# Шпаргалка по деревьям решений / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

## Содержание



### Деревья Решений: Базовый Алгоритм

Дерево решений — это один из фундаментальных алгоритмов машинного обучения. Он используется как самостоятельно, так и в качестве строительного блока для более сложных ансамблевых методов, таких как Случайный Лес и Градиентный Бустинг. Понимание принципов работы деревьев решений необходимо для освоения ансамблей.

#### Структура Дерева Решений

Дерево решений можно визуализировать как блок-схему, где каждый шаг представляет собой проверку некоторого условия:

- **Узлы (Nodes):** Представляют собой проверку условия (вопрос) по одному из признаков объекта (например, "Возраст > 30?"). Корневой узел (Root Node) — самый верхний узел. Внутренние узлы (Internal Nodes) — узлы, имеющие дочерние узлы.
- **Ветви (Edges/Branches):** Соединяют узлы и представляют собой результат проверки условия ("Да" / "Нет"). Они определяют путь объекта по дереву.
- **Листья (Leaves / Terminal Nodes):** Конечные узлы, не имеющие дочерних узлов. В листьях содержится итоговый прогноз (например, метка класса "Кликнет" / "Не кликнет" для классификации или среднее значение целевой переменной для регрессии).

## 1 I. Процесс Принятия Решения

#### Путь Объекта по Дереву

Для получения прогноза объект "проходит" по дереву, начиная с корневого узла. В каждом внутреннем узле проверяется соответствующее условие по одному из признаков объекта. В зависимости от результата проверки выбирается одна из ветвей, ведущая к следующему узлу. Этот процесс продолжается до тех пор, пока объект не достигнет листового узла. Прогноз, содержащийся в этом листе, является результатом работы дерева для данного объекта. *Аналогия: Процесс напоминает игру в "Угадай животное", где последовательность ответов на вопросы ("У него есть перья?", "Он умеет летать?") приводит к финальному ответу.*

## 2 II. Алгоритм Построения Дерева

### 2.1. A. Выбор Оптимального Разделения

#### Цель: Максимальная Однородность Групп

Ключевой этап построения дерева — рекурсивный выбор наилучшего разделения (split) для каждого узла. Алгоритм ищет признак и пороговое значение, которые разделят данные, попавшие в узел, на две (или более) дочерние подгруппы таким образом, чтобы эти подгруппы были максимально "чистыми" или однородными по целевой переменной. *Аналогия: При сортировке корзины с яблоками и грушами, вопрос "Объект круглый?" является хорошим разделяющим правилом, так как он эффективно отделяет одни фрукты от других.*

### 2.2. B. Критерии Качества Разделения (Классификация)

#### Измерение Нечистоты (Impurity)

В задачах классификации "чистота" узла означает, что в нем преобладают объекты одного класса. Для количественной оценки используются **меры нечистоты**. Алгоритм стремится выбрать разделение, которое максимально уменьшает нечистоту.

### Основные Меры Нечистоты

- **Критерий Джини (Gini Impurity):**
  Измеряет вероятность того, что случайно выбранный элемент из узла будет неправильно классифицирован, если его класс присвоить случайно в соответствии с распределением классов в этом узле.

$$G(D) = 1 - \sum_{k=1}^{K} p_k^2$$

  где $D$ — набор данных в узле, $K$ — количество классов, $p_k$ — доля объектов класса $k$ в узле $D$. *Значение Gini Impurity варьируется от 0 (абсолютно чистый узел) до $1 - 1/K$ (максимально смешанный узел).*
- **Энтропия (Entropy):**
  Мера неопределенности или хаоса в узле, основанная на теории информации. Используется в алгоритмах ID3, C4.5, C5.0.

$$E(D) = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

  где $p_k$ — доля объектов класса $k$. Если $p_k = 0$, то слагаемое $p_k \log_2(p_k)$ считается равным 0. *Энтропия равна 0 для чистого узла и достигает максимума ($\log_2 K$), когда все классы представлены равномерно.*

### Прирост Информации (Information Gain)

Для оценки эффективности разделения используется **Прирост Информации**. Он показывает, насколько уменьшилась нечистота (Gini или Entropy) после разделения узла $D$ на дочерние узлы $D_1, D_2, ..., D_m$.

$$IG(D, \text{split}) = \text{Impurity}(D) - \sum_{j=1}^{m} \frac{|D_j|}{|D|} \text{Impurity}(D_j)$$

где Impurity — выбранная мера нечистоты (Gini или Entropy), $|D|$ — количество объектов в узле $D$, $|D_j|$ — количество объектов в дочернем узле $D_j$. **Алгоритм выбирает признак и порог, которые дают максимальный прирост информации (Maximum Information Gain).**

## 2.3. C. Критерии Качества Разделения (Регрессия)

### Уменьшение Дисперсии (Variance Reduction)

В задачах регрессии целью является минимизация разброса (дисперсии) целевой переменной внутри узлов. Наиболее распространенные критерии:

- **Среднеквадратичная Ошибка (Mean Squared Error, MSE):** Вычисляется дисперсия целевой переменной $y$ для данных в узле $D$.

$$\text{MSE}(D) = \frac{1}{|D|} \sum_{i \in D} (y_i - \bar{y}_D)^2$$

  где $\bar{y}_D$ — среднее значение $y$ в узле $D$. Алгоритм выбирает разделение, которое максимально снижает суммарную взвешенную MSE в дочерних узлах по сравнению с MSE в родительском узле.
- **Средняя Абсолютная Ошибка (Mean Absolute Error, MAE):** Иногда используется как альтернатива MSE, менее чувствительная к выбросам.

Разделение выбирается так, чтобы максимально **уменьшить дисперсию** (Variance Reduction).

## 2.4. D. Критерии Остановки

### Когда Прекратить Разделение?

Рекурсивный процесс построения дерева останавливается для ветви, если выполняется одно из условий:

- Все объекты в текущем узле принадлежат одному классу (узел стал "чистым").
- Достигнута максимальная глубина дерева (`max_depth`).
- Количество объектов в узле стало меньше порога (`min_samples_split`).
- Количество объектов в будущем листовом узле меньше порога (`min_samples_leaf`).
- Дальнейшее разделение не приводит к существенному улучшению критерия качества (например, Information Gain меньше некоторого порога).

# 3 III. Проблема Переобучения Деревьев

### Склонность Деревьев к Переобучению

Деревья решений обладают высокой гибкостью и способны строить очень сложные границы решений. Без ограничений они могут продолжать делиться до тех пор, пока каждый лист не будет содержать минимальное количество образцов (в пределе — один).

### Последствия Неограниченного Роста

- **Запоминание данных:** Дерево идеально подстраивается под обучающую выборку, включая шум и аномалии. Это приводит к отличной производительности на данных, на которых оно обучалось.
- **Плохая обобщающая способность:** Сложная структура дерева плохо переносится на новые, ранее не виданные данные. Производительность на тестовой выборке оказывается значительно ниже.
- **Высокая Дисперсия (High Variance):** Модель становится очень чувствительной к небольшим изменениям в обучающих данных. Незначительно измененный набор данных может привести к построению совершенно другого дерева.

*Аналогия: Студент, который зазубрил ответы на конкретные вопросы из тренировочного набора, но не понял общие принципы. На экзамене с новыми, но похожими вопросами, он не сможет дать правильные ответы.*

# 4 IV. Регуляризация Деревьев Решений

## 4.1. A. Ограничение Роста (Pre-pruning)

### Контроль Сложности Во Время Построения

Pre-pruning заключается в установке ограничений на рост дерева *до или во время* его построения. Это достигается с помощью гиперпараметров:

- `max_depth`: **Максимальная глубина дерева.** Ограничивает длину самого длинного пути от корня до листа. *Аналогия: Ограничить общее число уточняющих вопросов.*
- `min_samples_split`: **Минимальное число объектов в узле для разделения.** Узел не будет разделяться, если содержит меньше объектов, чем указано. *Аналогия: Не делить группу, если в ней меньше N участников.*
- `min_samples_leaf`: **Минимальное число объектов в листовом узле.** Гарантирует, что каждый лист содержит не менее указанного числа объектов. Разделение узла возможно, только если оба дочерних узла будут удовлетворять этому требованию. *Аналогия: Финальное решение должно быть поддержано как минимум M примерами.*
- `max_features`: **Максимальное число признаков для поиска лучшего разделения.** На каждом шаге рассматривается только случайное подмножество признаков. Уменьшает дисперсию и корреляцию между деревьями в ансамблях.
- `min_impurity_decrease`: **Минимальное уменьшение нечистоты.** Узел будет разделен, только если это разделение уменьшает нечистоту на величину, большую или равную этому значению.

Оптимальные значения этих гиперпараметров обычно подбираются с помощью кросс-валидации.

## 4.2. B. Обрезка Ветвей (Post-pruning)

### Упрощение Дерева После Построения

Post-pruning (или просто pruning, "подрезка") применяется *после* того, как дерево полностью построено (часто до большой глубины или до выполнения минимальных критериев остановки). Идея состоит в удалении ("срезании") некоторых ветвей или узлов, которые считаются менее полезными или приводящими к переобучению.

### Cost Complexity Pruning (CCP)

Один из наиболее распространенных методов post-pruning — это **Pruning по минимальной стоимости-сложности (Cost Complexity Pruning, CCP)**.

- **Идея:** Метод вводит параметр сложности $\alpha \geq 0$. Для каждого $\alpha$ находится поддерево, которое минимизирует *стоимость-сложность*:

$$R_\alpha(T) = R(T) + \alpha|T|$$

  где $R(T)$ — суммарная ошибка (например, нечистота или MSE) на листьях дерева $T$, $|T|$ — количество листьев в дереве $T$.
- **Параметр `ccp_alpha`:** В библиотеках (например, scikit-learn) этот параметр контролирует процесс прунинга. $\alpha = 0$ соответствует полному дереву (без прунинга). Увеличение $\alpha$ приводит к созданию деревьев с меньшим числом листьев, т.е. к более сильной обрезке.
- **Процесс:** Обычно генерируется последовательность поддеревьев для разных значений $\alpha$, и оптимальное значение $\alpha$ (а следовательно, и оптимальное поддерево) выбирается с помощью кросс-валидации.

### Преимущество Post-pruning

Иногда post-pruning позволяет найти более сбалансированную модель, чем pre-pruning, так как оно оценивает "полезность" ветвей на уже построенном дереве, что может быть более информативно, чем остановка роста на основе локальных критериев.