Предобработка данных

Что такое данные?

В широком понимании данные представляют собой факты, текст, графики, картинки, звуки, аналоговые или цифровые видео-сегменты.
Данные могут быть получены в результате измерений, экспериментов, арифметических и логических операций.

Данные должны быть представлены в форме, пригодной для хранения, передачи и обработки.

Данные - это необработанный материал, предоставляемый поставщиками данных и используемый потребителями для формирования информации на основе данных.

Набор данных и их атрибутов

По горизонтали таблицы располагаются *атрибуты* объекта или его признаки. *По* вертикали таблицы - *объекты*.

Объект описывается как набор атрибутов.

Объект также известен как *запись*, случай, пример, строка таблицы и т.д.

Атрибут - свойство, характеризующее *объект*.

Например: цвет глаз, температура воды и т.д.

Атрибут также называют переменной, полем таблицы, измерением, характеристикой.

Переменная (variable) - свойство или характеристика, общая для всех изучаемых *объектов*, проявление которой может изменяться от *объекта* к *объекту*.

Значение (value) переменной является проявлением признака.

	Атрибуты				
Объекты	Код клиента	Возраст	Семейное положение	Доход	Класс
	1	18	Single	125	1
	2	22	Married	100	1
	3	30	Single	70	1
	4	32	Married	120	1
	5	24	Divorced	95	2
	6	25	Married	60	1
	7	32	Divorced	220	1
	8	19	Single	85	2
	9	22	Married	75	1
	10	40	Single	90	2

Все в науке о данных начинается с данных.

Предобработка данных является важнейшим этапом Data Mining, и если она не будет выполнена, то дальнейший анализ в большинстве случаев невозможен из-за того, что аналитические алгоритмы просто не смогут работать или результаты их работы будут некорректными. Иными словами, реализуется принцип **GIGO — garbage in, garbage out** (мусор на входе, мусор на выходе).

По оценкам специалистов 80% времени уходит на подготовку данных

Среди проблем, вызывающих снижение качества данных, можно выделить следующие:

- пропущенные значения;
- дубликаты;
- противоречия;
- аномальные значения и выбросы;
- шум;
- отсутствие полноты данных;
- нарушения целостности данных;
- некорректные форматы и представления данных;
- фиктивные значения;
- ошибки ввода данных;
- нарушения структуры.

https://wiki.loginom.ru/articles/data-quality.html

	0	1	2	3	4	5	6	7
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
5	5	116	74	0	0	25.6	0.201	30
6	3	78	50	32	88	31.0	0.248	26
7	10	115	0	0	0	35.3	0.134	29
8	2	197	70	45	543	30.5	0.158	53
9	8	125	96	0	0	0.0	0.232	54

Пропущенные значения

Реальные наборы данных могут иметь пропуски значений параметров.
Это может случиться из-за технических проблем, или если датасет собран из нескольких источников с разными наборами параметров; важно то, что в таблице присутствуют пустые ячейки

Примеры:

- Отзывы на кинопоиске

Случайность пропусков

Предполагается, что пропуски в матрице располагаются случайно

Примеры исключений:

- отказ респондентов отвечать на вопросы
- Абрахам Вальд и повреждения самолетов (2я мировая война)

Если их много — тренировка на таких данных сильно ухудшит качество модели, а то и окажется вовсе невозможной. **Многие алгоритмы машинного обучение** не только **требуют массив чисел (а не NaN или «missing»),** но и ожидают, что этот массив будет состоять из валидных данных, а в случае наличия пропущенных значений в массиве будет выбрасываться исключение.

Как быть?

1. Применение алгоритмов. **Отдельные алгоритмы умеют принимать во внимание и даже восстанавливать** пропущенные значения в данных. НО в таком случае нужно понимать, как именно алгоритм обойдётся с недостающими данными, иначе есть риск получить какие-нибудь артефакты и даже не узнать, какие именно.

2. Исключение и игнорирование строк с пропущенными значениями стало решением по умолчанию в некоторых популярных прикладных пакетах. НО такой метод применим только в том случае, когда малая часть объектов выборки имеет пропущенные значения. Преимуществом данного подхода является простота и невозможность испортить данные путем замены пропусков. В случае достаточно большого размера выборки метод может показывать хорошие результаты.

3. Замена пропусков зачастую и вполне обоснованно кажется более предпочтительным решением. Однако это не всегда так. Неудачный выбор метода заполнения пропусков может не только не улучшить, но и сильно ухудшить результаты.

Существует множество способов замены данных со своими их преимуществами и недостатками.

Методы замены пропусков на основе имеющейся информации часто объединяют в одну группу, называемую *Single-imputation methods*.

Замена данных средним/модой

В случае категориального признака все пропуски заменяются на наиболее часто встречающее значение, в случае количественного признака – на среднее значение по признаку. Данный метод позволяет учитывать имеющиеся данные.

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean()

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Плюсы:

- Просто и быстро.
- Хорошо работает на небольших наборах численных данных.

Минусы:

- Значения вычисляются независимо для каждого столбца, так что корреляции между параметрами не учитываются.
- Не работает с качественными переменными.
- Метод не особенно точный.

Пример

В тестовом примере 768 записей из них 376 содержат пропуски

Точность модели, рассчитанной алгоритмом LDA при исключении пропусков

0.78582892934

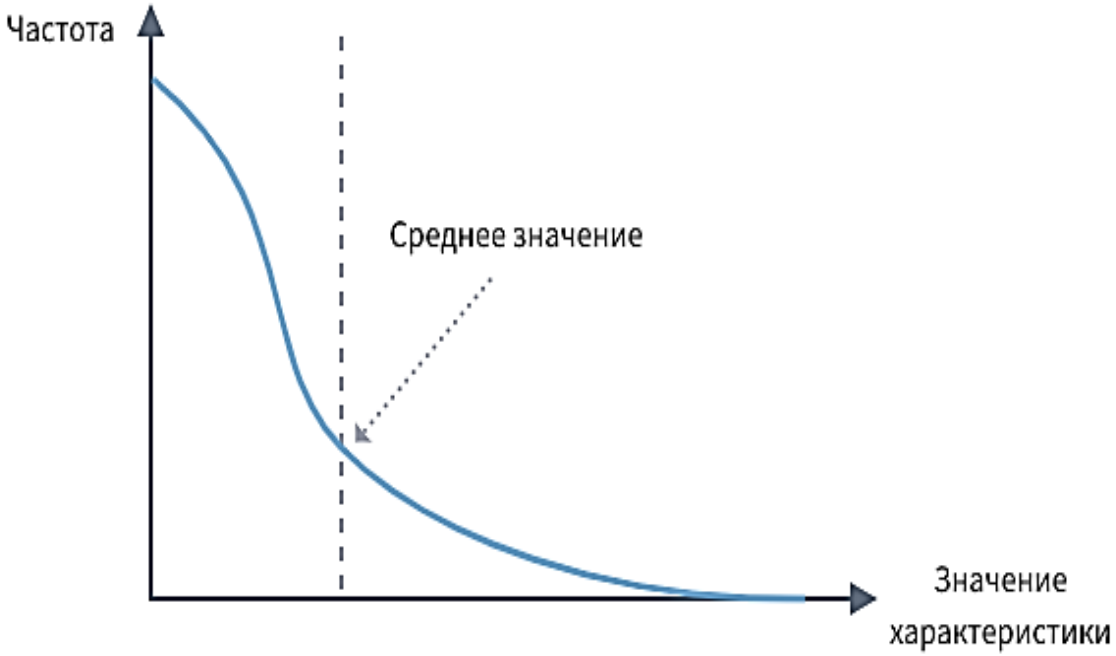
Точность модели, рассчитанной алгоритмом LDA при замене пропусков средним значением

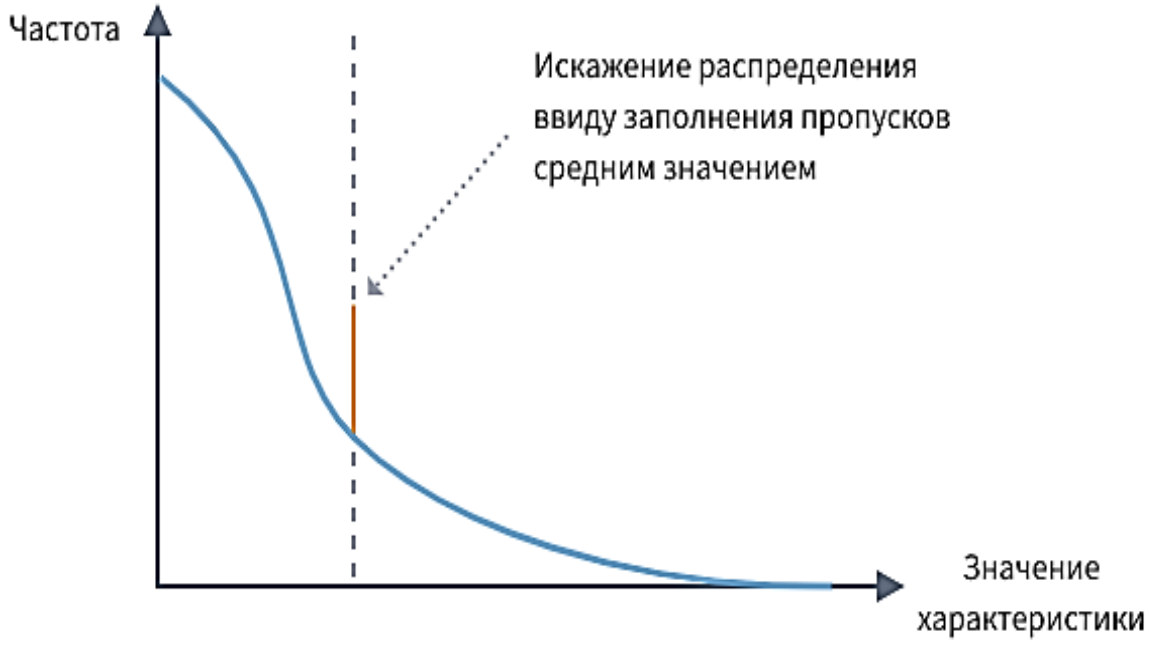
0.766927083333

В данном случае метод исключения пропущенных данных оказался предпочтительнее, чем замена пропусков средним значением.

При заполнении пропусков средним значением, модой, нулем или медианой свойственны одни и те же **недостатки**.

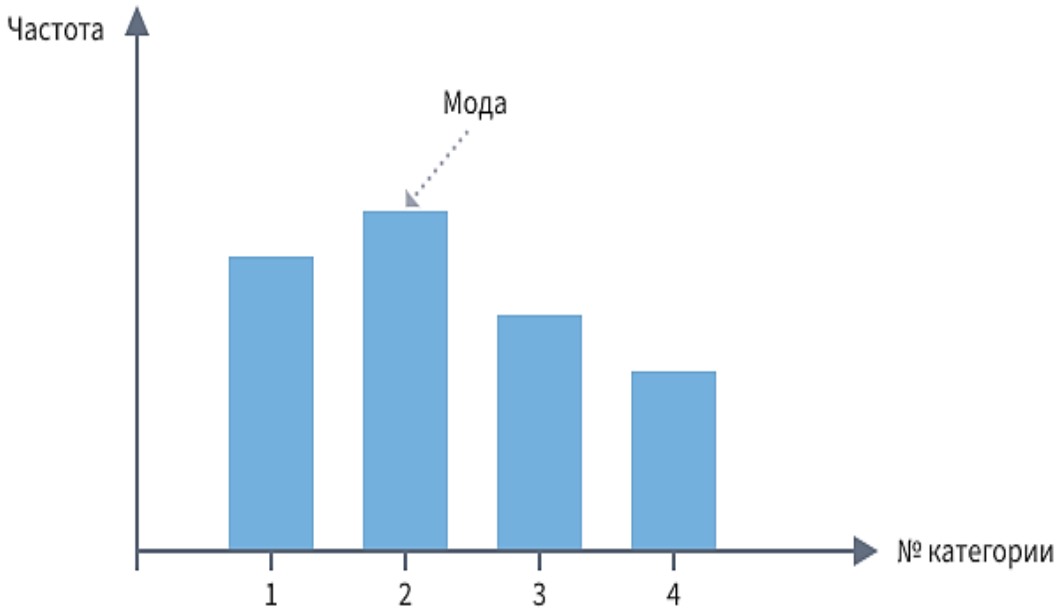
На рисунках: распределение значений непрерывной характеристики до заполнения пропусков **средним значением** и после него.

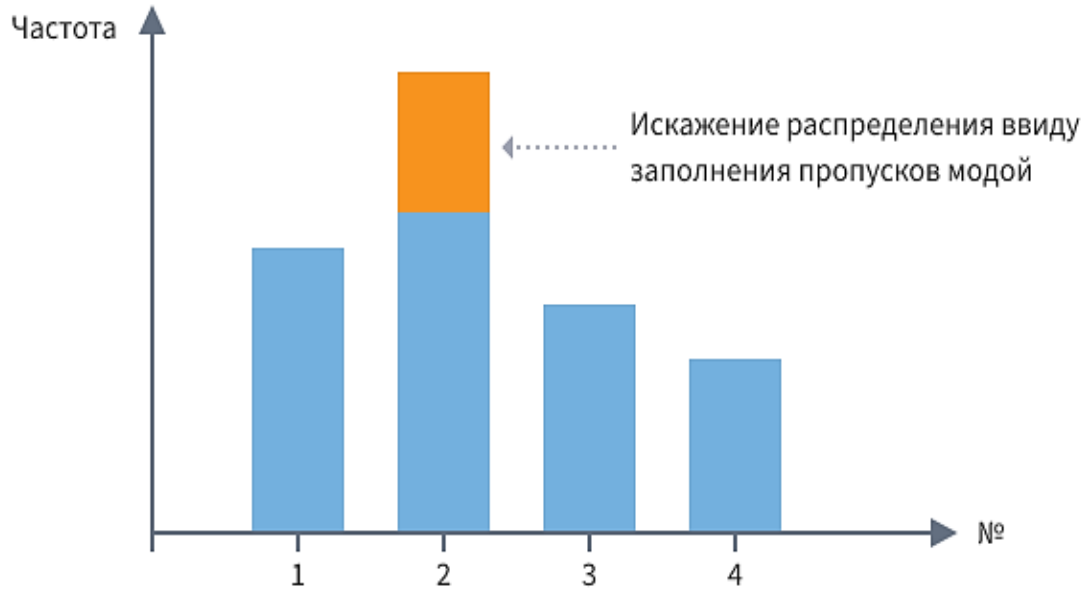




Данный метод приводит к существенному искажению распределения характеристики.
Это в итоге проявляется в искажении всех показателей, характеризующих свойства распределения (кроме среднего значения), заниженной корреляции и завышенной оценке стандартных отклонений.

В случае категориальной дискретной характеристики наиболее часто используется **заполнение модой.**
На рисунке 2 показано распределение категориальной характеристики до и после заполнения пропусков.





При заполнении пропусков категориальной характеристики модой проявляются те же недостатки, что и при заполнении пропусков непрерывной характеристики средним арифметическим (нулем, медианой и тому подобным).ы

Замена данных самым частым значением (модой) или константой

Замена самым часто встречающимся значением — ещё одна простая стратегия для компенсации пропущенных значений, не учитывающая корреляций между параметрами. Плюсы и минусы те же, что и в предыдущем пункте, но **этот метод предназначен для качественных переменных**.

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	0.0
1	9	0.0	9.0	0	7.0
2	19	17.0	0.0	9	0.0

Плюсы:

- Просто и быстро.
- Хорошо работает на небольших наборах численных данных.

Минусы:

- Значения вычисляются независимо для каждого столбца, так что корреляции между параметрами не учитываются.
- Не работает с качественными переменными.
- Метод не особенно точный.

Повторение результата последнего наблюдения

LOCF (Last observation carried forward) — повторение результата последнего наблюдения. Данный метод применяется, как правило, при заполнении пропусков во временных рядах, когда последующие значения априори сильно взаимосвязаны с предыдущими.

Когда применение LOCF обосновано.

Пример: если мы измеряем температуру воздуха в некоторой географической точке на открытом пространстве, причем измерения проводятся каждую минуту, то при нормальных условиях — если исключить природные катаклизмы — измеряемая величина априори не может резко (на 10–20 °C) измениться за столь короткий интервал времени между последующими измерениями. Следовательно, заполнение пропусков предшествующим известным значением в такой ситуации обоснованно.

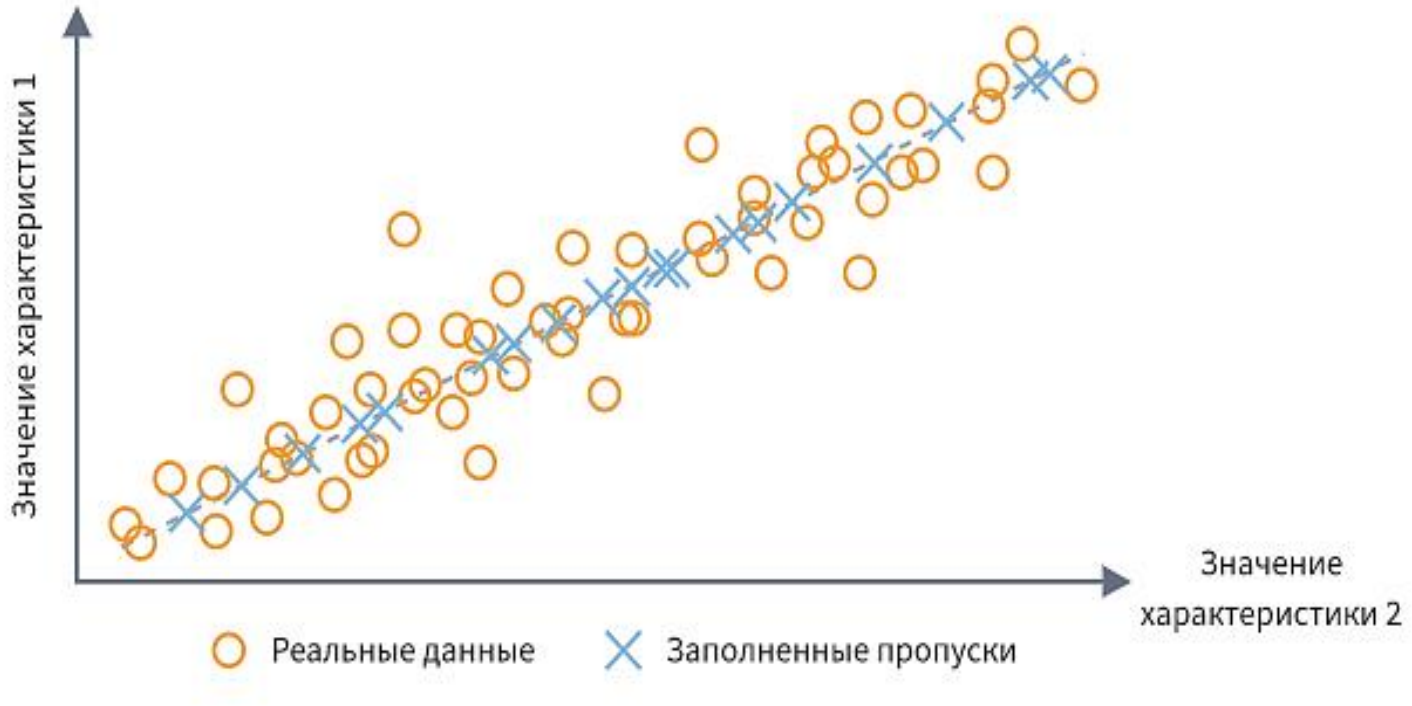
Хотя в описанной выше ситуации метод логичен и обоснован, он тоже может привести к существенным искажениям статистических свойств. Так, возможна ситуация, когда применение LOCF приведет к дублированию выброса (заполнению пропусков аномальным значением). Кроме того, если в данных много последовательно пропущенных значений, то гипотеза о небольших изменениях уже не выполняется и, как следствие, использование LOCF приводит к неправильным результатам.

Восстановление пропусков на основе регрессионных моделей

Данный метод заключается в том, что пропущенные значения заполняются с помощью модели линейной регрессии, построенной на известных значениях набора данных.

На рисунке показан пример результатов заполнения пропущенных значений характеристики 1 на основе известных значений характеристики 2.

Метод линейной регрессии позволяет получить правдоподобно заполненные данные.



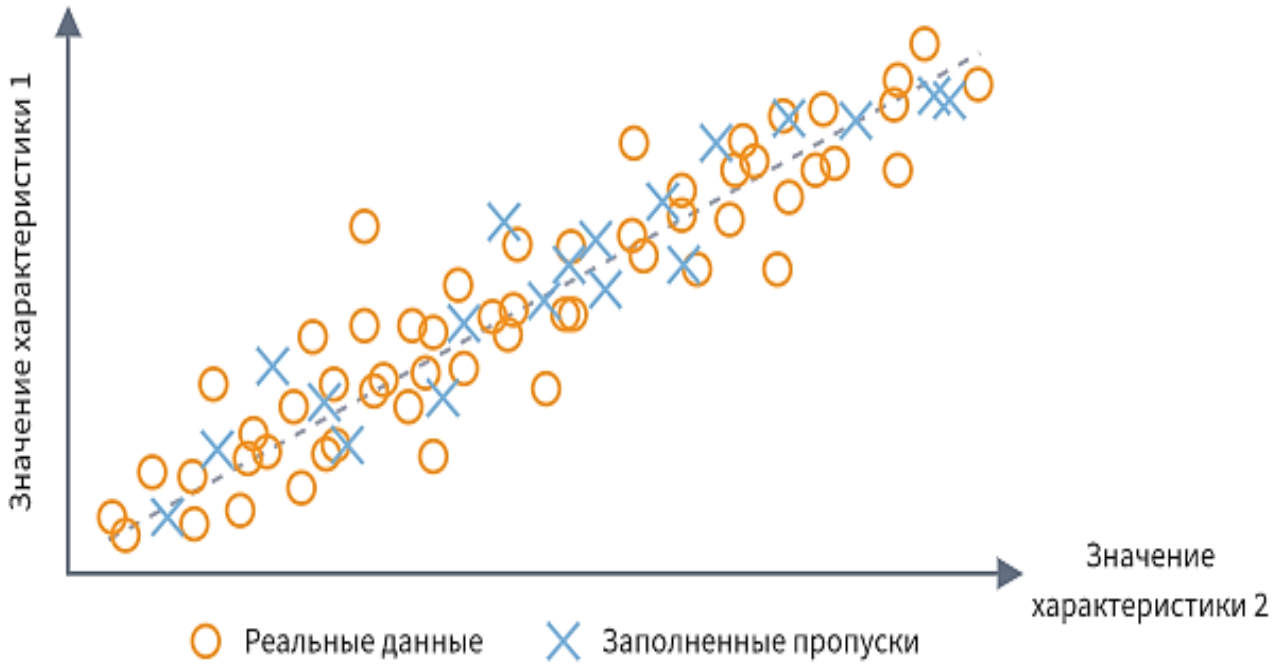
Однако реальным данным свойственен некоторый разброс значений, который при заполнении пропусков на основе линейной регрессии отсутствует. Как следствие, вариация значений характеристики становится меньше, а корреляция между характеристикой 2 и характеристикой 1 искусственно усиливается.

В результате данный метод заполнения пропусков становится тем хуже, чем выше вариация значений характеристики, пропуски в которой мы заполняем, и чем выше процент пропущенных строк.

Есть метод, решающий эту проблему: метод стохастической линейной регрессии.

На рисунке — **заполнение пропусков на основе стохастической линейной регрессии**

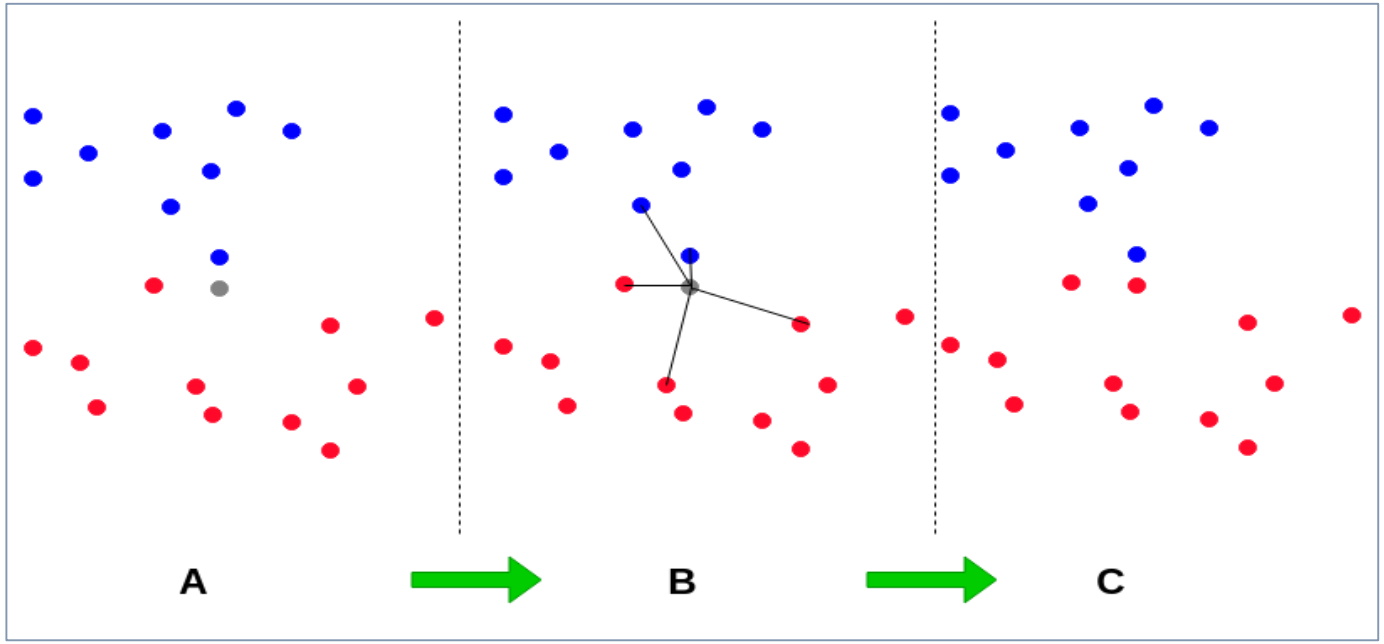
Модель стохастической линейной регрессии отражает не только линейную зависимость между характеристиками, но и отклонения от этой линейной зависимости. Этот метод обладает положительными свойствами заполнения пропусков на основе линейной регрессии и, кроме того, не так сильно искажает значения коэффициентов корреляции.



Заполнение пропусков с помощью стохастической линейной регрессии в общем случае приводит к наименьшим искажениям статистических свойств выборки.
А в случае, когда между характеристиками прослеживаются явно выраженные линейные зависимости, метод стохастической линейной регрессии нередко превосходит даже более сложные методы.

Замена данных с помощью метода k-NN

k-Nearest Neighbour (k ближайших соседей) — простой алгоритм классификации, который можно модифицировать для замены недостающих значений. Он использует сходство точек, чтобы предсказать недостающие значения на основании *k* ближайших точек, у которых это значение есть. Иными словами, выбирается *k* точек, которые больше всего похожи на рассматриваемую, и уже на их основании выбирается значение для пустой ячейки.



Плюсы:

- На некоторых датасетах может быть точнее среднего/медианы или константы.
- Учитывает корреляцию между параметрами.

Минусы:

- Вычислительно дороже, так как требует держать весь набор данных в памяти.
- Важно понимать, какая метрика дистанции используется для поиска соседей.
- Может потребоваться предварительная нормализация данных.
- Чувствителен к выбросам в данных.

Замена данных с помощью MICE (Multivariate Imputation by Chained Equations)

Этот подход основан на том, что замена каждого значения проводится много раз и дальнейший анализ многократно повторяется на разных наборах данных. Множественные замены, в отличие от однократных, позволяют понять, насколько надёжно или ненадёжно предложенное значение. Кроме того, MICE позволяет работать с переменными разных типов (например, двоичными и количественными), а также с различными артефактами в исходных данных.

Замена данных с помощью глубокого обучения

Глубокое обучение (нейросети) хорошо работает с дискретными и другими не-численными значениями.

Плюсы:

- Точнее других методов.
- Может работать с качественными параметрами.

Минусы

- Восстанавливает только один столбец.
- На больших наборах данных может быть вычислительно дорого.
- Нужно заранее решить, какие столбцы будут использоваться для предсказания недостающего значения.

https://ru.coursera.org/lecture/unsupervised-learning/obrabotka-propuskov-3mRlZ

Прочие этапы предобработки данных

Трансформация данных заключается в оптимизации их представлений и форматов с точки зрения решаемых задач и целей анализа. Трансформация не ставит целью изменить информационное содержание данных. Ее задача — представить эту информацию в таком виде, чтобы она могла быть использована наиболее эффективно.
Вообще, трансформация данных — очень широкое понятие, не имеющее четко очерченных границ. Иногда этот термин иногда распространяют на любые манипуляции с данными, независимо от их целей и методов. Однако в контексте анализа данных трансформация данных имеет вполне конкретные цели и задачи, а также использует достаточно стабильный набор методов. К основным из них относятся **нормализация, преобразование типов и форматов, сортировка, группировка, слияние и др**.

https://wiki.loginom.ru/articles/data-transformation.html

Квантование данных (Биннинг) — **это преобразование непрерывной переменной в категориальную переменную.** Например, если мы хотим применить условия к непрерывным столбцам, скажем, к столбцу «вес машины», мы можем создать новый категориальный столбец с помощью:
вес> 1500 и вес <2500 как «Легкий»
вес> 2500 и вес <3500 как «средний»
вес> 3500 и вес <4500 как "Heavy"
вес> 4500 как «очень тяжелый»

https://www.analyticsvidhya.com/blog/2020/09/pandas-speed-up-preprocessing/

Фильтрация данных

https://basegroup.ru/community/articles/data-filtration

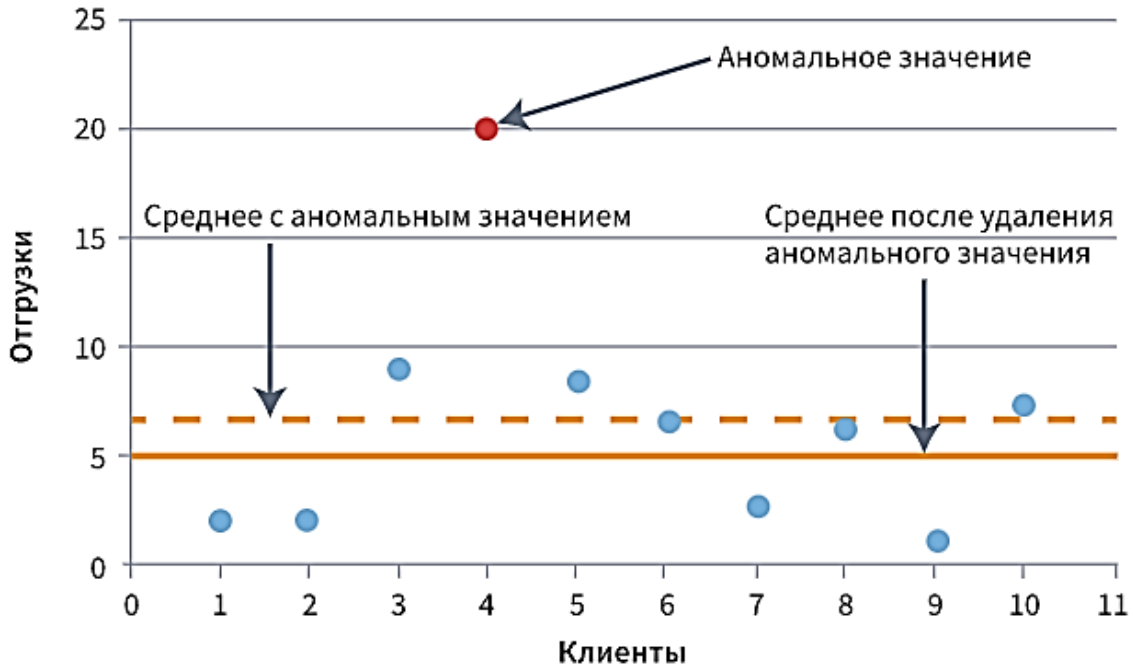
Прочие этапы предобработки данных

Аномальное значение (Outlier value)

Аномальными называются значения, которые не укладываются в общую модель поведения анализируемого процесса. Они сильно отличаются от окружающих данных и могут быть вызваны как ошибками измерений, так и некорректным вводом данных, или являться результатом их сильной изменчивости.

Аномальные значения необходимо подавить или удалить, поскольку они могут вызвать некорректную работу алгоритмов и привести к искажению результатов анализа данных.

Степень устойчивости алгоритма к наличию в данных аномальных значений называется **робастностью**.



Что делать?

- Оценить наличие
- Заменить/Исключить

Вопрос: как оценить данные на наличие выбросов

https://wiki.loginom.ru/articles/outlier.html

Как можно выявить пропущенные данные?

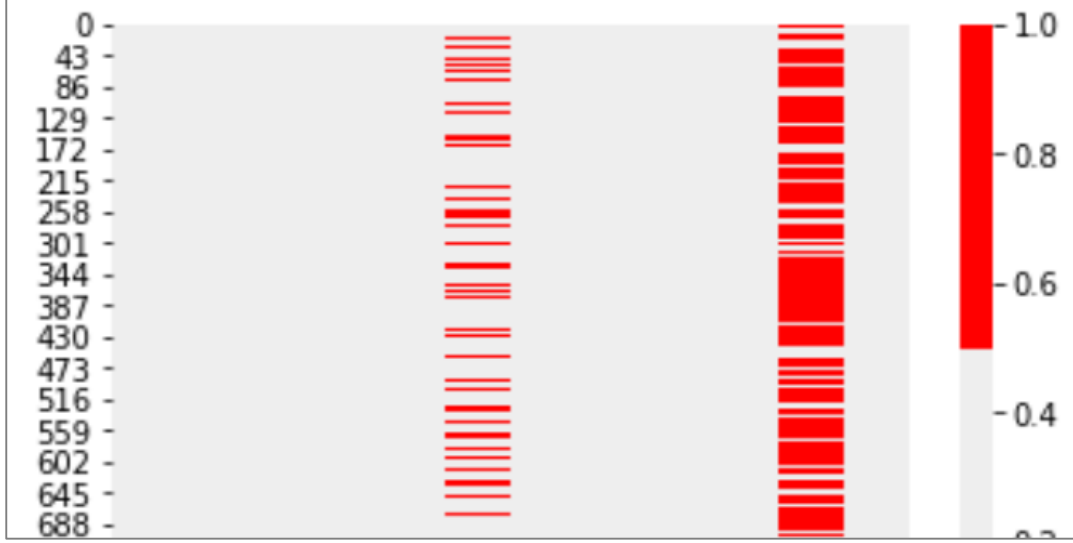
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
```

Визуализация пропущеных данных

Когда признаков в наборе не очень много, пропущенные значения можно визуализировать с помощью **тепловой карты**.

```
cols = data.columns[:12] # первые 30 колонок
# определяем цвета
# красный - пропущенные данные
colours = ['#eeeeee', '#ff0000']
sns.heatmap(data[cols].isnull(), cmap=sns.color_palette(colours))
```

```
<AxesSubplot:>
```



Как можно выявить пропущенные данные?

Функция **info()** выводит информацию о количестве ненулевых значений по столбцам

```
Ввод [56]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       768 non-null    int64
 1   1       763 non-null    float64
 2   2       733 non-null    float64
 3   3       541 non-null    float64
 4   4       394 non-null    float64
 5   5       757 non-null    float64
 6   6       768 non-null    float64
 7   7       768 non-null    int64
dtypes: float64(6), int64(2)
memory usage: 48.1 KB
```

Подсчет количества нулевых значений по столбцам

```
Ввод [48]: data[data.eq(0)].count()

Out[48]: 0    111
         1      5
         2     35
         3    227
         4    374
         5     11
         6      0
         7      0
         dtype: int64
```

Функции Pandas для работы с пропущенными данными

dropna(axis=0)	удаляет все строки с NaN значениями, при axis=1 - то же со столбцами
fillna()	заменяет все NaN значения на параметр, указанный в функции
fillna(value=None, method="ffill")	заменяет NaN значения на предыдущий параметр, при method=“bfill" на следующий параметр

Замена NaN средним значением

data.fillna(data.mean())

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.000000	35.00000	155.548223	33.600000	0.627	50
1	1	85.0	66.000000	29.00000	155.548223	26.600000	0.351	31
2	8	183.0	64.000000	29.15342	155.548223	23.300000	0.672	32
3	1	89.0	66.000000	23.00000	94.000000	28.100000	0.167	21
4	0	137.0	40.000000	35.00000	168.000000	43.100000	2.288	33
5	5	116.0	74.000000	29.15342	155.548223	25.600000	0.201	30
6	3	78.0	50.000000	32.00000	88.000000	31.000000	0.248	26
7	10	115.0	72.405184	29.15342	155.548223	35.300000	0.134	29
8	2	197.0	70.000000	45.00000	543.000000	30.500000	0.158	53
9	8	125.0	96.000000	29.15342	155.548223	32.457464	0.232	54

Замена NaN следующим значением

data.fillna(value=None, method="bfill")

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	94.0	33.6	0.627	50
1	1	85.0	66.0	29.0	94.0	26.6	0.351	31
2	8	183.0	64.0	23.0	94.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	32.0	88.0	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	70.0	45.0	543.0	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	23.0	846.0	37.6	0.232	54

Замена NaN следующим значением

data.fillna(value=None, method="bfill“, limit=1)

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	23.0	94.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	32.0	88.0	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	70.0	45.0	543.0	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	37.6	0.232	54

Замена лимитированного количества NaN предыдущим значением

data.fillna(value=None, method="ffill")

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	29.0	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	35.0	168.0	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	50.0	32.0	88.0	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	45.0	543.0	30.5	0.232	54