Решающие деревья: простая и понятная модель

Лазар В. И., Козлова Е. Р.

8 октября 2025 г.

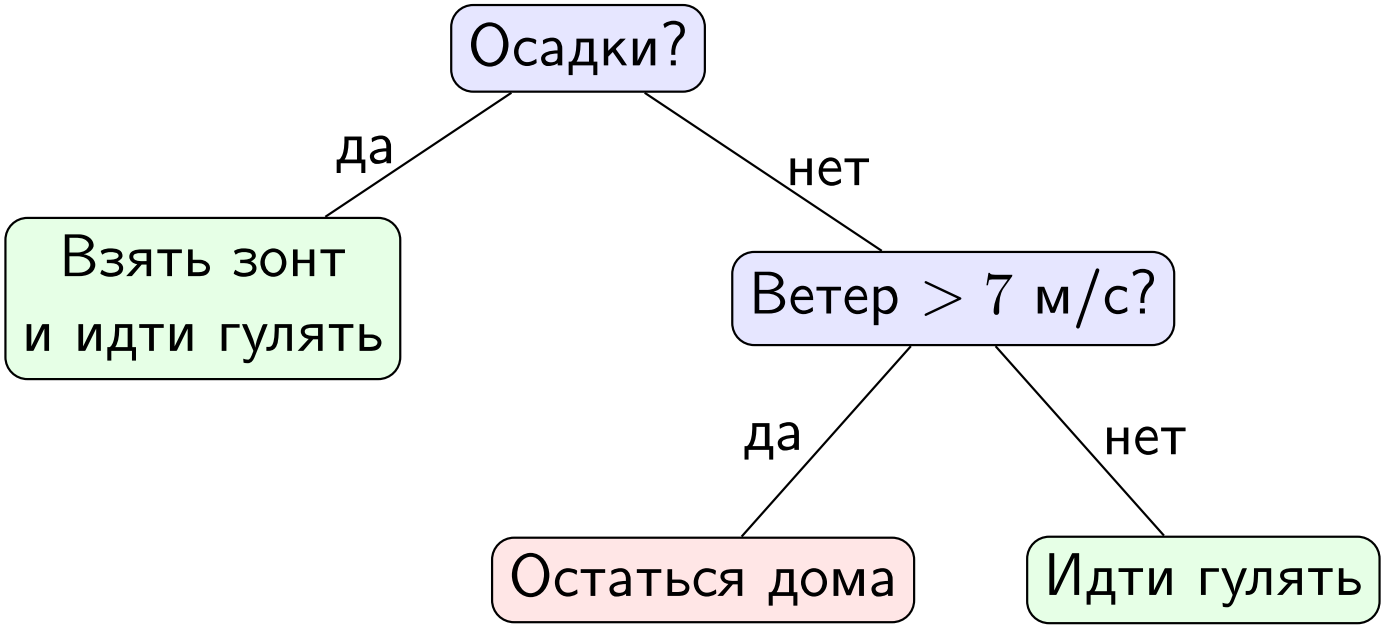
Зачем дерево?

- **Идея дерева**: задаём последовательность простых вопросов вида «Признак x_j больше порога?».
- Итог — путь от **корня** к **листу** даёт решение.

Части решающего дерева

- **Узел** — проверка правила (например, $x_j \leq t$).
- **Рёбра** — варианты ответа («да»/«нет»).
- **Лист** — предсказание: класс (классификация) или число (регрессия).
- **Глубина** — максимальное число вопросов от корня до листа.

Мини-пример дерева (погода и прогулка)



Как дерево делает предсказание (инференс)

Шаги:

1. Стартуем в корне.
2. Проверяем условие в узле (например, $x_j \leq t$).
3. Идём по ветке «да» или «нет» в следующий узел.
4. Повторяем, пока не попадём в лист.
5. В листе берём предсказание:
   классификация — самый частый класс в листе;
   регрессия — среднее значение ответов в листе.

Сложность: $O(\text{глубина})$ проверок на один объект.

Критерии для классификации

Пусть в узле есть выборка S с долями классов $p_1, \ldots, p_K$.

Индекс Джини

$$\text{Gini}(S) = 1 - \sum_{k=1}^{K} p_k^2$$

Энтропия

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

Прирост информации (качество сплита по признаку x_j и порогу t)

Пусть разбиение даёт дочерние множества S_L, S_R. Тогда

$$\Delta = \mathrm{Impurity}(S) - \frac{|S_L|}{|S|}\mathrm{Impurity}(S_L) - \frac{|S_R|}{|S|}\mathrm{Impurity}(S_R),$$

где $\mathrm{Impurity}$ — Gini или энтропия. Чем больше Δ, тем лучше сплит.

Критерии для регрессии

Пусть ответы в узле — $y_1, \ldots, y_n$, среднее $\bar{y}$.

Среднеквадратичная ошибка (MSE) в узле

$$\mathrm{MSE}(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} (y_i - \bar{y})^2$$

Снижение ошибки при сплите

$$\Delta = \mathrm{MSE}(S) - \frac{|S_L|}{|S|}\mathrm{MSE}(S_L) - \frac{|S_R|}{|S|}\mathrm{MSE}(S_R).$$

Выбираем признак и порог с максимальным Δ.

Идея обучения (жадно, сверху вниз)

1. В текущем узле перебираем **кандидатные сплиты**: признаки и пороги.
2. Считаем **снижение нечистоты** Δ (Gini/энтропия для классификации, MSE для регрессии).
3. Берём **лучший** сплит. Разделяем выборку на S_L и S_R.
4. Рекурсивно повторяем для дочерних узлов.
5. Останавливаемся по критериям: достигнута **макс. глубина**, мало объектов в узле, **улучшение незначительно** и т.п.
6. (Опционально) **Обрезаем** дерево (pruning), чтобы уменьшить переобучение.

1. Функция $\textsc{GrowNode}(S, depth)$:
   1. Если `depth` $\geq$ `max_depth` **или** $|S| <$ `min_samples_leaf` **или** все объекты одного класса:
      - создать **лист** с предсказанием (мода класса / среднее y).
      - вернуть лист.
   2. Иначе: для каждого признака $x_j \in \mathcal{F}$ и порога t из кандидатов:
      - разбить S на $S_L = \{x : x_j \leq t\}$ и $S_R = \{x : x_j > t\}$,
      - вычислить $\Delta(x_j, t)$ (снижение нечистоты).
   3. Выбрать $(j^*, t^*) = \arg\max \Delta(x_j, t)$. Если улучшение слишком мало — сделать лист.
   4. Рекурсивно построить детей: $\textsc{GrowNode}(S_L, depth + 1)$ и $\textsc{GrowNode}(S_R, depth+1)$.
   5. Вернуть внутренний узел $(x_{j^*} \leq t^*)$ с привязанными детьми.
2. Корень: $\textsc{GrowNode}(S, 0)$.
3. (Опц.) Пост-обрезка по стоимости сложности: минимизируем $R_\alpha(T) = R(T) + \alpha|T|$.

Работа с признаками

- **Числовые:** ищем пороги t (часто по отсортированным значениям).
- **Категориальные:** бинарные сплиты по подмножествам категорий (или one-hot кодирование).
- **Пропуски:** простой вариант — заполнять медианой/модой; продвинутый — surrogate splits.
- **Масштабирование:** деревьям в целом не требуется нормировка признаков.

Переобучение и как с ним бороться

- Глубокие деревья могут **запоминать** данные.
- **Пред-обрезка** (pre-pruning): ограничить `max_depth`, `min_samples_leaf`, `min_impurity_decrease`.
- **Пост-обрезка** (post-pruning): удалять ветви, если они почти не улучшают качество (например, по $R_\alpha(T)$).
- **Валидация**: отложенная выборка или кросс-валидация, чтобы выбрать параметры.

Сложность и масштабируемость

- **Обучение:** при аккуратной реализации около $O(n\,d\log n)$, где n — объектов, d — признаков.
- **Инференс:** $O(\text{глубина})$ проверок на объект.
- Хорошо работают на табличных данных, устойчивы к монотонным преобразованиям признаков.

Плюсы и минусы

Плюсы

- Понятны и интерпретируемы.
- Не требуют масштабирования.
- Умеют смешивать числовые и категориальные признаки.
- Быстрый инференс.

Минусы

- Склонность к переобучению без обрезки.
- Нестабильность при малых изменениях данных.
- Пороги — кусочно-постоянные решения (ступеньки).

Пример: классификация (индекс Джини)

В узле 20 объектов: 12 «положительный» и 8 «отрицательный» классы.

$$\mathrm{Gini}(S) = 1 - \left(\frac{12}{20}\right)^2 - \left(\frac{8}{20}\right)^2 = 1 - 0.36 - 0.16 = 0.48.$$

После сплита получили:

$$|S_L| = 10: \ (8^+, 2^-) \Rightarrow \mathrm{Gini}(S_L) = 1 - 0.64 - 0.04 = 0.32, \quad |S_R| = 10$$

Взвешенная нечистота:

$$\frac{10}{20} \cdot 0.32 + \frac{10}{20} \cdot 0.48 = 0.40.$$

Прирост: $\Delta = 0.48 - 0.40 = 0.08$. Чем больше Δ, тем лучше сплит.

Ключевые гиперпараметры

- `criterion`: `gini` / `entropy` (классификация), `mse` (регрессия).
- `max_depth`, `min_samples_leaf`, `min_impurity_decrease`.
- `max_features`: ограничение числа признаков при выборе сплита.

Почему ансамбли? Интуиция

- **Один классификатор** может ошибаться по разным причинам (шум, малая выборка).
- **Ансамбль** объединяет несколько моделей и чаще **снижает дисперсию** (колебания) и/или **смещение**.
- Идея: «голосование» или «усреднение» многих деревьев даёт более устойчивый результат.
- Три подхода: **бэггинг** (bagging), **бустинг** (boosting), **стэкинг** (stacking).

Бэггинг (Bootstrap Aggregating)

- Обучаем много деревьев на **бутстрэп-выборках** (случайные выборки с возвращением из обучающих данных).
- **Классификация**: итог — **голосование большинством**. **Регрессия**: итог — **среднее** предсказаний.
- **Эффект:** сильное снижение **дисперсии** (меньше переобучения).
- **OOB-оценка** (Out-of-Bag): объекты, не попавшие в бутстрэп, служат для быстрой оценки качества без отдельной валидации.
- **Случайный лес** = бэггинг + случайный поднабор признаков при поиске сплита (`max_features`) $\Rightarrow$ деревья становятся менее похожими, ансамбль — сильнее.

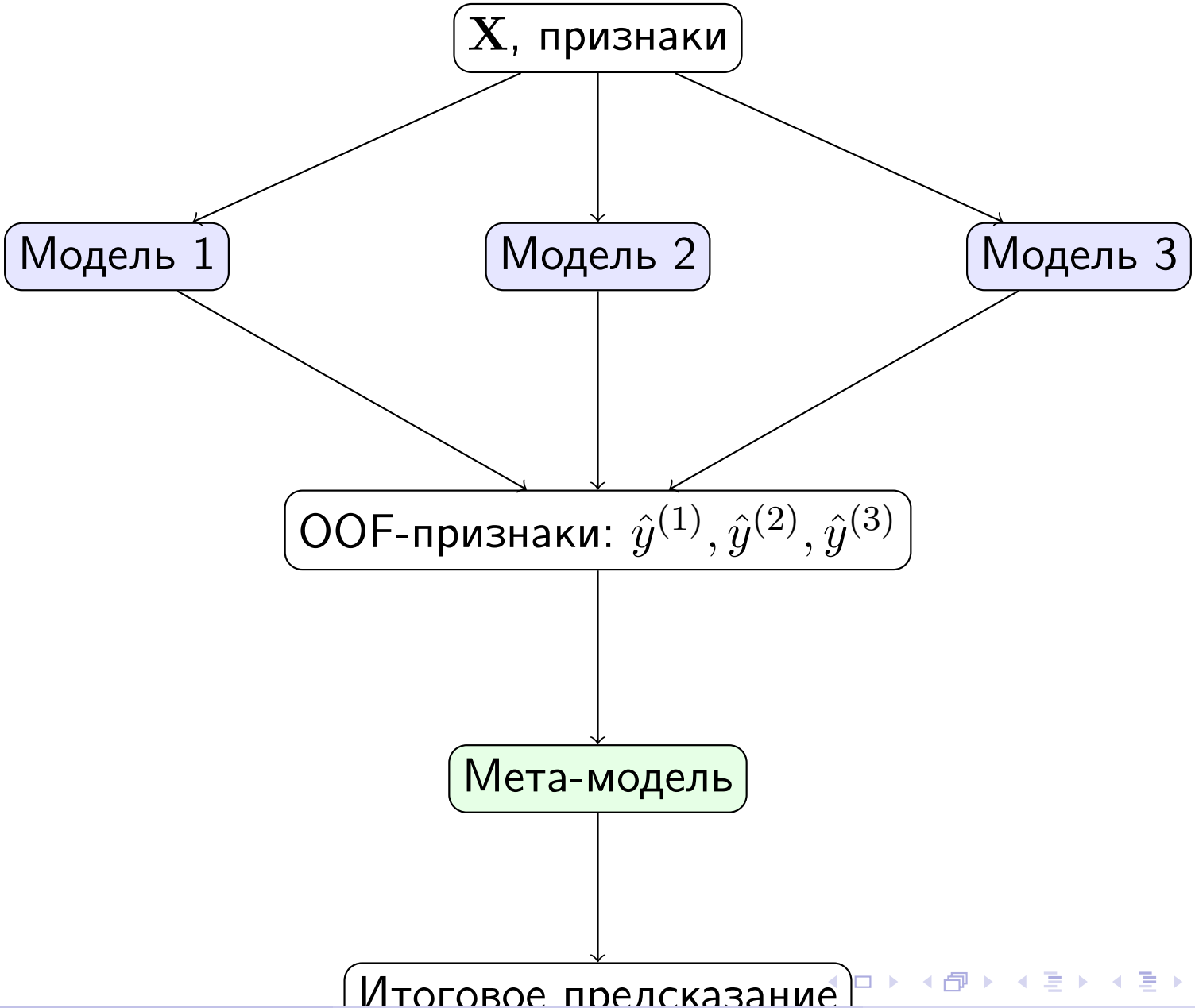
Бэггинг: ключевые параметры

- `n_estimators`: число деревьев (обычно десятки/сотни).
- `max_depth`, `min_samples_leaf`: сложность базовых деревьев.
- `max_features`: доля/число признаков для сплита (случайный лес).
- `bootstrap`: брать ли бутстрэп-выборки (обычно да).
- **Плюсы:** простота, устойчивость, хорошая база по умолчанию.
- **Минусы:** чуть меньшая интерпретируемость, скорость инференса ниже (много деревьев).

Стэкинг (Stacking): «модель над моделями»

- Обучаем **разные** базовые модели (например, дерево, kNN, логистическую регрессию).
- Формируем **признаки второго уровня**: предсказания базовых моделей.
- Обучаем **мета-модель** на этих признаках, чтобы **научиться комбинировать** их лучше простого среднего.
- Чтобы избежать «подглядывания», используем **OOF-предсказания**: K-fold разбиение, на каждом фолде базовые модели предсказывают на валидации; склеиваем OOF-стек-признаки и учим мета-модель.

Схема стэкинга (OOF)



Когда что выбирать? (кратко)

- **Случайный лес (бэггинг)** — сильная «база без тюнинга», устойчив к шуму, быстро стартует.
- **Бустинг** — лидер по качеству на табличных задачах при грамотной настройке.
- **Стэкинг** — когда есть **разнородные** сильные базовые модели, и вы хотите выжать максимум.

Ансамбли: плюсы/минусы и гиперпараметры

Плюсы

- Выше качество, чем у одного дерева.
- Снижение дисперсии (бэггинг) и/или смещения (бустинг).
- Гибкость стэкинга.

Минусы

- Меньше интерпретируемость.
- Дольше обучение и инференс.
- Нужна аккуратная валидация (особенно для стэкинга).

Частые параметры: `n_estimators`, `learning_rate`, `max_depth`, `max_features`, `subsample/colsample`.