BaseGroup™ Labs
ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ

Deductor

Руководство по алгоритмам

Версия 5.2.0

В руководстве описаны используемые в Deductor Studio алгоритмы интеллектуального анализа данных и визуализации полученных результатов. Книга предназначена для аналитиков, которых интересуют реализованные в узлах-обработчиках математические алгоритмы для проверки достоверности полученных результатов. Общеизвестные методы и алгоритмы описываются кратко со ссылками на первоисточники и научно-популярную литературу; алгоритмы, разработанные BaseGroup Labs, приводятся полностью.

Содержание

Введение

В процессе построения сценария аналитика может заинтересовать алгоритм работы того или иного узла-обработчика для того чтобы убедиться в достоверности полученных результатов.

Это Руководство планировалось как сборник ответов часто возникающие вопросы по реализованным алгоритмам обработки и визуализации данных. Вопросы анализа, построения сценариев, генерации отчетов, работы с визуализаторами подробно освещаются в документе «Руководстве аналитика». В «Руководстве администратора» можно найти ответы на вопросы по установке и обслуживанию аналитической платформы Deductor.

В случае, когда известный алгоритм реализован без изменений, приводится ссылка на первоисточник или научно-популярную литературу, а сам алгоритм сопровождается лишь кратким описанием. Поэтому в Руководстве не дается подробное описание нахождения известных функций и коэффициентов, которые изучаются в курсах высшей математики, математической статистики, численных методов и методов оптимизации.

В случае, когда алгоритм представляет собой модификацию существующего алгоритма, либо совершенно новый алгоритм, разработанный компанией BaseGroup Labs, он приводится полностью.

В данном руководстве используется ряд терминов и понятий.

Тип данных – тип данных, содержащихся в поле.

- Логический – данные в поле могут принимать только два значения: Истина или Ложь.
- Дата/время – поле содержит данные типа дата/время.
- Вещественный – данные в поле представляют собой числа с плавающей точкой.
- Целый – данные в поле представляют собой целые числа.
- Строковый – данные в столбце представляют собой строки символов.

Строковый тип делится на два подтипа:

- упорядоченные – значения можно упорядочить относительно друг друга;
- категориальные – значения нельзя упорядочить.

Вид данных – характер данных, содержащихся в столбце:

- Непрерывный – значения в столбце могут принимать любое значение в рамках своего типа. Как правило, непрерывными являются числовые данные.
- Дискретный – данные в столбце будут принимать ограниченное число значений. Как правило, дискретный характер носят строковые данные.

Строковые и логические типы могут иметь только дискретный вид данных.

Визуализаторы

Статистика

Статистические характеристики рассчитываются для каждого поля набора данных. В верхней части окна статистики отображается общее количество записей в наборе данных. В окне статистики для каждого поля отображается следующая информация.

1. Гистограмма. Для полей с дискретным видом данных число столбцов равно числу уникальных значений. Для полей с непрерывным видом данных оно рассчитывается по формуле

   $$[1 + 3{,}2 \cdot \lg(N)],$$

   где N – общее количество записей.
2. Минимум.
3. Максимум.
4. Среднее: $\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$.
5. Стандартное отклонение: $\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$.
6. Сумма: $\sum_{i=1}^{N} x_i$.
7. Сумма квадратов: $\sum_{i=1}^{N} x_i^2$.
8. Количество уникальных значений.
9. Количество пустых значений.

Нормализаторы и денормализаторы

Некоторые алгоритмы не способны напрямую работать со всеми типами и видами данных. Для этих целей предусмотрены нормализаторы и денормализаторы внутри узлов.

Нормализаторы

Преобразование полей с непрерывным видом данных

Пусть необходимо привести значения к диапазону $[a, b]$. Для этого используется следующая формула:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (b - a) + a,$$

где x – текущее значение, $x_{\min}$, $x_{\max}$ – минимальные и максимальные значения поля. Так работает **Линейный нормализатор**.

Такое преобразование производится, если активен флаг **Привести к диапазону** в настройках параметров нормализации данных, в противном случае нормализация не производится.

По умолчанию для входных полей a = –1, b = 1; для выходных a = 0, b = 1

Преобразование полей с дискретным видом данных

В данной ситуации кодирование осуществляется несколькими способами.

Если значения поля можно упорядочить, например в порядке: A, B, C, то будут присвоены следующие значения: A – 0, B – 1, C – 2 (нормализатор **Уникальные значения**). Затем эти значения приводятся к требуемому диапазону, как это происходит с данными непрерывного вида.

Для полей категорийного типа предусмотрен нормализатор **Битовая маска**, предусматривающий два способа кодирования:

- кодирование **Комбинация битов**;
- кодирование **Позиция бита**.

Согласно нормализатору **Битовая маска** каждому уникальному значению присваивается комбинация двух чисел a и b, где a – минимум диапазона линейного преобразования, b – максимум.

Используя кодирование **Комбинация бита**, маска будет состоять из n чисел, которые могут принимать значение a или b. Для вычисления n используется следующая формула:

$$n = \lceil \log_2 m \rceil,$$

где m – количество уникальных состояний, которые необходимо закодировать.

Согласно способу кодирования **Позиция бита** кодирование происходит путем замены на маску из m чисел. Первое уникальное значение имеет код, где крайним правым число является b, остальные – a. Каждое следующее получает новый код, путем смещения числа b на одну позицию. Последнее уникальное значение имеет код, где число b находится крайней левой позиции.

Рассмотрим эти нормализаторы на примере.

Пусть имеются следующие значения поля, которые необходимо закодировать в диапазоне [0; 1]: A, B, C, D.

В таблице 1 приведены коды для каждого из рассмотренных способов кодирования.

Таблица 1 – Пример кодирования

	Битовая маска	
	Комбинация битов	Позиция битов
Код A	00	0001
Код B	01	0010
Код C	10	0100
Код D	11	1000
Количество полей (длина маски)	2	4

Таким образом, как видно из таблицы 1, каждое входное поле заменяется на несколько входных полей, которые будут подаваться на вход той или иной модели (нейросеть, линейная регрессия и т.д.).

Дополнительную информацию можно найти в источнике [15, стр. 166].

Денормализаторы

Каждому нормализатору соответствует свой денормализатор, который преобразует выходные значения алгоритма в исходный диапазон или набор.

Вернемся к таблице 1. Пусть для выходного поля использовался нормализатор **Позиция бита**. Алгоритм кодирования на выходе выдал 0010. Тогда денормализатор преобразует его к значению B. Все действия по денормализации заложены внутри узлов. Если на выходе алгоритма получилось значение между 0 и 1, то такое значение округляется до ближайшего целого.

Узел *Парциальная обработка*

В основе данного узла лежат алгоритмы спектральной обработки (частотной фильтрации), восстановления пропущенных значений наиболее вероятными значениями и модифицированным фильтром Калмана, вейвлет преобразования и робастной фильтрации.

Настройка данного узла состоит из нескольких шагов, предлагаемых мастером, на каждом из которых определяются используемые алгоритмы и их параметры.

Восстановление пропущенных данных

В основе данного шага узла лежит алгоритм заполнения пропущенного значения среднем из наиболее вероятного интервала.

Если в мастере выбран параметр Аппроксимация, то используется модифицированный фильтр Калмана, который работает только с полями, содержащими данные непрерывного вида (описание данного алгоритма временно недоступно).

Если в мастере выбран параметр Максимальное правдоподобие, то используется следующий алгоритм, который работает только с полями, содержащими данные непрерывного вида.

Исходный диапазон значений Y разбивается на n интервалов X_i ($i = 1, 2, \ldots, n$):

$$X_1 \cup X_2 \cup ... \cup X_n = Y,$$

$$\forall_{l \neq m} : X_l \cap X_m = \varnothing,$$

$$|X_1| = |X_2| = ... = |X_n|,$$

$$n = [1 + 3{,}2 \cdot \lg(N)],$$

где N – общее количество записей, $|X_i|$ – количество примеров в интервале X_i.

Затем выбирается интервал, содержащий наибольшее количество примеров. Все пропуски заполняются средним значением из данного интервала, то есть:

$$k = \arg\max_i \left(|X_i| \right);$$

$$x_{new} = \overline{X}_k .$$

Дополнительная литература: [15, стр. 253]

Редактирование аномальных значений

В основе данного шага узла лежит алгоритм робастной фильтрации, который позволяет обнаруживать и корректировать аномальные значения упорядоченного ряда данных.

Пусть имеется (упорядоченный) временной ряд данных Y из n записей.

Шаг 1. Находятся изменения dY исследуемой переменной и медиану $S(dY)$ полученных изменений:

$$dy_i = \left| y_i - y_{i-1} \right| .$$

Шаг 2. Рассчитываются отклонения ddY каждого изменения dy_i от медианы $S(dY)$:

$$ddy_i = \left| S(dY) - dy_i \right|.$$

Находится медиана $D(ddY)$ среди всех отклонений ddy_i.

Шаг 3. Вычисляется порог:

$$K = S(dY) + D(ddY) \cdot w,$$

где $w = 4$ при слабой степени подавления шумов, $w = 3$ при средней степени подавления шумов, $w = 2$ при сильной степени подавления шумов.

Шаг 4. Поиск и коррекция аномальных значений ряда:

$$\text{Если } dy_i > K, \text{ то } \hat{y}_i = y_i \pm K.$$

Спектральная обработка

Алгоритм спектральной обработки включает три этапа:

1. преобразование Фурье;
2. обнуление значений высоких частот;
3. обратное преобразование Фурье.

Предварительная нормализация данных не производится, алгоритм с категориальными данными не работает.

На этапе **2** используется единственный устанавливаемый пользователем параметр – **Степень сглаживания** n, который показывает долю частот, значения которых необходимо обнулить. Эта доля есть ближайшее целое к числу k, которое рассчитывается по формуле:

$$k = \frac{n \cdot N}{100},$$

где N – общее количество записей набора данных.

Дополнительная литература: [24, 25].

Вейвлет преобразование

В основе данного шага обработчика лежит алгоритм построения вейвлетов, предложенный американским математиком И.Добеши. Его описание можно найти в источнике [32].

Узел *Факторный анализ*

В основе данного узла лежит метод главных компонент.

Цель факторного анализа заключается в понижении размерности пространства факторов. Понижение размерности необходимо в случаях, когда входные факторы коррелированны друг с другом, то есть взаимозависимы. В факторном анализе речь идет о выделении из множества измеряемых характеристик объекта новых факторов, более адекватно отражающих свойства объекта. Подробнее о методе главных компонент можно узнать в источнике [16].

Вычисления и параметры мастера обработки

Перед применением метода главных компонент происходит нормирование всех значений используемых полей по формуле:

$$x_i^* = \frac{x_i - \bar{x}}{\text{СКО}_x},$$

где СКО_x – среднее квадратичное отклонение, $\bar{x}$ – среднее значение.

Метод главных компонент реализован в узле без изменений, как это описывается в [16]. Для ковариационной матрицы находятся её собственные значения $\lambda_1 > \lambda_2 > \ldots > \lambda_k$ ($k \le n$, где n – исходная размерность пространства). Оси новых факторов соответствуют собственным векторам, причем i-ой главной компоненте соответствует собственное значение λ_i.

В соответствии с полученными значениями λ_i для каждой i-ой главной компоненты рассчитываются вклад в результат RV_i и суммарный вклад $SumV_i$:

$$RV_i = \frac{\lambda_i}{\sum_{j=1}^{k} \lambda_j} \cdot 100\%;$$

$$SumV_i = \frac{\sum_{j=1}^{i} \lambda_j}{\sum_{j=1}^{k} \lambda_j} \cdot 100\%.$$

Дополнительная литература: [10, 16]

Узел *Корреляционный анализ*

В данном узле рассчитывается корреляционная матрица.

Корреляционный анализ применяется для оценки зависимости выходных полей данных от входных факторов и устранения незначащих факторов. Принцип корреляционного анализа состоит в поиске таких значений, которые в наименьшей степени коррелированны (взаимосвязаны) с выходным результатом. Такие факторы могут быть исключены из результирующего набора данных практически без потери полезной информации. Критерием принятия решения об исключении является порог значимости. Если корреляция (степень взаимозависимости) между входным и выходным факторами меньше порога значимости, то соответствующий фактор отбрасывается как незначащий.

Вычисления

Мастер настройки узла позволяет указать как несколько входных полей, так и несколько выходных. Рассчитывается коэффициент корреляции между каждой парой входных и выходных полей.

Метод расчета **Коэффициент корреляции Пирсона** использует формулу

$$R(X,Y) = \frac{M[XY] - M[X]M[Y]}{\sqrt{M[X^2] - (M[X])^2} \cdot \sqrt{M[Y^2] - (M[Y])^2}},$$

где M – математическое ожидание, X – набор значений входного поля, Y – набор значений выходного поля.

При выборе метода расчета **Максимум взаимокорреляционной функции** будет вычислен максимум из коэффициентов корреляции двух процессов, рассчитанных при всевозможных временных сдвигах. Следует применять, если необходимо узнать линейную зависимость между двумя процессами или частями процессов происходящих с определённым временным лагом.

Расчет коэффициента корреляции Пирсона происходит с использованием алгоритма БПФ. Здесь можно выделить два шага:

Шаг 1: Быстрое преобразование Фурье.

$$X(k) = \sum_{n=0}^{N-1} x_n e^{-2\pi i n k / N}, k = 0, ..., N-1,$$

$$Y(k) = \sum_{n=0}^{N-1} y_n e^{-2\pi i n k / N}, k = 0, ..., N-1,$$

где N – число записей.

Шаг 2: Расчет коэффициента корреляции.

$$R(X,Y) = \frac{1}{N} \cdot F^{-1}[X^*(k)Y(k)],$$

где F^{-1} – обратное быстрое преобразование Фурье, $X^*(k)$ – комплексно-сопряженное множество к $X(k)$.

Подробнее про данный алгоритм нахождения корреляции можно найти информацию в источнике [22].

Узел *Ассоциативные правила*

В данном узле реализован алгоритм Apriori для поиска ассоциативных правил [3, 4]. Исторически ассоциативные правила стали применять для анализа рыночной корзины покупателей.

В основе алгоритма Apriori лежит понятие частого набора (frequent itemset), который также можно назвать частым предметным набором, часто встречающимся множеством (соответственно, он связан с понятием частоты). Под частотой понимается простое количество транзакций, в которых содержится данный предметный набор. Тогда частыми наборами будут те из них, которые встречаются чаще, чем в заданном числе транзакций.

Информацию по ассоциативным правилам также можно найти в источнике [27].

Вычисления

Описание модели

Пусть $I = I_1, I_2, \ldots, I_m$ – набор товаров (элементов транзакции), T – база данных транзакций, t – бинарный вектор, где $t[k] = 1$, если в транзакции t содержится товар I_k, в противной случае $t[k] = 0$, Y, X – набор некоторых продуктов из I. Под ассоциативным правилом понимается импликация вида $X \Rightarrow Y$, такая что $Y \in I$, $X \in I$, X и Y – не пустые множества.

Правило $X \Rightarrow Y$ соответствует T с достоверностью $0 \le c \le 1$ тогда и только тогда, когда (100·c)% транзакций в T, содержащих X, включают в себя и Y.

Нахождение правил и коэффициентов

Пользователь задает значения минимальной и максимальной поддержки и достоверности, а также максимальную мощность часто встречающегося набора.

Нахождение правил происходит согласно алгоритму, предложенному в [4], а вычисление достоверности, поддержки и лифта как в [15, стр. 281].

Узел *Нейросеть*

В данном узле строится многослойный персептрон – виртуальный механизм, способный суммировать сигналы с нескольких входов, затем сигнал, проходя через функцию активации, подается на выход.

Нейронная сеть – это упорядоченная структура из нейронов, связанных друг с другом определенным образом.

В узле используется многослойный персептрон, которым может быть обучен одним из двух алгоритмов: Back Propagation of error (алгоритм обратного распространения ошибки) или Resilient Propagation [5, 6].

Полученная модель решает задачи классификации и регрессии.

Многослойный персептрон

Многослойный персептрон – это разновидность нейронной сети. Согласно ее архитектуре, нейроны объединены в слои, которые взаимосвязаны друг с другом. Такая сеть имеет входной слой, несколько скрытых и выходной. Например, в многослойном персептроне с одним скрытым слоем каждый нейрон входного слоя связан с каждым нейроном в скрытом слое, в свою очередь, нейроны скрытого слоя связаны с нейронами выходного.

В узле предусмотрены следующие виды функций активации нейронов:

1. сигмоида;
2. гиперболический тангенс (гипертангенс);
3. арктангенс.

Каждая связь имеет свой вес, который определяет степень влияния одного нейрона на другой. Информация поступает на входной слой, проходит через скрытые, и попадает на выходной, где рассчитывается результат работы сети.

При решении задачи классификации нейронной сетью число нейронов в выходном слое равно количеству факторов, которое соответствует кодированию всех уникальных значений поля заданным пользователем нормализатором.

Перед началом использования нейронную сеть необходимо обучить. Для этого в данном узле предусмотрены алгоритмы Back Propagation of error (алгоритм обратного распространения ошибки) и Resilient Propagation.

Эпохой обучения называют один проход алгоритма по массиву данных. Все ошибки (максимальная и средняя) на обучающем и тестовом множестве рассчитываются в нормированном виде каждую эпоху.

Вычисление выходного значения y происходит следующим образом.

Пусть нейрон имеет n входных сигналов $x_k\,(1 \le k \le n)$, для каждого нейрона задана функция активации $f(S)$ с крутизной a. Учитывая, что x_0 = 1, выходное значение рассчитывается следующим образом:

$$S = \sum_{j=0}^{n} x_j w_j ,$$

$$y_i = f(S) ,$$

где w_j – вес связи текущего i-ого нейрона с j-ым из предыдущего слоя.

Функция $f(S)$ может быть следующая:

1. сигмоида $f(S) = \frac{1}{1+e^{-aS}}$;
2. гипертангенс $f(S) = \frac{e^{aS} - e^{-aS}}{e^{aS} + e^{-aS}}$;
3. арктангенс $f(S) = arctg(S)$.

Алгоритм Back Propagation

Алгоритм обратного распространения ошибки – это один из методов обучения многослойного персептрона [5]. Обучение производится только в режиме «онлайн». Коррекция весов производится после предъявления каждого примера обучающего множества.

Алгоритм Resilient Propagation (Rprop)

В узле по умолчанию предлагается алгоритм, названный Resilient Propagation (Rprop) который был предложен М. Ридмиллером (M.Riedmiller) и Г. Брауном (H.Braun) в источнике [6]. Обучение производится только в режиме «оффлайн».

Нормализация и кодирование

Все входные поля для нейронной сети должны быть представлены в числовом виде. Для этого все поля приводятся к диапазону значений $[a, b]$ (смотрите раздел Преобразование полей с непрерывным видом данных).

В данном узле для полей с дискретным видом данных доступны следующие нормализаторы (смотрите раздел Нормализаторы):

- уникальные значения;
- битовая маска.

Для полей с непрерывным видом данных – только Линейный нормализатор (денормализатор).

Выходные поля непрерывного вида также нормализуются (по умолчанию в диапазон от 0 до 1).

Обработка пропусков не осуществляется. Поля, содержащие пустые значения, непригодны для использования в узле.

Настройки узла

Структура нейронной сети

Параметры этого блока необходимы для создания структуры нейронной сети.

Число скрытых слоев. Входные и выходные слои создаются автоматически, и специальной настройки не требуют, пользователь должен задать только количество скрытых слоев. Минимально допустимое значение равно 1 (оно же по умолчанию).

Количество нейронов. Для каждого скрытого слоя пользователь может вручную указать количество нейронов. По умолчанию первый скрытый слой имеет два нейрона, все остальные (если они есть) – один.

Активационная функция: тип функции. Пользователь может выбрать один из следующих типов:

1. сигмоида (по умолчанию);
2. гипертангенс;
3. арктангенс.

Активационная функция: крутизна. Данный параметр доступен только тогда, когда в качестве типа функции выбрана либо сигмоида, либо гипертангенс. По умолчанию равен 1.

Настройка процесса обучения нейронной сети

В узле можно выбрать один из алгоритмов обучения: Back Propagation или Resilient Propagation (по умолчанию). Каждый алгоритм имеет свои параметры.

Алгоритм Back Propagation: Скорость обучения. Задает градиентную составляющую в суммарной величине коррекции веса. По умолчанию 0,1.

Алгоритм Back Propagation: Момент. Задает инерционную составляющую, учитывающую величину последнего изменения веса в суммарной величине коррекции веса. По умолчанию 0,9.

Алгоритм Resilient Propagation: Шаг спуска. В случае изменения знака градиентной составляющей ошибки для данного веса задает величину следующей коррекции веса. По умолчанию 0,5.

Алгоритм Resilient Propagation: Шаг подъема. В случае сохранения знака градиентной составляющей ошибки для данного веса задает величину следующей коррекции веса. По умолчанию 1,2.

Параметры остановки обучения

Ошибка, меньше которой пример считается распознанным. Задается в относительных единицах (нормализованное значение). По умолчанию 0,05.

Доступны следующие варианты окончания обучения:

- по достижению эпохи (по умолчанию 10000);
- если средняя ошибка на обучающем множестве меньше заданного пользователем значения (по умолчанию отключено);
- если максимальная ошибка на обучающем множестве меньше заданного пользователем значения; (по умолчанию отключено)
- если процент распознанных примеров на обучающем множестве больше заданного пользователем значения (по умолчанию отключено);
- если средняя ошибка на тестовом множестве меньше заданного пользователем значения (по умолчанию отключено);
- если максимальная ошибка на тестовом множестве меньше заданного пользователем значения (по умолчанию отключено);
- если процент распознанных примеров на тестовом множестве больше заданного пользователем значения (по умолчанию отключено).

Для расчета ошибок используются следующие формулы. Средняя ошибка для примера i с k выходными полями равна:

$$MSE_i = \frac{1}{k}\sum_{j=1}^{k}(\hat{y}_j - y_j)^2 ,$$

где $\hat{y}_j$ – нормализованный выходной сигнал, сформированный нейросетью, y_j – эталонный нормализованный выходной сигнал.

Ошибка MSE_i выводится для каждого примера в выходном наборе данных с постфиксом **_ERR**, добавляемому к имени и метке выходного поля.

Усредненная ошибка по всей выборке:

$$MSE_{avg} = \frac{1}{N}\sum_{i=1}^{N} MSE_i \ ,$$

где N – количество примеров.

Максимальная ошибка:

$$MSE_{\max} = \max_i \left(MSE_i \right).$$

Узел *Дерево решений*

В данном узле реализован модифицированный алгоритм построения дерева решающих правил на основе алгоритма C4.5. Полученная в узле модель решает задачу классификации.

Вычисления

В узле за основу взят алгоритм C 4.5, описанный в источнике [13], однако для более эффективной работы нами в него были внесены следующие изменения в процедуру разбиения значений.

Разбиение по полям **непрерывного** вида производится следующим образом.

1. Упорядочить записи по возрастанию.
2. Разбить исходное множество T на два – T_1 и T_2. Причем на первой итерации в T_1 попадает только первая запись, все остальные в T_2. Следующее разбиение получить путем перемещения первого элемента из T_2 в T_1.
3. Вычислить индекс $Gini_{split}$ для каждого из возможных способов разбиений T. Выбрать тот, при котором указанный индекс минимален. Для этого используются следующие формулы:

$$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2 ,$$

$$Gini_{split}(T) = \frac{N_1}{N} Gini(T_1) + \frac{N_2}{N} Gini(T_2) ,$$

где p_i – вероятность того, что пример класса i находится во множестве T, N – количество примеров.

Подобное изменение повышает скорость работы алгоритма.

Используются следующие случаи остановки дальнейшего разбиения узла:

1. если в узле содержится достаточное количество примеров;
2. если узел содержит примеры одного класса;
3. если количество нераспознанных примеров меньше минимального количества в примеров в узле.

Настройки узла

Минимальное количество примеров в узле, при котором будет создан новый, $Count_{min}$. Данный параметр используется во время построения дерева. Пусть в i-й узел попало n_i примеров, тогда узел будет удален, если $n_i < Count_{min}$.

Уровень доверия, используемый при отсечении узлов, %. Этот параметр необходим для оптимизации дерева после его построения (подробнее об его использовании можно узнать в [13] с. 37-43).

Основная идея использования уровня доверия заключается в следующем. Для каждого узла находят дополнительное количество ошибок по формуле:

$$AddErr = \begin{cases} 0, \text{ при } N = 0, \\ N, \text{ при } CF = 0 \text{ и } E < 10^{-6}, \\ N \cdot (1 - CF^{\frac{1}{N}}), \text{ при } CF \neq 0 \text{ и } E < 10^{-6}, \\ N \cdot (1 - CF^{\frac{1}{N}}) + E(AddErr^{*} - N \cdot (1 - CF^{\frac{1}{N}})), \text{ при } CF \neq 0 \text{ и } 10^{-6} \leq E < 0{,}9999, \\ N + E(AddErr^{*} - N), \text{ при } CF = 0 \text{ и } 10^{-6} \leq E < 0{,}9999, \\ N \cdot \dfrac{Err}{N + Coeff} - E, \text{ при } 0{,}9999 \leq E < (N - 0{,}5), \\ 0{,}67 \cdot (N - E), \text{ при } E \geq 0{,}9999 \text{ и } E \geq (N - 0{,}5), \end{cases}$$

где $Err = E + 0{,}5 + \dfrac{Coeff}{2} + \left(Coeff \cdot \left((E + 0{,}5) \cdot \left(1 - \dfrac{E + 0{,}5}{N} \right) + \dfrac{Coeff}{4} \right) \right)^2$, E – количество ошибок в узле, N – количество записей в узле, $CF = \dfrac{\text{Уровень доверия}}{100\%}$, $AddErr^{*} = AddErr$ при $E = 1$, $Coeff$ – коэффициент, рассчитываемый следующим образом:

$$Coeff = \left(Dev_{i-1} + (Dev_i - Dev_{i-1}) \cdot \frac{CF - Val_{i-1}}{Val_i - Val_{i-1}} \right)^2,$$

где значения Val_i, Val_{i-1}, Dev_i и Dev_{i-1} выбираются из таблицы 3, таким образом, чтобы они удовлетворяли одному из условий:

1. $Val_{i-1} < CF \leq Val_i$, если $0 < i < 8$;
2. $i = 1$, если $CF \leq Val_0$;
3. $i = 8$, если $CF \geq Val_8$.

Таблица 2 – Бета распределение с параметрами α = 1, β = 1

i	0	1	2	3	4	5	6	7	8
Val_i	0	0,001	0,005	0,01	0,05	0,10	0,20	0,40	1,00
Dev_i	4,0	3,09	2,58	2,33	1,56	1,28	0,84	0,25	0,00

Ожидаемое количество ошибок есть сумма фактических ошибок в узле, полученных при построении дерева и дополнительных ошибок, то есть:

$$SNE = E + AddErr.$$

Далее введем понятие *большая ветка* – это дочерний подузел, в который после разбиения переместилось б**о**льшее число примеров, по сравнению с другими ветвями дерева. Рассмотрим, как решается вопрос об отсечении.

Подузлы полностью удаляются, если одновременно выполняются условия:

1. текущий узел не лист;
2. ошибка в текущем узле меньше, чем сумма ошибок по его подузлам;

3. ошибка в текущем узле меньше, чем в большой ветке.

Узел заменяется на большую ветку, если одновременно выполняются условия:

1. большая ветка не является листом;
2. ожидаемая ошибка в большой ветке меньше, чем в других подузлах;
3. ожидаемая ошибка в большой ветке меньше, чем в текущем узле.

Визуализаторы

Правила

Достоверность представляет собой меру точности правила. Её можно просмотреть в процентном и количественном выражениях.

В первом случае достоверность правила Если {условие A}, то {класс X} находится следующим образом:

$$C(A,X) = \frac{N(A,X)}{N(A)} \cdot 100\%,$$

где $N(A, X)$ – количество примеров, содержащих как условия A, так и принадлежащих к классу X, $N(A)$ – количество записей, содержащих только условие A.

Во втором случае достоверность равна количеству примеров, содержащих условие следствие, то есть

$$C(A, X) = N(A, X).$$

Поддержка – это число примеров, содержащих только условие, иначе говоря, записи, попавшие в лист дерева. В процентном выражении она рассчитывается следующим образом:

$$S(A,X) = \frac{N(A)}{N} \cdot 100\%,$$

где $N(A)$ – количество примеров, содержащих только условие A, N – общее количество записей.

В количественном выражении вычисления имеют следующий вид:

$$S(A , X) = N(A).$$

Значимость атрибутов

Данный визуализатор представляет собой таблицу, построенную на основе предварительно рассчитанного показателя **Значимость**.

Значимость. Показатель, характеризующий, насколько сильно выходное поле зависит от каждого из входного. Рассчитывается после построения дерева классификационных правил.

Пусть всего g входных атрибутов, тогда формула для расчета значимости m-ого имеет следующий вид:

$$\text{Значимость}_m = \frac{\sum_{j=1}^{k_m}\left(E_{m,j} - \sum_{i=1}^{n_{m,j}} E_{m,j,i} \cdot \frac{N_{m,j,i}}{N_{m,j}}\right)}{\sum_{l=1}^{g}\sum_{j=1}^{k_l}\left(E_{l,j} - \sum_{i=1}^{n_{l,j}} E_{l,j,i} \cdot \frac{N_{l,j,i}}{N_{l,j}}\right)} \cdot 100\%,$$

где k_l – количество узлов, которые были разбиты по атрибуту l, $E_{l,j}$ – энтропия родительского узла, разбитого по атрибуту l, $E_{l,j,i}$ – дочерний узел для j-ого, который был разбит по атрибуту l, $N_{l,j}$, $N_{l,j,i}$ – количество примеров в соответствующих узлах, $n_{l,j}$ – количество дочерних узлов для j-ого родительского.

Узел *Карта Кохонена*

В данном узле реализован алгоритм построения и обучения нейронной сети Кохонена.

Самоорганизующиеся карты признаков (self organizing map — SOM) позволяют представлять результаты кластеризации в виде двумерных карт, где расстояния между объектами соответствуют расстояниям между их векторами в многомерном пространстве, а сами значения признаков отображаются различными цветами и оттенками [15, 16]. В основе такой карты в Deductor Studio лежит нейронная сеть Кохонена, которая была впервые предложена финским ученым Тайво Кохоненом в 1982 г. [7]. Такая сеть состоит из имеющих взвешенные друг с другом связи двух слоев нейронов: входного и выходного. Полученная модель способна решать задачи кластеризации и классификации.

Вычисления

Описание модели

Кластеризация нейронной сетью Кохонена проходит в два этапа. Во-первых, происходит обучение сети Кохонена и построение соответствующей ей карты, в результате чего записи распределяют по ячейкам. На втором шаге полученные ячейки объединяются в кластеры алгоритмом k-means или G-means.

Обучение сети Кохонена и построение карты

В данном узле сеть Кохонена построена по такой же структуре и обучается по тому же алгоритму, как описано в источнике [14].

Настройки узла

Нормализация и кодирование

Для полей с дискретным видом данных доступен нормализатор **Уникальные значения**, а с непрерывными – **Линейный**.

Подробнее про нормализаторы можно посмотреть в разделе Нормализаторы и денормализаторы.

При активном флаге Установить значимость поля при нахождении расстояния между объектами учитывается так же и указанная значимость. Таким образом, формула расстояния между векторами признаков $\mathbf{x}$ и $\mathbf{y}$ имеет следующий вид:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}\left(\frac{z_i}{100}(x_i - y_i)\right)^2},$$

где z_i – установленное значение значимости. В случае если флаг Установить значимость поля не активен z_i равен 100.

Настройка карты

Пользователем задаются следующие параметры, которые используются для построения карты.

Размер по оси X. Задает количество ячеек по горизонтали.

Размер по оси Y. Задает количество ячеек по вертикали.

Форма ячеек: прямоугольные или шестиугольные. Задает многоугольник, которым будет отображаться отдельная ячейка карты.

Параметры обучения карты

Способ начальной инициализации карты. В узле возможны три способа инициирования начальных весов:

- случайными значениями – начальные веса нейронов будут случайными значениями.
- из обучающего множества – в качестве начальных весов будут использоваться случайные примеры из обучающего множества.
- из собственных векторов (по умолчанию) – начальные веса нейронов карты будут проинициализированы значениями подмножества гиперплоскости, через которую проходят два главных собственных вектора матрицы ковариации входных значений обучающей выборки.

Скорость обучения:

- в начале обучения (по умолчанию 0,3);
- в конце обучения (по умолчанию 0,005).

Исходя из указанных значений, текущее значение скорости обучении на момент текущей эпохи T будет рассчитываться по формуле:

$$V = V_{start} \cdot \left(\frac{V_{end}}{V_{start}} \right)^{\frac{T}{T_{\max}}},$$

где V_{start}, V_{end} – скорости обучения в начале и в конце соответственно, T_{max} – максимальное количество эпох (задается в параметрах остановки обучения).

Радиус обучения:

- в начале обучения (по умолчанию 3);
- в конце обучения (по умолчанию 0,1).

Текущий радиус обучения в эпоху T вычисляется по формуле:

$$R = R_{start} \cdot \left(\frac{R_{end}}{R_{start}} \right)^{\frac{T}{T_{\max}}},$$

где R_{start}, R_{end} – радиусы обучения в начале и в конце соответственно.

Параметр **Функция соседства** определяет, какие нейроны и в какой степени будут считаться соседними по отношению к нейрону-победителю. Этот параметр может принимать два значения: Ступенчатая и Гауссова.

Если функция соседства Ступенчатая, то «соседями» для нейрона-победителя будут считаться все нейроны, линейное расстояние на 2-х мерной карте до которых не больше текущего радиуса обучения. При этом варианте функции соседства процесс обучения происходит немного быстрее, но качество результата может быть немного хуже, чем, если бы использовалась Гауссова функция соседства.

Если используется Гауссова функция соседства, то «соседями» для нейрона- победителя будут считаться все нейроны карты, но в разной степени полноты. При этом степень соседства определяется следующей функцией:

$$h = e^{-\frac{d^2}{2r}},$$

где h – значение, определяющее степень соседства, d – линейное расстояние от нейрона победителя до нейрона «соседа», r – текущий радиус обучения.

Кластеризация ячеек карты имеет следующие параметры настройки.

Автоматически определять количество кластеров (по умолчанию флажок стоит) – после установки этого флажка программа будет автоматически определять количество кластеров.

Уровень значимости (по умолчанию 0,1) – параметр автоматического определения кластеров. Чем больше этот параметр, тем большее количество кластеров будет получено. Со статистической точки зрения **Уровень значимости** представляет собой вероятность справедливости нулевой гипотезы о том, что значения в имеющемся наборе данных распределены по нормальному закону. Данный параметр используется для выделения кластеров алгоритмом G-means, описание которого приведено ниже.

Фиксированное количество кластеров (по умолчанию 7) – параметр доступный при ручном определении количества кластеров. Собственно задает желаемое количество кластеров, на которое будут разбиты нейроны карты Кохонена.

Параметры остановки обучения

Ошибка, меньше которой пример считается распознанным. Задается в относительных единицах (нормализованное значение). По умолчанию 0,05. Критерием останова в данном случае является условие, что рассогласование между эталонным и реальным выходом карты становится меньше заданного значения.

За ошибку принято расстояние от вектора признаков примера $\mathbf{x}$ до вектора признаков ближайшей ячейки $\mathbf{u}$:

$$Error = d(\mathbf{x}, \mathbf{u}).$$

Пользователь может установить следующие варианты окончания обучения:

- по достижению эпохи (по умолчанию 500);
- если средняя ошибка на обучающем множестве меньше заданного пользователем значения (по умолчанию отключено);
- если максимальная ошибка на обучающем множестве меньше заданного пользователем значения; (по умолчанию отключено);
- если процент распознанных примеров на обучающем множестве больше заданного пользователем значения (по умолчанию отключено);
- если средняя ошибка на тестовом множестве меньше заданного пользователем значения (по умолчанию отключено);
- если максимальная ошибка на тестовом множестве меньше заданного пользователем значения (по умолчанию отключено);
- если процент распознанных примеров на тестовом множестве больше заданного пользователем значения (по умолчанию отключено).

Визуализаторы

Карта Кохонена

Каждому нейрону соответствует свой вектор признаков, в зависимости от которого соответствующая ячейка на карте будет иметь свой цвет.

Поля с данными непрерывного вида. Пусть значения данного поля принадлежат диапазону $[a, b]$. Тогда цвет ячейки рассчитывается следующим образом.

Диапазон $[a, b]$ разбивается на 4 промежутка, границам которых соответствует свой цвет в кодировке RGB (Таблица 3).

Таблица 3 – Базовые цвета шкалы

i	x_i	Красный цвет (R)	Зеленый цвет (G)	Синий цвет (B)
0	a	99	99	255
1	$a + (b - a)\cdot 0{,}25$	99	255	255
2	$a + (b - a)\cdot 0{,}5$	99	255	99
3	$a + (b - a)\cdot 0{,}75$	255	255	99
4	b	255	99	99

Пусть новая точка x_{new} лежит между границами i и $(i - 1)$, то есть справедливо двойное неравенство $x_{i-1} < x_{new} < x_i$. Тогда насыщенность красного, зеленого и синего цветов будет рассчитываться соответственно по следующим формулам:

$$R_{x_{new}} = R_{i-1} + \left(R_i - R_{i-1}\right)\cdot\frac{x_{new} - x_{i-1}}{x_i - x_{i-1}},$$

$$G_{x_{new}} = G_{i-1} + \left(G_i - G_{i-1}\right)\cdot\frac{x_{new} - x_{i-1}}{x_i - x_{i-1}},$$

$$B_{x_{new}} = B_{i-1} + \left(B_i - B_{i-1}\right)\cdot\frac{x_{new} - x_{i-1}}{x_i - x_{i-1}}.$$

Поля с данными дискретного вида. Каждому уникальному значению назначается соответствующий ему на карте цвет.

Построение карты **проекция Саммона** проходит по алгоритму, описанному в источнике [8].

Профили кластеров

Описание приведено ниже (смотрите раздел Профили кластеров).

Узел Линейная регрессия

В данном узле реализован алгоритм построения множественной линейной регрессии, используемой для моделирования зависимостей между непрерывной выходной переменной и набором входных переменных. Если на вход алгоритма подается 1 поле, то имеем парную регрессионную модель.

Модель линейной регрессии в общем случае имеет следующий вид:

$$f(\boldsymbol{\alpha}, \mathbf{x}) = y = \alpha_0 + \alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 + \ldots + \alpha_n \cdot x_n.$$

В узле решаются задачи регрессии и классификации. Задача классификации решается путем доработки стандартного алгоритма линейной регрессии.

Вычисления

В данном узле реализован стандартный алгоритм расчета коэффициентов регрессии методом наименьших квадратов, не претерпевший изменений, и его можно найти в источниках [18, 19].

Для расчета ошибок используется следующие формулы. Средняя ошибка для примеров с k выходными полями:

$$ME = \frac{1}{kN} \sum_{i=1}^{N} \sum_{j=1}^{k} (\hat{y}_{ij} - y_{ij})^2 ,$$

где $\hat{y}_{ij} = f(\boldsymbol{\alpha}, \mathbf{x}_{ij})$ – значение, выдаваемое регрессией, y_{ij} – эталонное значение, i – номер примера в соответствующей выборке, N – количество примеров.

Максимальная ошибка:

$$MaxErr = \max_i \left(\frac{1}{k} \sum_{j=1}^{k} (\hat{y}_{ij} - y_{ij})^2 \right).$$

Если в наборе данных присутствуют категориальные поля, то результат расчета коэффициентов регрессии зависит от выбранных нормализаторов.

При использовании нормализатора **Уникальные значения** для выходного поля диапазон разбивается на соответствующие отрезки (путем назначения порогов отсечения), которые будут соответствовать определенному классу.

При использовании нормализатора **Битовая маска** результирующая регрессионная модель получается путем построения нескольких линейных регрессий, каждая из которых будет соответствовать одному из выходных факторов.

Если используется нормализатор **Позиция бита**, то сигнал с регрессии, выдавшей максимальное значение обработке примера, преобразуется в 1, со всех остальных в 0.

В случае использования нормализатора **Комбинация битов** для каждой регрессии выбирается порог отсечения, ниже которого значения преобразуются в 0, а иначе в 1.

При назначении нескольких выходных полей число регрессий в результирующей модели будет равно количеству выходных полей при выбранных нормализаторах.

$$y_i = f(\boldsymbol{\alpha}_i, \mathbf{x}),$$

где $1 \le i \le k$.

Подбор α происходит методом наименьших квадратов, суть которого заключается в нахождении такого вектора коэффициентов $\boldsymbol{\alpha}^*$, который бы минимизировал суммарную ошибку модели на всем обучающем множестве X^N:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} E(\boldsymbol{\alpha}, X^N), \text{ где } E(\boldsymbol{\alpha}, X^N) = \sum_{i=1}^{N} (f(\boldsymbol{\alpha}, x_i) - y_i)^2 .$$

Настройки узла

Настройка назначения столбцов

Все значения полей с данными непрерывного вида не подвергаются никакой нормализации.

Для входных полей с дискретным видом данных быть использованы нормализаторы **Битовая маска**, **Уникальные значения**.

Для выходных полей с дискретным видом данных доступны нормализаторы **Битовая маска**, **Уникальные значения**, **Опорная точка** (подробнее про способы кодирования смотрите раздел Нормализаторы). В этом случае будет решаться задача классификации путем построения ансамбля регрессионных моделей (см. выше).

Настройка ограничения диапазона выходных значений

В настройках данного узла есть возможность ограничить диапазон выходных значений как абсолютными значениями минимума и максимума, так и относительными. Основными параметрами здесь являются **Максимум** и **Минимум** диапазона.

Пусть Z_{min}, Z_{max} – минимальное и максимальное значения диапазона соответственно, $f(\mathbf{x})$ – функция, полученная после построения регрессии, тогда итоговые значения y будут находиться следующим образом:

$$y = \begin{cases} Z_{\min}, f(\mathbf{x}) < Z_{\min}, \\ f(\mathbf{x}), Z_{\min} \le f(x) \le Z_{\max}, \\ Z_{\max}, f(\mathbf{x}) > Z_{\max}. \end{cases}$$

Узел *Логистическая регрессия*

В данном узле строится модель бинарной логистической регрессии, которую в общем виде можно представить следующим образом:

$$P(\mathbf{x}) = \frac{1}{1 + e^{-g(\mathbf{x})}},$$

где $g(\mathbf{x}) = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$, $P(\mathbf{x})$ – вероятность наступления положительного исхода.

При анализе данных часто встречаются задачи, где выходная переменная является категориальной и тогда использование линейной регрессии затруднено. Поэтому при поиске связей между набором входных переменных и категориальной выходной переменной получила распространение логистическая регрессия [15, 17].

В данном узле решается задача бинарной классификации.

Вычисления

Используемый алгоритм построения логистической регрессии описан в источнике [17].

Подбор коэффициентов $b_0, b_1 \ldots b_n$ происходит по принципу максимизации логарифма произведения функций правдоподобия:

$$L(\mathbf{b}, X^n) = \ln \prod_{i=1}^{n} p(x_i, y_i) \to \max_{\mathbf{b}}.$$

Для поиска максимума функции правдоподобия используется метод Ньютона.

Начальные приближения для всех коэффициентов, кроме константы, равны 0. Для константы оно рассчитывается по формуле:

$$const = \ln\left(\frac{N(1)}{N(0)}\right),$$

где $N(0), N(1)$ – количество записей с выходным полем равным 0 и 1 соответственно.

Настройки узла

Настройка назначений столбцов

Выходным столбцом может быть только бинарное поле (то есть имеющее два уникальных значения). Кодируется он нормализатором **Уникальные значения** (смотрите раздел Нормализаторы).

Для входных полей с дискретным видом данных могут быть использованы нормализаторы **Битовая маска**, **Уникальные значения**, а с непрерывным – нормализация производится по следующей формуле:

$$x^* = \frac{x - \bar{x}}{\sigma_x},$$

где $\bar{x}$ – среднее значение, σ_x – среднеквадратическое отклонение.

Построение логистической модели

Максимальное число итераций. Если алгоритм не остановился ранее, то это произойдет по достижению заданного количества итераций.

Точность функции оценки. Алгоритм расчета коэффициентов завершится, когда очередное значение логарифмической функции правдоподобия $-2\ln(p(\mathbf{x}))$ прекратит изменяться в пределах заданной точности.

Порог отсечения. Задача бинарной классификации будет решена на основе заданного порога отсечения для поля со значением рейтинга. По умолчанию 0,5.

Считать пример распознанным, если ошибка меньше. По умолчанию 0,05.

Визуализатор *ROC-кривая*

Введем обозначения:

- TP (True Positives) – верно классифицированные положительные примеры (так называемые истинно положительные случаи);
- TN (True Negatives) – верно классифицированные отрицательные примеры (истинно отрицательные случаи);
- FN (False Negatives) – положительные примеры, классифицированные как отрицательные (ошибка I рода). Это так называемый «ложный пропуск» – когда интересующее нас событие ошибочно не обнаруживается (ложноотрицательные примеры);
- FP (False Positives) – отрицательные примеры, классифицированные как положительные (ошибка II рода); Это ложное обнаружение, т.к. при отсутствии события ошибочно выносится решение о его присутствии (ложноположительные случаи).

Алгоритм построения ROC-кривой.

Входы: L – множество примеров; $f[i]$ – рейтинг, полученный моделью, или вероятность того, что i-й пример имеет положительный исход; min и max – минимальное и максимальное значения, возвращаемые f; dx – шаг; P и N – количество положительных и отрицательных примеров соответственно.

1 $t = min$
2 повторять
3 $FP = TP = 0$
4 для всех примеров i принадлежит L {
5 если $f[i] \geq t$ тогда // этот пример находится за порогом
6 если i положительный пример тогда
7 { $TP = TP + 1$ }
8 иначе // это отрицательный пример
9 { $FP = FP + 1$ }
10 }
11 $Se = \frac{TP}{P} \cdot 100\%$

12 $(100 - Sp) = \frac{FP}{N}$ // расчет (100 минус Sp)

13 Добавить точку $(100 - Sp, Se)$ в ROC кривую

14 $t = t + dx$

15 пока $(t > max)$

Пусть ROC-кривая построена на N точках, упорядоченных по возрастанию x_i: $\{(x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N)\}$, где $x_i = 100 - Sp_i$, $y_i = Se$, тогда площадь под кривой находится по формуле:

$$UAC = \sum_{i=2}^{N} (x_i - x_{i-1}) \cdot y_i \ .$$

Подробнее про ROC-кривую можно узнать в [28].

Узел *Кластеризация*

В данном узле реализованы алгоритмы кластеризации k-means и G-means. Модели, построенные при помощи данного узла способны решать задачи кластеризации и классификации.

Вычисления

Описание алгоритма k-means можно найти в источнике [15, стр. 311], алгоритма G-means в [10].

Алгоритм G-means

Входные параметры: X – исходный набор данных, α – уровень значимости.

1. Инициализировать C как множество центров кластеров средними значениями.
2. $C \leftarrow kmeans(C, X)$.
3. $\{x_i \mid cluster\{x_i\} = j\}$ набор точек для кластера с центром c_j.
4. Статистическим тестом проверить гипотезу о том, что значения в каждом кластере j распределены по гауссовскому закону с уровнем значимости α.
5. Если распределение гауссовское, то запомнить кластер, иначе заменить c_j двумя центрами.
6. Повторять, начиная с шага **2**, до тех пор, пока количество центров не перестанет увеличиваться.

Настройки узла

Настройка нормализации

Для полей с дискретным видом данных доступны нормализаторы **Битовая маска**, **Уникальные значения**, а с непрерывным видом данных используется **Линейный нормализатор**.

Способы и параметры нормализации и кодирования описаны в разделе Нормализаторы.

При активном флаге Установить значимость поля при нахождении расстояния между объектами учитывается так же и указанная значимость. Таким образом, формула расстояния между векторами признаков $\mathbf{x}$ и $\mathbf{y}$ имеет следующий вид:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n} \frac{z_i}{100}\left(x_i - y_i\right)^2},$$

где z_i – установленное значение значимости. В случае если флаг Установить значимость поля не активен z_i равен 100.

Настройка параметров кластеризации

Уровень значимости, % (если выбрано автоматическое определение кластеров). Данный параметр используется алгоритмом G-means. Со статистической точки зрения **Уровень значимости** представляет собой вероятность справедливости нулевой гипотезы о том, что значения в имеющемся наборе данных распределены по нормальному закону.

Количество кластеров (при выборе фиксированного количества кластеров). Данный параметр используется алгоритмом k-means и задает требуемое количество кластеров.

Визуализаторы

Профили кластеров

Значимость атрибутов показывает их степень влияния на образования того или иного кластера.

Поля с данными непрерывного вида

Для таких полей значимость для каждого атрибута в отдельном кластере рассчитывается по t-критерию Студента.

Пусть N_1, N_2 – размеры выборок, M_1, M_2 – соответствующие им математические ожидания, S_1, S_2 – среднеквадратичные отклонения. Если каждая из выборок содержит 30 примеров и более, значение *t*-критерия Студента рассчитывается по следующей формуле:

$$t = \frac{M_1 - M_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}}.$$

В случае если хотя бы одна из выборок содержит менее 30 примеров, расчетная формула принимает вид:

$$t = \frac{M_1 - M_2}{\sqrt{\frac{VC}{N_1} + \frac{VC}{N_2}}},$$

где $VC = \frac{(N_1 - 1)\cdot V_1 + (N_2 - 1)\cdot V_2}{N_1 + N_2 - 2}$.

Вычислив *t*-критерий, полагая, что число степеней свободы $d = N_1 + N_2 - 2$, рассчитывается функция распределения Стьюдента, с помощью которой находится значимость:

$$Значимость(t) = \left[1 - I_{\frac{d}{d+t}}\left(\frac{d}{2}, \frac{1}{2}\right)\right]\cdot 100\%,$$

где I – неполная бета-функция.

Общая значимость атрибута по всем кластерам находится следующим образом.

1. Рассчитывается оценка дисперсии по разбросу **между** кластерами: $MS_B = \frac{\sum_j (\bar{x}_j - \bar{x}_G)^2 n_j}{k - 1}$, где $\bar{x}_G$ – среднее значение по всему набору данных, $\bar{x}_j$ – среднее значение в кластере, n_j – количество примеров в кластере j, k – количество кластеров.
2. Рассчитывается оценка дисперсии по разбросу **внутри** кластеров: $MS_W = \frac{\sum_{j=0}^{k-1} S_j}{k}$, где S_j – СКО внутри кластера.

3 Находится значение F-критерия по следующей формуле: $F = \frac{MS_B}{MS_W}$.

4 Вычисляется значимость, используя функцию распределения Фишера. Для этого используется следующая формула: $Значимость(F) = \left[1 - I_{\frac{d_2}{d_2 + d_1 F}}\left(\frac{d_1}{2}, \frac{d_2}{2}\right)\right] \cdot 100\%$, где $d_1 = k - 1$, $d_2 = n_G - k$, I – неполная бета-функция.

Поля с данными дискретного вида

В данной ситуации для каждого дискретного поля рассчитывается значение χ^2 Пирсона. Значимость находится через функцию распределения χ^2 следующим образом:

$$Значимость(\chi^2) = \gamma\left(\frac{d}{2}, \frac{\chi^2}{2}\right),$$

где d – число степеней свободы, γ – неполная гамма-функция.

Полученная модель способна также решать задачу классификации. Кластер присваивает метку класса исходя из большинства принадлежащих ему примеров.

Узел *Автокорреляция*

В данном узле происходит расчет функции автокорреляции.

В случае, когда изменение величины наблюдается во времени, то наблюдения в различные промежутки времени могут оказаться взаимосвязанными, или коррелированными. Эта корреляция измеряется с помощью коэффициента автокорреляции.

Целью автокорреляционного анализа является выяснение степени статистической зависимости между различными значениями (отсчетами) случайной последовательности, которую образует поле выборки данных. В процессе автокорреляционного анализа рассчитываются коэффициенты корреляции (мера взаимной зависимости) для двух значений выборки, находящихся друг от друга на определенном количестве отсчетов, называемые также лагом. Совокупность коэффициентов корреляции по всем лагам представляет собой автокорреляционную функцию ряда (АКФ):

$R(t) = corr(X(t), X(t + k))$, где $k > 0$ – целое число (лаг).

Подробнее по автокорреляции можно найти в источнике [21].

Вычисления

Расчет коэффициента корреляции происходит по следующей формуле:

$$r_k = \frac{\sum_{t=k+1}^{n} (Y_t - \overline{Y})(Y_{t-k} - \overline{Y})}{\sum_{t=1}^{n} (Y_t - \overline{Y})^2},$$

где r_k – коэффициент автокорреляции для запаздывания на k периодов, $\overline{Y}$ - среднее значение ряда, Y_t – наблюдение в момент времени t, Y_{t-k} – наблюдение на k периодов ранее, то есть в момент времени $t - k$.

Литература

Статьи

1 Kalman, R. E., A New Approach to Linear Filtering and Prediction Problems, Transaction of the ASME — Journal of Basic Engineering, pp. 35-45 (March 1960).

2 Brown R. G., Hwang P. Y. C. Introduction to Random Signals and Applied Kalman Filtering, Second Edition, John Wiley & Sons, Inc. 1992.

3 Agrawal R., Imielinski T., Swami A. N., Mining Association Rules between Sets of Items in Large Databases, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data

4 Rakesh Agrawal and Ramakrishnan Srikant, Fast Algorithms for Mining Association Rules, Proc. 20th Int. Conf. Very Large Data Bases (VLDB), 1994.

5 Rumelhart D.E., Hinton G.E., Williams R.J., Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986.

6 M. Riedmiller, H.Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. San Francisco,1993.

7 Kohonen, T. Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69, 1982

8 Sammon JW. A nonlinear mapping for data structure analysis". IEEE Transactions on Computers 18: 401–409, 1969.

9 Губанов В.А. Выделение тренда из временных рядов макроэкономических показателей // Научные труды: Институт народнохозяйственного прогнозирования РАН, 2005 – с. 24-40.

10 Hamerly G., Elkan C. Learning the k in k-means // In Proc. 17th NIPS, 2003.

11 Yang, Y., Guan, H., You. J. CLOPE: A fast and Effective Clustering Algorithm for Transactional Data In Proc. of SIGKDD'02, July 23-26, 2002, Edmonton, Alberta, Canada.

12 Wang, K., Xu, C.. Liu, B. Clustering transactions using large items. In Proc. CIKM'99, Kansas, Missouri, 1999.

Книги и учебные пособия

13 Quinlan, J.R. C4.5 Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann, 1993

14 Kohonen, T. Self-Organizing Maps, 3rd ed. New York: Springer-Verlag, 2001

15 Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям (+CD): Учеб. Пособие. 2-е изд., перераб. и доп. – СПб.: Питер, 2010. – 704 с.: ил.

16 Зиновьев А.Ю. Визуализация многомерных данных. — Красноярск: Изд-во КГТУ, 2000.

17 Hosmer D. W., Lemeshow S. Applied Logistic Regression (Second Edition). — Wiley Publishing, Inc., 2000.

18 Косоруков И. А. Методы количественного анализа в бизнесе: Учебник. — М.: Инфра-М, 2005.

19 Писарева О. М. Методы прогнозирования развития социально-экономических систем. — М.: Высшая школа, 2007.

20 Эконометрика: Учебник/И.И. Елисеева, С.В. Курышева, Т.В. Костеева и др.; Под ред. И.И. Елисеевой. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. – 576.: ил.

21 Ханк Д.Э., Уичерн Д.У., Райтс А.Дж. Бизнес-прогнозирование, 7-е издание.: Пер. с англ. — М.: Издательский дом «Вильямс», 2003. — 656 с.: ил. — Парал. тит. англ.

22 Айфичер Эммануил С., Джервис Барри У. Цифровая обработка сигналов: практический подход, 2-е издание. : Пер. с англ. — М.: Издательский дом «Вильямс», 2004. — 992 с.

23 Шеффе Г. Дисперсионный анализ. — М.: Наука. Главная редакция физико-математической литературы, 1980

Интернет-источники

24 Спектральный анализ (Глоссарий BaseGroup Labs)
http://www.basegroup.ru/glossary/definitions/spectral_analysis/

25 Преобразование Фурье (статья в википедии)
http://ru.wikipedia.org/wiki/Преобразование_Фурье

26 Некипелов Н. Калмановская фильтрация
http://www.basegroup.ru/library/cleaning/kalmanfilter/

27 Статьи по ассоциативным правилам на сайте BaseGroup Labs
http://www.basegroup.ru/library/analysis/association_rules/

28 Паклин Н.Б. Логистическая регрессия и ROC-анализ – математический аппарат
http://www.basegroup.ru/library/analysis/regression/logistic/

29 Сенин А.В. Методы отбора переменных в регрессионные модели
http://www.basegroup.ru/library/analysis/regression/feature_selection/

30 Глоссарий BaseGroup Labs. Тест Вальда
http://www.basegroup.ru/glossary/definitions/wald_test/

31 Паклин Н.Б. Кластеризация категорийных данных: масштабируемый алгоритм CLOPE
http://www.basegroup.ru/library/analysis/clusterization/clope/

32 Киселев А. Вейвлет своими руками http://www.basegroup.ru/library/cleaning/making_wavelet/

Контакты

Адрес:

Россия, 390046, г.Рязань, ул.Введенская 115, оф. 447, этаж 4.

Здание «Нефтехиммашсистемы»

Телефон: +7 (4912) 24-09-77

24-06-99, 25-83-97

Факс: +7 (4912) 24-09-77

E-mail:

info@basegroup.ru – общая информация

sale@basegroup.ru – служба продаж

deductor@basegroup.ru – служба поддержки Deductor

education@basegroup.ru – обучение и тренинги