Lecture 2
Boosting
Feature Importance estimation

Vladislav Goncharenko

Moscow, 2021

Outline

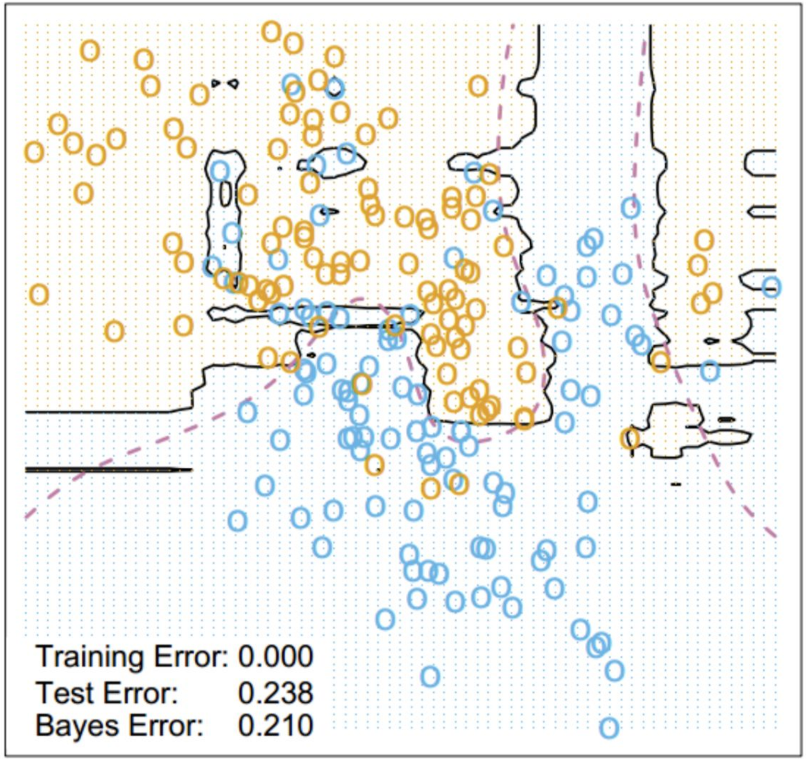
1. Ensembling methods recap
2. Boosting intuition
3. Gradient boosting
4. Feature importance estimation
5. Shap values

Random Forest

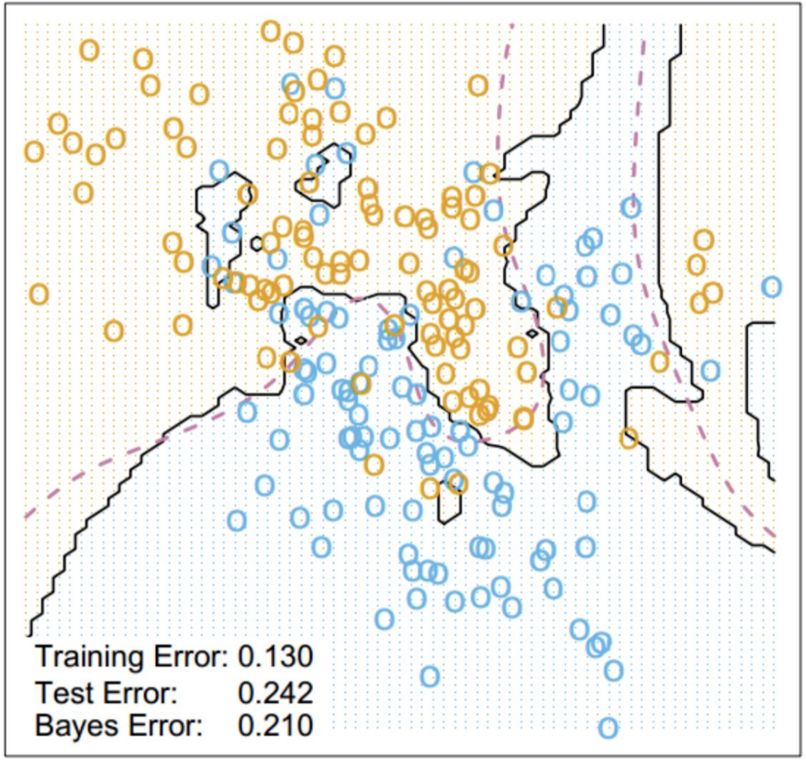
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

$$\mathrm{OOB} = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right)$$

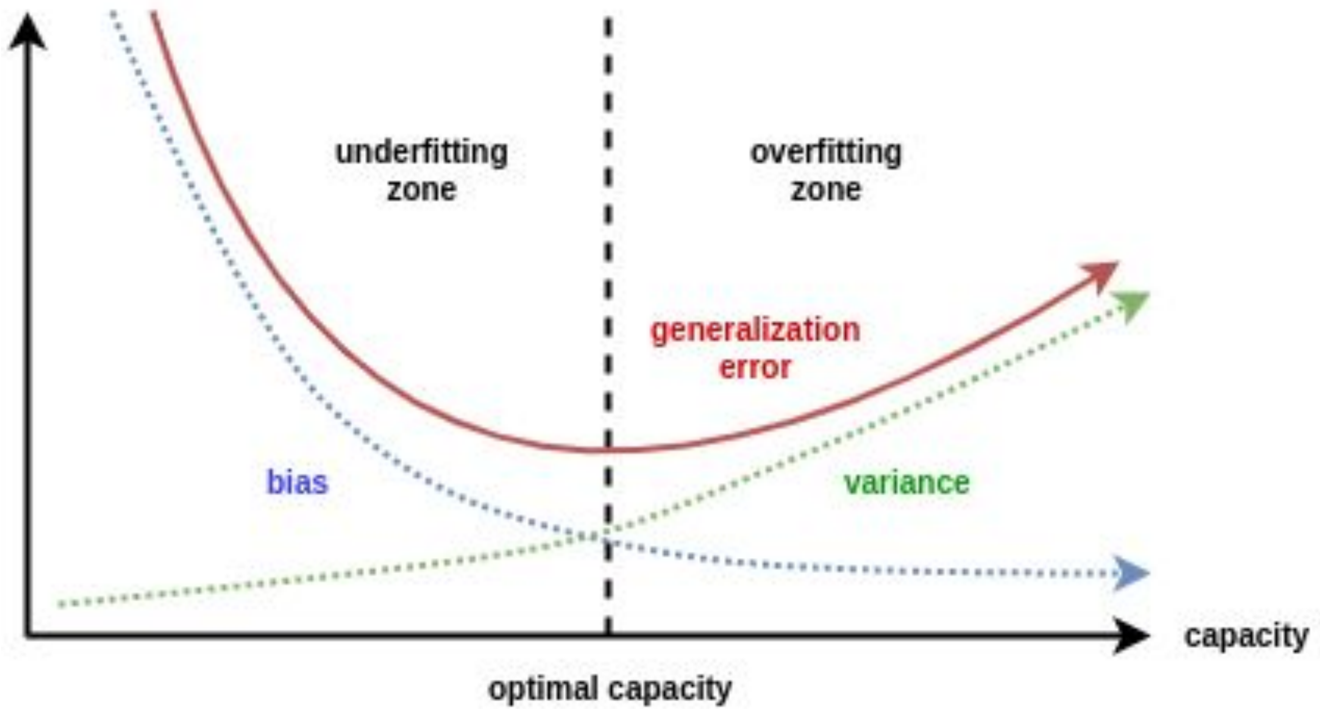
Random Forest Classifier



3−Nearest Neighbors

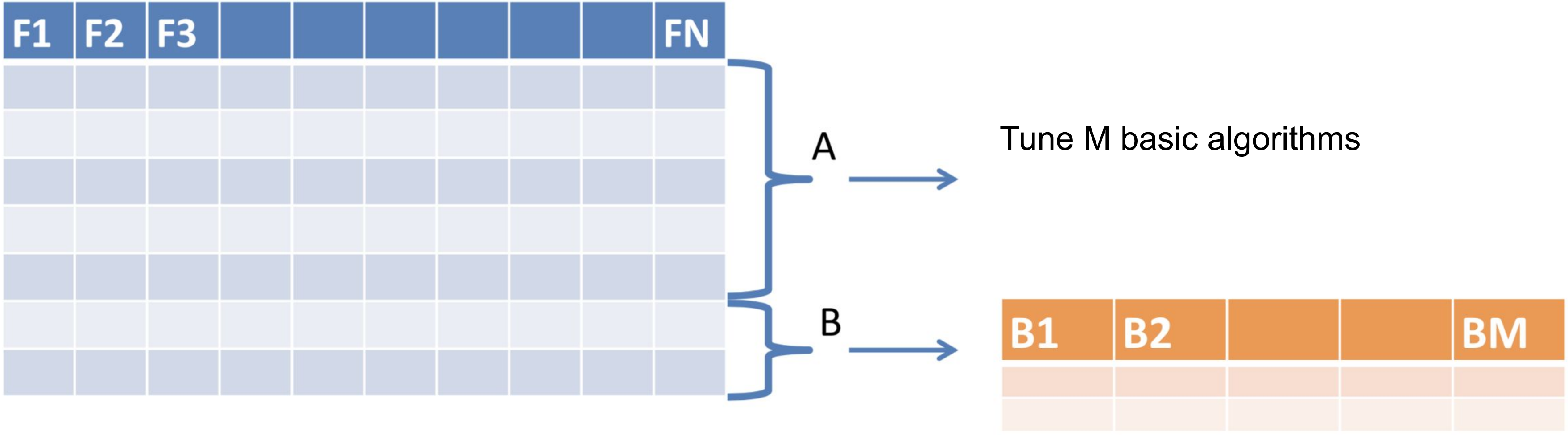


Bias-variance tradeoff



Stacking

How to build an ensemble from *different* models?



$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

e.g.

Stacking

How to build an ensemble from *different* models?

- Use different datasets (or datasets parts) for different level models.
- Experiment with different models (linear, trees ensembles, simple networks, etc.)
- Or just different GBT ensembles (hola, kaggle :)

Blending

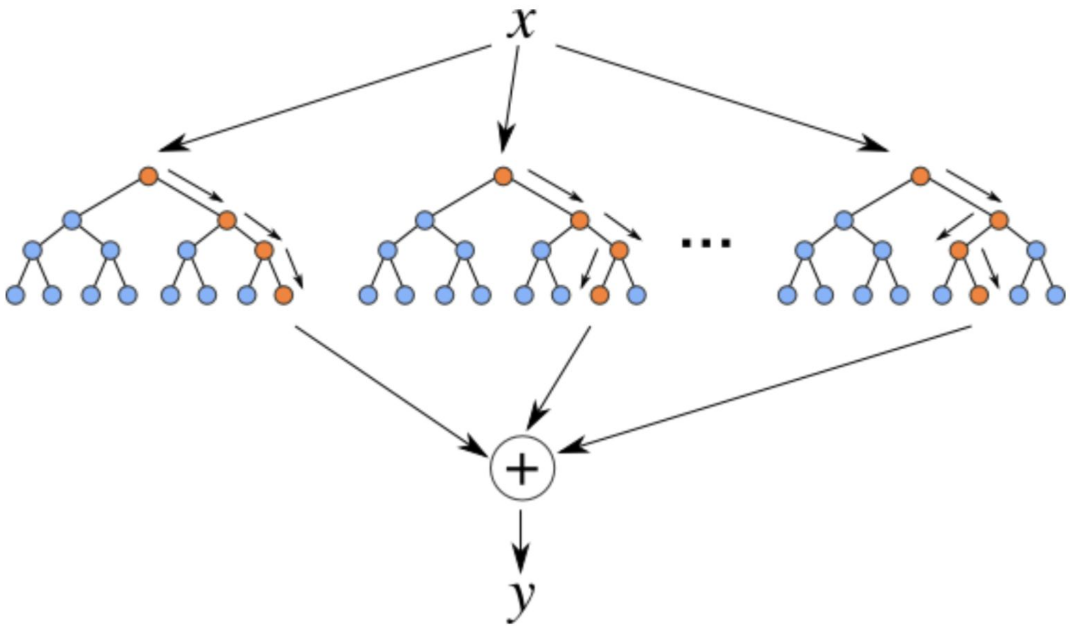
Just combine several *strong/complex* models.

Weights should sum up to 1
and come from [0; 1]

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

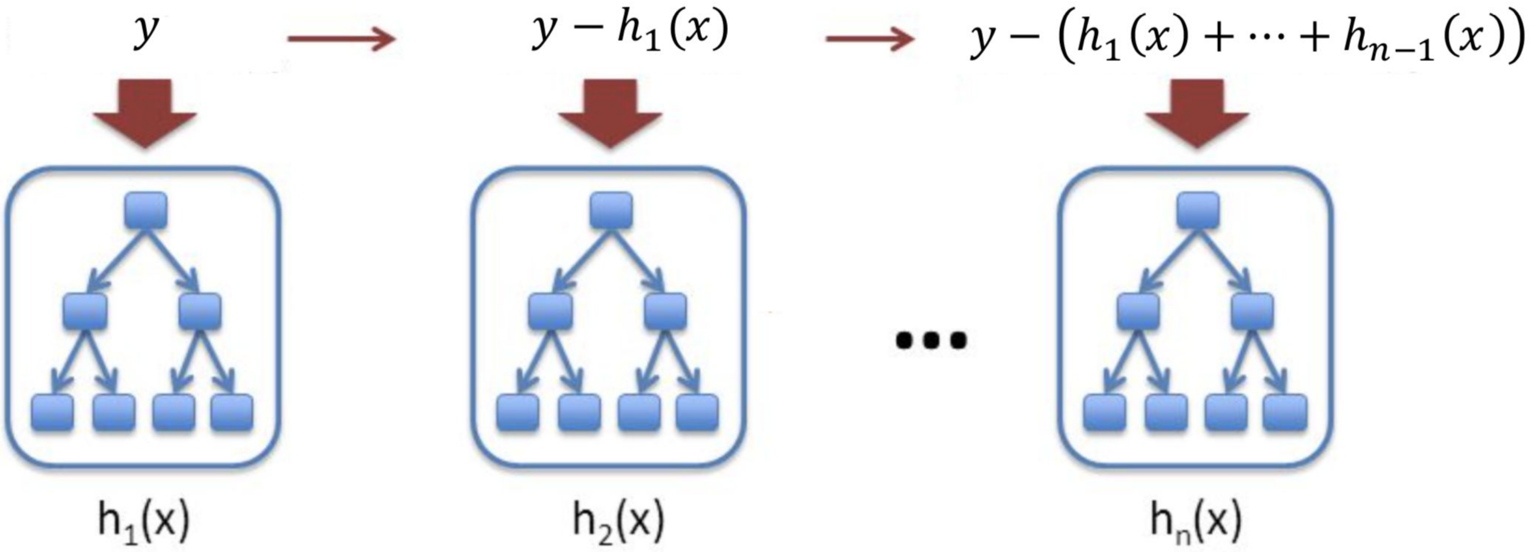
- Simple and intuitive ensembling method.
- Finding optimal weights could be tricky.
- Linear composition is not always enough.

Bagging + RSM = Random Forest



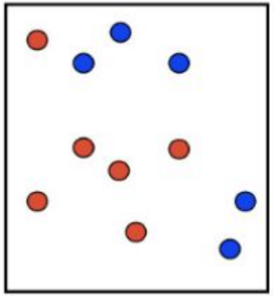
Gradient boosting

$$a_n(x) = h_1(x) + \cdots + h_n(x)$$

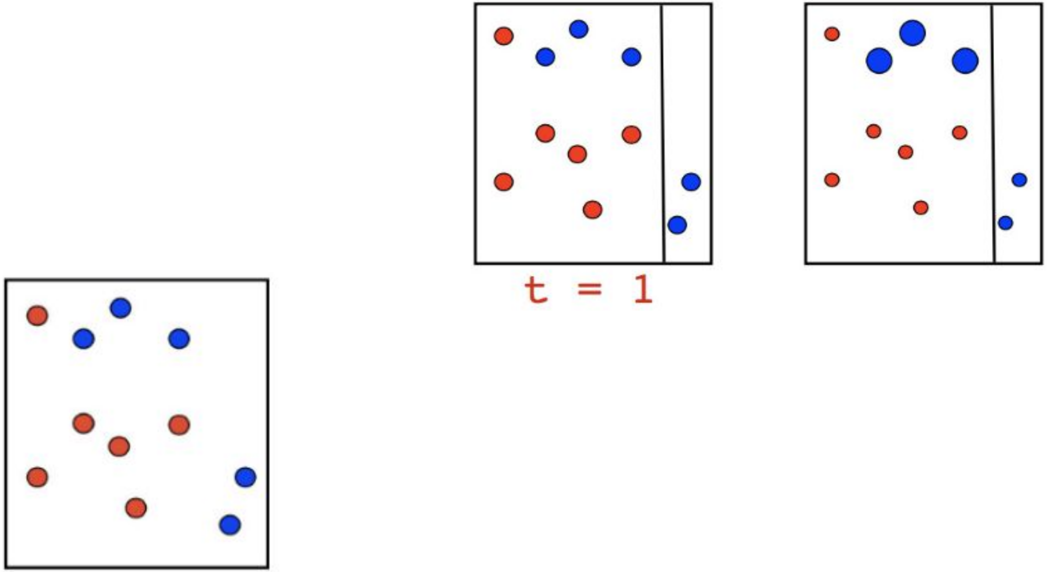


Boosting: intuition

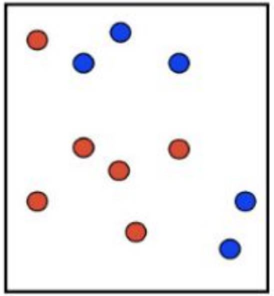
Binary classification problem.
Models - decision stumps.

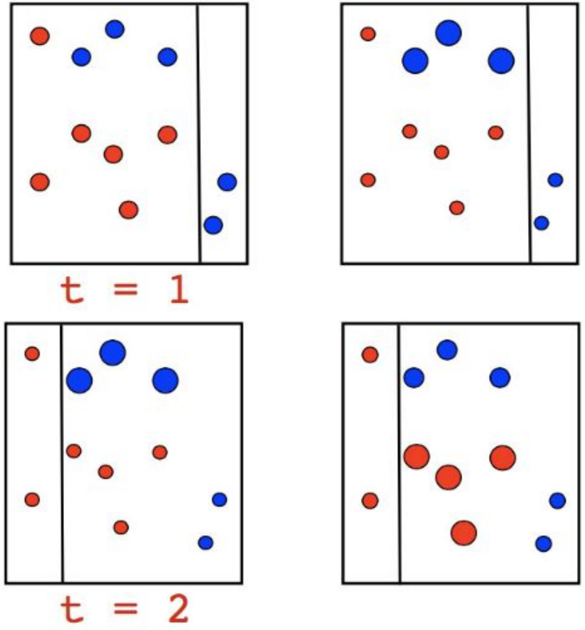


Boosting: intuition

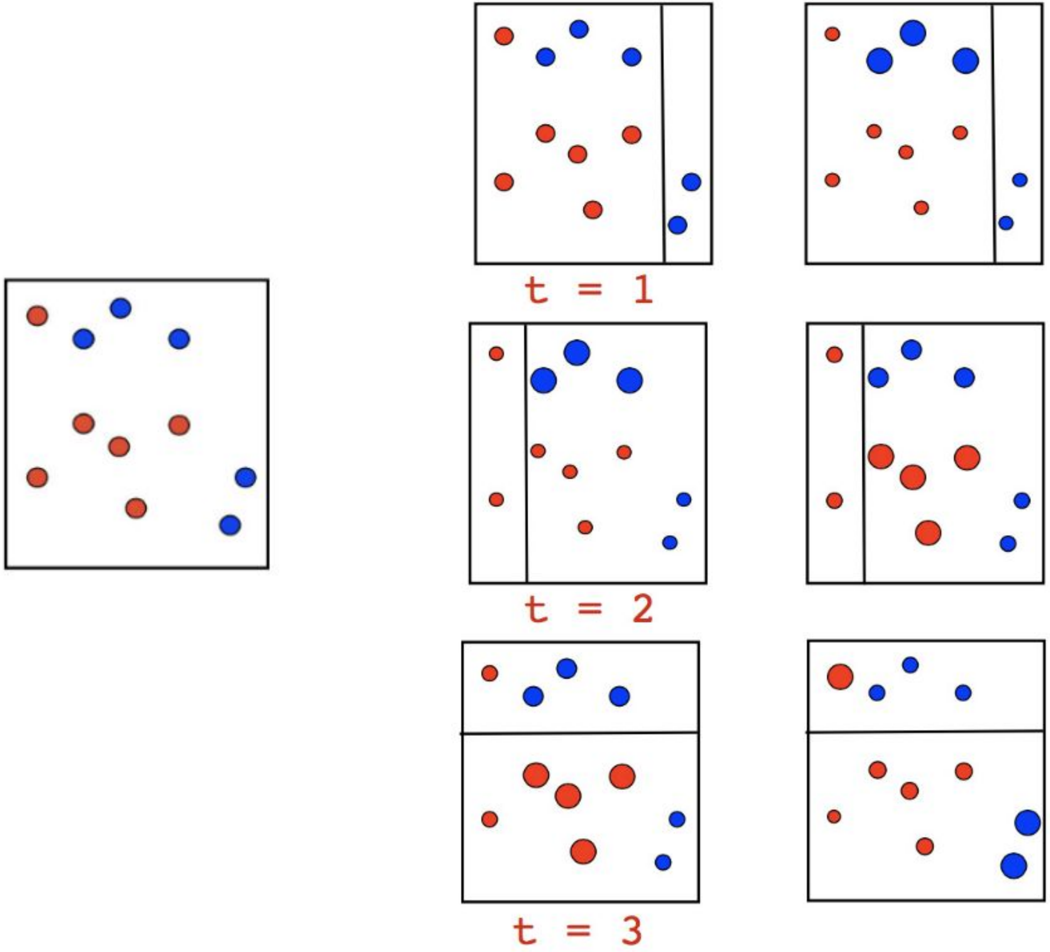


Boosting: intuition



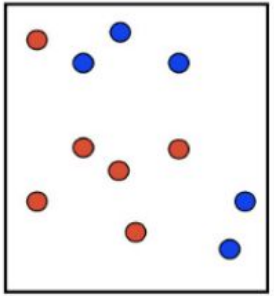


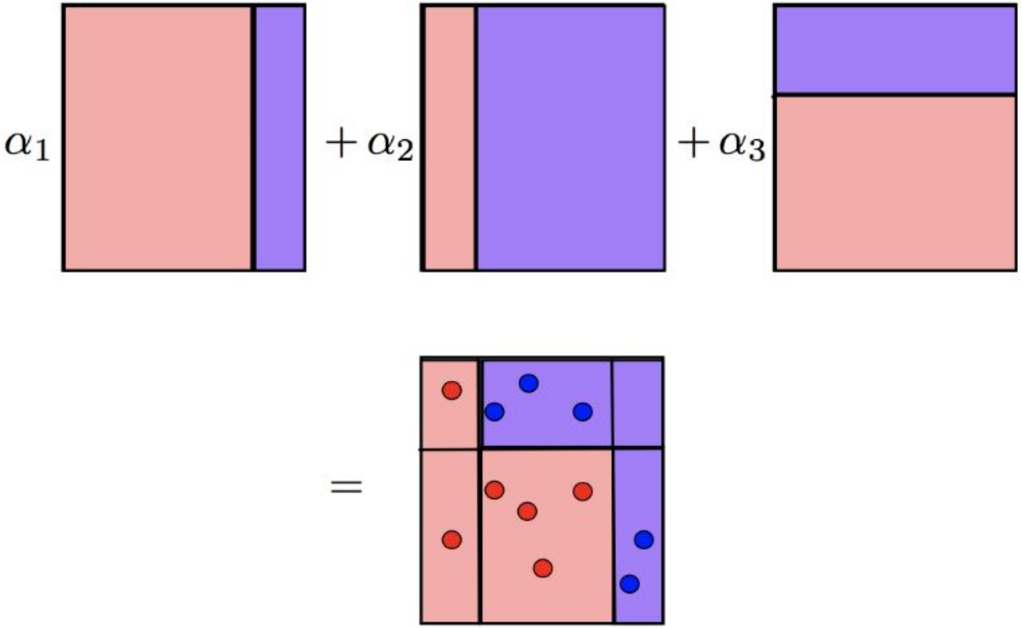
Boosting: intuition



Boosting: intuition

Binary classification problem.
Models - decision stumps.





Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\dots,n}$, loss function $L(y, f)$.

Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\ldots,n}$, loss function $L(y, f)$.

Optimal model:

$$\hat{f}(x) = \underset{f(x)}{\arg\min}\ L(y, f(x)) = \underset{f(x)}{\arg\min}\ \mathbb{E}_{x,y}[L(y, f(x))]$$

Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\ldots,n}$, loss function $L(y, f)$.

Optimal model:

$$\hat{f}(x) = \underset{f(x)}{\arg\min}\, L(y, f(x)) = \underset{f(x)}{\arg\min}\, \mathbb{E}_{x,y}[L(y, f(x))]$$

Let it be from parametric family: $\hat{f}(x) = f(x, \hat{\theta})$,

$$\hat{\theta} = \underset{\theta}{\arg\min}\, \mathbb{E}_{x,y}[L(y, f(x, \theta))]$$

Gradient boosting: theory

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \mathbb{E}_{x,y}[L(y, \hat{f}(x) + \rho \cdot h(x, \theta))],$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

Gradient boosting: theory

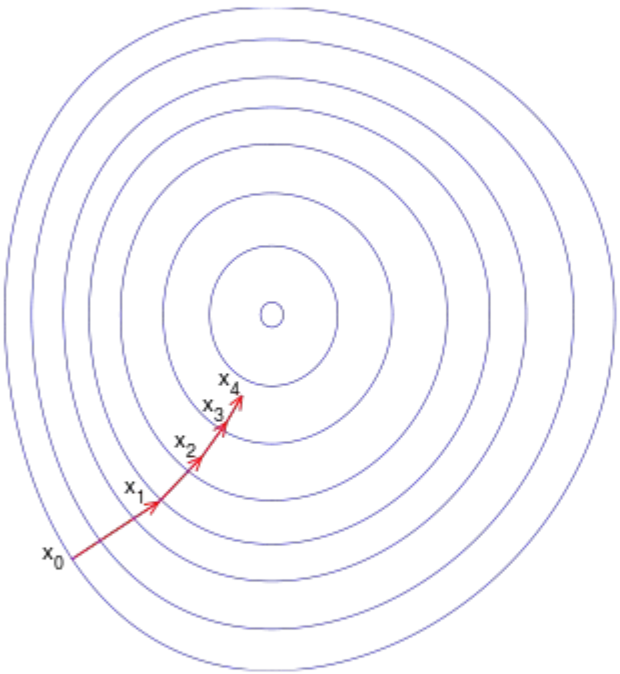
$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \mathbb{E}_{x,y}[L(y, \hat{f}(x) + \rho \cdot h(x, \theta))],$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

What if we could use gradient descent in *space of our models*?

Gradient boosting: theory



What if we could use gradient descent in *space of our models*?

Gradient boosting: theory

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

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

$$\theta_t = \arg\min_{\theta} \sum_{i=1}^{n} (r_{it} - h(x_i, \theta))^2,$$

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta_t))$$

Gradient boosting: theory

In linear regression case with MSE loss:

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)} = -2(\hat{y}_i - y_i) \propto \hat{y}_i - y_i$$

Gradient boosting: theory

What we need:

- Data.
- Loss function and its gradient.
- Family of algorithms (with constraints on hyperparameters if necessary).
- Number of iterations M.
- Initial value (GBM by Friedman): constant.

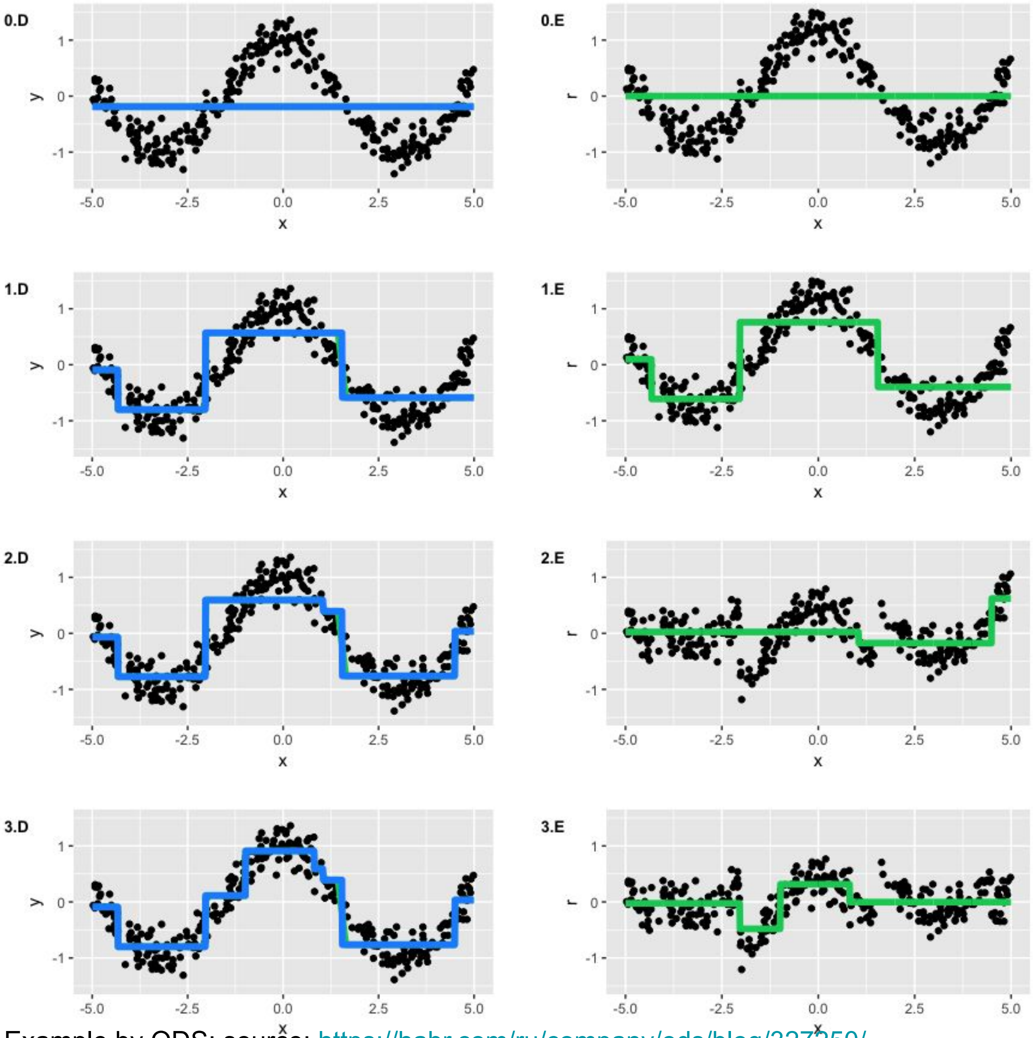
Gradient boosting: example

What we need:

- Data: toy dataset $y = cos(x) + \epsilon, \epsilon \sim \mathcal{N}(0, \frac{1}{5}), x \in [-5, 5]$
- Loss function: MSE
- Family of algorithms: decision trees with depth 2
- Number of iterations M = 3
- Initial value: just mean value

Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/

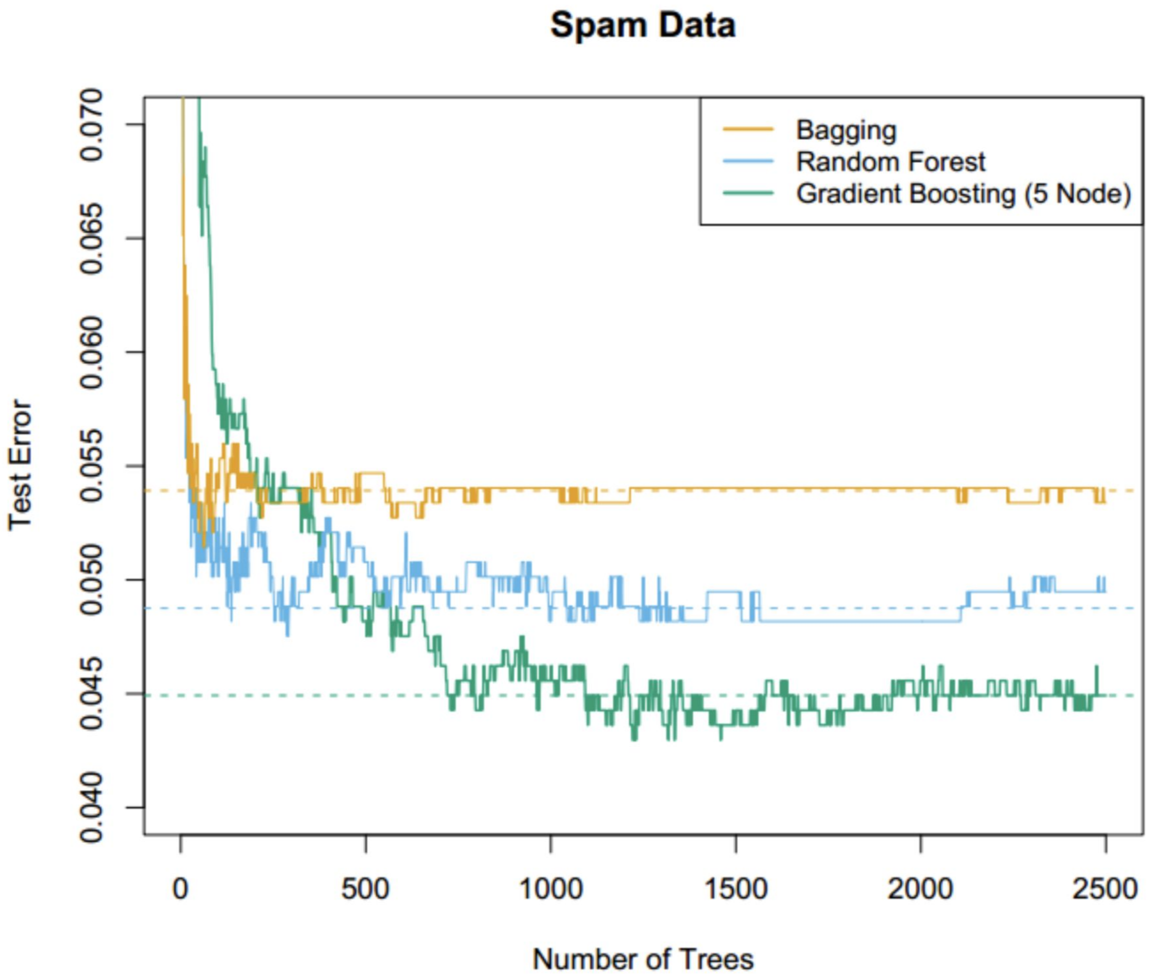
Gradient boosting: example

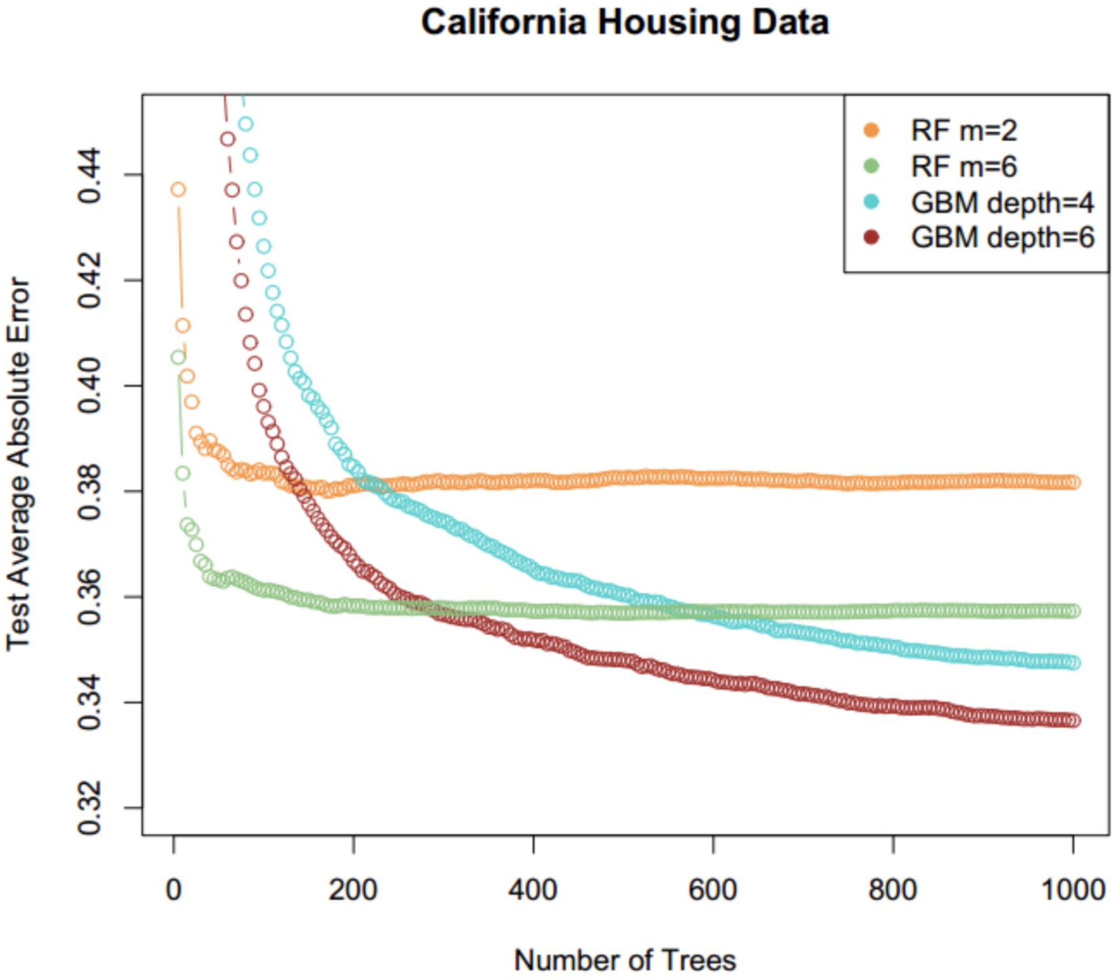


Left: full ensemble on each step.

Right: additional tree decisions.

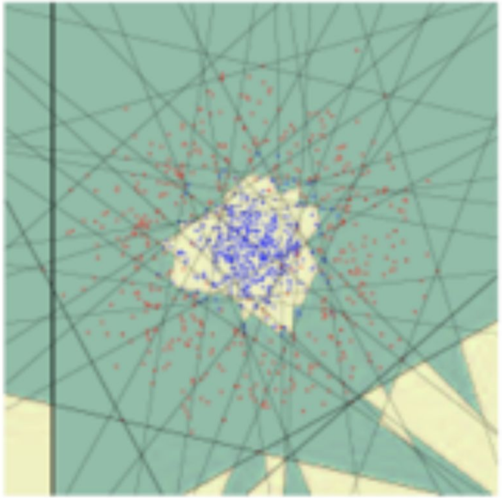
Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/

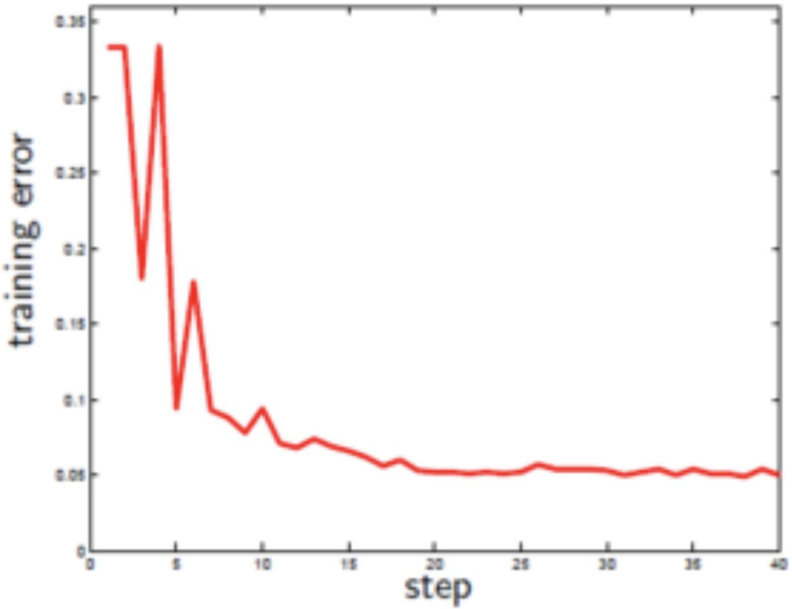

Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees


California Housing Data
RF m=2
RF m=6
GBM depth=4
GBM depth=6
Test Average Absolute Error
0.32
0.34
0.36
0.38
0.40
0.42
0.44
0
200
400
600
800
1000
Number of Trees

Boosting with linear classification methods

$t = 40$





Technical side: training in parallel

Which of the ensembling methods could be parallelized?

Technical side: training in parallel

Which of the ensembling methods could be parallelized?

- Random Forest: parallel on the forest level (all trees are independent)

Technical side: training in parallel

Which of the ensembling methods could be parallelized?

- Random Forest: parallel on the forest level (all trees are independent)
- Gradient boosting: parallel on one tree level

Recap: ensembling methods

1. Bagging.
2. Random subspace method (RSM).
3. Bagging + RSM + Decision trees = Random Forest.
4. Gradient boosting.
5. Stacking.
6. Blending.

Great demo: http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

Feature importance estimation

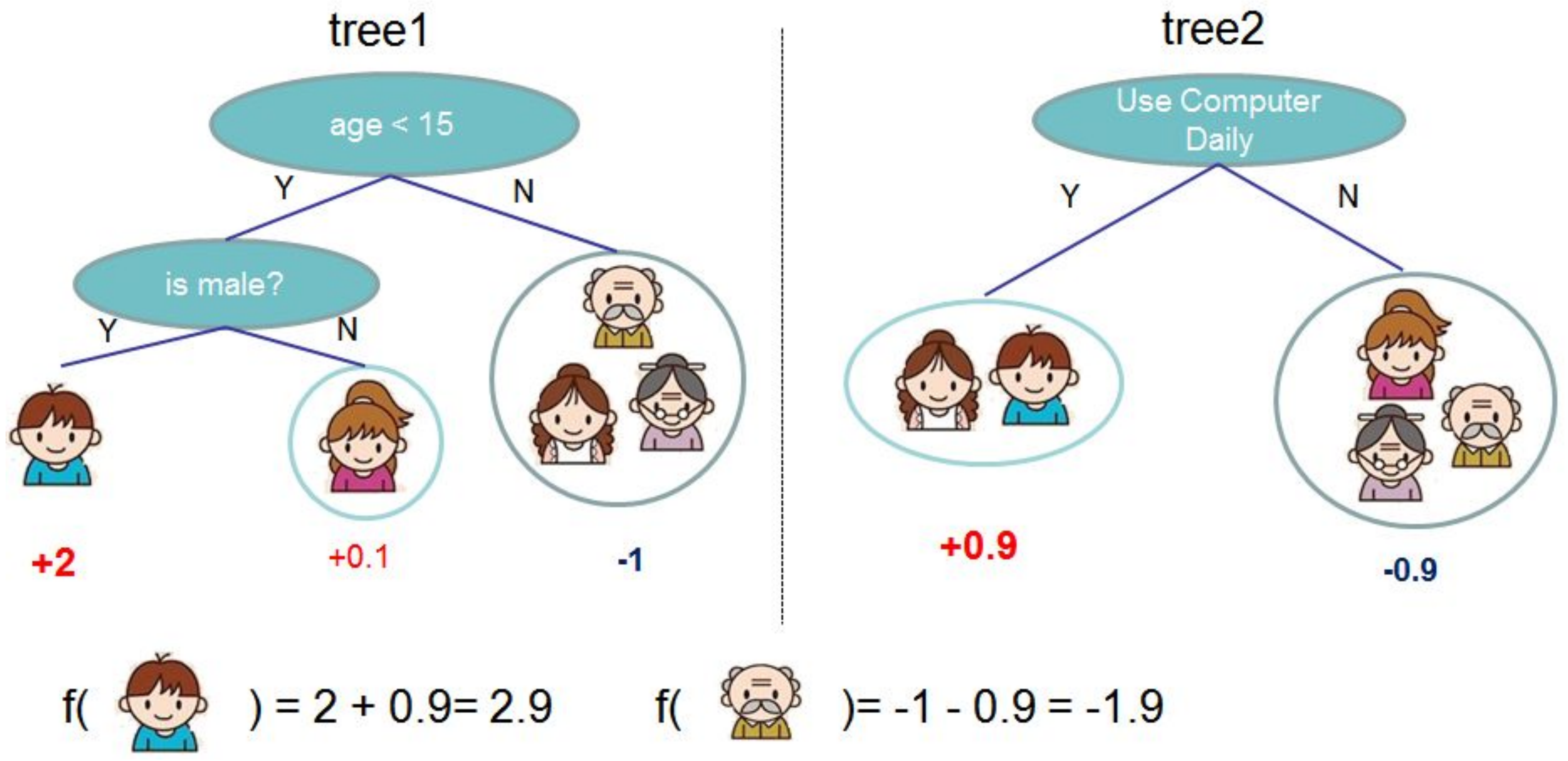


Image source: https://habr.com/ru/post/428213/

Feature importance estimation

1. Permutation importance
2. Partial Dependence Plots (PDP)
3. Tree specific:
   a. Gain
   b. Frequency (Split Count)
   c. Cover (weighted Split Count)
4. Shap

Permutation importance

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

Image source: https://www.kaggle.com/dansbecker/permutation-importance

Permutation importance

Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

Train model

Observe changes caused by feature random permutations

Image source: https://www.kaggle.com/dansbecker/permutation-importance

Partial Dependence Plots

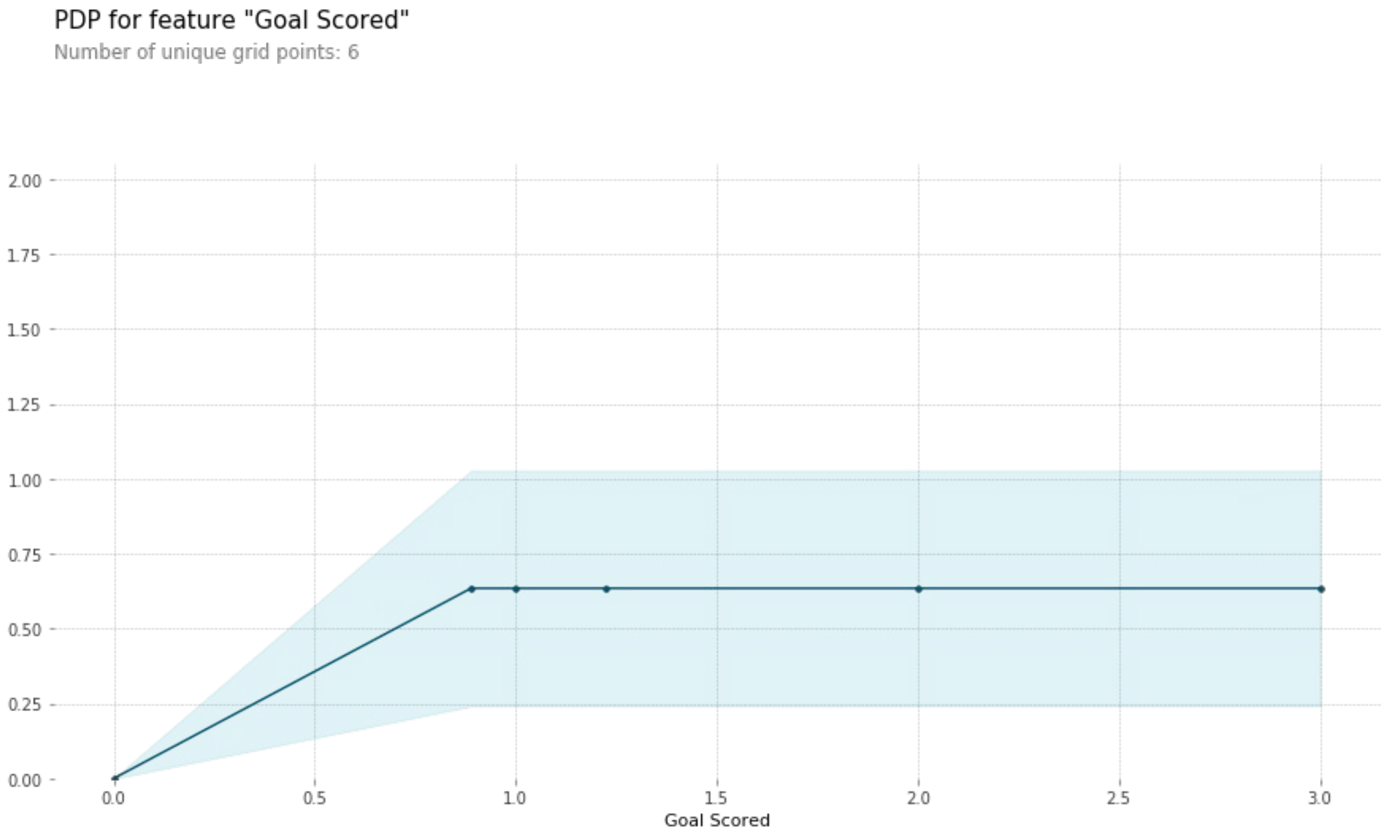


Image source: https://www.kaggle.com/dansbecker/partial-plots

Importance estimation problems

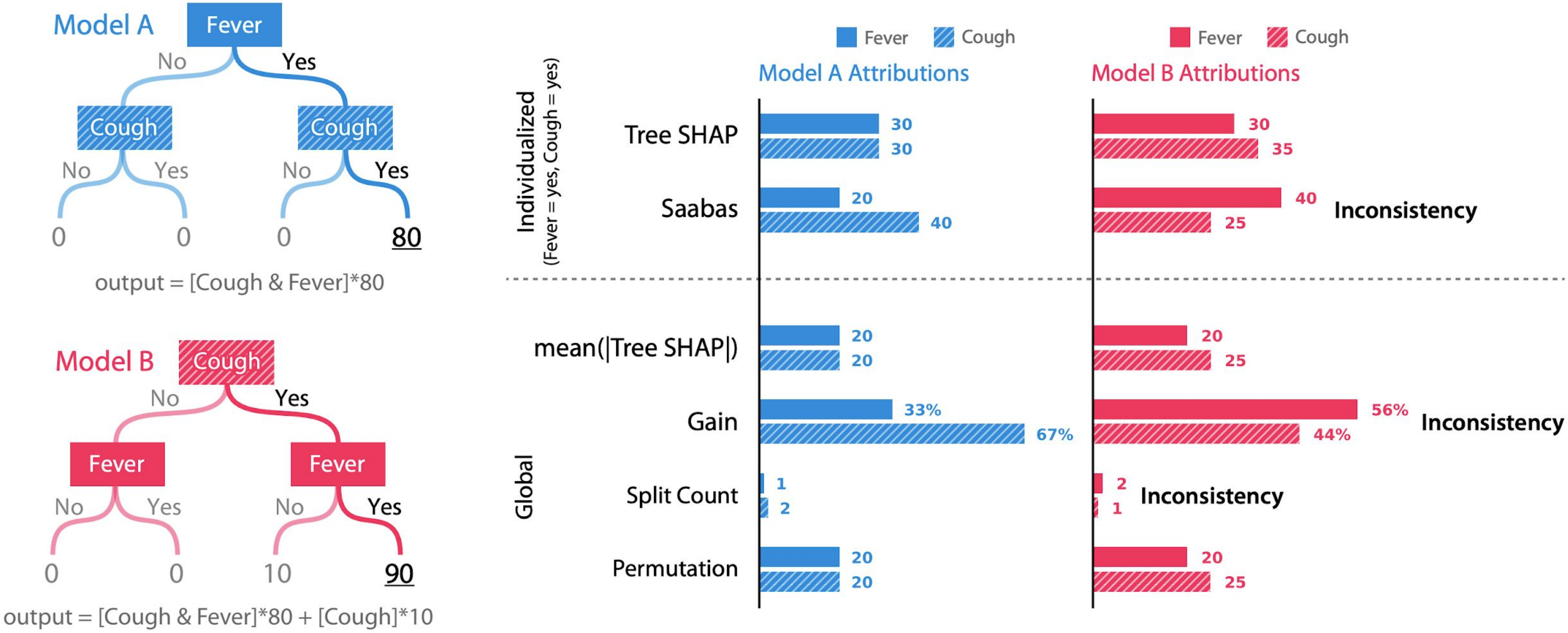


Image source: [“Consistent Individualized Feature Attribution for Tree Ensembles”]() paper

Shap values

Consider *i-th* feature. Shap value will be

$$\phi_i(p) = \sum_{S \subseteq N/\{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(p(S \cup \{i\}) - p(S))$$

where $p(S \cup \{i\})$ is model prediction on feature subset S with *i-th* feature added.

Shap values

Consider *i-th* feature. Shap value will be

$$\phi_i(p) = \sum_{S \subseteq N/\{i\}} \frac{|S|!(n-|S|-1)!}{n!}(p(S \cup \{i\}) - p(S))$$

where $p(S \cup \{i\})$ is model prediction on feature subset S with *i-th* feature added.

SHAP values are the only consistent and locally accurate individualized feature attributions

See [Consistent Individualized Feature Attribution for Tree Ensembles]() paper for more info

Outro

1. Bagging + RSM + Decision trees = Random Forest.
2. Gradient boosting is powerful but prone to overfitting
3. Stacking & Blending are great techniques
4. Consider using SHAP values to estimate feature importances.

Great demo: http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html