

# Машинное обучение, ФКН ВШЭ

## Семинар №4

### Предобработка данных

## 1 Пропущенные значения

В реальных задачах значения некоторых признаков у некоторых объектов отсутствуют. Это может происходить по разным причинам: ошибки при записи данных, отказ респондента отвечать на вопрос, невозможность описать конкретное свойство у конкретного объекта (в таблице с данными об автомобилях будут пропуски у электромобилей в графе «объём топливного бака»). Многие алгоритмы машинного обучения (в частности, линейная регрессия) не могут работать с пропущенными данными, поэтому эти пропуски необходимо заполнить.

Заполнять пропуски у объектов можно различными способами:

1. Константным уникальным значением — неудачный вариант для линейных методов (модель начнёт считать пропуск близким к некоторому другому значению выборки), но быстрый и популярный способ с другими алгоритмами в машинном обучении.
2. Средним арифметическим, медианой, модой — сохранение статистик выборки, но потеря информации о наличии пропуска в данных.
3. Предсказаниями другого алгоритма — затратно по времени (однако всё равно не приносит новой информации в датасет, хотя и может положительно сказаться на общем качестве).

Заметим, что в некоторых случаях наличие пропуска в данных несёт определённую информацию об объекте (например, отказ в ответе на вопрос о доходах клиента банка), поэтому полезно добавлять новые признаки — индикаторы пропусков. Иногда признаки содержат слишком много пропусков и их выгоднее удалить.

## 2 Выбросы

На практике могут встречаться объекты, сильно отличающиеся от остальных. Их называют выбросами. Отличия могут выражаться как в значениях признаков, так и в целевой величине. Причины бывают различными: ошибки в заполнении данных (добавили лишний ноль), «исключительность» отдельных объектов (низкие цены на дома могут быть связаны с попыткой обхода налогов, а не с их характеристиками).

Выбросы могут сильно сказываться на решении — например, квадратичная функция ошибок «реагирует» на выбросы и линейная регрессия с таким функционалом ошибки отклоняется в их сторону, в отличие от модели, оптимизирующей среднюю абсолютную ошибку (рис. 1).

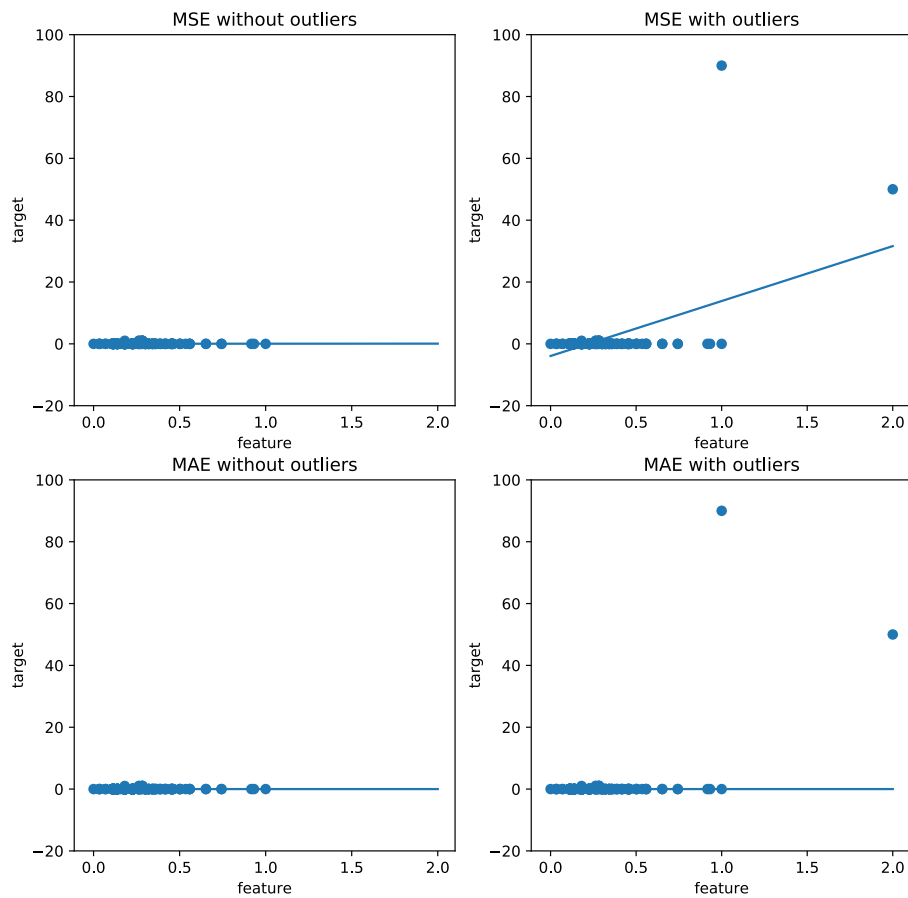


Рис. 1. Влияние выбросов на обученную линейную регрессию для MSE и MAE в качестве функции потерь.

Искать выбросы можно следующим образом. Выбросы в признаках можно обнаружить, исследуя распределение признаков и в особенности хвосты распределений. Выбросы в целевой величине можно искать, считая ошибку предсказания модели на объектах обучающей выборки (вспомогательная модель не должна наблюдать при обучении проверяемый объект). Если ошибка велика (алгоритм с уверенностью предсказывает отрицательный класс, хотя метка у объекта положительная), то объект можно считать выбросом (если, конечно, дело не в плохой модели). Объекты-выбросы чаще всего не корректируют, а удаляют из выборки.

Заметим, что не всегда необходимо удалять объекты-выбросы из выборки. В одном [конкурсе](#) помог следующий подход: оставить выбросы, чтобы не изменилось среднее предсказание алгоритма, при этом качество модели считать только по «нормальным» объектам, чтобы исключить шум от объектов-выбросов.

Как уже было сказано выше, некоторые функции потерь чувствительнее относятся к выбросам, поэтому в таких ситуациях имеет смысл использовать более устойчивые функции потерь для обучения моделей.

## 3 Обработка категориальных признаков

Часто в данных встречаются категориальные по смыслу признаки. Если они представлены в виде чисел, то, подавая напрямую в модель, мы задаём порядок над этими категориями, что обычно неправильно. Например, если красный цвет кодировался как «1», зелёный — «2», а синий — «3», то модель будет считать зелёный цвет находящимся ровно между красным и синим. Если же категориальный признак представлен в выборке не в виде чисел, то мы и вовсе не можем использовать его для обучения модели. Изучим способы кодирования категориальных признаков.

### §3.1 Label encoding

В простом случае, если категориальный признак представлен в виде нечисловых данных, можно построить обратимое отображение для каждого уникального значения в некоторое число. Это позволит использовать признак для обучения модели. Например, зелёному цвету будет соответствовать «1», красному — «2» и так далее.

У этого подхода две основные проблемы: задание порядка над категориями и работа с неизвестными в процессе обучения значениями категориального признака (нужно не забывать обрабатывать такой случай отдельно).

### §3.2 One-hot encoding

Другой способ кодирования заключается в добавлении признаков-индикаторов категориальных значений. Например, появятся новые бинарные признаки: «красный цвет», «зелёный цвет» и так далее. В этом случае решается проблема с заданием порядка над категориями и новыми значениями категориального признака (у такого объекта просто будут «нули» по всем признакам индикаторам).

Проблема этого подхода в увеличении количества признаков (а это затрачиваемая память и скорость обучения модели) пропорционально количеству категорий. Эти разреженные признаки допускают хранение в виде разреженных матриц, но только некоторые алгоритмы умеют работать с ними. Также можно для экономии памяти не создавать признаки для редко встречающихся категорий.

### §3.3 Mean target encoding

Более сложный метод кодирования категориальных признаков — кодирование средним значением целевой переменной. Идея в том, что алгоритму для предсказания цены необходимо знать не конкретный цвет автомобиля, а то, как этот цвет сказывается на цене. Поэтому заменим каждую категорию на среднее значение целевой переменной по всем объектам этой категории. Для бинарной классификации новый признак будет выглядеть следующим образом:

$$g_j(x, X) = \frac{\sum_{i=1}^{\ell} [f_j(x) = f_j(x_i)] [y_i = +1]}{\sum_{i=1}^{\ell} [f_j(x) = f_j(x_i)]},$$

где  $f_j(x_i)$  –  $j$ -й признак  $i$ -го объекта,  $y_i$  – класс  $i$ -го объекта.

Но в таком случае для редких категорий получим некорректные средние значения целевой переменной. Например, в выборке было только три золотистых автомобиля, которые оказались старыми и дешёвыми. Из-за этого наш алгоритм начнёт считать золотистый цвет дешёвым. Для исправления этой проблемы будем регуляризовать средним значением целевой переменной по всем категориям так, чтобы у редких категорий значение было близко к среднему по всей выборке, а для популярных к среднему значению по категории. Формально для задачи бинарной классификации это выражается так:

$$g_j(x, X) = \frac{\sum_{i=1}^{\ell} [f_j(x) = f_j(x_i)] [y_i = +1] + \frac{C}{l} \sum_{i=1}^{\ell} [y_i = +1]}{\sum_{i=1}^{\ell} [f_j(x) = f_j(x_i)] + C},$$

где  $C$  – коэффициент, отвечающий за баланс между средним значением по категории и глобальным средним значением.

Однако если мы вычислим значения  $g_j(x, X)$  по всей выборке, то столкнёмся с переобучением, так как мы внесли информацию о целевой переменной в признаки (новый признак слабая, но модель, предсказывающая целевое значение). Поэтому вычисление таких признаков следует производить по фолдам, то есть вычислять средние значения на основе одних фолдов для заполнения на другом фолде (аналогично процессу кросс валидации). Если же ещё планируется оценка качества модели с помощью кросс валидации по фолдам, то придётся применить «двойную кросс валидацию» для подсчёта признаков. Этот подход заключается в кодировании категориальных признаков по фолдам внутри глобальных фолдов, по которым оценивается качество модели.

Разберём этот процесс. Иллюстрация на рис. 2. Представим, что хотим посчитать качество модели на 3-м фолде. Для этого:

1. Разбиваем все фолды, кроме 3-го, на другие фолды. Количество внутренних фолдов может не совпадать с количеством внешних (на иллюстрации их также 3).
2. Для каждого из внутренних фолдов считаем значение mean target признаков на основе средних значений целевой переменной по фолдам, исключая текущий. Для 3-го внешнего фолда вычисляем как среднее вычисленных признаков по каждому из внутренних фолдов.
3. Обучаем модель на всех фолдах, кроме 3-го, делаем предсказание на 3-м и считаем на нём качество.

Существуют альтернативы кодированию категориальных признаков по фолдам. Во-первых, кодирование по порядку (среднее значение считается по всем объектам, расположенным выше текущего), порядок при этом может задаваться случайно или соответствовать временной шкале. Такой способ применяется в библиотеке CatBoost. Во-вторых, добавление шума к закодированным значениям.

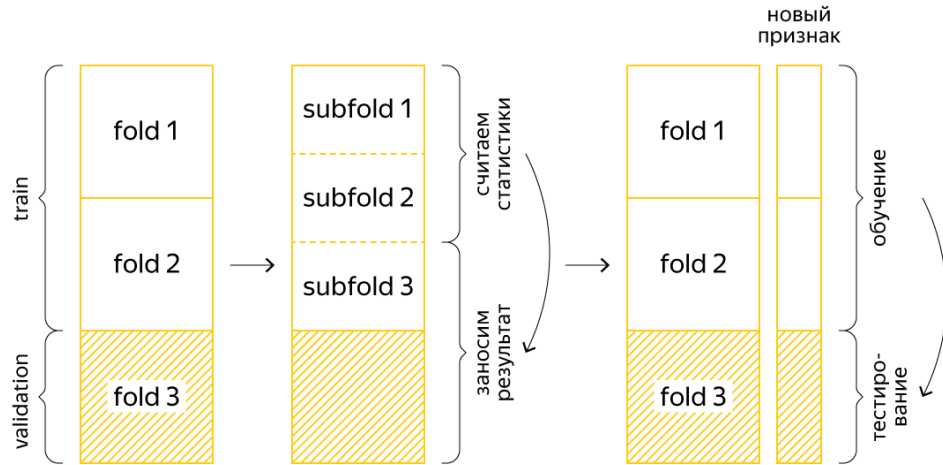


Рис. 2. Кросс валидация при кодировании средним значением

## 4 Извлечение признаков из текстов

При решении некоторых задач мы сталкиваемся с тем, что объекты выборки целиком или частично описываются в виде текстов (под текстами имеем в виду строки, содержащие как минимум пару слов, иначе такой признак можно рассматривать как категориальный признак). Поэтому стоит задача представления текста в виде векторов чисел фиксированной длины.

### §4.1 Bag-of-words

Простой способ заключается в подсчёте, сколько раз встретилось каждое слово в тексте. Получаем вектор длиной в количество уникальных слов, встречающихся во всех объектах выборки. В таком векторе много нулей, поэтому его удобнее хранить в разреженном виде. Такой способ представления текстов называют мешком слов.

### §4.2 TF-IDF

Очевидно, что не все слова полезны в задаче прогнозирования. Например, мало информации несут слова, встречающиеся во всех текстах. Это могут быть как стоп-слова, так и слова, свойственные всем текстам выборки (в текстах про автомобили употребляется слово «автомобиль»). Эту проблему решает TF-IDF преобразование текста. Вычисляются две величины:

TD (Term Frequency) — количество вхождений слова в отношении к общему числу слов в тексте:

$$\text{tf}(t, d) = \frac{n_{td}}{\sum_{t \in d} n_{td}},$$

где  $n_{td}$  — количество вхождений слова  $t$  в текст  $d$ .

IDF (Inverse Document Frequency):

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|},$$

где  $|\{d \in D : t \in d\}|$  — количество текстов в коллекции, содержащих слово  $t$ .

Тогда для каждой пары (слово, текст)  $(t, d)$  вычислим величину:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D).$$

Это и будем значением нового признака (вместо количества каждого слова в тексте в случае мешка слов). Отметим, что значение  $\text{tf}(t, d)$  корректируется для часто встречающихся общеупотребимых слов при помощи значения  $\text{idf}(t, D)$ .

### §4.3 *N*-граммы

На практике каждое слово в языке может иметь несколько смыслов, а мы в изученных выше подходах даже не учитываем их порядок. Чтобы передать модели больше информации, можно передавать не только отдельные слова, но и их словосочетания или *n*-граммы. *N*-граммы — последовательность из *n* подряд идущих слов текста. Последовательности накладываются друг на друга.

Например, из предложения “Ты же знаешь, что Максим строго проверяет работы” получатся следующие 3-граммы: “ты же знаешь”, “же знаешь что”, “знаешь что Максим”, “что Максим строго”, “Максим строго проверяет”, “строго проверяет работы”.

Каждая *n*-грамма для мешка слов считается отдельным словом. На практике часто берут *n*-граммы сразу разных размеров, (например, от 1 до 3). Заметим, что размер мешка слов растёт экспоненциально с ростом *n*.

### §4.4 Лемматизация и стемминг

Заметим, что одно и то же слово может встречаться в различных формах (особенно для русского языка), но описанные выше методы интерпретируют их как различные слова, что делает признаковое описание избыточным. Устранить эту проблему можно при помощи лемматизации и стемминга.

Стемминг — это процесс нахождения основы слова. В результате применения данной процедуры однокоренные слова, как правило, преобразуются к одинаковому виду. Например, вагон — вагон, вагонов — вагон и важная — важн, важно — важн.

Лемматизация — процесс приведения слова к его нормальной форме (лемме):

- для существительных — именительный падеж, единственное число;
- для прилагательных — именительный падеж, единственное число, мужской род;
- для глаголов, причастий, деепричастий — глагол в инфинитиве.

Лемматизация — процесс более сложный по сравнению со стеммингом. Стеммер просто «режет» слово до основы. Реализация лемматизаторов и стеммеров можно найти в различных библиотеках (nlTK, pymorphy).