# Введение в машинное обучение

«Деревья решений. Bagging. Random Forest. Boosting.»

Лекция 6

Комаров Иван Владимирович
НГУ
2020

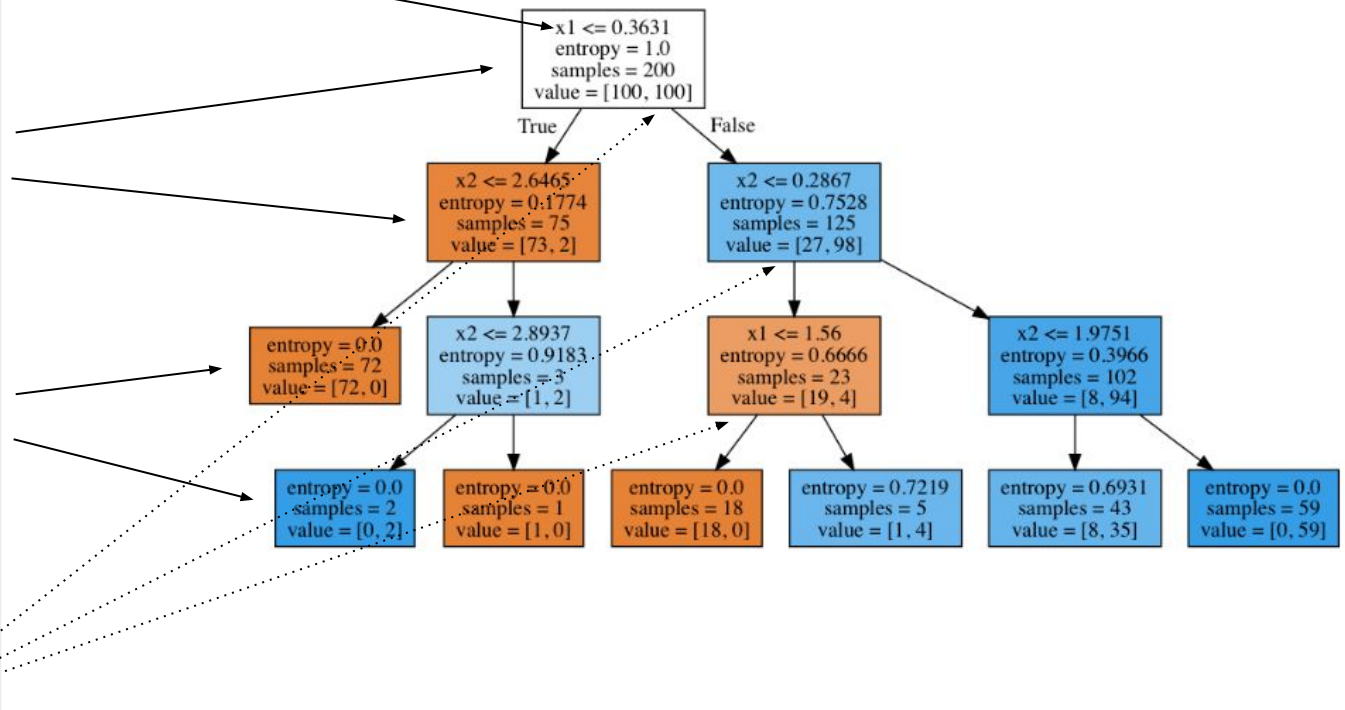
# 2 основных вида

- **CART (Classification and Regression Tree)** – это алгоритм построения **бинарного** дерева решений. Каждый узел дерева при разбиении имеет только двух потомков. Как видно из названия алгоритма, решает задачи классификации и регрессии. Используется **в scikit-learn**.

- **C4.5** – алгоритм построения дерева решений, количество потомков у узла не ограничено. Решает только задачи **классификации**.

# Терминология

| | |
|---|---|
| Проверка | Условие в узле |
| Узел | Внутренний узел дерева, узел проверки |
| Лист | Конечный узел дерева, узел решения |
| Глубина | Самое большое количество родителей (3) |



# **Жадные:**
# **«берут все, что дают**, не отдают»

Большинство из известных алгоритмов являются "жадными алгоритмами".

**Перебирают все признаки и все возможные значения в выборке.**

*Если один раз был выбран атрибут, и по нему было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другой атрибут, который дал бы лучшее разбиение.*

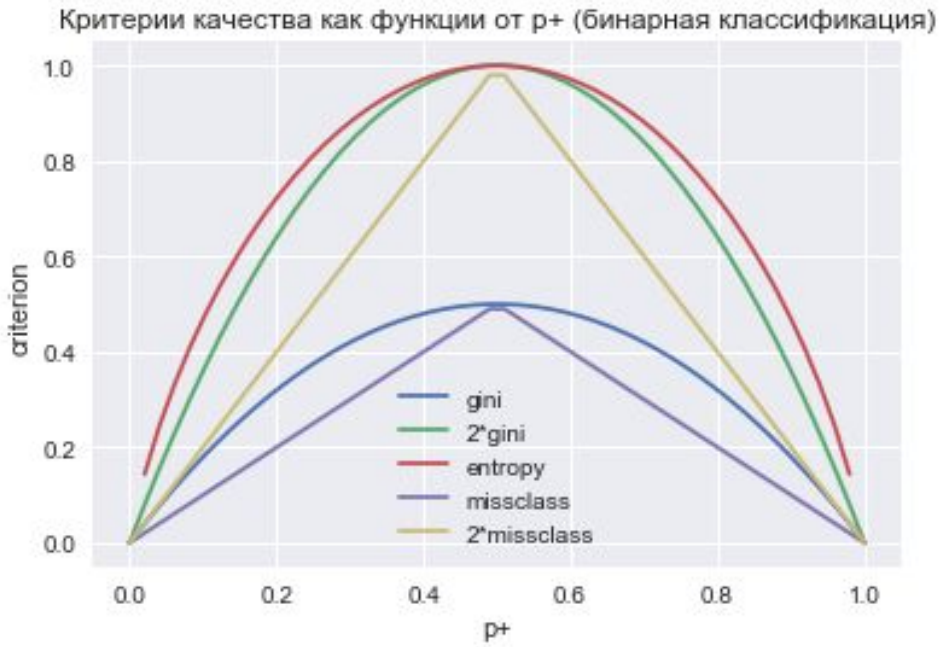
*И поэтому на этапе построения нельзя сказать даст ли выбранный атрибут, в конечном итоге, оптимальное разбиение.*

По https://basegroup.ru/community/articles/description

# Алгоритм

- Первое множество (корень) : есть ли такие **два** подмножества, которые *улучшат предсказания*? Подмножества определим по какому-то признаку и его значению. Делим первое множество.
- И так поступаем далее. Однако если улучшения нет, или достигли порог ошибки, или решили остаться на этой глубине дерева, то это лист.

# Критерии разбиения: Классификация




If a target is a classification outcome taking on values 0,1,...,K-1, for node $m$, representing a region $R_m$ with $N_m$ observations, let

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k)$$

be the proportion of class k observations in node $m$

Common measures of impurity are Gini

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

Entropy

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$$

and Misclassification

$$H(X_m) = 1 - \max(p_{mk})$$

where $X_m$ is the training data in node $m$

https://habr.com/en/company/ods/blog/322534/#kak-stroitsya-derevo-resheniy

# Критерии разбиения: Регрессия. «Улучшат предсказания».

Mean Squared Error:

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

Mean Absolute Error:

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y}_m|$$

The impurity at $m$ is computed using an impurity function $H()$, the choice of which depends on the task being solved (classification or regression)

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

Select the parameters that minimises the impurity

$$\theta^* = \operatorname{argmin}_\theta G(Q, \theta)$$

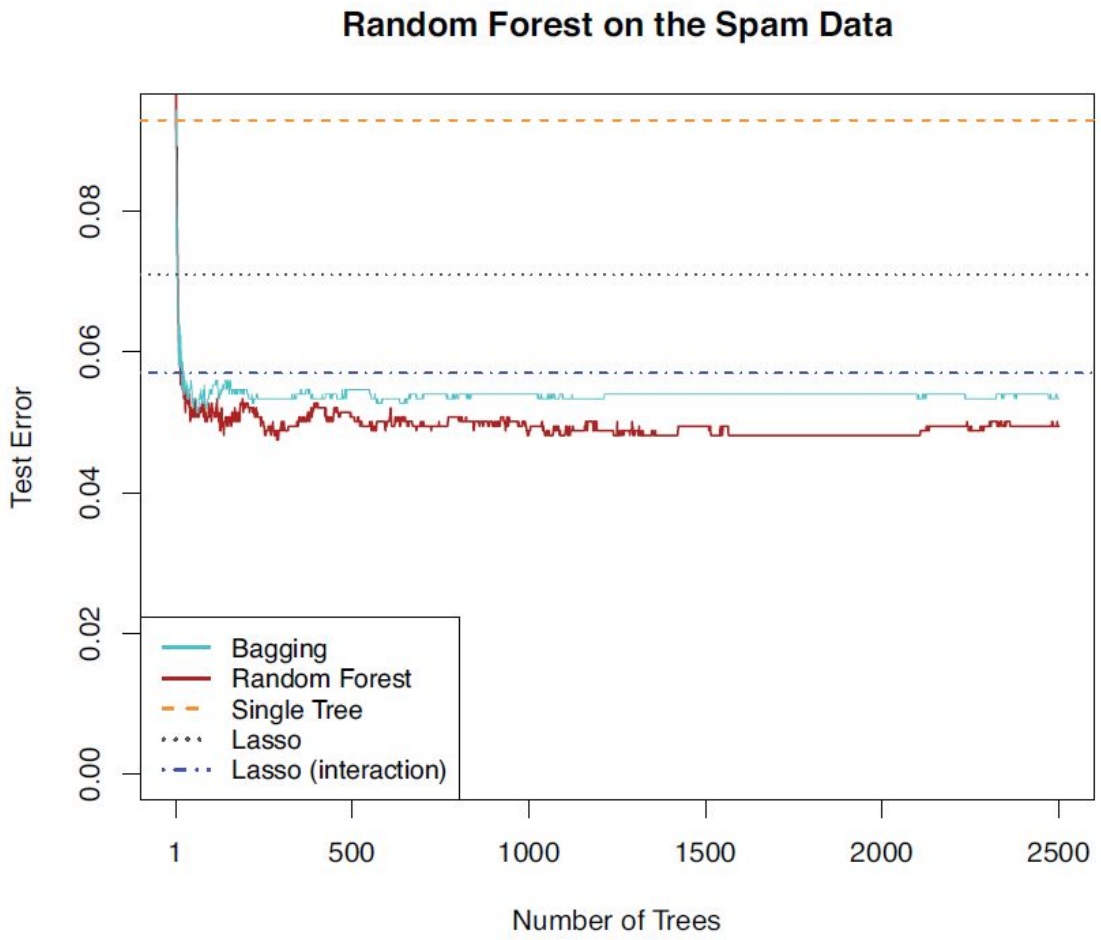
# Bagging, Random Forests, Boosting

Random forest Grow many deep regression trees to randomized versions of the training data, and average them. Here “randomized” is a wide ranging term, and includes bootstrap sampling (bagging) and/or subsampling of the observations, as well as subsampling of the variables.

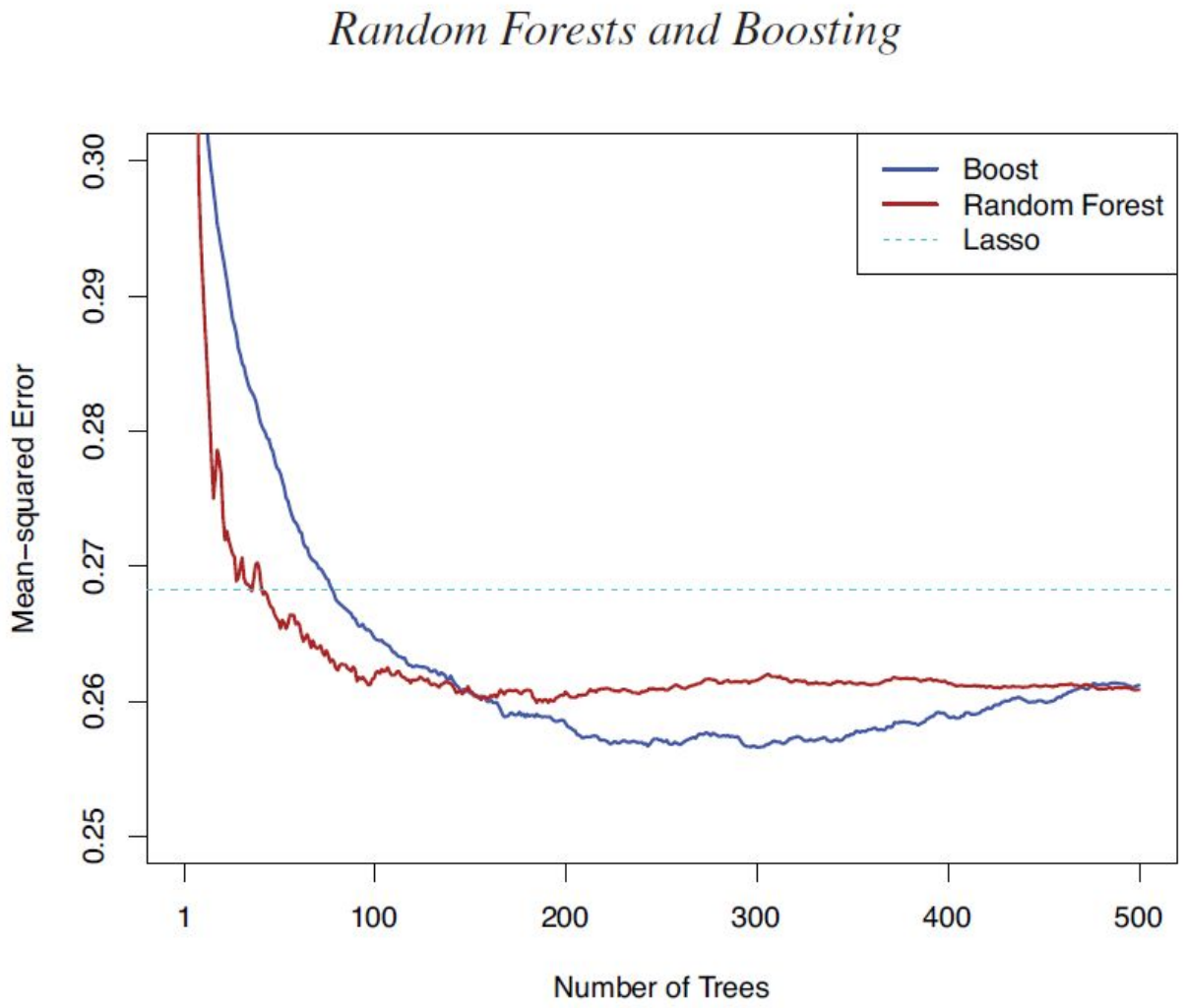
Boosting Repeatedly grow shallow trees to the residuals, and hence build up an additive model consisting of a sum of trees.

https://web.stanford.edu/~hastie/CASI_files/PDF/casi.pdf

# Random Forest лучше Lasso, Tree, Bagging в этом показательном примере



# Boosting бьет Random Forest в этом показательном примере




As is often the case, boosting slightly outperforms a random forest here, but at a price. ***Careful tuning of boosting*** requires considerable extra work, with time-costly rounds of cross-validation, whereas random forests are almost automatic.

# Gradient Boosting

Интуитивно: ошибка = - градиент

см. слайды Chang Li