Шпаргалка по предобработке данных и feature engineering / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

April 2, 2025

Contents

1 Обработка пропусков (Missing Values)

Почему пропуски - это проблема?

Многие ML-алгоритмы не могут работать с пропущенными значениями (NaN, None). Пропуски могут исказить результаты анализа и снизить производительность модели.

- **Причины пропусков:** Ошибки ввода, сбои сенсоров, отказ пользователя отвечать, данные недоступны.

Стратегии обработки пропусков

Выбор стратегии зависит от количества пропусков, типа данных и специфики задачи. (Указанные проценты - это эвристики, а не строгие правила).

1. **Удаление (Deletion):**
   - **Удаление строк (Listwise deletion):** Удаляем весь объект (строку), если в нем есть хотя бы один пропуск. *Когда?* Мало пропусков (< 5-10%), данные пропущены случайно (**MCAR** - Missing Completely At Random, т.е. сам факт пропуска не зависит ни от других признаков, ни от самого пропущенного значения), большой датасет. *Минус:* Потеря данных.
   - **Удаление столбцов (Dropping features):** Удаляем весь признак (столбец), если в нем слишком много пропусков (> 50-70%). *Когда?* Признак не очень важен, или пропусков слишком много для заполнения. *Минус:* Потеря потенциально полезной информации.
2. **Заполнение (Imputation):** Заменяем пропуски оценочными значениями.
   - **Простые методы:**
     - **Среднее (Mean):** Для числовых признаков без сильных выбросов. Чувствительно к выбросам.
     - **Медиана (Median):** Для числовых признаков с выбросами. Более робастно.
     - **Мода (Mode):** Для категориальных признаков.
   - **Более сложные методы (кратко):**
     - Заполнение с помощью **k-ближайших соседей (k-NN Imputer):** Использует значения соседей для оценки пропуска.
     - Заполнение с помощью **моделей (Regression Imputation):** Предсказывает пропуск с помощью регрессии на основе других признаков.
     - Создание **индикаторного признака:** Добавляем новый бинарный столбец, указывающий, было ли значение пропущено (1 - да, 0 - нет), а сам пропуск заполняем (например, нулем или средним). Позволяет модели учесть сам факт пропуска, **так как иногда сам факт отсутствия данных несет полезную информацию**.

   Плюс: Сохраняет данные. *Минус:* Может внести смещение (bias) в данные.

Аналогия: Детектив и недостающие улики

Представьте, что вы детектив, расследующий дело.

- **Удаление строки:** Если по свидетелю совсем нет информации (много пропусков), вы можете решить его не учитывать (удалить строку). Риск: потеряете ключевого свидетеля.
- **Удаление столбца:** Если какой-то тип улик (например, отпечатки пальцев) не удалось собрать почти нигде на месте преступления (много пропусков в столбце), вы можете перестать на него полагаться (удалить столбец). Риск: упустите важный тип улик.
- **Заполнение (Imputation):** Если у вас нет точного времени события, вы можете предположить его на основе других фактов: среднее время похожих преступлений (**mean**), наиболее вероятное время (**mode**), или предсказать его с помощью сложной модели (**k-NN/Regression**). Риск: ваше предположение может быть неверным.
- **Индикаторный признак:** Вы отмечаете в деле, что точное время *неизвестно* (индикатор=1), но записываете предполагаемое время (например, медиану). Теперь факт неизвестности времени - это тоже улика!

2 Кодирование категориальных признаков

Зачем кодировать?

Большинство ML-моделей понимают только числа. Категориальные признаки (например, "цвет": "красный", "синий"; "город": "Москва", "СПб") нужно преобразовать в числовой формат.

One-Hot Encoding (OHE) vs Label Encoding

Два самых популярных метода:

1. One-Hot Encoding (OHE):

- **Как работает:** Создает новый бинарный (0 или 1) столбец для каждой уникальной категории признака. В каждой строке только один из этих новых столбцов равен 1, остальные — 0.
- **Когда использовать:** Для **номинальных** признаков (где нет естественного порядка категорий, например, "цвет", "страна"). Подходит для большинства моделей, особенно линейных, SVM, нейронных сетей.
- **Плюсы:** Не вносит ложного порядка.
- **Минусы:** Сильно увеличивает количество признаков ("проклятие размерности"), если категорий много. Может привести к мультиколлинеарности (часто удаляют один из OHE-столбцов — 'drop='first'').

2. Label Encoding (LE):

- **Как работает:** Присваивает каждой уникальной категории целое число (0, 1, 2, ...).
- **Когда использовать:**
  - Для **порядковых** признаков (где есть естественный порядок, например, "размер": "S" < "M" < "L" -> 0, 1, 2).
  - Иногда для **древовидных моделей** (Решающее дерево, Случайный лес, Градиентный бустинг), так как они могут разбивать признак по значениям (например, '< 1.5'). *Но будьте осторожны: OHE часто все равно дает лучший результат даже для деревьев.*
- **Плюсы:** Не увеличивает количество признаков.
- **Минусы:** Вносит **ложный порядок** для номинальных признаков (например, "Москва" = 0, "СПб" = 1, "Казань" = 2 подразумевает, что "СПб" "больше" "Москвы", что неверно). Это может запутать модели, чувствительные к значениям (линейные, SVM).

Итог (Частый вопрос на собеседовании):

- Есть порядок (ordinal)? → **Label Encoding**.
- Нет порядка (nominal)?
  - Модель - дерево? → Можно попробовать **Label Encoding** (но OHE часто лучше).
  - Модель - НЕ дерево (линейная, SVM, NN)? → **One-Hot Encoding**.
  - Очень много категорий? → Рассмотреть другие методы (Target Encoding, Embedding) или Feature Engineering.

Аналогия: Маркировка одежды на складе

- **Label Encoding (для порядковых):** Размеры футболок "S", "M", "L". Можно просто пронумеровать полки 0, 1, 2. Порядок сохранен, все понятно.
- **Label Encoding (для номинальных - ПЛОХО):** Цвета футболок "Красный", "Синий", "Зеленый". Если пронумеровать полки 0, 1, 2, то это подразумевает, что "Синий" (1) чем-то "больше" "Красного" (0), что бессмысленно. Алгоритм может сделать неверные выводы.
- **One-Hot Encoding (для номинальных):** Для цветов "Красный", "Синий", "Зеленый" создаем три отдельные секции на складе. Если футболка красная, кладем ее в секцию "Красный" (1), а в секциях "Синий" и "Зеленый" ее нет (0). Порядка нет, но признаков (секций) стало больше.

3 Масштабирование признаков (Feature Scaling)

Зачем масштабировать?

Если числовые признаки имеют разные диапазоны значений (например, возраст [0-100] и зарплата [10k-1M]), то модели, использующие **меры расстояния** (KNN, SVM, K-Means) или **градиентный спуск** (Линейная/Логистическая регрессия, Нейронные сети), могут придать неоправданно больший "вес" признакам с бОльшими значениями. Масштабирование приводит все признаки к сопоставимому диапазону.

Нормализация vs Стандартизация

Два основных метода:
1. Нормализация (Normalization / Min-Max Scaling):

- **Как работает:** Сжимает данные в диапазон [0, 1] (или [-1, 1]).
- **Формула:**

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Когда использовать:** Когда нужен строгий диапазон [0, 1] (например, для обработки изображений, где пиксели 0-255). Если данные не имеют гауссова распределения. Когда известно, что выбросов мало или они обработаны.
- **Минусы:** Чувствительна к **выбросам**, так как X_{min} и X_{max} могут сильно сдвинуться из-за одного аномального значения.

2. Стандартизация (Standardization / Z-score Scaling):

- **Как работает:** Преобразует данные так, чтобы они имели среднее значение 0 и стандартное отклонение 1.
- **Формула:**

$$X_{std} = \frac{X - \mu}{\sigma}$$

где μ - среднее, σ - стандартное отклонение.

- **Когда использовать:** **Чаще всего рекомендуется**. Особенно для алгоритмов, которые предполагают нормальное (гауссово) распределение данных (хотя работает и для других распределений). Менее чувствительна к выбросам, чем нормализация. Подходит для PCA, линейных моделей, SVM, KNN.
- **Минусы:** Не приводит данные к строгому диапазону (значения могут быть > 1 или < -1).

Когда масштабирование НЕ обязательно (или менее критично):

- **Древовидные модели** (Решающее Дерево, Случайный Лес, Градиентный Бустинг): Они смотрят на пороги разбиения для каждого признака независимо, поэтому масштаб не так важен. *(Но иногда масштабирование все же может немного улучшить сходимость/стабильность даже в деревьях, хотя и не так критично, как для других моделей).*

Аналогия: Сравнение оценок из разных школ

Ученик А получил 90 баллов из 100 (шкала 0-100). Ученик Б получил 4 балла из 5 (шкала 0-5). Кто лучше?

- **Без масштабирования:** 90 > 4, кажется, что А лучше. Но это некорректно.
- **Нормализация (в шкалу [0, 1]):** Ученик А: (90 - 0) / (100 - 0) = 0.9 Ученик Б: (4 - 0) / (5 - 0) = 0.8 Теперь видно, что результат А немного выше.
- **Стандартизация (относительно среднего по школе):** Допустим, средний балл в школе А был 70 (std=10), а в школе Б - 3 (std=0.5). Ученик А: (90 - 70) / 10 = +2 сигмы (значительно выше среднего по своей школе). Ученик Б: (4 - 3) / 0.5 = +2 сигмы (тоже значительно выше среднего по своей школе). Стандартизация показала, что оба ученика выступили одинаково хорошо *относительно своей группы*.

4 Feature Engineering (Очень Кратко)

Идея: Создание лучших признаков для модели

Это процесс использования знаний о данных и предметной области для создания признаков, которые делают работу ML-моделей более эффективной. Включает в себя как **создание** новых признаков, так и **отбор** или **извлечение** существующих. **Часто это самый важный шаг для улучшения модели!**

Основные направления Feature Engineering

1. Отбор признаков (Feature Selection):

- **Идея:** Выбрать подмножество *наиболее важных* исходных признаков.
- **Цель:** Уменьшить сложность, ускорить обучение, бороться с переобучением, улучшить интерпретируемость.
- **Примеры методов (названия):** Фильтры (корреляция, Хи-квадрат, ANOVA F-value), Обертки (Recursive Feature Elimination - RFE), Встроенные (Lasso-регрессия L1, Feature Importance из деревьев).

2. Извлечение признаков (Feature Extraction):

- **Идея:** Создать *новые* признаки путем комбинации или трансформации исходных, часто с понижением размерности. Новые признаки обычно не имеют прямого физического смысла исходных.
- **Цель:** Уменьшить размерность данных, сохранив при этом максимум полезной информации, бороться с шумом.
- **Примеры методов (названия):** Метод главных компонент (**PCA** - Principal Component Analysis), Линейный дискриминантный анализ (**LDA**), разложение матриц (SVD).

3. Создание признаков (Feature Creation):

- **Идея:** Генерация новых признаков из существующих на основе знаний о предметной области или анализа данных.
- **Цель:** Добавить в модель полезную информацию, которой не было в исходных признаках в явном виде.
- **Примеры:**
  - Полиномиальные признаки (x_1^2, $x_1 \times x_2$) - часто для линейных моделей.
  - Взаимодействия признаков (x_1/x_2, $x_1 + x_2$, разница между датами).
  - Признаки из дат (день недели, месяц, час, является ли день праздником/выходным).
  - Агрегированные признаки (например, для клиента: средняя сумма покупки за месяц, количество покупок за неделю).
  - Биннинг (группировка) непрерывных признаков (например, возраст → возрастные группы: "молодой", "средний", "пожилой").

Аналогия: Подготовка к походу

- **Feature Selection (Отбор):** У вас много снаряжения (признаков). Вы выбираете только самое необходимое для конкретного похода (самые важные признаки): нож, палатку, спальник. Отбрасываете ненужное (менее важные признаки): утюг, фен. Вы используете *исходные* предметы.
- **Feature Extraction (Извлечение):** У вас есть много разных продуктов (исходные признаки). Вместо того чтобы нести их все, вы готовите из них концентрированную, калорийную походную еду (новые признаки), например, сублиматы. Эта еда содержит энергию из исходных продуктов, но сама по себе является чем-то новым и занимает меньше места (понижение размерности).
- **Feature Creation (Создание):** У вас есть орехи и сухофрукты (исходные признаки). Вы смешиваете их и делаете питательный энергетический батончик (новый признак). Батончик - это комбинация исходных, но он удобнее и дает энергию по-другому. Или вы смотрите на карту (данные) и понимаете, что нужно пересечь реку (знание о данных) - вы создаете признак "нужен ли плот" (новый признак).