Inherently Interpretable models II

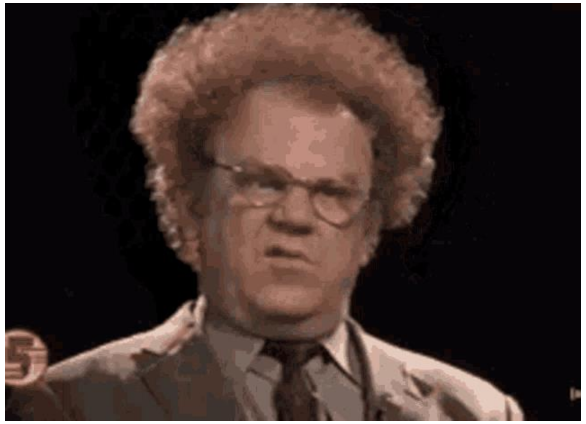
Szymon Bobek

Jagiellonian University
2024

JAGIELLONIAN UNIVERSITY
IN KRAKÓW

https://geist.re

Chinese room



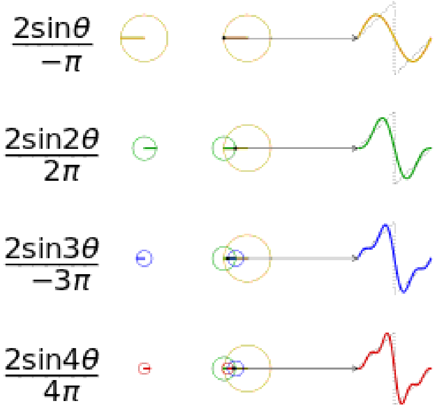
- Invented by John Searle (1980): Critique of AI's potential for true understanding.
- Thought Experiment Setup: A person manipulates Chinese symbols using instructions.
- Key Point: The person follows rules without understanding the language's meaning.
- Challenge to Strong AI: Machines can simulate but not truly comprehend language.
- Conclusion: Syntax alone is insufficient for real understanding or consciousness.
- Is Chinese room interretable/explainable?

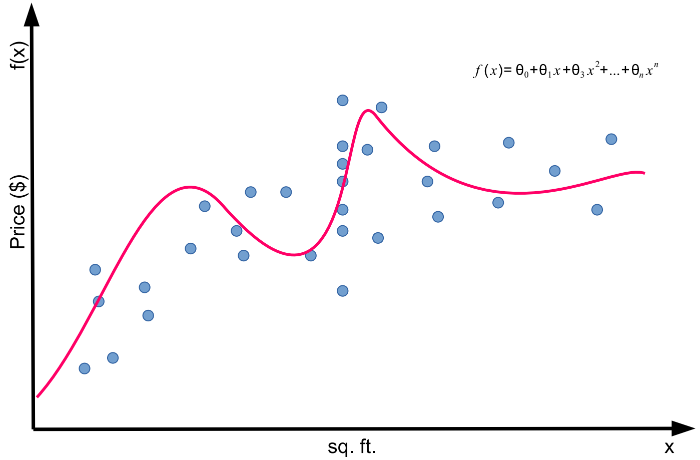
Neural Networks are Decision Trees

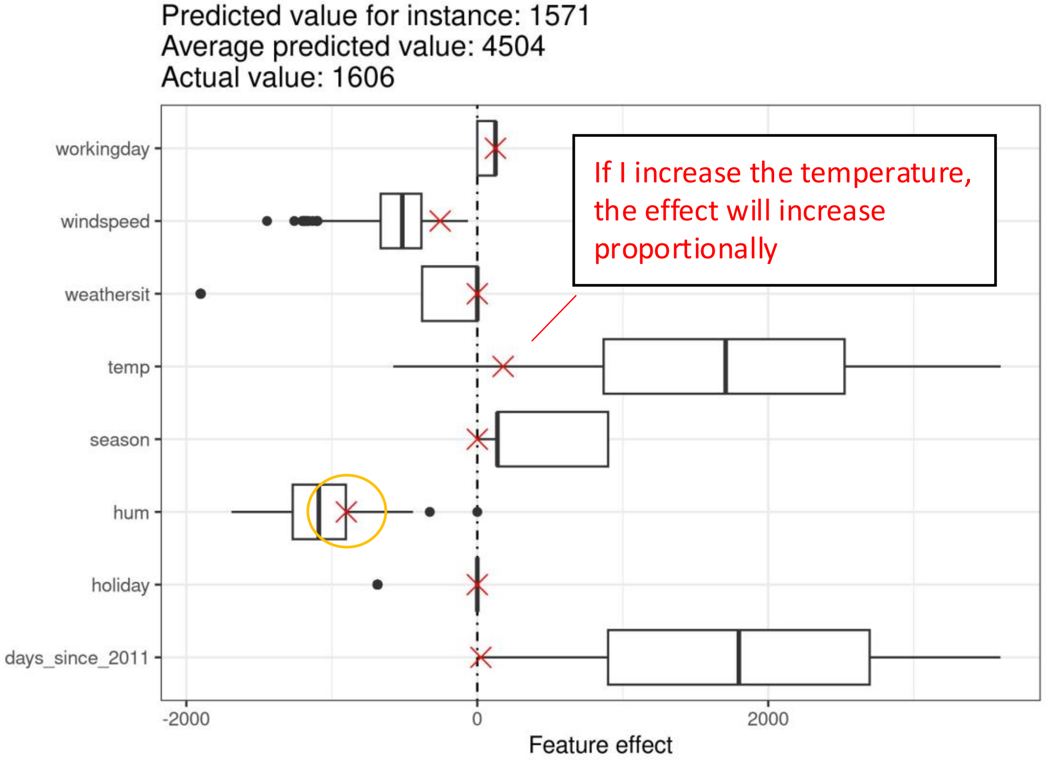
Global model-agnostic explanations

Interpretability issues of additive models

- Feature transformations can break interpretability
- Multicolinearity can break the interpretability
- **Feature interaction can break interpretability**







The effect of a feature for linear regression represents the effect of a feature value on a prediction, assuming all other features are fixed

Partial dependence function?

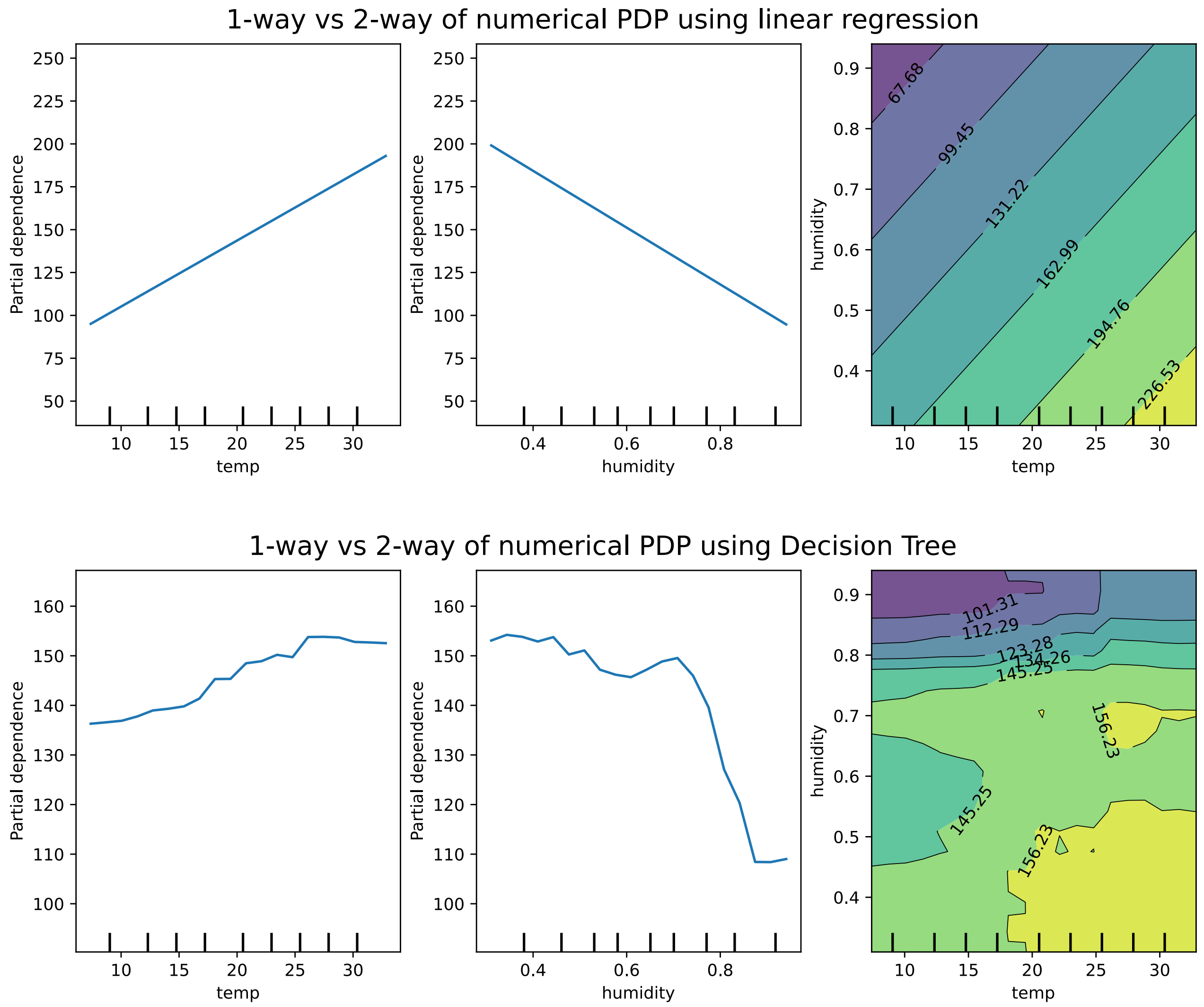
- **Feature interactions cannot be captured by all types of models**

Average over all instances

$$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_S, x_C^{(i)})$$

Features for which we estimate partial dependence. Cannot be correlated with other features

Features values from dataset we are not interested in



Partial dependence function?

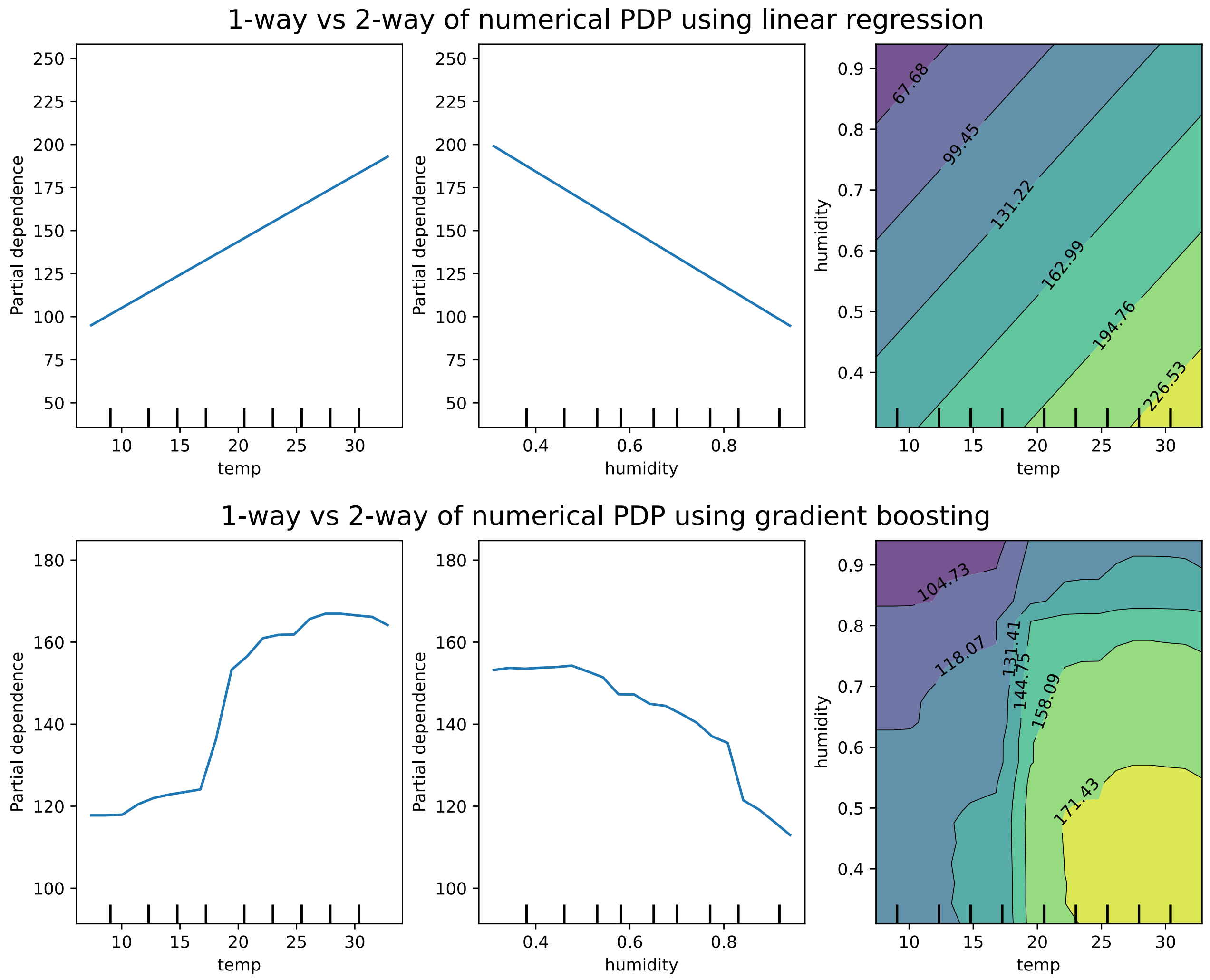
- **Feature interactions cannot be captured by all types of models**

Average over all instances

$$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_S, x_C^{(i)})$$

Features for which we estimate partial dependence. Cannot be correlated with other features

Features values from dataset we are not interested in



Fow to measure feature interaction?

- Partial dependence function
- **H-statistic**

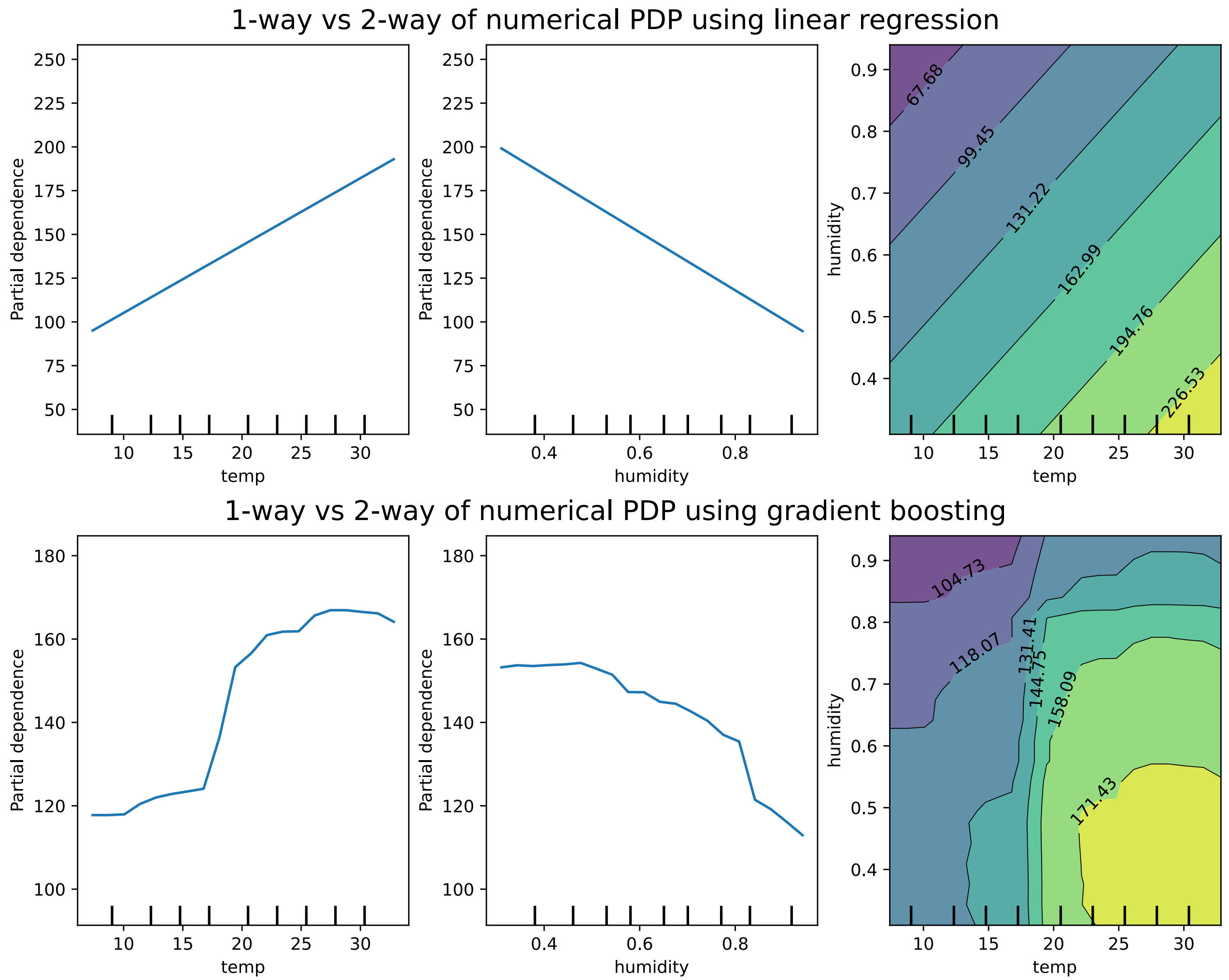
$$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}(x_S, x_C^{(i)})$$

$$PD_{jk}(x_j, x_k) = PD_j(x_j) + PD_k(x_k)$$

$$\hat{f}(x) = PD_j(x_j) + PD_{-j}(x_{-j})$$

$$H_{jk}^2 = \frac{\sum_{i=1}^{n}\left[PD_{jk}(x_j^{(i)}, x_k^{(i)}) - PD_j(x_j^{(i)}) - PD_k(x_k^{(i)})\right]^2}{\sum_{i=1}^{n} PD_{jk}^2(x_j^{(i)}, x_k^{(i)})}$$

$$H_j^2 = \frac{\sum_{i=1}^{n}\left[\hat{f}(x^{(i)}) - PD_j(x_j^{(i)}) - PD_{-j}(x_{-j}^{(i)})\right]^2}{\sum_{i=1}^{n}\hat{f}^2(x^{(i)})}$$

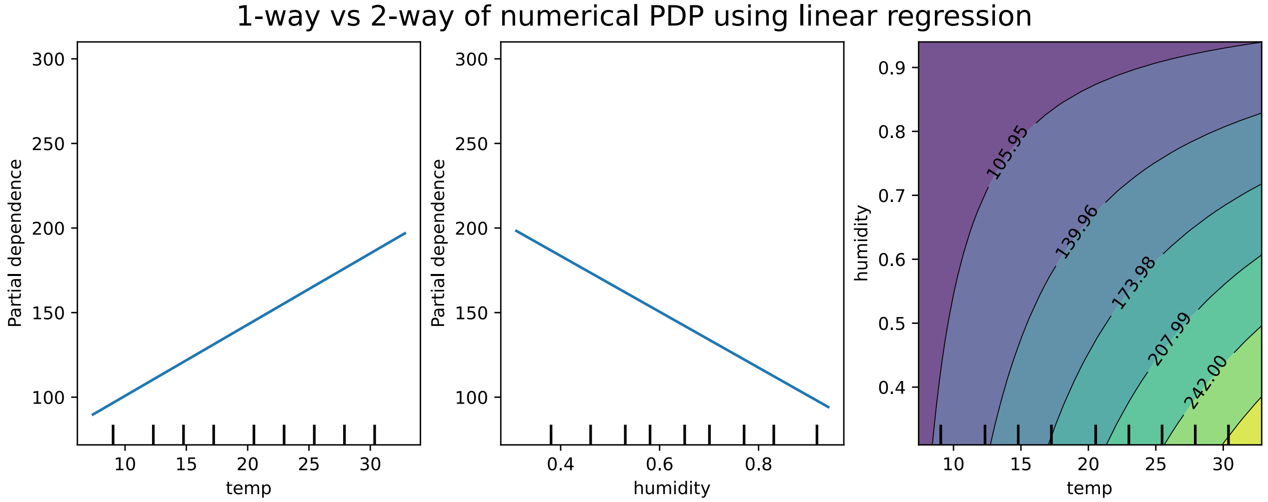


How to add interactions to linear model?

- We can add interactions manually
- We can use feature engineering tool to generate multiple features
- But this breaks the interpretability
- We can use decision trees
- We can use decision rules

Add interactions by creating features that are products of each other.
How to interpret that?

$$X = \{x_1, x_2, \ldots, x_n\} \to \{x_1x_2, x_1x_3, \ldots, x_nx_k\}$$



Decision rules

What are decision rules

- **OneR** learns rules from a single feature. OneR is characterized by its simplicity, interpretability and its use as a benchmark.
- **Sequential covering** is a general procedure that iteratively learns rules and removes the data points that are covered by the new rule. This procedure is used by many rule learning algorithms
- **Bayesian Rule Lists** combine pre-mined frequent patterns into a decision list using Bayesian statistics. Using pre-mined patterns is a common approach used by many rule learning algorithms.

$C : c_1 \wedge c_2 \wedge \ldots \wedge c_n$ $\quad$ $D : d_1 \wedge d_2 \wedge \ldots \wedge d_n$

IF $\boxed{C}$ THEN $\boxed{D}$

$$\text{Support of } [C \Rightarrow D] = P(C) \leftarrow \text{This is different than in association rules}$$

$$\text{Confidence of } [C \Rightarrow D] = P(C|D) = \frac{P(C \cap D)}{P(C)}$$

$$\text{Lift of } [C \Rightarrow D] = \frac{P(C \cap D)}{P(C) \times P(D)}$$

Rules form sets. This does not imply any order in which they should be processed. That is why conflict-resolution techniques are used to determine which rule should be fired.

OneR

Discretize the continuous features by choosing appropriate intervals.

For each feature

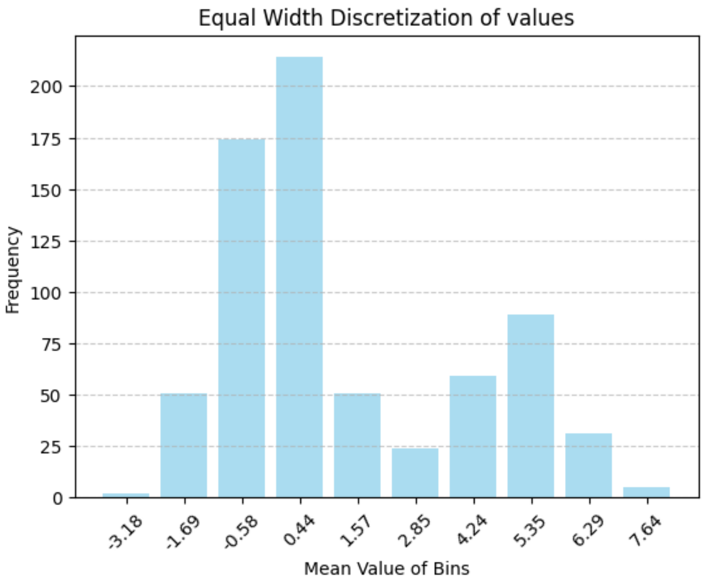
- Create a cross table between the feature values and the (categorical) outcome.
- For each value of the feature, create a rule which predicts the most frequent class of the instances that have this particular feature value
- Calculate the total error of the rules for the feature.

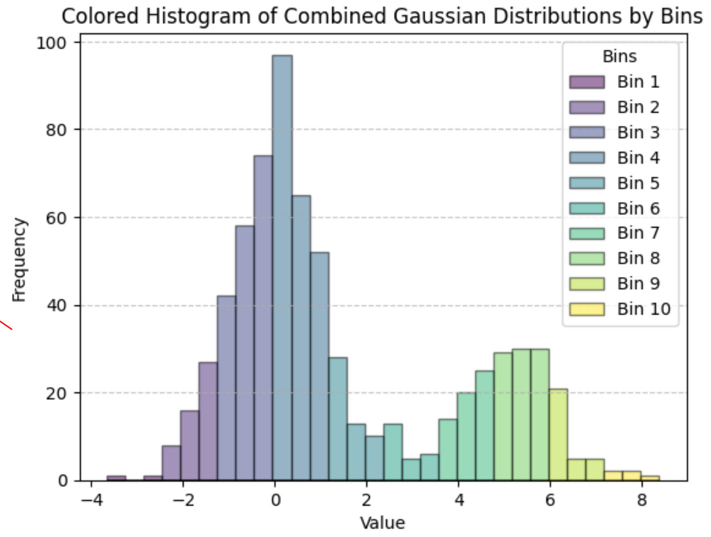
Select the feature with the smallest total error.

Discretize continuous features

- Equal-width discretization
- Equal-frequency discretization
- Discretization with clustering algorithm
- Discretization using decision trees (or EBM, or any other model that cuts continuotus values)
- Other

The width of the bin is constant

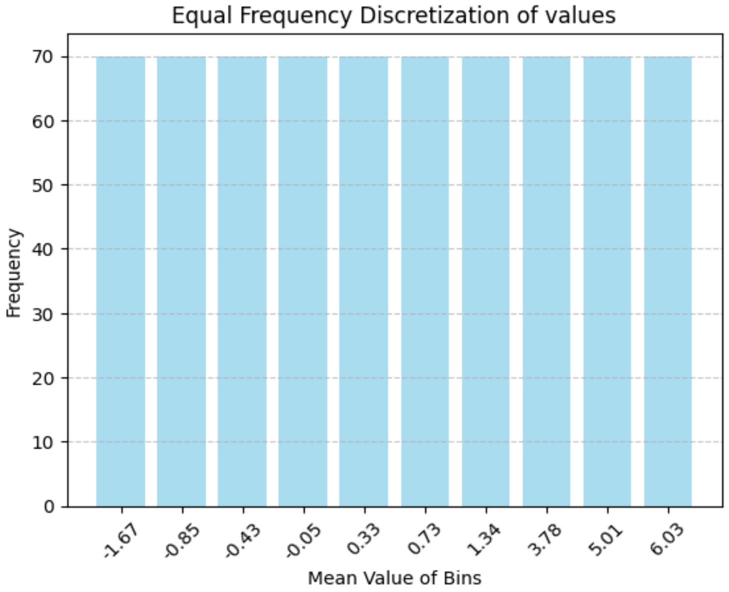


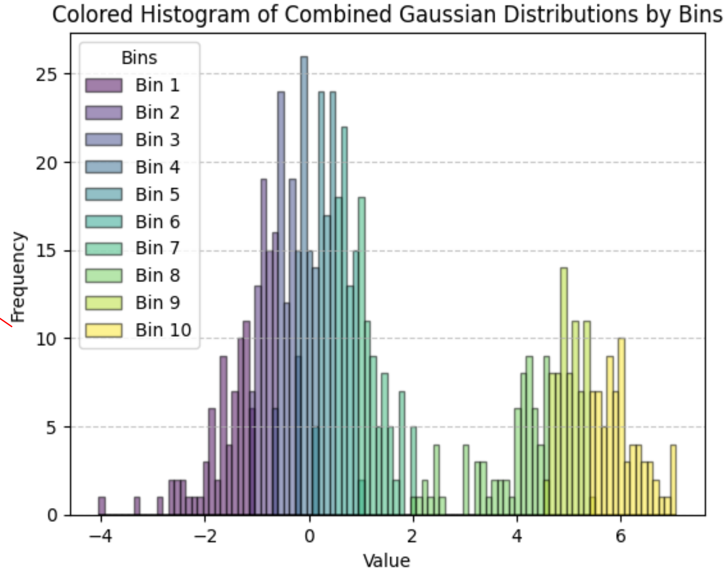


Discretize continuous features

- Equal-width discretization
- Equal-frequency discretization
- Discretization with clustering algorithm
- Discretization using decision trees (or EBM, or any other model that cuts continuotus values)
- Other

The width of the bin is variable (hence overlapping) in the second plot

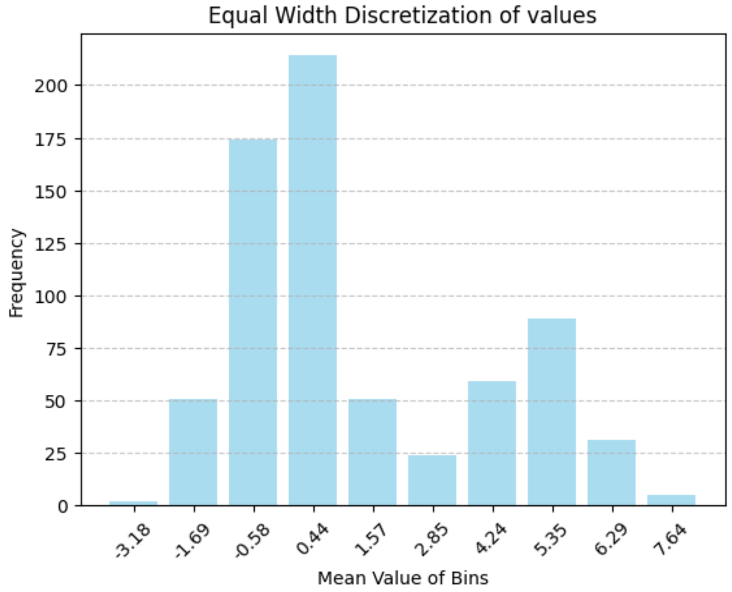


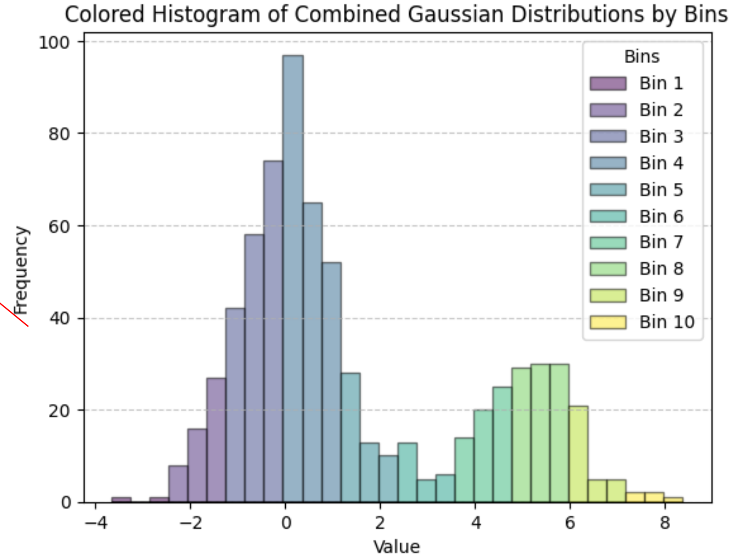


Discretize continuous features

- Equal-width discretization
- Equal-frequency discretization
- Discretization with clustering algorithm
- Discretization using decision trees (or EBM, or any other model that cuts continuotus values)
- Other

We can discretize multiple values at once

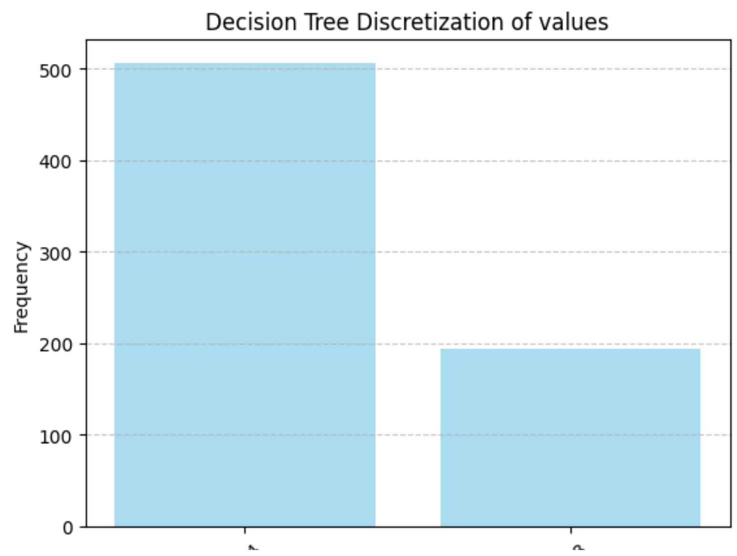


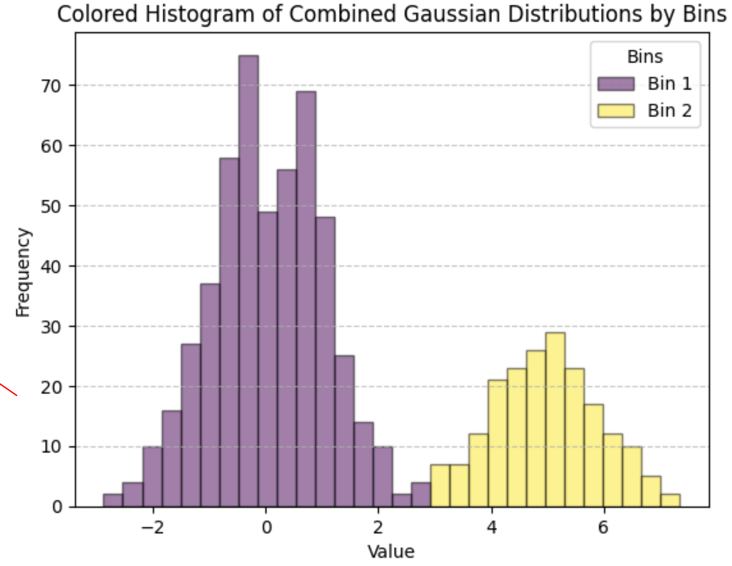


Discretize continuous features

- Equal-width discretization
- Equal-frequency discretization
- Discretization with clustering algorithm
- Discretization using decision trees (or EBM, or any other model that cuts continuotus values)
- Other

In this example we assume that there are two classess, and that the class 1 is for values > 3





Generate OneR

location	size	renovated	value
good	small	no	high
good	big	yes	high
good	big	yes	high
bad	medium	yes	medium
good	medium	partially	medium
good	small	partially	medium
bad	medium	no	medium
bad	small	no	low
bad	medium	no	low
bad	small	yes	low

location	value=low	value=medium	value=high
bad	3	2	0
good	0	2	3

size	value=low	value=medium	value=high
big	0	0	2
medium	1	3	0
small	2	1	1

renovated	value=low	value=medium	value=high
yes	1	1	2
partially	0	2	0
no	2	1	1

Generate OneR

Error = No Mistakes / All Predictions

location	size	renovated	value
good	small	no	high
good	big	yes	high
good	big	yes	high
bad	medium	yes	medium
good	medium	partially	medium
good	small	partially	medium
bad	medium	no	medium
bad	small	no	low
bad	medium	no	low
bad	small	yes	low

location	value=low	value=medium	value=high
bad	$\boxed{3}$	2	0
good	0	2	$\boxed{3}$

size	value=low	value=medium	value=high
big	0	0	$\boxed{2}$
medium	1	$\boxed{3}$	0
small	$\boxed{2}$	1	1

renovated	value=low	value=medium	value=high
yes	1	1	$\boxed{2}$
partially	0	$\boxed{2}$	0
no	$\boxed{2}$	1	1

Sequential covering

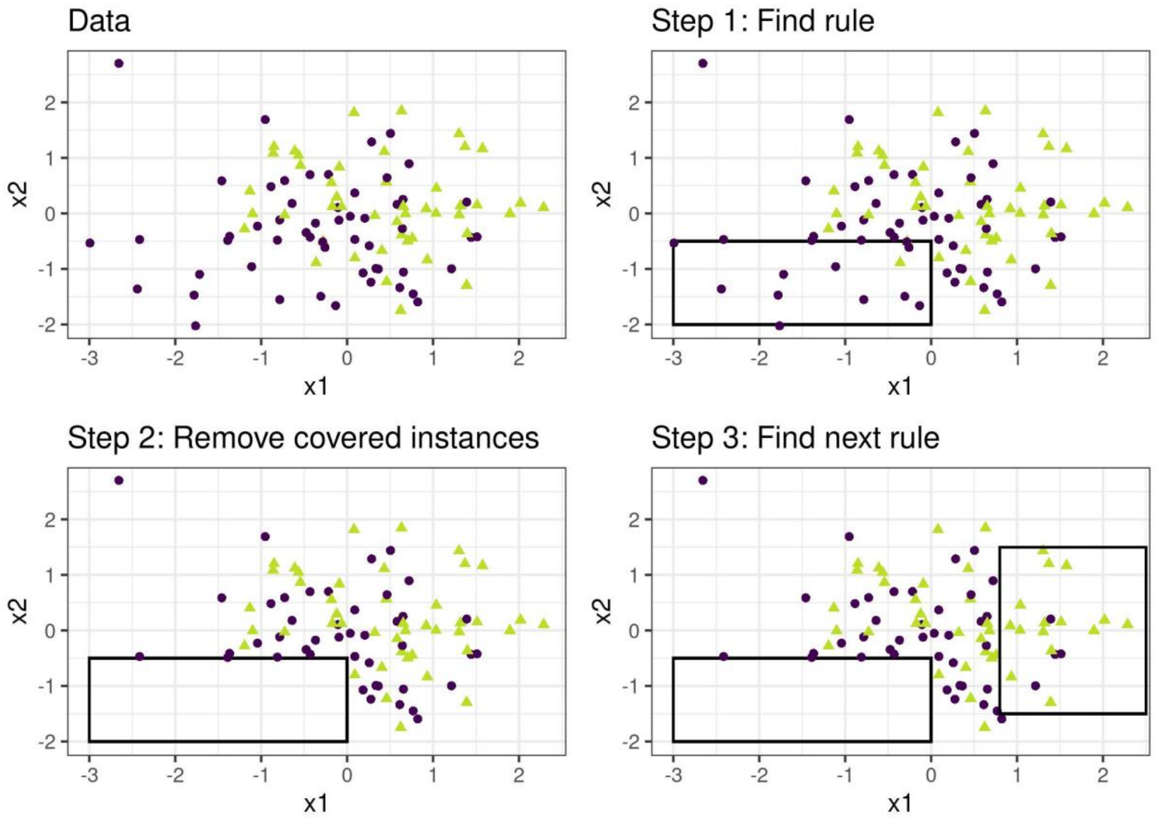
Start with an empty list of rules (RList).

Learn a rule R

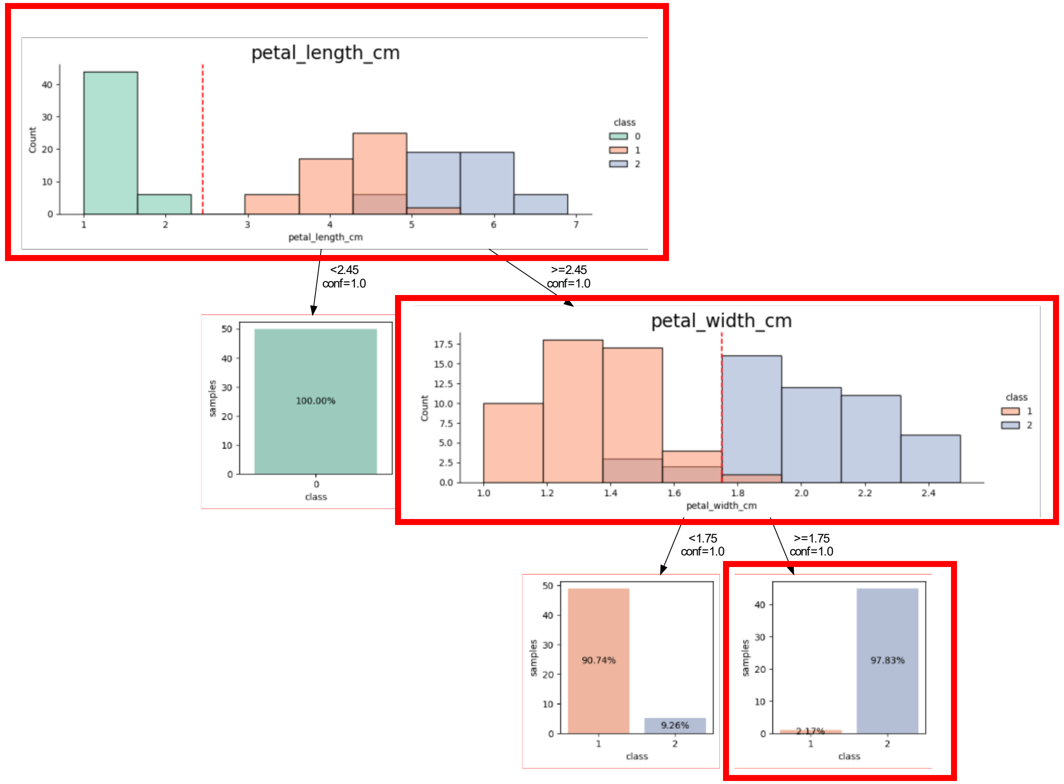
While the list of rules is below a certain quality threshold (or positive examples are not yet covered):

- Add rule R to RList.
- Remove all data points covered by rule R
- Learn another rule on the remaining data.

Return the decision list.



How to learn a rule? Decision trees revisited



- Decision trees can be used to extract rules
- There are many tree-based methods that have excellent performance on tabular data
- Can we use strengths of decision trees/decision rules of capturing the interactions and linear models' simplicity?
- Implementation in Python (RIPPER)

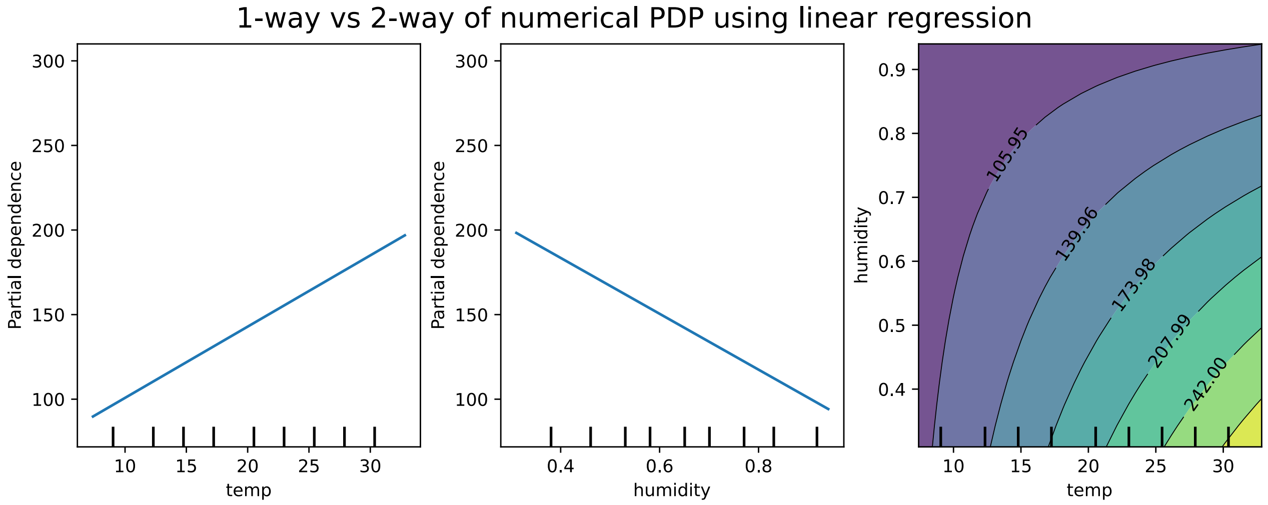
RuleFit

How to add interactions to linear model?

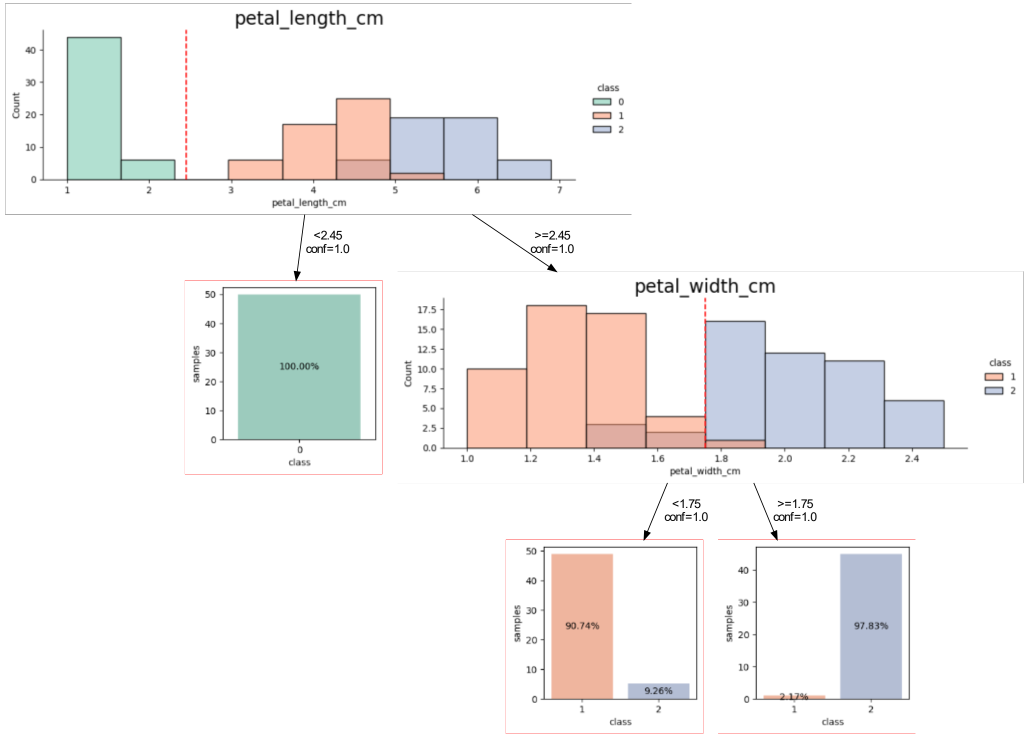
- We can add interactions manually
- We can use feature engineering tool to generate multiple features
- But this breaks the interpretability
- We can use decision trees
- We can use decision rules
- We can **combine strengths of decision rules, trees and linear models**

Add interactions by creating features that are product of each other.
How to interpret that?

$$X = \{x_1, x_2, \ldots, x_n\} \rightarrow \{x_1 x_2, x_1 x_3, \ldots, x_n x_k\}$$



One-Hot-Encoding with rules/trees



We are not that much interested about the preediction, but in rules capturing interactions

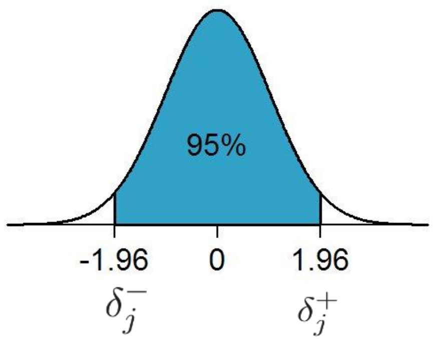
$$
\begin{aligned}
r_1(x) &= (\text{petal\_length\_cm} < 2.45)\\
r_2(x) &= (\text{petal\_length\_cm} >= 2.45)\\
r_3(x) &= (\text{petal\_length\_cm} >= 2.45 \wedge \text{petal\_width\_cm} < 1.75)\\
r_4(x) &= (\text{petal\_length\_cm} >= 2.45 \wedge \text{petal\_width\_cm} >= 1.75)
\end{aligned}
$$

Number of terminal nodes in a binary tree of depth N is 2^N

We can get more trees: number of rules we can generate from M trees with t_m terminal nodes

$$K = \sum_{m=1}^{M} 2(t_m - 1)$$

Linear model of rules and variables



Mathematical version of OHE of rule

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{k=1}^{K} \hat{\alpha}_k r_k(x) + \sum_{j=1}^{p} \hat{\beta}_j l_j(x_j)$$

$$r_m(x) = \prod_{j \in \mathrm{T}_m} I(x_j \in s_{jm})$$

$$r_k(x) \leftarrow r_k(x)/t_k$$

$$l_j(x_j) = 0.4 \cdot l_j^*(x_j)/std(l_j^*(x_j))$$

Normalization term to give all the linear terms same prior influence as a typical rule

$$l_j^*(x_j) = min(\delta_j^+, max(\delta_j^-, x_j))$$

Removing outliers, by *clipping* values with quantiles

$$\lim_{N \to \infty} \left[\frac{1}{N} \sum t_k(s_k) \right] = 0.4, \text{there } s_k \sim U(0,1))$$

Average standard deviation of a rule with support drawn from uniform distribution

$$t_k = \sqrt{s_k(1 - s_k)}$$

Scale of a rule (standard deviation). It's calculated as for binomial distribution, because the rule terms are defined as OHE (0 or 1)

$$s_k = \frac{1}{N} \sum_{i=1}^{N} r_k(x_i)$$

Support of a rule

$$K = \sum_{m=1}^{M} 2(t_m - 1)$$

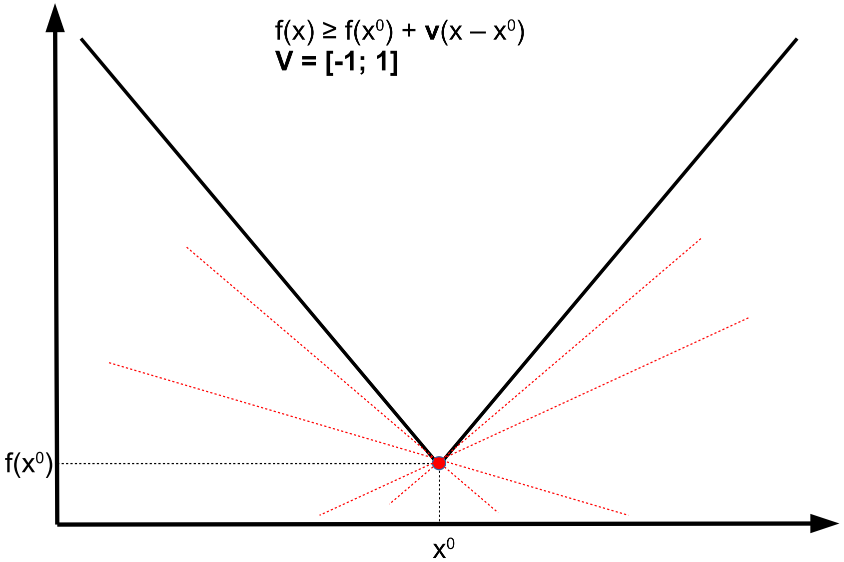
$$(\{\hat{\alpha}\}_1^K, \{\hat{\beta}\}_0^p) = \quad argmin_{\{\hat{\alpha}\}_1^K, \{\hat{\beta}\}_0^p} \sum_{i=1}^{n} L(y^{(i)}, \hat{f}(x^{(i)})) \\ + \qquad \lambda \cdot \left(\sum_{k=1}^{K} |\alpha_k| + \sum_{j=1}^{p} |\beta_j| \right)$$

Number of parameters used in this optimizaiton function can grow very fast. This is against the interpretability!

Friedman, Jerome H, and Bogdan E Popescu. “Predictive learning via rule ensembles.” The Annals of Applied Statistics. JSTOR, 916–54. (2008).

Lasso – solution to large rule set

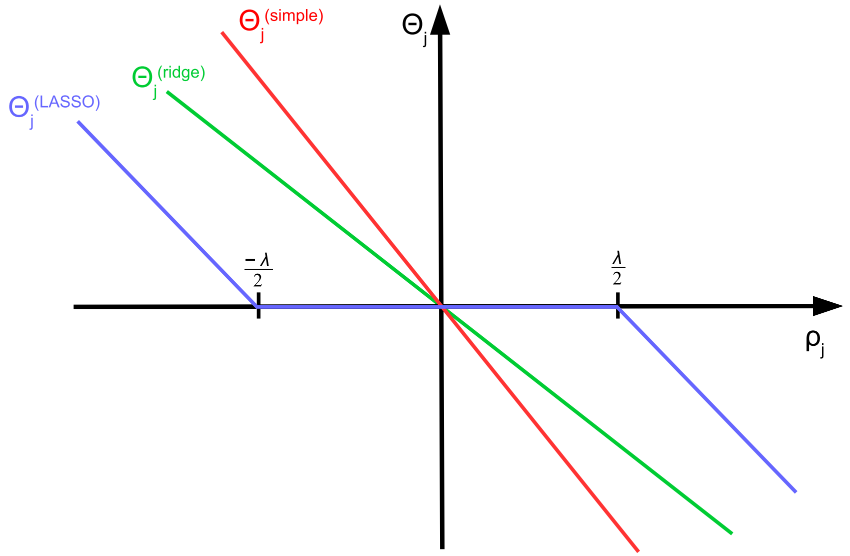
Lasso and subgradients



$$J(\theta) = \frac{1}{2N}\sum_{i}^{N}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2N}\sum_{i=2}^{N}|\theta_i|$$

$$\boxed{\bar{x}_j{}^{(i)} = \frac{x_j^{(i)}}{\sqrt{\sum_i^N (x_j^{(i)})^2}}}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{N}\sum_{i}^{N}\left[\sum_{k}^{D}(\theta_k \bar{x}_k^{(i)} - y^{(i)})\bar{x}_j^{(i)}\right] + \frac{\lambda}{2N}\frac{\partial|\theta|}{\partial \theta_j}$$



$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{N}\underbrace{\sum_{i}^{N}\left[\sum_{k\neq j}^{D}\theta_k(\bar{x}_k^{(i)}) - y^{(i)})\bar{x}_j^{(i)}\right]}_{\rho_j} + \theta_j\frac{1}{N}\sum_{i}^{N}\boxed{\left[(\bar{x}_j^{(i)})^2\right]} + \frac{\lambda}{2N}\frac{\partial|\theta|}{\partial \theta_j} =$$

$$= \frac{1}{N}\rho_j + \frac{1}{N}\theta_j + \begin{cases} -\frac{\lambda}{2N} & \text{if } \theta_j < 0 \\ [-\frac{\lambda}{2N}; \frac{\lambda}{2N}] & \text{if } \theta_j = 0 \\ \frac{\lambda}{2N} & \text{if } \theta_j > 0 \end{cases} \qquad \theta_j = \begin{cases} \theta_j = -\rho_j + \frac{\lambda}{2} & \text{if } \rho_j > \frac{\lambda}{2} \\ \theta_j = 0 & \text{if } \rho_j \in < -\frac{\lambda}{2}; \frac{\lambda}{2} > \\ \theta_j = -\rho_j - \frac{\lambda}{2} & \text{if } \rho_j < -\frac{\lambda}{2} \end{cases}$$

Optimal solution when gradient is 0

Interpretation of RuleFit models

- The interpretation of the importance proposed in RuleFit is the absolute version of standarized predictor coefficient
- The standarized coefficient is measured in units of standard deviation
- We initially standarized the features, so we do not interpret the coefficients in terms of effect
- If we want, whe should scale them back

$\cancel{I_j = |\hat{\beta}_j| \cdot std(l_j^*(x_j))}$ $$I_j = \frac{|\hat{\beta}_j|}{std(l_j^*(x_j))}$$ $$l_j^*(x_j) = min(\delta_j^+, max(\delta_j^-, x_j))$$

$\cancel{I_k = |\hat{\alpha}_k| \cdot \sqrt{s_k(1-s_k)}}$ $$I_k = \frac{|\hat{\alpha}_k|}{\sqrt{s_k(1-s_k)}}$$ $$t_k = \sqrt{s_k(1-s_k)}$$

This seems to be incorrect in the original paper, as the trained coefficents are already standarized

$$J_j(x) = I_j(x) + \sum_{x_j \in r_k} I_k(x)/m_k$$

I_k the importance of the decision rules in which x_j appears, and m_k is the number of features constituting the rule r_k

$$J_j(X) = \sum_{i=1}^{n} J_j(x^{(i)})$$

Global importance of a feature

Explainable Boosting Machines

Gradient boosting (re)explainer

Step 1: $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

Initialize the first model with constant value. It happens to be average. Try it – calculate gradient, make it equal zero, compute constant

Step 2: For $m = 1 \to M$ **repeat:**

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n$$

Calculate pseudo-residuals

Train model $h_m(x)$ to predict pseudo-residuals

Basically use dataset: $\{(x_i, r_{im})\}_{i=1}^{n}$

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$$

Solve simple 1D optimization problem with respect to γ_m

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Update model by adding gradient of residuals to the previous guess.

In practise the sum is modified by the learning rate parameter. Here the learning rate is 1

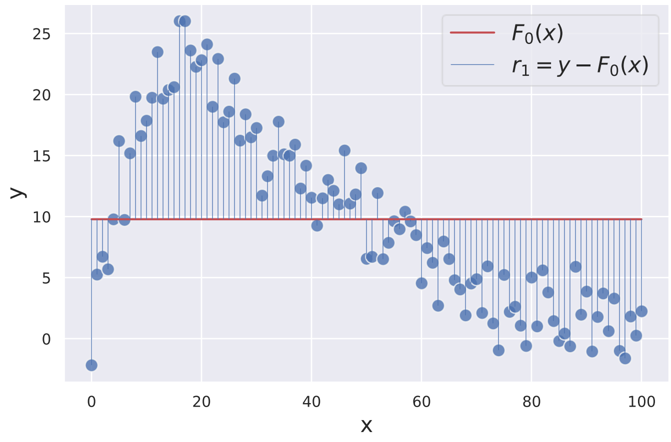
Step 3: $F_M(x)$

Return

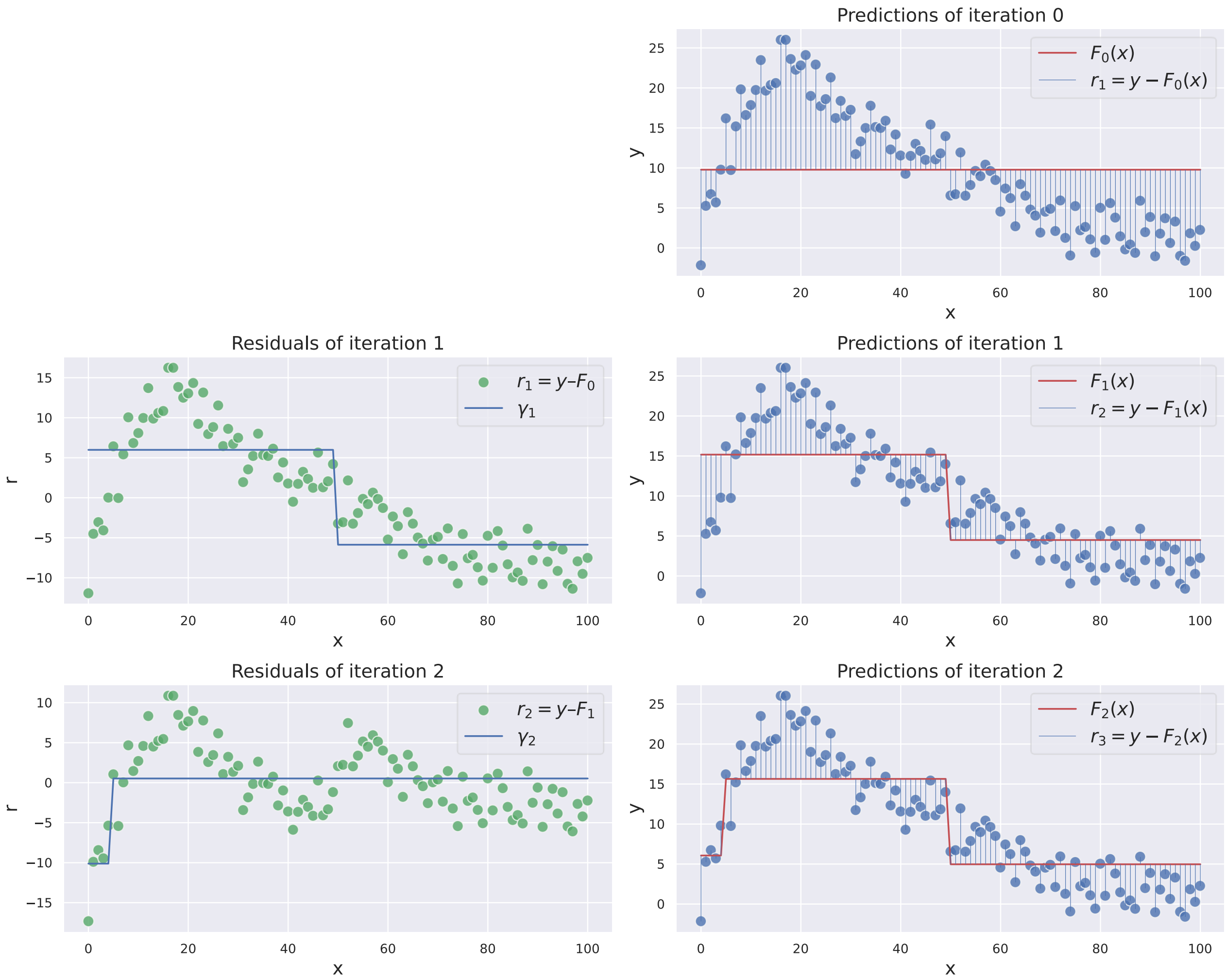
$$L(y_i, F(x_i)) = \frac{1}{2}\left[y_i - F(x_i)\right]^2$$

$$\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} =$$

$$= \frac{2}{2}\left[y_i - F(x_i)\right] \cdot 1 = y_i - F(x_i)$$



Gradient Boosting example



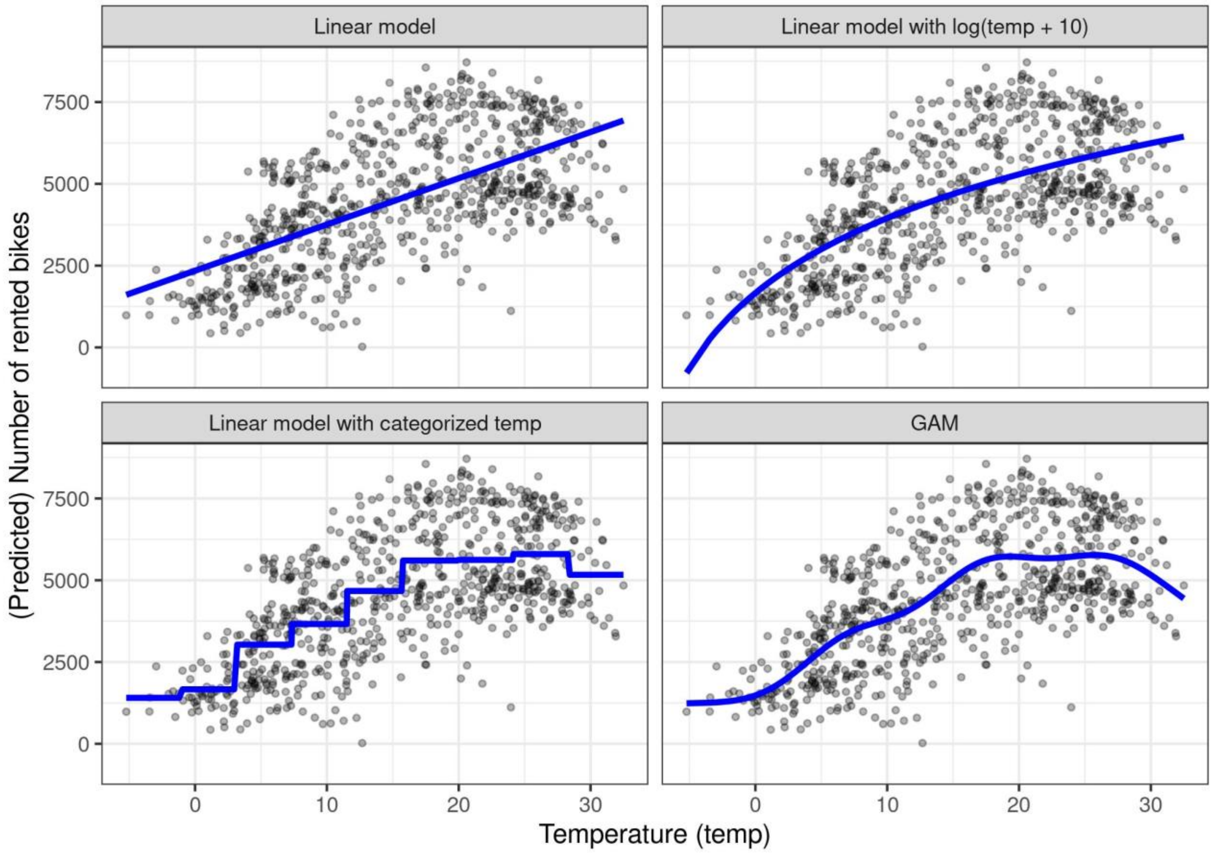
- One can easily see that in our case the γ_m is always the average of residuals
- In each step, the residual component is added to the main function
- It basicaly works as gradient descent, but in the feature-values space, not parameter space

Want to learn more? Here is a nice set of videos (very beggining level): [Video]

Generalized Additive Models

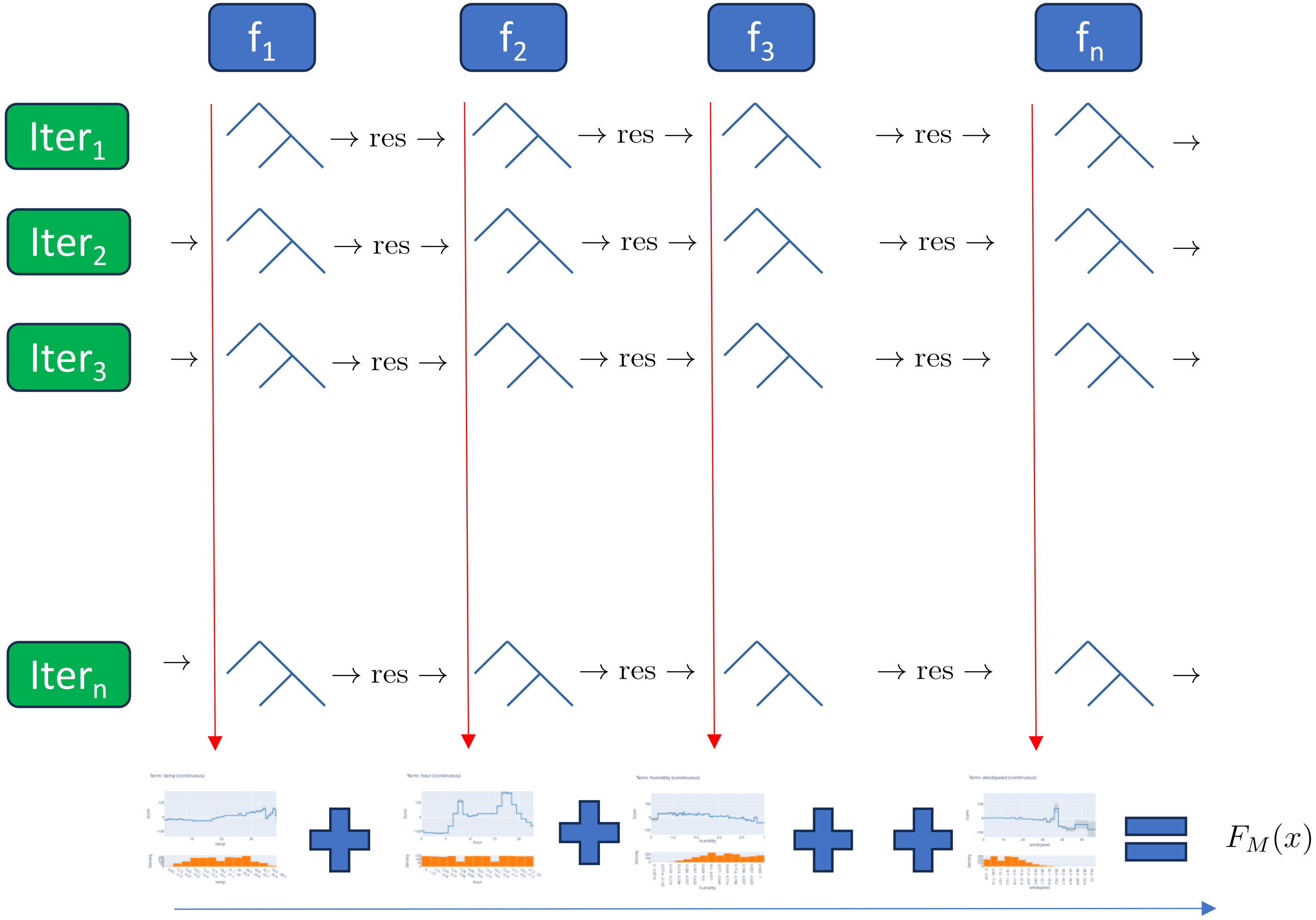
$$g(E_Y(y|x)) = \beta_0 + \beta_1 x_1 + \dots \beta_p x_p \rightarrow \quad g(E_Y(y|x)) = \beta_0 + f_1(x_1) + f_2(x_2) + \dots + f_p(x_p)$$

- GAMs are generalizations of linear models, where linear terms can now be nonlinear functions
- The question is how to learn the nonlinear functions?
- Splines are one of the solutions



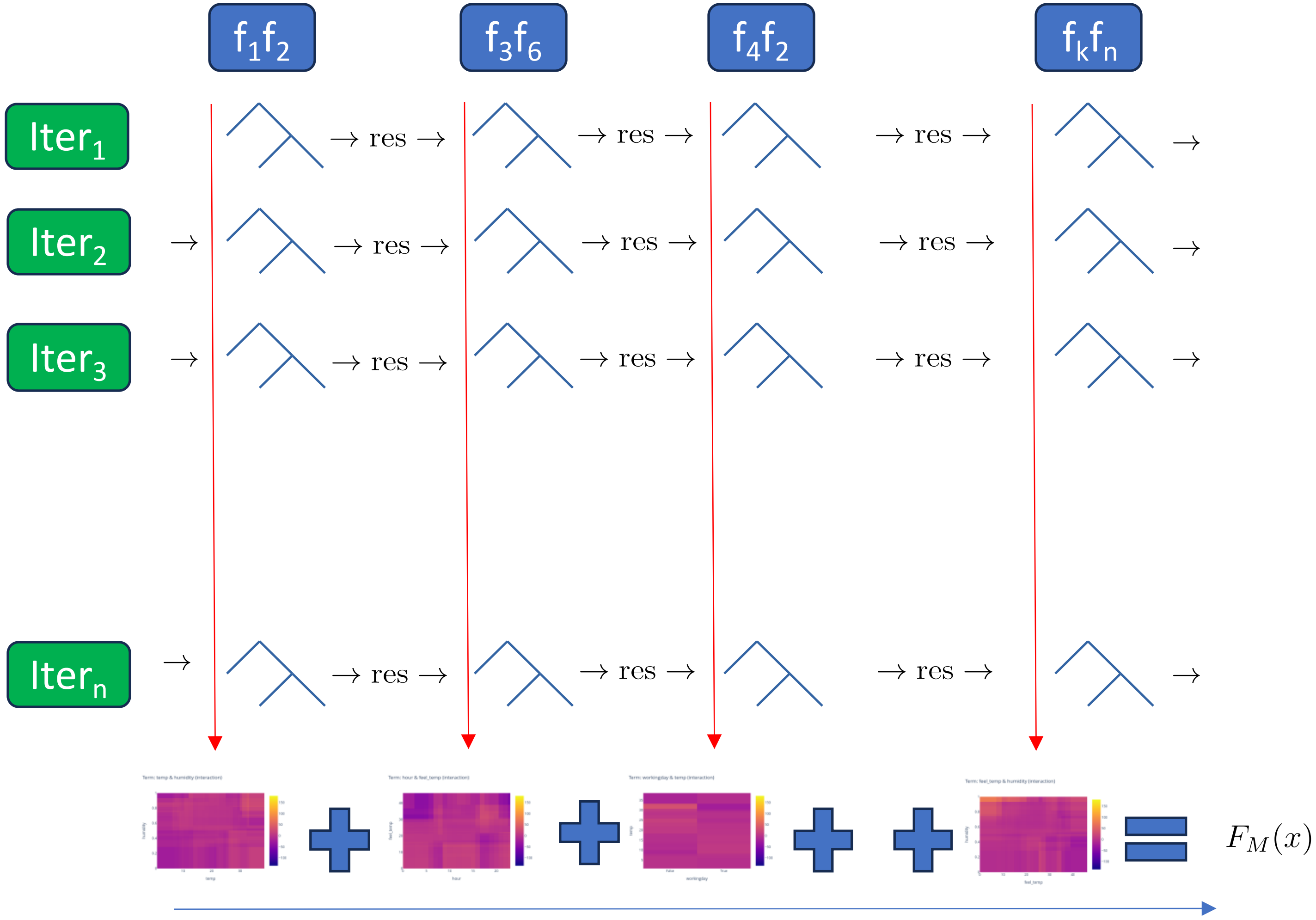
Explainable Boosting Machines

- Combines idea of gradient boosting and GAMs
- Allows to include pairwise interactions in the model
- Is as efficient as blackbox gradient boosting models, but gives intelligibility
- It is one of very few models that is editable!



Explainable Boosting Machines

- Learning rate is very small, so the order of the features does not matter
- The features are selected in round-robin manner
- After model is fitted, the interactions are added
- The interactions are added automatically, by previously estimating their strength



Thank you for your attention!

JAGIELLONIAN UNIVERSITY
IN KRAKÓW

https://geist.re