Лекция

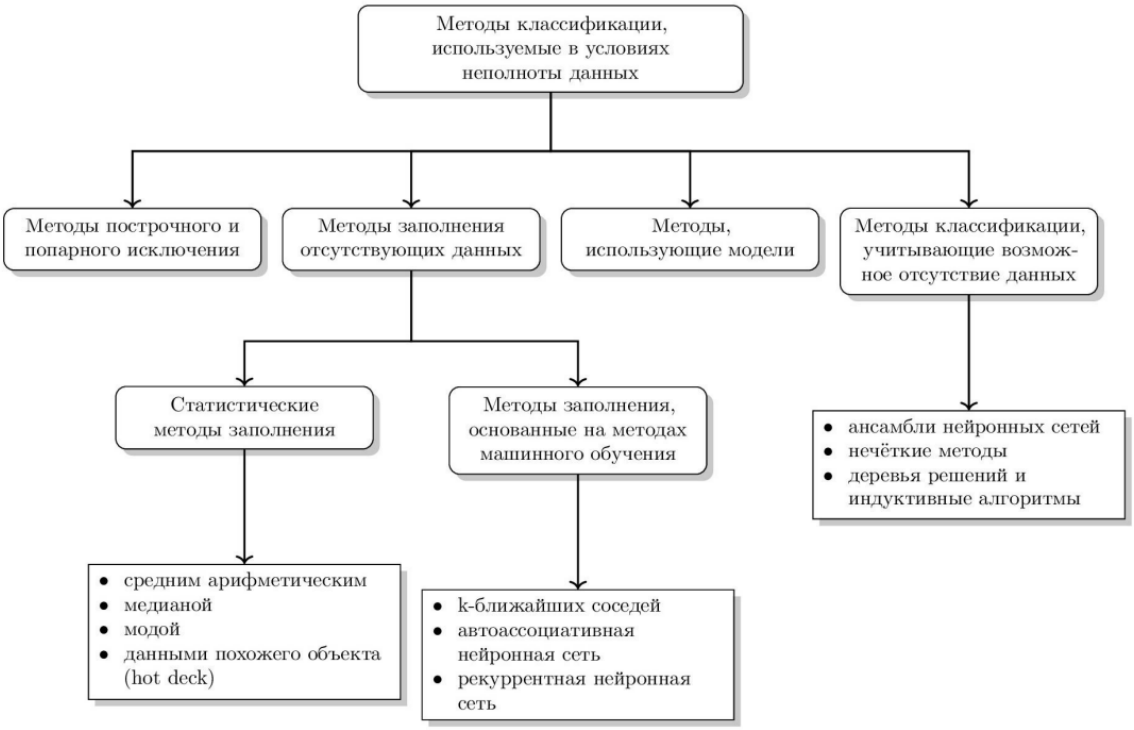
Типы пропусков в исходных данных

Способ обработки отсутствующих данных следует выбирать в соответствии с причинами отсутствия значений признаков [1]. Выделяют следующие типы пропусков [1].

1. Полностью случайные (missing completely at random; MCAR). Вероятность появления таких пропусков не зависит ни от значений самого измеряемого признака, ни от значений других признаков. Примером возникновения пропусков данного типа является потеря образца крови пациента, следствием которой является невозможность измерения ряда значений признаков.

2. Частично случайные (missing at random; MAR). Вероятность появления таких пропусков не зависит от значений непосредственно измеряемого признака, но обусловлена значениями других признаков. В качестве примера возникновения можно привести противопоказание некоторых анализов пациентам с поставленными диагнозами, при которых проведение данных анализов может нанести вред организму.

3. Неслучайные (not missing at random; NMAR). Вероятность появления неслучайных пропусков в данных зависит от значений самого признака. Если значение измеряемого признака не попадает в диапазон чувствительности измерительного прибора, то такие пропуски принято называть неслучайными. При первых двух типах (MCAR, MAR) во время анализа данных причину пропусков можно проигнорировать [1], используя более простые методы обработки отсутствующих данных.



Конструирование признаков включает

Выбор признаков - выбор самых полезных признаков для обучения на существующих признаках

Выделение признаков сочетание существующих признаков для выпуска более полезного признака

Создание новых признаков путем сбора новых данных.

Большинство алгоритмов МО не способны работать с недостающими признаками.

Игнорирование объектов с пропущенными значениями - Простейшим методом решения проблемы пропущенных значений в выборке является игнорирование объектов, имеющих пропуски. Такой метод применим только в том случае, когда малая часть объектов выборки имеет пропущенные значения.

Замена специальным значением - замена пропусков на специальное заранее определенное значение такое, как, например, 0 или -1. Данный подход позволяет не уменьшать размер выборки, однако может вносить значения, сильно отличающие от настоящих. Для дальнейшего применения методами, основанными на деревьях, разумно заполнять пропуск с помощью значения, не встречающегося в выборке, например, -1 для неотрицательных значений признаков. Для методов, чувствительным к масштабу признаков, пропуск заменяется с помощью 0.

Замена самым частым или средним значением - метод восстановления пропусков является замена на моду или среднее значение по конкретному признаку. В случае категориального признака все пропуски заменяются на наиболее часто встречающее значение, в случае количественного признака – на среднее значение по признаку. Данный метод, в отличие от двух предыдущих, учитывает имеющиеся данные и усредняет их. Преимуществом подхода является простота, однако на практике возникает проблема в определении, является ли конкретный признак категориальным или количественным, особенно при их большом количестве

Замена с помощью линейной регрессии

Замена с помощью случайного леса - Заменить пропущенные значения в конкретном признаке можно, предсказав его по другим признакам с помощью одного из алгоритмов машинного обучения. Так как среди признаков, по которым производится обучение, имеются пропущенные значения, то необходимо их изначально заменить с помощью одного из простейших методов восстановления пропусков. В данном случае инициализацию пропущенных значений производиться с помощью замены средним значением по признаку, в качестве алгоритма для уточнения пропусков используется случайный лес. Для каждого признака, имеющего пропущенные значения, производится обучение по объектам, не имеющим пропусков в данном признаке, для оставшихся объектов производится замена значений данного признака с помощью обученного алгоритма. Эта процедура

повторяется в течение нескольких итераций до сходимости или до максимального установленного числа итераций.

Замена с помощью метода ближайших соседей - близкие объекты по значениям среди заполненных признаков близки в признаках, значение которых может быть пропущено (гипотеза компактности), возникает применение метода k ближайших соседей для восстановления пропусков в данных. Реализация аналогична использованию классического метода k ближайших соседей за исключением того, предсказывается сразу несколько пропущенных признаков для каждого объекта.

Замена с помощью метода k средних Аналогично методу k ближайших соседей предполагается, что близкие по одним признакам объекты должны быть близки и по другим признакам. Однако в отличие от метода k ближайших соседей ищутся не ближайшие соседи для каждого объекта с пропущенными значениями, а используется информация о центре кластера, в который попал конкретный объект с пропусками. Заметим, что для разбиения на кластеры необходима начальная инициализация пропущенных значений. В данном случае выполняется инициализация пропущенных значений с помощью замены средним значением по признаку, кластеризация производится методом k средних. Пропущенные значения заменяются на соответствующие им значения центра кластера, в который попал каждый объект с пропусками. Данная процедура производится в течение нескольких итераций до сходимости или по достижению максимального заданного числа итераций.

Индикация пропусков в данных

Использование некоего значения-индикатора

К примеру, можно использовать число -9999 или редко встречающееся сочетание битов. Более часто встречающийся способ — условное обозначение через NaN. NaN — это специальное значение, определенное спецификацией IEEE для чисел с плавающей точкой.

NaT

У метода есть ограничения. Во-первых использование значений индикаторов может привести к дополнительным не оптимизированным расчетам. Во-вторых NaN доступен не для всех типов данных.

Использование масок. Можно создать отдельный булевый массив, индицирующий пропущенные значения. В ряде языков выделяется отдельный бит для разметки пропусков в массиве данных локально. Оба подхода влекут за собой перерасход памяти.

Pandas построена на NumPy, в котором отсутствует понятие пропуска для всех данных кроме данных с плавающей точкой. NumPy поддерживает маски, но использование такого подходжа в Pandas влечет значительные накладные расходы на хранение, вычисление и поддержку кода.

- индикаторы-числа
- NaN из Numpy
- None из Python

Объект None

None - объект python. Его нельзя использовать в NumPy и во всех производных массивах Pandas. None используется только в массивах с типом object. Когда мы создаем массив, используя None, автоматически создается массив с типом object.

Тип object означает, что NumPy не смог установить тип объектов массива, единственное что он знает — это то, что это объекты python. Операции с такими массивами будут производится на уровне языка python, т.е. со всеми накладными расходами, присущими языку с динамической типизацией. Оптимизация NumPy работать не будет.

Кроме того, функции агрегирования по массиву, например, massive.sum() или massive.min() выбросят ошибку, так как операции между численным значением и значением None не определены

Объект NaN

Объект NaN определяет отсутствие числового значения с плавающей точкой. Это вызывает некоторые проблемы — если NaN попадает в массив,

все данные приводятся к числам с плавающей точкой. Кроме того, все операции с NaN приводят к NaN, в том числе и функции агрегирования.

Строки всегда хранятся как object

Typeclass	Conversion When Storing NAN	NAN Sentinel Value
`floating`	No change	`np.nan`
`object`	No change	`None` or `np.nan`
`integer`	Cast to `float64`	`np.nan`
`boolean`	Cast to `object`	`None` or `np.nan`

Операции над пустыми значениями

В Pandas доступно несколько методов:

- isnull() — генерирует булеву маску для отсутствующих значений
- notnull() — тоже для непустых
- dropna() — фильтрация данных по отсутствующим значениям
- fillna() — замена пропусков с возвратом копии

Методы доступны как для объектов Series так и для dataFrame (с выбором измерения).

Для dropna() можно задать дополнительные параметры. how=’any’ задан по дефолту, можно переопределить как ‘all’ — будут отбрасываться только полностью пустые строки/столбцы. thresh задает минимальное значение непустых значений, выше которого строки/столбцы не отбрасываются.

Для fillna() доступно несколько аргументов. method=’ffill’ и method=’bfill’ определяют какими значениями будут заполняться пропуски (предыдущими или последующими в массиве).

`fillna`
Fill missing values using different methods.
`scipy.interpolate.Akima1DInterpolator`
Piecewise cubic polynomials (Akima interpolator).
`scipy.interpolate.BPoly.from_derivatives`
Piecewise polynomial in the Bernstein basis.
`scipy.interpolate.interp1d`
Interpolate a 1-D function.
`scipy.interpolate.KroghInterpolator`
Interpolate polynomial (Krogh interpolator).
`scipy.interpolate.PchipInterpolator`
PCHIP 1-d monotonic cubic interpolation.
`scipy.interpolate.CubicSpline`
Cubic spline data interpolator.

Задача интерполяции

Пусть функция $f(x)$ задана набором точек (x_i, y_i) на интервале $[a,b]$:

$$y_i = f(x_i),\ i = 0,1,\ldots,n,\ a \le x_i \le b \qquad (3.1)$$

Задача интерполяции – найти функцию $F(x)$, принимающую в точках x_i те же значения y_i. Тогда, **условие интерполяции**:

$$F(x_i) = y_i \qquad (3.2)$$

При этом предполагается, что среди значений x_i нет одинаковых.

Точки x_i называют **узлами интерполяции**.

Если $F(x)$ ищется только на отрезке $[a,b]$ – то это задача **интерполяции**, а если за пределами первоначального отрезка, то это задача **экстраполяции**.

Задача нахождения интерполяционной функции $F(x)$ имеет много решений, так как через заданные точки x_i, y_i можно провести бесконечно много кривых, каждая из которых будет графиком функции, для которой выполнены все условия интерполяции. Для практики важен случай интерполяции функции многочленами:

$$F(x) = P_m(x_i) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_m \cdot x^m,\ i = 0,1,\ldots,m \qquad (3.3)$$

При этом искомый полином называется **интерполяционным полиномом.**

При построении одного многочлена для всего рассматриваемого интервала $[a,b]$ для нахождения коэффициентов многочлена необходимо решить систему уравнений,

построенную на основе полинома (3.3). Данная система содержит $n+1$ уравнение, следовательно, с ее помощью можно определить $n+1$ коэффициент. Поэтому максимальная степень интерполяционного многочлена $m=n$, и многочлен принимает вид

$$P_n(x_i)=a_0+a_1\cdot x+a_2\cdot x^2+\ldots+a_n\cdot x^n,\ i=0,1,\ldots,n \qquad (3.4)$$

Локальная и глобальная интерполяция

Если задан $n+1$ узел интерполяции, то на этих узлах можно построить один интерполяционный многочлен *n*-й степени, $n-1$ многочленов первой степени и большой набор многочленов степени меньше *n*, опирающиеся на некоторые из этих узлов.

Теоретически максимальную точность обеспечивает многочлен более высокой степени. Однако на практике наиболее часто используют многочлены невысоких степеней, во избежание погрешностей расчета коэффициентов при больших степенях многочлена.

Если функция $f(x)$ интерполируется на отрезке $[a,b]$ с помощью единого многочлена $P_m(x)$ для всего отрезка, то такую интерполяцию называют **глобальной**. В случае **локальной интерполяции** на каждом интервале $[x_i,x_{i+1}]$ строится отдельный интерполяционный полином невысокой степени.

Кусочно-линейная интерполяция

Простейшим и часто используемым видом локальной интерполяции является линейная (или кусочно-линейная) интерполяция. Она заключается в том, что узловые точки соединяются отрезками прямых (рис.3.1), то есть через каждые две точки (x_i,y_i) и (x_{i+1},y_{i+1}) проводится прямая, то есть составляется полином первой степени:

$$F(x)=a_0+a_1\cdot x,\ \text{при}\ x_{i-1}\le x\le x_i \qquad (3.5)$$

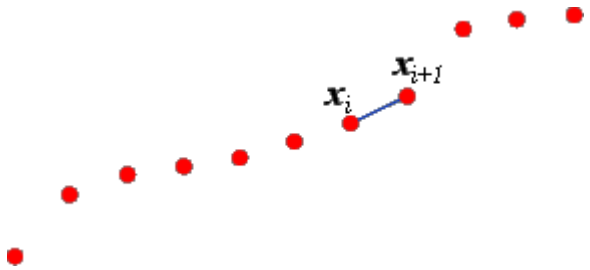


Рис. 3.1. Кусочно-линейная интерполяция

Коэффициенты a_0 и a_1 разные на каждом интервале $[x_i,x_{i+1}]$, и находятся из выполнения условий интерполяции на концах отрезка:

$$\begin{cases} f_{i-1} = a_0 + a_1 \cdot x_{i-1} \\ f_i = a_0 + a_1 \cdot x_i \end{cases} \quad (3.6)$$

Из системы уравнений (3.6) можно найти коэффициенты:

$$a_0 = f(x_{i-1}) - a_1 \cdot x_{i-1}, \quad a_1 = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \quad (3.7)$$

При использовании кусочно-линейной интерполяции сначала нужно определить интервал, в который попадает значение *x*, а затем подставить его в выражение (3.5), используя коэффициенты для данного интервала.

Кусочно-квадратичная интерполяция

В случае квадратичной интерполяции, для каждых трех узловых точек (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}), строится уравнение параболы:

$$F(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2, \text{ при } x_{i-1} \le x \le x_{i+1} \quad (3.8)$$

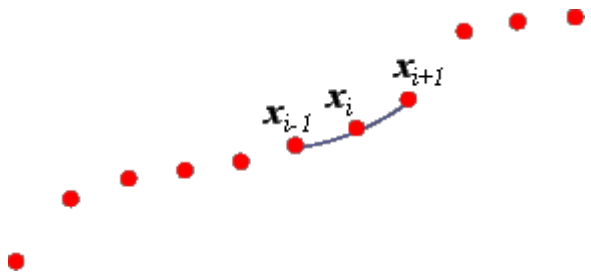


Рис.3.2. Кусочно-квадратичная интерполяция

Здесь коэффициенты a_0, a_1 и a_2 разные на каждом интервале $[x_{i-1}, x_{i+1}]$ и определяются решением системы уравнений для условия прохождения параболы через три точки:

$$\begin{cases} f_{i-1} = a_0 + a_1 \cdot x_{i-1} + a_2 \cdot x_{x-i}^2 \\ f_i = a_0 + a_1 \cdot x_i + a_2 \cdot x_x^2 \\ f_{i+1} = a_0 + a_1 \cdot x_{i+1} + a_2 \cdot x_{x+i}^2 \end{cases} \quad (3.9)$$

Из системы уравнений (3.9) можно найти коэффициенты:

$$a_0 = f(x_{i-1}) - a_1 \cdot x_{i-1} - a_2 \cdot x_{i-1}^2$$
$$a_1 = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} - a_2 \cdot (x_i + x_{i-1})$$
$$a_2 = \frac{f(x_{i+1}) - f(x_{i-1})}{(x_{i+1} - x_{i-1}) \cdot (x_{i+1} - x_i)} - \frac{f(x_i) - f(x_{i-1})}{(x_i - x_{i-1}) \cdot (x_{i+1} - x_i)} \quad (3.10)$$

Многочлен Лагранжа

При глобальной интерполяции на всем интервале $[a,b]$ строится единый многочлен. Одной из форм записи интерполяционного многочлена для глобальной интерполяции является многочлен Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} y_i \cdot l_i(x) \quad (3.11)$$

где $l_i(x)$ – базисные многочлены степени *n*:

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)} \quad (3.12)$$

То есть многочлен Лагранжа можно записать в виде:

$$L_n(x) = \sum_{i=0}^{n} y_i \cdot \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \quad (3.13)$$

Многочлен $l_i(x)$ удовлетворяет условию $l_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$. Это условие означает, что многочлен равен нулю при каждом x_j кроме x_i, то есть $x_0, x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n$ – корни этого многочлена. Таким образом, степень многочлена $L_n(x)$ равна *n* и при $x \neq x_i$ обращаются в ноль все слагаемые суммы, кроме слагаемого с номером $i = j$, равного y_i.

Выражение (3.11) применимо как для равноотстоящих, так и для не равноотстоящих узлов. Погрешность интерполяции методом Лагранжа зависит от свойств функции $f(x)$, от расположения узлов интерполяции и точки *x*. Полином Лагранжа имеет малую погрешность при небольших значениях *n* (*n*<20). При больших *n* погрешность начинает расти, что свидетельствует о том, что метод Лагранжа не сходится (то есть его погрешность не убывает с ростом *n*).

Многочлен Лагранжа в явном виде содержит значения функций в узлах интерполяции, поэтому он удобен, когда значения функций меняются, а узлы интерполяции неизменны. Число арифметических операции, необходимых для построения многочлена Лагранжа, пропорционально n^2 и является наименьшим для всех форм записи. К недостаткам этой формы записи можно отнести то, что с изменением числа узлов приходится все вычисления проводить заново.

Кусочно-линейная и кусочно-квадратичная локальные интерполяции являются частными случаями интерполяции многочленом Лагранжа.

Многочлен Ньютона

Другая форма записи интерполяционного многочлена – интерполяционный многочлен Ньютона с разделенными разностями. Пусть функция $f(x)$ задана с произвольным шагом, и точки таблицы значений пронумерованы в произвольном порядке.

Разделенные разности нулевого порядка совпадают со значениями функции в узлах. Разделенные разности первого порядка определяются через разделенные разности нулевого порядка:

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \qquad (3.14)$$

Разделенные разности второго порядка определяются через разделенные разности первого порядка:

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i} \qquad (3.15)$$

Разделенные разности *k*-го порядка определяются через разделенные разности порядка $k-1$:

$$f(x_i, x_{i+1}, \ldots, x_{i+k}) = \frac{f(x_{i+1}, \ldots, x_{i+k}) - f(x_i, \ldots, x_{i+k-1})}{x_{i+k} - x_i} \qquad (3.16)$$

Используя понятие разделенной разности интерполяционный многочлен Ньютона можно записать в следующем виде:

$$P_n(x) = f(x_0) + f(x_0, x_1)\cdot(x - x_0) + f(x_0, x_1, x_2)\cdot(x - x_0)\cdot(x - x_1) + \ldots + \\ + f(x_0, x_1, \ldots, x_n)\cdot(x - x_0)\cdot(x - x_1)\ldots\cdot(x - x_{n-1}) \qquad (3.17)$$

За точностью расчета можно следить по убыванию членов суммы (3.17). Если функция достаточно гладкая, то справедливо приближенное

равенство $f(x)-P_n(x)\approx P_{n+1}(x)-P_n(x)$. Это приближенное равенство можно использовать для практической оценки погрешности интерполяции: $\varepsilon_n=\left|P_{n+1}(x)-P_n(x)\right|$.

http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava3.html#p_3_3

https://devpractice.ru/pandas-indexing-part3/

https://konstantinklepikov.github.io/2020/07/25/empty-data-processing-in-pandas.html