Шпаргалка по Random forest / Концепции

Cheatsheet (XeLaTeX)

Краткий справочник

April 4, 2025

Contents

Случайный Лес: Введение

Случайный Лес (Random Forest, RF) — это ансамблевый метод машинного обучения, который строит множество деревьев решений во время обучения и выводит класс, который является модой классов (классификация) или средним предсказанием (регрессия) отдельных деревьев. Это один из самых популярных и эффективных "из коробки" алгоритмов.
Аналогия: Представь, что тебе нужно принять важное решение. Вместо того чтобы спросить одного эксперта (одно дерево), ты собираешь **комитет разных экспертов** (много деревьев), каждый из которых смотрит на проблему немного под своим углом, а затем принимаешь решение на основе их коллективного мнения. Случайный лес делает то же самое, но с деревьями решений.

1 Идея Бэггинга (Bagging)

Bagging

Бэггинг — это основной принцип, лежащий в основе Случайного Леса. Он состоит из двух шагов:

- **Bootstrap (Бутстрэп):** Создается множество (N) подвыборок из исходного обучающего датасета. Каждая подвыборка формируется путем случайного выбора объектов **с возвращением**. Это означает, что некоторые объекты могут попасть в одну подвыборку несколько раз, а некоторые — ни разу. Размер каждой подвыборки обычно равен размеру исходного датасета.
- **Aggregating (Агрегация):** На каждой подвыборке независимо обучается своя модель (в случае RF — дерево решений). Затем предсказания всех N моделей усредняются (для регрессии) или определяется самый популярный класс (для классификации — голосование большинством).

Цель бэггинга: Снизить **дисперсию (variance)** модели. Индивидуальные деревья могут сильно переобучаться (высокая дисперсия), но усреднение их предсказаний сглаживает ошибки и делает итоговую модель более устойчивой.

Объекты, не попавшие в конкретную бутстрэп-выборку ($\approx 37\%$), называются **Out-of-Bag (OOB)** и могут использоваться для оценки качества модели (OOB-оценка) без необходимости отдельной валидационной выборки.

2 Случайный Выбор Признаков (Feature Subsampling)

Дополнительная Случайность

В отличие от простого бэггинга деревьев, Случайный Лес вносит **дополнительный элемент случайности** при построении каждого дерева:

- При поиске лучшего разбиения (split) в каждом узле дерева, алгоритм рассматривает не все доступные признаки, а только их **случайное подмножество** (размер подмножества, `max_features`, является гиперпараметром).
- Для задачи классификации обычно берут $\sqrt{p}$ признаков, для регрессии — $p/3$, где p — общее число признаков.

Зачем это нужно? Это делается для **декорреляции** деревьев. Если бы все деревья видели все признаки, и был бы один очень сильный признак, большинство деревьев использовали бы его для первого разбиения. В результате деревья были бы очень похожи (скоррелированы), и усреднение не дало бы такого сильного эффекта снижения дисперсии. Случайный выбор признаков заставляет деревья быть более разнообразными.
Аналогия: Возвращаясь к комитету экспертов. Чтобы они не пришли к одному и тому же выводу, опираясь на самый очевидный факт, ты просишь каждого эксперта при анализе сосредоточиться только на **случайном наборе аспектов** проблемы. Это побуждает их исследовать разные стороны вопроса.

3 Как Уменьшает Дисперсию

Борьба с Переобучением через Усреднение

Ключевая сила Случайного Леса — в его способности значительно **уменьшать дисперсию** по сравнению с одним деревом решений, не сильно увеличивая (или даже немного уменьшая) **смещение (bias)**.

- **Одно дерево решений:** Имеет низкое смещение (может хорошо подогнаться под обучающие данные), но высокую дисперсию (сильно меняется при небольшом изменении данных, легко переобучается).
- **Случайный Лес:**
  - **Бэггинг (усреднение):** Усреднение предсказаний N моделей, ошибки которых не полностью скоррелированы, приводит к снижению общей дисперсии ансамбля. Чем больше деревьев (N), тем ниже дисперсия (до определенного предела).
  - **Случайный выбор признаков (декорреляция):** Уменьшает корреляцию между деревьями, что делает усреднение еще более эффективным для снижения дисперсии.

В итоге, RF получает модель, которая все еще достаточно гибкая (относительно низкое смещение, унаследованное от деревьев), но гораздо более стабильная и устойчивая к переобучению (значительно сниженная дисперсия). Это классический пример улучшения модели через управление **компромиссом смещения-дисперсии (Bias-Variance Tradeoff)**.
Аналогия "Мудрость Толпы": Один человек может сильно ошибаться в оценке (высокая дисперсия), но если усреднить оценки большой группы людей (где ошибки случайны и не связаны), итоговая оценка будет гораздо ближе к истине (низкая дисперсия).

4 Важность Признаков (Feature Importance)

Зачем Оценивать Важность?

Хотя Случайный Лес — это ансамбль, что усложняет прямую интерпретацию, он предоставляет методы для оценки вклада каждого признака в итоговый результат. Это помогает:

- Понять, какие данные действительно влияют на модель.
- Упростить модель через отбор признаков (Feature Selection).
- Получить инсайты о предметной области.

Существует два основных подхода:

4.1 1. Mean Decrease in Impurity (MDI) / Gini Importance

Идея: Вклад в Чистоту Узлов

Этот метод оценивает важность признака на основе того, насколько сильно его использование для разделений в деревьях **уменьшает нечистоту (Impurity)** узлов (например, Gini Impurity для классификации или MSE для регрессии). Признак считается важным, если он часто выбирается для разделения и эти разделения значительно "очищают" данные. Расчет происходит **на обучающей выборке** во время построения леса.
Как считается (детально):

1. Обучаем Random Forest.
2. Для **каждого дерева** в лесу:
    - Для **каждого внутреннего узла**, где произошло разделение по признаку F:
    - Рассчитываем **уменьшение нечистоты (Information Gain или Variance Reduction)** в этом узле: $\Delta Impurity_{node} = Impurity(parent) - WeightedImpurity(children)$.
    - Умножаем это значение на долю объектов, прошедших через узел, относительно всех объектов (N_{node}/N_{total}), чтобы получить взвешенное уменьшение нечистоты для этого узла.
3. Для **каждого признака** F:
    - Суммируем взвешенные уменьшения нечистоты ($\sum N_{node} \times \Delta Impurity_{node}$) по **всем узлам всех деревьев**, где признак F использовался для разделения. Это дает "общую важность" $TotalImportance(F)$.
4. **Нормализация:** Общую важность каждого признака делят на сумму важностей всех признаков: $Importance(F) = \frac{TotalImportance(F)}{\sum_j TotalImportance(F_j)}$. В итоге, сумма всех важностей равна 1.

Формула (концептуально):

$$Importance_{MDI}(F) \propto \sum_{\text{trees}} \sum_{\substack{\text{nodes split} \\ \text{on } F}} N_{\text{node}} \cdot \Delta Impurity_{\text{node}}$$

Плюсы:

- **Быстро** считается (информация доступна сразу после обучения).
- Обычно предоставляется по умолчанию в библиотеках (например, `feature_importances_` в scikit-learn).

Минусы:

- Склонен **завышать важность** числовых признаков и категориальных признаков с большим количеством уникальных значений (высокой кардинальностью).
- Может давать **неадекватные результаты для скоррелированных признаков** (важность может "делиться" между ними или присваиваться только одному).
- Показывает, насколько признак был *полезен для построения деревьев* на обучающих данных, но не обязательно, насколько он важен для *предсказаний* на новых данных.

4.2 2. Mean Decrease in Accuracy (MDA) / Permutation Importance

Идея: Влияние "Поломки" Признака на Качество

Этот метод оценивает важность признака, измеряя, насколько **ухудшится качество предсказания** модели (например, Accuracy, F1, R^2, MSE), если "сломать" связь между этим признаком и целевой переменной путем случайного перемешивания его значений. Расчет происходит **на отложенной (не обучающей!) выборке**.
Как считается (детально):

1. Обучаем Random Forest.
2. Выбираем **отложенную выборку** (Out-of-Bag, валидационную или тестовую).
3. Рассчитываем **базовую метрику качества** $Score_{base}$ модели на этой выборке.
4. Для **каждого признака** F:
    - Создаем копию отложенной выборки.
    - В этой копии **случайно перемешиваем значения** только в столбце признака F. Остальные столбцы остаются без изменений.
    - Делаем предсказания модели на этой **модифицированной** выборке.
    - Рассчитываем метрику качества $Score_{permuted}(F)$ на предсказаниях для перемешанной выборки.
    - **Важность признака** F = $Score_{base} - Score_{permuted}(F)$.
5. (Опционально, для стабильности) Повторяем шаг 4 несколько раз с разными случайными перемешиваниями для каждого признака и усредняем полученные значения важности.

Формула (концептуально):

$$Importance_{Permutation}(F) = Score_{base} - \mathbb{E}[Score_{permuted}(F)]$$

где $\mathbb{E}[\cdot]$ означает ожидаемое значение по разным перемешиваниям.
Плюсы:

- Более **надежен**, чем MDI, особенно при наличии скоррелированных признаков (хотя интерпретация требует осторожности).
- Напрямую измеряет влияние признака на **предсказательную способность** модели на новых данных.
- Идея метода **модель-агностична** (можно применять к любой модели, не только RF).

Минусы:

- **Вычислительно затратен** (требует многократных предсказаний модели).
- Результат может зависеть от конкретной отложенной выборки и случайности перемешивания (рекомендуется усреднять по нескольким запускам).
- Интерпретация при **сильно скоррелированных признаках** сложна: удаление одного может не сильно влиять на метрику, если модель использует его "заменитель". Может занизить важность обоих.

4.3 Сравнение MDI и Permutation Importance

Ключевые Различия (Частый Вопрос на Собеседованиях)

Характеристика	MDI (Gini Importance)	Per
Что измеряет?	Насколько признак **использовался** для уменьшения нечистоты узлов при **обучении**.	Нас **пре**
На каких данных?	Обучающая выборка	Отл
Скорость	Быстро	Ме
Надежность	Менее надежен, предвзят к типу признаков	Бол сил
Скоррел. признаки	Может "делить", завышать/занижать важность	Мо
Модель-агностичность	Специфичен для деревьев	Иде
Основное Применение	Быстрый анализ, оценка по умолчанию	На

Вывод: Permutation Importance обычно считается более надежным показателем реальной важности признака для **производительности** модели.

Что Спрашивают на Собеседованиях

Часто спрашивают разницу между MDI и Permutation Importance. Важно понимать:

- MDI измеряет, насколько признак **использовался** деревьями при построении (на основе обучающей выборки).
- Permutation Importance измеряет, насколько признак **влияет** на итоговое качество предсказания модели (на основе отложенной выборки). Permutation Importance обычно считается более надежным показателем реальной важности.

5 Ключевые Гиперпараметры

Основные параметры для настройки Случайного Леса:

- **`n_estimators`:** Количество деревьев в лесу. Чем больше, тем лучше (до некоторого плато), но дольше обучение. Обычно выбирают достаточно большим (100, 500, 1000+).
- **`max_features`:** Количество признаков, рассматриваемых при поиске лучшего сплита в каждом узле. Ключевой параметр для контроля корреляции деревьев и борьбы с переобучением. Значения по умолчанию ($\sqrt{p}$ / $p/3$) часто работают хорошо, но стоит подбирать.
- **Параметры деревьев:** Гиперпараметры базовых деревьев решений также влияют на лес (например, `max_depth`, `min_samples_split`, `min_samples_leaf`). Часто оставляют деревья достаточно глубокими в RF, полагаясь на усреднение для борьбы с переобучением, но иногда их ограничение тоже помогает.

6 Сравнение с Конкурентами

RF vs. Одно Дерево Решений

- **Плюсы RF:**
  - Значительно **меньше переобучается**, более **устойчив** (низкая дисперсия).
  - Обычно **выше точность** и обобщающая способность.
- **Минусы RF:**
  - **Менее интерпретируем** ("черный ящик" по сравнению с одним деревом).
  - Требует **больше ресурсов** для обучения и предсказания (память и время).

RF vs. Линейные Модели (Логистическая/Линейная Регрессия)

- **Плюсы RF:**
  - Легко улавливает **нелинейные зависимости** и **взаимодействия** между признаками без необходимости их явного добавления (как в линейных моделях).
  - **Не требует масштабирования** признаков (деревьям не важен масштаб).
  - Менее чувствителен к **выбросам** (решающие правила деревьев устойчивы).
  - Хорошо работает "из коробки" с минимальной настройкой.
- **Минусы RF:**
  - **Менее интерпретируем**, чем линейные модели (где можно смотреть на веса).
  - Может быть **медленнее** в обучении и предсказании на очень больших данных.
  - Плохо **экстраполирует** за пределы диапазона значений признаков, виденных в обучении (предсказание ограничено значениями в "листьях"). Линейные модели могут экстраполировать.
  - Может требовать **больше памяти**.