## Подготовка данных для машинного обучения

## Вступление

Мы в разделе 2 нашего курса уже рассматривали частично подготовку данных. Иногда знания предметной области позволяют эксперту предложить преобразования данных, которые сделают зависимость между входными данными и предсказываемой величиной более очевидными. Эти преобразования могут быть простыми математическими функциями (например, возведение величины в квадрат, взятие квадратного корня, либо логарифма), либо достаточно сложными формулами (например, для предсказания поведения двигателя может быть полезно рассчитать его рабочий объём, применяя формулу объёма цилиндра и информацию о количестве цилиндров в этой модели двигателя).

Все техники создания производных данных на основе данных, имеющихся в нашей обучающей выборке, сохраняют свою актуальность. Всегда при подготовке данных для машинного обучения о них стоит помнить, и при необходимости применять.

Также стоит помнить о методиках заполнения пропусков в данных и удаления выбросов. Их тоже обязательно стоит применять к данным практически во всех случаях. Они универсальны.

Сегодня мы рассмотрим более узкоспециализированные шаги подготовки данных, которые позволят улучшить качество обучения моделей машинного обучения определённых типов.

## Требования к данным у различных моделей

У различных моделей машинного обучения могут быть специфические требования к данным. Чаще всего эти требования вызываются тем, что математика, лежащая в основе модели, включает в себя определённые предположения о структуре наших данных. Если данные нарушают эти предположения, модель работать не будет. Давайте посмотрим, какие модели какие ограничения накладывают.

### Требования к данным: линейная регрессия

Линейная регрессия по своей сути работает только с числовыми данными. Если у нас данные – категории, их придётся преобразовывать к числам. Способы преобразования данных в числа мы рассмотрим чуть позже.

Кроме этого, если некоторые столбцы наших данных функционально зависят от других столбцов (например, у нас 2 копии одного столбца, или столбец 17 это столбец 3, умноженный на 2), линейная регрессия тоже не сойдётся. В таких случаях придётся искать столбцы, между которыми сильная зависимость, и удалять их из обучающей выборки.

Также, линейная регрессия не способна справляться с пропусками. Прежде чем обучать модель линейной регрессии, обязательно нужно будет устранить все пропуски в данных обучающей выборки – либо заполнить их какими-то значениями, либо удалить записи с пропусками из выборки, если присвоить им значения не представляется возможным.

### Требования к данным: деревья принятия решений.

Деревья принятия решений, в отличие от линейной регрессии (и многих других алгоритмов) способны сразу, без преобразований работать с категориальными перемеными. Это значит, что преобразовывать категориальные переменные для дерева принятия решений не нужно.

Также, деревья принятия решений нечувствительны к масштабу числовых переменных. Переменная со значениями от 0 до 1 и переменная со значениями от 1 миллиона до 100 миллионов с точки зрения дерева принятия решений одинаково удобны.

Это делает деревья принятия решений (и основанные на них алгоритмы, такие как случайный лес и GBDT) очень удобными в использовании с данными, в которых много категориальных переменных и числовых переменных с существенно отличающимся масштабом.

Тем не менее, у деревьев принятия решений есть и недостатки. В частности, все основанные на них алгоритмы очень чувствительны к шумам (выбросам) в обучающей выборке. Поэтому, перед обучением такой модели стоит приложить дополнительные усилия к поиску и устранению из обучающей выборки всех шумов, ошибочных данных и аномалий. Если этого не сделать, качество модели существенно пострадает.

### Требования к данным: нейронные сети.

Нейроные сети могут работать только с числовыми данными.

В зависимости от выбора функции активации, предпочтительно, чтобы данные, с которыми работает нейронная сеть, находились в диапазоне от 0 до 1, либо от -1 до 1.

Если данные выходят за пределы этих диапазонов, придётся их масштабировать.

Кроме этого, нейронные сети не могут работать с категориальными переменными, и требуется значительное их преобразование (кодирование), для того, чтобы нейронные сети смогли их понимать.

Также, в случае, если мы работаем с данными за определенное время, и хотим делать прогноз на будущее на основе прошлого, нужно будет сделать множество преобразований данных, поскольку каждый прогноз нейронной сети (и подавляющего большинства других моделей машинного обучения) не зависит от данных предыдущих прогнозов.

## Кодирование категориальных переменных.

### One-hot кодирование

Целью one-hot является присвоение чисел категориальным переменным таким образоа, чтобы не внести в наши данные ложных, бессмысленных, отсутствующих в реальности зависимостей.

Например, если у вас есть категориальные переменные «Яблоко», «Апельсин», «Вишня», «Помидор» и вы присваиваете им числовые значения 0, 1, 2, 3, то эти числовые значения имеют интерпретацию «Вишня находится между помидором и яблоком, но ближе к помидору», потому что 2 находится между 0 и 3, но ближе к 3. Это неправда. Это вредная для нас неправда, потому что алгоритмы анализа данных (например, регрессии или что-то еще) могут уловить эти закономерности и делать на их основе предсказания.

Если вместо этого вы представляете «Яблоко», «Апельсин», «Вишня» и «Помидор» как кортежи из четырех чисел (1,0,0,0), (0,1,0,0), (0,0, 1,0) и (0,0,0,1), то у вас нет этой проблемы. Каждая координата имеет значение 0 или 1 и измеряет «Яблковость», «Апельсиновость», «Вишнёвость» или «Томатность» вашего фрукта. Это и есть one-hot кодирование.

В качестве примера предположим, что среднее яблоко весит 200 граммов, средний апельсин весит 150 граммов, средняя вишня 30 граммов и средний помидор 100 граммов. С «горячим» (one hot) кодированием (x1,x2,x3,x4) , этот средний вес является линейной функцией кодировки: 200x1+150x2+30x3+100x4 . Это то, что регрессия может выяснить на основе данных. При кодировке 0, 1, 2, 3 не существует удобной функции, которая бы выдала вам средний вес фрукта по его номеру.

### Dummy-кодирование

В то время как one hot кодирование использует N двоичных переменных для N категорий, dummy кодирование использует N-1 столбцов для представления N меток/категорий.

Все столбцы – те же, что в one hot кодировании, просто последней категории не выделяется столбец. Мы можем определить, что закодирована последняя категория, если в столбцах всех категорий стоят нули. (Выделено жирным шрифтом на слайде)

Зачем нужно dummy-кодирование: у горячего кодирования есть проблема, заключающаяся в том, что линейная функция не уникальна.

В частности, «горячее» кодирование может не работать с линейной регрессией и другими алгоритмами, которые предъявляют требования к независимости переменных.

### Частотное кодирование

Частотное кодирование - это метод, используемый для преобразования категориальных переменных в числовые значения путем представления каждой категории как доли встречаемости этой категории в наборе данных. Оно используется для обработки категориальных переменных со многими уровнями.

Например, допустим, у нас есть набор данных со столбцом под названием «product», который содержит следующие значения: «apple», «orange», «banana», «apple», «orange», «apple», «banana». Частотное кодирование заменит каждую категорию количеством раз, которое она появляется в наборе данных. Полученные данные могут выглядеть следующим образом: (слайд)

### Целевое кодирование:

Целевое кодирование - это метод, используемый для преобразования категориальных переменных в числовые значения путем представления каждой категории как среднего значения целевой переменной для этой категории. Этот метод используется, когда категориальная переменная имеет большое количество уровней, а также полезен в ситуациях, когда данные сильно несбалансированы.

Например, предположим, что у нас есть набор данных со столбцом под названием «product» и целевой переменной под названием «sales», которая содержит следующие значения:

|  |  |
| --- | --- |
| product | sales |
| apple | 100 |
| orange | 200 |
| banana | 50 |
| apple | 150 |
| orange | 300 |
| apple | 50 |
| banana | 20 |

Целевая кодировка заменила бы каждую категорию в столбце "product" средним значением столбца "sales" для этой категории. Результирующие данные могут выглядеть следующим образом:

|  |  |  |
| --- | --- | --- |
| product | sales | target\_encoded |
| apple | 100 | 83.333 |
| orange | 200 | 250.0 |
| banana | 50 | 35.0 |
| apple | 150 | 83.333 |
| orange | 300 | 250.0 |
| apple | 50 | 83.333 |
| banana | 20 | 35.0 |

Как вы можете видеть, исходный столбец "product" все еще присутствует в таблице, но был добавлен новый столбец "target\_encoded", который содержит целевые значения в кодировке для каждого продукта. Строки в столбце "target\_encoded" теперь содержат десятичные значения, представляющие среднее значение столбца "sales" для каждого продукта.

Масштабирование данных

Мы уже много раз упоминали, что для