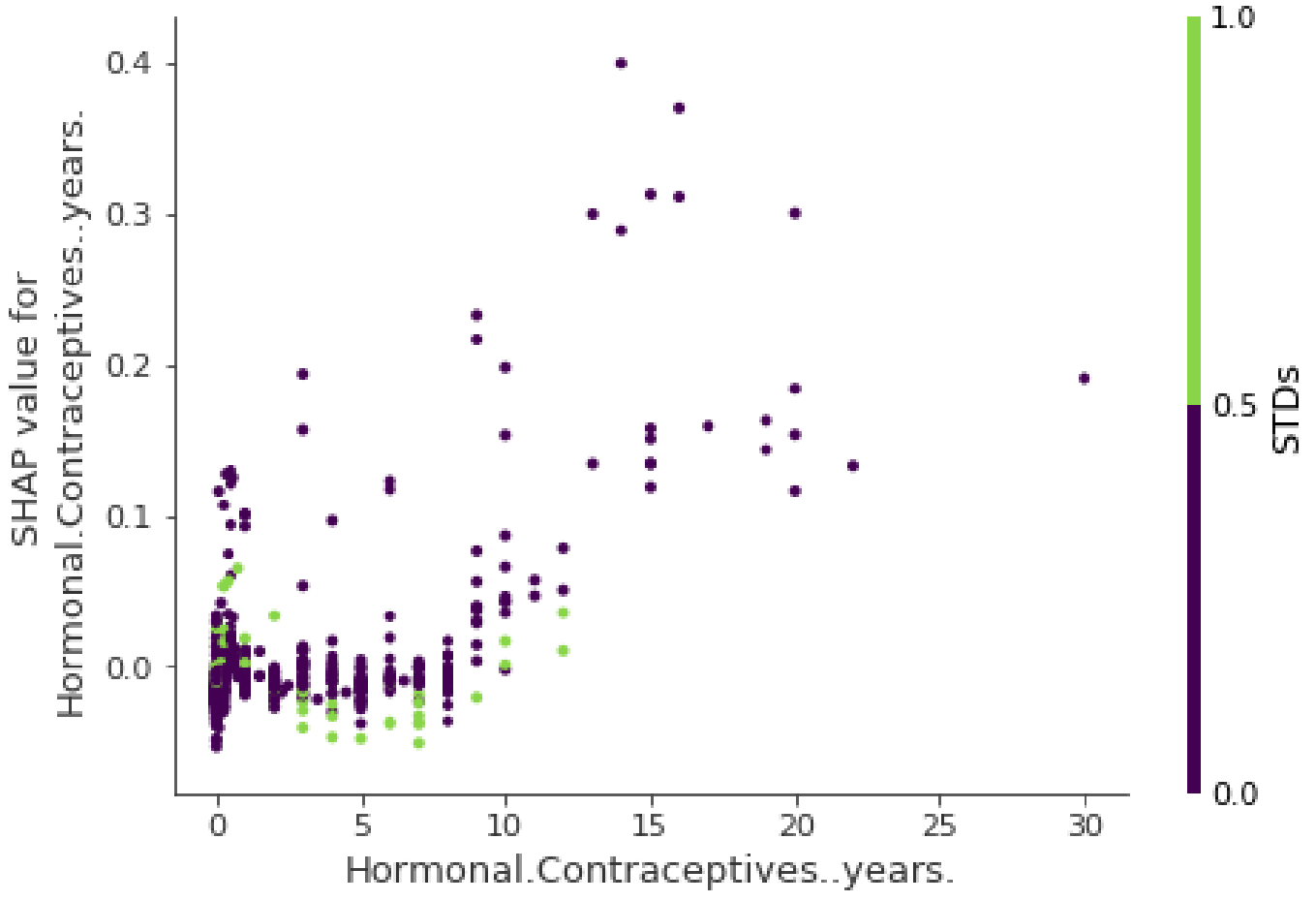

Source: Molnar, 2022

# Dependence plots




Source: Molnar, 2022

# Dependence plots with interactions




Source: Molnar, 2022

# Pros and cons

Pros

- Theoretical guarantees
- Easy to interpret
- Fast(ish) for tree-based models
- Local, but global aggregations

Cons

- Not fast(ish) for non-tree models
- True to the data or model for tree models
  - Or unrealistic datapoints for kernel
- Can be gamed (Slack et al., 2020)

# Further information

The rest of the book by Molnar has more methods

- e.g. counterfactual explanations

Text and image also have methods developed for them

- Not covered due to a mainly tabular focus in this course

# References

Janzing, D., Minorics, L., & Blöbaum, P. (2020, June). Feature relevance quantification in explainable AI: A causal problem. In International Conference on artificial intelligence and statistics (pp. 2907-2916). PMLR.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., … & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. Nature machine intelligence, 2(1), 56-67.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

Molnar, C. (2022). Interpretable machine learning, https://christophm.github.io/interpretable-ml-book/ (2022-12-14)

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020, February). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (pp. 180-186).

# To the exercises!