**1. Основная Идея Деревьев Решений (Стр. 1)**

* **Назначение:** Деревья решений - это модели машинного обучения, используемые как для **классификации** (предсказание категории, например, вид животного), так и для **регрессии** (предсказание числового значения, например, цена).
* **Принцип Работы:** Они работают, последовательно задавая простые вопросы о признаках данных. Это похоже на игру "20 вопросов". Каждый вопрос помогает сузить круг возможных ответов.
* **Структура:** Эти вопросы и ответы организуются в иерархическую структуру, похожую на дерево:
  + **Узлы (Nodes):** Представляют собой "вопросы" или "тесты" относительно какого-то признака (например, "Есть перья?", "Признак X[1] <= 0.0596?"). Корневой узел (root) - самый верхний.
  + **Ребра (Edges):** Соединяют узлы и представляют собой ответы на вопросы ("Да"/"Нет", "Истина"/"Ложь"). Ведут к следующему вопросу или к финальному ответу.
  + **Листья (Leaves):** Терминальные узлы (конечные точки дерева), которые больше не задают вопросов, а содержат итоговый ответ (предсказанный класс или значение).

**2. Построение Деревьев Решений (Стр. 2-4)**

* **Цель Построения:** Найти последовательность вопросов (тестов), которая наилучшим образом разделяет данные по целевой переменной (классу или значению) и приводит к ответу максимально коротким путем.
* **Тесты:** В машинном обучении эти "вопросы" называются **тестами**.
  + Для бинарных признаков (как в примере с животными): "Признак = Да/Нет?".
  + Для непрерывных признаков (как в two\_moons): "Признак i <= значение a?" (например, x[1] <= 0.0596).
* **Рекурсивное Разбиение:** Процесс построения дерева - рекурсивный:
  + Начинаем со всего набора данных (в корневом узле).
  + **Выбор Лучшего Теста:** Алгоритм перебирает множество возможных тестов (разные признаки и разные пороговые значения для непрерывных признаков). Выбирается тот тест, который является **наиболее информативным**, т.е. лучше всего разделяет данные на группы, соответствующие разным классам (в классификации) или приводящие к более однородным значениям (в регрессии). *Цель - максимизировать "чистоту" получаемых дочерних узлов.*
  + **Разделение Данных:** Данные делятся на две (или более, но обычно две) группы в соответствии с выбранным тестом. Каждая группа попадает в свой дочерний узел.
  + **Повторение:** Процесс выбора лучшего теста и разделения повторяется рекурсивно для каждого дочернего узла, пока не будет выполнено условие остановки.
* **Условие Остановки (без ограничений):** По умолчанию, дерево растет до тех пор, пока все точки данных в каждом листе не будут принадлежать одному классу (в классификации) или иметь очень близкие значения (в регрессии). Такой лист называется **чистым (pure)**.

**Чистые листы = переобучение**

**5. Контроль Сложности (Предотвращение Переобучения) (Стр. 5, 15)**

* **Цель:** Ограничить сложность дерева, чтобы оно лучше обобщало данные.
* **Стратегии:**
  + **Предварительная обрезка (Pre-pruning):** Остановить рост дерева раньше, до того как листья станут чистыми. scikit-learn реализует именно этот подход. Критерии остановки (гиперпараметры):
    - max\_depth: Максимальная глубина дерева (максимальное число последовательных вопросов).
    - max\_leaf\_nodes: Максимальное количество листьев.
    - min\_samples\_leaf: Минимальное количество образцов данных, которое должно быть в листе. Если при разделении узел-потомок будет содержать меньше образцов, разделение не производится.
  + **Пост-обрезка (Post-pruning):** Построить полное (переобученное) дерево, а затем удалить или объединить малоинформативные узлы/ветви. (В тексте упомянуто, но не используется в scikit-learn).

**6. Анализ Деревьев Решений (Стр. 6-11)**

* **Визуализация:** Деревья (особенно неглубокие) легко визуализировать (export\_graphviz, graphviz, pydotplus). Это помогает понять, как модель принимает решения. В узлах визуализации обычно показывают:
  + Тест (условие разделения).
  + samples: Количество образцов, дошедших до этого узла.
  + value: Распределение образцов по классам в этом узле.
  + class: Мажоритарный класс в этом узле.
* **Важность Признаков (Feature Importance):**
  + Деревья позволяют оценить, насколько каждый признак важен для принятия решений во всем дереве. Это число от 0 до 1 (feature\_importances\_).
  + 0 означает, что признак не использовался. 1 означает, что признак идеально предсказывает результат (редко). Сумма важностей всех признаков равна 1.
  + **Ограничение:** Важность признака всегда положительна. Она **не говорит**, *как* именно признак влияет на результат (например, увеличивает или уменьшает вероятность класса 1), а только то, что он *был полезен* для разделения. Взаимосвязь может быть немонотонной (Рис. 8, 9).

**7. Особенности Деревьев Регрессии (Стр. 12-14)**

* **Сходство:** Принцип построения и анализа очень похож на деревья классификации.
* **Ключевое Ограничение - Экстраполяция:** Деревья регрессии (DecisionTreeRegressor) **не могут экстраполировать**, то есть предсказывать значения вне диапазона целевой переменной, который был в обучающих данных. Если дереву подать на вход признак, значение которого выходит за рамки обучающего диапазона, оно просто предскажет значение из ближайшего листа (обычно это среднее значение точек на границе обучающего диапазона).

**1. Ансамбли (Ensembles) - Общая Идея (Стр. 1)**

* **Определение:** Ансамбли — это методы машинного обучения, которые **объединяют несколько моделей** (часто одного типа) для получения **более мощной и точной итоговой модели**, чем любая из составляющих моделей по отдельности.
* **Ключевые Модели:** В тексте выделяются два очень эффективных и широко используемых ансамблевых метода:
  + **Случайный лес (Random Forest)**
  + **Градиентный бустинг деревьев решений (Gradient Boosting)**
* **Базовые Строительные Блоки:** Оба этих метода используют **деревья решений** в качестве основы (строительных блоков).

**2. Случайный Лес (Random Forest) (Стр. 1-5)**

* **Мотивация:** Основной недостаток отдельных деревьев решений — их склонность к **переобучению**. Случайный лес — это способ борьбы с этой проблемой.
* **Основная Идея:**
  + Строится **множество (ансамбль) деревьев решений**.
  + Каждое дерево в ансамбле строится так, чтобы оно **немного отличалось** от других. Это достигается внесением **случайности** в процесс построения каждого дерева.
  + Предполагается, что каждое отдельное дерево может хорошо прогнозировать, но при этом немного переобучено на *своей* части данных.
  + **Усреднение предсказаний** от множества таких разных деревьев позволяет **уменьшить переобучение** и получить более робастную и точную модель.
* **Внесение Случайности (Как сделать деревья разными):**
  + **Бутстреп-выборка (Bootstrap Sample / Bagging):**
    - Для построения *каждого* дерева из леса используется не весь исходный обучающий набор данных, а **бутстреп-выборка** из него.
    - Бутстреп-выборка формируется путем случайного выбора n\_samples (размер исходного набора) примеров из исходного набора **с возвращением**.
    - Это означает, что некоторые примеры из исходных данных попадут в выборку для одного дерева несколько раз, а другие (примерно 1/3) не попадут вовсе. Каждое дерево обучается на немного разном подмножестве данных.
  + **Случайный Выбор Признаков для Разбиения (Feature Subsampling):**
    - При построении *каждого узла* (точки принятия решения) внутри одного дерева, алгоритм ищет лучший признак для разделения данных не среди *всех* доступных признаков, а только среди их **случайно выбранного подмножества**.
    - Количество признаков в этом подмножестве контролируется параметром max\_features.
    - Это приводит к тому, что даже на одних и тех же данных (бутстреп-выборке) разные деревья будут использовать разные признаки для разделения в узлах, что еще больше увеличивает их разнообразие.
* **Построение Случайного Леса:**
  + Определить количество деревьев (n\_estimators).
  + Для каждого дерева (от 1 до n\_estimators):
    - Создать бутстреп-выборку из обучающих данных.
    - Построить дерево решений на этой выборке, при этом в каждом узле выбирать лучший признак для разбиения из случайного подмножества признаков (размером max\_features).
* **Прогнозирование с Помощью Случайного Леса:**
  + **Регрессия:** Получить прогнозы от *каждого* дерева в лесу и **усреднить** их.
  + **Классификация:** Использовать **"мягкое голосование"**. Каждое дерево выдает вероятности принадлежности к каждому классу. Эти вероятности **усредняются** по всем деревьям. Предсказывается тот класс, у которого итоговая средняя вероятность максимальна.
* **Анализ Случайного Леса:**
  + Отдельные деревья в лесу могут иметь сложные границы решений, но итоговая граница леса (полученная усреднением) обычно более гладкая и обобщающая (Рис. 1 на стр. 3).
  + **Важность признаков:** Рассчитывается путем агрегирования (обычно усреднения) важности признаков по всем деревьям в лесу. Считается **более надежной**, чем важность, полученная от одного дерева. Случайный лес часто присваивает ненулевую важность большему числу признаков, чем одно дерево (Рис. 2 на стр. 5).
* **Параметры и Настройка (Стр. 1, 2, 6):**
  + n\_estimators: Количество деревьев. Больше = лучше (до определенного предела), но требует больше времени и памяти. Основной параметр для настройки.
  + max\_features: Количество признаков, рассматриваемых в каждом узле. Контролирует случайность. Уменьшение может снизить переобучение. Значения по умолчанию часто работают хорошо (sqrt(n\_features) для классификации, n\_features для регрессии - *согласно тексту*).
  + Параметры предварительной обрезки деревьев (max\_depth, max\_leaf\_nodes и т.д.): Могут применяться к отдельным деревьям, но часто менее важны, чем для одиночного дерева, так как усреднение само по себе борется с переобучением.
  + random\_state: Важен для воспроизводимости результатов, так как алгоритм использует случайность.
  + n\_jobs: Позволяет распараллелить построение деревьев на несколько ядер процессора (-1 = использовать все ядра), ускоряя обучение.
* **Преимущества и Недостатки (Стр. 5, 6):**
  + **Преимущества:** Один из самых эффективных методов "из коробки", высокая точность, устойчив к переобучению (по сравнению с одним деревом), не требует масштабирования признаков, хорошо работает с разными типами признаков.
  + **Недостатки:** Менее интерпретируем, чем одно дерево (сложно понять логику сотен деревьев), требует больше памяти и времени на обучение/прогнозирование, может плохо работать на очень разреженных данных высокой размерности (например, текст).

**3. Градиентный Бустинг Деревьев Регрессии (Gradient Boosting) (Стр. 6-9)**

* **Основная Идея:** Строит ансамбль деревьев **последовательно**. Каждое следующее дерево **пытается исправить ошибки**, допущенные предыдущими деревьями ансамбля. Фокусируется на тех примерах, на которых предыдущие модели ошибались.
* **Отличие от Случайного Леса:**
  + **Последовательное построение:** Деревья строятся одно за другим (нельзя распараллелить по деревьям, как в RF).
  + **Меньше случайности (по умолчанию):** Обычно не использует бутстреп и случайный выбор признаков (хотя такие вариации существуют), стремится строить деревья детерминированно.
  + **Слабые ученики:** Часто использует очень **неглубокие деревья** (например, глубина 1-5), которые сами по себе являются слабыми моделями. Сила ансамбля достигается их последовательным комбинированием.
* **Параметры и Настройка (Стр. 7, 9, 10):**
  + n\_estimators: Количество деревьев. **В отличие от RF, увеличение n\_estimators без подстройки других параметров может привести к переобучению.**
  + learning\_rate (скорость обучения): Контролирует, насколько сильно каждое новое дерево влияет на итоговую модель (насколько сильно оно "исправляет" ошибки). Меньшее значение learning\_rate требует больше деревьев (n\_estimators) для достижения той же точности, но часто приводит к лучшей обобщающей способности. **Ключевой параметр, тесно связанный с n\_estimators.**
  + max\_depth (или max\_leaf\_nodes): Контролирует сложность каждого отдельного дерева. **Обычно устанавливается очень низким** (1-5 уровней), так как используются слабые ученики. Сильная предварительная обрезка — стандартная практика.
* **Преимущества и Недостатки (Стр. 7, 9):**
  + **Преимущества:** Один из самых мощных и точных методов машинного обучения, часто показывает лучшие результаты в соревнованиях.
  + **Недостатки:** Более **чувствителен к настройке параметров** (n\_estimators, learning\_rate, max\_depth), может требовать больше времени на подбор гиперпараметров. Обучение может быть дольше, чем у RF (из-за последовательности). Склонен к переобучению при неправильной настройке. Как и RF, плохо работает на разреженных данных высокой размерности.
* **Когда использовать (Стр. 9):**
  + Начать можно со Случайного Леса (более устойчив, меньше настроек).
  + Если требуется максимальная точность и есть время на подбор параметров, попробовать Градиентный Бустинг.
  + Упоминается пакет xgboost как популярная, быстрая и эффективная реализация градиентного бустинга.

В целом, оба метода — мощные ансамбли на основе деревьев, но Случайный Лес проще в использовании и настройке, тогда как Градиентный Бустинг может дать более высокую точность, но требует более тщательного подбора гиперпараметров.