Gradient Boosted Decision Trees

Dr. Daniele Cattaneo, Prof. Dr. Josif Grabocka

Machine Learning Course
Winter Semester 2023/2024

Albert-Ludwigs-Universität Freiburg

cattaneo@informatik.uni-freiburg.de, grabocka@informatik.uni-freiburg.de

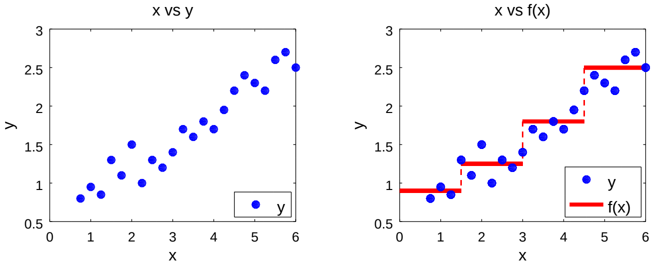
11.12.2023

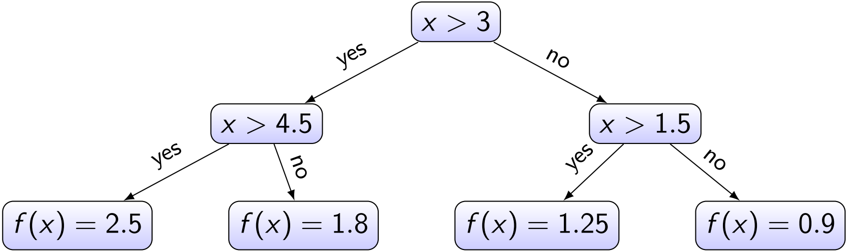
Prediction Model of a Decision Tree

- A tree having T leaves outputs the weights $w \in \mathbb{R}^T$.
- Let $q : \mathbb{R}^M \to \{1, \ldots, T\}$ denote the leaf index $q(x_n)$ where instance x_n belongs to, then
- The prediction model of a tree is:

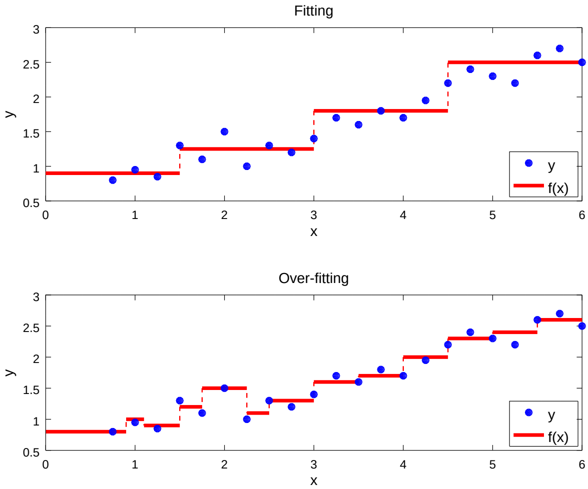
$$f(x_n) = w_{q(x_n)}$$

Decision Tree as a Step-wise Function





Tree Over-fitting



Tree over-fits if too many steps (nodes) and high jumps (large leaf weights)

Tree Regularization

- *Note:* Too many steps $\approx$ Too many leaves (T)
- *Note:* Too large step jumps $\approx$ Too large leaves' output values (w)
- Penalize the number of leaves and leaves' weights, e.g.:

$$\Omega(f) = \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

Boosting

- Weak learners (single trees) are combined to create more expressive models

- Unite single trees into an ensemble of k trees
- The estimation is aggregated over the individual trees' predictions:

$$
\begin{aligned}
\hat{y}_n^{(1)} &:= f^{(1)}(x_n), \;\; \hat{y}_n^{(2)} := \hat{y}_n^{(1)} + f^{(2)}(x_n), \;\ldots \\
\hat{y}_n^{(k)} &:= \hat{y}_n^{(k-1)} + f^{(k)}(x_n) = \sum_{l=1}^{k} f^{(l)}(x_n)
\end{aligned}
$$

Boosted Ensemble Loss

- Add one tree at a time to the ensemble (greedy strategy)
- The loss created as a result of adding the contribution of the k-th tree is:

$$
\begin{aligned}
& \operatorname*{argmin}_{f^{(k)}} \left[\sum_{n=1}^{N} \mathcal{L}^{(k)}(y_n, \hat{y}_n^{(k-1)} + f^{(k)}(x_n)) \right] + \Omega(f^{(k)}) \\
:= \quad & \operatorname*{argmin}_{f^{(k)}} \left[\sum_{n=1}^{N} \mathcal{L}_n^{(k)} \right] + \Omega(f^{(k)})
\end{aligned}
$$

- How to find the optimal k-th tree $f^{(k)}$?

Strategy

- Given the split rules what are the optimal leaves w?
- How to split the tree into further leaves?

Tailor Approximation

Remember Tailor Expansion (2nd degree):

$$F(x + \Delta x) \approx F(x) + \frac{dF(x)}{dx}\Delta x + \frac{1}{2}\frac{d^2F(x)}{dx^2}\Delta x^2$$

While our case:

$$\mathcal{L}_n^{(k)} = \mathcal{L}^{(k)}(y_n, \hat{y}_n^{(k-1)} + f^{(k)}(x_n))$$

Where $\mathcal{L}_n^{(k)}(y_n, \hat{y}_n^{(k-1)} + f^{(k)}(x_n))$ is equivalent to $F(x + \Delta x)$ with:

$F := \mathcal{L}_n^{(k-1)}$, $x := \hat{y}_n^{(k-1)}$ and $\Delta x := f^{(k)}(x_n)$

Tailor Approximation (cont.)

$$F(x+\Delta x) \approx F(x) + \frac{dF(x)}{dx}\Delta x + \frac{1}{2}\frac{d^2F(x)}{dx^2}\Delta x^2$$

Leads to:

$$\begin{aligned}
\mathcal{L}_n^{(k)} &\approx \mathcal{L}_n^{(k-1)} + \frac{\partial \mathcal{L}_n^{(k-1)}}{\partial \hat{y}_n^{(k-1)}} f^{(k)}(x_n) + \frac{1}{2}\frac{\partial^2 \mathcal{L}_n^{(k-1)}}{\partial \left(\hat{y}_n^{(k-1)}\right)^2}\left(f^{(k)}(x_n)\right)^2 \\
\mathcal{L}_n^{(k)} &\approx \mathcal{L}_n^{(k-1)} + G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n)\right)^2
\end{aligned}$$

$$\text{where } G_n := \frac{\partial \mathcal{L}_n^{(k-1)}}{\partial \hat{y}_n^{(k-1)}}, \quad H_n := \frac{\partial^2 \mathcal{L}_n^{(k-1)}}{\partial \left(\hat{y}_n^{(k-1)}\right)^2}$$

Learning Objective

Applying the Taylor expansion of the loss:

$$
\begin{aligned}
&\underset{f^{(k)}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} \mathcal{L}_n^{(k)}\right] + \Omega(f^{(k)}) \\
&\approx \underset{f^{(k)}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} \mathcal{L}_n^{(k-1)} + G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n)\right)^2\right] + \Omega(f^{(k)})
\end{aligned}
$$

Since $\mathcal{L}_n^{(k-1)}$ is constant w.r.t. $f^{(k)}$, then rewrite the objective as:

$$
\underset{f^{(k)}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n)\right)^2\right] + \Omega(f^{(k)})
$$

Learning Objective (cont.)

The objective is:

$$\underset{f^{(k)}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n) \right)^2 \right] + \Omega(f^{(k)})$$

where the regularization term:

$$\Omega(f^{(k)}) = \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

We need to express the objective in terms of w.

Learning Objective (cont.)

$$\underset{f^{(k)}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n) \right)^2 \right] + \Omega(f^{(k)})$$

- Remember $f^{(k)}(x) := w_{q(x)}$
- The ultimate objective is:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{n=1}^{N} \left[G_n w_{q(x_n)} + \frac{1}{2} H_n w_{q(x_n)}^2 \right] + \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

Rewrite objective in terms of leaves

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{n=1}^{N} \left[G_n w_{q(x_n)} + \frac{1}{2} H_n w_{q(x_n)}^2 \right] + \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

- Let indices of all instances belonging into the j-th leaf be $I_j := \{n \mid q(x_n) = j\}$.
- We can rewrite the objective as:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{j=1}^{T} \left[\left(\sum_{n \in I_j} G_n \right) w_j + \frac{1}{2} \left(\lambda + \sum_{n \in I_j} H_n \right) w_j^2 \right] + \gamma T$$

Optimal Tree Leaves

- Given the objective:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{j=1}^{T} \left[\left(\sum_{n \in I_j} G_n \right) w_j + \frac{1}{2} \left(\lambda + \sum_{n \in I_j} H_n \right) w_j^2 \right] + \gamma T$$

- Denote $A = \sum_{n \in I_j} G_n$ and $B = \lambda + \sum_{n \in I_j} H_n$
- Optimal leaves can be computed in a closed-form solution:

$$w^{(\mathrm{opt})} = \underset{w}{\operatorname{argmin}} \, Aw + \frac{1}{2} B w^2 = -\frac{A}{B}$$

- The optimal leaf weights w are:

$$w_j = -\frac{\sum\limits_{n \in I_j} G_n}{\lambda + \sum\limits_{n \in I_j} H_n}, \quad j = 1, \ldots, T$$

Optimal Objective Function

- Given the objective:

$$\underset{w_1,\dots,w_T}{\operatorname{argmin}} \quad \sum_{j=1}^{T}\left[\left(\sum_{n\in I_j} G_n\right) w_j + \tfrac{1}{2}\left(\lambda + \sum_{n\in I_j} H_n\right) w_j^2\right] + \gamma T$$

- Knowing that $w^{(\text{opt})} = -\frac{A}{B}$:

$$\min_w Aw + \frac{1}{2}Bw^2 = Aw^{(\text{opt})} + \frac{1}{2}B{w^{(\text{opt})}}^2 = -\frac{A^2}{2B}$$

- The optimal objective function is:

$$\mathcal{O}(G,H) := -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum\limits_{n\in I_j} G_n\right)^2}{\left(\lambda+\sum\limits_{n\in I_j} H_n\right)} + \gamma T$$

How to grow trees?

- Decompose $\mathcal{O}(G, H) := \sum_{j=1}^{T} \mathcal{O}_j$ as per-leaf objectives:

$$\mathcal{O}_j := -\frac{1}{2} \frac{\left(\sum_{n \in I_j} G_n\right)^2}{\left(\lambda + \sum_{n \in I_j} H_n\right)} + \gamma$$

- When splitting leaf j after a decision split we yield two sub-leaves $j^{(\text{Left})}$ and $j^{(\text{Right})}$
- The gain in minimizing the objective after splitting leaf j:

$$\text{Gain}_j := \mathcal{O}_j - \left(\mathcal{O}_{j^{(\text{Left})}} + \mathcal{O}_{j^{(\text{Right})}}\right)$$

Gain of splitting a leaf

- Given:

$$\mathcal{O}_j := -\frac{1}{2}\frac{\left(\sum_{n\in I_j} G_n\right)^2}{\left(\lambda+\sum_{n\in I_j} H_n\right)}+\gamma, \quad \text{Gain}_j := \mathcal{O}_j - \left(\mathcal{O}_{j(\text{Left})} + \mathcal{O}_{j(\text{Right})}\right)$$

- Derive:

$$\text{Gain}_j := \frac{1}{2}\left[\underbrace{\frac{\left(\sum_{n\in I_j^{(\text{Left})}} G_n\right)^2}{\left(\lambda+\sum_{n\in I_j^{(\text{Left})}} H_n\right)}}_{\text{Objective of left child}} + \underbrace{\frac{\left(\sum_{n\in I_j^{(\text{Right})}} G_n\right)^2}{\left(\lambda+\sum_{n\in I_j^{(\text{Right})}} H_n\right)}}_{\text{Objective of right child}} - \underbrace{\frac{\left(\sum_{n\in I_j} G_n\right)^2}{\left(\lambda+\sum_{n\in I_j} H_n\right)}}_{\text{Objective of parent}}\right] - \underbrace{\gamma}_{\substack{\text{Regularize}\\\text{additional}\\\text{leaf}}}$$

Stopping Condition

$$\text{Gain}_j := \frac{1}{2}\left[\frac{\left(\sum_{n\in I_j^{(\text{Left})}} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j^{(\text{Left})}} H_n\right)} + \frac{\left(\sum_{n\in I_j^{(\text{Right})}} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j^{(\text{Right})}} H_n\right)} - \frac{\left(\sum_{n\in I_j} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j} H_n\right)}\right] - \gamma$$

We can decide to not split further if $\text{Gain}_j \leq 0$, i.e. stop if:

$$\frac{\left(\sum_{n\in I_j^{(\text{Left})}} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j^{(\text{Left})}} H_n\right)} + \frac{\left(\sum_{n\in I_j^{(\text{Right})}} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j^{(\text{Right})}} H_n\right)} - \frac{\left(\sum_{n\in I_j} G_n\right)^2}{\left(\lambda + \sum_{n\in I_j} H_n\right)} \leq \gamma$$

Notice that γ is a hyper-parameter that controls the minimum split gain.

Split rule search

- For each node, exhaustively visit all splitting rules:
  - For each feature $m = 1, \ldots, M$ of the data $X \in \mathbb{R}^{N \times M}$
    - Sort the instances $n = 1, \ldots, N$ of the m-th feature $x_{:,m} \in \mathbb{N}$
    - Denote the unique sorted values $\mathcal{V}_m \in \mathbb{R}^{N'}$, where $N' \leq N$
    - Generate all split rules:

$$\left[x_{:,m}; \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2}\right], \text{ for } n' = 1, \ldots, N' - 1$$

- Select the split rule that maximizes the gain

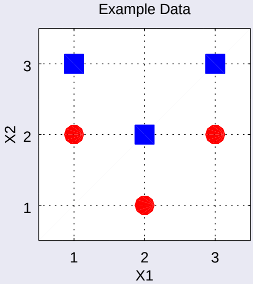
$$\underset{\substack{\left[x_{:,m}; \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2}\right] \\ \forall m \in \{1,\ldots,M\} \\ \forall n' \in \{1,\ldots,|\mathcal{V}_{m,:}|-1\}}}{\operatorname{argmax}} \quad \mathcal{O}_j - \left(\mathcal{O}_{j^{(\text{Left})}} + \mathcal{O}_{j^{(\text{Right})}}\right)$$

where

$$I_j^{(\text{Left})} = \left\{ n \mid x_{n,m} \leq \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2} \right\}$$

$$I_j^{(\text{Right})} = \left\{ n \mid x_{n,m} > \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2} \right\}$$

Exercise



n	x_1	x_2	y
1	1	2	0
2	2	1	0
3	3	2	0
4	1	3	1
5	2	2	1
6	3	3	1

- Learn an ensemble of 2 trees to estimate:
  - Limit maximum depth of trees to two.
  - Use logistic loss
  - Set $\gamma = 1$, $\lambda = 1$.
  - Ignore the stopping criterion on gain for this exercise.

Exercise - Step 1: Gradients and Hessians

- Before building each tree compute the gradients and Hessians:

$$\begin{aligned}
\mathcal{L}_n &= -y_n \log(\sigma(\hat{y}_n)) - (1 - y_n) \log(1 - \sigma(\hat{y}_n)) \\
G_n &= \frac{\partial \mathcal{L}_n}{\partial \hat{y}_n} = \sigma(\hat{y}_n) - y_n \\
H_n &= \frac{\partial^2 \mathcal{L}_n}{\partial (\hat{y}_n)^2} = \frac{\partial G_n}{\partial \hat{y}_n} = \sigma(\hat{y}_n)(1 - \sigma(\hat{y}_n))
\end{aligned}$$

- Remember the prediction model of a boosted ensemble:

$$\hat{y}_n^{(k)} = \hat{y}_n^{(k-1)} + f^{(k)}(x_n)$$

- For the first tree, assume $\hat{y}_n^{(0)} = 0$, yielding

$$\hat{y}_n^{(1)} = f^{(1)}(x_n)$$

Exercise - Step 1: Gradients and Hessians (II)

- Knowing $\sigma(\hat{y}_n) = \left(1 + e^{-\hat{y}_n}\right)^{-1}, \quad G_n = \sigma(\hat{y}_n) - y_n, \quad H_n = \sigma(\hat{y}_n)(1 - \sigma(\hat{y}_n))$
- Compute once before growing each tree:

n	X_1	X_2	y	$\hat{y}^{(0)}$	$\sigma(\hat{y}^{(0)})$	G	H
1	1	2	0	0	0.5	0.5	0.25
2	2	1	0	0	0.5	0.5	0.25
3	3	2	0	0	0.5	0.5	0.25
4	1	3	1	0	0.5	-0.5	0.25
5	2	2	1	0	0.5	-0.5	0.25
6	3	3	1	0	0.5	-0.5	0.25

Exercise - Step 2: Enumerate split rules

- For first feature $m = 1$
  - Unique sorted values $\mathcal{V}_1 = \{1, 2, 3\}$
  - Rules $[x_{:,1}; 1.5]$ and $[x_{:,1}; 2.5]$
- For second feature $m = 2$:
  - Unique sorted values $\mathcal{V}_2 = \{1, 2, 3\}$
  - Rules $[x_{:,2}; 1.5]$ and $[x_{:,2}; 2.5]$

- In the beginning there is only the root $j = 1$, where:
  - All instances belong to the root: $I_1 = \{1, 2, 3, 4, 5, 6\}$
- Which rule $[x_{:,1}; 1.5]\,, [x_{:,1}; 2.5]\,, [x_{:,2}; 1.5]\,, [x_{:,2}; 2.5]$ maximizes the gain of splitting the root?

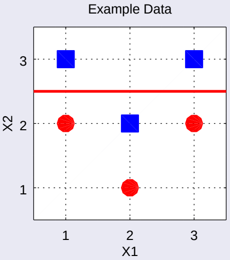
Exercise - Step 3: Best split rule = Maximal Gain

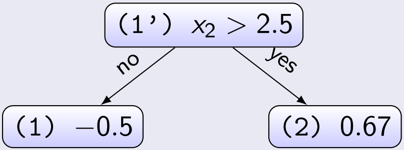
n	X_1	X_2	y	$\hat{y}^{(0)}$	$\sigma(\hat{y}^{(0)})$	G	H
1	1	2	0	0	0.5	0.5	0.25
2	2	1	0	0	0.5	0.5	0.25
3	3	2	0	0	0.5	0.5	0.25
4	1	3	1	0	0.5	-0.5	0.25
5	2	2	1	0	0.5	-0.5	0.25
6	3	3	1	0	0.5	-0.5	0.25

- Rule $[x_{:,1}; 1.5]$:
  - $I_1^{(\text{Left})} = \{1, 4\}$ and $I_1^{(\text{Right})} = \{2, 3, 5, 6\}$, thus $Gain_1 = -1$
- Rule $[x_{:,1}; 2.5]$:
  - $I_1^{(\text{Left})} = \{1, 2, 4, 5\}$ and $I_1^{(\text{Right})} = \{3, 6\}$, thus $Gain_1 = -1$
- Rule $[x_{:,2}; 1.5]$:
  - $I_1^{(\text{Left})} = \{2\}$ and $I_1^{(\text{Right})} = \{1, 3, 4, 5, 6\}$, thus $Gain_1 = -0.84$
- Rule $[x_{:,2}; 2.5]$:
  - $I_1^{(\text{Left})} = \{1, 2, 3, 5\}$ and $I_1^{(\text{Right})} = \{4, 6\}$, thus $Gain_1 = -0.41$ (best)

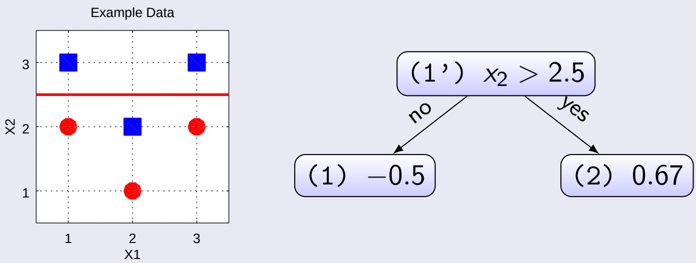
Our first tree with depth 1!

- The best rule we found $[x_{:,2}; 2.5]$:
  - Splits node ($j = 1'$) into $I_{1'}^{(\text{Left})} = \{1, 2, 3, 5\}$, $I_{1'}^{(\text{Right})} = \{4, 6\}$
  - Left child ($j = 1$) with weight $w_1 = -\frac{G_1+G_2+G_3+G_5}{H_1+H_2+H_3+H_5+\lambda} = -0.5$
  - Right child ($j = 2$) with weight $w_2 = -\frac{G_4+G_6}{H_4+H_6+\lambda} = 0.66$





Our first tree with depth 1! (cont.)



- Interpretation of the outcome $y_n^{(1)} = f^{(1)}(x_n) = w_{q(x_n)}$:
  - $\sigma(\hat{y}_n^{(1)}) = \sigma(-0.5) = 0.37, \ \forall n \in \{1, 2, 3, 5\}, \ q(x_n) = 1$
  - $\sigma(\hat{y}_n^{(1)}) = \sigma(0.67) = 0.66, \ \forall n \in \{4, 6\}, \ q(x_n) = 2$

Grow the tree further

- Follow the same procedure to compute the best rules for further splitting the nodes 1 and 2
- Proceed until the maximum allowed depth is reached.
- For subsequent trees in the ensemble follow the same procedure, but note that:
  - For the first tree $\hat{y}_n^{(0)} = 0$
  - For the second tree $\hat{y}_n^{(1)} = f^{(1)}(x_n)$
  - For the third tree $\hat{y}_n^{(2)} = f^{(1)}(x_n) + f^{(2)}(x_n)$, etc ...
- Finish the exercise at home!

Algorithmic Complexity

- Compute sorted unique feature values $O(MN\log(N))$
- There are $O(MN)$ many split rules in a dataset
- Computing the gradients and Hessians for each tree is $O(N)$

- The gain of one split rule at a node with N instances is $O(N)$
- The gain of all splits at a node naively is $O(MN^2)$
  - Can be incrementally computed in $O(MN)$

- In a balanced tree all the splits at each level are $O(MN)$
- For a tree having depth $O(\log(T))$ the tree is computed in $O(MN\log(T))$
- An ensemble with k trees is $O(kMN\log(T))$

Advantages of Gradient Boosted Decision Trees

- Work out of the box, no need for data preprocessing
- Work well with categorical features
- Ability to work with arbitrary loss functions

- Very low-bias, yet relatively low-variance
- Very fast algorithm $O(kMN \log(T))$

- XGBoost platform won numerous data science competitions
- Off-the-shelf tool for tabular data