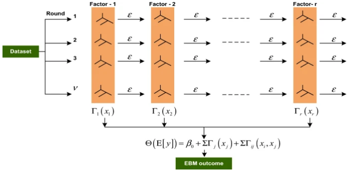
# Interpretable Machine Learning

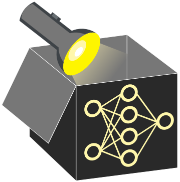
# Interpretable Models 2
# Explainable Boosting Machines (EBM)



## Learning goals

- Understand link between GAM and EBM
- Learn univariate EBMs ≙ GAM + boosting + shallow bagged trees
- Extend to GA2M: GAMs with selected pairwise interactions
- Detect interactions efficiently using FAST algorithm

# RECAP: SPLIT SELECTION DECISION TREE



- **Impurity (Regression):** Variance of target $Y$ in a node:

$$\text{Var}(Y) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \bar{y})^2 = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)})^2 - \bar{y}^2$$

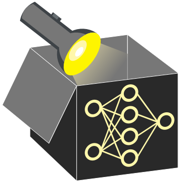
- **Sum of squared errors (SSE) = residual sum of squares (RSS)**:

$$\text{RSS} = n \cdot \text{Var}(Y) = \sum_{i=1}^{n}(y^{(i)} - \bar{y})^2 = \cdots = \sum_{i=1}^{n}(y^{(i)})^2 - \frac{1}{n}\left(\sum_{i=1}^{n} y^{(i)}\right)^2$$

$$\text{Hence: } \boxed{\text{RSS} = SS_n - \frac{S_n^2}{n} \quad \text{with} \quad S_n = \sum_{i=1}^{n} y^{(i)},\ SS_n = \sum_{i=1}^{n}(y^{(i)})^2}$$

- **Split criterion:**
  - **Minimize post-split RSS:** $\text{RSS}_{\text{split}} = \text{RSS}_L + \text{RSS}_R$
  - **Maximize reduction in RSS:** $\Delta\text{RSS} = \text{RSS}_{\text{parent}} - (\text{RSS}_L + \text{RSS}_R)$

# NAIVE SPLIT SELECTION: EXPLICIT COMPUT.



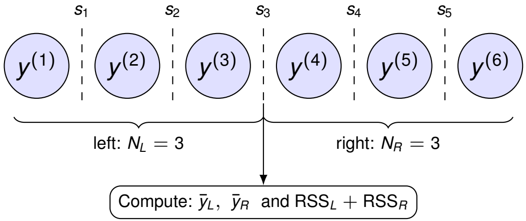
- For a given feature $X_j$, sort the pairs $(x_j^{(i)}, y^{(i)})$ by increasing $x_j^{(i)}$.
- For each of the $n-1$ potential split points at $s_k = \frac{1}{2}(x_j^{(k)} + x_j^{(k+1)})$:
  - Define partitions: $\mathcal{I}_L = \{i : x^{(i)} \leq s_k\}, \quad \mathcal{I}_R = \{i : x^{(i)} > s_k\}$
  - Compute group means and counts after splitting at $s_k$:

    $$\bar{y}_L = \frac{1}{N_L} \sum_{i \in \mathcal{I}_L} y^{(i)}, \quad \bar{y}_R = \frac{1}{N_R} \sum_{i \in \mathcal{I}_R} y^{(i)}, \text{ with } N_L = |\mathcal{I}_L|, \quad N_R = |\mathcal{I}_R|$$

  - Compute RSS after splitting at $s_k$:

    $$\text{RSS}_{\text{split}}(s_k) = \text{RSS}_L(s_k) + \text{RSS}_R(s_k) = \sum_{i \in \mathcal{I}_L} (y^{(i)} - \bar{y}_L)^2 + \sum_{i \in \mathcal{I}_R} (y^{(i)} - \bar{y}_R)^2$$

- Select split point $s_k$ that minimizes $\text{RSS}_{\text{split}}(s_k)$
- **Compute cost:** $O(n^2)$ per feat. (recompute mean & RSS at each split)



$\mathcal{O}(n^2)$ operations (recompute for each split $s_j$ per feature)

# EFFICIENT SPLIT SELECTION

- **Setup:** For feature $X_j$, sort the data $(x_j^{(i)}, y^{(i)})_{i=1}^n$ by increasing $x_j^{(i)}$
- **Define group statistics (cumulative sums) after split at $s_k$:**

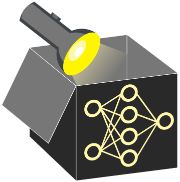
$$\begin{aligned} S_L &= \textstyle\sum_{i \in \mathcal{I}_L} y^{(i)}, \qquad & SS_L &= \textstyle\sum_{i \in \mathcal{I}_L} (y^{(i)})^2, \qquad & N_L &= |\mathcal{I}_L| \\ S_R &= S_n - S_L, \qquad & SS_R &= SS_n - SS_L, \qquad & N_R &= n - N_L \end{aligned}$$

- **RSS for child nodes and parent node:**

$$\text{RSS}_L(s_k) = SS_L - \frac{S_L^2}{N_L}, \text{RSS}_R(s_k) = SS_R - \frac{S_R^2}{N_R}, \text{RSS}_{\text{parent}} = SS_L + SS_R - \frac{S_n^2}{n}$$

HB

# EFFICIENT SPLIT SELECTION



- **Setup:** For feature $X_j$, sort the data $(x_j^{(i)}, y^{(i)})_{i=1}^n$ by increasing $x_j^{(i)}$
- **Define group statistics (cumulative sums) after split at $s_k$:**

$$S_L = \sum_{i \in \mathcal{I}_L} y^{(i)}, \qquad SS_L = \sum_{i \in \mathcal{I}_L} (y^{(i)})^2, \qquad N_L = |\mathcal{I}_L|$$
$$S_R = S_n - S_L, \qquad SS_R = SS_n - SS_L, \qquad N_R = n - N_L$$

- **RSS for child nodes and parent node:**

$$\text{RSS}_L(s_k) = SS_L - \frac{S_L^2}{N_L}, \text{RSS}_R(s_k) = SS_R - \frac{S_R^2}{N_R}, \text{RSS}_{\text{parent}} = SS_L + SS_R - \frac{S_n^2}{n}$$

- **Reduction in RSS:**

$$\Delta\text{RSS}(s_k) = \text{RSS}_{\text{parent}} - (\text{RSS}_L + \text{RSS}_R) = \frac{S_L^2}{N_L} + \frac{S_R^2}{N_R} - \frac{S_n^2}{n}$$

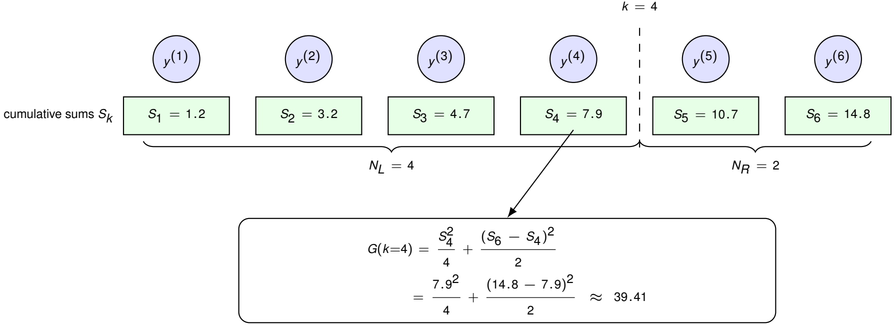
*All squared-target terms $SS_L$, $SS_R$ cancel. Only first-order sums needed.*

- **Search:** Choose best split $s_k^\star = \arg\max_{s_k} \Delta\text{RSS}(s_k)$
- **Complexity per feature:**
  $O(n \log n)$ (sorting) + $O(n)$ (cumulative sums and scan)

HB

# EFFICIENT SPLIT SELECTION - EXAMPLE

$$\boxed{y^{(1)}=1.2,\;\; y^{(2)}=2.0,\;\; y^{(3)}=1.5,\;\; y^{(4)}=3.2,\;\; y^{(5)}=2.8,\;\; y^{(6)}=4.1} \quad (\, x_j^{(1)} \leq \cdots \leq x_j^{(6)} \,)$$



$k = 4$

$y^{(1)}$, $y^{(2)}$, $y^{(3)}$, $y^{(4)}$, $y^{(5)}$, $y^{(6)}$

cumulative sums $S_k$: $S_1 = 1.2$, $S_2 = 3.2$, $S_3 = 4.7$, $S_4 = 7.9$, $S_5 = 10.7$, $S_6 = 14.8$

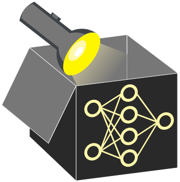
$N_L = 4$

$N_R = 2$

$$\begin{aligned} G(k{=}4) &= \frac{S_4^2}{4} + \frac{(S_6 - S_4)^2}{2} \\ &= \frac{7.9^2}{4} + \frac{(14.8 - 7.9)^2}{2} \approx 39.41 \end{aligned}$$

- $G(k)$ omits $-S_n^2/n$ (identical for all splits $\Rightarrow$ does not affect $\arg\max$).
- Only cumulative sums $S_k$ are required, no $SS_k$ is stored or updated.
- $\mathcal{O}(1)$ per split $\Rightarrow$ $\mathcal{O}(n)$ per feature.

# EXPLAINABLE BOOSTING MACHINES (EBM)



**Recall GAM**:

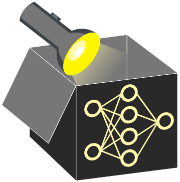
$$g\big(\mathbb{E}[y \mid \mathbf{x}]\big) = \theta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p),$$

- One shape function $f_j$ per feature $x_j$
  $\rightsquigarrow$ **Feature-level interpretability**
- Captures non-linear univariate effects
  $\rightsquigarrow$ **Better performance / more flexible than GLMs**

**EBM idea: GAMs** train with **gradient boosting** over **shallow bagged trees**

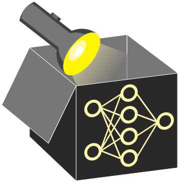
- **GAMs** - feature-wise interpretability via separate shape functions $f_j(x_j)$
  $\rightsquigarrow$ Potentially include pairwise interactions manually
- **Gradient Boosting** - incrementally fits residuals to improve predictive performance while retaining additivity
- **Shallow Bagged Trees** - low-depth trees (2-4 leaves) reduce variance and create interpretable shape functions

# EBM - TWO-STAGE MODEL CONSTRUCTION



1. **Stage 1: Fit Main Effects (Univariate Terms)** ▸ "Lou et al." 2012
   - Train EBM using only feature-wise shape functions $f_j(x_j)$
   - Freeze the univariate model after convergence
2. **Stage 2: Add Selected Pairwise Interactions** ▸ "Lou et al." 2013
   - Apply **FAST** to rank all $O(p^2)$ feat pairs by potential reduction in RSS
   - Select top $K$ pairwise interactions and store them in $\mathcal{K}$
   - Use boosting to fit pairwise interaction terms $f_{ij}(x_i, x_j)$ on residuals
   - Final model: $\hat{f}(\mathbf{x}) = \sum_{j=1}^{p} f_j(x_j) + \sum_{(i,j)\in\mathcal{K}} f_{ij}(x_i, x_j)$

# UNIVARIATE EBM - INITIALIZATION



- Set all shape functions to zero:

$$f_j^{[0]}(x_j) = 0 \quad \text{for all } j = 1, \ldots, p$$

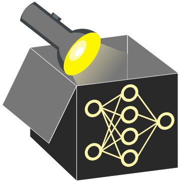
- Compute initial model prediction:

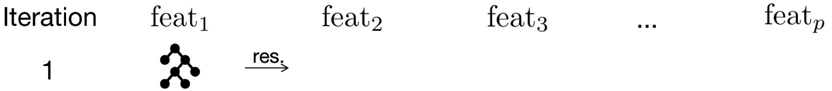
$$\hat{y}^{[0]} = \sum_{j=1}^{p} f_j^{[0]}(x_j) = 0$$

- Compute initial pseudo-residuals (e.g., for squared loss):

$$\tilde{r}^{[0]} = -\frac{\partial L}{\partial \hat{y}} = y - \hat{y}^{[0]} = y$$

# UNIVARIATE EBM - FIRST FEATURE UPDATE





Iteration $\text{feat}_1$ $\text{feat}_2$ $\text{feat}_3$ ... $\text{feat}_p$

1 $\xrightarrow{\text{res.}}$

- Fit shallow bagged tree $T_1^{[1]}$ (2-4 leaves) to training data $\left\{\left(x_1, \tilde{r}^{[0]}\right)^{(i)}\right\}_{i=1}^{n}$

  $\rightsquigarrow$ Use only feature $x_1$ as input and $\tilde{r}^{[0]}$ as target
- Update first shape function with learning rate $\eta$:

$$f_1^{[1]}(x_1) = f_1^{[0]}(x_1) + \eta \cdot T_1^{[1]}(x_1)$$

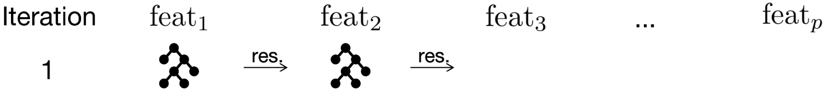
- Update prediction:

$$\hat{y}^{[1]} = \sum_{j=1}^{p} f_j^{[1]}(x_j)$$

- Recompute pseudo-residuals:

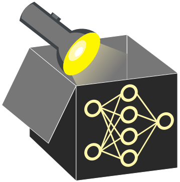
$$\tilde{r}^{[1]} = -\frac{\partial L}{\partial \hat{y}} = y - \hat{y}^{[1]}$$

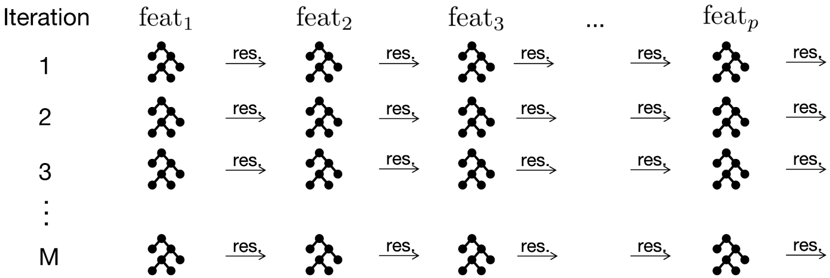
# UNIVARIATE EBM - CYCLE THROUGH FEATURES



- 1st boosting iteration:
  Cycle through each feature $j = 2, \ldots, p$:
  - Fit shallow bagged tree $T_j^{[1]}$ using feature $x_j$ and previous residual $\tilde{r}^{[j-1]}$
  - Update $f_j$: $f_j^{[1]}(x_j) = f_j^{[0]}(x_j) + \eta \cdot T_j^{[1]}(x_j)$
  - Recompute $\hat{y}$ and residuals: $\tilde{r}^{[j]} = y - \hat{y}^{[j]}$
- After one full pass over features, we complete one boosting iteration

# UNIVARIATE EBM - ITERATE BOOSTING PROCESS





- Repeat feature-wise updates for $M$ boosting iterations (e.g., $M = 10000$)
- In each boosting iteration:
  - Cycle over all features $j = 1, \dots, p$ individually
  - Update only one $f_j$ at a time using residuals from previous state
- Use small learning rate $\eta$ to ensure smooth updates and order-invariance

# UNIVARIATE EBM - PREDICTION & INTERPRETABILITY

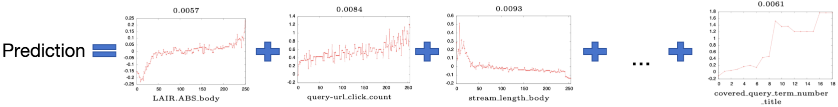
- Final model consists of $M$ shallow trees per feature:

$$\text{EBM Model} = \sum_{j=1}^{p} \sum_{m=1}^{M} \eta \cdot T_j^{[m]}(x_j)$$

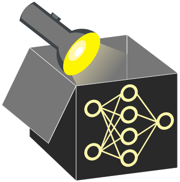
- For each feature $x_j$, combine its $M$ trees into a shape function:

$$\hat{f}_j(x_j) = \sum_{m=1}^{M} \eta \cdot T_j^{[m]}(x_j)$$

- Plot $\hat{f}_j(x_j)$ vs. $x_j$ $\rightsquigarrow$ Shows univariate marginal effect of feature $j$
- One plot per feature $\rightsquigarrow$ Model is fully explainable via $p$ additive plots



# EBM WITH PAIRWISE INTERACTIONS



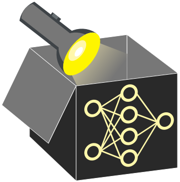
**Generalized Additive Models plus Interactions (GA2M):**

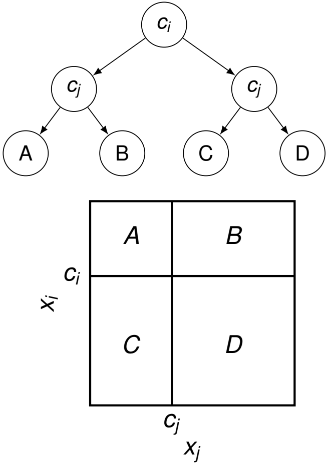
$$g\big(\mathbb{E}[y \mid \mathbf{x}]\big) = \theta_0 + \sum_{j=1}^{p} f_j(x_j) + \sum_{i<j} f_{ij}(x_i, x_j)$$

- **Motivation**: Univariate EBM does not model interactions
- **Challenge**: $O(p^2)$ potential pairwise interactions ⇝ often infeasible
- **Solution - FAST algorithm** ▸ “Lou et al.” 2013:
  - Efficiently estimates importance of all feature pairs
  - Ranks pairs by reduction in residual sum of squares (RSS)
  - Avoids fitting EBM with each pairwise interaction
- **Result**:
  Add only top-ranked interactions $f_{ij}$ via asecond-stage boosting step
  ⇝ Performed after the univariate EBM has been trained
- **Interpretability preserved**: Each $f_{ij}(x_i, x_j)$ visualized as a 2D heatmap

# FAST: PAIR-WISE INTERACTION STRENGTH

We evaluate a 4-leaf, axis-aligned tree $T_{ij}$ over the 2D feature projection $(x_i, x_j)$.



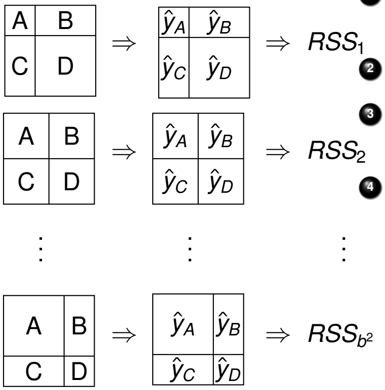


tree $T_{ij}$ with 4 leaves

1. **Discretize** : Map each axis to $b \leq 256$ ordered bins (quantile or equal-width).

# FAST: PAIR-WISE INTERACTION STRENGTH

We evaluate a 4-leaf, axis-aligned tree $T_{ij}$ over the 2D feature projection $(x_i, x_j)$.



A B C D ⇒ $\hat{y}_A$ $\hat{y}_B$ $\hat{y}_C$ $\hat{y}_D$ ⇒ $RSS_1$
A B C D ⇒ $\hat{y}_A$ $\hat{y}_B$ $\hat{y}_C$ $\hat{y}_D$ ⇒ $RSS_2$
⋮ ⋮ ⋮
A B C D ⇒ $\hat{y}_A$ $\hat{y}_B$ $\hat{y}_C$ $\hat{y}_D$ ⇒ $RSS_{b^2}$

1. **Discretize** : Map each axis to $b \leq 256$ ordered bins (quantile or equal-width).
2. **Iterate** over $b^2$ candidate cuts $(c_i, c_j)$.
3. **Fit** : For each cut, assign a constant $\hat{y}_r = \text{mean}(y \in r)$ to $r \in \{A, B, C, D\}$.
4. **Compute RSS summed over all regions**:

$$\text{RSS}(c_i, c_j) = \sum_r \sum_{(x,y)\in r} (y - \hat{y}_r)^2$$

$$= \sum_r \left( \sum_{(x,y)\in r} y^2 - \frac{1}{n_r} \left( \sum_{(x,y)\in r} y \right)^2 \right)$$

5. **Select** : Keep the split with minimal RSS.
⇝ largest RSS drop = strongest interaction.

# FAST: USE RSS DROP

To evaluate a cut pair $(c_i, c_j)$, we use precomputed per-region statistics:

- For each region $r \in \{A, B, C, D\}$, compute:

$$S_r = \sum_{(x,y)\in r} y, \quad n_r = |\{(x, y) \in r\}|, \quad \hat{y}_r = S_r / n_r$$

- Plug into RSS summed over all regions:

$$\text{RSS}(c_i, c_j) = \sum_r \left( \sum_{(x,y)\in r} y^2 - \frac{1}{n_r} \left( \sum_{(x,y)\in r} y \right)^2 \right) = \sum_r \sum_{(x,y)\in r} y^2 + \sum_r \frac{S_r^2}{n_r}$$

- For a candidate cut, compute **RSS drop**:

$$\begin{aligned} \Delta\text{RSS}(c_i, c_j) &= \text{RSS}_{\text{parent}} - \text{RSS}(c_i, c_j) \\ &= \left( \sum_{i=1}^{n} \left( y^{(i)} \right)^2 - \frac{S_n^2}{n} \right) - \sum_r \sum_{(x,y)\in r} y^2 + \sum_r \frac{S_r^2}{n_r} \end{aligned}$$

# FAST: USE RSS DROP

Because $\sum_{i=1}^{n} \left(y^{(i)}\right)^2 = \sum_r \sum_{(x,y)\in r} y^2$, *all squared target terms cancel*:

$$\boxed{\Delta \text{RSS}(c_i, c_j) = \sum_r \frac{S_r^2}{n_r} - \frac{S_n^2}{n}}$$

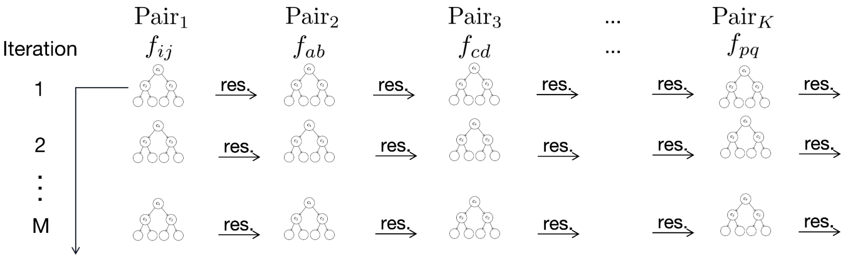
The parent term $S_n^2/n$ is constant across all cuts. Hence

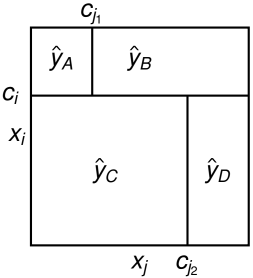
$$\textbf{maximize } \Delta \text{RSS}(c_i, c_j) = \sum_r \frac{S_r^2}{n_r} \iff \textbf{minimize } \text{RSS}(c_i, c_j).$$

**Why is this efficient?**

- Precompute cummulative sums of $y$ and counts across the binned grid
- Enables fast lookup of region statistics $S_r, n_r$ for any cut
- No additional data scan or recomputation needed across the $b^2$ candidate cuts
- For the best cut: Compare and select the largest $\Delta \text{RSS}(c_i, c_j)$.

# EBM - BOOSTING PAIRWISE INTERACTIONS

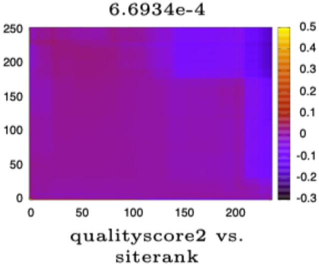




- **Goal:** Fit each selected interaction $f_{ij}(x_i, x_j)$ on residuals from main effects
- Use tree-like predictor, inspired by FAST
  - Use two axis-aligned cuts $(c_i, c_j)$
  - Plus one refinement cut to increase flexibility while keeping interpretability
- Reuse region-wise sums from FAST lookup tables
- Greedy search for cut config minimizing RSS

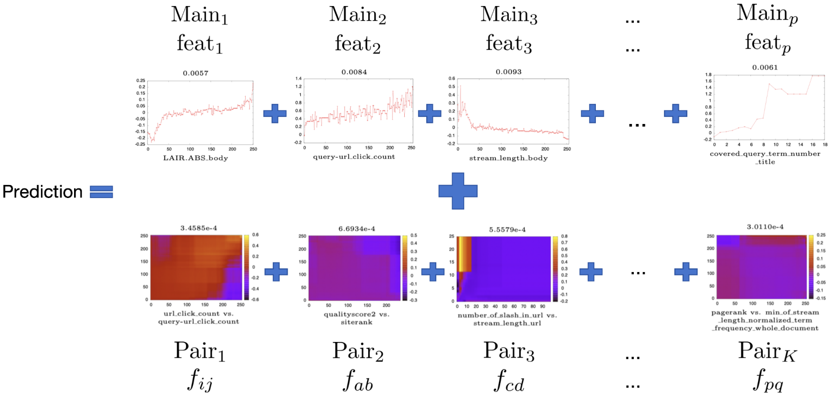
# EBM - PREDICTION WITH PAIRWISE INTERACTIONS

- Each selected pair $(x_i, x_j)$ is modeled by $M$ boosted predictors trained on their residual interaction
- These are aggregated into a single bivariate function $f_{ij}(x_i, x_j)$
- The function is visualized as a 2D heatmap:
  - Axes: feature values of $x_i$ and $x_j$
  - Color: contribution to the final prediction
  - Preserves human interpretability
- One heatmap is generated per selected pairwise interaction



# EBM - FINAL MODEL STRUCTURE

- **Main effects:** One shape function $f_j(x_j)$ per feature (visualized as 1D plots)
- **Pairwise interactions:** Selected functions $f_{ij}(x_i, x_j)$ added for top $K$ pairs (visualized as 2D heatmaps)
- **Prediction:** Additive sum of all univariate and selected bivariate contributions



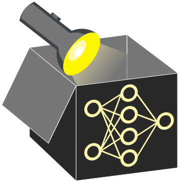
$\text{Main}_1$ $\text{feat}_1$ $\text{Main}_2$ $\text{feat}_2$ $\text{Main}_3$ $\text{feat}_3$ ... ... $\text{Main}_p$ $\text{feat}_p$

Prediction =

$\text{Pair}_1$ $f_{ij}$ $\text{Pair}_2$ $f_{ab}$ $\text{Pair}_3$ $f_{cd}$ ... ... $\text{Pair}_K$ $f_{pq}$

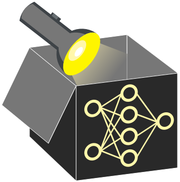
0.0057
LAIR.ABS_body
0.0084
query-url_click_count
0.0093
stream_length_body
...
0.0061
covered_query_term_number _title
3.4585e-4
url_click_count vs. query-url_click_count
6.6934e-4
qualityscore2 vs. siterank
5.5579e-4
number_of_slash_in_url vs. stream_length_url
...
3.0110e-4
pagerank vs. min_of_stream _length_normalized_term _frequency_whole_document

# EBM *VS.* MODEL-BASED BOOSTING



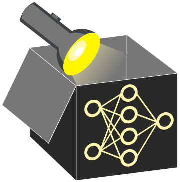
- **Base learner**
  - **EBM**: bagged 2–4-leaf trees, *one feature* per tree ⇒ step-function shape $f_j$ ▸ “Lou et al.” 2012
  - **MB-boost**: user chooses component-wise learner (linear term, P-spline, tree, random effect, . . . ) ▸ “Bühlmann & Hothorn” 2007

# EBM *VS.* MODEL-BASED BOOSTING



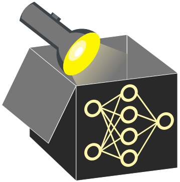
- **Base learner**
  - **EBM**: bagged 2–4-leaf trees, *one feature* per tree ⇒ step-function shape $f_j$ ▸ “Lou et al.” 2012
  - **MB-boost**: user chooses component-wise learner (linear term, P-spline, tree, random effect, . . . ) ▸ “Bühlmann & Hothorn” 2007
- **Iteration policy**
  - **EBM**: round-robin ($\forall j$) each boosting pass; tiny learning rate $\eta \approx 0.01$.
  - **MB-boost**: greedy; update the *single* component that yields the largest loss reduction.

# EBM *VS.* MODEL-BASED BOOSTING



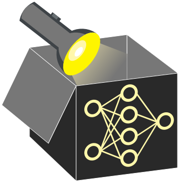
- **Base learner**
  - **EBM**: bagged 2–4-leaf trees, *one feature* per tree ⇒ step-function shape $f_j$ ▸ "Lou et al." 2012
  - **MB-boost**: user chooses component-wise learner (linear term, P-spline, tree, random effect, . . . ) ▸ "Bühlmann & Hothorn" 2007
- **Iteration policy**
  - **EBM**: round-robin ($\forall j$) each boosting pass; tiny learning rate $\eta \approx 0.01$.
  - **MB-boost**: greedy; update the *single* component that yields the largest loss reduction.
- **Regularisation**
  - **EBM**: many iterations $M$ (5–10k); early stopping via *internal* CV on out-of-bag samples; bagging further lowers variance.
  - **MB-boost**: shrinkage $\nu \in (0, 1]$; early stop by CV/AIC; component selection acts like an $L_0/L_1$ penalty → sparsity.

# EBM *VS.* MODEL-BASED BOOSTING



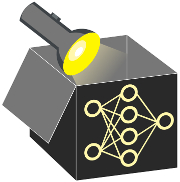
- **Interactions**
  - **EBM**: FAST ranks and selects top-$K$ interaction pairs, fitted as bivariate trees ⇒ GA2M ▸ "Lou et al." 2013
  - **MB-boost**: interactions are modelled only when the user supplies dedicated interaction base learners; no automatic pairwise search

# EBM *VS.* MODEL-BASED BOOSTING



- **Interactions**
  - **EBM**: FAST ranks and selects top-$K$ interaction pairs, fitted as bivariate trees $\Rightarrow$ GA2M ▸ “Lou et al.” 2013
  - **MB-boost**: interactions are modelled only when the user supplies dedicated interaction base learners; no automatic pairwise search
- **Interpretability**
  - **EBM**:
    - one 1-D step plot for each $f_j$
    - small number of 2-D heat-maps for selected $f_{ij}$
  - **MB-boost**: depends on selected learner: linear coefficients, smooth splines, random-effect curves, etc.

# EBM *VS.* MODEL-BASED BOOSTING



- **Interactions**
  - **EBM**: FAST ranks and selects top-$K$ interaction pairs, fitted as bivariate trees $\Rightarrow$ GA2M ▸ "Lou et al." 2013
  - **MB-boost**: interactions are modelled only when the user supplies dedicated interaction base learners; no automatic pairwise search
- **Interpretability**
  - **EBM**:
    - one 1-D step plot for each $f_j$
    - small number of 2-D heat-maps for selected $f_{ij}$
  - **MB-boost**: depends on selected learner: linear coefficients, smooth splines, random-effect curves, etc.
- **Take-away**
  - *EBM* provides fast, interpretable, and interaction-sparse models
  - *MB-boost* offers flexible stat modeling with built-in variable selection