BaseGroup™ Labs
ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ

Deductor

Руководство аналитика

Версия 5.2

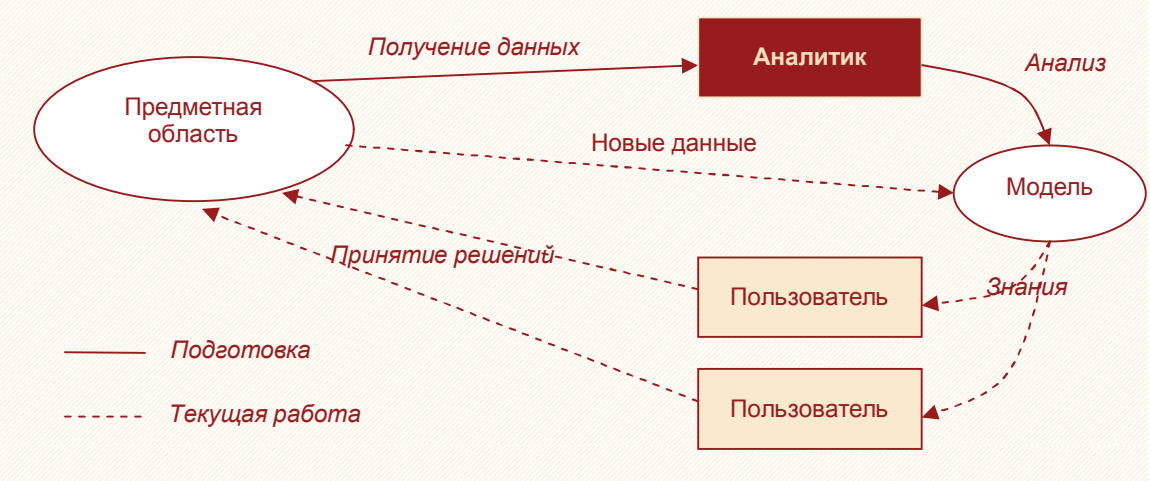
В руководстве описана аналитическая платформа Deductor 5.2: идеология анализа данных, реализованные механизмы, составные части и архитектура, демонстрируются типовые задачи анализа бизнес-данных и способы их решения при помощи Deductor Studio. Книга предназначена для аналитиков, руководителей подразделений и других специалистов, которым необходимо применение в работе современных методов анализа. Специальных знаний в области анализа данных не требуется, но предполагается, что читатель знаком с базовыми сведениями вузовского курса высшей математики и является квалифицированным пользователем компьютера.

Содержание

Введение

Анализ информации является неотъемлемой частью ведения бизнеса и одним из важных факторов повышения его конкурентоспособности. При этом в подавляющем большинстве случаев анализ сводится к применению одних и тех же базовых механизмов. Они являются универсальными и применимы к любой предметной области, благодаря чему имеется возможность создания унифицированной программной платформы, в которой реализованы основные механизмы анализа, такой как Deductor.

Обычно анализ производят аналитики и эксперты предметной области предприятия. Они подготавливают данные к пригодному для анализа виду, применяют к ним различные методы анализа, приводят результаты к легко воспринимаемому виду. Результаты анализа необходимы лицам предприятия, принимающим решения, например, руководителям отделов, менеджерам. Они могут совершенно не разбираться в методах анализа, но у них есть потребность в их результатах. Таким образом, требуется, с одной стороны, выделить и формализовать знание эксперта о предметной области, с другой, обеспечить возможность использовать эти знания человеком, не разбирающимся в особенностях использования механизмов анализа, т.е. решить проблему тиражирования знаний (см. рисунок).



Deductor 5 предназначен для эффективного решения проблемы тиражирования знаний. Deductor – это аналитическая платформа, основа для создания законченных прикладных решений в области анализа данных. Реализованные в Deductor технологии позволяют на базе единой архитектуры пройти все этапы построения аналитической системы: от создания хранилища данных до автоматического подбора моделей и визуализации полученных результатов.

Deductor 5 состоит из пяти частей:

1. *Warehouse* – хранилище данных, консолидирующее информацию из разных источников. В системе поддерживается концепция виртуальных хранилищ данных;
2. *Studio* – аналитическое приложение, позволяющее пройти все этапы построения прикладного решения;
3. *Viewer* – рабочее место конечного пользователя, одно из средств тиражирования знаний;
4. *Server* – служба, обеспечивающая удаленную аналитическую обработку данных;
5. *Client* – клиент доступа к Deductor Server. Обеспечивает доступ к серверу из сторонних приложений и управление его работой.

Deductor 5 содержит большое количество методов подготовки, трансформации, обработки и визуализации данных.

В этом руководстве речь пойдет о применении Deductor при решении задач анализа данных. Руководство рассчитано на аналитика, занимающегося практическими вопросами анализа

информации на основе платформы Deductor. Оно требует от читателя владения лишь базовыми основами анализа и используемых в его процессе математических методов.

Руководство имеет следующую структуру.

В главе **«Анализ данных – основные принципы»** описываются общие вопросы анализа данных, базовые подходы и методики проведения анализа, рассматривается место аналитической системы в анализе данных.

В главе «**Состав и назначение аналитической платформы Deductor»** рассматриваются основные возможности, область применения и задачи, решаемые с использованием платформы Deductor.

Архитектура составных частей платформы – аналитического приложения Deductor Studio и хранилища данных Deductor Warehouse – описываются в разделах **«Архитектура Deductor Studio – аналитическое приложение»** и **«Архитектура Deductor Warehouse – многомерное хранилище данных»**.

Одним из важных методов представления данных и проведения оперативного анализа является технология OLAP. Ее основы и реализация в программах Deductor рассматриваются в главе **«Работа с OLAP-кубом»**.

Большое внимание в Руководстве уделено описанию разнообразных алгоритмов анализа, реализованных в Deductor. Для каждого алгоритма описаны принцип работы, исходные данные и получаемые результаты, доступные настройки и, кроме того, приводятся примеры их практического использования. Вся эта информация сгруппирована в главе **«Описание аналитических алгоритмов»**.

В главе «Интерпретация результатов» рассказано о способах интерпретации построенных моделей, возможностях работы с ними и механизмах оценки их качества.

Одним из важнейших аспектов анализа является сбор исходных данных. Вопросы определения важности данных для анализа, объемов выборки и представления данных рассматриваются в главе **«Подготовка данных для анализа»**.

В главе **«Пример создания законченного аналитического решения»** рассматривается процесс создания решения на базе платформы Deductor. В ней на конкретном примере подробно описываются все этапы, которые требуется пройти, начиная от постановки задачи и заканчивая подготовкой системы отчетности. Читатель Руководства будет ознакомлен со всеми аспектами применения платформы Deductor в анализе данных и сможет самостоятельно приступить к разработке аналитических моделей и готовых решений на их базе, а также их использованию на практике.

В главе **«Что делать при возникновении ошибок»** рассказывается о наиболее распространенных ошибках, с которыми сталкивается аналитик при создании решения и о способах борьбы с подобными ошибками.

В конце руководства приведен список литературы, наиболее активно использовавшейся в ходе разработки платформы Deductor. В ней рассматриваются как общие вопросы Data Mining и Knowledge Discovery, так и более узкие темы – нейронные сети, статистические метода анализа, генетические и нечеткие алгоритмы, подходы к решению отдельных проблем анализа (прогнозирование, кластеризация, факторный, корреляционный и другие виды анализа). Кроме того, к списку литературы добавлены несколько полезных ссылок на сайты в Internet, имеющие непосредственное отношение к анализу данных.

Анализ данных – основные принципы

Два подхода к анализу данных

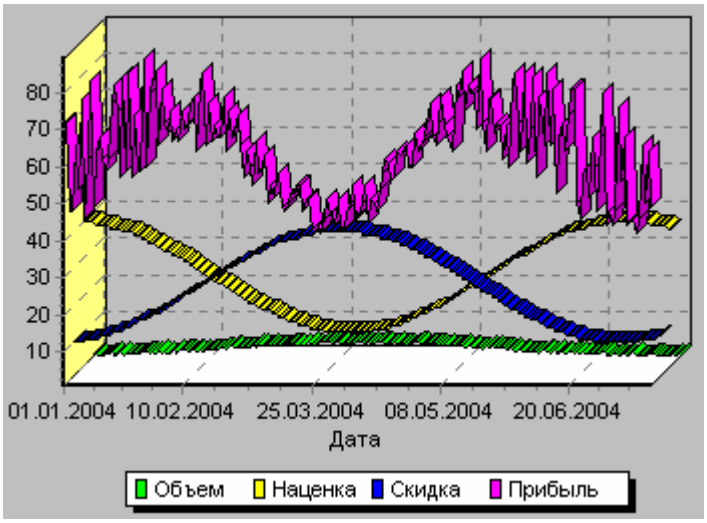
Любая организация в процессе своей деятельности стремится повысить прибыль и уменьшить расходы. В этом ей помогают новые компьютерные технологии, использование разнообразных программ автоматизации бизнес-процессов. Это учетные, бухгалтерские и складские системы, системы управленческого учета и многие другие. Чем аккуратнее и полнее ведется сбор и систематизация информации, тем полнее будет представление о процессах в организации. Современные носители информации позволяют хранить десятки и сотни гигабайт информации, но без использования специальных средств анализа накопленной информации такие носители превращаются просто в свалку бесполезных сведений. Очень часто принятие правильного решения затруднено тем, что хотя данные и имеются, они являются неполными, или, наоборот, избыточными, замусорены информацией, которая вообще не имеет отношения к делу, несистематизированными или систематизированными неверно. Тогда прибегают к помощи программных средств, которые позволяют привести информацию к виду, который дает возможность с достаточной степенью достоверности оценить содержащиеся в ней факты и повысить вероятность принятия оптимального решения.

Есть два подхода к анализу данных с помощью информационных систем.

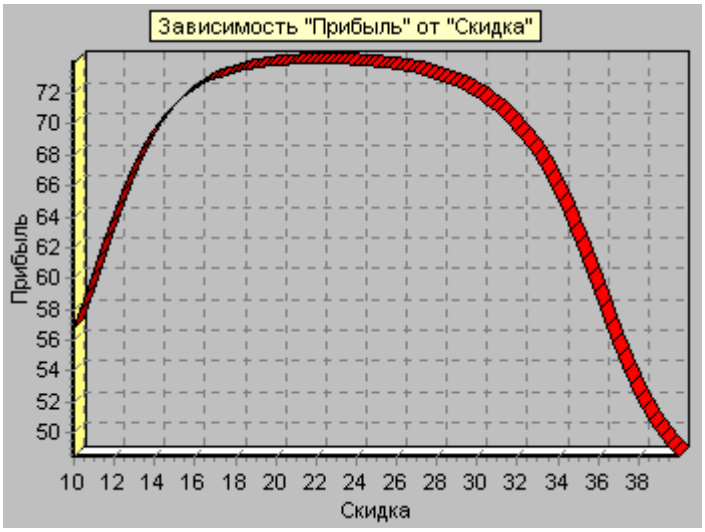
В первом варианте программа используется для *визуализации* информации - извлечения данных из источников и предоставления их человеку для самостоятельного анализа и принятия решений. Обычно данные, предоставляемые программой, являются простой таблицей, и в таком виде их очень сложно анализировать, особенно если данных много, но имеются и более удобные способы отображения: кубы, диаграммы, гистограммы, карты, деревья…

Второй вариант использования программного обеспечения для анализа – это *построение моделей*. Модель имитирует некоторый процесс, например, изменение объемов продаж некоторого товара, поведение клиентов и другое. Для построения модели необходимо сделать предобработку данных и далее к ним применять математические методы анализа: кластеризацию, классификацию, регрессию и т. д. Построенную модель можно использовать для принятия решений, объяснения причин, оценки значимости факторов, моделирования различных вариантов развития…

Рассмотрим пример. Предоставление скидки покупателям является стимулом для увеличения объемов закупок. Чем больше продается некоторого товара, тем больше прибыль. С другой стороны, чем больше предоставляется скидка, тем меньше наценка на товар и тем меньше прибыли приносят продажи этого товара. Пусть есть история продаж, представленная таблицей со столбцами: дата, объем продаж, скидка в процентах, наценка и прибыль. При проведении анализа «вручную» можно рассмотреть диаграмму.



На диаграмме видно, что при максимальной скидке прибыль минимальна. Максимум же прибыли достигается где-то посередине между минимальной и максимальной скидкой. Точный ответ об оптимальной скидке по этой диаграмме получить довольно трудно. Для этого можно воспользоваться другим подходом к анализу. Например, построить модель зависимости прибыли от скидки.



По этой зависимости легко определить, что максимум прибыли достигается при скидке 22%.

Как визуализация, так и построение моделей осуществляются путем применения к данным базовых методов анализа. Это достаточно известные методы, и они используются в самых разнообразных сферах деятельности.

Базовые методы анализа

Online Analytical Processing

Любая система поддержки принятия решений, прежде всего, должна обладать средствами отбора и предоставления пользователю данных в удобной для восприятия и анализа форме. Как правило, наиболее удобными для анализа являются многомерные данные, описывающие предметную область сразу с нескольких точек зрения. Для описания таких наборов данных вводится понятие многомерных кубов (гиперкубов, метакубов). По осям такого куба размещаются параметры, а в ячейках – зависящие от них данные. Вдоль каждой оси представлены различные уровни детализации данных. Использование такой модели данных позволяет повысить эффективность работы с ними: генерировать сложные запросы, создавать отчеты, выделять подмножества данных и т.д. Технология комплексного многомерного анализа данных и предоставления результатов этого анализа в удобной для использования форме получила название OLAP.

OLAP (Online Analytical Processing) – оперативная аналитическая обработка данных. OLAP дает возможность в реальном времени генерировать описательные и сравнительные сводки данных и получать ответы на различные другие аналитические запросы. OLAP-кубы представляют собой проекцию исходного куба данных на куб данных меньшей размерности. При этом значения ячеек агрегируются, то есть объединяются с применением функции агрегации – сумма, среднее, количество, минимум, максимум. Такие проекции или срезы исходного куба представляются на экране в виде кросс-таблицы.

Проиллюстрируем идею OLAP-куба на простом примере. Пусть информацию, хранящуюся в базе данных, или подмножество данных, получаемое в результате выполнения запроса, можно представить в виде следующей таблицы.

№ п/п	Город	Годовой объем продаж (руб.)			Итого
		Товар 1	Товар 2	Товар 3	
1	Москва	525 000	234 000	325 000	1 084000
2	Тула	224 000		12 560	236 560
3	Владимир		123 000	425 000	548 000
4	Самара	78 000	234 000	45 000	357 000
5	Тверь		45 000	78 000	123 000

Основываясь на данных из таблицы, можно дать ответы на несколько вопросов, которые могут возникнуть при анализе объемов продаж. Например, каков объем продаж каждого товара по одному из городов?

	Владимир
Товар 1	
Товар 2	123000
Товар 3	425000

Данную выборку можно интерпретировать как одномерную, поскольку объемы продаж расположены только вдоль одного измерения с наименованием товара.

Или: каков объем продаж каждого из товаров по городам?

Город	Товар 1	Товар 2	Товар 3
Москва	525 000	234 000	325 000
Тула	224 000		12 560
Владимир		123 000	425 000
Самара	78 000	234 000	45 000
Тверь		45 000	78 000

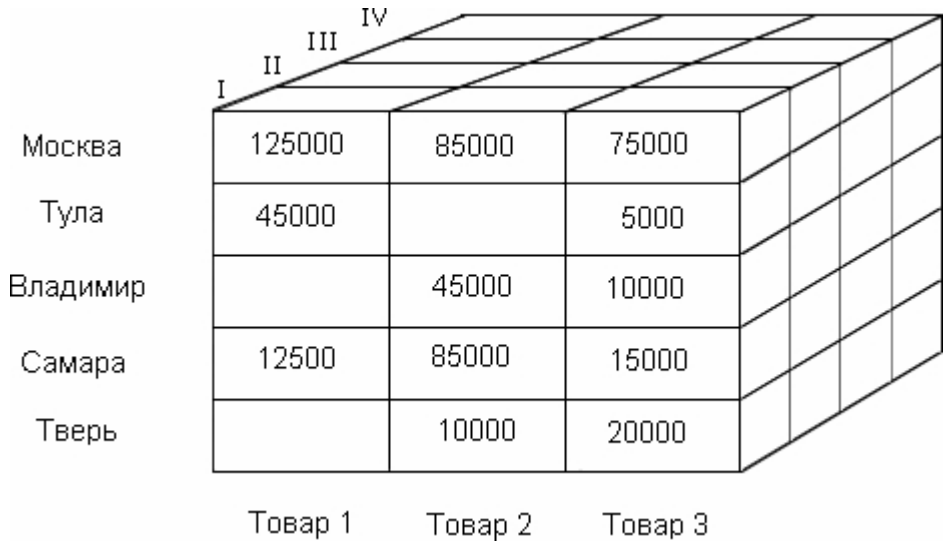
Данная выборка является двумерной и может быть представлена в виде «плоского куба», в котором 3-е измерение «отключено».

Москва	525000	234000	325000
Тула	224000		12560
Владимир		123000	425000
Самара	78000	234000	45000
Тверь		45000	78000
	Товар 1	Товар 2	Товар 3

Чтобы представленный куб превратился в трехмерный, необходимо добавить еще одно измерение данных, т.е. привлечь дополнительную информацию по анализируемым параметрам. Например, если в базе данных предусмотрена информация о квартальном объеме продаж каждого товара по всем городам, то номер квартала может стать этим дополнительным измерением.

Город	Поквартальный объем продаж (тыс. руб.)											
	Товар 1				Товар 2				Товар 3			
	I	II	III	IV	I	II	III	IV	I	II	III	IV
Москва	125,0	125,0	150,0	125,0	85,0	70,0	45,0	34,0	75,0	75,0	75,0	100,0
Тула	45,0	55,0	100,0	24,0					5,0	5,0		2,56
Владимир					45,0	40,0	25,0	13,0	10,0	10,0	10,0	12,5
Самара	12,5	15,5	32,0	18,0	85,0	65,0	50,0	34,0	15,0	5,0	17,0	13,0
Тверь					10,0	10,0	10,0	15,0	20,0	15,0	25,0	18,0

На основе подобной выборки можно построить трехмерный куб.



Такая модель представления данных позволяет получать нужную информацию, производя соответствующие сечения (срезы) OLAP-куба.

Нет необходимости пытаться произвести геометрическую интерпретацию OLAP-куба с размерностью более 3. Действительно, поскольку человеческое сознание «приспособлено» к восприятию трехмерной действительности, все прочие представления сложны для восприятия, тем более, что речь идет не о реальном, а об информационном пространстве. Само понятие «многомерный куб» есть не что иное, как служебный термин, используемый для описания метода.

В принципе, используемое число измерений может быть любым. Однако следует отметить, что задача с большим числом измерений, во-первых, является трудоемкой с точки зрения ее выполнения на ПК и, во-вторых, ее осмысление и интерпретация результатов аналитиком могут быть затруднены и даже приводить к ошибочным решениям. Поэтому с методической точки зрения сложные задачи, требующие анализа данных большой размерности, следует по возможности сводить к нескольким более простым. Аналогично, сложные таблицы, которые содержат большое количество полей и записей, являющихся трудными для чтения, восприятия и анализа, можно разбить на несколько более простых таблиц. Это сделает работу с ними намного более удобной.

Knowledge Discovery in Databases

KDD (Knowledge Discovery in Databases) – извлечение знаний из баз данных. Это процесс поиска полезных знаний в «сырых данных». KDD включает в себя вопросы подготовки данных, выбора информативных признаков, очистки данных, применения методов Data Mining (DM), постобработки данных и интерпретации полученных результатов.

Привлекательность этого подхода заключается в том, что вне зависимости от предметной области мы применяем одни и те же операции:

1. Подготовка исходного набора данных. Этот этап заключается в создании набора данных, в том числе слиянии сведений из различных источников, определение выборки, которая и будет в последствии анализироваться. Для этого должны существовать развитые инструменты доступа к различным источникам данных: файлам разных форматов, базам данных, учетным системам.
2. Предобработка и очистка данных. Для того чтобы эффективно применять методы анализа, следует обратить серьезное внимание на вопросы предобработки данных. Данные могут содержать пропуски, шумы, аномальные значения и т.д. Кроме того, данные могут быть избыточны, недостаточны и т.д. В некоторых задачах требуется дополнить данные некоторой априорной информацией. Наивно предполагать, что если подать любые данные на вход системы в существующем виде, то на выходе получим полезные знания. Данные должны быть качественны и корректны с точки зрения используемого метода анализа. Более того, иногда размерность исходного пространства может быть очень большой, и тогда желательно применение специальных алгоритмов понижения размерности: отбор наиболее значимых признаков и отображение данных в пространство меньшей размерности.
3. Трансформация данных. Для различных методов анализа требуются данные, подготовленные в специальном виде. Например, некоторые методы анализа в качестве входных полей могут использовать только числовые данные, а некоторые, наоборот, только категориальные.
4. Data Mining. На этом шаге применяются различные алгоритмы для нахождения знаний. Это нейронные сети, деревья решений, алгоритмы кластеризации и установления ассоциаций и т.д. Для этого могут использоваться как классические статистические методы, так и самообучающиеся алгоритмы и машинное обучение.
5. Постобработка данных. Тестирование, интерпретация результатов и практическое применение полученных знаний в бизнесе.

Описанный процесс повторяется итеративно, а реализация этих этапов позволяет автоматизировать процесс извлечения знаний.

Например, нужно сделать прогноз объемов продаж на следующий месяц. Есть сеть магазинов розничной торговли. Первым шагом будет сбор истории продаж в каждом магазине и объединение ее в общую выборку данных. Следующим шагом будет предобработка собранных данных. Например, их группировка по месяцам, сглаживание кривой продаж, устранение факторов, слабо влияющих на объемы продаж. Далее следует построить модель зависимости объемов продаж от выбранных факторов. Это можно сделать с помощью линейной регрессии или нейронных сетей. Имея такую модель, можно получить прогноз, подав на вход модели нашу историю продаж. Зная прогнозное значение, его можно использовать, например, для оптимизации закупок товара.

Data Mining

Data Mining (DM) – «добыча» данных. Это метод обнаружения в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных для интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. DM обеспечивает решение всего пяти задач — классификация, кластеризация, регрессия, ассоциация, последовательность:

1. Классификация — установление функциональной зависимости между входными и *дискретными* выходными переменными. При помощи классификации решается задача отнесение объектов (наблюдений, событий) к одному из заранее известных классов.
2. Регрессия – установление функциональной зависимости между входными и *непрерывными* выходными переменными. Прогнозирование чаще всего сводится к решению задачи регрессии.
3. Кластеризация — это группировка объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность объектов. Объекты внутри кластера должны быть «похожими» друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем больше похожи объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация.
4. Ассоциация — выявление зависимостей между связанными событиями, указывающих, что из события X следует событие Y. Такие правила называются ассоциативными. Впервые эта задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом потребительской корзины (market basket analysis).
5. Последовательные шаблоны — установление закономерностей между связанными во времени событиями. Например, после события X через определенное время произойдет событие Y.

Иногда специально выделяют задачу *анализа отклонений* — выявление наиболее нехарактерных шаблонов.

Примеры бизнес-задач, где применяются эти методы.

Классификация используется в случае, если заранее известны классы отнесения объектов. Например, отнесение нового товара к той или иной товарной группе, отнесение клиента к какой-либо категории. При кредитовании это может быть, например, отнесение клиента по каким-то признакам к одной из групп риска.

Регрессия чаще всего используется при прогнозировании объемов продаж, в этом случае зависимой величиной являются объемы продаж, а факторами, влияющими на эту величину, могут быть предыдущие объемы продаж, изменение курса валют, активность конкурентов и т.д. или, например, при диагностике оборудования, когда оценивается зависимость надежности от различных внешних факторов, показателей датчиков, износа оборудования.

Кластеризация может использоваться для сегментирования и построения профилей клиентов (покупателей). При достаточно большом количестве клиентов становится трудно подходить к

каждому индивидуально. Поэтому клиентов удобно объединить в группы – сегменты со сходными признаками. Выделять сегменты клиентов можно по нескольким группам признаков. Это могут быть сегменты по сфере деятельности, по географическому расположению. После сегментации можно узнать, какие именно сегменты являются наиболее активными, какие приносят наибольшую прибыль, выделить характерные для них признаки. Эффективность работы с клиентами повышается за счет учета их персональных или групповых предпочтений.

Ассоциации помогают выявлять совместно приобретаемые товары. Это может быть полезно для более удобного размещения товара на прилавках, стимулирования продаж. Тогда человек, купивший пачку спагетти, не забудет купить к ним бутылочку соуса.

Последовательные шаблоны могут быть использованы, например, при планировании продаж или предоставлении услуг. Например, если человек приобрел фотопленку, то через неделю он отдаст ее на проявку и закажет печать фотографий.

Для *анализа отклонений* необходимо сначала построить шаблон типичного поведения изучаемого объекта. Например, поведение человека при использовании кредитных карт. Тогда будет известно, что клиент (покупатель) использует карту регулярно два раза в месяц и приобретает товар в пределах определенной суммы. Отклонением будет, например, не запланированное приобретение товара по данной карте на большую сумму. Это может говорить об ее использовании другим лицом, то есть о факте мошенничества.

Перечисленные выше базовые методы анализа данных используются для создания аналитических систем. Причем, под такой системой понимается не только какая-то одна программа. Некоторые механизмы анализа могут быть реализованы на бумаге, некоторые на компьютере с использованием электронных таблиц, баз данных и других приложений. Однако, такой подход при частом использовании не эффективен. Намного лучшие результаты даст применение единого хранилища данных и единой программы, содержащей в себе всю функциональность, необходимую для реализации концепции KDD.

Состав и назначение аналитической платформы Deductor

Deductor состоит из пяти компонентов: аналитического приложения Deductor Studio, многомерного хранилища данных Deductor Warehouse, средства тиражирования знаний Deductor Viewer, аналитического сервера Deductor Studio и клиента для доступа к серверу Deductor Client.

Deductor Warehouse – многомерное кросс-платформенное хранилище данных, аккумулирующее всю необходимую для анализа предметной области информацию. Использование единого хранилища позволяет обеспечить непротиворечивость данных, их централизованное хранение и автоматически обеспечивает всю необходимую поддержку процесса анализа данных. Deductor Warehouse оптимизирован для решения именно аналитических задач, что положительно сказывается на скорости доступа к данным.

Deductor Studio – это программа, предназначенная для анализа информации из различных источников данных. Она реализует функции импорта, обработки, визуализации и экспорта данных. Deductor Studio может функционировать и без хранилища данных, получая информацию из любых других подключений, но наиболее оптимальным является их совместное использование.

Deductor Viewer – это облегченная версия Deductor Studio, предназначенная для отображения построенных в Deductor Studio отчетов. Она не включает в себя механизмов создания сценариев, но обладает полноценными возможностями по их выполнению и визуализации результатов. Deductor Viewer является средством тиражирования знаний для конечных пользователей, которым не требуется знать механику получения результатов или изменять способы их получения.

Deductor Server – сервер удаленной аналитической обработки. Он позволяет выполнять на сервере операции «прогона» данных через существующие сценарии и переобучение моделей. Deductor Server ориентирован на обработку больших объемов данных и работу в территориально распределенной системе.

Deductor Client – это клиент доступа в Deductor Server. Обеспечивает обмен данными и управление сервером.

Чаще всего аналитик не работает непосредственно с Server и Client, всю работу по настройке сценариев и визуализации выполняется им в Studio. Server всего лишь обрабатывает сценарии, подготовленные ранее аналитиком в Deductor Studio. Настройка работы с сервером выполняется сотрудниками IT служб: администраторами или программистами. В данном руководстве описываются только модули, с которыми непосредственно работает аналитик: Warehouse, Studio и Viewer.

Studio и Viewer базируются на одной архитектуре, поэтому все дальнейшее описание, касающееся визуализации и интерпретации результатов, в равной мере относится к обоим приложениям.

Поддержка процесса от разведочного анализа до отображения данных

Deductor Studio позволяет пройти все этапы KDD, перечисленные выше.

Схема на рисунке отображает процесс извлечения знаний из данных.

На начальном этапе в программу импортируются данные из какого-либо источника. Хранилище данных Deductor Warehouse также является одним из источников/приемников данных. Обычно в программу загружаются не все данные, а какая-то выборка, необходимая для дальнейшего анализа. После получения выборки можно получить подробную статистику по ней, просмотреть, как выглядят данные на диаграммах и гистограммах. Следующим шагом является принятие решения о необходимости предобработки данных. Например, если выясняется, что в выборке есть пустые значения (пропуски данных), можно применить фильтрацию для их устранения. Предобработанные данные далее подвергаются трансформации. Например, нечисловые данные преобразуются в числовые, что является необходимым условием для некоторых алгоритмов. Непрерывные данные могут быть разбиты на интервалы, то есть производится их квантование. К трансформированным данным применяются методы более глубокого анализа. На этом этапе выявляются скрытые зависимости и закономерности в данных, на основании которых строятся различные модели. Модель представляет собой шаблон, который содержит в себе формализованные знания. Следующий этап – интерпретация – предназначен для того, чтобы из формализованных знаний получить знания на языке предметной области. Наконец, последним этапом является тиражирование знаний – предоставление людям, принимающим решения, возможности практического применения построенных моделей.

Тиражирование знаний

Одной из наиболее важных функций любой аналитической системы является поддержка процесса тиражирования знаний, т. е. обеспечение возможности сотрудникам, не разбирающимся в методиках анализа и способах получения того или иного результата, получать ответ на основе моделей, подготовленных экспертом.

Например, сотрудник, оформляющий кредиты, должен внести данные по потребителю, а система автоматически выдать ответ, на какую сумму кредита данных потребитель может рассчитывать. Либо сотрудник отдела закупок при оформлении заказа должен получить автоматически рассчитанный, рекомендуемый объем закупки каждого товара.

Потребность в тиражировании знаний является объективной, так как, с одной стороны, интеллектуальная составляющая бизнеса становится все более значимой, с другой стороны, невозможно требовать от каждого специалиста, чтобы он разбирался в механизмах анализа. Создание моделей, поиск зависимостей и прочие задачи анализа нетривиальны.

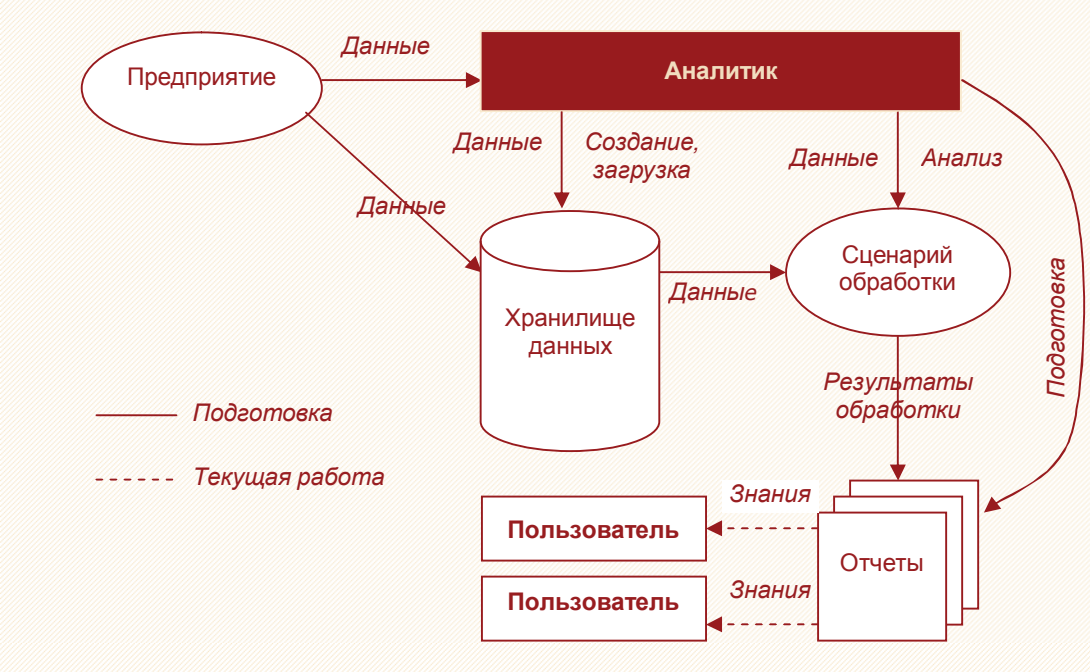
Подготовить систему, которая могла бы на всех данных гарантированно давать качественные результаты (прогнозы, рекомендации и прочее) не представляется возможным. Слишком широк спектр решаемых задач и слишком отличается логика бизнеса в различных компаниях. Но использование аналитической платформы Deductor позволяет по-другому подойти к решению задачи построения и использования моделей. Эксперт готовит сценарии обработки (модели) с учетом особенностей конкретного бизнеса, а остальные пользователи просто используют уже готовые модели, получая отчеты, прогнозы, правила, не задумываясь о том, как эти модели работают.

Для реализации концепции тиражирования знаний необходимо решить 4 задачи:

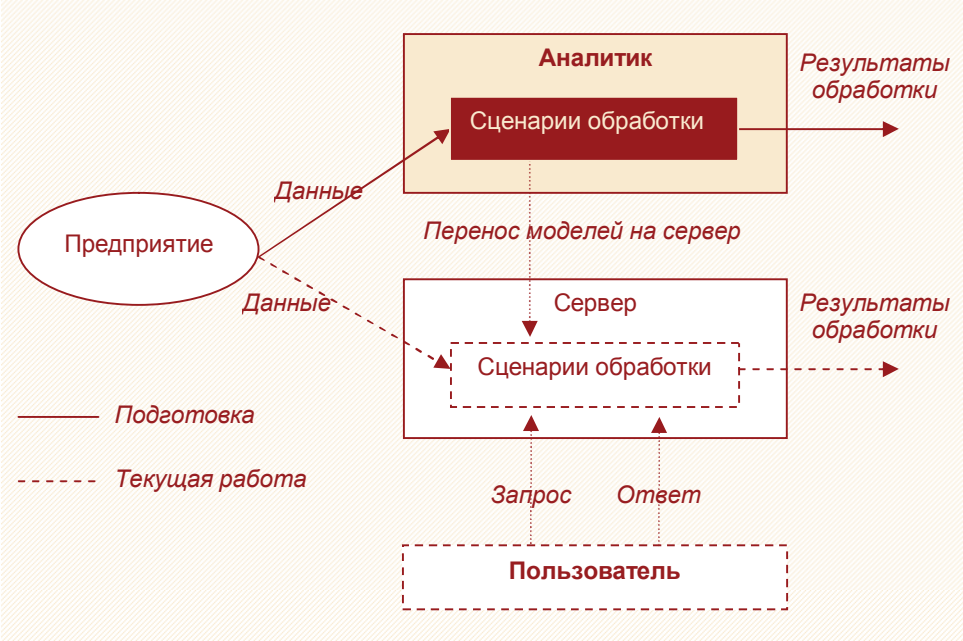
1. Аккумулировать данные, необходимые для анализа, обеспечив унифицированный простой способ получения любой информации, необходимой для анализа.
2. Формализовать знания экспертов. Трансформировать знания экспертов в модели, пригодные для автоматизированной обработки.
3. Подобрать способы визуализации и отобразить результаты обработки наиболее удобным способом.
4. Предоставить возможность работать сотрудникам с формализованными знаниями экспертов как с «черным ящиком», т.е. без необходимости вникать в то, каким образом реализована обработка внутри.

Делается это следующим образом (см. рисунок):

1. Создается хранилище данных – Deductor Warehouse, консолидирующее всю необходимую для анализа информацию. Настраиваются механизмы автоматического обновления сведений в хранилище. Данная операция гарантирует оперативное получение актуальной непротиворечивой и целостной информации для анализа.
2. Эксперт настраивает сценарии обработки, т.е. определяет последовательность шагов, которую необходимо провести для получения нужного результата. Почти всегда результат нельзя получить за одну операцию. Обычно это целая цепочка различного рода обработок. Например, для получения качественного прогноза необходимо получить сгруппированные нужным образом данные, провести очистку их от выбросов, сгладить, построить модель и «прогнать» через эту модель новые данные. Подготовка сценариев наиболее сложная часть работы, требующая серьезных знаний в предметной области и понимание методов анализа, но эту работу может провести один человек в компании.
3. Далее возможно несколько вариантов использования:
   (1) При работе в интерактивном режиме. Вывести на панель **Отчеты** ту информацию, которую необходимо получить пользователю системы, сгруппировав ее при этом в папки в зависимости от решаемой задачи. Настроить способы визуализации полученных данных, подобрать наиболее оптимальные методы отображения. Можно предоставить возможность просматривать полученные результаты при помощи специального приложения Deductor Viewer – рабочего места конечного пользователя.



(2) При работе в автоматическом режиме. Построенные модели переносятся на сервер, настраиваются клиентские приложения, взаимодействующие с Deductor Server таким образом, что в момент принятия решения происходит обращение к серверу, который «прогоняет» новые данные через построенные сценарии и выдает ответ.



В результате пользователю будут доступны результаты обработки в наиболее удобном для анализа и принятия решения виде.

При работе в интерактивном режиме и при выборе того или иного отчета система автоматически проведет все необходимые операции и предоставит результат. Лучше всего использовать для этого Deductor Viewer. Он не содержит средств для самостоятельного извлечения или построения сценариев, поэтому можно предоставить каждому пользователю отчеты, содержащие только нужную ему для работы информацию. Доступа к другим данным из хранилища и баз данных этот пользователь не получит.

При работе в автоматическом режиме вся обработка будет сведена к обращению с запросом к Deductor Server и получении с него готового ответа. Все необходимые действия будут выполнены автоматически.

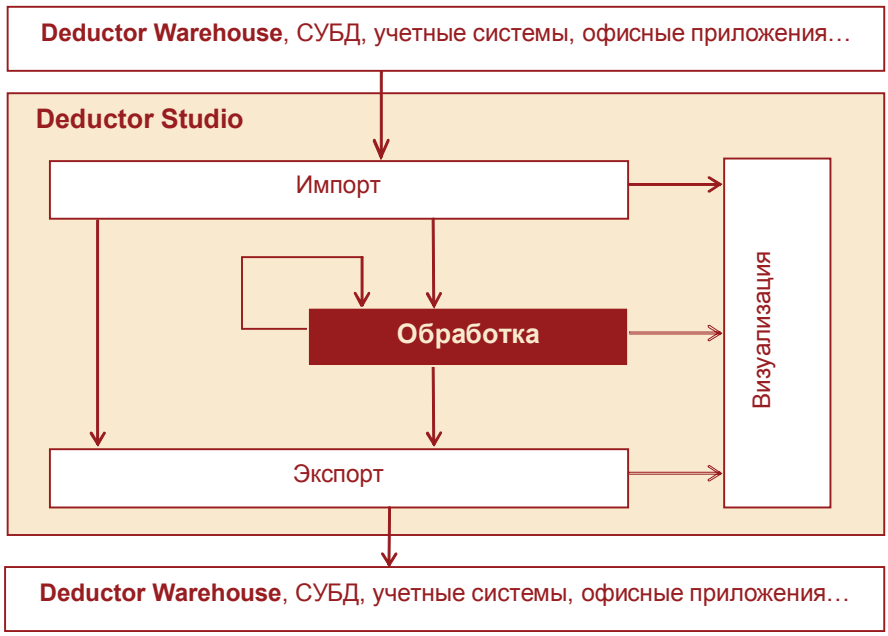
Подобные механизмы позволяют отделить процедуру подготовки сценариев, требующую определенных знаний и собственно получение результатов с использованием готовых сценариев. Так решается задача тиражирования знаний.

Архитектура Deductor Studio – аналитическое приложение

Основные модули

Вся работа по анализу данных в Deductor Studio базируется на выполнении следующих действий:

- импорт данных;
- обработка данных;
- визуализация;
- экспорт данных.



На рисунке показана схема функционирования Deductor Studio. Отправной точкой для анализа всегда является процедура импорта данных. Полученный набор данных может быть обработан любым из доступных способов. Результатом обработки также является набор данных, который в свою очередь опять может быть обработан. Импортированный набор данных, а также данные, полученные на каждом этапе обработки, могут быть экспортированы. Результаты каждого действия можно отобразить различными способами. Способ возможных отображений зависит от выбранного метода обработки данных. Например, нейросеть содержит визуализатор **Граф нейросети**, специфичный только для нее. В Studio включено множество специализированных визуализаторов. Есть способы визуализации, пригодные почти для всех методов обработки, например, в виде таблицы, диаграммы или гистограммы.

Последовательность действий, которые необходимо провести для анализа данных, называется **сценарием**. Сценарий можно автоматически выполнять на любых данных.

Подготовка сценариев

Перечисленные выше действия реализуются с помощью четырех Мастеров: импорта, обработки, визуализации и экспорта. Для построения сценария достаточно использовать только эти Мастера и ничего более.

Сценарий отображается на панели сценариев. Показать или скрыть эту панель можно, выбрав в главном меню **Вид ► Сценарии** или нажав на кнопку на панели инструментов. Сверху на панели сценариев расположены кнопки для вызова Мастеров.

Построение сценария начинается с вызова Мастера импорта. *Мастер импорта* предназначен для автоматизации получения данных из любого источника, предусмотренного в системе. Чтобы вызвать это действие, достаточно воспользоваться кнопкой Мастер импорта в верхней части панели или выбрать соответствующую команду из контекстного меню, вызываемого щелчком правой кнопки мыши в любом месте панели **Сценарии**. На первом шаге мастера импорта открывается список всех настроенных в системе типов источников данных. Среди них следует выбрать нужный тип источника и для перехода на следующий шаг щелкнуть по кнопке **Далее**. Число шагов мастера импорта, а также набор настраиваемых параметров отличается для разных типов источников.

Например, если исходные данные хранятся в текстовом файле с разделителями, нужно выбрать источник данных Text (Direct). Direct – означает прямой доступ к текстовому файлу, что увеличивает скорость работы с ним. Важным шагом в мастере импорта является настройка полей. Каждому полю источника данных можно присвоить метку столбца, которая будет использоваться для дальнейшей работы в программе. Например, если в источнике данных поле имеет имя «Name», ему можно задать метку «Наименование», что гораздо удобнее при дальнейшем отображении этого поля в таблицах или диаграммах.

Далее каждому полю нужно указать тип:

- 0/1 логический – данные в поле могут принимать только два значения - 0 или 1 (ложь или истина);
- дата/время – поле содержит данные типа дата/время;
- 9.0 вещественный – значения поля - числа с плавающей точкой;
- 12 целый – данные в поле представляют собой целые числа;
- ab строковый – данные в столбце представляют собой строки символов.

Затем указывается вид данных:

- непрерывный - значения в столбце могут принимать любое значение в рамках своего типа. Непрерывными могут быть только числовые данные и дата/время;
- дискретный - данные в столбце могут принимать ограниченное число значений. Как правило, дискретный характер носят строковые данные.

В зависимости от содержимого поля «Тип данных» на выбор вида данных накладываются ограничения, например, строковые данные не могут быть непрерывными. К выбору типа и вида данных нужно относится серьезно, так как это влияет на возможность дальнейшего использования этого поля. Неправильное указание типа данных может также привести к потере информации.

Мастер обработки предназначен для настройки всех параметров выбранного алгоритма. Для вызова Мастера обработки достаточно воспользоваться кнопкой Мастер обработки в верхней части панели или выбрать соответствующую команду из контекстного меню, вызываемого щелчком правой кнопки мыши в любом месте панели **Сценарии**. В окне первого шага мастера обработки представлены все доступные в системе методы обработки данных. Как правило, на следующем шаге мастера обработки производится настройка назначений полей. В зависимости от выбранного алгоритма предлагается выбрать некоторые из перечисленных назначений:

- непригодное – данные в поле не пригодны для данного способа обработки (программа автоматически указывает полю это назначение). Например, для преобразования даты поле должно иметь тип «Дата/время». Если оно будет иметь, например, строковый тип, то программа автоматически укажет для него назначение «Непригодное».

- неиспользуемое – запрещает использование поля в обработке данных и исключает его из выходного набора. В отличие от непригодного поля такие поля в принципе могут использоваться, если в этом будет необходимость;
- ключ – поле будет использоваться в качестве первичного ключа;
- входное – поле таблицы, построенное на основе столбца, будет являться входным полем обработчика (нейронной сети, дерева решений и т.д.).
- выходное – поле таблицы, построенное на основе столбца, будет являться выходным полем обработчика (например, целевым полем для обучения нейронной сети).
- информационное – поле содержит вспомогательную информацию, которую часто полезно отображать, но не следует использовать при обработке;
- измерение – поле будет использоваться в качестве измерения в многомерной модели данных;
- факт – значения поля будут использованы в качестве фактов в многомерной модели данных;
- атрибут – поле содержит описание свойств или параметров некоторого объекта;
- ID транзакция – поле, содержащее идентификатор событий, происходящих совместно (одновременно). Например, номер чека, по которому приобретены товары. Тогда покупка товара – это событие, а их совместное приобретение по одному чеку - транзакция;
- элемент – поле, содержащее элемент транзакции (события).

Мастер визуализации позволяет в пошаговом режиме выбрать и настроить наиболее удобный способ представления данных. В зависимости от выбранного способа представления будут настраиваться различные параметры, а мастер, соответственно, будет содержать различное число шагов.

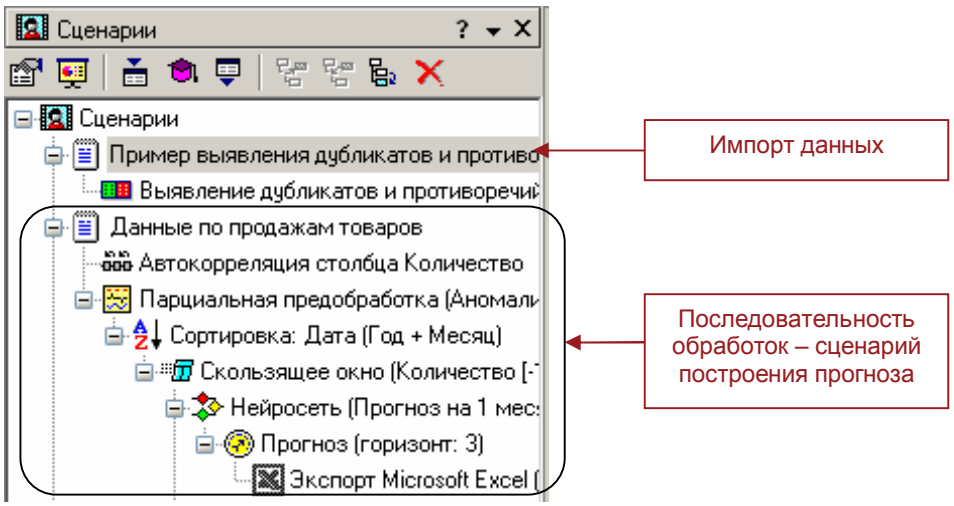
Для вызова мастера визуализации можно воспользоваться кнопкой **Мастер визуализации** на панели сценариев, предварительно выделив нужную ветвь в сценарии обработки или выбрав соответствующую команду из контекстного меню, вызываемого для данной ветви сценария. В зависимости от метода обработки, в результате которого была получена ветвь сценария обработки, список доступных для нее визуализаторов будет различным. Например, после построения деревьев решений их можно отобразить с помощью визуализаторов «Деревья решений» и «Правила». Эти способы отображения не доступны для других обработчиков.

Одновременно может быть выбрано несколько способов визуализации, при этом каждое из них будет открыто на отдельной закладке. Если одновременно выбрано насколько способов отображения данных, то соответствующие страницы с настройками каждого способа визуализации будут последовательно включены в общую процедуру настройки. Например, если выбраны и диаграмма и гистограмма, то в мастера отображений будут последовательно включены отдельные шаги для настройки диаграммы и гистограммы.

Мастер экспорта позволяет в пошаговом режиме выполнить экспорт данных в файлы наиболее распространенных форматов, различных баз данных и хранилища данных Deductor Warehouse.

Для вызова мастера экспорта можно воспользоваться кнопкой **Мастер экспорта** на панели сценариев. На первом шаге мастера экспорта представлен список приемников данных, в которые может быть выполнен экспорт данных. Среди них следует выбрать нужный и далее следовать шагам мастера. В результате набор данных будет выгружен в выбранный приемник.

С помощью последовательного применения мастеров импорта, обработки и экспорта можно построить сценарий. Пример сценария на рисунке.



Выполнение сценария – это последовательное применение обработчиков, начиная от узла импорта данных до выбранного узла сценария. Для выполнения сценария необходимо выбрать нужный узел двойным щелчком мыши. Например, если дважды щелкнуть мышью по узлу «Прогноз (горизонт: 3)» (см. рисунок), то выполнится сценарий: «Данные по продажам товаров» (импорт данных), «Парциальная обработка» (очистка данных), «Сортировка» (трансформация данных), «Скользящее окно» (трансформация данных), «Нейросеть» (построение модели), «Прогноз» (прогнозирование). В результате будут получены прогнозные значения объемов продаж, для которых, например, может быть построена диаграмма прогноза.

Судить о том, выполнен узел сценария или нет, можно по его иконке, т.к. у активного узла она цветная, у неактивного – серая. Отключать однажды исполненный узел может иметь смысл для экономии памяти или для пересчета построенной модели, например, после того, как были изменены исходные данные. Это можно сделать, вызвав всплывающее меню щелчком правой кнопки мыши на нужном узле и выбрав пункт **Активный**. В результате выполненный узел деактивируется. При повторном нажатии узел будет снова выполнен с новыми данными или настройками.

Для каждого узла сценария существует возможность изменения настроек. Для этого из всплывающего меню или панели инструментов выбирается команда **Настроить...**, в результате чего появляется окно мастера обработки, импорта или экспорта в зависимости от типа узла. В нем можно изменить любые параметры узла. Внесение изменений может быть отменено на любом шаге мастера. Изменения принимаются на последнем этапе (ввод имени узла, метки и описания), после чего текущий и все подчиненные узлы деактивируются. При повторном выполнении сценария все внесенные изменения уже будут учтены.

При разработке сценария можно изменять порядок следования узлов. Для этого выделенный узел перемещается вверх-вниз по дереву в пределах подчинения своему родителю с помощью комбинации клавиш <Ctrl-↑> и <Ctrl-↓>.

На панели инструментов и в контекстном меню окна сценариев дополнительно доступны следующие команды управления деревом проекта:

- *Вставить узел* – вставляет новый узел перед текущим узлом и вызывает для него Мастер обработки. Вставить узел перед узлом импорта данных нельзя, т.к. импорт – это всегда первый узел любого сценария.
- *Вырезать узел* – удаляет текущий узел из дерева. Все его потомки при этом перемещаются на один уровень вверх и начинают подчиняться родителю удаленного узла.
- *Копировать ветвь* – копирует ветвь сценария, начиная с текущего узла. Родителем новой ветви станет родительский узел оригинальной ветви.
- *Удалить ветвь* – удаляет ветвь сценария, начиная с текущего узла.

- *Переименовать* – переименовать узел сценария.
- *Сведения* – изменить имя, метку и описание узла, а так же отобразить параметры объекта сценария в специальном окне в виде дерева без активизации узла.
- *Статус пакетной обработки* – определить, будет ли данный узел выполняться или переобучаться при пакетном выполнении сценария.
-  *Добавить в избранное* – включить узел в список избранных для возможности последующего быстрого перехода к нему.
- *Сохранить ветвь* – сохраняет ветвь сценария, начиная с текущего узла в файл ветви или файл проекта для последующего использования.
- *Загрузить ветвь* – загружает из файла проекта или файла ветви сценарий обработки. Если сценарий не включает узла импорта данных, то его родителем станет текущий узел. В противном случае его родителем будет корневой узел дерева сценариев.

При выполнении операций вставки и удаления узла могут возникнуть проблемы с последующим выполнением сценария. Если узел выполнял важные для дальнейшей обработки действия и был удален из дерева, то его потомки могут стать неработоспособными. Аналогично, вставка узла, который изменяет столбцы данных, например, удаляет или переименовывает их, может помешать выполнению узлов-потомков.

Перечисленные операции позволяют легко вносить изменения в дерево сценариев, а значит изменять порядок и свойства обработки узлов в ходе выполнения. Однако результат обработки сам по себе может оказаться не столь удобным для анализа. Ведь он должен быть представлен аналитику в простом и доступном виде, поэтому в следующем разделе речь пойдет о важной части функционала платформы – визуализации данных.

Визуализация данных

На любом этапе обработки можно визуализировать данные. Система самостоятельно определяет, каким способом она может это сделать, например, если будет обучена нейронная сеть, то помимо таблиц и диаграмм можно просмотреть граф нейросети. Пользователю необходимо выбрать нужный вариант из списка доступных и настроить несколько параметров.

Возможные способы визуализации данных:

1. *Таблица*. Стандартное табличное представление с возможностью фильтрации данных, сортировки и быстрого расчета статистики (он-лайн статистика).
2. *Статистика*. Статистические показатели выборки.
3. *Диаграмма*. График изменения любого показателя. Имеется возможность выбора различных вариантов диаграмм: столбчатые, линейные, круговые и прочее.
4. *Многомерная диаграмма*. Отображает данные в многомерном виде – поверхность или топографическим способом.
5. *Диаграмма размещения* – показывает объекты, размещенные в пространстве.
6. *Гистограмма*. График разброса показателей. Гистограмма предназначена для визуальной оценки распределения данных. Распределение данных оказывает значительное влияние на процесс построения модели. Кроме того, по гистограмме можно судить о величине отклонений различной степени (гистограмма распределения ошибок).
7. *Куб*. Многомерное представление данных. Любые данные, используемые в программе, можно посмотреть в виде кросс-таблицы и кросс-диаграммы. Пользователю доступен весь набор механизмов манипуляции многомерными данными – группировка, фильтрация, сортировка, произвольное размещение измерений, детализация, выбор любого способа агрегации, отображение в абсолютных числах и в процентах.
8. *Дубликаты и противоречия*. Специальный визуализатор, сделанный на основе таблицы, для более удобного отображения результатов поиска дубликатов и противоречий.

9 *Матрица корреляции*. Отображает зависимость (корреляцию) между входными и выходными полями обработчика «Корреляционный анализ».

10 *Граф нейросети*. Визуальное отображение обученной нейросети. Отображается структура нейронной сети и значения весов.

11 *Дерево решений*. Отображение дерева решений, полученного при помощи соответствующего алгоритма. Имеется возможность посмотреть детальную информацию по любому узлу и фильтровать попавшие в него данные.

12 *Правила*. Отображают в текстовом виде правила, полученные при помощи алгоритма построения деревьев решений или поиска ассоциаций. Такого рода информация легко интерпретируется человеком.

13 *Значимость атрибутов*. Отображается степень влияния каждого входного атрибута на результат построения дерева решений. Параметр значимость тем выше, чем больше вклад вносит конкретный входной атрибут при классификации выходного поля. Фактически данный визуализатор показывает степень нелинейной зависимости между выходным и входными полями.

14 *Карта Кохонена*. Отображение самоорганизующихся карт, построенных при помощи соответствующего алгоритма. Широкие возможности настройки – выбор количества кластеров, фильтрация по узлу/кластеру, выбор отображаемых полей. Мощный и гибкий механизм отображения кластеризованных данных.

15 *Профили кластеров*. Статистическая информация по кластерам, которые получаются на выходе обработчиков «Карта Кохонена» и «Кластеризация k-means»

16 *ROC-анализ*. Доступен после обработчика «Логистическая регрессия». Позволяет оценить прогностическую силу модели, рассчитать оптимальный порог отсечения, проанализировать модель на чувствительность и специфичность.

17 *Коэффициенты регрессии*. Отображает коэффициенты, рассчитанные при помощи алгоритма «Линейная регрессия» и «Логистическая регрессия».

18 *Популярные наборы*. Отображение наиболее часто встречающихся в ассоциативных правилах множеств в виде списка с возможностью фильтрации и сортировки.

19 *Дерево правил*. Отображение в иерархическом виде (в виде дерева) ассоциативных правил. Содержит всегда два уровня. На первом – условие, на втором – следствие правила (или наоборот).

20 *Что-если*. Таблица и диаграмма. Позволяют «прогонять» через построенную модель любые интересующие пользователя данные и оценить влияние того или иного фактора на результат. Активно используется для решения задач оптимизации;

21 *Обучающий набор*. Выборка, используемая для построения модели. Выделяются цветом данные, попавшие в обучающее и тестовое множество с возможностью фильтрации. Необходима для понимания того, какие записи и каким образом использовались при построении модели.

22 *Диаграмма прогноза*. Применяется после использования метода обработки – Прогнозирование. Прогнозные значения на диаграмме выделяются цветом;

23 *Таблица сопряженности*. Предназначена для оценки результатов классификации вне зависимости от используемой модели. Таблица сопряженности отображает результаты сравнения категориальных значений исходного выходного столбца и категориальных значений рассчитанного выходного столбца. Используется для оценки качества классификаторов.

24 *Диаграмма рассеяния*. График отклонения значений, прогнозируемых при помощи модели, от реальных. Может быть построена только для непрерывных величин и только после использования механизмов построения модели, например, нейросети, линейной регрессии или пользовательской модели. Используется для визуальной оценки качества построенной модели.

25 *Диаграмма сезонных индексов*. Показывает тренд и сезонные индексы после применения обработчика «Декомпозиция временного ряда».

26 *Сведения*. Текстовое описание параметров импорта/обработки/экспорта в дереве сценариев обработки.

27 *Метаданные* – визуализатор для отображения метаданных хранилища данных.

28 *Навигатор* – навигатор по объектам базы данных.

Практически все механизмы визуализации поддерживают экспорт результатов в таблицы (MS Excel, MS Word, HTML, текстовые файлы…) или в графические файлы (GIF, BMP, EMF…).

Настроенные визуализаторы могут быть вынесены на панель **Отчеты**. Таким образом, конечный пользователь сможет просто получить и просмотреть необходимый результат, не задумываясь, каким способом он был получен.

Работа с отчетами

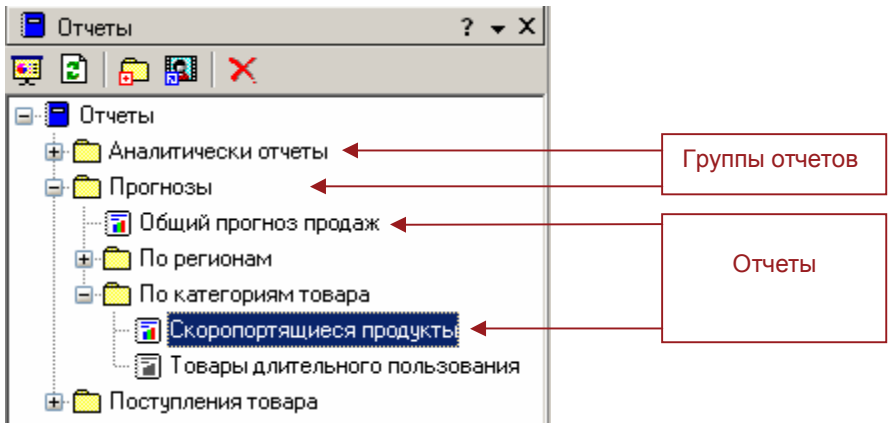
Задача тиражирования знаний заключается в предоставлении возможности сотрудникам, не разбирающимся в методиках анализа и способа получения того или иного результата, получать ответ на аналитические запросы на основе моделей, подготовленных экспертом. Для эксперта предназначена панель сценариев, в которой он строит различные модели. Для конечного же пользователя предназначена панель отчетов. Открыть или скрыть эту панель можно, выбрав в меню **Вид ► Отчеты** или нажав кнопку  на панели инструментов.

Отчеты представлены в виде древовидного иерархического списка, каждым узлом которого является отдельный отчет или папка, содержащая несколько отчетов. Чтобы добавить новый отчет, нужно щелкнуть по кнопке  **Добавить узел** или выбрать соответствующую команду из контекстного меню. В результате откроется окно **Выбор узла**, в котором следует выделить узел дерева сценария, где содержится нужная выборка данных и щелкнуть по кнопке **Выбрать**.

Чтобы добавить новую папку, нужно щелкнуть по кнопке  **Добавить папку** или выбрать соответствующую команду в контекстном меню. В результате в списке отчетов появится новая папка с открытым полем имени, куда следует ввести имя папки. После ввода имени для его сохранения щелкнуть по любому узлу списка. Чтобы поместить отчет в папку, нужно перед вызовом команды **Добавить узел** выделить папку, в которую необходимо поместить отчет. При создании узлов отчетов поддерживается технология drag&drop – просто перетащите нужный узел из панели **Сценарии** в панель **Отчеты**.

Для удобства пользователя предусмотрена возможность изменения порядка расположения отчетов и папок в дереве. Для этого выделенный узел перемещается вверх-вниз по дереву в пределах подчинения своему родителю с помощью комбинации клавиш <Ctrl-↑> и <Ctrl-↓>. Для того чтобы определить, на основе какого узла сценария построен отчет, следует выбрать пункт всплывающего меню **Найти узел в сценарии**.

Отчеты желательно группировать в папки по их смысловому содержанию. Например, папка «Аналитические отчеты» может содержать различные кубы данных, папка «Прогнозы» может содержать диаграммы прогнозов каких-либо величин. Тогда конечный пользователь открывает панель отчетов, выбирает нужную папку и в этой папке активизирует нужный отчет. После такого выбора программа автоматически выполняет сценарий, соответствующий этому отчету, и выдает результат в зависимости от настроенного отображения отчета.



Дерево отчетов может содержать один и тот же узел дерева сценариев несколько раз. Такая необходимость возникает для отображения одного и того же набора данных разными способами. Например, в одном случае набор будет отображен в виде диаграммы, в другом – в куба, а в третьем – в виде куба, но с другими измерениями. Для настройки отображения данных в дереве отчетов предназначена кнопка  , после нажатия которой открывается мастер визуализации, как в дереве сценариев. Надо отметить, что изменение отображения узла в панели отчетов не приводит к изменению отображения узла в панели сценариев. Это позволяет аналитику настраивать отображение в сценарии так, как ему удобно, а конечному пользователю вывести на панель отчетов более простой вариант визуализации.

Для формирования отчета достаточно выбрать его в дереве отчетов двойным щелчком мыши. При этом автоматически выполнится сценарий, соответствующий этому отчету. Например, если выбрать отчет «Общий прогноз продаж», выполнится сценарий из предыдущего примера. Результаты будут отображены в соответствии с настройками для панели отчетов, например, в виде диаграммы прогноза. Отчеты, для которых открыт хотя бы один визуализатор, показаны в дереве цветными иконками, в то время как остальные – серыми.

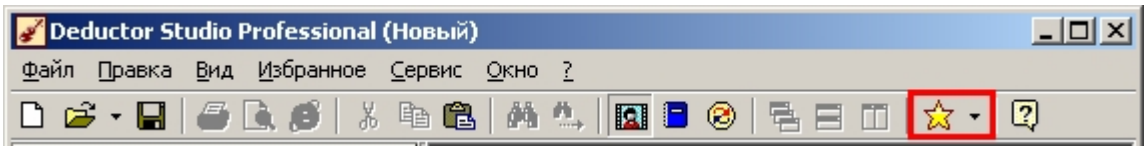
Созданный сценарий и дерево отчетов можно сохранить в файле проекта. Для этого необходимо выбрать пункт главного меню **Файл ► Сохранить**. Откроется диалоговое окно сохранения файла. В нем нужно выбрать путь, куда будет сохранен файл проекта, и указать имя файла.

На практике часто встречаются ситуации, когда пользователю требуется получить отчет по некоторому подмножеству всех доступных данных, например, только по одному поставщику или клиенту, по нескольким группам товаров или регионам. В терминах многомерной модели данных такое подмножество называется срезом. Аналитик может создавать отчеты в предопределенных, наиболее востребованных разрезах, но не в силах предсказать все виды отчетов, которые могут потребоваться пользователю. Поэтому он может дать возможность пользователю самому настраивать срез данных для отчета. Пользовательский срез данных может указываться при импорте данных из хранилища. В этом случае при выполнении отчета будет открыто окно фильтрации данных, в котором пользователю будет предложено выбрать интересующее его подмножество данных. Он может указать любой срез по предусмотренным аналитиком измерениям и получить в результате из хранилища только нужную информацию.

Работа с избранными узлами

Очень часто при проектировании сценариев значительная часть работы ведется с несколькими ключевыми узлами, в которых и определяются наиболее важные параметры обработки. Для упрощения работы с ними, в частности для того, чтобы их было легче найти в больших проектах в Deductor Studio реализована работа с избранными узлами. Избранным может быть любой сценария.

Для того, чтобы добавить узел к избранным можно воспользоваться способом drag&drop – «схватить» узел мылкой и перетащить его в область избранных на главной панели программы, выделенный на рисунке красным.



Кроме того можно добавить в избранное текущий узел, выбрав соответствующее действие из всплывающего панели сценариев или главного меню.

Для перехода к избранному узлу нужно выбрать его из списка на главной панели приложения или из главного меню **Избранное**.

Пакетная обработка

Предположим, что в построенном дереве сценариев есть узлы, содержащие экспорт данных во внешние файлы. Для экспорта результатов обработки можно открыть файл проекта в Deductor Studio, в дереве сценариев найти и выполнить все узлы экспорта. Тогда данные, представленные таблицами, будут экспортированы во внешние файлы. Это способ экспорта «вручную».

Но можно воспользоваться пакетной обработкой файла проекта. Для этого необходимо выполнить команду вида:

<путь к программе> <путь к файлу проекта> /параметр_запуска /параметры_лога

Например,

"C:\Program Files\BaseGroup\Deductor\Bin\DStudio.exe" C:/project.ded /run

Эту команду можно сохранить и выполнить несколькими равнозначными способами.

Первый – это создать ярлык. Для этого в проводнике нужно нажать правой кнопкой мыши, выбрать пункт **Создать** и подпункт **Ярлык**, затем, нажав кнопку **Обзор**, выбрать путь к Deductor Studio и в конце строки дописать <путь к файлу проекта> /run.

Второй способ – это создать пакетный файл – файл с расширением **.bat**. В этом файле следует написать всего одну строчку: <путь к программе> <путь к файлу проекта> /параметр_запуска.

Параметр_запуска может принимать два значения:

- **run**. Этот параметр используется для автоматического экспорта данных из программы.
- **teach**. Этот параметр используется для переобучения моделей, используемых в сценарии. Обычно однажды построенная модель используется в дальнейшем без изменений. Но иногда она перестает давать хорошие результаты на новых данных. Тогда и следует воспользоваться пакетной обработкой с этим параметром.

Допустим, мы создали первым способом ярлык и назвали его Export. После его запуска файл проекта открывается в Deductor Studio, автоматически обнаруживаются все узлы с экспортом данных и выполняются все ветви сценария, содержащие такие узлы. После обработки всех узлов экспорта Deductor Studio закрывается.

В Deductor есть возможность управлять пакетной обработкой при создании сценария. Для этого у каждого узла предусмотрены два флага: «Выполнить» и «Переобучить» – доступные через пункт всплывающего меню **Статус пакетной обработки**. Если флажок «Выполнить» сброшен, то при запуске в пакетном режиме с параметром **run** узел и все его потомки исключаются из процесса выполнения. При установленном флаге узел выполняется в нормальном режиме. Сброшенный флаг «Переобучить» указывает системе, что при запуске с параметром **teach** переобучать узел не требуется. При установленном флаге узел будет переобучен. По умолчанию для вновь создаваемых узлов оба флага установлены, т.е. в пакетном режиме узел может и выполняться и переобучаться.

Параметры_лога предназначены для управления лог-файлом пакетной обработки. Они могут принимать следующие значения:

- /log – включить подробный режим лог-файла; в логе будет сохраняться информация о времени начала и окончания обработки каждого узла.
- /logfile=<Имя файла> – указывает имя лог-файла.
- /logmode[=overwrite] – определяет режим работы с лог-файлом, если указать =overwrite, то используется режим перезаписи лог-файла, в противном случае записи добавляются в конец существующего лог-файла.

Если в командной строке не указаны параметры управления лог-файлом, то используются настройки, сделанные в окне «Настройка», пункт главного меню **Сервис ► Настройка**.

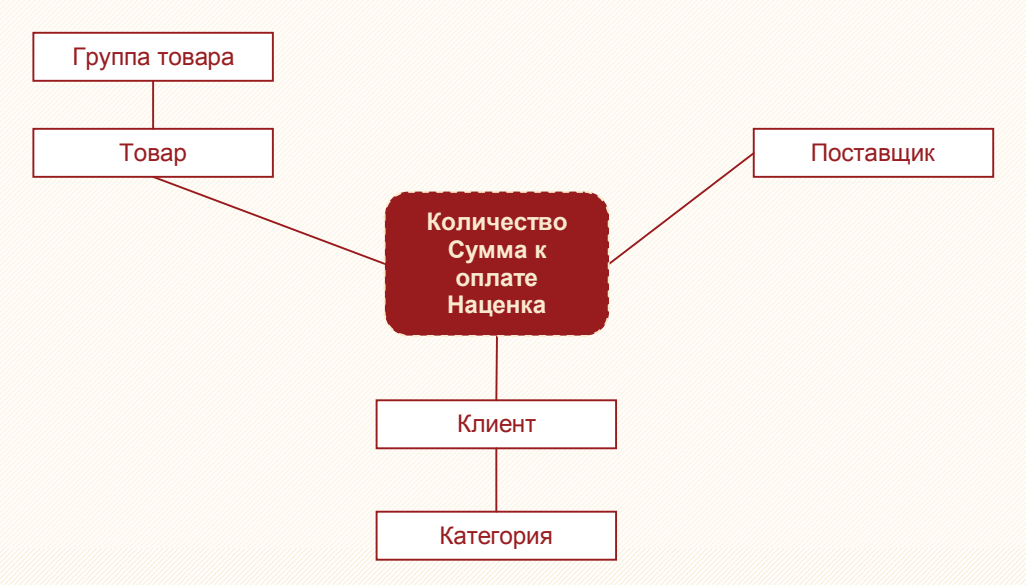
Выполнить сценарии и переобучить модели можно еще двумя способами: воспользоваться Deductor Studio как OLE сервером или использовать Deductor Server. Для этого требуется программирование, поэтому применение Deductor в таких режимах описано в Руководстве разработчика.

Архитектура Deductor Warehouse – многомерное хранилище данных

Многомерное представление данных

Deductor Warehouse 6 – многомерное хранилище данных, аккумулирующее всю необходимую для анализа предметной области информацию.

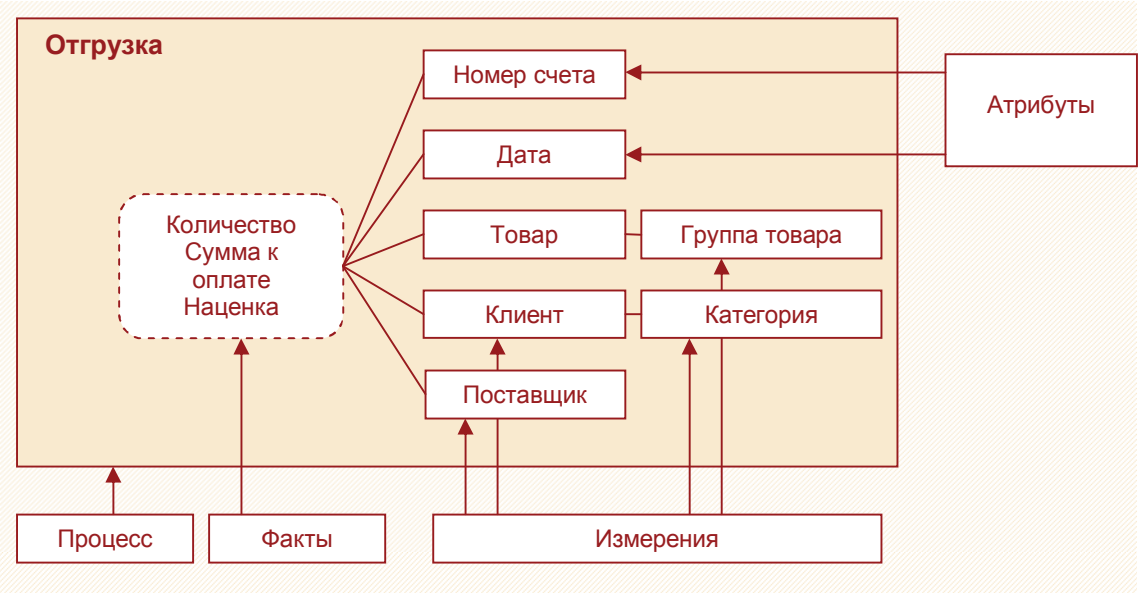
Начиная с пятой версии Deductor Warehouse, вся информация в хранилище хранится в структурах типа «снежинка», где в центре расположены таблицы фактов, а «лучами» являются измерения, причем измерение может ссылаться на другие измерения.



Такая архитектура хранилища наиболее адекватна задачам анализа данных, т.к. аналитик на практике оперирует многомерными понятиями. Каждая «снежинка» называется **процессом** и описывает определенное действие, например, продажи товара, отгрузки, поступления денежных средств и прочее. В Deductor Warehouse 6 может одновременно храниться множество процессов, имеющие общие измерения, например, *Товар*, фигурирующий в *Поступления* и в *Отгрузка*.

Замечание

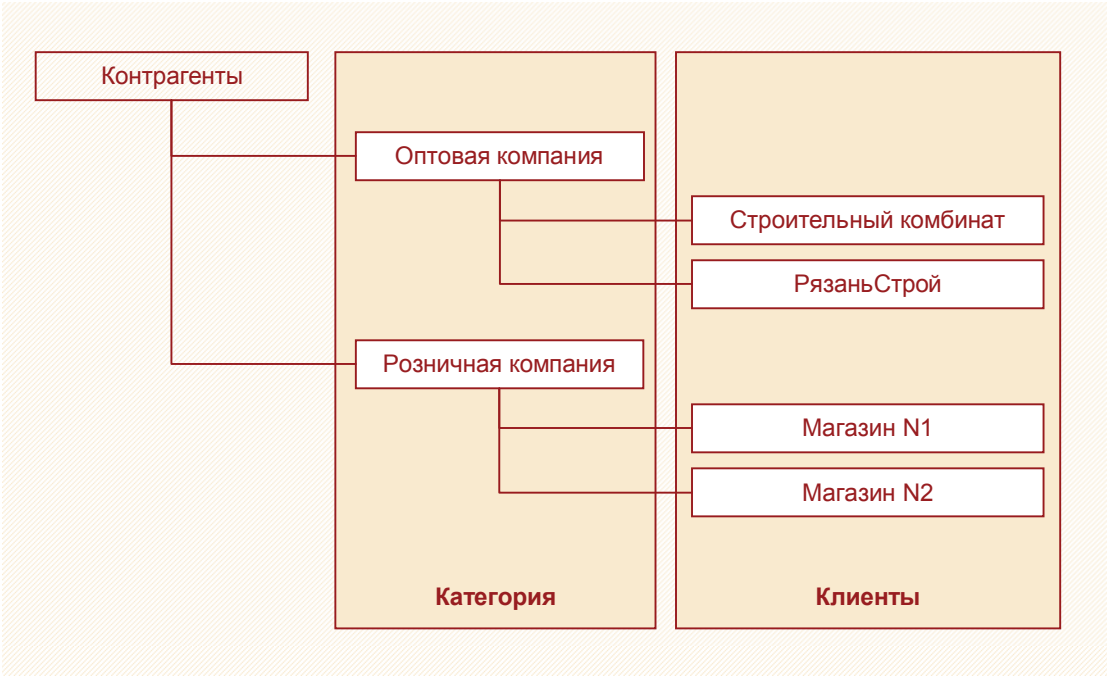
Отличие Deductor Warehouse версии 5 от версии 6 главным образом заключается во внутренней структуре хранения данных. Если в 5-й версии все данные процессов хранились в одной таблице, а данные измерений – в трех таблицах, то в 6-ой версии реализована полноценная ROLAP-модель (для каждого измерения и каждого процесса создается отдельная таблица). Достоинством 6-ой версии является значительно возросшая производительность. В то же время, любое серьезное изменение структуры хранилища в новой версии требует проведения операций по добавлению/удалению таблиц и столбцов.



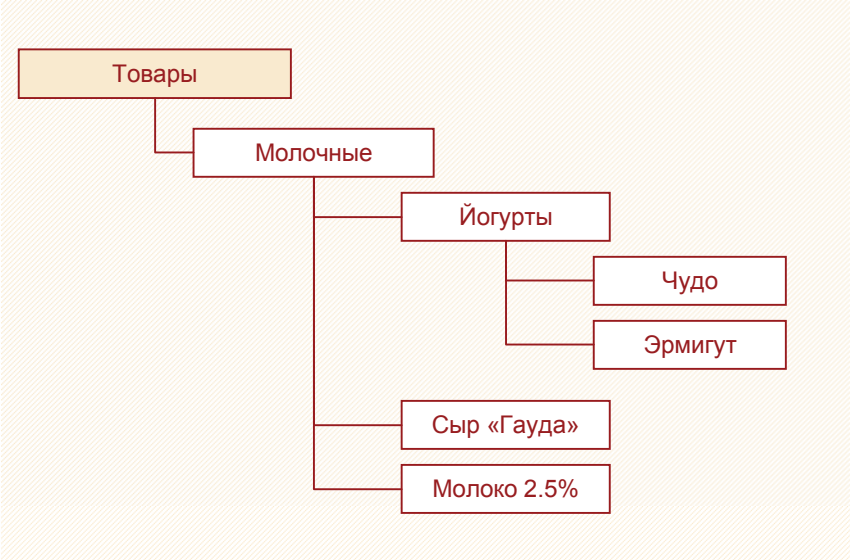
Измерения могут быть как простыми списками, например, Клиент, так и содержать дополнительные столбцы, называемые **атрибутами измерений**. Например, измерение *Товар* может состоять из *Наименование товара* - собственно измерение (первичный ключ) и *Вес*, *Объем* и прочее – атрибуты данного измерения.

Измерение может ссылаться на другое измерение, в свою очередь следующее измерение тоже ссылаться на измерение и так далее. Таким способом организуется **иерархия измерений**. Поддержку иерархий обеспечивает схема «снежинка». В Deductor Warehouse 6 поддерживаются только сбалансированные иерархии. Сбалансированной называется иерархия, где все ветви иерархии опускаются до одного уровня, и логическим «родителем» каждого элемента является уровень непосредственно над элементом. Например, последним уровнем иерархии *Контрагенты* может быть только клиент, и каждый клиент обязательно относится к одной из категорий клиентов. В несбалансированной иерархии количество уровней может быть разным, и конечный элемент иерархии может быть на любом из уровней.

Пример сбалансированной иерархии:



Пример несбалансированной иерархии:



Так же и измерения, процессы могут иметь атрибуты, которые так и называются **– атрибуты процесса**. Атрибут процесса в отличие от измерения не определяет координату в многомерном пространстве. Это справочное значение, относящееся к процессу, например, № накладной, Валюта документа и так далее. Значение атрибута процесса в отличие от измерения может быть не всегда определено.

Часто сложно определиться, что делать атрибутом процесса, а что измерением. Универсальных рецептов на все случаи не существует. Но можно дать общие рекомендации:

- совокупность измерений процесса должна однозначно определять единственную запись в таблице процесса («точку» в многомерном пространстве);
- если существуют иерархии, то выбор должен быть в пользу измерения;
- если по объекту хранилища данных предполагается в будущем делать частые «срезы», то снова лучше отдать предпочтение измерению;
- наличие возможных пропусков (необязательное поле) говорит о том, что объект лучше сделать атрибутом процесса.

Физическая реализация Deductor Warehouse

Физически Deductor Warehouse 6 – это реляционная база данных, содержащая таблицы для хранения информации и таблицы связей, обеспечивающие целостное хранение сведений. Поверх реляционной базы данных реализован специальный семантический слой, который преобразует реляционное представление к многомерному. Многомерное представление используется потому, что оно намного лучше реляционного соответствует идеологии анализа данных. Благодаря этому слою пользователь оперирует не полями и колонками таблиц базы данных, а многомерными понятиями, такими как измерение, факт, и система автоматически производит все необходимые манипуляции, необходимые для работы с реляционной СУБД.

Хранилище данных Deductor Warehouse прозрачно для пользователя проводит все необходимые операции по подключению к реляционной СУБД и выборке нужной информации. Системой поддерживается прозрачная работа хранилища данных на базе трех СУБД: Firebird, Microsoft SQL, Oracle.

С хранилищем данных на базе Firebird есть возможность локальной работы. Пользователю остается лишь создать или подключить хранилище данных к Deductor Studio.

Поддержка нескольких различных по стоимости и производительности СУБД в качестве платформы хранилища позволяет в каждом конкретном случае использовать наиболее пригодную для данного случая базу данных. Кросс-платформенный Deductor Warehouse является удобной базой для создания распределенных хранилищ данных, витрин данных и прочее.

Deductor Warehouse 6 реализует универсальное многомерное хранение, т.е. может содержать множество процессов с различным количеством измерений и фактов. Настройка процессов, задание измерений, атрибутов и фактов может осуществляться с помощью редактора метаданных, встроенного в Deductor Studio.

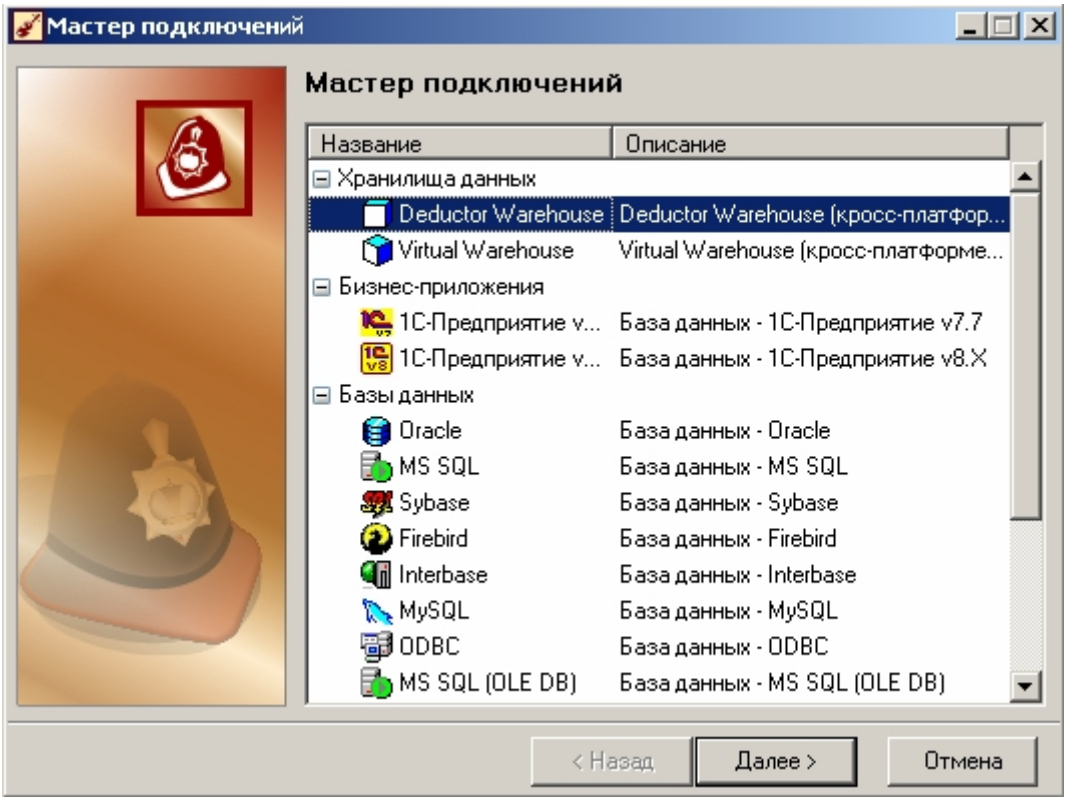
Создание хранилища данных

В этом разделе кратко описываются процессы создания и подключения локального традиционного хранилища данных на базе Firebird, создания его структуры, загрузки и импорта данных. Необходимо отметить, что при использовании хранилища на другой платформе эти действия идентичны, требуется только обеспечить доступ к соответствующему серверу базы данных и настроить подключение. Но есть одно отличие: локально работать можно только с хранилищем на базе Firebird.

Очень подробно концепция источников/приемников данных в Deductor, процессы создания и подключения хранилища, импорта и экспорта данных в хранилище описывается в документе «*Руководство по импорту и экспорту данных*». Рекомендуется при возникновении вопросов по работе с хранилищем данных первоначально изучить этот документ. Дополнительную информацию о подключении удаленного хранилища, оптимизации работы и текущему администрированию можно найти в «*Руководстве администратора*».

Создание хранилища данных производится на панели **Подключения**. Открыть или скрыть эту панель можно, выбрав в главном меню **Вид ► Подключения**.

Для создания хранилища данных необходимо выполнить следующее. На панели инструментов закладки **Подключения** нажать кнопку  . В результате откроется мастер подключений, в котором можно выбрать и настроить все доступные в системе источники/приемники данных:

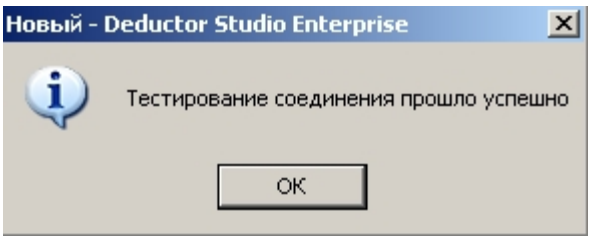


В данном мастере подключений доступны несколько типов хранилища данных: Deductor Warehouse (кросс-платформенный), Virtual Warehouse (кросс-платформенный).

Рассмотрим работу с Deductor Warehouse (кросс-платформенный). В Мастере подключений выбираем хранилище данных «Deductor Warehouse (кросс-платформенный)». Переходим на следующий этап настройки, нажав кнопку **Далее**. На следующем этапе выбираем тип базы данных. Система поддерживает следующие СУБД: Oracle, MS SQL, Firebird. После выбора типа базы данных на следующем шаге в мастере подключений необходимо указать параметры базы данных:

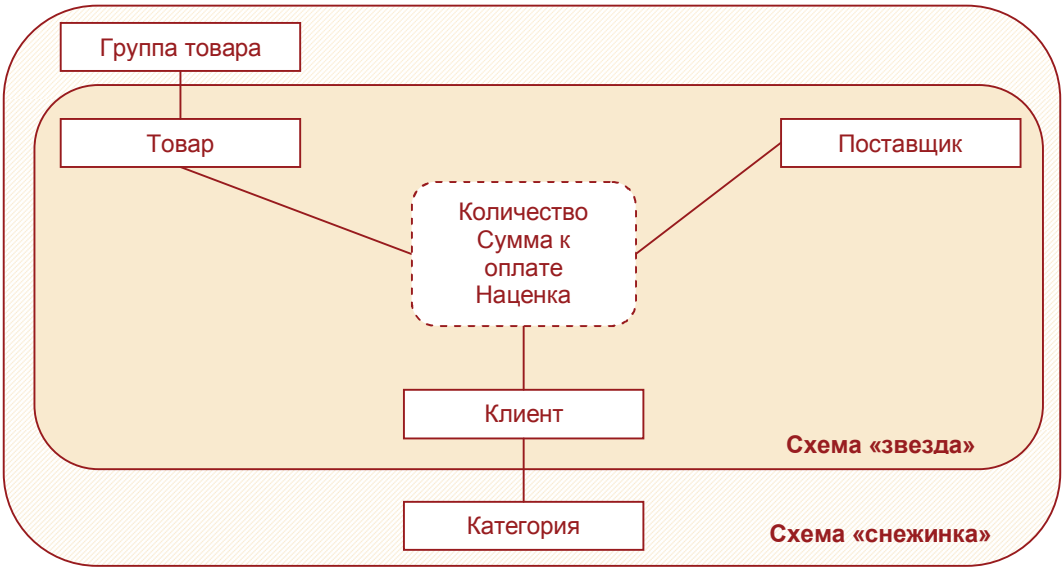
- *База данных* – необходимо указать имя базы данных;
- *Логин/пароль* – необходимо указать логин и пароль подключения к базе данных. По умолчанию в Firebird логин «sysdba» пароль «masterkey»;
- *Параметры* – имеется несколько следующих параметров:
  (1) Спрашивать логин/пароль при подключении;
  (2) Сохранять пароль;
  (3) Показывать системные таблицы.

На этом же шаге можно проверить правильность настроек подключения к базе данных. Для этого необходимо нажать кнопку. Если все параметры подключения указаны верно, то система выдаст следующее сообщение:



При использовании хранилища данных на базе Firebird имеется возможность выбора версии Deductor Warehouse 6, 5 или 4. Версии 5 и 4 включены в программу для совместимости с предыдущими версиями хранилищ данных. 4-ая версия поддерживает схему хранения «звезда».

В схеме «звезда» измерение может ссылаться только на таблицу фактов, а в «снежинке» измерение может ссылаться на другие измерения, которые в свою очередь ссылаются на таблицу фактов. Можно говорить, что «звезда» – это частный случай схемы «снежинка».



На следующем шаге в мастере подключений доступны дополнительные инструменты работы с хранилищем данных:

- **Тест** – проверка наличия необходимой структуры метаданных;
- **SQL скрипт** – создание файла с SQL скриптом, для создания необходимой структуры метаданных. Данный скрипт необходимо запустить на сервере используемой СУБД, чтобы создать там необходимую для Deductor Warehouse структуру метаданных;
- **Создать** – создать файл базы данных с необходимой структурой метаданных. Данный инструмент доступен в случае, когда хранилище данных Deductor Warehouse строится на платформе Firebird.

После выполнения всех настроек в мастере подключений на закладке **Подключения** в папке **Deductor Warehouse** появится новый узел с хранилищем. Хранилище готово к использованию.

Для того чтобы изменить настройки существующего хранилища данных, достаточно либо щелчком правой кнопки мыши на узле нужного хранилища открыть меню и выбрать там пункт **Настроить**, либо на панели инструментов закладки **Подключения** нажать кнопку . При выполнении этого действия откроется мастер подключений, в котором станет доступным внесение изменений в параметры на каждом его шаге.

Вновь созданное хранилище данных первоначально не содержит в себе никакой информации. В нем пока еще нет данных и не определены процессы, измерения, факты. Структура хранилища создается с помощью Редактора метаданных.

Подключение к Deductor Warehouse

Для начала работ с уже существующим хранилищем данных нужно зарегистрировать его в Deductor Studio, сообщив местонахождение и параметры подключения к базе данных. Эти действия называются подключением к хранилищу данных. Мы только что проделывали подобные шаги при создании нового хранилища. Подключение существующего хранилища проводится аналогично.

После настройки всех необходимых параметров можно проверить доступ к новому хранилищу данных. Для этого следует воспользоваться кнопкой  , в результате чего будет предпринята попытка соединения с базой данных хранилища. Если соединение будет успешным, то появится сообщение

Соединение успешно протестировано

и хранилище будет готово к работе. В противном случае будет выдано сообщение об ошибке, вследствие которой соединение невозможно. В этом случае нужно проверить параметры подключения, при необходимости внести в них изменения и протестировать подключение еще раз.

Теперь подключенное хранилище можно использовать для экспорта и импорта данных.

Создание структуры хранилища с помощью Редактора метаданных

Перед тем, как приступить к загрузке данных во вновь созданное хранилище, необходимо задать его структуру, т.е. определить, какие процессы, измерения, факты и свойства будут в нем содержаться. Для этого предназначен Редактор метаданных.

Редактор метаданных предоставляет удобный интерфейс для работы со своими объектами Deductor Warehouse. В окне Редактора можно создавать новые процессы и измерения, добавлять к процессам факты, а к измерениям – атрибуты, удалять процессы, факты, атрибуты и неиспользуемые измерения, производить очистку процессов и измерений, т.е. выполнять основные операции с метаданными хранилища.

Редактор метаданных может быть вызван с помощью всплывающего меню, нажатием кнопки  на панели инструментов закладки **Подключения**. В левой части окна Редактора показано дерево объектов хранилища (кубы, процессы, измерения, атрибуты и факты). Кубы, измерения, атрибуты и факты процессов сгруппированы в одноименные папки.

В правой части окна отображаются параметры выделенного объекта:

- *Имя* – внутреннее название объекта, используется при формировании запросов, в названии допустимы только латинские символы и числа;
- *Метка* – уникальное обозначение объекта, видимое пользователям Deductor, его можно изменить для любого объекта, введя новое значение в этом поле;
- *Описание* – комментарий к объекту;
- *Тип данных* – тип данных объекта: строковый, числовой и т.д.
- *Видимый* – признак включения объекта в множество объектов, доступных конечному пользователю при импорте;
- *Вид данных* и *Назначение данных* – установки, подсказывающие последующим узлам-обработчикам настройки столбцов по умолчанию.

Для объекта «Измерение» присутствуют опции:

- *Измерение* – ссылка на измерение, используемое в процессе.
- *Значение по умолчанию* – какое значение будет загружаться в измерение по умолчанию.
- *Область для данных*, *Область для индексов* – установки для тонкой настройки базы данных для хранилища для повышения производительности извлечения данных.

Для объекта «Процесс» присутствуют опции:

- *Измерение* – ссылка на измерение, используемое в процессе.
- *Область для данных*, *Область для индексов*.
- *Время последнего обновления* – информационное поле, содержащее служебную информацию о дате и времени последней загрузки данных. Устанавливается автоматически; при создании нового процесса значение параметра пустое.

В Редакторе метаданных предусмотрены следующие управляющие кнопки:

- *Раскрыть дерево* – отображает в виде дерева структуру хранилища данных;
- *Свернуть дерево* – скрывает структуру хранилища данных;
- *Добавить...* – добавить новый объект в хранилище;
- *Удалить…* – удаляет выделенный объект хранилища;
- *Мастер отладки…* – запускает диалоговое окно Мастера отладки SQL-запроса;
- *Собрать статистику* – сбор статистики по индексам по всему ХД;
- *Экспорт* – доступны три формата экспорта: HTML, Excel и Word.

Из любого объекта можно удалить все данные с помощью команды **Очистить** из всплывающего меню. Эта операция полезна при внесении изменений в структуру хранилища или ошибочной загрузке в хранилище неверных данных, но при ее использовании нужно проявлять особую осторожность, т.к. этим действием можно удалить всю информацию.

Загрузка данных в хранилище

Для загрузки данных в хранилище нужно воспользоваться Deductor Studio. Для этого необходимо выполнить следующие действия:

1. С помощью мастера импорта загрузить нужный набор данных в программу.
2. Загруженные данные могут подвергаться любой обработке или преобразованию средствами Deductor Studio. Этот шаг может быть пропущен, если данные не требуют обработки.
3. Запустить мастер экспорта и в списке приемников данных выбрать ветвь «Deductor Warehouse».
4. На следующем шаге выбрать хранилище, с которым планируется работать.
5. Далее объект в хранилище: процесс или измерение. В первом случае будет загружена весь набор данных в виде процесса, а во втором – значения измерения, выбранного пользователем. Далее необходимо указать соответствие элементов объекта хранилища данных с полями входного источника данных.
6. Настроить параметры загрузки и, собственно, запустить сам процесс загрузки данных.

Процессы

При загрузке в хранилище данных процесса нужно произвести следующие настройки:

1. Указать соответствие элементов объекта хранилища данных с полями входного источника данных. Все объекты процесса сгруппированы в три папки:
   1) *атрибуты* – значения будут загружены в это поле как значения атрибутов (свойств) процесса (в таблицу процесса);
   2) *факты* – значения будут загружены в это поле как значения фактов (в центр «снежинки»);
   3) *измерение* – значения будут загружены в это поле как значения измерений (в лучи «снежинки»).
2. На следующем шаге нужно указать измерения, по которым необходимо удалить строки из хранилища данных.
3. Определить атрибуты и факты, по которым необходимо произвести группировку перед загрузкой в хранилище. Если на предыдущем шаге были выставлены измерения, по которым будут удаляться данные из хранилища, и указана группировка данных, то загрузка будет проводиться без дополнительных проверок, что значительно ускорит процесс экспорта данных.
4. Задать параметры загрузки (сбор статистики и прочее) и запустить процесс загрузки данных.

Замечание

Перед тем, как загружать данные в процесс либо измерение, эти объекты необходимо создать в Редакторе метаданных.

Нужно проявлять осторожность при настройке полей, по которым будет производиться удаление из хранилища данных. Шаг мастера «Удалить из хранилища, используя измерение» дает возможность выбрать измерения, по которому будет производиться очистка процесса. Его установка позволяет значительно ускорить загрузку данных, но может привести к потерям информации из хранилища. Обычно для каждой загружаемой записи в процессе ищется запись с теми же значениями измерений. Если такая запись есть, то факты в ней обновляются, в противном случае в процесс добавляется новая запись. При установке флага **Удалить из хранилища, используя измерение** для входной выборки данных строится список уникальных значений по указанным измерениям, затем из процесса существующего в хранилище удаляются все строки, комбинации значений измерений которых содержатся в списке. Затем производится быстрая загрузка данных без выполнения каких-либо проверок.

Подробное описание параметров экспорта в Deductor Warehouse дано в *«Руководстве по импорту и экспорту данных»*.

Замечание

Перед загрузкой процесса предварительно должны быть созданы все измерения и рекомендуется отдельно загрузить все значения измерений, участвующие в процессе. В противном случае при загрузке процесса выдается ошибка. Например, если в процессе есть измерение «Менеджер» с 3 значениями Иванов, Петров, Сидоров, то предварительно нужно загрузить все эти значения в измерение «Менеджер» и только после этого приступать к загрузке процесса. Значения измерений могут загружаться и автоматически вместе с загрузкой процесса, но к данной операции нужно подходить с осторожностью, чтобы не «захламлять» хранилище.

Измерения

Загружать измерение рекомендуется отдельно, а не автоматически при загрузке процесса. Это позволяет лучше контролировать качество загружаемой информации. В случае же если измерение имеет дополнительные атрибуты (свойства), то их можно загрузить только при помощи отдельной вести сценария. Например, у измерения *Товар* могут быть атрибуты: *Вес*, *Цвет*, *Размер*. При загрузке процесса атрибуты измерения автоматически не загружаются, только само значение измерения. Для того чтобы добавить атрибуты в хранилище, измерение следует загрузить отдельно.

В случае если есть иерархия измерений, например, *Товарная группа* и *Товар*, то необходимо первоначально загрузить в хранилище измерения более высокого уровня иерархии (*Товарную группу*), а потом измерения более низкого уровня (*Товар*). При загрузке иерархии низкого уровня в загружаемой таблице должно быть поле с ссылкой на элемент иерархии более высокого уровня.

Пример

Пусть во внешнем источнике хранится информация об отгрузках, представленная следующей таблицей.

Таблица 1 – Отгрузка товаров

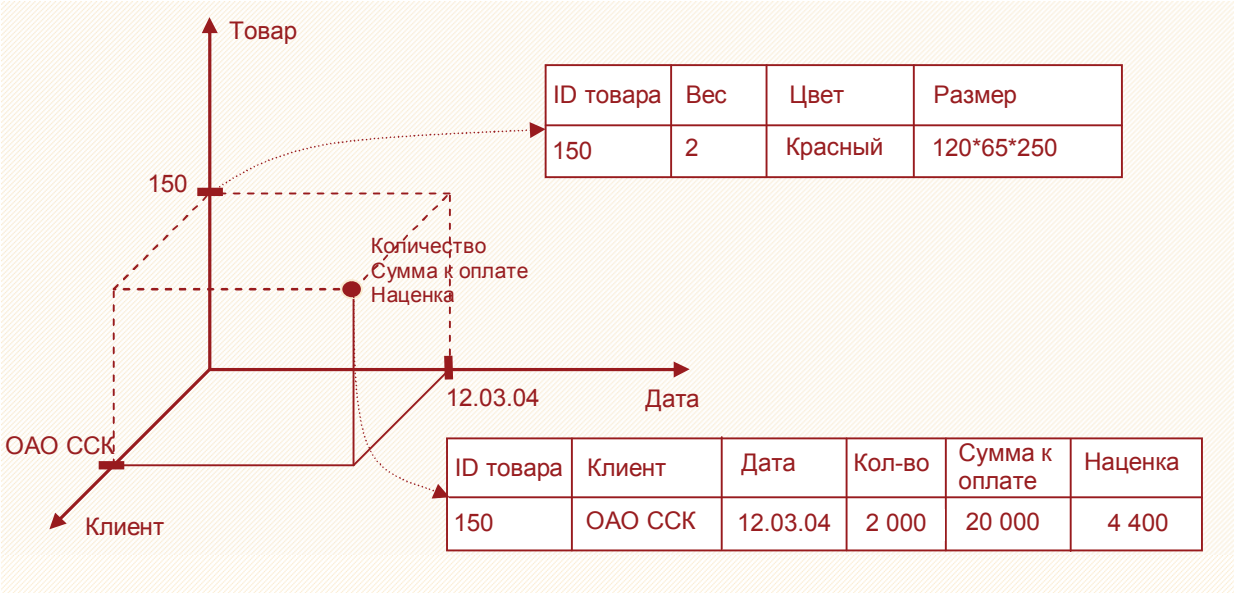
ID товара	Дата	Клиент	Номер накладной	Количество	Сумма к оплате	Наценка
80	01.03.04	ОАО «ССК»	051/2004	1000	7500	1000
90	10.03.04	ООО «ДСК»	052/2004	1500	15000	3000
150	12.03.04	ООО «ДСК»	053/2004	900	9000	2000
150	12.03.04	ОАО «ССК»	054/2004	2000	20000	4400
160	15.03.04	ООО «ДСК»	055/2004	500	6000	1000

В другой таблице хранится информация о товаре.

Таблица 2 – Товары

ID товара	Наименование	Группа	Цвет	Вес	Ширина	Высота	Длина
80	ТКСМ-100	Силикатный	Белый	1	120	85	250
90	ТКСМ-125	Силикатный	Тонированный	2	60	88	190
150	М100-125	Керамический	Красный	2	120	65	250
160	М100-175	Керамический	Песочный	1	120	65	250

Графически эти данные можно представить в многомерном пространстве следующим образом.



ID товара	Вес	Цвет	Размер
150	2	Красный	120*65*250

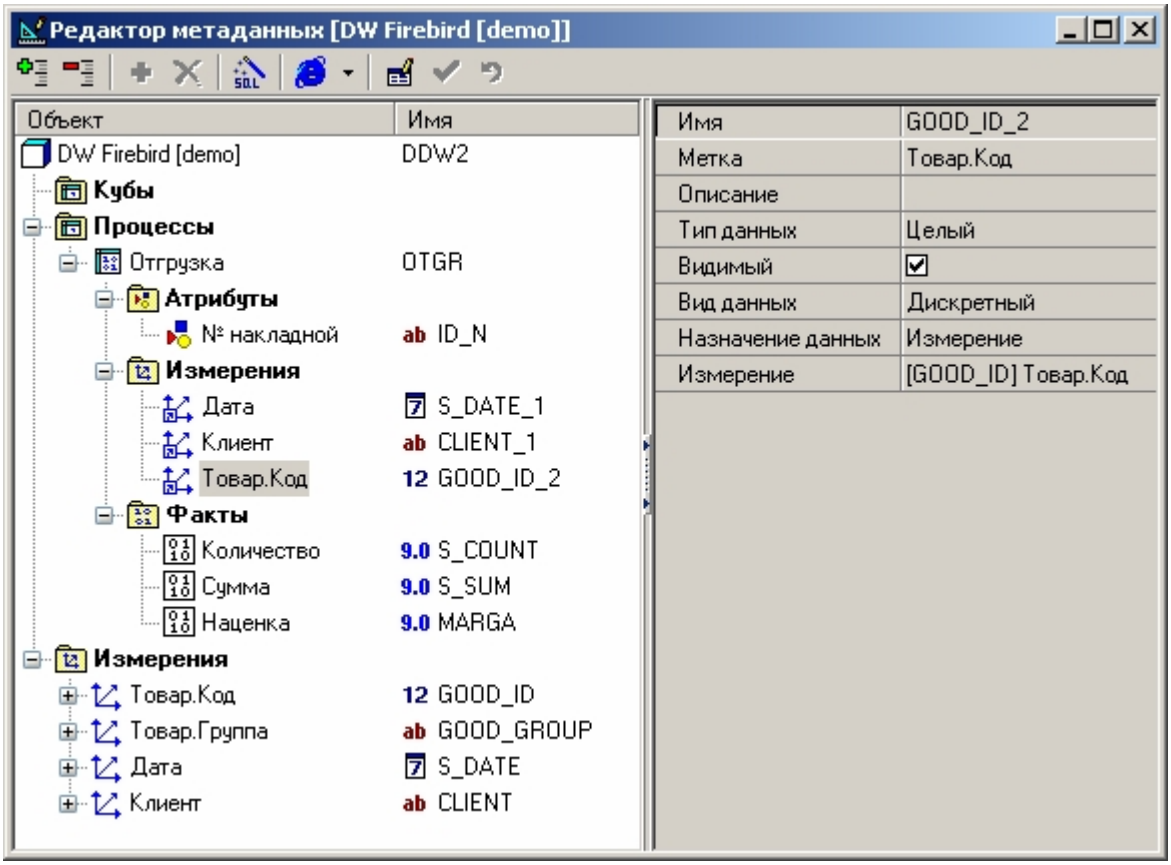
ID товара	Клиент	Дата	Кол-во	Сумма к оплате	Наценка
150	ОАО ССК	12.03.04	2 000	20 000	4 400

Для начала необходимо импортировать эти данные в программу. Для этого вызвать два раза Мастер импорта и загрузить (т.е. создать узел импорта) данные сначала о товаре, а затем об отгрузке.

Пока наше хранилище пусто и не содержит никаких процессов и измерений. Для создания его структуры нужно использовать Редактор метаданных хранилища. Но сначала нужно определиться, что является процессом, фактами и измерениями. Процесс – это отгрузка товаров. Он представлен первой таблицей. Измерения – это *ID товара*, *Дата* и *Клиент*. В нашем случае комбинация этих трех измерений уникально идентифицирует точку в многомерном пространстве, т.е. предполагается, что в день один товар отгружается клиенту только один раз. *ID товара* является измерением и однозначно определяет товар. Группа, цвет, ширина, высота и длина – атрибуты товара. Они могут быть различными у товаров. Кроме того, их впоследствии можно будет изменить. Факты – это количество, сумма к оплате и наценка. Номер накладной лучше сделать атрибутом процесса – это справочное значение к каждой записи в таблице процесса, и нет смысла его помещать в измерение. Измерение *Товар* содержит атрибуты *Наименование*, *Цвет*, *Ширина*, *Высота* и *Длина*.

Откроем Редактор метаданных Deductor Warehouse и создадим перечисленные измерения и процесс. Редактировать метаданные можно только после нажатия кнопки  , выйти из этого режима – нажав кнопку  **Принять изменения**.

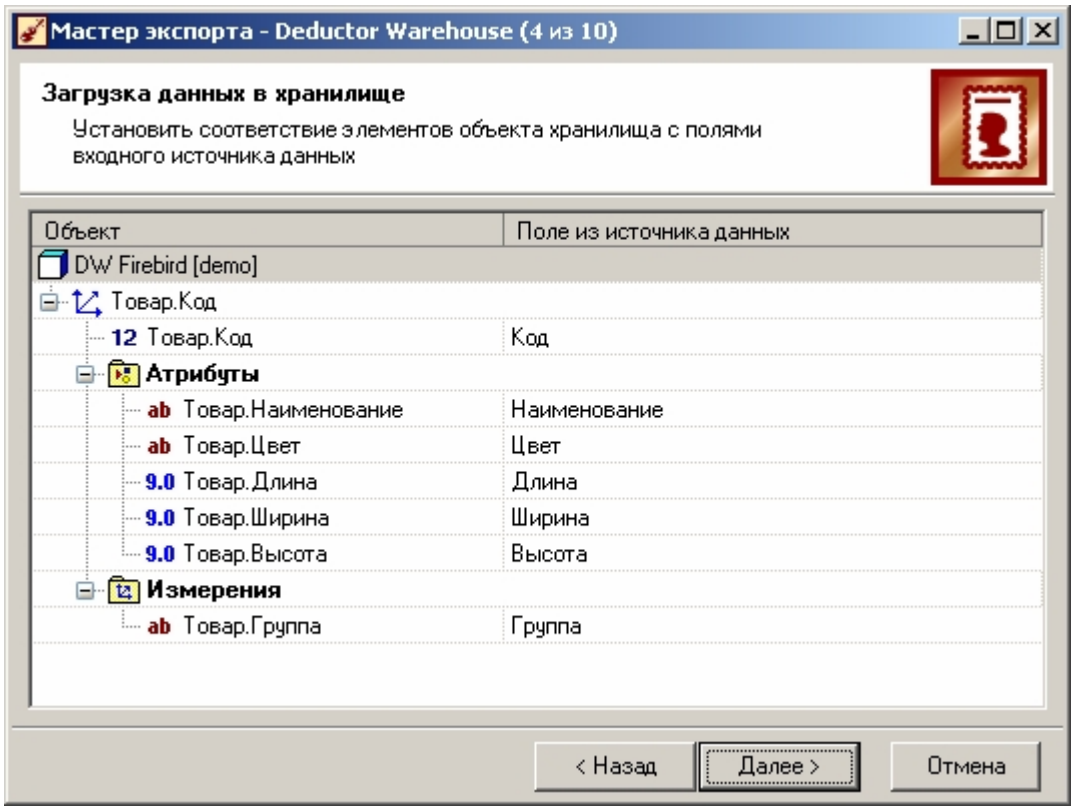
Осталось разобраться с товарной группой. Налицо иерархия товаров, т.е. в каждую товарную группу входит определенное число товаров. Поэтому добавим измерение *Группа*, а у измерения *Товар* – ссылку на это измерение («Группа»). В итоге получим ХД, готовое к загрузке данных (см. рисунок).



Экспорт начнем с измерений, иначе процесс *Отгрузка* успешно не загрузится. Нам необходимо загрузить 4 измерения. Причем загрузка измерений должна начинаться от верхних уровней иерархии к нижним, то есть в нашем примере сначала нужно загрузить измерение *Группа*, и только потом – *Товар.Код*. Такая последовательность работы требуется потому, что при загрузке изменения *Товар.Код* необходимо сразу указать *Группу*, к которой он относится, и эта *Группа* уже должна присутствовать в хранилище. В противном случае хранилище данных не допустит загрузки в связи с нарушением ссылочной целостности, т.е. имеется товар, который ссылается на несуществующую группу.

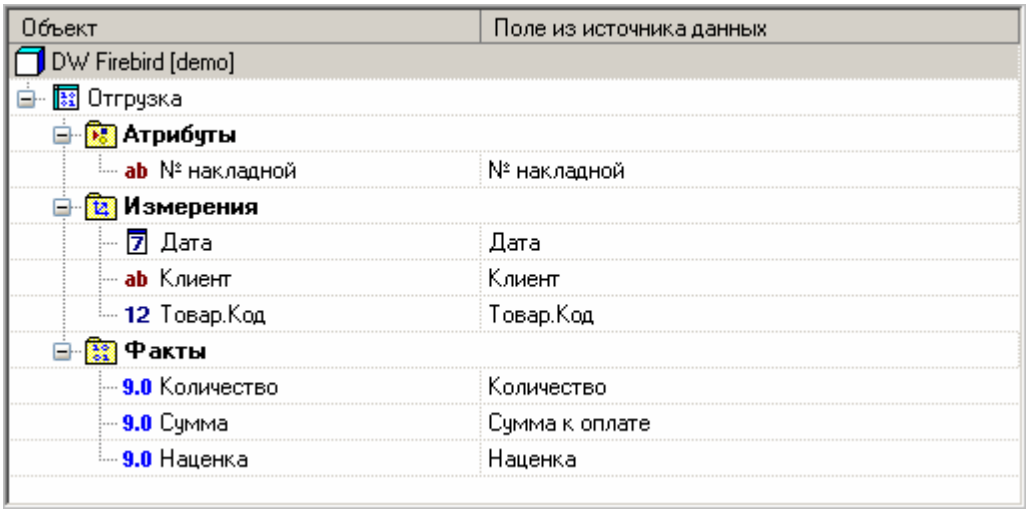
Для загрузки нужен список уникальных (неповторяющихся) значений групп. Его можно получить при помощи обработчика «Группировка», указав в качестве измерения то поле, по которому мы хотим получить список уникальных значений (в данном случае – *Группа*).

Теперь мы можем загрузить измерение *Товар.Код*, имеющего дополнительные атрибуты. Для этого в сценарии выберем узел с таблицей товаров и вызовем мастер экспорта. В этом мастере из окна **Deductor Warehouse** укажем объект *Товар.Код* в группе «Измерение», загрузим наши исходные данные в это измерение, указав соответствия полей, как показано на следующем рисунке.

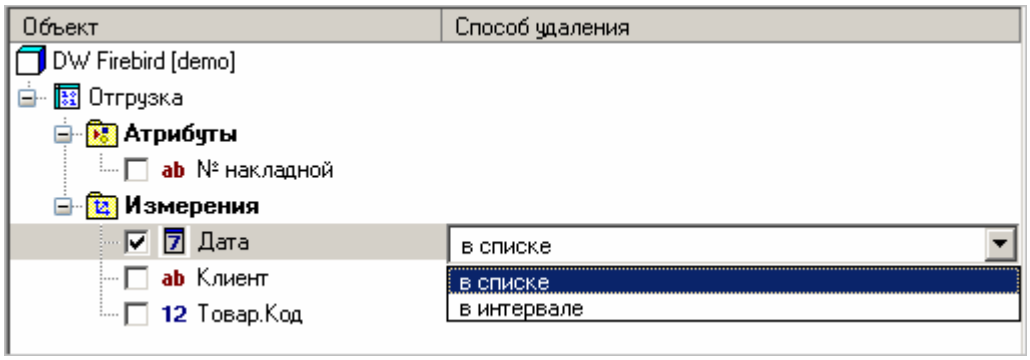


Измерения без атрибутов *Дата*, *Клиент* можно не загружать отдельными узлами сценария, а сделать это во время загрузки в процесс.

Последнее, что осталось сделать – загрузить данные в процесс. Воспользуемся снова мастером экспорта, только на этот раз в качестве объекта экспорта выберем процесс *Отгрузка*. Укажем соответствие полей, как это показано на рисунке.



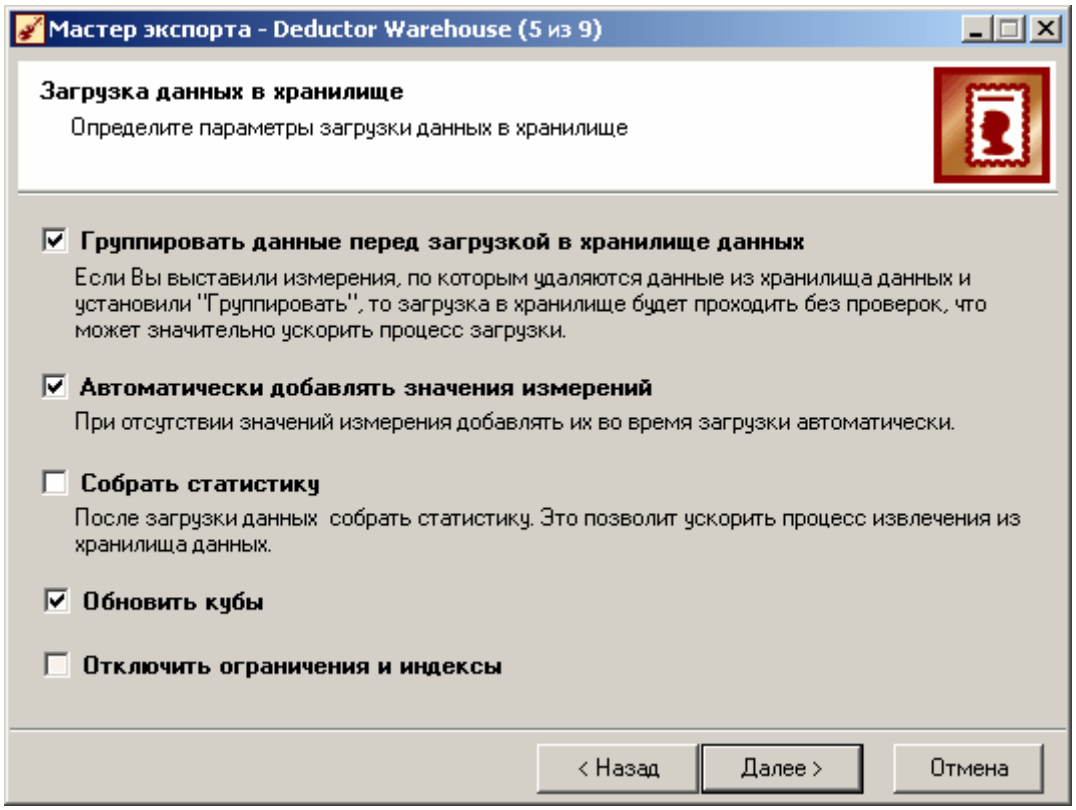
На следующем шаге определим удаление данных из хранилища по измерению *Дата*. В данном примере при повторной загрузке в процесс из него будут удалены и загружены заново данные на те даты, которые совпадают в источнике и в хранилище. Например, если в хранилище есть данные о том, что на 01.03.04 данному клиенту продано определенного товара в количестве 1000, а теперь загружается количество 1200, то реально будет храниться именно 1200. Таким образом, достигается и контролируется непротиворечивость данных.



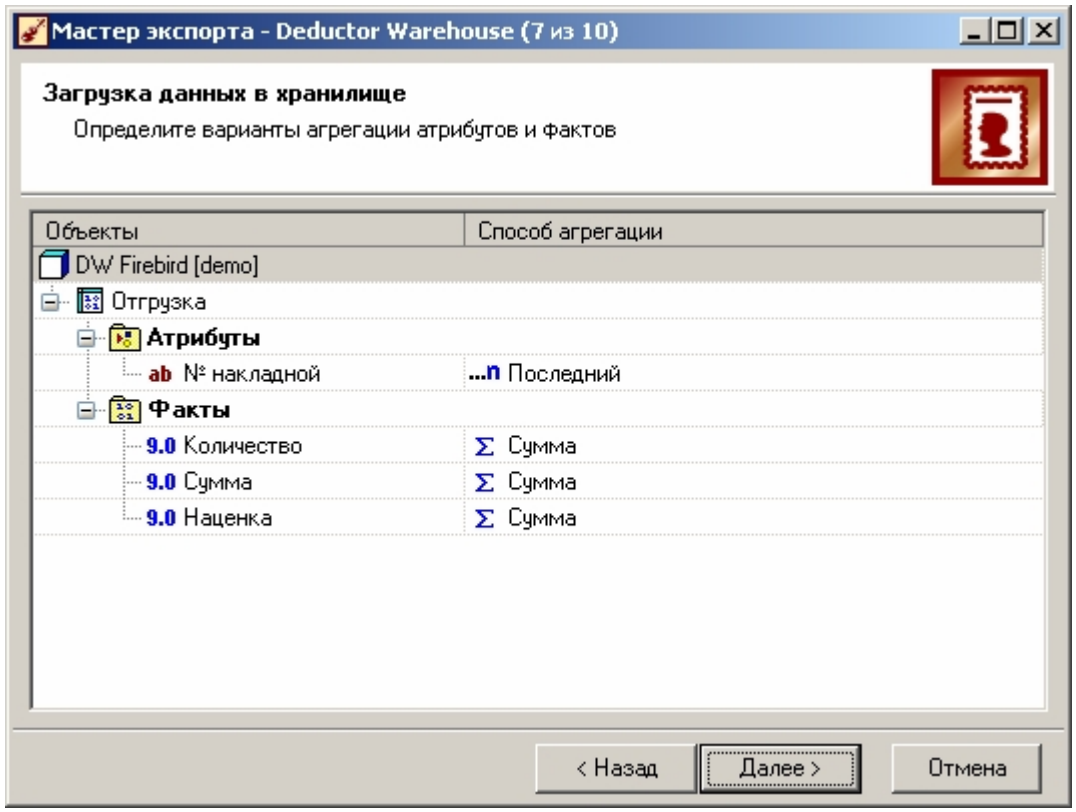
Для измерения здесь доступны две опции в колонке **Способ удаления** (по умолчанию предлагается *в списке*):

- *в списке* – при выполнении операции удаления для измерения списком перечисляются все значения;
- *в интервале* – у загружаемых значений измерения определяются минимальное и максимальное значения и из ХД удаляются все значения, лежащие внутри этого диапазона, поэтому операция производится быстрее. Рекомендуется для дат.

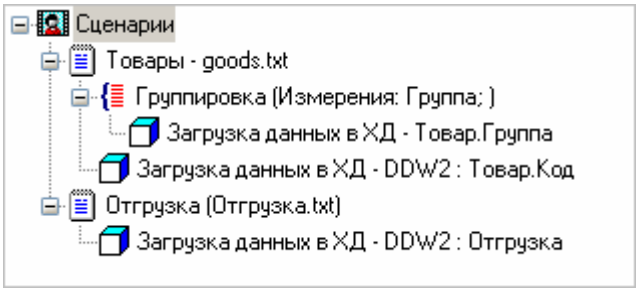
В следующем окне поставим флаг **Собрать статистику**, так как его использование значительно повышает быстродействие при импорте данных из хранилища. Флаг **Автоматически добавлять значения измерений** также включим, поскольку измерения *Дата* и *Клиент* мы отдельными ветками не экспортировали. Остальные настройки оставим нетронутыми, в том числе активным флаг **Обновить кубы**. Он говорит о том, что после загрузки данных в ХД имеющиеся кубы с активной опцией **Автоматическое обновление** будут заново перестроены (см. подраздел «Кубы в хранилище данных»).



Если на предыдущем шаге флаг **Группировать данные перед загрузкой...** был поднят, то появится еще одна вкладка. На ней нужно определить варианты агрегации атрибутов и фактов в том случае, если их комбинация в источнике данных не обеспечивает уникальности точки в пространстве. Оставим их предлагаемыми по умолчанию.



Таким образом, мы создали структуру хранилища и загрузили данные об отгрузках товара. Окончательный сценарий описанных действий, состоящий из 6 узлов, приведен на рисунке.



Автоматическая загрузка данных в хранилище

Информация в хранилище данных должна постоянно обновляться, поэтому этот процесс нуждается в автоматизации. Для этих целей в Deductor включены средства автоматического выполнения сценариев.

В сценарии проекта загрузки данные импортируются из различных источников и экспортируются в измерения и процессы хранилища. Для автоматического обновления хранилища данных имеется возможность выполнить действия по загрузке в пакетном режиме.

Пример

Продолжим пример из предыдущего раздела.

Сохраним созданный сценарий выгрузки данных в хранилище под именем **load.ded** в корне диска C:. Теперь создадим ярлык, в котором укажем команду:

"C:\Program Files\BaseGroup\Deductor\Bin\DStudio.exe" C:\load.ded /run

При запуске данного ярлыка будет осуществляться автоматическая загрузка данных в хранилище.

Кроме пакетного выполнения в Deductor Studio, автоматически выполнить сценарий можно при помощи Deductor Server. Подробнее работа с сервером описана в «Руководстве администратора».

Импорт данных из хранилища

Импорт данных из хранилища производится с помощью мастера импорта, в котором необходимо выбрать в качестве типа источника данных Deductor Warehouse.

Для начала нужно выбрать хранилище данных, из которого требуется выполнить импорт. В окне выбора хранилища данных представлены две колонки «Хранилище данных» и «Описание». В колонке «Хранилище данных» указывается имя хранилища, под которым оно зарегистрировано в системе, а в списке «Описание» (это необязательный параметр) – краткая характеристика содержимого хранилища, которая вводилась при его создании.

На следующем шаге необходимо определиться что будем импортировать процесс или измерение. И то и другое отображается в списке доступных объектов Deductor Warehouse.

Импорт процесса

Если интересует процесс, то его нужно выбрать на странице со списком объектов хранилища данных. Информация, относящаяся к какому-либо объекту или бизнес-процессу, представлена в

хранилище в виде «снежинки», где в центре расположены таблицы фактов, а «лучами» являются измерения, причем «лучи» могут ссылаться на другие «лучи». Каждая такая структура называется процессом. В общем случае, таких процессов в хранилище содержится несколько. Поэтому каждый раз при обращении к хранилищу необходимо выбрать процесс, из которого должны импортироваться данные.

Далее нужно определить, какие измерения, атрибуты и факты из выбранного на предыдущем шаге процесса должны быть импортированы. Это необходимо потому, что процесс может содержать множество измерений и фактов, а пользователю будут нужны только некоторые из них. Выбор только тех измерений и фактов, которые нужны в данном случае, позволит сэкономить время при импорте данных и избежать появления ненужной информации в выборке.

Все объекты выбранного процесса представлены в виде дерева, где атрибуты, измерения и факты образуют ветви. Атрибуты обозначены значком  , измерения –  , а факты –  . Слева от каждого объекта расположен флажок. Установка флажка позволяет выбрать соответствующий атрибут, измерение или факт процесса для импорта, а сброс флажка, наоборот, исключает их из числа импортируемых. Кроме этого, с измерением могут быть связаны один или несколько атрибутов измерения. В этом случае атрибуты образуют подветвь ветви измерения, где каждый атрибут обозначен значком  .

Далее требуется задать срезы, т.е. указать значения измерений и/или атрибутов, выбранных на предыдущем шаге, которые будут импортированы. Этот шаг желателен, потому что количество значений измерения может быть очень большим, а загрузка всех значений нецелесообразна. Поэтому выбор только тех значений измерения, которые представляют интерес в данном случае, поможет сэкономить время загрузки данных и не будет загромождать полученную выборку. Чтобы задать параметры среза для данного измерения или атрибута, необходимо выделить его в дереве объектов, выбрать условие и щелкнуть по кнопке выбора значений  . В результате откроется диалоговое окно, в котором нужно задать параметры фильтрации.

Список условий содержит несколько значений, основные из которых следующие:

- *<пусто>* – фильтрация по указанному объекту не производится;
- = – указывается значение, которое *нужно* импортировать;
- <> – указывается значение, которое *не нужно* импортировать;

При фильтрации по значению в диалоговом окне отображается список всех значений данного измерения, из которого нужно выбрать единственный вариант.

- *в списке* – указываются значения, которые *входят* в список импортируемых;
- *вне списка* –- указываются значения, которые *не входят* в список импортируемых.

В случае фильтрации по списку в диалоговом окне будет представлен список всех значений данного измерения. Чтобы выбрать значение, достаточно выделить его щелчком мыши (или установить флажки напротив тех значений, которые интересуют).

- *содержит* – указывается подстрока, которая *должна содержаться* в импортируемых значениях измерений;
- *не содержит* – указывается подстрока, которая *не должна содержаться* в импортируемых значениях измерений;
- *начинается на* – указывается подстрока, с которой *должны начинаться* импортируемые значения измерений;
- *начинается на* – указывается подстрока, с которой *не должны начинаться* импортируемые значения измерений;
- *заканчивается на* – указывается подстрока, на которую *должны заканчиваться* импортируемые значения измерений;
- *не заканчивается на* – указывается подстрока, на которую *не должны заканчиваться* импортируемые значения измерений;

Для измерений типа вещественное и целое число дополнительно доступны следующие основные фильтры:

- *<, <=, >, >=* – соответствующие операции сравнения чисел;
- *в интервале* – выбираются значения, *попадающие* в заданный числовой интервал;
- *вне интервала* – выбираются значения *за пределами* заданного числового интервала;

Для измерений типа дата/время дополнительно доступны следующие основные фильтры:

- *в интервале* – выбираются записи, для которых измерение *лежит в заданном диапазоне* дат;
- *вне интервала* – выбираются записи, для которых измерение *не входит в заданный диапазон* дат;
- *последний* – выбираются записи, для которых измерение лежит в указанном промежутке времени (промежуток задается), предшествующем указанной дате;
- *не последний* –- выбираются все записи, кроме тех, для которых измерение лежит в указанном промежутке времени, предшествующем указанной дате.

На этом же шаге мастера имеется возможность настроить динамические срезы в перечисляемом списке **Тип фильтра**. В нем три установки:

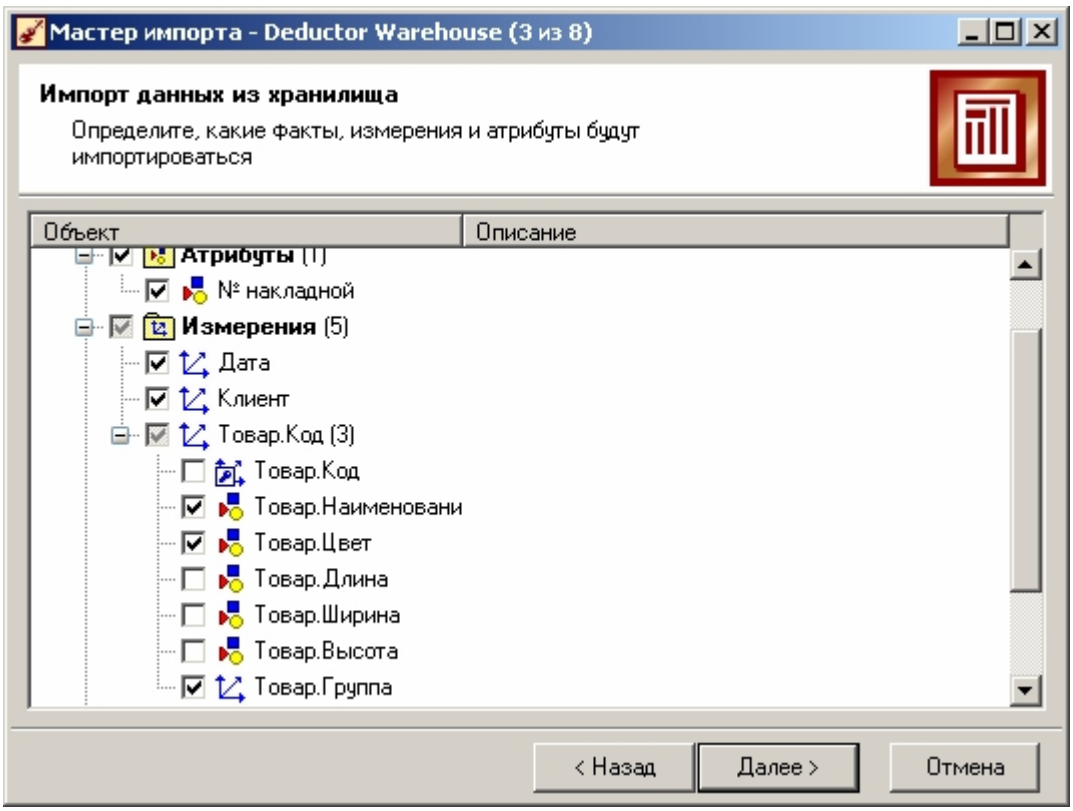
- *статический фильтр* – срез не меняется ни при каких условиях;
- *пользовательский фильтр* – при каждом выполнении узла импорта пользователю будет выводиться окно, в котором он сможет указать требуемые разрезы по измерению(измерениям);
- *фильтр с использованием переменных* – значения для формирования срезов будут браться из переменных проекта Deductor или переменных приложения (см. раздел «Использование переменных»).

Последние две настройки позволяют строить динамические отчеты, в которых пользователю предоставляется только интересующая его информация, а конкретные условия фильтрации он выбирает в момент импорта данных, либо условия фильтрации передаются через переменные при пакетном/серверном запуске сценария.

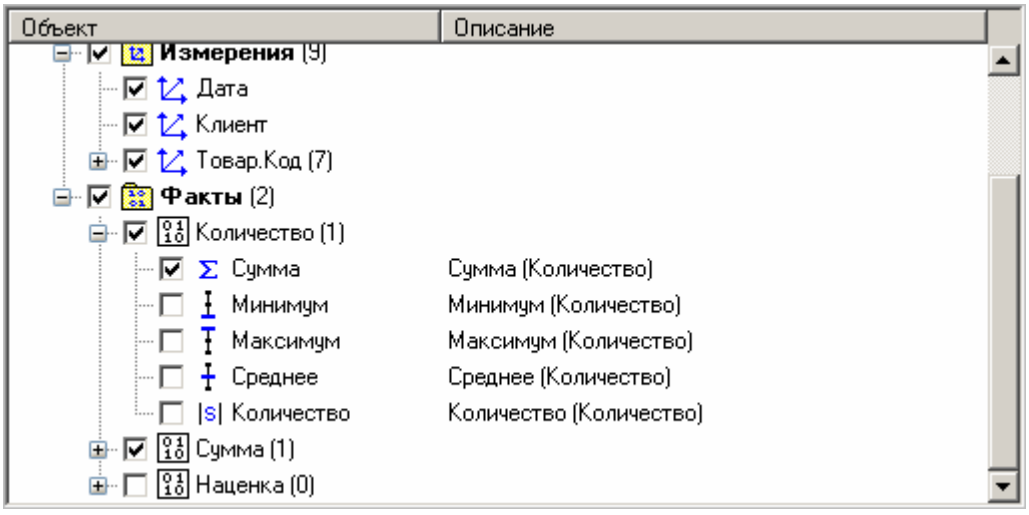
Например, пользователю требуется получать информацию об отгрузках по каждому дилеру. Можно было бы подготовить отдельную ветку сценария и отчеты по каждому дилеру, либо фильтровать по дилеру в кубе. Однако первый вариант не подходит из-за того, что число дилеров может быть очень большим либо постоянно меняться. Тогда пришлось бы каждый раз перестраивать систему отчетов. С помощью динамического фильтра можно эффективно решать подобные проблемы. На этапе импорта данных из хранилища пользователь сам сможет указать, в каком именно разрезе ему нужны данные, и работать только с ними. При этом для всех возможных разрезов готовятся всего один сценарий обработки и одна система отчетов.

Пример

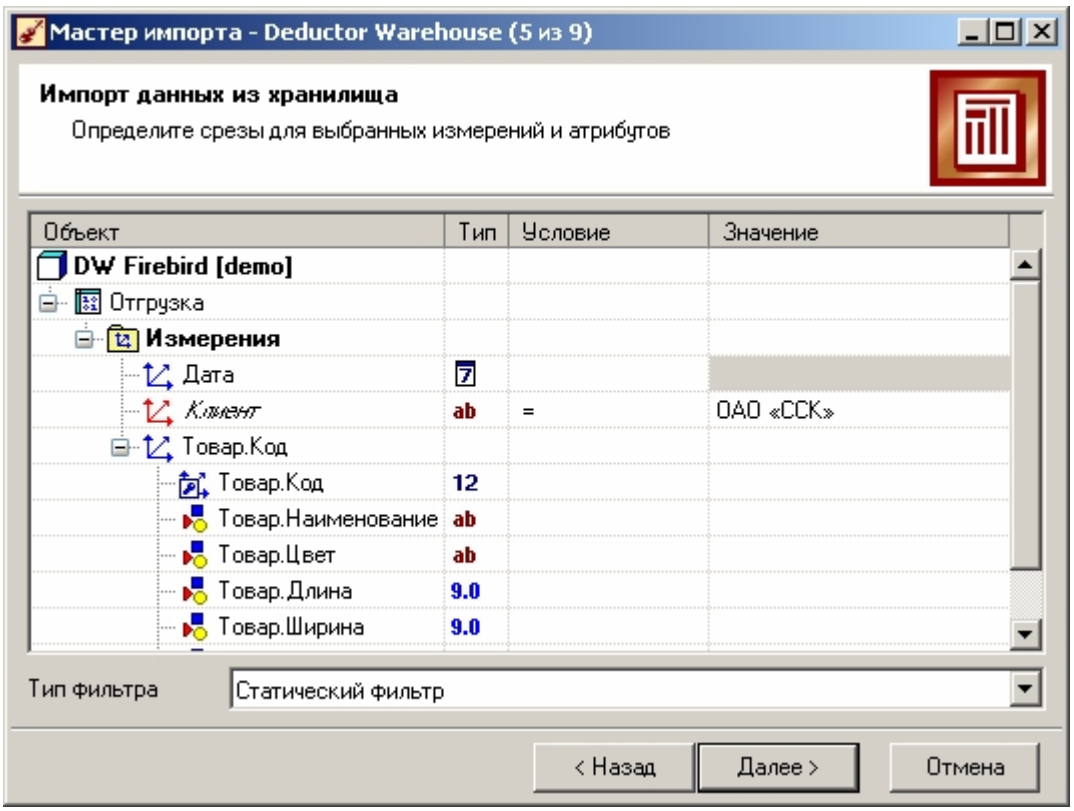
Импортируем из хранилища данные о количестве отгруженного товара в разрезе дат и товаров по клиенту *ОАО «ССК»*, оставив атрибут товара *Цвет*. Вызовем мастер импорта и выберем источник «Deductor Warehouse». Далее выберем наше хранилище из списка и процесс *Отгрузка*. Отметим измерения, импортируемые из хранилища, как показано на рисунке (атрибут «№ накладной» импортировать не будем). Обратите внимание, что в измерении *Товар.Код* имеется возможность выбрать товарную группу (иллюстрация иерархии).



Далее укажем импортируемые факты (факт – количество, вариант агрегации – сумма).



На последнем шаге зададим статический фильтр по клиенту.



Будет получен набор данных следующего содержания.

	Дата	№ накладной	Клиент	Товар.Код			Количество
				Товар.Группа	Товар.Наименование	Товар.Цвет	Сумма
	01.03.2004	051/2004	ОАО «ССК»	Силикатный	ТКСМ-100	Белый	1000
▶	12.03.2004	054/2004	ОАО «ССК»	Керамический	М100-125	Красный	2000

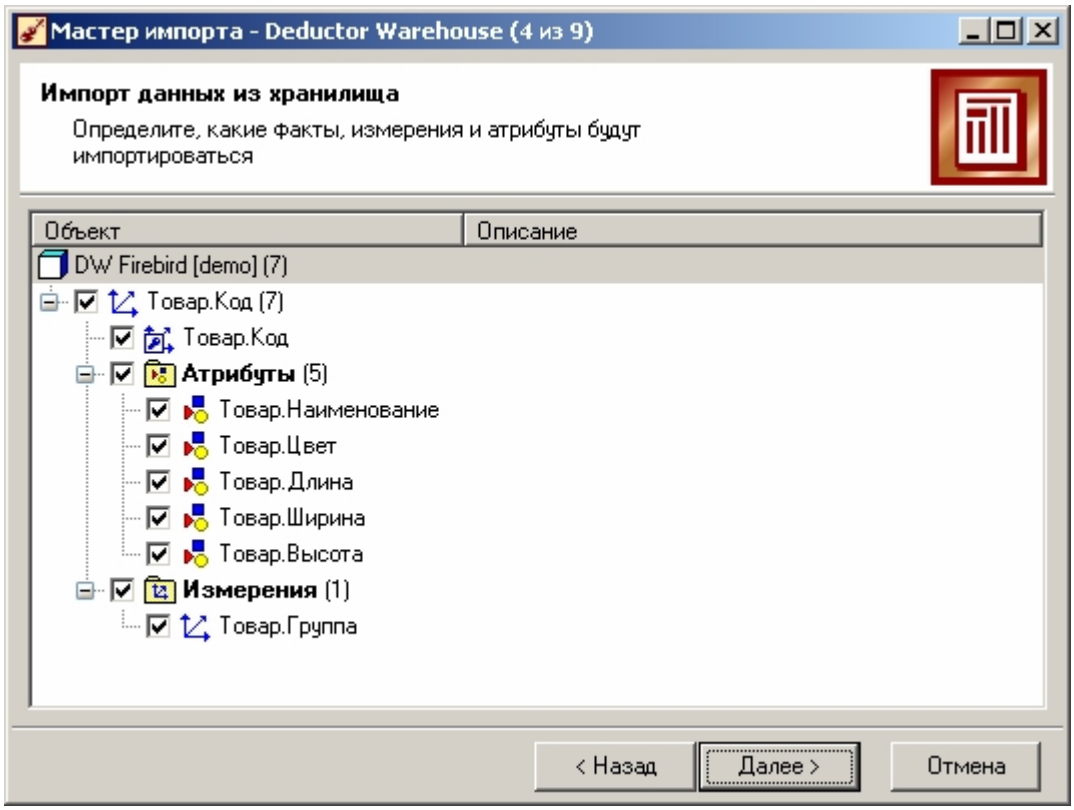
То есть получили количество в разрезе товара, даты и конкретного клиента с указанием номера накладной.

Импорт измерения

Если интересуют сведения по измерению, то необходимо выбрать интересующее измерение из списка доступных объектов базы данных. В ветке «Измерение» дерева объектов Deductor Warehouse представлены наименования измерений, содержащихся в хранилище. Дальше нужно выбрать атрибуты текущего измерения, которые будут включены в выходной набор данных и связанные измерения (если существует иерархия измерений). Для этого требуемые атрибуты или связанные измерения помечаются флагом, расположенным слева от имени атрибута. Неотмеченные объекты не будут включены в итоговый набор.

Пример

Импортируем из хранилища данные о товарах, которые продает компания, и информацию о том, в какую группу входит тот или иной товар. Для этого выберем объекты измерения *Товар.Код* так, как это показано на следующем рисунке.



В результате будет получена таблица следующего содержания:

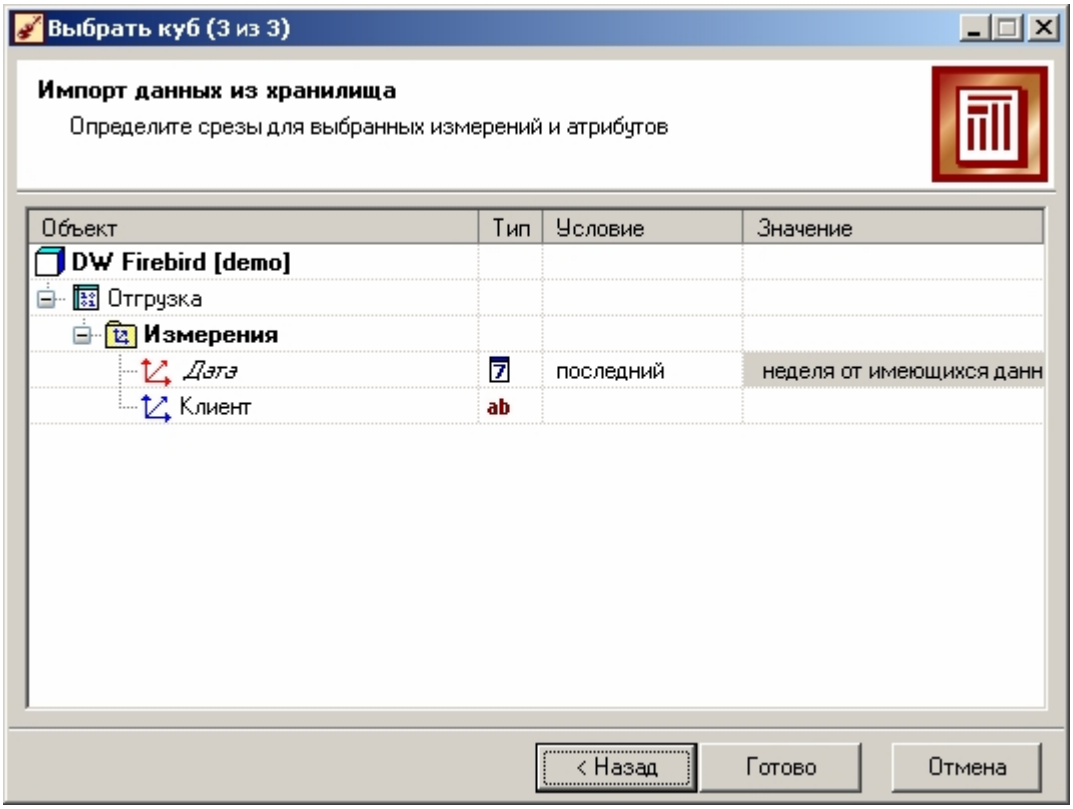
Товар.Код	Товар.Наименование	Товар.Цвет	Товар.Длина	Товар.Ширина	Товар.Высота	Товар.Группа
1	ТКСМ-100	Белый	250	120	85	Силикатный
2	ТКСМ-125	Тонированный	190	60	88	Силикатный
3	М100-125	Красный	250	120	65	Керамический
4	М100-175	Песочный	250	120	65	Керамический

Кубы в хранилище данных

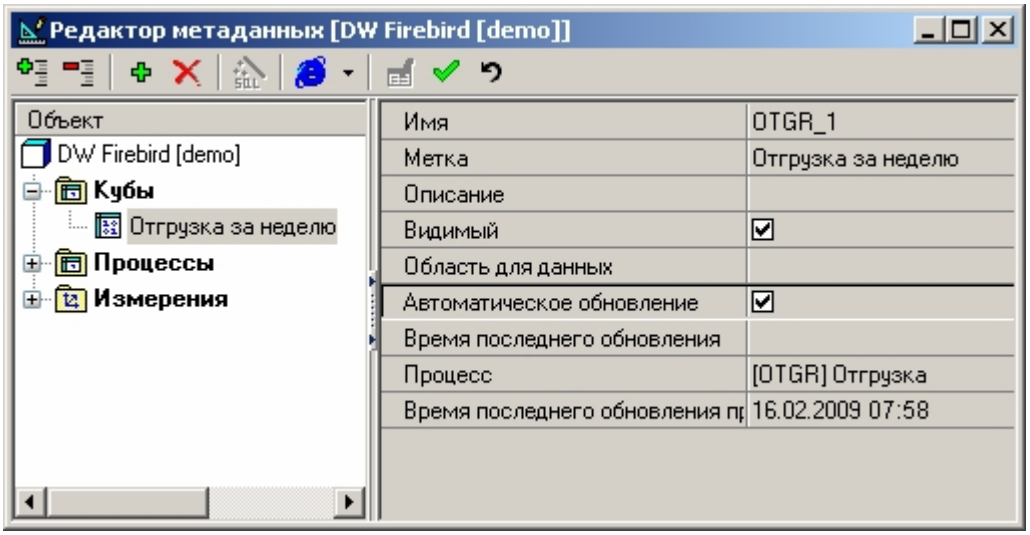
В Deductor Warehouse кроме импорта из процесса/измерения имеется возможность «доставать» информацию из так называемых кубов, которые представляют собой заранее настроенные срезы процесса. Они «обсчитываются» заранее и хранятся в отдельных таблицах, поэтому операция импорта из куба выполняется значительно быстрее, чем непосредственно из процесса. Кубы полезны, когда в хранилище миллионы записей, а время отклика на запрос критично и нужно его минимизировать. В этом случае выгодно настроить срез в виде куба. Недостатком подхода является то, что при любом добавлении данных в процесс куб приходится заново «пересчитывать», что требует определенного времени. Тем не менее, если эти регламентные процедуры проводятся, скажем, в ночное время, то особой проблемы это не вызывает.

Пример

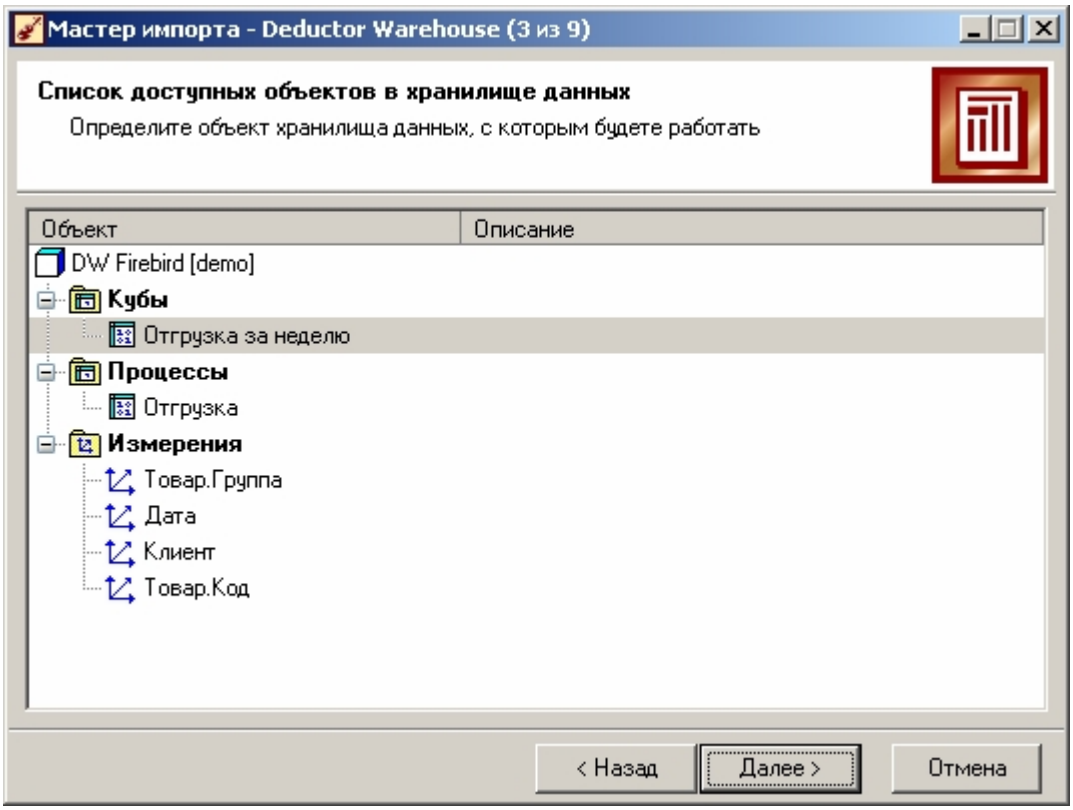
Настроим куб, в котором будет храниться информация об отгрузках по клиентам за последнюю неделю. Откроем «Редактор метаданных», войдем в режим редактирования и добавим куб. Шаги по его созданию аналогичны шагам импорта из процесса. Выберем измерения *Клиент* и *Дата*, все доступные факты и по измерению *Дата* настроим статический фильтр с условием «последний» и значением «неделя от имеющихся данных».



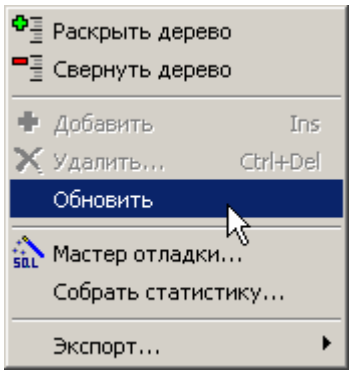
Дадим кубу название *Отгрузка за неделю*. В редакторе для него стал доступен специфичная опция в виде флага **Автоматическое обновление**. Поднятый флаг говорит о том, что при попадании новых данных в процесс куб будет автоматически пересчитываться, при этом в поле «Время последнего обновления» будет заноситься дата и время последнего обновления.



Теперь куб готов к работе. В сценарии при вызове мастера импорта из хранилища помимо процесса и измерения появится возможность выбрать куб.



Имеется возможность принудительно перестроить куб. Для этого в Редакторе метаданных нужно встать на нужный объект и в контекстном меню нажать кнопку **Обновить**.

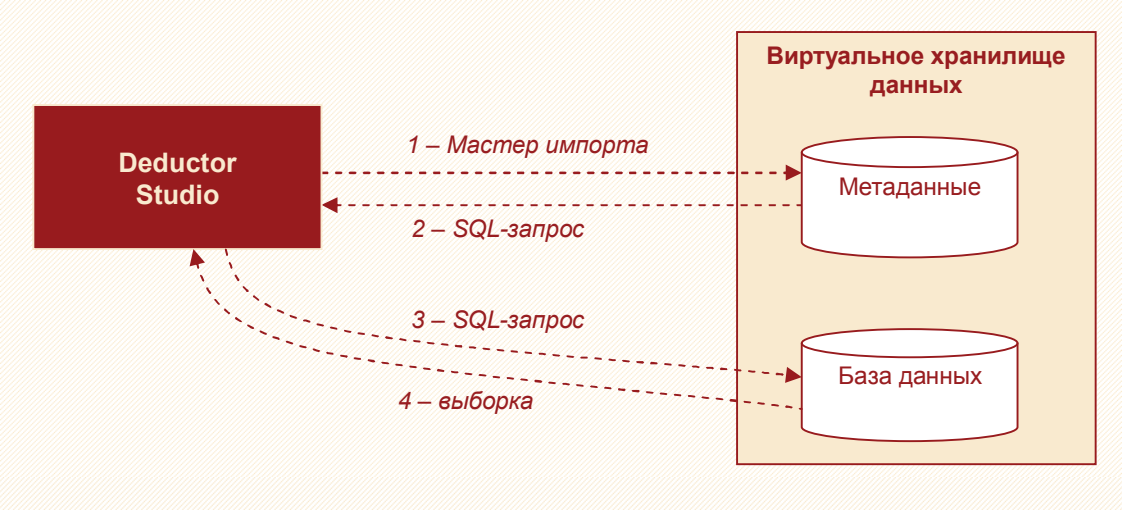


Виртуальное хранилище Virtual Warehouse

В случае отказа от традиционного хранилища данных альтернативой является применение *виртуального хранилища данных* – **Virtual Warehouse 5**.

Отличие здесь в том, что данные из учетных систем не переносятся в специальную базу данных, а извлекаются и преобразуются к требуемому виду непосредственно при выполнении аналитических запросов к учетной базе данных. Пользователь работает с Virtual Warehouse через такой же семантический слой, как и в случае с традиционным хранилищем, т.е. он оперирует многомерными понятиями, хотя обращается к обычной реляционной базе данных, не ориентированной на аналитическую обработку.

Виртуальное хранилище данных – это логически единый источник информации, включающий в себя базу данных с анализируемыми сведениями и разработанный для нее семантический слой, хранящийся отдельно.



Работа с настроенным виртуальным хранилищем происходит следующим образом. Сначала с помощью Мастера импорта в Deductor Studio пользователь задает информацию, которую он хочет получить из хранилища данных. Система самостоятельно «за кулисами» формирует SQL запрос к базе данных. Мастер импорта как бы помогает «перевести» то, что хочет получить аналитик (который рассуждает в терминах измерения, факт, процесс) на язык SQL. Дальше этот SQL-запрос выполняется (идет обращение к базе данных), подписываются нужным образом поля, производятся другие вспомогательные операции, и результат становится доступен пользователю, который «видит» данные через семантический слой виртуального хранилища данных.

Достоинства Virtual Warehouse:

- Минимизация дискового пространства.
- Пользователь оперирует данными на семантическом уровне.
- Быстрота настройки и запуска.
- Работа с текущими, детализированными данными.

Недостатки Virtual Warehouse:

- Увеличивается время обработки запросов.
- Время обработки запросов к виртуальному хранилищу данных может значительно превышать соответствующие показатели для традиционного хранилища. Структуры оперативных баз данных, рассчитанные на интенсивное обновление одиночных записей, в высокой степени нормализованы. Для выполнения же аналитического запроса требуется объединение большого числа таблиц, что также приводит к снижению быстродействия.

- Увеличивается нагрузка на учетные системы, т.к. база данных должна обеспечивать как текущую обработку транзакция, так и выполнение ресурсоемких аналитических запросов.
- Требуется постоянная доступность всех источников данных.
- Временная недоступность хотя бы одного из источников может привести либо к невыполнению аналитических запросов, либо к неверным результатам.
- Отсутствие единого непротиворечивого взгляда на объект управления. Это случается при наличии нескольких учетных систем, когда информация между ними обновляется асинхронно.

Deductor Studio имеет встроенную реализацию виртуального хранилища данных – Virtual Warehouse 5. Для аналитика работа с таким хранилищем в плане пользовательского интерфейса абсолютно идентична работе с традиционным хранилищем. Исключена только возможность загрузки информации в виртуальное хранилище данных.

Проектирование семантического слоя Virtual Warehouse относится скорее к задаче подключения к источнику данных и требует квалификации администратора базы данных и знания языка SQL-запросов. Эта работа должна выполняться специалистом с соответствующей подготовкой. Вопросы настройки Virtual Warehouse подробно освещены в специальном руководстве по настройке виртуального хранилища данных, поставляемым с системой.

Примечание

Возможность работы с Virtual Warehouse имеется только в Deductor Enterprise.

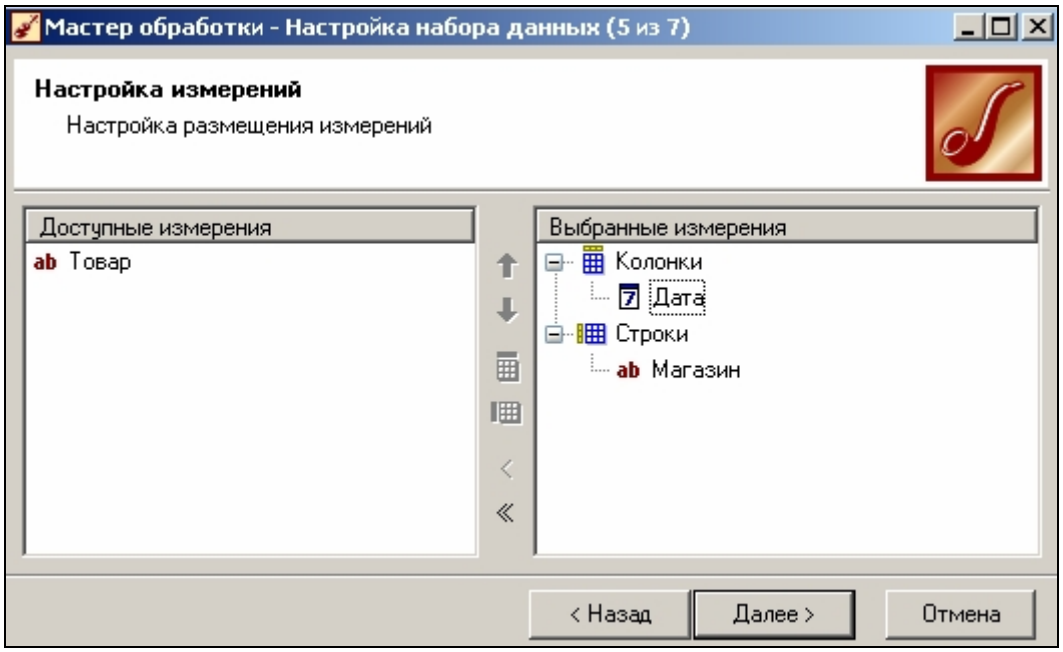
Работа с OLAP-кубом

Кросс-таблица

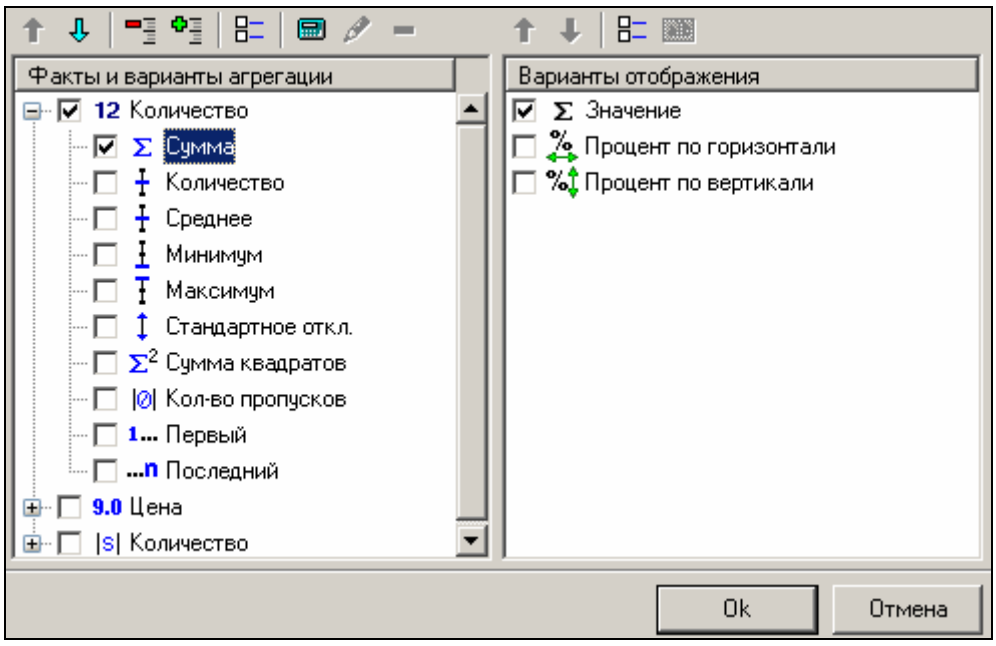
Кросс-таблица представляет собой способ отображения многомерных данных на 2-х мерной плоскости. Следовательно, прежде чем строить эту таблицу, необходимо указать измерения и факты. Например, если рассматривать данные из предыдущего примера, измерения – это *Месяц*, *Товар.Наименование*, *Товар.Группа*, а факт – *Количество* (проданного товара).

Размещение измерений

Измерения могут быть размещены в строках и столбцах куба. В мастере визуализации изначально весь список доступных измерений отображается в окне «Настройка размещения измерений». Нажимая кнопки  и  , можно размещать выбранные измерения в колонках и стоках таблицы. Переносить измерения в нужную область можно и при помощи операции drag&drop. Пример приведен на рисунке.

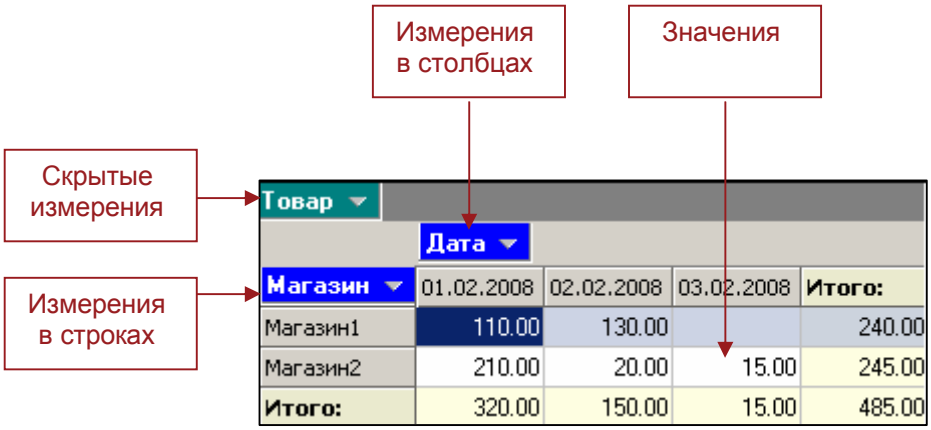


На следующем шаге нужно выбрать факты, отображаемые в кубе на пересечении измерений, функцию их агрегации (объединения) и вариант отображения. В данном примере факт *Количество* будет суммироваться, а отображаться будет его абсолютное значение.



Как видно на рисунке, отображать можно не только абсолютные значения фактов, но и долю в процентах по строкам или колонкам куба.

Построенный куб для данного примера будет выглядеть следующим образом.



Измерения в кубе изображаются специальными полями. Синий фон указывают для измерений, участвующих в построении двухмерного среза таблицы. Зелеными полями отображаются скрытые измерения, не участвующие в построении среза. Есть возможность перестраивать таблицу с помощью мыши «на лету». Сделать это можно, если перетаскивать поля с заголовками измерений.

Приведем различные варианты изменения таблицы таким способом.

Сделать измерение, участвующее в построении таблицы, скрытым. Для этого нужно перетащить поле с заголовком измерения в строку со скрытыми измерениями. Перетащим, например, поле *Дата*.

Товар / Дата				
Магазин	01.02.2008	02.02.2008	03.02.2008	**Итого:**
Магазин1	110.00	130.00		240.00
Магазин2	210.00	20.00	15.00	245.00
Итого:	320.00	150.00	15.00	485.00

Результат:

Дата / Товар	
Магазин	Σ Количество
Магазин1	240.00
Магазин2	245.00
Итого:	485.00

Сделать скрытое измерение участвующим в построении таблицы. При этом его можно добавить к измерениям в строках или столбцах. В исходной таблице перетащим измерение *Товар* и поместим его рядом с измерением *Магазин*.

Товар / Дата				
Магазин	01.02.2008	02.02.2008	03.02.2008	**Итого:**
Магазин1	110.00	130.00		240.00
Магазин2	210.00	20.00	15.00	245.00
Итого:	320.00	150.00	15.00	485.00

Результат:

		Дата			
− + Магазин	**Товар**	01.02.2008	02.02.2008	03.02.2008	**Итого:**
Магазин1	Товар 1	100.00			100.00
	Товар 2	10.00			10.00
	Товар 3		110.00		110.00
	Товар 4		20.00		20.00
	Итого:	110.00	130.00		240.00
Магазин2	Товар 1	90.00			90.00
	Товар 2		20.00		20.00
	Товар 3	120.00			120.00
	Товар 4			15.00	15.00
	Итого:	210.00	20.00	15.00	245.00
Итого:		320.00	150.00	15.00	485.00

При этом можно было расположить измерение *Товар* как слева, так и справа от измерения *Магазин*.

При помощи кнопок и измерения можно сворачивать и разворачивать.

Свернуть все измерения можно нажатием кнопки .

		Дата			
Магазин	Товар	01.02.2008	02.02.2008	03.02.2008	Итого:
Магазин1		110.00	130.00		240.00
Магазин2		210.00	20.00	15.00	245.00
Итого:		320.00	150.00	15.00	485.00

Свернуть лишь одно измерение можно нажатием кнопки

		Дата			
Магазин	Товар	01.02.2008	02.02.2008	03.02.2008	Итого:
Магазин1		110.00	130.00		240.00
Магазин2	Товар 1	90.00			90.00
	Товар 2		20.00		20.00
	Товар 3	120.00			120.00
	Товар 4			15.00	15.00
	Итого:	210.00	20.00	15.00	245.00
Итого:		320.00	150.00	15.00	485.00

Аналогично можно разворачивать измерения, используя кнопки и

При необходимости можно поменять местами два измерения.

		Дата		
Товар	Магазин	01.02.2008	02.02.2008	03.02.2008
Товар 1	Магазин1	100.00		
	Магазин2	90.00		
Товар 2	Магазин1	10.00		
	Магазин2		20.00	
Товар 3	Магазин1		110.00	
	Магазин2	120.00		
Товар 4	Магазин1		20.00	
	Магазин2			15.00

Такое представление данных позволяет увидеть продажи каждого товара за день.

Изменять расположение измерений можно, используя операцию транспонирования куба. В результате данные, ранее отображавшиеся в строках, отображаются в столбцах, а данные в столбцах преобразуются в строки. Транспонирование во многих случаях позволяет оперативно

сделать таблицу более удобной для восприятия. Пример транспонирования таблицы представлен на рисунке.

Исходный куб

	Товар			
Дата	Товар 1	Товар 2	Товар 3	Товар 4
01.02.2008	190.00	10.00	120.00	
02.02.2008		20.00	110.00	20.00
03.02.2008				15.00

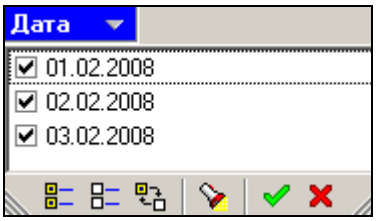
Такая таблица может не поместиться в пределы экрана и будет неудобна для просмотра, когда товаров много. В результате транспонирования она примет более удобный вид.

Транспонированный куб

	Дата		
Товар	01.02.2008	02.02.2008	03.02.2008
Товар 1	190.00		
Товар 2	10.00	20.00	
Товар 3	120.00	110.00	
Товар 4		20.00	15.00

Для того чтобы применить операцию транспонирования следует воспользоваться кнопкой  **Транспонировать таблицу** на панели инструментов, либо нажать Ctrl+T.

В приведенных выше примерах куб строится по всем значениям измерений. Однако иногда возникает необходимость построить куб в разрезе лишь некоторых значений, например, за первый и третий кварталы года. Включать или исключать значения измерений в таблице можно, нажав на треугольник в поле заголовка интересующего измерения. Например, если нажать на треугольник в поле заголовка измерения *Дата*, откроется список его значений.



Включать или исключать значения можно, устанавливая или снимая галочку рядом с наименованием. Кнопки внизу списка означают:

-  – установить все отметки;
-  – убрать все отметки;
-  – инверсия: все сброшенные отметки устанавливаются, а все установленные – сбрасываются;
-  – выделение значений по маске;
-  – применить все изменения для построения таблицы;
-  – отменить все сделанные изменения.

Отфильтрованные измерения выделяются красным фоном. Выбирать фильтруемые значения можно и для скрытых измерений.

Способы агрегации и отображения фактов

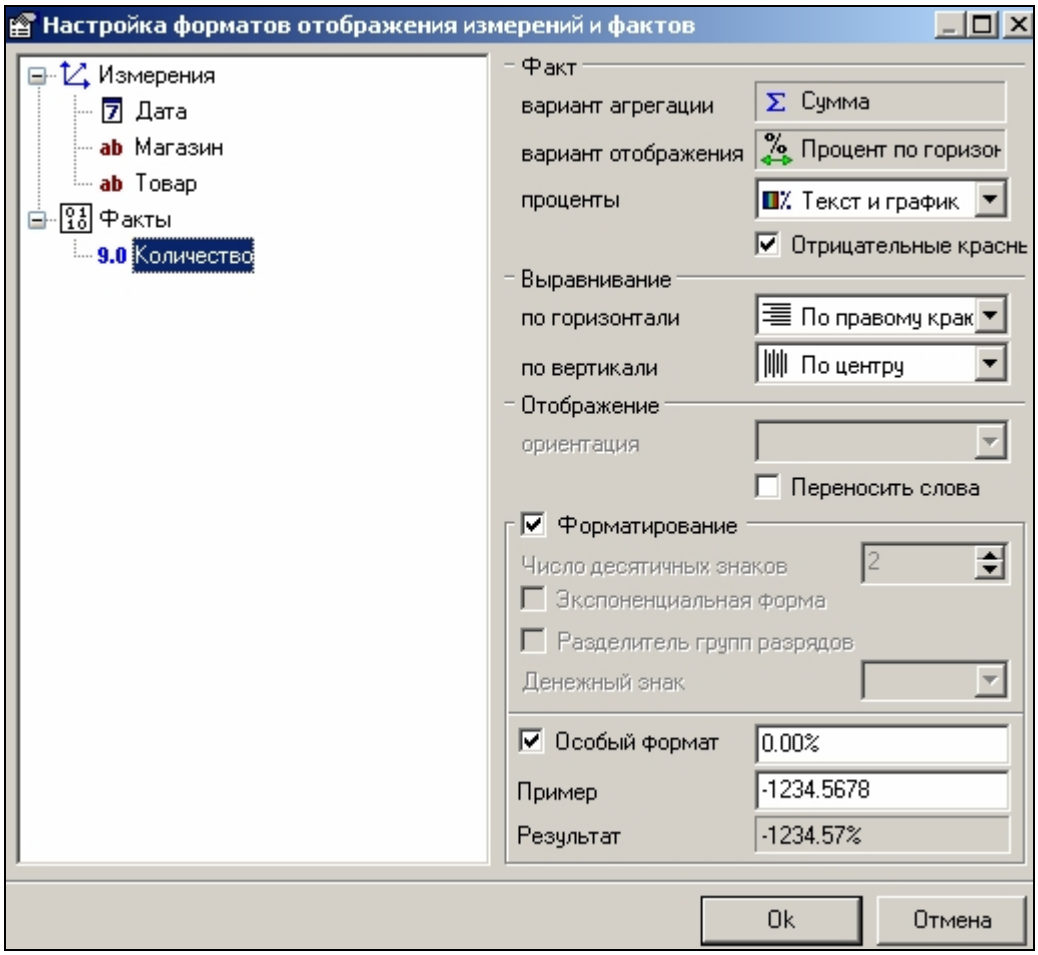
Предусмотрено несколько способов агрегации фактов в кросс-таблице:

- *сумма* – вычисляется сумма объединяемых фактов;
- *минимум* – среди всех объединяемых фактов в таблице отображается только минимальный;
- *максимум* - среди всех объединяемых фактов в таблице отображается только максимальный;
- *среднее* – вычисляется среднее значение объединяемых фактов;
- *количество* – в кубе будет отображаться количество объединенных фактов.

Для изменения способа агрегации фактов нужно вызвать окно «Настройка фактов», нажав кнопку  на панели инструментов.

В кубе по умолчанию факты отображаются с двумя знаками после запятой и выровнены по правому краю ячейки. Доступна возможность изменить форматирование ячеек таблицы, вызвав окно «Настройка форматов» (кнопка  на панели инструментов).

Пример окна на рисунке.



Назначение полей следующее:

- *Проценты* – в случае отображения процентов по горизонтали или вертикали, можно задать способ представления в кубе в виде числа, графика или комбинации числа и графика.
- *Отрицательные красным* – отображение красным цветом отрицательных значений фактов.
- *Выравнивание* – определяет выравнивание значений в ячейках по горизонтали и вертикали. Может принимать значения: «По левому краю», «По центру», «По правому краю»;
- *Отображение* – задает ориентацию меток текста (повороты на 90 и 270 градусов) и опцию переноса по словам при недостатке ширины поля;
- *Форматирование* – применить особое форматирование или оставить все как есть;
- *Число десятичных знаков* – определяет число знаков после запятой;
- *Экспоненциальная форма* – если установлен, то число будет отображаться в экспоненциальной форме. Например, число 153,47 будет выглядеть 1,5347E+2;
- *Разделитель групп разрядов* – отображать или не отображать разделитель разрядов, т. е. число может выглядеть так 1289 или так 1,289;
- *Денежный знак* – можно в конце значения добавить знак денежной единицы;
- *Особый формат* – позволяет задать формат с помощью строки формата. В поле «Результат» отображается результат применения формата к числу, введенному в поле «Пример».

Селектор – фильтрация данных в кубе

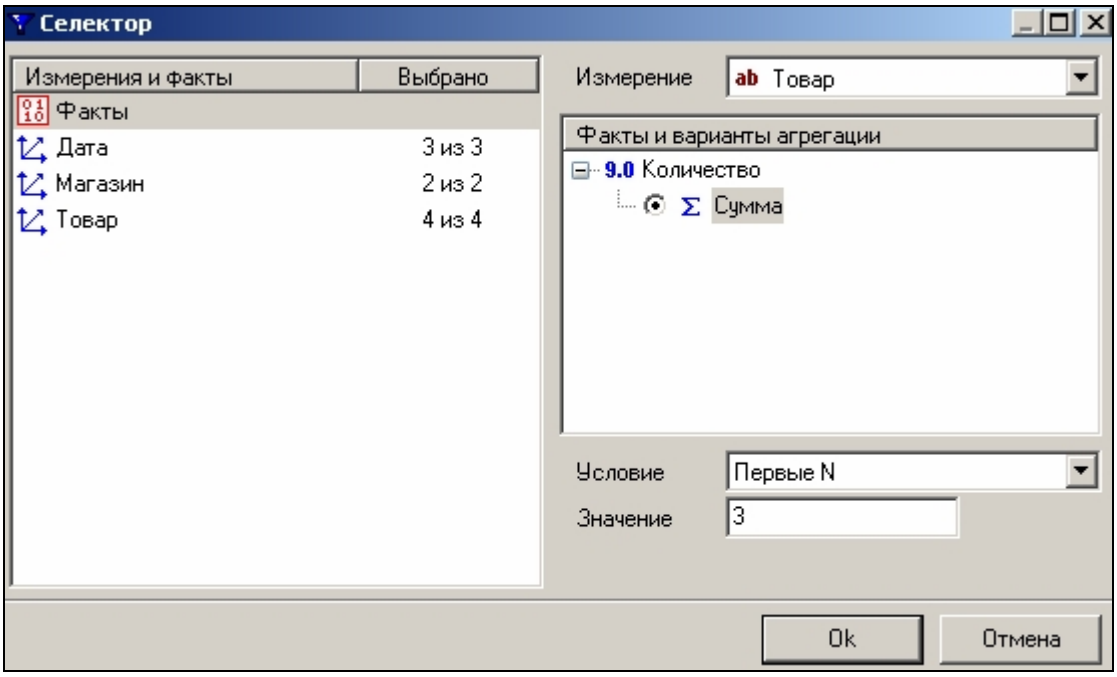
Селектор является мощным средством фильтрации данных в кросс-таблице. Фильтрация может производиться двумя способами:

по значениям фактов;

по значениям измерений путем непосредственного выбора значений из списка или отбора их по условию. Фильтрация задается отдельно по каждому измерению.

Чтобы приступить к работе с селектором, достаточно на панели инструментов нажать на кнопку  **Селектор...**, после чего будет открыто окно селектора.

В окне селектора слева отображаются поле «Факты» и все доступные измерения.



Это пример окна для фильтрации данных по значениям фактов. В этом случае в правой части окна задаются следующие параметры:

- *Измерение*. Фильтрация подразумевает, что в таблице останется лишь часть значений некоторого измерения. Это поле как раз и задает измерение, значения которого будут отфильтрованы;
- *Факты и вариант агрегации*. В кубе может содержаться один и более фактов. Фильтрация будет происходить по значениям выбранного здесь факта. К нему будет применена функция агрегации, в соответствии с которой следует выполнить отбор записей. В результате будут выбраны только те записи, агрегированные значения которых удовлетворяют выбранному условию;
- *Условие*. Условие отбора записей по значениям выбранного факта.

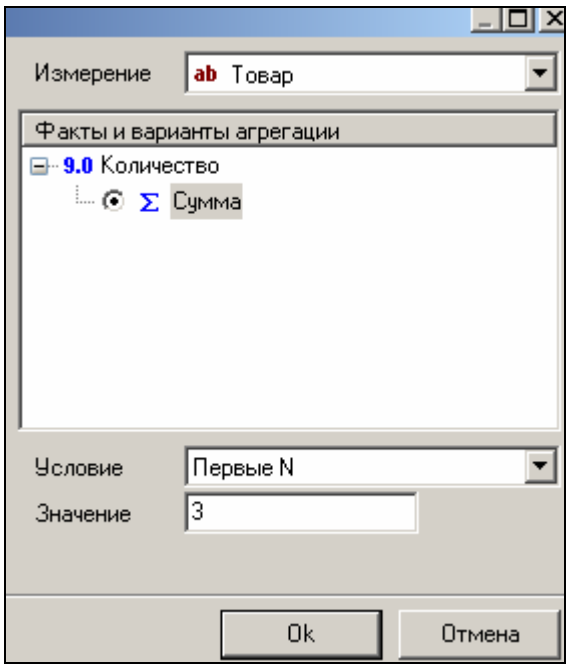
Поле «Условие» может принимать следующие значения:

- *Первые N*. Значения измерения сортируются в порядке убывания факта, и выбираются первые N значений измерений. Таким образом, можно, например, находить лидеров продаж – первые 10 наиболее продаваемых товаров, или первые 5 наиболее удачных дней;
- *Последние N*. Значения измерения сортируются в порядке убывания факта, и выбираются последние N значений измерений. Например, 10 наименее популярных товаров;
- *Доля от общего*. Значения измерения сортируются в порядке убывания факта. В этой последовательности выбирается столько первых значений измерения, что они в сумме давали заданную долю от общей суммы. Например, можно отобрать клиентов, приносящих 80% прибыли – группа A по ABC классификации;
- *Диапазон*. Результатом отбора будут записи, для которых значение соответствующего факта лежит в заданном диапазоне;
- *Больше*. Будут отобраны записи, значение соответствующего факта для которых будет больше указанного значения;
- *Больше или равно*. Будут отобраны записи, значение соответствующего факта для которых будет больше или равно указанного значения;

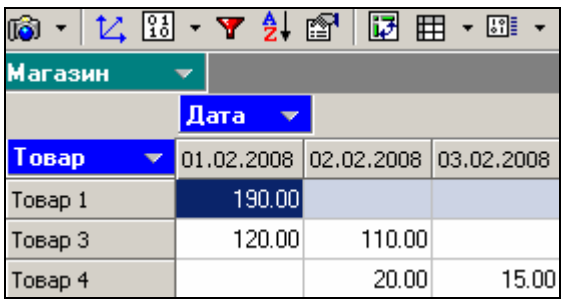
- *Меньше*. Будут отобраны записи, значение соответствующего факта для которых будет меньше указанного значения;
- *Меньше или равно*. Будут отобраны записи, значение соответствующего факта для которых будет меньше или равно указанного значения;
- *Равно*. Будут отобраны записи, значение соответствующего факта для которых будет равно указанному значению;
- *Не равно*. Будут отобраны записи, значение соответствующего факта для которых будет не равно указанному значению.

Пример фильтрации по фактам

Пусть нам нужно определить товары, пользующиеся наибольшим спросом. Исходный куб содержит 4 товара. Применим к нему селектор.



При проведении данной операции значок фильтра становится красным .В результате получим 3 наиболее популярных товаров.



Такую выборку можно получить по любому факту. В данном примере это количество. Поэтому мы наблюдаем наиболее продаваемые товары.

Если, к примеру, отфильтровать по наценке, то получим наиболее прибыльный товар.

При работе с кубом доступна операция детализации ячейки. При нажатии кнопки **Детализация ячейки** для выделенной ячейки в нижней части экрана открывается таблица детализации. Можно выделить сразу несколько ячеек, в детализации отображаются все записи исходного набора данных, которые вносят свой вклад в формирование значения выделенной ячейки (ячеек).

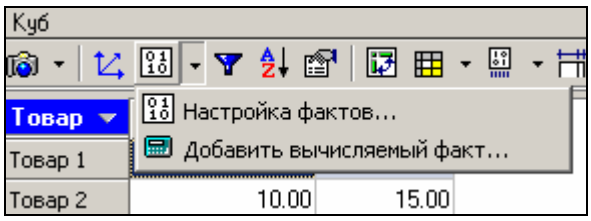
В таблице детализации будут находиться столбцы, для которых при настройке назначения полей куба было указано назначение: измерение, факт или информационное. Информационные поля не отображаются в самом кубе, но отображаются при детализации. Например, информацию о номере выписанного счета очень часто показывать в кубе не имеет смысла, но если указать для него назначение *Информационное*, то для любой отгрузки будет легко его узнать, вызвав окно детализации.

Функция «Калькулятор»

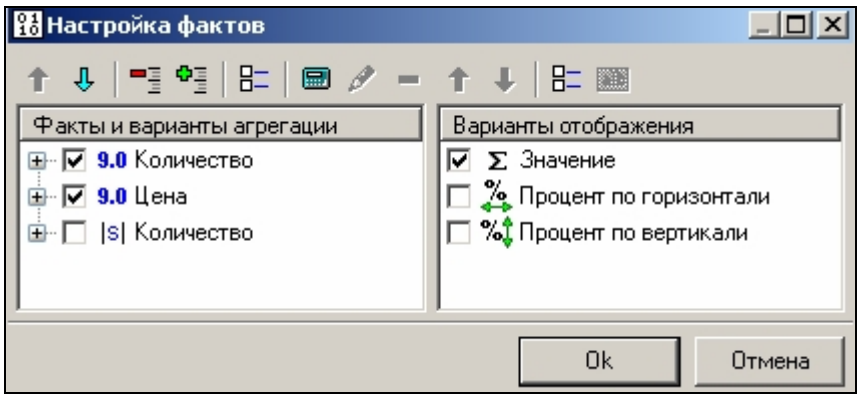
Иногда для более корректного отображения данных в OLAP-кубе требуется вычисление новых фактов на основе уже имеющихся. Функция «Калькулятор», встроенная в визуализатор **Куб**, позволяет проводить вычисления «на лету» непосредственно в отчете.

Добавить вычисляемый факт можно следующими способами.

- На панели инструментов выбрать кнопку **Добавить вычисляемый факт**.



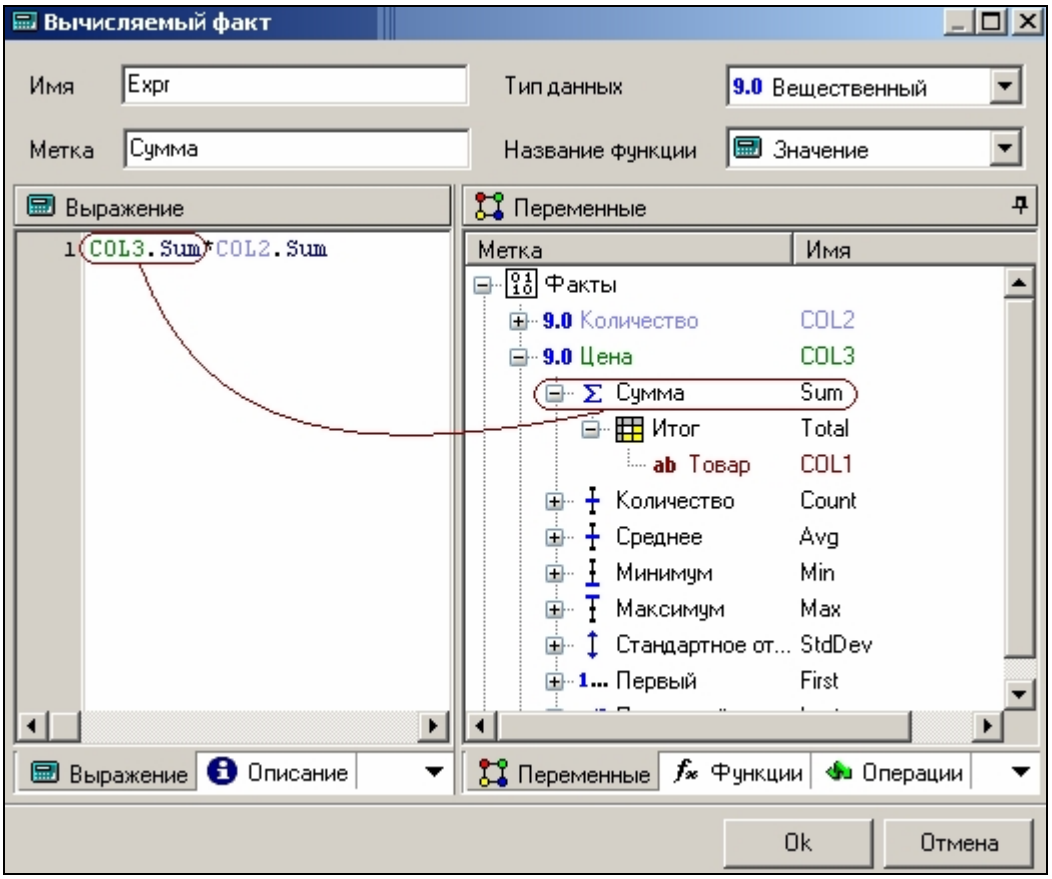
- В окне «Настройка фактов» выбрать кнопку **Добавить вычисляемый факт**.



Пример

Пусть в исходных данных содержится: *Наименование*, *Количество* и *Цена* товара. Необходимо определить сумму по каждому виду продукции.

В калькуляторе в закладке **Переменные** выбираем для полей *Цена* и *Количество* способ агрегации фактов – Сумма, Это позволит при расчете использовать значения, отнесённые к конкретному виду товара.



В результате добавления нового факта результирующая таблица в OLAP-кубе имеет вид:

Товар	Σ Количество	Σ Цена	Сумма
Товар 1	100	25.00	2 500.00
Товар 2	10	15.00	150.00
Товар 3	110	34.00	3 740.00
Товар 4	20	21.00	420.00
Итого:	240	95.00	22 800.00

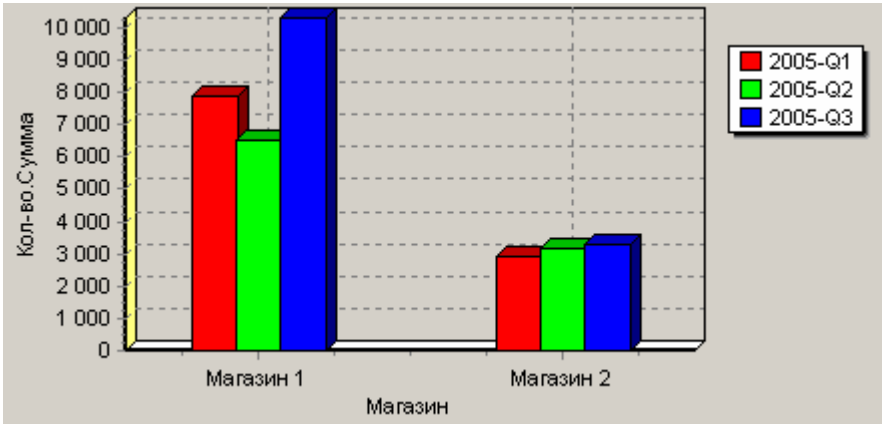
Кросс-диаграмма

Кросс-диаграмма представляет собой график заданного типа, построенный на основе куба. Основное отличие кросс-диаграммы от обычной диаграммы в том, что она однозначно соответствует текущему состоянию куба и при любых ее изменениях изменяется соответственно.

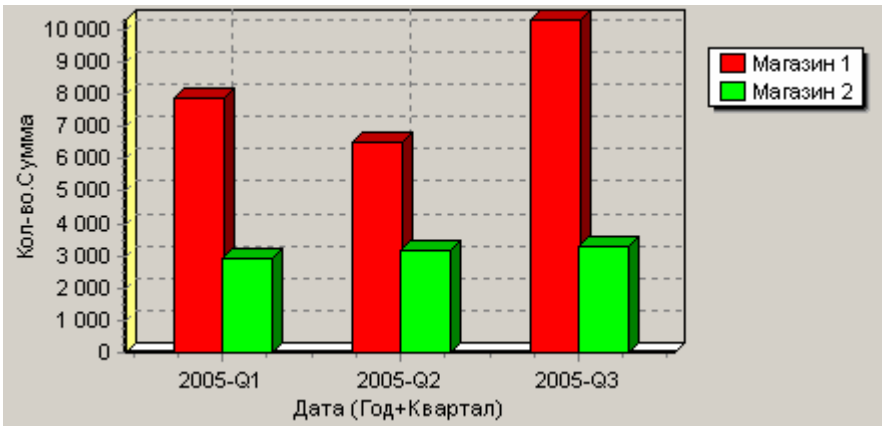
Приведем пример кросс-диаграммы для следующего куба.

	Дата (Год+Квартал)			
Магазин	2005-Q1	2005-Q2	2005-Q3	**Итого:**
Магазин 1	7 856	6 504	10 308	24 668
Магазин 2	2 894	3 172	3 313	9 379
Итого:	10 750	9 676	13 621	34 047

Кросс-диаграмма для него имеет следующий вид.



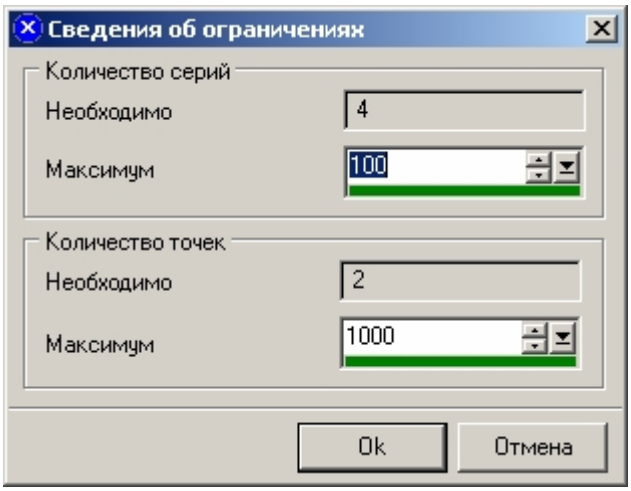
На ней можно наблюдать поквартальную тенденцию продаж в различных магазинах. К кросс-диаграмме, так же как и к кубу, можно применять транспонирование. Результат транспонирования будет следующий:



Кросс-диаграмма может строиться по нескольким фактам. Выбрать интересующие факты можно, нажав кнопку .

При построении диаграммы вводятся ограничения числа серий и числа точек в каждой серии, отображаемых на графике. Данное ограничение вызвано, с одной стороны, большими вычислительными затратами при построении диаграммы, а с другой, сложностью восприятия больших диаграмм. Кнопка **Ограничения** на панели инструментов будет иметь синий цвет, если ограничения не превышены, и красный в противоположном случае.

Максимальное количество отображаемых серий и точек можно настроить, нажав на кнопку **Ограничения** на панели инструментов.



Щелчок по этой кнопке выводит окно «Сведения об ограничениях», в котором представлена информация:

- *Количество серий* – сколько серий необходимо для того, чтобы отобразить все данные из кросс-диаграммы и максимальное количество отображаемых серий.
- *Количество точек* – сколько точек необходимо для того, чтобы отобразить все данные из кросс-диаграммы и максимальное количество отображаемых точек.

Под «точками» понимаются группы столбцов, которые соответствуют значениям измерения по строкам. Каждому значению измерения по строкам могут соответствовать несколько значений измерения по столбцам. Для каждого из них строится свой столбец в каждой точке кросс-диаграммы. Количество столбцов в группе называется «серией».

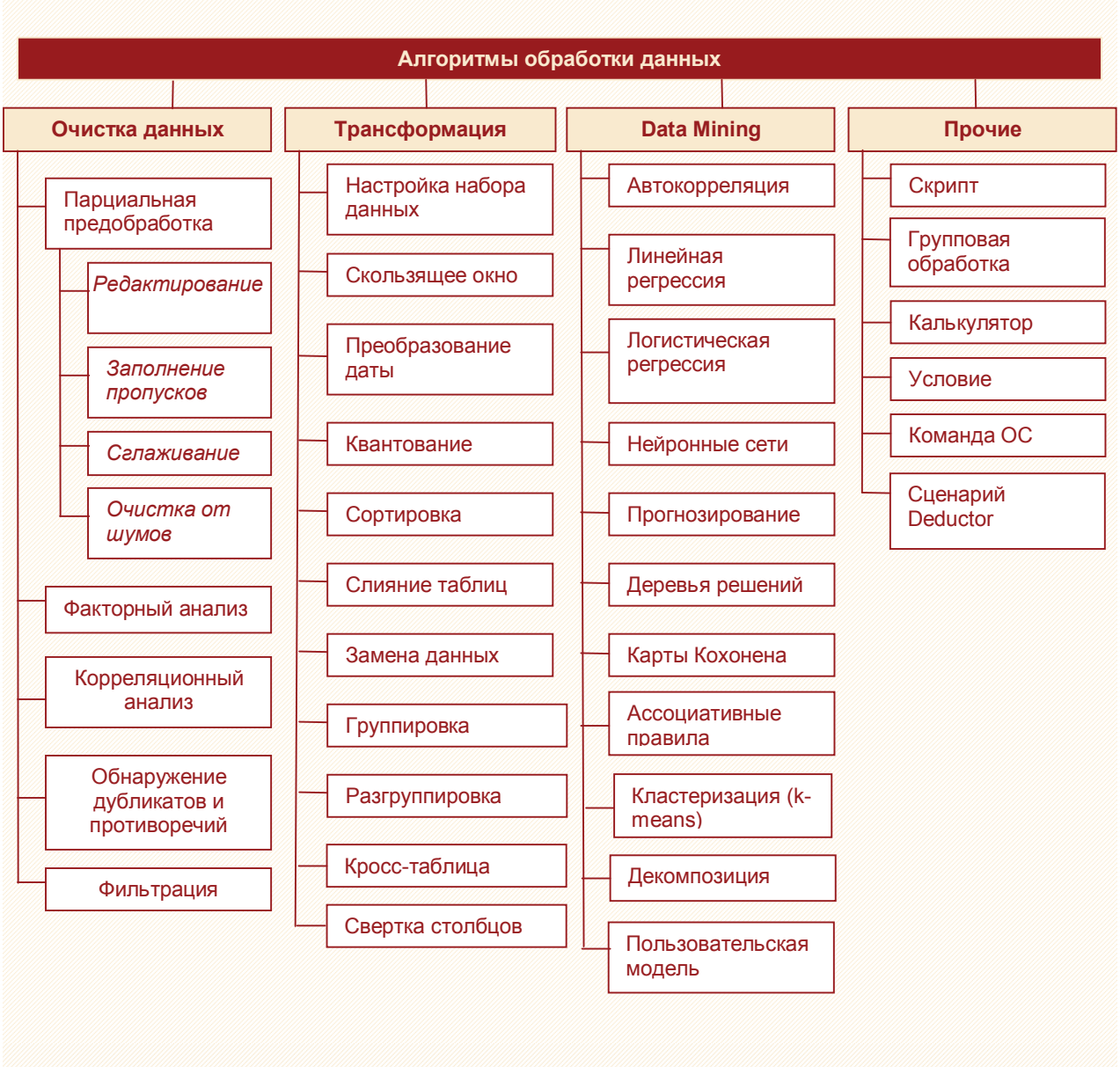
Таким образом, если фактическое количество серий и точек не превышает заданный поток, то в диаграмме отображается вся информация.

Описание аналитических алгоритмов

Работа по созданию законченного решения на базе Deductor не сводится только к консолидации и визуализации данных. Необходимо выполнение и других действий, в частности таких как:

- *Очистка данных*. На этом этапе проводится редактирование аномалий, заполнение пропусков, сглаживание, очистка от шумов, обнаружение дубликатов и противоречий.
- *Трансформация данных*. Производится замена пустых значений, квантование, табличная замена значений, преобразование к скользящему окну, изменение формата набора данных.
- *Data Mining*. Строятся модели с использованием нейронных сетей, деревьев решений, самоорганизующихся карт, ассоциативных правил и других методов.

На рисунке представлены алгоритмы, которые используются в программе, сгруппированные по назначению.



Очистка данных

Если анализируемые данные не соответствуют определенным критериям качества, то их предварительная обработка становится необходимым шагом для обеспечения удовлетворительного результата анализа. Необходимость в предварительной обработке возникает независимо от того, какие алгоритмы и технологии используются. Более того, эта задача может представлять самостоятельную ценность в областях, не имеющих непосредственное отношение к анализу данных. Очевидно, что исходные («сырые») данные чаще всего нуждаются в очистке.

Задач, решаемых на этапе очистки данных множество: аномалии, пропуски, шумы и прочие. Ниже описанные механизмы решения задач очистки данных, реализованные в Deductor.

Парциальная обработка

В процессе парциальной обработки восстанавливаются пропущенные данные, редактируются аномальные значения, проводится спектральная обработка. В Deductor Studio при этом используются алгоритмы, в которых каждое поле анализируемого набора обрабатывается независимо от остальных полей, то есть данные обрабатываются по частям. По этой причине такая предобработка получила название *парциальной*. В числе процедур предобработки данных, реализованных в Deductor Studio, входят сглаживание, удаление шумов, редактирование аномальных значений, заполнение пропусков в рядах данных.

Заполнение пропусков

Назначение

Часто бывает так, что в столбце некоторые данные отсутствуют в силу каких-либо причин (данные неизвестны, либо их забыли внести и т.п.). Раньше из-за этого пришлось бы убрать из обработки все строки, которые содержат пропущенные данные. Чтобы этого не происходило, в программе предусмотрено два способа заполнения пропущенных данных.

- Аппроксимация – пропущенные данные восстанавливаются методом аппроксимации.
- Максимальное правдоподобие – алгоритм подставляет наиболее вероятные значения вместо пропущенных данных.

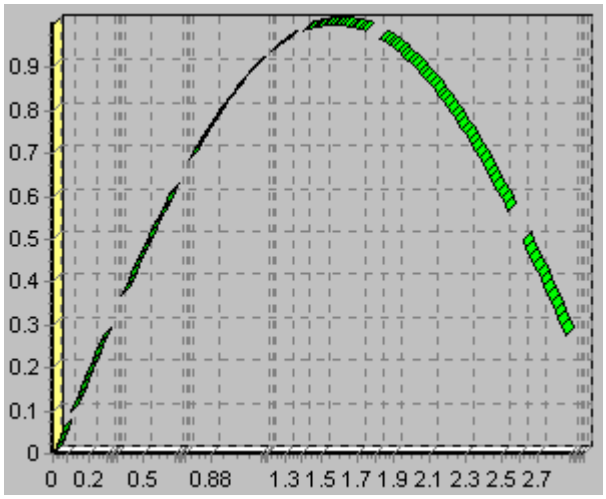
Принцип работы

Метод аппроксимации используется только для упорядоченных данных, чаще всего это временные ряды. Этот метод использует последовательный рекуррентный фильтр второго порядка (фильтр Калмана). Входные данные последовательно подаются на вход фильтра, и если очередное значение ряда отсутствует, оно заменяется значением, которое экстраполируется фильтром.

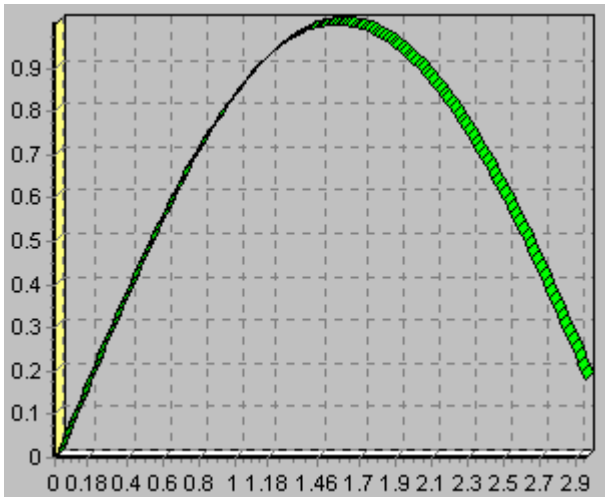
Метод максимального правдоподобия рекомендуется использовать на неупорядоченных данных. При использовании этого метода строится плотность распределения вероятностей, и отсутствующие данные заменяются значением, соответствующим ее максимуму.

Пример

На рисунке представлены упорядоченные данные с пропусками.



После применения алгоритма аппроксимации эти данные выглядят так:



Редактирование аномалий

Назначение

Аномалии – это отклонения от нормального (ожидаемого) поведения чего-либо. Это может быть, например, резкое отклонение величины от ее ожидаемого значения.

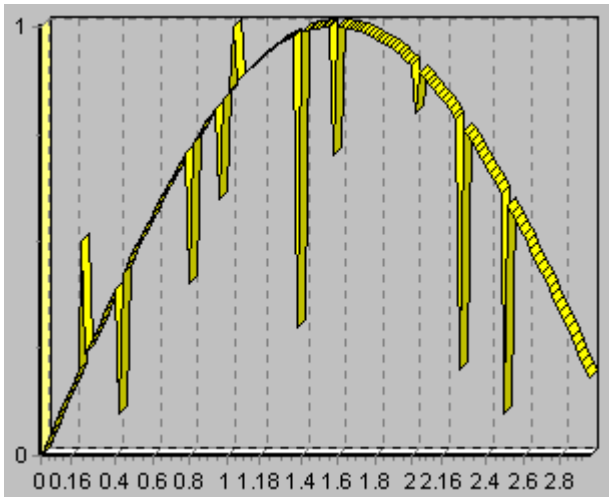
Автоматическое редактирование аномальных значений осуществляется с использованием методов робастной фильтрации, в основе которых лежит использование робастных статистических оценок, таких, например, как медиана. При этом можно задать эмпирически подобранный критерий того, что считать аномалией. Например, задание в качестве степени подавления аномальных данных значения «слабая» означает наиболее терпимое отношение к величине допустимых выбросов.

Настройки

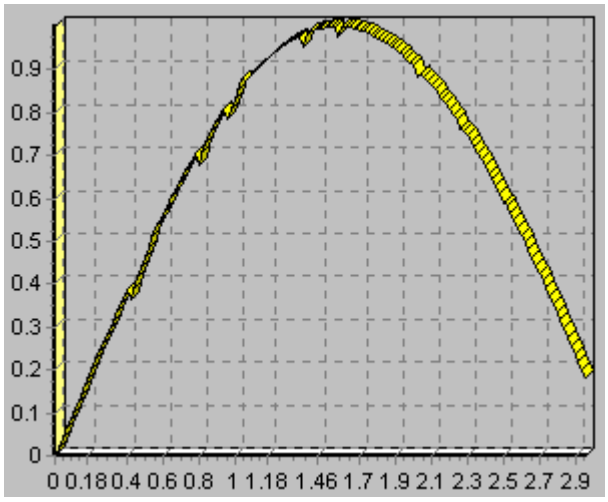
Для применения алгоритма удаления аномалий необходимо указать поле таблицы, к которому его нужно применить (которое содержит аномалии), и указать степень подавления аномальных данных – малую, среднюю или большую.

Пример

На рисунке приведен пример величины с аномалиями.



После применения алгоритма удаления аномалий та же величина представляется следующим образом:



Здесь была использована большая степень подавления аномалий.

Сглаживание

Для сглаживания рядов данных в программе используются два алгоритма.

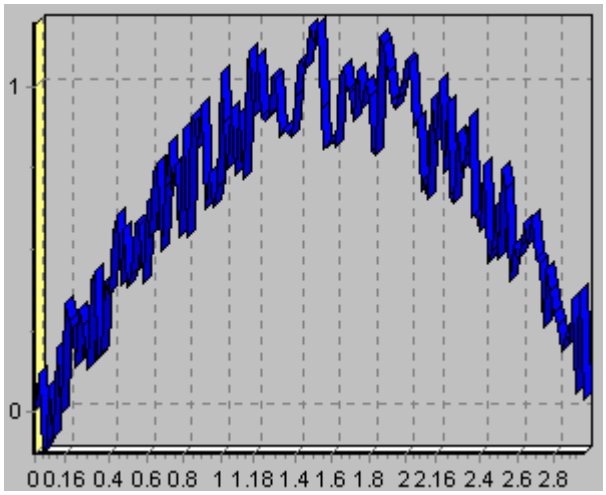
Первый способ сглаживания – это низкочастотная фильтрация с использованием быстрого преобразования Фурье. При этом задается верхнее значение полосы пропускаемых частот, т.е. отсекается все, что выше данного порога. Высокочастотная составляющая временного ряда соответствует резко изменяющимся данным, а низкочастотная – плавно изменяющимся.

При подавлении шумов на основе анализа распределения составляющих Фурье спектра на выход фильтра пропускаются спектральные составляющие, превышающие некоторый порог, рассчитанный по эмпирическим формулам в соответствии с заданным критерием степени вычитания шума. Чем больше требуется сгладить данные, тем меньше должно быть значение полосы. Однако слишком узкая полоса может привести к потере полезной информации. Следует заметить, что этот алгоритм наиболее эффективен, если анализируемые данные являются суммой полезного сигнала и белого шума.

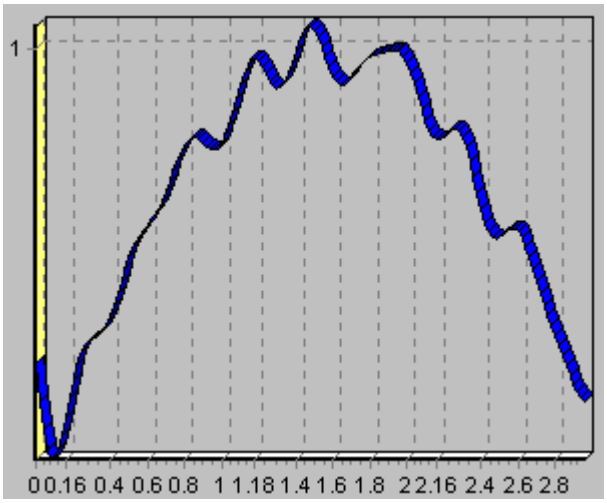
Второй способ сглаживания – это вейвлет-преобразование. Если выбран данный метод, то необходимо задать глубину разложения и порядок вейвлета. Глубина разложения определяет «масштаб» отсеиваемых деталей: чем больше эта величина, тем более «крупные» детали в исходных данных будут отброшены. При достаточно больших значениях параметра (порядка 7-9) выполняется не только очистка данных от шума, но и их сглаживание («обрезаются» резкие выбросы). Использование слишком больших значений глубины разложения может привести к потере полезной информации из-за слишком большой степени «огрубления» данных. Порядок вейвлета определяет гладкость восстановленного ряда данных: чем меньше значение параметра, тем ярче будут выражены «выбросы», и, наоборот, при больших значения параметра «выбросы» будут сглажены.

Пример

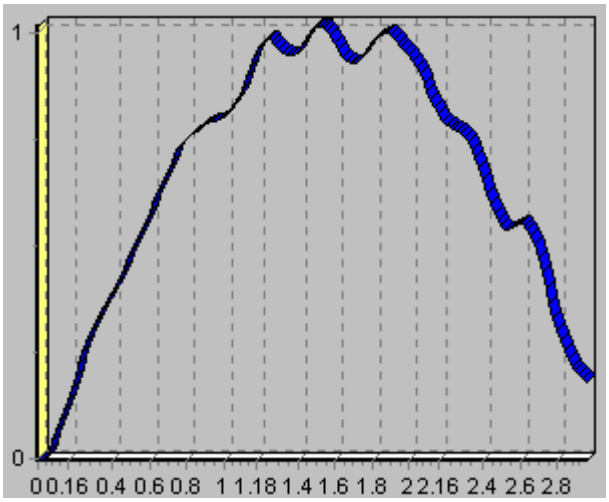
На рисунке показан пример данных с шумом.



На рисунке диаграмма после применения сглаживания с полосой пропускания равной 15.



На рисунке диаграмма после применения вейвлет-преобразования с глубиной разложения 3 и порядком 6.



Параметры алгоритма сглаживания задаются в «Парциальной обработке» на странице «Спектральная обработка».

Очистка от шумов

Назначение

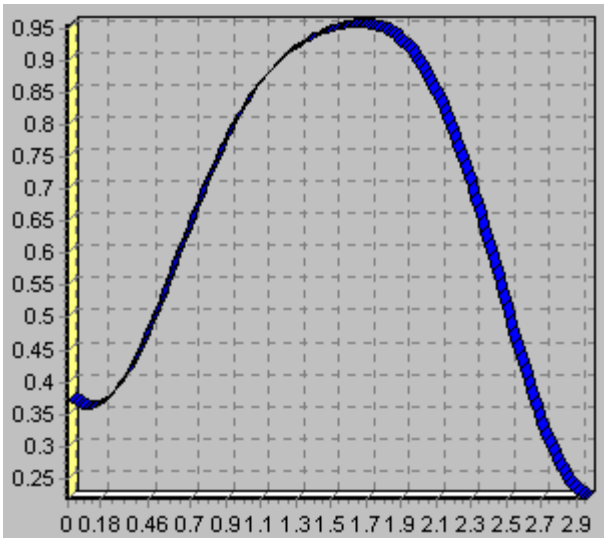
Шумы в данных не только скрывают общую тенденцию, но и проявляют себя при построении модели прогноза. Из-за них модель может получиться с плохими обобщающими качествами.

При выборе режима очистки от шумов необходимо задать степень вычитания шума: малую, среднюю или большую. При использовании вычитания шума следует соблюдать осторожность, т.к. реализованный здесь эвристический алгоритм гарантирует удовлетворительный результат лишь при выполнении двух условий:

- Дисперсия шума значительно меньше энергии полезного сигнала;
- Шум имеет нормальное распределение.

Пример удаления шумов

Применим алгоритм для данных с шумом из предыдущего примера. Результат на рисунке.



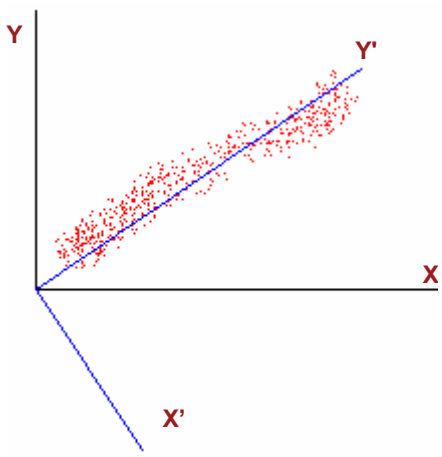
Параметры алгоритма очистки от шумов задаются в «Парциальной обработке» на странице «Спектральная обработка».

Факторный анализ

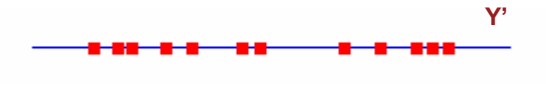
При исследовании сложных объектов и систем часто нельзя непосредственно измерить величины, определяющие свойства этих объектов (так называемые *факторы*), а иногда неизвестно даже число и содержательный смысл факторов. Для измерений могут быть доступны иные величины, h способом зависящие от этих факторов. При этом, когда влияние неизвестного фактора проявляется в нескольких измеряемых признаках, эти признаки могут обнаруживать тесную связь между собой (например, коррелированность). Поэтому общее число факторов может быть гораздо меньше числа измеряемых переменных, которое обычно выбирается исследователем в некоторой степени произвольно. Для обнаружения влияющих на измеряемые переменные факторов используются методы факторного анализа, реализованные в обработчике «Факторный анализ».

В обработчике используется метод главных компонент. Этот метод сводится к выбору новой ортогональной системы координат в пространстве наблюдений. В качестве первой главной компоненты избирают направление, вдоль которого массив данных имеет наибольший разброс. Выбор каждой главной последующей компоненты происходит так, чтобы разброс данных вдоль нее был максимальным и чтобы эта главная компонента была ортогональна другим главным компонентам, выбранным прежде. В результате получаем несколько главных компонент, каждая следующая из которых несет все меньше информации из исходного набора. Следующим шагом является выбор наиболее информативных главных компонент, которые будут использоваться в дальнейшем анализе.

Посмотрим на следующий рисунок. На нем изображено двумерное пространство наблюдений в осях X и Y, соответствующих двум измеряемым параметрам.



Как видно, разброс данных велик по обоим направлениям. Теперь повернем систему координат так, чтобы оси Y соответствовало направление наибольшего разброса массива данных, т.е. перейдем в систему координат X' – Y'. Теперь по оси X' дисперсия данных невелика, и появляется возможность отбросить это направление, перейдя к одномерному пространству.



В этом случае потери некоторой части информации могут компенсироваться удобством работы с данными меньшей размерности. Аналогичные действия выполняются в многомерном случае: система координат последовательно вращается таким образом, чтобы каждый следующий поворот минимизировал остаточный разброс массива данных.

Выбор главных компонент в процессе факторного анализа может осуществляться полуавтоматически: пользователь задает уровень значимости (вклад в результат), который в сумме должны давать главные компоненты. В результирующем наборе остаются главные компоненты, расположенные в порядке убывания значимости, суммарный вклад которых не менее заданного пользователем уровня.

Факторный анализ широко используется в следующей ситуации. В очень большом исходном наборе данных есть много полей, некоторые из которых взаимозависимы. На этом наборе данных требуется, к примеру, обучить нейронную сеть. Для того чтобы снизить время, требуемое на обучение сети, и требования к объему обучающей выборки, с помощью факторного анализа осуществляют переход в новое пространство факторов меньшей размерности. Так как большая часть информативности исходных данных сохраняется в выбранных главных компонентах, то качество модели ухудшается незначительно, зато на много сокращается время обучения сети.

Корреляционный анализ

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 модуль корреляции (степень взаимозависимости) между входным и выходным факторами меньше порога значимости, то соответствующий фактор отбрасывается как незначащий.

В процессе обработки значащие факторы могут выбираться вручную или автоматически. При ручном выборе около имени каждого входного поля устанавливается флажок, если это поле нужно включить в выходную выборку, и снимается в противном случае. В автоматическом

режиме исключаются все факторы, корреляция которых с выходными полями меньше порога задаваемого уровня значимости.

Замечание

На практике считается, что корреляция больше 0,6 означает очень высокую связь между рядами, меньше 0,3 – отсутствие зависимости, а промежуточные значения констатируют наличие определенной связи. В другом подходе полагается, что зависимость существует, если корреляцию больше 2 поделить на корень из объема выборки.

Пример

В качестве примера рассмотрим, как определить товары-заменители и сопутствующие товары, имея временные ряды объемов продаж. У товаров-заменителей должна быть большая отрицательная корреляция, т.к. увеличение продаж одного товара ведет к спаду продаж второго. А у сопутствующих товаров – большая положительная корреляция.

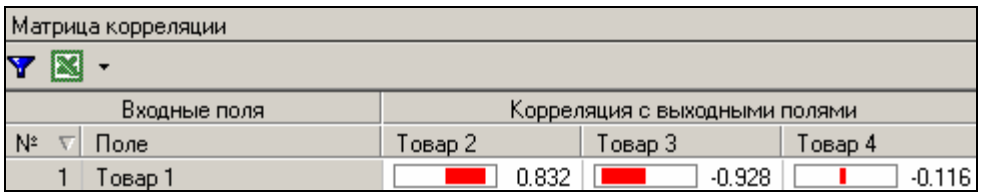
Пусть есть такие временные ряды продаж товаров:

Товар 1	Товар 2	Товар 3	Товар 4
10	20	15	25
12	22	12	26
14	25	9	26
13	24	10	25
14	25	9	24
14	25	9	23
12	21	12	24
10	18	14	23
16	24	9	22
13	21	9	23
17	25	7	25

Определим корреляцию *Товар 1* с остальными товарами.

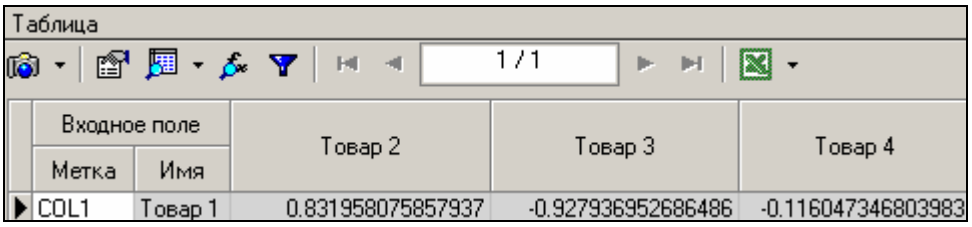
Входные поля	Корреляция с выходными полями		
	Товар 2	Товар 3	Товар 4
☑ Товар 1	0.832	-0.928	-0.116

Одним из доступных способов визуализации результатов является визуализатор «Матрица корреляции». В данном примере эта матрица имеет следующий вид:



Как видно из рисунка, ряд продаж для *Товар 2* имеет очень большую положительную, а *Товар 3* – отрицательную корреляцию. Из этого можно сделать вывод, что *Товар 2*, возможно, является сопутствующим товаром, а *Товар 3* – заместителем *Товар 1*. Корреляция с продажами *Товар 4* с *Товар 1* является отрицательной, но при этом абсолютное значение корреляции невелико, и, следовательно, можно говорить об отсутствии взаимосвязи между продажами *Товар 1* и продажами *Товар 4*.

После проведения корреляционного анализа становится доступным обработчик «Матрица корреляции» (так называемый *зависимый обработчик*, см. подраздел «Зависимые обработчики»). С его помощью результаты можно представить в виде таблицы, в которой представлены коэффициенты связи входного поля с выходными.



Обнаружение дубликатов и противоречий

Назначение

При построении модели регрессии или классификации в анализируемых таблицах нужно определить входные и выходные поля, зависимости между которыми и исследуются. Предполагается, что значения входных полей полностью определяют значения выходных. При подобной постановке задачи возможно возникновение противоречий, то есть присутствие групп записей, значения в ключевых (входных) полях которых полностью совпадают, а в целевых (выходных) – различаются. Например, если значения в ключевых полях – это коды товаров, а в целевых – цены этих товаров, то присутствие двух записей с одинаковым кодом, но с разной ценой как раз и создает противоречие. Обычно бывает так, что только одна запись из группы противоречивых является правильной, а остальные – ошибочными. Очевидно, что присутствие ошибочных данных искажает результаты анализа, поэтому противоречивые данные чаще всего лучше вообще исключить из исходной выборки. Однако следует заметить, что искусственное введение противоречий в исходные данные может быть полезным, например, если нужно ввести некоторую неопределенность в данные, кроме того противоречия могут отражать особенности поведения анализируемого объекта.

Также в данных могут встречаться записи с одинаковыми входными факторами и одинаковыми выходными, т.е. дубликаты. Эти данные чаще всего избыточны, хотя присутствие дубликатов в анализируемых данных можно рассматривать как способ повышения «значимости» дублирующейся информации. В некоторых случаях такой прием может быть полезен, например, если при обучении нейросети нужно особо выделить и усилить влияние некоторых наборов значений. Однако в других случаях дублирование может указывать на ошибки при подготовке исходных данных. Дубликаты могут искажать результаты некоторых методов анализа, например, статистического.

Так или иначе, в процессе анализа иногда возникает проблема выявления дубликатов и противоречий в данных. В Deductor Studio для автоматизации этого процесса есть соответствующий инструмент – обработчик «Дубликаты и противоречия».

- *Дубликаты* – записи в таблице, все входные и выходные поля которых одинаковые.
- *Противоречия* – записи в таблице, у которых все входные поля одинаковые, но отличаются хотя бы по одному выходному полю.

Суть обработки состоит в том, что определяются входные и выходные поля. Алгоритм ищет во всем наборе записи, для которых одинаковым входным полям соответствуют одинаковые (дубликаты) или разные (противоречия) выходные поля. На основании этой информации создаются два дополнительных логических поля – «Дубликат» и «Противоречие», принимающие значения «истина» или «ложь», и дополнительные числовые поля «Группа дубликатов» и «Группа противоречий», в которые записываются номер группы дубликатов и группы противоречий, содержащих данную запись. Если запись не является дубликатом или противоречием, то соответствующие поля будут пустыми.

Настройка выявления дубликатов и противоречий заключается в выборе назначений полей исходной выборки данных, то есть в выборе, какие поля входные, а какие – выходные.

Пример

Рассмотрим таблицу. Исходная таблица – это таблица с полями *Поле 1* … *Поле 4*. Поля 1 и 2 – входные, поля 3 и 4 – выходные.

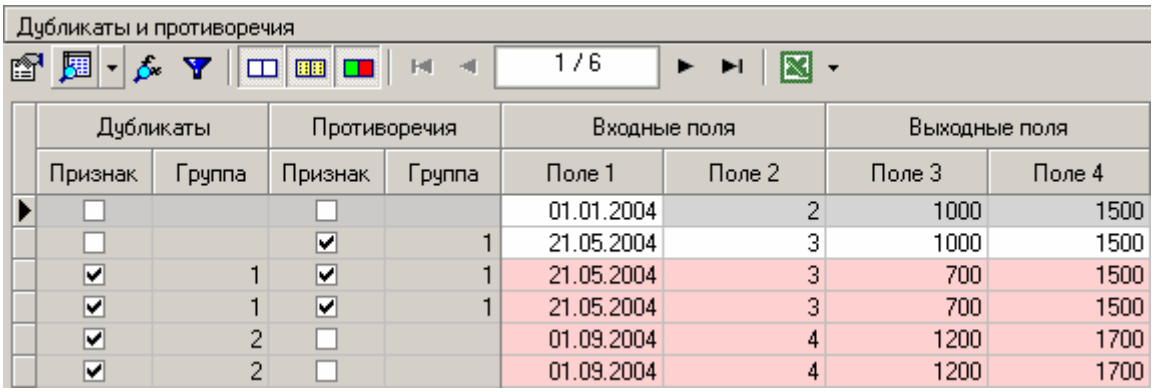
Поле 1	Поле 2	Поле 3	Поле 4
01.01.2004	2	1000	1500
21.05.2004	3	1000	1500
21.05.2004	3	700	1500
21.05.2004	3	700	1500
01.09.2004	4	1200	1700
01.09.2004	4	1200	1700

Отображение полученного результата возможно в виде таблицы следующего вида.

Входные поля		Выходные поля		Противоречие	Дубликат	Группа противоречий	Группа дубликатов
Поле 1	Поле 2	Поле 3	Поле 4				
01.01.04	2	1000	1500	Нет	Нет		
21.05.04	3	1000	1500	Да	Нет	1	
21.05.04	3	700	1500	Да	Да	1	1
21.05.04	3	700	1500	Да	Да	1	1
01.09.04	4	1200	1700	Нет	Да		2
01.09.04	4	1200	1700	Нет	Да		2

Как видно из таблицы, после применения алгоритма добавлены еще две пары полей: *Противоречие* и *Дубликат*; *Группа противоречий* и *Группа дубликатов*. Вторая строка противоречит третьей. Третья строка противоречит второй и является дубликатом четвертой. Четвертая строка противоречит второй и является дубликатом третьей. Пятая и шестая строки – дубликаты.

При использовании обработчика «Дубликаты и противоречия» возможно отображение результатов обработки с помощью одноименного визуализатора «Дубликаты и противоречия», который отображает в виде таблицы информацию о дубликатах и противоречиях.



Дубликаты		Противоречия		Входные поля		Выходные поля	
Признак	Группа	Признак	Группа	Поле 1	Поле 2	Поле 3	Поле 4
☐		☐		01.01.2004	2	1000	1500
☐		☑	1	21.05.2004	3	1000	1500
☑	1	☑	1	21.05.2004	3	700	1500
☑	1	☑	1	21.05.2004	3	700	1500
☑	2	☐		01.09.2004	4	1200	1700
☑	2	☐		01.09.2004	4	1200	1700

Опции:

- – включение данной опции позволяет отображать в таблице строки, *не являющиеся дубликатами или противоречиями*;
- – включение данной опции позволяет отображать в таблице строки, содержащие *дубликаты*;
- – включение данной опции позволяет отображать в таблице строки, содержащие *противоречия*.

Обработка дубликатов или противоречий не проводится в тех случаях, когда дубликаты или противоречия были преднамеренно введены в исходные данные. Как правило, этот метод применяется только к одной из описываемых аномалий, то есть либо только дубликаты, либо только противоречия остаются без обработки. Кроме того, дубликаты или противоречия могут быть вполне естественными для анализируемого процесса, но чаще всего специальная обработка подобных данных требуется.

Наличие дубликатов и противоречий может приводить к полному обесцениванию строк, содержащих подобные отклонения. Считается, что присутствие подобных ошибок делает информацию недостоверной. Такая ситуация возникает, например, при обработке социологических данных, когда наличие дубликатов или противоречий свидетельствует о недобросовестности респондента и вызывает недоверие ко всей предоставленной им информации. В этом случае все записи, формирующие группу дубликатов или противоречий, должны быть удалены. Это первый способ обработки.

Существует еще один, наиболее естественный, способ обработки дубликатов. Поскольку все дубликаты представляют собой копии одних и тех же данных, они могут быть сведены к одной записи набора данных, содержащей уникальную копию таких значений. К противоречиям также применим подобный метод обработки, но с некоторыми ограничениями. Напомним, что противоречивые записи содержат одинаковые входные значения, но различные выходные. Приведение таких записей к одной, уникальной, возможно на основе статистической агрегации, то есть вычисления максимума, минимума или среднего из выходных значений и подстановки этой величины в соответствующее поле формируемой уникальной записи. Следует заметить, что такую операцию следует выполнять с осторожностью; семантика, то есть смысл данных, должна допускать возможность вычисления таких статистических значений. Например, статистическая

агрегация допустима для цены товара или величины пропускной способности, но бессмысленна для номеров квартир или кодов налогоплательщиков.

Фильтрация

С помощью операции фильтрации можно оставить в таблице только те записи, которые удовлетворяют заданным условиям, а остальные исключить из набора данных.

Параметры фильтрации задаются в виде списка условий, который содержит следующие столбцы:

- *Операция* – позволяет установить функцию отношения «И» или «ИЛИ» между полями, для каждого из которых выполняется фильтрация. Возможна фильтрация по нескольким условиям для нескольких полей одновременно. Практически в результате фильтрации по каждому из полей или условий будет получено отдельное множество значений. Тогда функция в поле «Операция» устанавливает отношение между этими множествами. Если используется отношение «И», то в результирующий набор будут включены записи, удовлетворяющие условиям фильтрации по обоим полям, если используется отношение «ИЛИ», то в выходной набор будут включены данные, удовлетворяющие хотя бы одному из условий. Установка отношений возможна, только если настроены два или более условия фильтрации. Для выбора операции следует дважды щелкнуть левой кнопкой мыши в столбце «Операция» для соответствующего условия и из списка, открываемого кнопкой, выбрать нужную функцию отношения. По умолчанию устанавливается отношение «И».
- *Имя поля* – позволяет выбрать поле, по значениям которого должна быть выполнена фильтрация. Для этого дважды щелкнуть в столбце «Имя поля» и с помощью кнопки открыть список полей текущей выборки, где щелкнуть по нужному полю. Одно и то же поле может быть использовано в нескольких условиях.
- *Условие* – указывается условие, по которому нужно выполнить фильтрацию для данного поля. Для выбора условия достаточно дважды щелкнуть мышью в соответствующей ячейке и в списке условий, открываемом кнопкой, выделить нужное условие. Доступны следующие условия фильтрации:
    - *= (равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), <> (не равно)* – отбираются только те записи, значения которых в данном поле удовлетворяют заданному выражению;
    - *пустой* – отбираются только те записи, для которых в данном поле содержится пустое значение. В этом случае поле «Значение» не используется;
    - *не пустой* – отбираются только те записи, для которых в данном поле не содержится пустое значение. В этом случае поле «Значение» не используется;
    - *содержит* – отображаются только те записи, которые в данном столбце содержат указанное значение;
    - *не содержит* – отображаются только те записи, которые в данном столбце не содержат указанное значение;
    - *в интервале, вне интервала* – для числовых полей и полей типа «Дата/время» отбираются только те записи, значения которых в данном столбце лежат в выбранном диапазоне (вне выбранного диапазона);
    - *в списке, вне списка* – отбираются только те записи, которые содержатся в выбранном списке (вне выбранного списка);
    - *начинается на, не начинается на* – для строковых полей отбираются записи, значения которых в данном столбце начинаются (не начинаются) на введенную последовательность символов.
    - *заканчивается на, не заканчивается на* – для строковых полей отбираются записи, значения которых в данном столбце заканчиваются (не заканчиваются) на введенную последовательность символов.

- *первый, не первый* – для полей типа «Дата/время» - по данному полю отбираются первые (не первые) N периодов от выбранной даты. Периодом может быть день, неделя, месяц, квартал, год. Например, если выбрать условие «первые 3 дня от 29.11.2004», то будут отобраны записи, в которых значение данного поля равно «29.11.2004», «30.11.2004», «01.12.2004» – 3 последующих дня.
- *последний, не последний* – для полей типа «Дата/время» отбираются последние (не последние) N периодов от выбранной даты. Периодом может быть день, неделя, месяц, квартал, год. Например, если выбрать условие «последние 3 дня от 29.11.2004», то будут отобраны записи, в которых значение данного поля равно «29.11.2004», «28.11.2004», «27.11.2004» – 3 предыдущих дня.

- *Значение* – указывается значение(я), по которому будет производиться фильтрация записей в соответствии с заданным условием. Способ ввода значения будет различным в зависимости от типа данных и выбранного условия. Допустим, в качестве условия выбрана операция отношения «=», «<>», «>» и т.д. Если данные в поле являются непрерывными (т.е. числовыми), то достаточно дважды щелкнуть мышью в соответствующей ячейке, чтобы появился курсор, затем ввести значение (число). Если поле, по которому выполняется фильтрация, имеет тип «строка» (т.е. является дискретным), то в результате двойного щелчка в столбце «Значение» появится кнопка выбора, которая откроет окно «Список уникальных значений», где будут отображены все уникальные значения поля и их количество. Чтобы выбрать значение для условия отбора, достаточно выделить его и щелкнуть **Ok**, либо просто дважды щелкнуть мышкой на нужном значении. Если выбрано условие «между» или «не между», тогда после щелчка мышки откроется окно, в котором необходимо указать верхнюю и нижнюю границы интервала, и так далее…

Фильтрация может быть полезна для применения различных алгоритмов к группам данных, так как позволяет выделить из выборки только нужную часть. Тем не менее, если требуется провести анализ только известной части данных, желательно загружать в программу уже отфильтрованный набор. Такая возможность имеется, например, при загрузке данных из хранилища. В этом случае значительно экономится память, занимаемая данными, и увеличивается скорость обработки.

Например, имеется несколько групп товаров, и нужно провести определенный анализ для каждой группы отдельно. Тогда можно воспользоваться фильтрацией при импорте из хранилища данных, оставив в наборе данные только по одной группе, провести анализ. Затем к исходному набору снова применить фильтрацию, оставив другую группу товара, и провести анализ для нее. И так для каждой товарной группы.

Трансформация данных

Анализируемая информация, представленная в виде набора данных, имеет определенный формат. Под форматом данных подразумевается отнесение их к определенному типу (целочисленные, строковые, даты), задание вида (дискретные или непрерывные) и т. п. Для анализа различных аспектов информации может потребоваться преобразование ее формата или трансформация. Кроме преобразования форматов трансформация включает в себя изменение представления данных и другие операции, связанные с преобразованиями входного набора данных.

Настройка набора данных

Обработчик «Настройка набора данных» предназначена для изменения имени, метки, типа, вида и назначения полей текущей выборки данных и кэширования выходного набора.

У каждого поля можно изменить метку столбца, которая будет использоваться для дальнейшей работы в программе. Если в текущей выборке данных поле имеет имя *Name*, ему можно задать метку *Наименование*, что гораздо удобнее при дальнейшем отображении этого поля в таблицах или диаграммах.

Изменение имени поля удобно в тех случаях, когда имена столбцов могут измениться в источнике данных или при перенастройке узлов верхних уровней. В этом случае в узле

«Настройка набора данных» имя исходного столбца заменяется другим, на которое и настраиваются все дочерние узлы. После такой операции изменение имен полей на верхних уровнях не потребует перенастройки всех дочерних узлов в дереве сценариев.

Далее каждому полю можно изменить тип:

- 0/1 Логический – данные в поле могут принимать только два значения - 0 или 1 (ложь или истина);
- Дата/время – поле содержит данные типа дата/время;
- 9.0 Вещественный – значения поля – числа с плавающей точкой;
- 12 Целый – данные в поле представляют собой целые числа;
- ab Строковый – данные в столбце представляют собой строки символов.

Затем можно указать вид данных:

- Непрерывный – значения в столбце могут принимать любое значение в рамках своего типа. Обычно непрерывными являются числовые данные;
- ''' Дискретный – данные в столбце могут принимать ограниченное число значений. Обычно дискретный характер носят строковые данные.

В зависимости от содержимого поля «Тип данных» на выбор вида данных накладываются ограничения, например, строковые данные не могут быть непрерывными. К выбору типа и вида данных нужно относится серьезно, так как это влияет на возможность дальнейшего использования этого поля.

Далее можно изменить назначение полей. В зависимости от дальнейшего использования выборки данных предлагается изменить текущие назначения полей на следующие:

- *Непригодное* – данные в поле не пригодны для данного способа обработки (программа автоматически указывает полю это назначение). Например, для преобразования даты поле должно иметь тип «Дата/время». Если оно будет иметь, например, строковый тип, то программа автоматически укажет для него назначение «Непригодное».
- *Неиспользуемое* – запрещает использование поля в обработке данных и исключает его из выходного набора. В отличие от непригодного поля такие поля в принципе могут использоваться, если будет в этом необходимость.
- *Ключ* – поле будет использоваться в качестве первичного ключа.
- *Входное* – поле таблицы, построенное на основе столбца, будет являться входным полем обработчика (нейронной сети, дерева решений и т.д.).
- *Выходное* – поле таблицы, построенное на основе столбца, будет являться выходным полем обработчика (например, целевым полем для обучения нейронной сети).
- *Информационное* – поле содержит вспомогательную информацию, которую часто полезно отображать, но не следует использовать при обработке.
- *Измерение* – поле будет использоваться в качестве измерения в многомерной модели данных.
- *Факт* – значения поля будут использованы в качестве фактов в многомерной модели данных.
- *Атрибут* – поле содержит описание свойств или параметров некоторого объекта.
- ID *Транзакция* – поле, содержащее идентификатор событий, происходящих совместно (одновременно). Например, номер чека, по которому приобретены товары. Тогда покупка товара – это событие, а их совместное приобретение по одному чеку – транзакция.

-  *Элемент* – поле, содержащее элемент транзакции (событие).

Для установки первоначальных параметров полей необходимо выделить поле или список полей и нажать на кнопку **Сброс параметров**.

Одно из важных применений обработчика «Настройка полей» состоит в кэшировании данных. Кэширование – это загрузка часто используемой информации в оперативную память для быстрого доступа к ней, минуя многократные считывания с жесткого диска. Кэширование может заметно повысить скорость работы сценария в следующих случаях:

- Перед настройкой полей находится узел, в котором проводится большое количество сложных и длительных вычислений. В узле эти вычисления производятся каждый раз по мере обращения к записям, поэтому обработка подобных данных может занимать много времени. В этом случае установка кэша позволяет однократно вычислить все выражения и сохранить результаты в памяти, в последствии используя уже рассчитанные значения;
- Из узла настройки полей выходит несколько ветвей сценария обработки. Deductor Studio спроектирован таким образом, что старается минимизировать расход оперативной памяти, поэтому если какое-то значение может быть рассчитано, то оно не сохраняется, а всегда рассчитывается «на лету». Если в каком-нибудь узле будут требоваться данные, то программа по цепочке сценариев дойдет до источника данных и считает информацию оттуда. В случае, если из одного узла отходят несколько ветвей сценария обработки, оптимальным по скорости будет вариант, при котором кэшируются данные, и последующие узлы сценария обработки используют информацию из оперативной памяти без необходимости доступа и извлечения их из источника данных. В этом случае использование кэша позволяет однократно загружаются сведения в память и в дальнейшем обращаться только к ОЗУ.

Если объем кэшируемых данных превышает доступные ресурсы оперативной памяти, то значительного прироста в быстродействии может не наблюдаться, т.к. частично они все равно окажутся выгруженными на диск.

Включить кэширование данных в узле «Настройка полей» можно с помощью установки флага **Кэшировать данные столбца**. Кэширование производится по каждому полю в отдельности.

Скользящее окно

При решении некоторых задач, например, при прогнозировании временных рядов при помощи нейросети, требуется подавать на вход модели значения нескольких смежных отсчетов из исходного набора данных. Такой метод отбора данных называется скользящим окном (окно – поскольку выделяется только некоторый непрерывный участок данных, скользящее – поскольку это окно «перемещается» по всему набору). При этом эффективность реализации заметно повышается, если не выбирать данные каждый раз из нескольких последовательных записей, а последовательно расположить данные, относящиеся к конкретной позиции окна, в одной записи.

Значения в одном из полей записи будут относиться к текущему отсчету, а в других – смещены от текущего отсчета «в будущее» или «в прошлое». Таким образом, преобразование скользящего окна имеет два параметра: «глубина погружения» - количество «прошлых» отсчетов, попадающих в окно, и «горизонт прогнозирования» – количество «будущих» отсчетов. Следует отметить, что для граничных (относительно начала и конца всей выборки) положений окна будут формироваться неполные записи, т.е. записи, содержащие пустые значения для отсутствующих прошлых или будущих отсчетов. Алгоритм преобразования позволяет исключить такие записи из выборки (тогда для нескольких граничных отсчетов записи формироваться не будут) либо включить их (тогда формируются записи для всех имеющихся отсчетов, но некоторые из них будут неполными). Отметим, что для правильного формирования скользящего окна данные должны быть соответствующим образом упорядочены.

Пример

Есть история продаж за половину года по месяцам, представленная таблицей:

Первый день месяца	Объем продаж (тыс. руб.)
01.01.2004	1000
01.02.2004	1160
01.03.2004	1210
01.04.2004	1130
01.05.2004	1250
01.06.2004	1300

Если задать глубину погружения 2 и горизонт прогнозирования 1, то получим следующую таблицу с неполными записями.

Первый день месяца	Объем продаж 2 месяца назад	Объем продаж месяц назад	Объем продаж в текущий месяц	Объем продаж на следующий месяц
				1000
01.01.2004			1000	1160
01.02.2004		1000	1160	1210
01.03.2004	1000	1160	1210	1130
01.04.2004	1160	1210	1130	1250
01.05.2004	1210	1130	1250	1300
01.06.2004	1130	1250	1300	
	1250	1300		
	1300			

Или следующую таблицу с полными записями.

Первый день месяца	Объем продаж 2 месяца назад	Объем продаж 1 месяц назад	Объем продаж в текущий месяц	Объем продаж на следующий месяц
01.03.2004	1000	1160	1210	1130
01.04.2004	1160	1210	1130	1250
01.05.2004	1210	1130	1250	1300

Такую таблицу можно использовать при построении моделей, например, для прогнозирования. При этом на вход модели для ее обучения будут подаваться поля с текущим и двумя предыдущими месяцами, а на выход – поле с объемом продаж на следующий месяц.

Эту таблицу также можно использовать для вычисления оборотов за определенное количество месяцев, например, вычисляя разницу между столбцами с объемом продаж за текущий месяц и объемом продаж за предыдущий месяц.

Преобразование даты

Преобразование даты служит для анализа всевозможных показателей за определенный период (год, квартал, месяц, неделя, день, час, минута, секунда). Суть преобразования заключается в том, что на основе столбца с информацией о дате/времени формируются один или несколько столбцов, в которых указывается, к какому заданному интервалу времени принадлежит строка данных. Тип интервала задается аналитиком, исходя из того, что он хочет выделить из даты.

Такая операция требуется потому, что очень часто интересным для анализа является не сама дата, а ее производная. Например, для анализа посещаемости магазина интересен день недели, а для оценки загруженности касс – час.

Значения нового столбца, полученного после применения преобразования даты, могут быть одного из трех типов: строка, число или дата. Например, нужно из даты «10.04.2004» получить только месяц. Тогда в столбце строкового типа будет содержаться «2004-M04», и его уже нельзя использовать как дату, например, к нему нельзя снова применить преобразование даты. А в столбце типа «дата» будет значение «01.04.2004» – первый день месяца. К нему снова можно применить преобразование и получить, например, номер квартала. Новый столбец будет содержать значение 2 числового типа.

Пример

Пример использования преобразования даты приведен в таблице. Первый столбец *Дата* – это исходный столбец. Остальные получены после обработки.

Дата	Год + Квартал	Год + Месяц	Год + Неделя	Квартал	Месяц	Неделя	День года	День недели	День недели
01.01.2004	01.01.2004	01.01.2004	01.01.2004	1	1	1	1	4	4 Четверг
09.01.2004	01.01.2004	01.01.2004	05.01.2004	1	1	2	9	5	5 Пятница
17.01.2004	01.01.2004	01.01.2004	12.01.2004	1	1	3	17	6	6 Суббота
25.01.2004	01.01.2004	01.01.2004	19.01.2004	1	1	4	25	7	7 Воскресенье
02.02.2004	01.01.2004	01.02.2004	02.02.2004	1	2	6	33	1	1 Понедельник
10.02.2004	01.01.2004	01.02.2004	09.02.2004	1	2	7	41	2	2 Вторник
18.02.2004	01.01.2004	01.02.2004	16.02.2004	1	2	8	49	3	3 Среда

Есть таблица с информацией о продажах. Пусть необходимо посмотреть объемы продаж по клиентам с разбивкой по месяцам. Для этого заменим дни продаж месяцем, в который попадает этот день. Объемы продаж удобно посмотреть с помощью OLAP-куба.

	Дата (Год+Месяц)			
Клиент	2004-M1	2004-M2	2004-M3	Итого:
Покупатель 3	408.00	912.00		1 320.00
Покупатель 5	444.00	675.00	882.00	2 001.00
Покупатель 6	345.00	966.00	396.00	1 707.00
Покупатель 7	930.00	261.00	243.00	1 434.00
Покупатель 8	984.00	66.00		1 050.00
Покупатель 10	720.00	990.00	843.00	2 553.00
Покупатель 11	12.00	342.00	933.00	1 287.00
Покупатель 13	504.00	633.00	9.00	1 146.00
Покупатель 14	228.00	624.00		852.00
Покупатель 15	150.00	972.00		1 122.00
Итого:	4 725.00	6 441.00	3 306.00	14 472.00

Квантование значений

При выполнении этой операции осуществляется разбиение диапазона числовых значений на указанное количество интервалов определенным методом и замена каждого обрабатываемого значения на число, связанное с интервалом, к которому оно относится, либо метку интервала. Интервалы разбиения включают в себя нижнюю границу, но не включают верхнюю кроме последнего интервала, который включает в себя обе границы. Результатом преобразования может быть: номер интервала, значение нижней или верхней границы интервала разбиения, среднее значение интервала разбиения, метка интервала или автоматическая метка.

Квантование (или дискредитация) может быть осуществлено интервальным или квантильным алгоритмом. Интервальное квантование подразумевает разбиение диапазона значений на указанное количество значений равной длины. Например, если значения в поле попадают в диапазон от 0 до 10, то при интервальном квантовании на 10 интервалов мы получим отрезки от 0 до 1, от 1 до 2 и т. д. При этом 0 будет относиться к первому интервалу, 1 – ко второму, а 9 и 10 – к десятому. Квантильное квантование подразумевает разбиение диапазона значений на равновероятные интервалы, то есть на интервалы, содержащие равное (или, по крайней мере, примерно равное) количество значений. Нарушение равенства возможно только тогда, когда значения, попадающие на границу интервала, встречаются в наборе данных несколько раз. В этом случае все они относятся к одному определенному интервалу и могут вызвать «перевес» в его сторону.

Настройки

Для настройки квантования требуется для каждого используемого при разбиении поля указать:

1. Способ разбиения – *по интервалам* или *по квантилям*.
2. Количество интервалов.
3. Значение, подставляемое вместо значения интервала – *номер интервала*, *нижняя граница*, *верхняя граница*, *середина интервала*, *метка интервала* или *автоматическая метка* интервала. Если выбрана метка интервала, то нужно еще задать для каждого интервала метку, то есть его наименование.
4. Вид данных – *дискретный* или *непрерывный*.

После окончания автоматического расчета границ интервалов на основе имеющихся данных можно вручную изменить вычисленные границы. При этом нижняя граница любого интервала не может быть больше верхней, хотя совпадать они могут. Ручное изменение границ может потребоваться в тех случаях, когда исходная выборка данных не отражает всего диапазона значений, которые может принимать исследуемая величина на практике.

Пример

Допустим, у нас есть таблица с информацией о кредиторах и с суммой взятых кредитов. Нужно узнать активность разных возрастных групп кредиторов.

N п/п	Возраст	Сумма
1	37	7000
2	38	7500
3	60	14500
4	28	15000
5	59	32000
6	25	11500
7	57	5000
8	45	61500
…	…	…

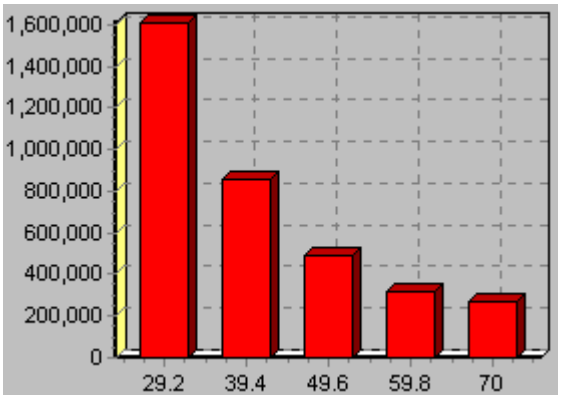
Статистика показывает, что возраст кредиторов лежит в диапазоне от 19 до 70 лет. Разобьем возраст на 5 равных интервалов, заменив возраст номером интервала.

Номер интервала	Нижняя граница	Верхняя граница
1	19	29,2
2	29,2	39,4
3	39,4	49,6
4	49,6	59,8
5	59,8	70

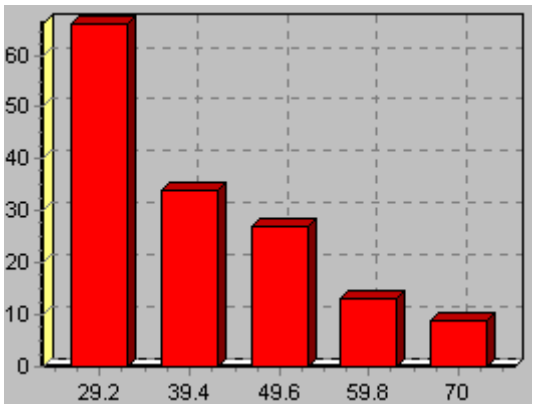
Получим таблицу.

N п/п	Номер интервала	Сумма
1	2	7000
2	2	7500
3	5	14500
4	1	15000
5	4	32000
6	1	11500
7	4	5000
8	3	61500
…	…	…

Теперь можно посмотреть количество кредиторов в каждой возрастной группе и сумму взятых кредитов по этим группам.



Сумма кредитов



Количество кредиторов

По таким данным можно делать выводы о необходимости, например, стимулирования малоактивных возрастных групп либо изменении рекламной политики с учетом наиболее активной возрастной категории.

Сортировка

С помощью сортировки можно изменять порядок следования записей в исходной выборке данных в соответствии с заданным пользователем алгоритмом сортировки. Результатом выполнения сортировки будет новая выборка данных, записи в которой будут следовать в соответствии с заданными параметрами сортировки.

Если сортировка производится по одному полю, то все записи исходной выборки располагаются в порядке возрастания или убывания его значений. Если сортировка производится по двум или более полям, то действует следующий алгоритм:

1. Сначала записи сортируются в заданном порядке для первого поля.
2. В каждом наборе одинаковых значений первого поля записи располагаются в заданном порядке для второго поля.

И так далее для всех полей, подлежащих сортировке.

В окне настройки параметров сортировки представлен список условий сортировки, в котором содержатся две графы:

- *Имя поля* – содержит имена полей, по которым следует выполнить сортировку.
- *Порядок сортировки* – содержит порядок сортировки данных в соответствующем поле – по возрастанию или по убыванию.

Слияние

Обработчик «Слияние» предназначен для соединения двух наборов данных по ключевым полям. Для этого необходимо задать общие поля двух таблиц. Предполагается, что в присоединяемом наборе данных есть поля, которые соответствуют полям в исходной таблице, это и есть ключевые поля или поля связи. Кроме того, в таблицах могут быть поля, которые имеются только во входящем или присоединяемом наборе данных. Такие поля можно добавить к результирующей выборке, образующейся после слияния.

Для слияния двух узлов необходимо выполнить следующие шаги:

- В мастере обработки определить узел связи, с которым будет осуществляться соединение, и определить тип слияния данных. При слиянии двух узлов возможны следующие варианты:
  -  Объединение. Объединение включает в результирующий набор данных все строки из входящего набора данных, дополненные снизу строками из связываемого набора данных;
  -  Внутреннее соединение. Внутреннее соединение включает в результат все строки, для которых найдено совпадение ключевых полей входящего и связываемого набора данных;
  -  Внешнее левое соединение. Внешнее левое соединение включает в результат все строки из входящего набора данных, дополненные значениями столбцов из связываемого набора данных, которые совпадают по ключевым полям.
  -  Внешнее правое соединение. Внешнее правое соединение включает в результат все строки из связываемого набора данных, дополненные значениями столбцов из входящего набора данных, которые совпадают по ключевым полям.
  -  Полное внешнее соединение. Полное внешнее соединение включает в результат все строки из входящего и связываемого наборов данных. Если ключевые поля совпадают, то значения столбцов заполняются реальными значениями. В несовпадающих строках столбцы заполняются пустыми значениями (null - значениями).
- На следующем шаге в мастере обработки необходимо указать связь между наборами данных, каким полям из входящего набора соответствуют поля в связанной таблице.
- Следующим шагом в мастере обработки необходимо указать поля, которые должны быть включены в результирующий выходной набор данных. Для этого щелчком левой кнопки мыши нужно установит галочку напротив метки поля, которое необходимо включить в выходной набор данных. На этой же странице мастера можно задать имена полей в результирующей таблице.
- На последнем этапе в мастере обработки существует возможность описания узла «Слияние», где можно указать детализированную информацию о соединяемых источниках данных и т.д.

Компонент «Слияние» необходим, когда к информации, содержащейся в некотором наборе данных, необходимо добавить дополнительную информацию из другого набора данных.

Работу указанного обработчика укажем на следующем примере.

Пример

Пусть дана исходная таблица.

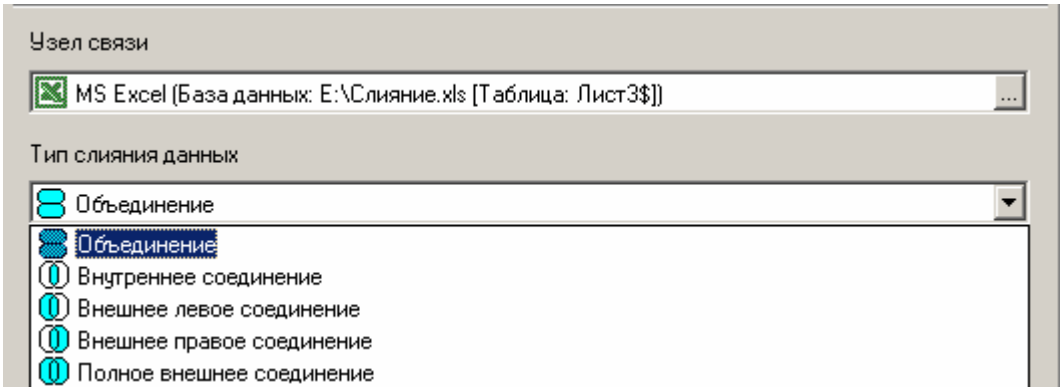
Поставщик	Товар
ЖБИ	Бетон
ЖБИ	Плита
КРЗ	Рубероид
КРЗ	Картон

Допустим, необходимо присоединить к имеющейся таблице следующие данные по истории продаж:

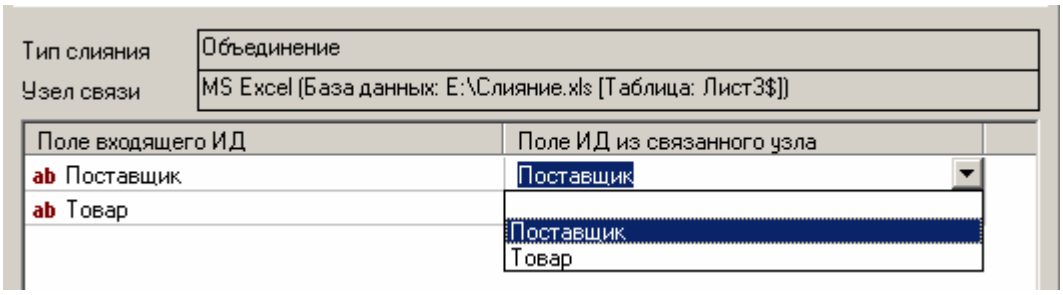
Дата	Поставщик	Товар	Количество
10.02.2004	ЖБИ	Бетон	100
10.03.2004	КРЗ	Рубероид	10
10.03.2004	КРЗ	Рубероид	20
10.03.2004	ЖБИ	Плита	5
10.03.2004	ЖБИ	Бетон	130
11.03.2004	КРЗ	Картон	20

А. Объединение

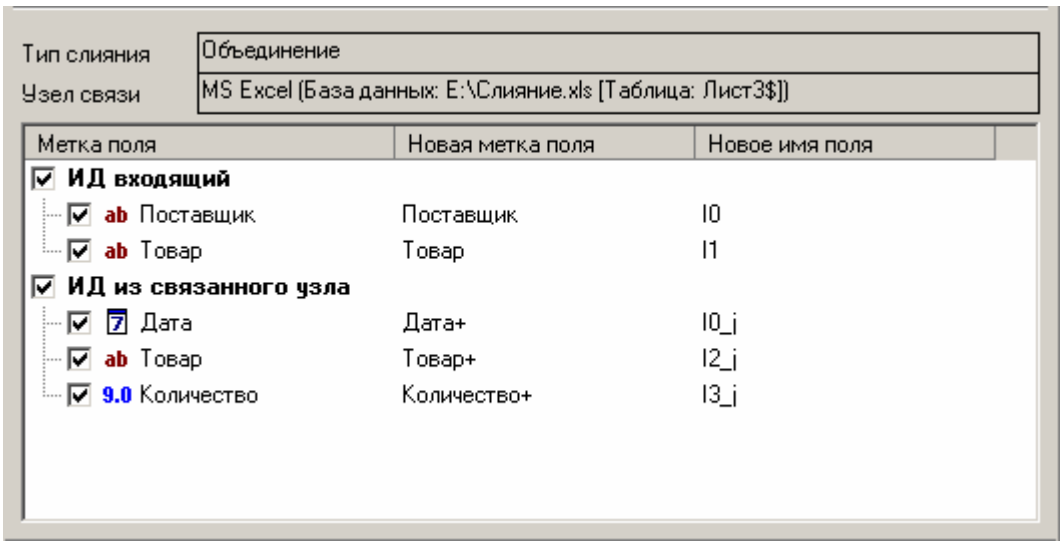
На первом шаге в Мастере обработки выбираем тип слияния «*Объединение*»:



На следующем шаге в мастере обработки указываем поля, по которым будут связываться наборы данных. В данном примере соединение ведется по полю *Поставщик*:



Далее в мастере обработки указываем те поля, которые будут отображаться в выходном наборе данных. В данном случае указываем для отображения все поля обоих источников данных:



Соединяя эти таблицы по полю *Поставщик* с типом слияния «*Объединение*» и включая все поля обеих таблиц в выходной результат, получится следующий набор данных.

Дата+	Количество+	Поставщик	Товар	Товар+
		ЖБИ	Бетон	
		ЖБИ	Плита	
		КРЗ	Рубероид	
		КРЗ	Картон	
10.02.2004	100	ЖБИ		Бетон
10.03.2004	10	КРЗ		Рубероид
10.03.2004	20	КРЗ		Рубероид
10.03.2004	5	ЖБИ		Плита
11.03.2004	130	ЖБИ		Бетон
11.03.2004	20	КРЗ		Картон

Как видно, в результате выполнения такого слияния к исходной таблице добавились столбцы и строки из связываемой таблицы. Причем добавление строк происходит снизу.

В. Внутреннее соединение

Рассмотрим соединение двух таблиц с типом слияния «*Внутреннее соединение*».

В качестве исходных данных возьмем следующие таблицы:

Поставщик	Товар
ЖБИ	Бетон
КРЗ	Картон
НЕФТЕБАЗА	Мазут

Таблица с историей продаж:

Дата	Поставщик	Товар	Количество
10.02.2004	ЖБИ	Бетон	100
11.03.2004	КРЗ	Картон	20
12.03.2004	ДСК	Бетон	30

Осуществляя внутреннее соединение этих таблиц по полю *Поставщик*, получится следующий результат:

Поставщик	Товар	Дата+	Поставщик+	Товар+	Количество+
ЖБИ	Бетон	10.02.2004	ЖБИ	Бетон	100
КРЗ	Картон	11.03.2004	КРЗ	Картон	20

Как и в предыдущем случае в выходной набор данных были включены все поля исходной и связываемой таблицы. В результирующую таблицу были добавлены те строки и столбцы из исходной и связываемой таблицы, для которых нашлись совпадающие значения в поле *Поставщик* связываемых таблиц. В данном случае такими значениями являются «ЖБИ» и «КРЗ».

C. Внешнее левое соединение

Рассмотрим соединение двух таблиц с типом слияния «*Внешнее левое соединение*».

Осуществляя внешнее левое соединение таблиц из примера B) по полю *Поставщик*, получится следующий результат.

Поставщик	Товар	Дата+	Поставщик+	Товар+	Количество+
ЖБИ	Бетон	10.02.2004	ЖБИ	Бетон	100
КРЗ	Картон	11.03.2004	КРЗ	Картон	20
НЕФТЕБАЗА	Мазут				

Как и в предыдущем случае в выходной набор данных были включены все поля исходной и связываемой таблицы. В результате к исходной таблице были присоединены только те строки и столбцы из связываемой таблицы, для которых нашлись совпадающие значения в поле *Поставщик* из обеих таблиц. Поскольку такими значениями являются «ЖБИ» и «КРЗ», то запись со значением «ДСК» поля *Поставщик* из присоединяемой таблицы в итоговую таблицу не будет включена.

D. Внешнее правое соединение

Рассмотрим соединение двух таблиц с типом слияния «*Внешнее правое соединение*».

Осуществляя «Внешнее правое соединение» таблиц из примера B) по полю *Поставщик,* получится следующий результат:

Поставщик	Товар	Дата+	Поставщик+	Товар+	Количество+
ЖБИ	Бетон	10.02.2004	ЖБИ	Бетон	100
КРЗ	Картон	11.03.2004	КРЗ	Картон	20
		12.03.2004	ДСК	Бетон	30

В выходной набор данных были включены все поля исходной и связываемой таблицы. В результате ко второй таблице были присоединены только те строки и столбцы из исходной таблицы, для которых нашлись совпадающие значения в поле *Поставщик* из обеих таблиц. Поскольку такими значениями являются «ЖБИ» и «КРЗ», то запись со значением «НЕФТЕБАЗА» поля *Поставщик* из исходной таблицы в итоговую таблицу не будет включена.

Е. Полное внешнее соединение

Рассмотрим соединение двух таблиц с типом слияния «*Полное внешнее соединение*».

Осуществляя «Полное внешнее соединение» таблиц из примера В) по полю *Поставщик*, получится следующий результат:

Поставщик	Товар	Дата+	Поставщик+	Товар+	Количество+
ЖБИ	Бетон	10.02.2004	ЖБИ	Бетон	100
КРЗ	Картон	11.03.2004	КРЗ	Картон	20
НЕФТЕБАЗА	Мазут				
		12.03.2004	ДСК	Бетон	30

В выходной набор данных были включены все поля исходной и связываемой таблицы. Результирующая таблица получается следующим образом: при совпадении значений ключевого поля «Поставщик», по которому происходит соединение, в результирующую таблицу включаются все поля из обеих таблиц с реальными данными. Значение «НЕФТЕБАЗА» поля *Поставщик* исходной таблицы отсутствует в соответствующем поле присоединяемой таблицы, поэтому значения в присоединенных столбцах в этой строке принимают пустые значения. Соответственно значения «ДСК» поля «Поставщик» нет в аналогичном столбце исходной таблицы, поэтому в результирующей таблице значения столбцов *Поставщик* и *Товар* в последней строке принимают пустые значения.

Компонент «Слияние» позволяет соединять и обрабатывать необходимым образом наборы данных, полученные из разных, не связанных между собой источников данных.

Замена данных

В результате выполнения этой операции производится замена значений по таблице подстановки, которая содержит пары, состоящие из исходного значения и выходного значения. Например, 0 – «красный», 1 – «зеленый», 2 – «синий». Или «зима» – «январь», «весна» – «апрель», «лето» – «июль», «осень» - «октябрь». Для каждого значения исходного набора данных ищется соответствие среди исходных значений таблицы подстановки. Если соответствие найдено, то значение меняется на соответствующее выходное значение из таблицы подстановки. Если значение не найдено в таблице, оно может быть либо заменено значением, указанным для замены «по умолчанию», либо оставлено без изменений (если такое значение не указано). Кроме того, можно указать значения, которые нужно вставить вместо пустых ячеек.

Пример

Пусть, например, есть список клиентов и каким-либо образом каждый клиент был отнесен в одну из трех групп. Группа задана номером. Таблица может быть такой.

Наименование клиента	Группа
Клиент 1	1
Клиент 2	3
Клиент 3	2
Клиент 4	1
Клиент 5	2
…	…

Понять, что представляет каждый клиент по такой таблице, невозможно. Но известно, что соответствует каждой группе.

Группа	Наименование
1	Постоянные
2	Случайные
3	Потерянные

Это таблица подстановки. Воспользуемся ей для замены значений группы в таблице клиентов.

Наименование клиента	Группа
Клиент 1	Постоянные
Клиент 2	Потерянные
Клиент 3	Случайные
Клиент 4	Постоянные
Клиент 5	Случайные
…	…

Группировка

Назначение

Аналитику для принятия решения часто необходима сводная информация, т.е. сгруппированные данные. Совокупные данные намного более информативны, особенно если их можно получить в

разных разрезах. В Deductor Studio предусмотрен инструмент, реализующий сбор сводной информации – «Группировка». Группировка позволяет объединять записи по полям-измерениям, агрегируя данные в полях-фактах для дальнейшего анализа.

Настройки

Для настройки группировки требуется указать, какие поля являются измерениями, а какие – фактами. Для каждого факта требуется указать функцию агрегации.

Стандартные варианты агрегации: сумма, среднее, максимум, минимум, количество. Количество – это число агрегируемых значений фактов для каждой комбинации измерений. Для строковых полей в качестве функции агрегации можно указать только максимум, минимум и количество. При этом максимум (минимум) двух строк рассчитывается посимвольным сравнением. Сначала сравниваются два первых символа строк. Если коды этих символов одинаковы, сравниваются вторые символы и т.д. Как только в строках появится первый несовпадающий символ, функция агрегации принимает значение строки, код символа в которой оказался больше (меньше).

Помимо стандартных вариантов агрегации можно еще рассчитать медиану, стандартное отклонение, выбрать первый и последний элемент в группе. Медиана рассчитывается следующим образом: все строки, попавшие в группу, сортируются по факту, по которому рассчитывается медиана, и из отсортированного списка выбирается средний по расположению в списке элемент. Медиана – это альтернатива среднему значению, устойчивое к аномальным выбросам. Стандартное отклонение – это показатель рассеивания значений параметра около его математического ожидания. Используется при нахождении стандартной ошибки, построении доверительных интервалов и т.д. Первый и последний элемент в группе выбирается в соответствии с естественным порядком, в котором эти элементы следуют в исходном наборе данных.

Принцип работы

В таблице данных ищутся записи с одинаковыми значениями полей, являющихся измерениями. Значения полей, выбранных как факты применяются функции агрегации.

Пример

Сгруппируем объемы продаж по клиентам и месяцам. Для этого нужно назначить измерениями поля: клиент и месяц. Поле с объемом продаж назначить фактом и указать для него функцию агрегации – сумма.

К результату можно снова применить операцию группировки. Например, можно задать измерением поле месяц, а фактом – объемы продаж, указав в качестве функции агрегации среднее. Результатом будут объемы продаж в месяц в среднем по всем клиенту.

Зачем проводить группировку данных, если это может сделать OLAP-куб? При использовании инструмента «Группировка» формируется таблица со сгруппированными значениями, которую в отличие от OLAP-куба можно использовать для обработки другими алгоритмами программы.

Например, необходимо построить прогноз объемов продаж. Обычно данные о продажах собираются в определенный промежуток времени, например, раз в день. В таком случае желательно группировать объемы продаж по неделям. Использование выборки по дневным продажам даст плохие результаты, так как продажи по каждому дню в отдельности могут очень сильно отличаться. Однако объемы продаж за неделю или за месяц в среднем не так сильно зашумлены. Поэтому перед построением прогноза желательно применить две обработки: преобразование даты для приведения даты к неделе, в который она попадает, и группировка для вычисления объемов продаж за неделю.

Разгруппировка

Назначение

Группировка используется для объединения фактов по каким-либо измерениям. При этом под объединением понимается применение некоторой функции агрегации. Если в исходном наборе данных присутствовали какие-либо другие измерения, то теряется информация о значениях фактов в разрезе этих измерений. Алгоритм разгруппировки позволяет восстановить эти факты, но их значения восстанавливаются не точно, а пропорционально известному вкладу в сгруппированные значения.

Пример

Пусть есть таблица с объемами продаж некоторого товара за два месяца.

Месяц	Наименование товара	Количество
1	Товар 1	100
1	Товар 2	10
2	Товар 1	110
2	Товар 2	20

Построена модель, которая прогнозирует продажи на 2 месяца вперед. Для этого используется группировка с измерением *Месяц* и фактом *Количество* с последующим построением модели прогноза и применением обработчика «Прогнозирование». Результаты прогнозирования представляются в следующем виде:

Месяц	Количество
3	140
4	155

В результирующей таблице указаны общие объемы продаж на 3 и 4 месяц. Для расчета объемов продаж каждого товара по отдельности за месяц необходимо сделать разгруппировку. Результат будет следующим:

Месяц	Наименование товара	Количество
3	Товар 1	122,50
3	Товар 2	17,50
4	Товар 1	135,62
4	Товар 2	19,37

Поясним, как выполнена такая разгруппировка. Общее количество проданного товара за первый и второй месяцы 100 + 10 + 110 + 20 = 240. При этом первый товар внес в это количество 100 + 110 = 210 или (210/240)*100 = 87,5%.

После прогнозирования выяснилось, что общее прогнозное количество товара за третий месяц равно 140. Из них 87.5% приходится на «Товар 1». Это составляет 122,50. Прогноз количества товара за 4 месяц равен 155. Из них 87,5% приходится на *Товар 1*. Это составляет 135,62. Точно

также восстанавливаются значения для второго товара. Так удалось перейти от общего прогноза к прогнозу по каждой позиции.

В этом примере мы посчитали вклад *Товар 1* в сумму по всем месяцам. Однако такой расчет может оказаться неактуальным, так как пропорциональное соотношение продаваемого товара может изменяться с течением времени. Поэтому можно посчитать вклад первого товара в общее количество по последнему месяцу (в общем случае, разгруппировывать можно по любому числу последних месяцев, недель, дней и вообще по произвольному числу значений любого измерения). Тогда получим такую таблицу:

Месяц	Наименование товара	Количество
3	Товар 1	118,46
3	Товар 2	21,54
4	Товар 1	131,15
4	Товар 2	23,85

Общий объем за второй месяц 110 + 20 = 130. Из них 110 приходится на первый товар. Это (110/130)*100 = 84,6%. На второй приходится 15,4%.

Для настройки разгруппировки нужно выбрать поле, значения которого нужно восстановить, и указать ему назначение «Факт» (в примере это *Количество*). Затем следует выбрать восстанавливаемое измерение (в примере это *Наименование товара*). Для восстановления значений факта необходимо выбрать столбец, по которому проводится разгруппировка. В нашем примере прогнозируемое количество продаж товаров восстанавливается по полю *Количество*, т.е. разгруппировка рассчитывается на основании значений столбца *Количество* за прошлые периоды времени. Далее нужно выбрать способ восстановления: по всей выборке (в примере – по всем месяцам) или по последним N значениям какого-либо измерения (в примере – по одному последнему месяцу). В последнем случае предлагается выбрать измерение и количество его последних значений.

Кросс-таблица

Назначение

Обработчик «Кросс-таблица» предназначен для изменения структуры таблицы, а именно, перенесения значений полей в заголовки столбцов. Напоминает операцию транспонирования измерений в OLAP-кубе.

Пример

Пусть есть таблица с объемами продаж некоторых товаров за два месяца.

Месяц	Наименование товара	Количество
01.01.2008	Товар 1	100
01.01.2008	Товар 2	10
01.02.2008	Товар 1	110
01.02.2008	Товар 2	20

Применим к исходному набору данных обработчик «Кросс-таблица». На этапе настройки назначения полей зададим следующие параметры:

- Колонки – *Наименование товара*. Из значений этого поля будут сформированы новые столбцы в кросс-таблице.
- Строки – *Месяцы*. Из значений этого поля будут сформированы строки в кросс-таблице.
- Факты – *Количество*. Значения этого поля будут располагаться в «теле» кросс-таблицы.

При работе с обработчиком может возникнуть ситуация, когда в полях, по которым были сформированы столбцы, появляются новые значения. Эти значения изначально не были учтены, поэтому, чтобы напомнить пользователю о их появлении, в выходном наборе данных может появиться столбец *Прочие значения*. В него будут агрегироваться все факты, относящиеся к новым данным. Аналогично, если в исходном поле имеются пропуски в данных, то факты для них будут агрегироваться в столбце *Пропущенные значения*.

Результат преобразования исходной таблицы представлен ниже.

	Месяц	Товар 1	Товар 2
		Количество	Количество
▶	01.01.2008	100	10
	01.02.2008	110	20

Свертка столбцов

Назначение

Обработчик предназначен для изменения структуры таблицы, а именно, перенесения заголовков полей в значения строк и столбцов.

Пример

Пусть есть таблица с объемами продаж некоторых товаров за три месяца.

Группа	Товар	Январь	Февраль	Март
Группа 1	Товар 1	54		
Группа 1	Товар 2		31	46
Группа 1	Товар 3	63	61	
Группа 2	Товар 1	77	19	93
Группа 2	Товар 4	63	51	70

Применим к исходному набору данных обработчик «Свертка столбцов». На этапе настройки назначения полей зададим следующие параметры:

- Информационные поля – *Группа* и *Товар*. Поля, помещенные в этот узел, не будут изменяться.
- Транспонируемые поля – *Январь*, *Февраль* и *Март*. Из значений полей, помещенных в этот узел, будут сформированы два новых столбца один со списком из меток, другой со списком значений. Транспонируемые столбцы должны быть одного типа.

Результат преобразования исходной таблицы представлен ниже.

Группа	Товар	Месяц	Количество
Группа 1	Товар 1	Январь	54
Группа 1	Товар 2	Февраль	31
Группа 1	Товар 2	Март	46
Группа 1	Товар 3	Январь	63
Группа 1	Товар 3	Февраль	61
Группа 2	Товар 1	Январь	77
Группа 2	Товар 1	Февраль	19
Группа 2	Товар 1	Март	93
Группа 2	Товар 4	Январь	63
Группа 2	Товар 4	Февраль	51
Группа 2	Товар 4	Март	70

Data Mining

Автокорреляция

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, отстоящих друг от друга на определенное количество отсчетов, называемые также лагом. Совокупность коэффициентов корреляции по всем лагам представляет собой автокорреляционную функцию ряда (АКФ):

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

По поведению АКФ можно судить о характере анализируемой последовательности и наличии периодичности (например, сезонной).

Очевидно, что при k = 0, автокорреляционная функция будет максимальной и равной 1, т.е. значение последовательности полностью коррелированно само с собой, степень статистической взаимозависимости максимальна. Действительно, если факт появления данного значения имел место, то и соответствующая вероятность равна 1. По мере увеличения числа лагов, т.е. увеличения расстояния между двумя значениями, для которых вычисляется коэффициент корреляции, значения АКФ будут убывать из-за уменьшения статистической взаимозависимости между этими значениями (вероятность появления одного из них все меньше влияет на вероятность появления другого). При этом чем быстрее убывает АКФ, тем быстрее изменяется анализируемая последовательность. И наоборот, если АКФ убывает медленно, то и соответствующий процесс является относительно гладким. Если в исходной выборке имеет место тренд (плавное увеличение или уменьшение значений ряда), то плавное изменение АКФ также будет иметь место. При наличии сезонных колебаний в исходном наборе данных, АКФ также будет иметь периодические всплески.

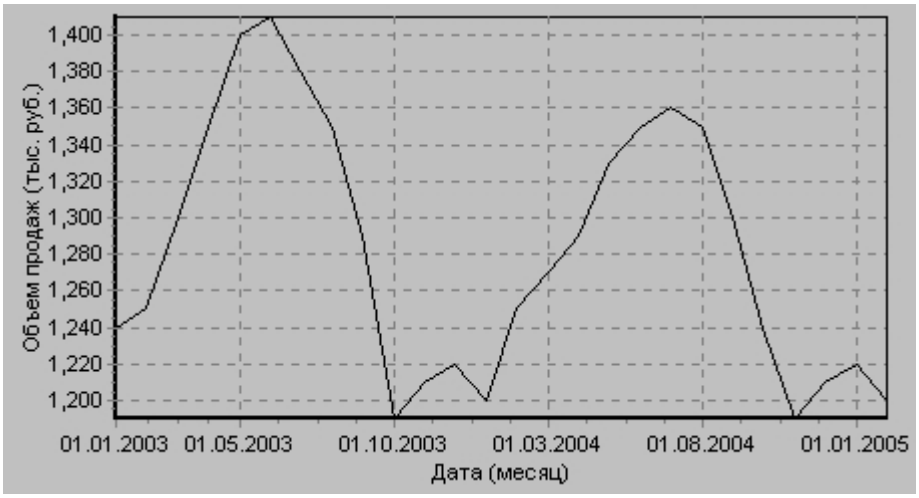
Для применения алгоритма автокорреляции в Deductor Studio необходимо выбрать поле, для которого вычисляется АКФ. В поле «Количество отсчетов» требуется указать количество отсчетов, для которых будут рассчитаны значения АКФ.

Пример

Есть таблица продаж некоторого товара за два с небольшим года.

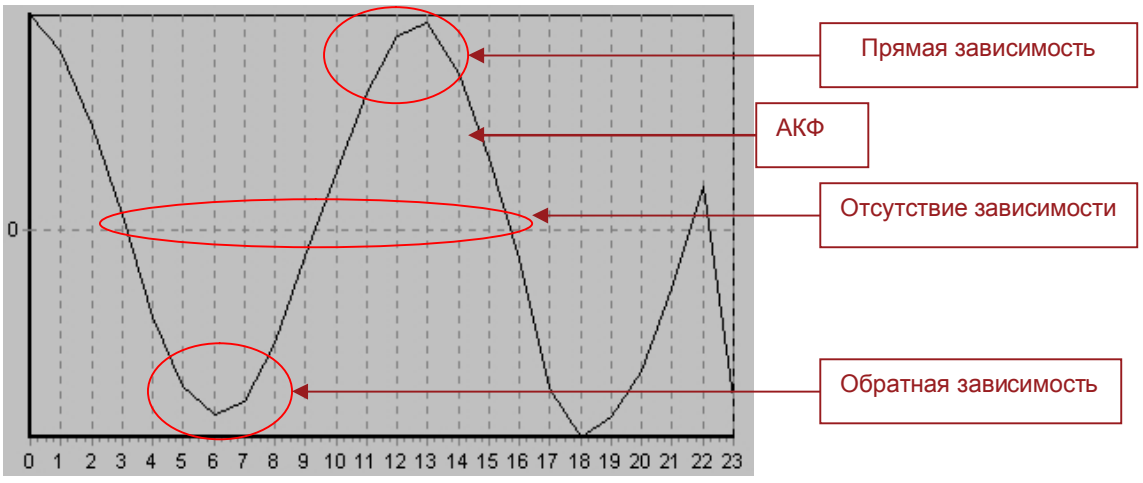
Дата (месяц)	Объем продаж
01.01.2003	1240,0
01.02.2003	1250,0
01.03.2003	1300,0
01.04.2003	1350,0
01.05.2003	1400,0
…	…

График зависимости поля *Объем продаж* от поля *Дата (месяц)* показан на графике.



Попробуем определить наличие сезонных зависимостей продаж этого товара.

Для оценки сезонности выбирают количество отсчетов, обычно, больше 12, если один отсчет соответствует месяцу, то есть сезонность ищется за период больше одного года. Вычислим автокорреляционную функцию для поля *Объем продаж*, установив количество отсчетов равным 24 (два года). На диаграмме АКФ выглядит следующим образом.



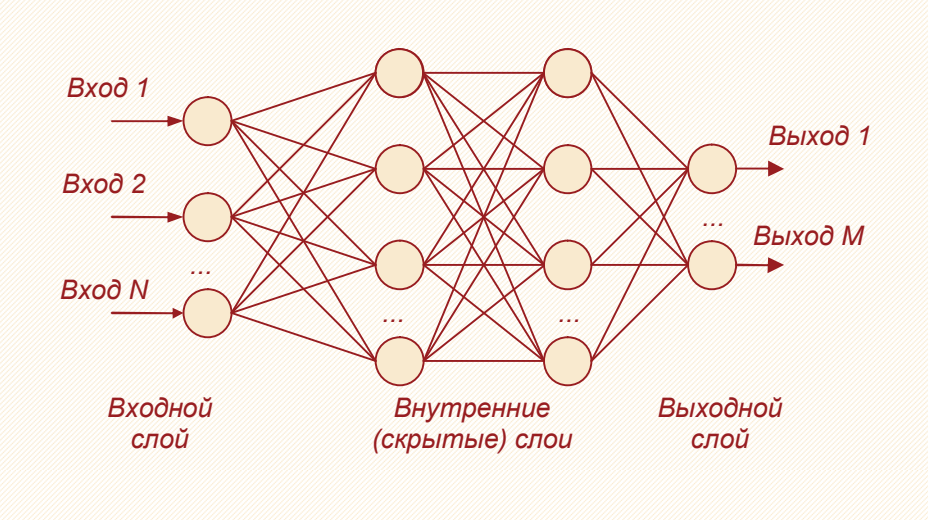
Первое максимальное значение АКФ находится на 12 и 13-м отсчетах (лагах). Это соответствует периоду сезонности, оказавшемуся равным одному году. Таким образом, номер отсчета, соответствующий максимуму АКФ, показывает количество месяцев, по прошествии которого наблюдается та же тенденция продаж.

На 6 отсчете АКФ имеет первое минимальное значение и это значение близко к –1. Это так называемая обратная автокорреляционная зависимость, она присутствует не всегда. В нашем случае это соответствует половине периода сезонности.

Нейронные сети

Нейронные сети (НС) представляют собой вычислительные структуры, моделирующие простые биологические процессы, аналогичные процессам, происходящим в человеческом мозге. Нейросети – это распределенные и параллельные системы, способные к адаптивному обучению путем реакции на положительные и отрицательные воздействия. В основе построения сети лежит элементарный преобразователь, называемый *искусственным нейроном* или просто *нейроном* по аналогии с его биологическим прототипом.

Структуру нейросети можно описать следующим образом. Нейросеть состоит из нескольких слоев: входной, внутренние (скрытые) и выходной слои. Входной слой реализует связь с входными данными, выходной – с выходными. Внутренних слоев может быть от одного и больше. В каждом слое содержится несколько нейронов.



Между нейронами есть связи, называемые *весами*.

В Deductor в основе обработчика «Нейросеть» лежит многослойный персептрон с двумя алгоритмами обучения – классическим BackProp и его модификацией RProp.

Назначение и подготовка обучающей выборки

Нейросеть способна имитировать какой-либо процесс. Любое изменение входов нейросети ведет к изменению ее выходов. Причем выходы нейросети однозначно зависят от ее входов.

Перед тем как использовать нейросеть, ее необходимо обучить. Задача обучения здесь равносильна задаче аппроксимации функции, то есть восстановление функции по отдельно взятым ее точкам – таблично заданной функции. Таким образом, для обучения нужно подготовить таблицу с входными значениями и соответствующими им выходными значениями, то

есть подготовить обучающую выборку. По такой таблице нейросеть сама находит зависимости выходных полей от входных. Далее эти зависимости можно использовать, подавая на вход нейросети некоторые значения. На выходе будут восстановлены зависимые от них значения. Причем на вход можно подавать значения, на которых нейросеть не обучалась.

Важно следующее. Обучающая выборка не должна содержать противоречий, так как нейросеть однозначно сопоставляет выходные значения входным. После обучения на вход нейросети необходимо подавать значения из диапазона, на котором она обучалась. Например, если при обучении нейросети на один из ее входов подавались значения от 0 до 100, то в дальнейшем следует на этот вход подавать значения из диапазона от 0 до 100. Допускается подавать значения, лежащие рядом с диапазоном.

Настройка назначения полей

В самом начале работы с нейросетью нужно определиться, что является ее входами, а что – выходами. Предполагается, что у нас уже есть таблица с обучающей выборкой. Обычно для ее подготовки пользуются методами очистки и трансформации данных – редактируются аномалии, заполняются или удаляются пропуски, устраняются дубликаты и противоречия, производится квантование и табличная замена, данные приводятся к скользящему окну, преобразуется формат данных.

Нормализация значений полей

После того, как указаны входные и выходные поля, следует нормализация данных в обучающей выборке. Целью нормализации значений полей является преобразование данных к виду, наиболее подходящему для обработки алгоритмом. Для таких обработчиков как нейронная сеть, дерево решений, линейная модель прогнозирования данные, поступающие на вход, должны иметь числовой тип, а их значения должны быть распределены в определенном диапазоне. Нормализатор может преобразовать дискретные данные к набору уникальных индексов или значения, лежащие в произвольном диапазоне к диапазону [0..1]. Для нейросети доступны следующие виды нормализации полей.

1. *Линейная нормализация.* Используется только для непрерывных числовых полей. Позволяет привести числа к диапазону [min..max], то есть минимальному числу из исходного диапазона будет соответствовать min, а максимальному – max. Остальные значения распределяются между min и max.
2. *Уникальные значения.* Используется для дискретных значений. Такими являются строки, числа или даты, заданные дискретно. Чтобы привести непрерывные числа в дискретные, можно, например, воспользоваться обработкой «квантование». Так следует поступать для величин, для которых можно задать отношение порядка, то есть, если для двух любых дискретных значений можно указать, какое больше, а какое меньше. Тогда все значения необходимо расположить в порядке возрастания. Далее они нумеруются по порядку, и значения заменяются их порядковым номером.
3. *Битовая маска.* Используется для дискретных значений. Этот вид нормализации следует использовать для величин, которые можно только сравнивать на равенство или неравенство, но нельзя сказать, какое больше, а какое меньше. Все значения заменяются порядковыми номерами, а номер рассматривается в двоичном виде или в виде маски из нулей и единиц. Тогда каждая позиция маски рассматривается как отдельное поле, содержащее ноль или единицу. К такому полю можно применить линейную нормализацию, то есть заменить ноль некоторым минимальным значением, а единицу – максимальным. После такой нормализации на вход нейросети будет подаваться не одно это поле, а столько полей, сколько разрядов в маске.

Пример нормализации полей

1. Линейная нормализация. Приведем значения к диапазону [0..1].

Поле до нормализации	Поле после нормализации
–5	0
2,3	0,81111
1,1	0,67778
4	1
3,5	0,94444

Графически такое преобразование можно представить так.

2. Уникальные значения.

Поле до нормализации	Поле после нормализации
Маленький	1
Средний	2
Большой	3
Огромный	4

3. Битовая маска. Заменим значения битовой маской и приведем к диапазону [-1..1].

Поле до нормализации	Маска	Поля после нормализации	
Москва	00	-1	-1
Воронеж	01	-1	1
Рязань	10	1	-1
Тула	11	1	1

Кроме того, существует настройка нормализации полей по умолчанию, т.е. этап нормализации можно пропустить. В этом случае нормализация будет произведена автоматически в зависимости от вида данных полей:

1 *Дискретный* – нормализация битовой маской со способом кодирования – комбинация битов.

2 *Непрерывный* – линейная нормализация в диапазоне [-1..1].

Настройка обучающей выборки

После нормализации полей следует настроить обучающую выборку. Обучающую выборку разбивают на два множества – обучающее и тестовое.

Обучающее множество включает записи (примеры), которые будут использоваться непосредственно для обучения сети, т.е. будут содержать входные и желаемые выходные (целевые) значения.

Тестовое множество также включает записи (примеры), содержащие входные и желаемые выходные (целевые) значения, но используемое не для обучения сети, а для проверки результатов обучения.

Разбивать исходную выборку на эти множества можно двумя способами: либо по порядку, либо случайно. Если разбиение происходит по порядку, то тестовое множество выбирается либо из начала, либо из конца исходной выборки.

Настройка структуры нейросети

Далее задаются параметры, определяющие структуру нейронной сети – количество скрытых слоев и нейронов в них, а также активационная функция нейронов.

К выбору количества скрытых слоев и количества нейронов для каждого скрытого слоя нужно подходить осторожно. Хотя до сих пор не выработаны четкие критерии выбора, дать некоторые общие рекомендации все же возможно. Считается, что задачу любой сложности можно решить при помощи двухслойной нейросети, поэтому конфигурация с количеством скрытых слоев, превышающих 2, вряд ли оправдана. Для решения многих задач вполне подойдет однослойная нейронная сеть. При выборе количества нейронов следует руководствоваться следующим правилом: «количество связей между нейронами должно быть значительно меньше количества примеров в обучающем множестве». Количество связей рассчитывается как связь каждого нейрона со всеми нейронами соседних слоев, включая связи на входном и выходном слоях. Слишком большое количество нейронов может привести к так называемому «переобучению» сети, когда она выдает хорошие результаты на примерах, входящих в обучающую выборку, но практически не работает на других примерах.

Обучение нейросети

После настройки конфигурации сети следует выбрать алгоритм ее обучения.

Метод *обратного распространения ошибки* – итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущих значений выходов сети от требуемых. Одним из важнейших свойств алгоритма обратного распространения ошибки является высокая устойчивость, а, следовательно, надежность. Хотя нейронные сети, использующие алгоритм обратного распространения, являясь мощным инструментом поиска закономерностей, прогнозирования и качественного анализа, получили широкое распространение, им свойственны некоторые недостатки. К ним относится невысокая скорость сходимости (большое число требуемых итераций), что делает процесс обучения слишком долгим, и поэтому данный алгоритм оказывается неприменимым для широкого круга задач, требующих быстрого решения. Другие алгоритмы обучения нейросетей хотя и работают быстрее, в большинстве случаев обладают меньшей устойчивостью.

Для алгоритма обратного распространения ошибки нужно указать два параметра:

- *Скорость обучения* – определяет величину шага при итерационной коррекции весов в нейронной сети (рекомендуется в интервале 0…1). При большой величине шага сходимость будет более быстрой, но имеется опасность «перепрыгнуть» через решение. С другой стороны, при малой величине шага обучение потребует слишком многих итераций. На практике величина шага берется пропорциональной крутизне склона так, что алгоритм

замедляется вблизи минимума. Правильный выбор скорости обучения зависит от конкретной задачи и обычно делается опытным путем.

- *Момент* – задается в интервале 0…1. Рекомендуемое значение 0,9 ±0,1.
- Метод *Resilent Propogation (Rprop)* – эластичное распространение. Алгоритм использует так называемое «обучение по эпохам», когда коррекция весов происходит после предъявления сети всех примеров из обучающей выборки. Преимущество данного метода заключается в том, что он обеспечивает сходимость, а, следовательно, и обучение сети в 4-5 раз быстрее, чем алгоритм обратного распространения.

Для алгоритма Resilient Propagation указываются параметры:

- *Шаг спуска* – коэффициент увеличения скорости обучения, который определяет шаг увеличения скорости обучения при недостижении алгоритмом оптимального результата.
- *Шаг подъема* – коэффициент уменьшения скорости обучения. Задается шаг уменьшения скорости обучения в случае пропуска алгоритмом оптимального результата.

Далее необходимо задать условия, при выполнении которых обучение будет прекращено.

- *Считать пример распознанным, если ошибка меньше* – если рассогласование между эталонным и реальным выходом сети становится меньше заданного значения, то пример считается верно распознанным.
- *По достижении эпохи* – установка данного режима позволяет задать число эпох (циклов обучения), по достижении которого обучение останавливается независимо от величины ошибки. Если флажок сброшен, то обучение будет продолжаться, пока ошибка не станет меньше заданного значения, но при этом есть вероятность зацикливания, когда ошибка никогда не будет достигнута. Поэтому, желательно установить флажок по «Достижению эпохи», чтобы алгоритм был остановлен в любом случае.
- *Обучающее множество* – остановка обучения производится по достижении на обучающем множестве заданной средней ошибки, максимальной ошибки или процента распознанных примеров. Распознанным считается пример, для которого отклонение расчетного и реального значения не больше параметра «Считать пример распознанным, если ошибка меньше».
- *Тестовое множество* – остановка обучения производится по достижении на тестовом множестве заданной средней ошибки, максимальной ошибки или процента распознанных примеров. Распознанным считается пример, для которого отклонение расчетного и реального значения не больше параметра «Считать пример распознанным, если ошибка меньше».

Остановка обучения происходит по достижению любого из заданных условий остановки.

Теперь все готово к процессу обучения сети. В начале все веса нейросети инициализируются случайными значениями. После обучения эти веса принимают определенные значения. Обучение может с большой долей вероятности считаться успешным, если процент распознанных примеров на обучающем и тестовом множествах достаточно велик (близок к 100%).

Пример

Рассмотрим пример построения системы оценки кредитоспособности физического лица. Предположим, что эксперты определили основные факторы, определяющие кредитоспособность. Ими оказались: возраст, образование, площадь квартиры, наличие автомобиля, длительность проживания в данном регионе. В организации была накоплена статистика возвратов или невозвратов взятых кредитов. Эта статистика представлена таблицей.

Сумма кредита	Возраст	Образование	Площадь квартиры	Автомобиль	Срок проживания	Давать кредит
7000	37	Специальное	37	отечественная	22	Да
7500	38	Среднее	29	импортная	12	Да
14500	60	Высшее	34	Нет	30	Нет
15000	28	Специальное	14	отечественная	21	Да
32000	59	Специальное	53	отечественная	29	Да
11500	25	Специальное	28	отечественная	9	Да
5000	57	Специальное	18	отечественная	34	Да
61500	29	Высшее	26	Нет	18	Нет
13500	37	Специальное	46	отечественная	28	Нет
25000	36	Специальное	20	Нет	21	Нет
25500	68	Высшее	45	отечественная	30	Нет
…	…	…	…	…	…	…

Это обучающая выборка.

Теперь необходимо нормализовать поля. Поля «Сумма кредита», «Возраст», «Площадь квартиры» и «Длительность проживания» – непрерывные значения, которые преобразуем к интервалу [–1..1]. Образование представлено тремя уникальными значениями, которые можно сравнивать на большее или меньшее, а точнее лучшее или худшее, т.е. образование можно упорядочить так: среднее, специальное, высшее. Значения поля с наличием автомобиля упорядочить нельзя. Его нужно преобразовать к битовой маске. Для кодирования трех значений требуется два бита. Следовательно, это поле будет разбито на два.

Наличие автомобиля	Первый бит маски	Второй бит маски
Импортная	0	0
Отечественная	0	1
Нет	1	0

На этом нормализация закончена.

Обучающую выборку разобьем на обучающее и тестовое множества так, как программа предлагает это сделать по умолчанию, т.е. в обучающее множество попадут случайные 95 процентов записей, а остальные 5 процентов – в тестовое.

Конфигурация сети будет такой: во входном слое – 7 нейронов, то есть по одному нейрону на один вход (в обучающей выборке 6 столбцов, но столбец «Автомобиль» представлен битовой маской из двух бит, для каждого из которых создан новый вход). Сделаем один скрытый слой с двумя нейронами. В выходном слое будет один нейрон, на выходе которого будет решение о выдаче кредита.

Выберем алгоритм обучения сети Resilient Propagation с настройками по умолчанию. Условие окончания обучения оставим без изменения.

Обученную таким образом нейросеть можно использовать для принятия решения о выдаче кредита физическому лицу. Это можно сделать, применяя анализ «что-если». Для его включения нужно выбрать визуализатор «Что-если». Тогда откроется форма, представленная на рисунке.

Поле	Значение
Входные	
9.0 Сумма кредита	7000
9.0 Возраст	37
ab Образование	специальное
9.0 Площадь квартиры	37
ab Автомобиль	отечественная
9.0 Срок проживания	22
Выходные	
ab Давать кредит	Да

После изменения в этой таблице входных полей система сама принимает решение о выдаче кредита и в поле *Давать кредит* проставляет либо *Да*, либо *Нет*. Столбцы *Минимум* и *Максимум* определяют диапазон значений, на которых обучалась нейросеть. Следует придерживаться этих ограничений, хотя и возможно взять значения немного выходящие за границы диапазона.

Кроме такой таблицы анализ «что-если» содержит диаграмму, на которой отображается зависимость выходного поля от одного из входных полей при фиксированных значениях остальных полей. Например, требуется узнать, на какую сумму кредита может рассчитывать человек, обладающий определенными характеристиками. Это можно определить по диаграмме.

То есть человек в возрасте 37 лет со специальным образованием, имеющий квартиру площадью 37 кв. м, отечественный автомобиль и проживающий в данной местности 22 года может рассчитывать на сумму кредита не больше 20000.

Качество построенной модели можно определить по таблице сопряженности, которая является одним из визуализаторов.

	Классифицировано		
Фактически	Да	Нет	Итого
Да	50	9	59
Нет	27	63	90
Итого	77	72	149

Чем больше правильно классифицированных записей, тем лучше построенная модель.

Линейная регрессия

В результате работы данного компонента строится линейная модель данных. Применяется следующий алгоритм построения модели.

Пусть имеется набор входных значений $\mathbf{X}_i$, где $i = 1 \ldots n$, т.е $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$. Тогда можно указать такой набор выходных значений $\mathbf{Y}_j$ $(j = 1 \ldots m)$, который будет соответствовать линейной комбинации входных значений с коэффициентами a_i $(i = 1 \ldots n)$:

$$[1, x_1, x_2, \ldots, x_n]\,[a_0, a_1, a_2, \ldots, a_n] = [y_1, y_2, \ldots, y_m]$$

Если для простоты предположить, что выходное значение одно, то можно записать:

$$a_0 + x_1 a_1 + x_2 a_2 + \ldots + x_n a_n = y.$$

Таким образом, задача сводится к подбору коэффициентов a_i. Их оценка производится путем метода наименьших квадратов (МНК).

Нужно помнить, что линейная регрессия предназначена для поиска линейных зависимостей в данных. Если же зависимости нелинейные, то модель будет плохого качества. Это будет сразу видно на диаграмме рассеяния, т.к. прогнозные значения величины будут сильно разбросаны относительно действительных значений. В этом случае нужно использовать более мощные алгоритмы, например, нейронные сети.

Подготовка и настройка обучающей выборки, настройка назначений полей и их нормализация производятся так же, как и для нейронных сетей.

Пример

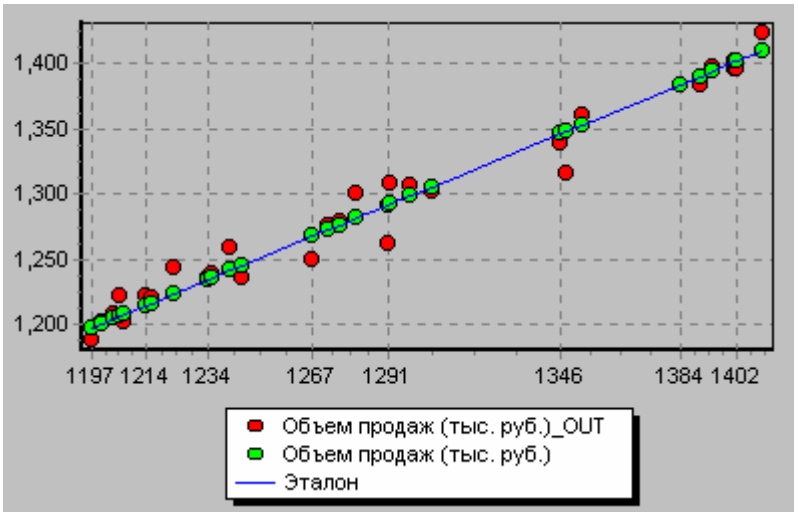
Есть данные о продажах, представленные таблицей.

Первый день месяца	Объем продаж (тыс. руб.)
…	…
01.01.2004	1000
01.02.2004	1160
01.03.2004	1210
01.04.2004	1130
01.05.2004	1250
01.06.2004	1300

Пусть, например, необходимо составить прогноз объемов продаж на следующий месяц, основываясь на продажах за три предыдущих месяца. Для этого преобразуем таблицу к скользящему окну.

Первый день месяца	Объем продаж три месяца назад (X-3)	Объем продаж два месяца назад (X-2)	Объем продаж месяц назад (X-1)	Объем продаж в текущем месяце (X)
…	…	…	…	…
01.04.2004	1000	1160	1210	1130
01.05.2004	1160	1210	1130	1250
01.06.2004	1210	1130	1250	1300

Входными полями у модели являются X-3, X-2, X-1, а выходным – X. Способ разбиения множеств оставим по умолчанию и перейдем сразу к построению модели. Запустим процесс построения модели. Для оценки качества построенной модели воспользуемся диаграммой рассеяния.



Судя по диаграмме, разброс между эталонными значениями выходного поля и значениями, рассчитанными моделью, достаточно невелик. Из этого можно сделать следующий вывод. Временной ряд хорошо укладывается в линейную модель и, следовательно, на основании этой модели можно строить прогноз на будущие периоды времени.

Прогнозирование

Прогнозирование позволяет получать предсказание значений временного ряда на число отсчетов, соответствующее заданному горизонту прогнозирования. Алгоритм прогнозирования работает следующим образом. Пусть в результате преобразования методом скользящего окна была получена последовательность временных отсчетов:

$$x(-n), \ldots, x(-2), x(-1), x$$

где x – текущее значение. Прогноз на $x(+1)$ строится на основании построенной модели. Чтобы построить прогноз для значения $x(+2)$, нужно сдвинуть всю последовательность на один отсчет

влево, чтобы ранее сделанный прогноз $x(+1)$ тоже вошел в число исходных значений. Затем снова будет запущен алгоритм расчета прогнозируемого значения и $x(+2)$ будет рассчитан с учетом $x(+1)$ и так далее в соответствии с заданным горизонтом прогноза.

Для настройки алгоритма прогнозирования необходимо задать горизонт прогноза, а также поля таблицы, которые необходимо подавать на вход модели для построения прогноза (для вычисления выходного поля модели).

Пример

Продолжим предыдущий пример. На основании построенной модели временного ряда спрогнозируем продажи на следующий месяц.

Для этого на входы модели необходимо подать значения о продажах за 3 месяца – 01.04.2004, 01.05.2005, 01.06.2004. А это есть поля $x-2$, $x-1$ и x. Следовательно, необходимо сопоставить поля, как показано в таблице.

Столбец	При очередном шаге брать значения из
$x-3$	$x-2$
$x-2$	$x-1$
$x-1$	x
x	

После применения алгоритма прогноза будет получена таблица.

Первый день месяца	x – 3	x – 2	x – 1	x	Шаг прогноза
…	…	…	…	…	
01.03.2004	1000	1160	1210	1130	
01.04.2004	1160	1210	1130	1250	
01.05.2004	1210	1130	1250	1300	
01.05.2004	1130	1250	1300	**1400**	1

Столбец x показывает историю продаж, включая спрогнозированное значение на следующий месяц. Для получения прогнозного значения 1400 на вход модели были поданы значения: 1130, 1250, 1300. Если задать горизонт прогноза 2, то для получения значения на второй шаг прогноза будут использоваться значения: 1250, 1300, 1400, то есть в расчете новой величины будет участвовать прогнозное значение, полученное на предыдущем шаге и так далее на любое число шагов прогноза. В связи с тем, что для получения прогноза на большое число шагов используются не реальные данные, а вычисленные моделью, ошибка такого прогноза может быть очень велика. Поэтому при построении прогноза не следует заглядывать слишком далеко вперед, так как с увеличением погрешности ценность полученного прогноза очень быстро падает.

Логистическая регрессия

Во многих приложениях наряду с классификацией объектов требуется ещё оценивать степень их принадлежности тому или иному классу или «степень уверенности» классификации. Это позволяет делать логистическая регрессия – распространенный статистический инструмент для решения задач регрессии и классификации. Иными словами, с помощью логистической регрессии можно оценивать вероятность того, что событие наступит для конкретного испытуемого (больной/здоровый, возврат кредита/дефолт и т.д.).

Логистическая регрессия – это разновидность множественной регрессии, общее назначение которой состоит в анализе линейной связи между несколькими независимыми переменными и зависимой переменной. Когда предсказываемых классов два, то говорят о бинарной логистической регрессии. В традиционной множественной линейной регрессии существует следующая проблема: алгоритм не «знает», что переменная отклика бинарна по своей природе. Это неизбежно приведет к модели с предсказываемыми значениями большими 1 и меньшими 0. Но такие значения вообще не допустимы для первоначальной задачи. Таким образом, множественная регрессия просто игнорирует ограничения на диапазон значений для y.

Для решения проблемы задача регрессии может быть сформулирована иначе: вместо предсказания бинарной переменной мы предсказываем непрерывную переменную со значениями на отрезке [0,1] при любых значениях независимых переменных. Это достигается применением логит-преобразования вида: $P = 1/(1 + e^{-y})$, где P – вероятность того, что произойдет интересующее событие; e – основание натуральных логарифмов 2,71…; y – стандартное уравнение регрессии:

$$y = a + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n .$$

Существует несколько способов нахождения коэффициентов логистической регрессии. На практике часто используют *метод максимального правдоподобия*. Он применяется в статистике для получения оценок параметров генеральной совокупности по данным выборки. Основу метода составляет *функция правдоподобия* (likehood function), выражающая плотность вероятности (вероятность) совместного появления результатов выборки. Для поиска максимума, как правило, используется оптимизационный метод Ньютона, для которого здесь всегда выполняется условие сходимости. Для облегчения вычислительных процедур максимизируют не саму функцию правдоподобия, а ее логарифм. В результатах обычно выводят численное значение $(-2 * \mathrm{Log\ likehood})$ либо на каждом шаге алгоритма, либо на последнем шаге.

Бинарная логистическая регрессия эквивалента построению рейтинговой или балльной модели, т.к. если признак f_j наблюдается у объекта x, то к сумме баллов добавляется вес a_j. Классификация производится путём сравнения набранной суммы баллов с *пороговым значением*. Благодаря свой простоте подсчёт баллов или скоринг (scoring) пользуется большой популярностью у экспертов в таких областях, как медицина, геология, банковское дело, социология, маркетинг и др.

Подготовка обучающей выборки

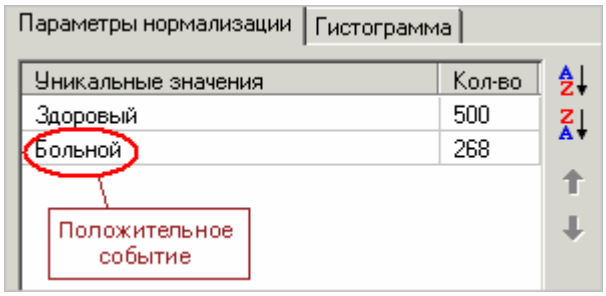
Для построения модели логистической регрессии готовится обучающая выборка так же, как это описано для нейросети. Но выходное поле может быть только дискретного типа и бинарное (т.е. количество уникальных значений по нему должно быть равно двум).

На этапе определения входов модели необходимо помнить, что естественное стремление учесть как можно больше потенциально полезной информации приводит к включению избыточных шумовых признаков. Экспериментально установлено, что для успешного обучения число примеров должно в несколько раз (примерно в 5) превосходить число входных признаков. Но даже если все признаки информативны, количества обучающих примеров может просто не хватить для надёжного определения коэффициентов регрессии при всех признаках. Когда данных мало, приходится искусственно упрощать структуру регрессионной модели, оставляя наиболее существенные признаки.

Нормализация значений полей

Для полей, подаваемых на входы, задается нормализация. Можно задать либо нормализацию битовой маской, либо нормализацию уникальными значениями (описание см. в разделе по нейросетям).

Для выходного поля (зависимой переменной) необходимо определиться с тем, что является отрицательным (negative), а что – положительным событием (positive). Это зависит от конкретной задачи. Например, если мы прогнозируем вероятность наличия заболевания, то положительным исходом будет класс «Больной пациент», отрицательным – «Здоровый пациент». И наоборот, если мы ходим определить вероятность того, что человек здоров, то положительным исходом будет класс «Здоровый пациент» и так далее. Отрицательное событие должно следовать в списке уникальных значений первым, положительное – вторым. По умолчанию они сортируются по алфавиту. Для переопределения типа события используйте окно **Нормализация**, вкладку **Параметры нормализации**.



Настройка обучающей выборки

Настройка обучающей выборки такая же, как для нейросети и других алгоритмов построения моделей.

Обучение логистической регрессионной модели

Под обучением понимается расчет коэффициентов регрессионной модели. В Deductor строится бинарная логистическая регрессия путем решения нелинейного уравнения итерационным методом Ньютона. Параметры обучения логистической модели следующие:

- *Максимальное число итераций* – алгоритм расчета коэффициентов завершится, когда очередное значение логарифмической функции правдоподобия $-2*\text{Log likehood}$ прекратит изменяться в пределах заданной точности. Если данная опция не включена, то ограничения на число итераций отсутствует.
- *Точность функции оценки* – параметр влияет на количество итераций алгоритма до его успешной сходимости.
- *Порог отсечения* – задача бинарной классификации будет решена на основе заданного порога отсечения для поля со значением рейтинга, по умолчанию порог равен 0,5.

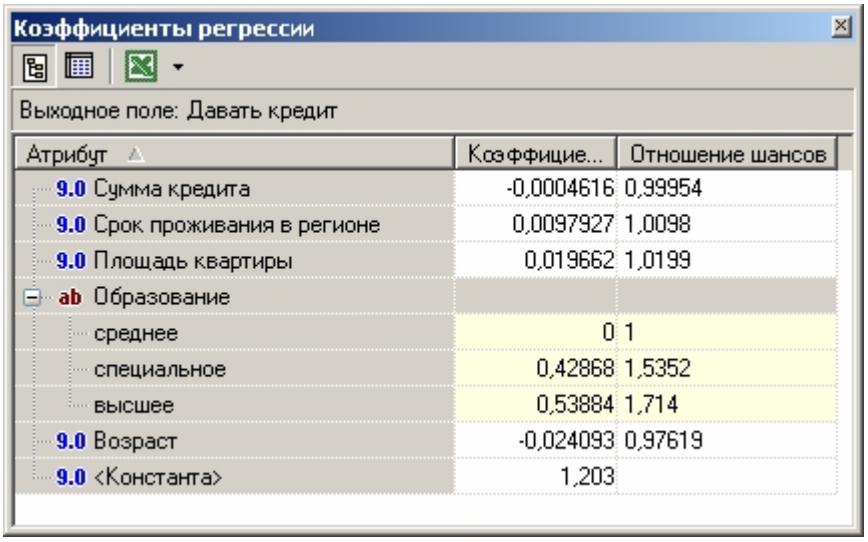
В результате работы алгоритма на выходе обработчика к исходному набору данных добавляются два поля:

- *<Имя выходного поля> Рейтинг* – значение выхода в уравнении логистической регрессии от 0 до 1;
- *<Имя выходного поля>_OUT* – выходное поле, полученное на основе поля с рейтингом с использованием порога отсечения k: всем примерам, большим или равным k, приписывается положительный исход, остальным – отрицательный исход прогнозируемого события.

В самом простом случае в качестве порога можно взять значение 0,5. Однако, при наличии некоторого критерия качества можно определить оптимальный порог (оптимальный балл). Это позволяет сделать специальный визуализатор ROC-анализа (см. одноименный раздел *ROC-анализ*). Также доступна для визуализации таблица сопряженности.

Пример

Продолжим пример об оценке кредитоспособности физических лиц, рассмотренный в пункте, посвященном нейронным сетям. Покажем, как при помощи логистической регрессии построить несложную скоринговую карту – список характеристик заемщика с их весами (баллами). Всех заемщиков можно разделить на два класса – кредитоспособных и некредитоспособных. Поскольку в кредитовании общепринято, что чем выше кредитоспособность клиента, тем больше его рейтинговый балл, обозначим в обучающем множестве положительным исходом успешный возврат кредита, а отрицательным – дефолт по займу. Для лучшей интерпретации регрессионных коэффициентов всем качественным переменным рекомендуется делать нормализацию битовой маской по способу кодирования «Позиция бита», в этом случае для каждого уникального значения качественного признака будет подбираться свой коэффициент (балл). От фактора «Автомобиль» здесь откажемся. Тогда при кодировании качественной переменной «Образование» в уравнение регрессии будут введены 2 дополнительных независимых переменных. В итоге получим 7 переменных плюс 1 переменная для константы. Откроем визуализаторы «Коэффициенты регрессии» , «Таблица сопряженности» и «Что-если».



В первом визуализаторе мы наблюдаем по сути скоринговую карту. Помимо коэффициента для каждой регрессионной переменной в таблице рассчитывается *отношение шансов* (odds ratio), т.к. именно оно помогает интерпретировать модель.

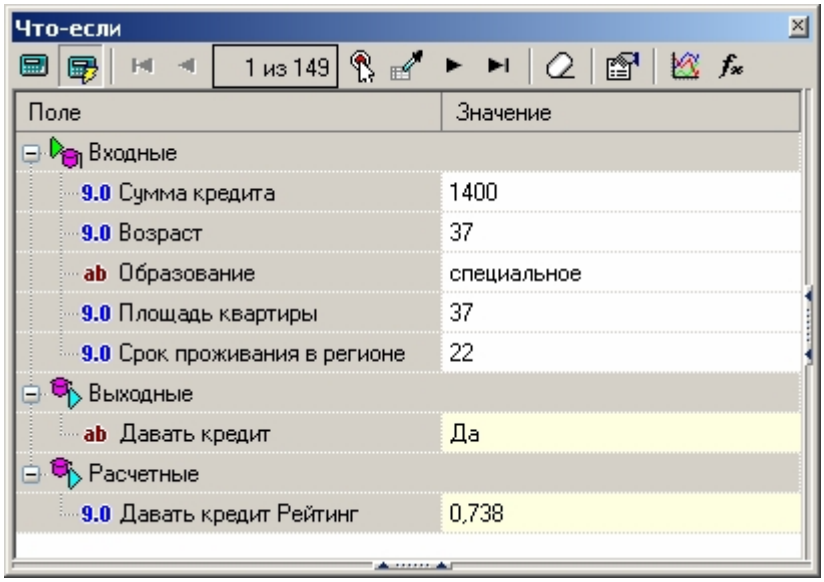
Отношение шансов OR – это отношение вероятности того, что событие произойдет к вероятности того, что событие не произойдет: $OR = p / (1 - p)$, где p – вероятность успеха.

В логистической регрессии коэффициенты x_i дают не только веса признаков, но и обладают полезным свойством. Рассчитав $\exp(x_i)$ для конкретной переменной, мы получим отношение шансов для этой переменной.

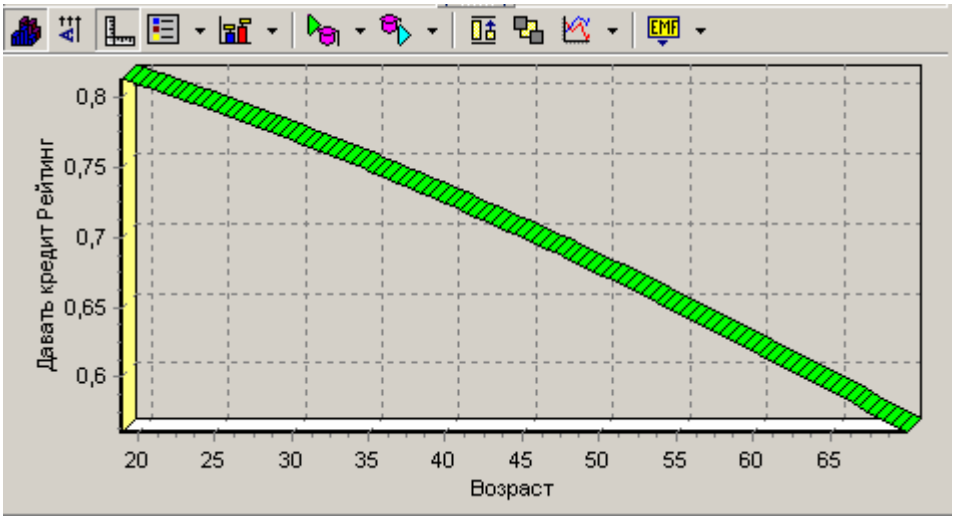
В нашем примере категориальный признак *Образование* имеет три веса: 1 при среднем образовании, 1,54 – при специальном и 1,71 – при высшем. Это значит, что у заемщика со специальным образованием шансы наступления того, что он окажется благонадежным, в 1,54 раза более вероятны, чем со средним. У заемщика с высшим образованием они еще выше (в 1,71 раза).

Для непрерывных переменных удобно рассчитать отношение шансов для заданного приращения переменной посредством вычисления $\exp(c \cdot x_i)$, где c – число единиц измерения. Например, для признака *Срок проживания в регионе* $\exp(10 \cdot 0{,}0098) = 1{,}1$, что означает: дополнительные 10 лет проживания в регионе увеличивают шансы возврата кредита в 1,1 раза.

Расчет рейтинга по такой скоринговой карте можно вести даже вручную. Но лучше воспользоваться инструментом «Что-если».



На диаграмме «Что-если» можно, к примеру, увидеть, как будет изменяться рейтинг этого заемщика с увеличением возраста:



Видно, что балльная модель демонстрирует линейную зависимость между возрастом и рейтингом, что в реальности встречается не всегда.

Оценить качество логистической регрессии как классификатора можно на основе таблицы сопряженности. По умолчанию порог отсечения равен 0,5.

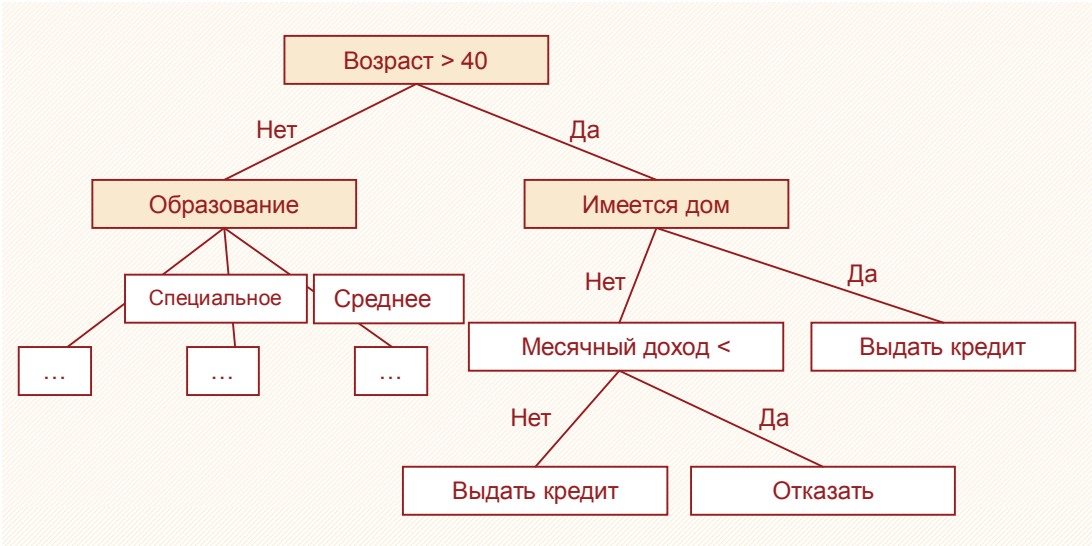
	Классифицировано		
Фактически	Да	Нет	Итого
Да	37	22	59
Нет	18	72	90
Итого	55	94	149

В этой таблице сопряженности зафиксировано 18 случаев ложного обнаружения (заемщик признан благонадежным, тогда как по факту он «плохой») и 22 случая ложного пропуска («хорошему» клиенту было отказано). Доля верно классифицированных случаев составила чуть более 73%. Это не самый высокий показатель, и его, скорее всего, можно улучшить, подобрав оптимальную пороговую точку. Это позволяет сделать ROC-анализ (см. соответствующий раздел настоящего Руководства).

Деревья решений

Деревья решений (decision trees) являются одним из наиболее популярных подходов к решению задач добычи данных. Они создают иерархическую структуру классифицирующих правил типа «ЕСЛИ…ТО…» (if-then), имеющую вид дерева. Чтобы принять решение, к какому классу следует отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид «значение параметра A больше B?». Если ответ положительный, осуществляется переход к правому узлу следующего уровня. Затем снова следует вопрос, связанный с соответствующим узлом, и т. д. Приведенный пример иллюстрирует работу так называемых бинарных деревьев решений, в каждом узле которых ветвление производится по двум направлениям (т. е. на вопрос, заданный в узле, имеется только два варианта ответов, например, *Да* или *Нет*). Однако, в общем случае ответов и, следовательно, ветвей, выходящих из узла, может быть больше.

Дерево решений состоит из узлов, где производится проверка условия и листьев – конечных узлов дерева, указывающих на класс (узлов решения).



Область применения деревьев решений в настоящее время весьма широка, но все задачи, решаемые этим аппаратом, могут быть объединены в три класса.

1. *Описание данных.* Деревья решений позволяют хранить информацию о данных в компактной форме. Вместо громоздких массивов данных можно хранить дерево решений, которое содержит точное описание объектов.
2. *Классификация.* Деревья решений отлично справляются с задачами классификации, т.е. отнесения объектов к одному из заранее известных классов.
3. *Регрессия.* Если целевая переменная является непрерывной, деревья решений позволяют установить зависимость целевой переменной от независимых (входных) переменных. Например, к этому классу относятся задачи численного прогнозирования (предсказания значений целевой переменной).

В Deductor в основе обработчика «Дерево решений» лежит модифицированный алгоритм C4.5, который позволяет решать только задачи классификации. Кроме того, предусмотрен режим полуручного построения

Подготовка обучающей выборки

Для построения дерева решений готовится обучающая выборка так же, как это описано для нейросети. Разница заключается в том, что выходное поле для дерева решений может быть только одно и только дискретно.

Нормализация значений полей

Для полей, подаваемых на входы и выход дерева решений, также задается нормализация. Можно задать либо линейную нормализацию, либо нормализацию уникальными значениями (описание в разделе по нейросетям).

Настройка обучающей выборки

Настройка обучающей выборки такая же, как для нейросети.

Обучение дерева решений

Параметры обучения дерева решений следующие:

- *Минимальное количество примеров*, при котором будет создан новый узел. Задается минимальное количество примеров, которое возможно в узле. Если примеров, которые попадают в данный узел, будет меньше заданного, узел считается листом (т.е. дальнейшее ветвление прекращается). Чем больше этот параметр, тем менее ветвистым получается дерево.
- *Строить дерево с более достоверными правилами в ущерб сложности*. Включает специальный алгоритм, который, усложняя структуру дерева, увеличивает достоверность результатов классификации. При этом дерево получается, как правило, более ветвистым.
- *Уровень доверия, используемый при отсечении узлов дерева*. Значение этого параметра задается в процентах и должно лежать в пределах от 0 до 100. Чем больше уровень доверия, тем более ветвистым получается дерево, и, соответственно, чем меньше уровень доверия, тем больше узлов будет отсечено при его построении.

Качество построенного дерева после обучения можно оценить по нескольким параметрам. Во-первых, это число распознанных примеров в обучающем и тестовом наборах данных. Чем выше это число, тем качественнее построенное дерево. Во-вторых, это количество узлов в дереве. При очень большом их числе дерево становится трудным для восприятия. Это также означает очень слабую зависимость выходного поля от входных полей.

Каждое правило характеризуется поддержкой и достоверностью.

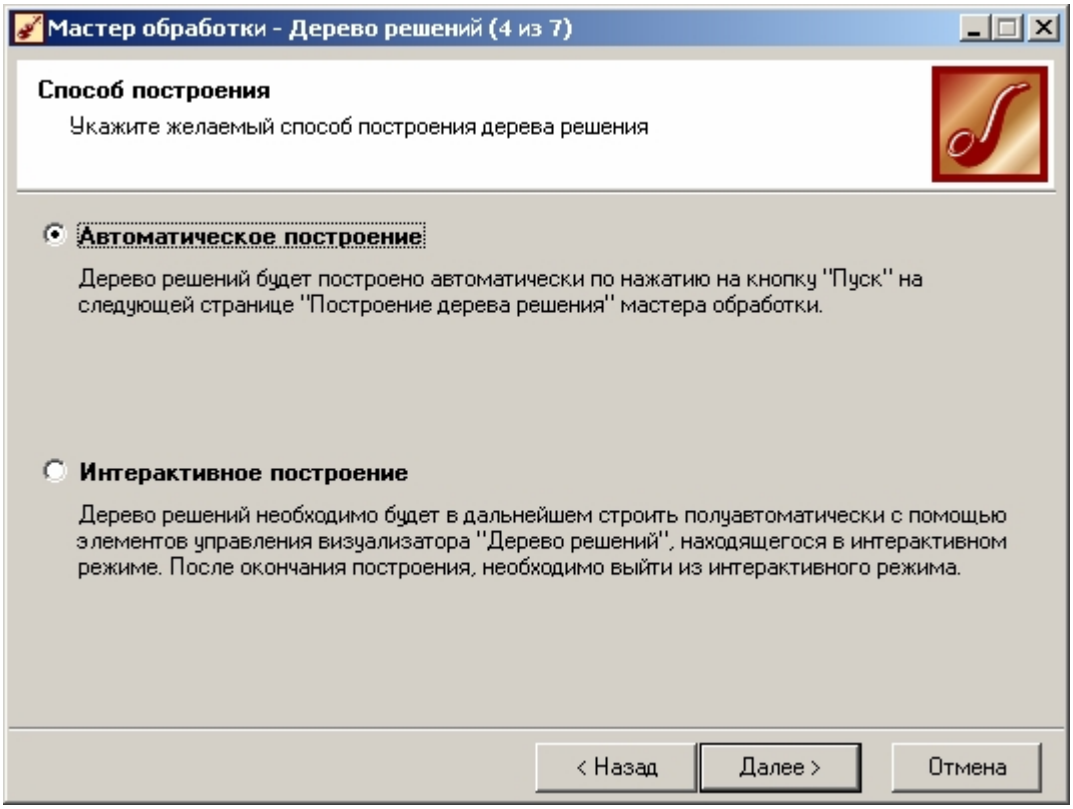
- *Поддержка* – общее количество примеров классифицированных данным узлом дерева.
- *Достоверность* – количество правильно классифицированных данным узлом примеров.

Пример

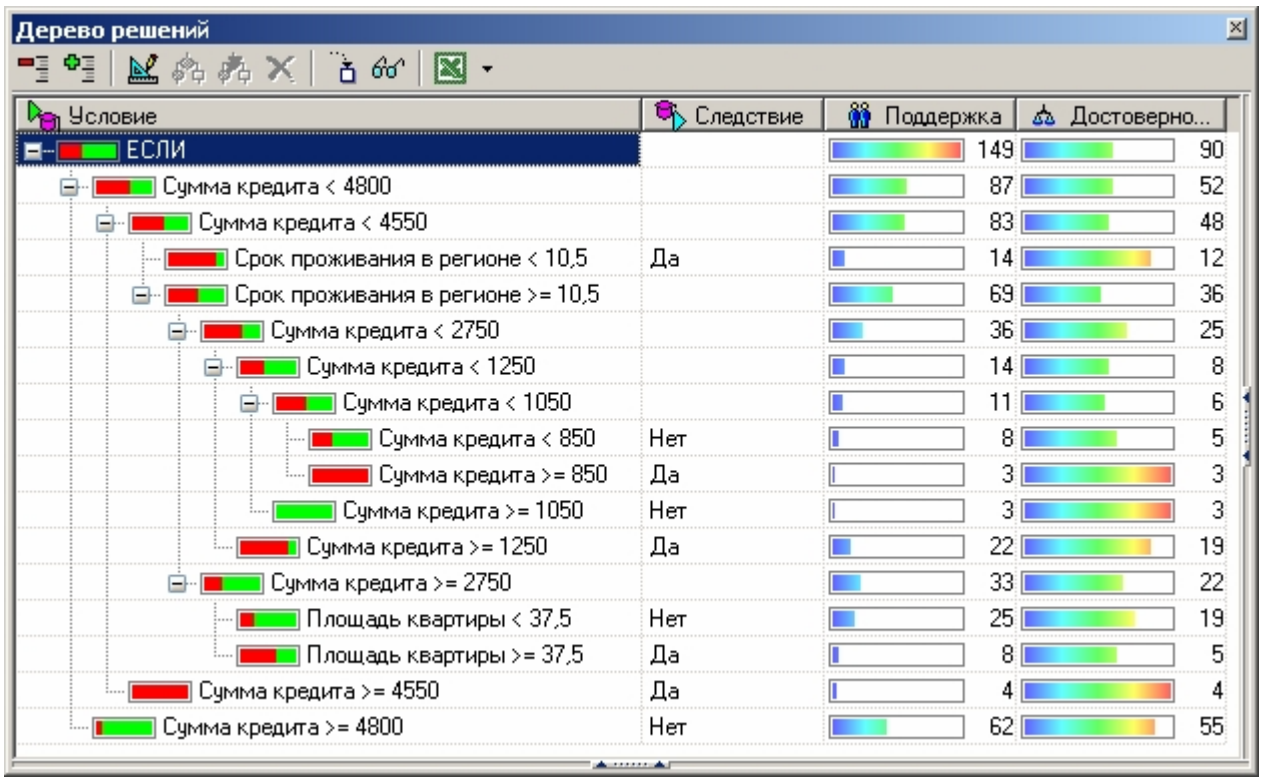
Продолжим рассматривать пример с оценкой кредитоспособности физических лиц. Очевидно, существуют некоторые правила отнесения заемщиков к тому или иному классу. Но при достаточно большом числе выбранных характеристик вручную практически невозможно определить эти правила. Решить эту задачу позволяют деревья решений. К тому же, в отличие от логистической регрессии деревья решений способны выявить нелинейные зависимости и нетипичные (редкие) случаи.

Обучающая выборка, а также правила получения обучающего и тестового множеств будут теми же, что и в примере с нейронными сетями и логистической регрессии. Нормализация полей в дереве решений не требуется. При построении правил зададим минимальное количество примеров, при котором будет создан новый узел равным 3. Будем строить дерево с более достоверными правилами в ущерб сложности.

Доверимся полностью автоматическому алгоритму построения дерева.



Полученное дерево решений содержит 17 узлов и 9 правил. Откроем визуализатор «Дерево решений».



Полученное дерево содержит в себе правила, следуя которым можно отнести заемщика в одну из групп риска и сделать вывод о выдаче кредита. Правила читаются с узлов, расположенных правее. Например, если сумма кредита меньше 4550 и срок проживания меньше 10,5, тогда выдать кредит. Следует заметить, что характеристики, лежащие ближе к вершине дерева, то есть левее, являются более значимыми.

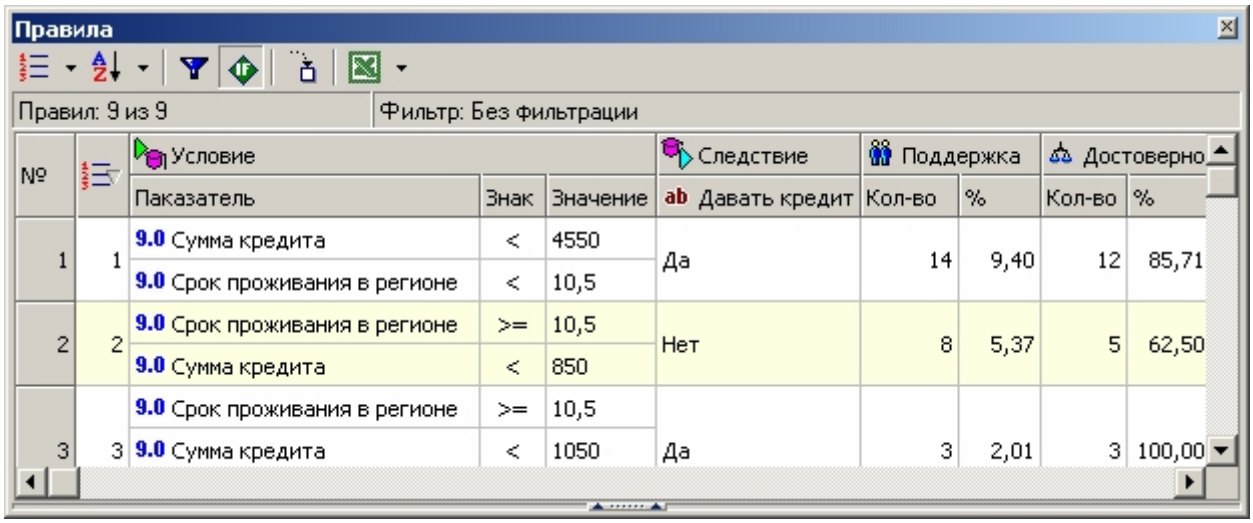
Построенные правила просматриваются в виде списка правил в визуализаторе «Правила».

Нажав на кнопку **Упрощать условия** можно сразу облегчить их восприятие. Например, условие в правиле

Если *Сумма кредита* < 4800 И *Сумма кредита* < 4550 И Срок проживания < 10,5

преобразуется к виду

Если *Сумма кредита* < 4550 И Срок проживания < 10,5.



№		Условие			Следствие	Поддержка		Достоверно	
		Показатель	Знак	Значение	Давать кредит	Кол-во	%	Кол-во	%
1	1	9.0 Сумма кредита	<	4550	Да	14	9,40	12	85,71
		9.0 Срок проживания в регионе	<	10,5					
2	2	9.0 Срок проживания в регионе	>=	10,5	Нет	8	5,37	5	62,50
		9.0 Сумма кредита	<	850					
3	3	9.0 Срок проживания в регионе	>=	10,5	Да	3	2,01	3	100,00
		9.0 Сумма кредита	<	1050					

Как и для нейросети, общее количество правильно классифицированных примеров можно посмотреть в таблице сопряженности.

Карты Кохонена

Самоорганизующиеся карты (карты Кохонена) могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации.

Алгоритм функционирования самоорганизующихся карт (Self Organizing Maps – SOM) представляет собой один из вариантов кластеризации многомерных векторов – алгоритм проецирования с сохранением топологического подобия.

Примером таких алгоритмов может служить алгоритм k-ближайших средних (k-means). Важным отличием алгоритма SOM является то, что в нем все нейроны (узлы, центры классов) упорядочены в некоторую структуру (обычно двумерную сетку). При этом в ходе обучения модифицируется не только нейрон-победитель (нейрон карты, который в наибольшей степени соответствует вектору входов и определяет, к какому классу относится пример), но и его соседи, хотя и в меньшей степени. За счет этого SOM можно считать одним из методов проецирования многомерного пространства в пространство с более низкой размерностью. При использовании этого алгоритма вектора, близкие в исходном пространстве, оказываются рядом и на полученной карте.

SOM подразумевает использование упорядоченной структуры нейронов. Обычно используются одно- и двумерные сетки. При этом каждый нейрон представляет собой n-мерный вектор-столбец, где n определяется размерностью исходного пространства (размерностью входных векторов). Применение одно- и двумерных сеток связано с тем, что возникают проблемы при отображении пространственных структур большей размерности (при этом опять возникают проблемы с понижением размерности до двумерной, представимой на мониторе).

Обычно нейроны располагаются в узлах двумерной сетки с прямоугольными или шестиугольными ячейками. При этом, как было сказано выше, нейроны также взаимодействуют друг с другом. Величина этого взаимодействия определяется расстоянием между нейронами на карте.

При реализации алгоритма SOM заранее задается конфигурация сетки (прямоугольная или шестиугольная), а также количество нейронов в сети. Некоторые источники рекомендуют использовать максимально возможное количество нейронов в карте. При этом начальный радиус обучения (neighbourhood в англоязычной литературе) в значительной степени влияет на способность обобщения при помощи полученной карты. В случае, когда количество узлов карты превышает количество примеров в обучающей выборке, успех использования алгоритма в большой степени зависит от подходящего выбора начального радиуса обучения. Однако, в случае, когда размер карты составляет десятки тысяч нейронов, время, требуемое на обучение карты, обычно бывает слишком велико для решения практических задач. Таким образом, необходимо достигать допустимый компромисс при выборе количества узлов.

Подготовка обучающей выборки

Отличием в подготовке обучающей выборки в этом алгоритме заключается в том, что выходные поля в такой выборке могут отсутствовать совсем. Даже если в обучающей выборке будут присутствовать выходные поля, они не будут участвовать при обучении нейросети, однако они будут участвовать при отображении карт.

Нормализация значений полей

Нормализация полей здесь такая же, как для нейросетей.

Настройка обучающей выборки

Обучающая выборка настраивается также, как и для нейросети и дерева решений.

Обучение

Перед началом обучения карты необходимо проинициализировать весовые коэффициенты нейронов. Удачно выбранный способ инициализации может существенно ускорить обучение и привести к получению более качественных результатов.

Существуют три способа инициирования начальных весов.

- *Случайными значениями*, когда всем весам даются малые случайные величины.
- *Из обучающего множества*, когда в качестве начальных значений задаются значения случайно выбранных примеров из обучающей выборки;
- *Из собственных векторов*. В этом случае веса инициируются значениями векторов, линейно упорядоченных вдоль линейного подпространства, проходящего между двумя главными собственными векторами исходного набора данных.

Визуализация

Полученную в результате обучения карту можно представить в виде слоеного пирога, каждый слой которого представляет собой раскраску, порожденную одной из компонент исходных данных. Полученный набор раскрасок может использоваться для анализа закономерностей, имеющихся между компонентами набора данных. После формирования карты получается набор узлов, который можно отобразить в виде двумерной картинки. При этом каждому узлу карты можно поставить в соответствие участок на рисунке (четырех или шестиугольный), координаты которого определяются координатами соответствующего узла в решетке. Теперь для визуализации остается только определить цвет ячеек этой картинки. Для этого и используются значения компонент. Самый простой вариант – использование градаций серого. В этом случае ячейки, соответствующие узлам карты, в которые попали элементы с минимальными значениями компонента или не попало вообще ни одной записи, будут изображены черным цветом, а ячейки, в которые попали записи с максимальными значениями такого компонента, будут соответствовать ячейке белого цвета. Более удобной является использование для раскраски цветной палитры. В принципе можно использовать любую градиентную палитру для раскраски.

Полученные раскраски в совокупности образуют атлас, отображающий расположение компонент, связи между ними, а также относительное расположение различных значений компонент.

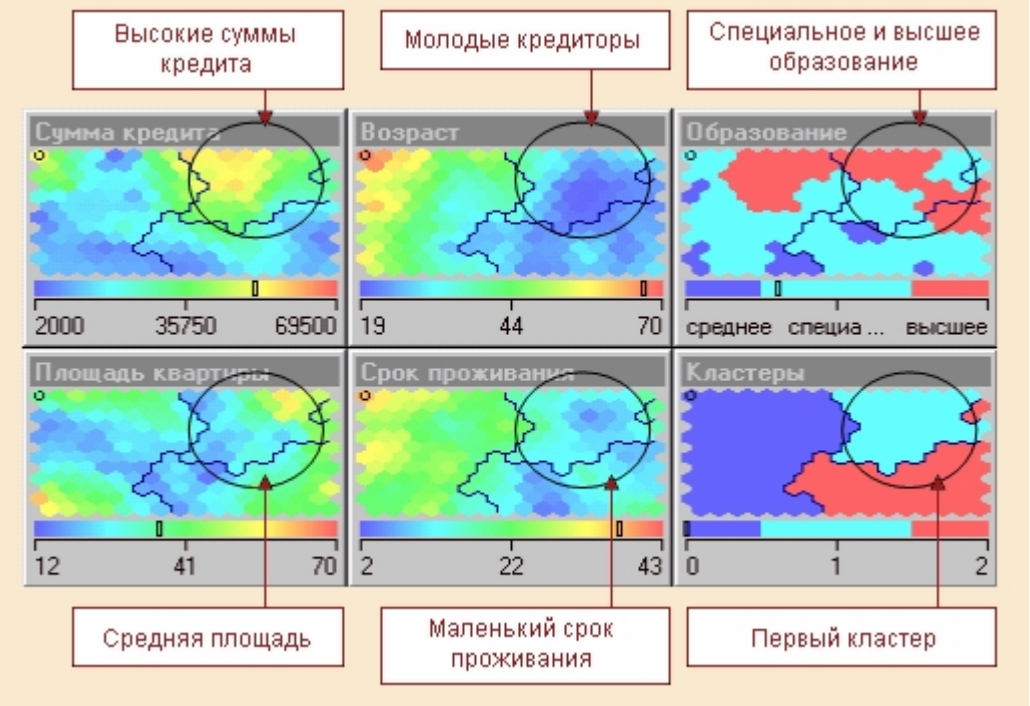
Пример

Продолжим пример с кредитованием физических лиц. С помощью самоорганизующихся карт Кохонена можно посмотреть зависимости между различными характеристиками заемщиков и выделить сегменты заемщиков, объединив их по схожим признакам.

Обучающей выборкой будет та же, что и для нейросетей и деревьев решений. Но поле «Автомобиль» использовать не будем. Это нецелесообразно, так как данные, подаваемые на вход карт Кохонена, должны быть такими, чтобы между ними можно было вычислить расстояние или, по крайней мере, была возможность расположить их в порядке возрастания или убывания. Значения же «отечественная», «импортная» и «нет автомобиля» сравнивать между собой не совсем корректно.

Нормализация полей будет такая же, как для деревьев решений. Разбиение обучающей выборки на два множества оставим по умолчанию, как и остальные настройки.

Результаты работы алгоритма отображаются на картах. Каждому входному полю соответствует своя карта.



К примеру, в один и тот же кластер были сгруппированы молодые заемщики с маленьким сроком проживания в данной местности, специальным или средним образованием, средней жилплощадью, берущие высокие суммы кредита. В результате кластеризации заемщики со схожими характеристиками попадут в один кластер, и поэтому для них можно применять одинаковые правила выдачи кредита, т.е. для каждого кластера определить, стоит ли выдавать кредит его представителям.

Для поиска зависимостей удобно использовать выходные поля в картах Кохонена. Вообще, самообучающиеся карты – это алгоритм обучения без учителя, поэтому выходных полей в том виде, какие они бывают при решении задач регрессии и классификации здесь нет. В данном случае выходным называется, которое не используется при обучении модели, но может отображаться на построенной карте. Например, в случае банковским кредитованием таким полем может быть признак возврата/невозврата кредита. Это поле не будет использоваться при построении модели, но можно будет просмотреть как сгруппированы на построенной карте хорошие и плохие заемщики и на при наличии закономерностей в их размещении, делать выводы относительно наличия зависимостей.

Общий принцип подобного способа анализа заключается в том, что новый объект «прогоняется» сквозь построенную карту и попадает в некоторый кластер, далее определяется какого соотношение хороших и плохих заемщиков в данном кластере и на основании этого делается вывод о вероятности возврата кредита.

Результаты кластеризации алгоритмом Кохонена можно увидеть не только на карте, но и специальном визуализаторе «Профили кластеров». Принцип его работы поясняется при описании обработчика «Кластеризация».

Кластеризация (*k-means* и *g-means*)

Кластеризация семействами алгоритмов *k-means* и *g-means* используется в тех же задачах, что сети Кохонена, но ее результаты уже не просмотреть в виде двумерной раскрашенной карты.

В основе работы *алгоритма k-means* лежит принцип оптимального в определенном смысле разбиения множества данных на k кластеров. Алгоритм пытается сгруппировать данные в кластеры таким образом, чтобы целевая функция алгоритма разбиения достигала экстремума.

Выбор числа k может базироваться на теоретических соображениях или интуиции. Введем понятие *центра кластера* – средние значения переменных объектов, входящих в кластер.

Алгоритм состоит из двух этапов.

1. Первоначальное распределение объектов по кластерам.

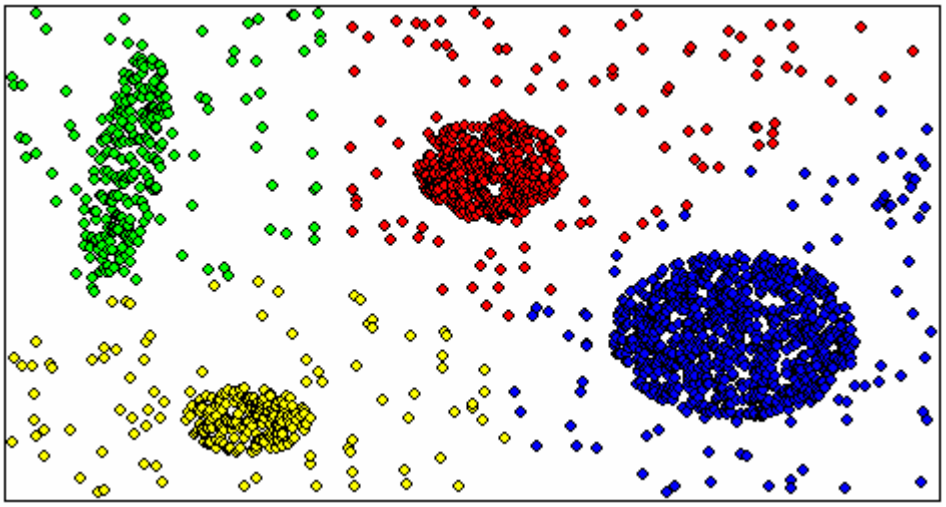
   Задается число k, и на первом шаге эти точки считаются «центрами» кластеров. Каждому кластеру соответствует один центр. Выбор начальных центров осуществляется случайным образом. В результате каждый объект назначен определенному кластеру.

2. Итерационный процесс.

   Вычисляются новые центры кластеров и объекты перераспределяются.

Процесс вычисления центров и перераспределения объектов продолжается до тех пор, пока не стабилизируются центры кластеров, т.е. все объекты будут принадлежать кластеру, которому они принадлежали до текущей итерации.

Иллюстрация работы алгоритма k-means приведена на рисунке.



Если число кластеров назначить затруднительно, то можно использовать алгоритм $g\text{-}means$. Он определяет число кластеров в модели на основании последовательного выполнения статистического теста на то, что данные внутри каждого кластера подчиняются определенному гауссовскому (*Gaussian*, отсюда и название алгоритма) закону распределения. Если тест дает отрицательный результат, кластер разбивается на два новых кластера (алгоритмом $k\text{-}means$) с центрами, расположенными на оси главных компонент.

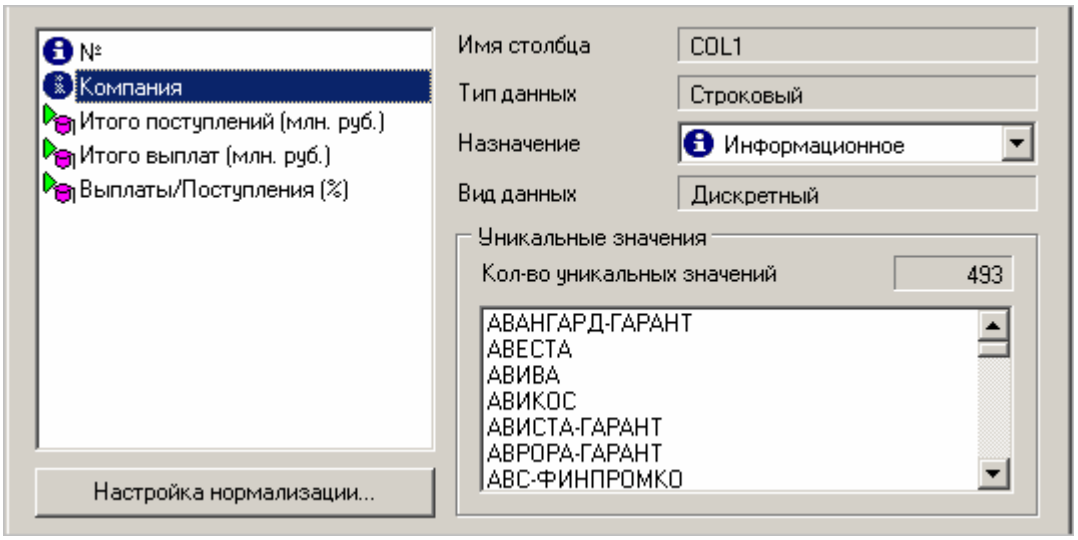
Важно помнить, что алгоритмы $k\text{-}means$ и $g\text{-}means$ ориентированы на гипотезу о компактности, которая предполагает, что данные обучающей выборки в виде многомерных векторов образуют в пространстве компактные сгустки сферической формы. В противном случае (например, вложенные друг в друга шары) кластеры, найденные этими алгоритмами, будут малоинформативными.

В качестве функции расстояния $k\text{-}means$ в Deductor использует:

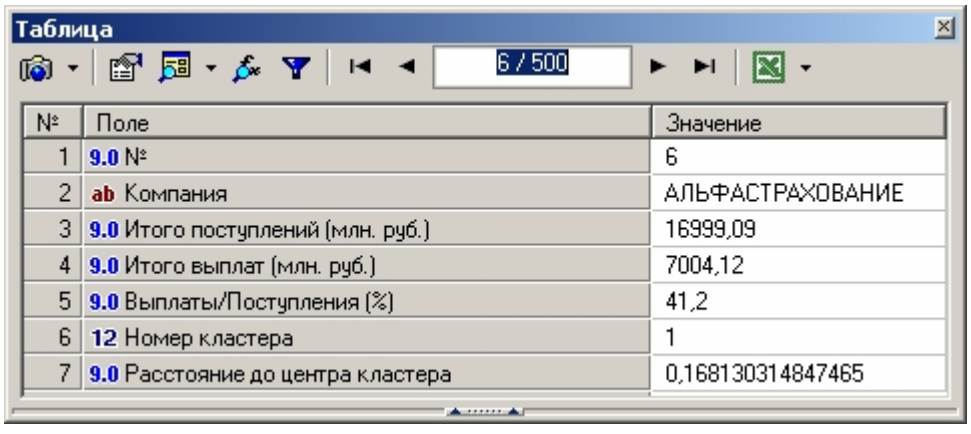
- для непрерывных числовых полей, а также упорядоченных категориальных признаков – евклидово расстояние;
- для неупорядоченных категориальных признаков – функцию отличия.

Пример

Возьмем открытые данные о 500 страховых компаниях РФ за 2008 год. Для каждой компании, кроме названия и позиции в рейтинге, известны три характеристики: *Итого поступлений (млн. руб.)*, *Итого выплат (млн. руб.)*, *Выплаты/Поступления (%)*. Очевидно, что страховые компании можно разбить на сегменты по этим показателям. Воспользуемся кластеризацией и назначим входные поля. Поля *№* и *Компания* сделаем информационными.

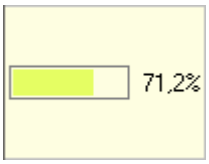
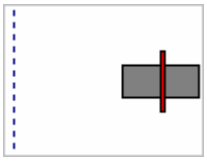


На вкладке разбиения исходного множества данных на обучающее и тестовое все записи отнесем к обучающим. Зададим число кластеров вручную равным 7. В качестве визуализаторов выберем «Таблица» и «Профили кластеров». В таблице появятся два новых поля – *Номер кластера* и *Расстояние до центра кластера*.

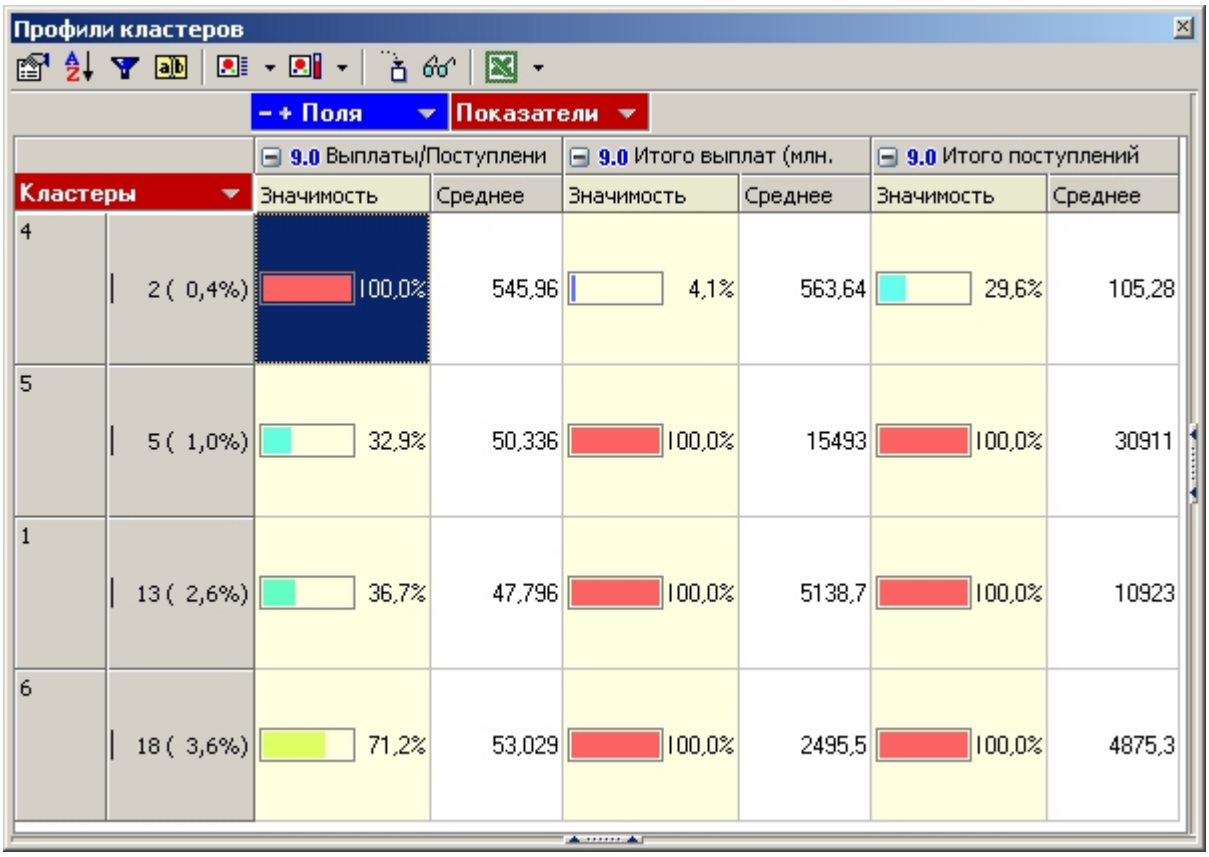


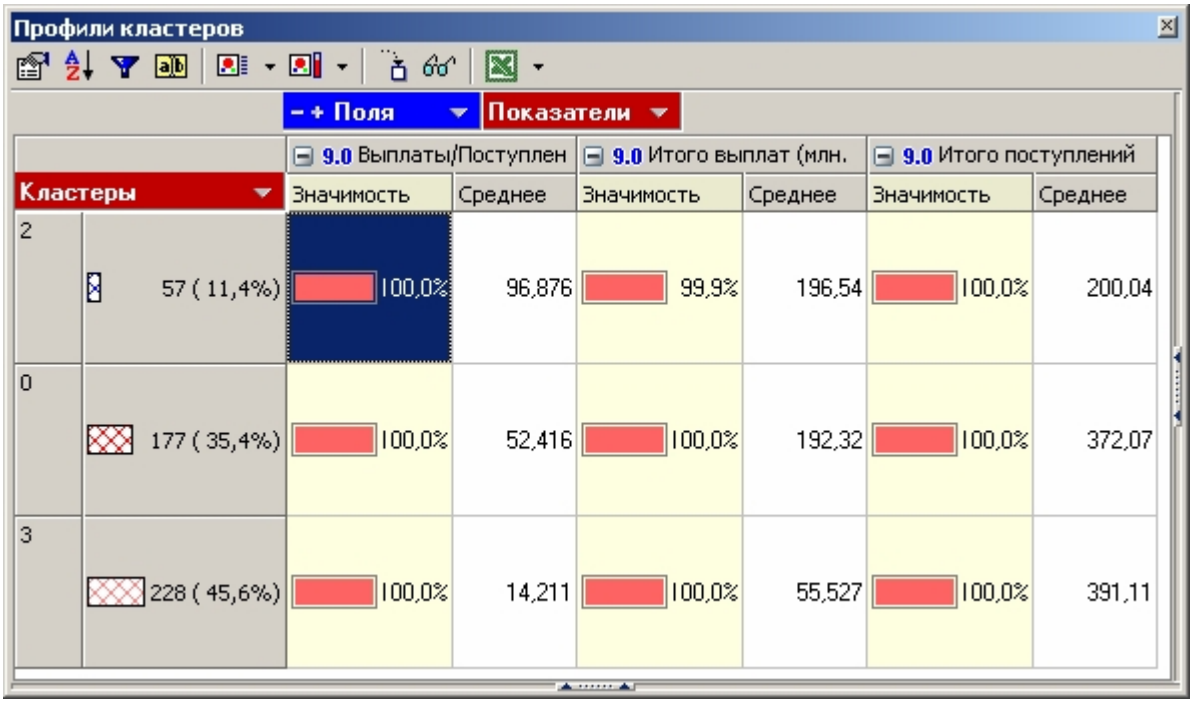
№	Поле	Значение
1	9.0 №	6
2	ab Компания	АЛЬФАСТРАХОВАНИЕ
3	9.0 Итого поступлений (млн. руб.)	16999,09
4	9.0 Итого выплат (млн. руб.)	7004,12
5	9.0 Выплаты/Поступления (%)	41,2
6	12 Номер кластера	1
7	9.0 Расстояние до центра кластера	0,168130314847465

Теперь обратимся к «Профилям кластеров». Это кросс-таблица с двумя измерениями – *Кластеры* и *Поля*. На их пересечении отображаются следующие показатели.

Показатель	Пример	Описание
Значимость		1 минус вероятность нулевой гипотезы. Значимость выражается в процентах. Для непрерывных полей используется *t*-критерий Стьюдента, а для дискретных полей – критерий хи-квадрат. Общая значимость поля определяется по F-критерию Фишера.
Доверительный интервал		Графическое изображение 95% доверительного интервала для среднего значения кластера (темно-серая область). Кроме этого, показываются: ▪ среднее значение по кластеру – красной линией; ▪ среднее значение по всей выборке – синей штрихпунктирной линией.
Среднее	–	Среднее значение по полю, рассчитанное для объектов, попавших в кластер.
Стандартное отклонение	–	Стандартное отклонение по полю, рассчитанное для объектов, попавших в кластер.
Стандартная ошибка	–	Стандартная ошибка по полю, рассчитанная для объектов, попавших в кластер.

Настроим сортировку в порядке убывания поддержки (или мощности кластеров), в показателях оставим только значимость и среднее (см. рисунок). Проинтерпретируем кластеры.





В кластер № 4 попали 2 компании.

№	Компания	Итого поступлений (млн. руб.)	Итого выплат (млн. руб.)	Выплаты/Поступления (%)
261	КАПИТАЛЪ СТРАХОВАНИЕ	151,68	791,75	522
374	ПРОГРЕСС-НЕВА	58,87	335,52	569,92

Это убыточные компании, у которых выплаты по страховым случаям в 5 и более раз превышают поступления, поэтому поле *Выплаты/Поступления* оказалось самым значимым при образовании этого кластера.

В кластере № 5 обнаруживаются пять компаний – безусловные лидеры рынка, на долю которых приходится значительная часть оборота всех страховых услуг страны. Средняя сумма поступлений там составила более 30 000 млн. руб. в год, причем на страховые выплаты приходится около половины из них.

№	Компания	Итого поступлений (млн. руб.)	Итого выплат (млн. руб.)	Выплаты/Поступления (%)
1	ИНГОССТРАХ	42146,18	21778,4	51,67
2	СОГАЗ	38406,03	17555,03	45,71
3	РЕСО-ГАРАНТИЯ	30132,07	15775,67	52,36
4	ВОЕННО-СТРАХОВАЯ КОМ	21984,48	10958,29	49,85
5	РОСНО	21886,15	11400,04	52,09

В следующие два кластера № 1 и 6 попали по 13 и 18 компаний соответственно. Это тоже довольно крупные компании, однако их годовой оборот значительно ниже – примерно 11 000 и 5 000 млн. руб. Рентабельность страховых операций также находится в районе 50%.

В кластере № 0 сосредоточились средние и мелкие, но устойчивые страховые компании со средней рентабельностью, близкой к 50% (177 компаний). А в кластере № 2, наоборот, наблюдаем компании с очень низким уровнем рентабельности и порой убытками (57 компаний). Наконец, в кластере № 3 сгруппировались тоже, как и в кластере № 0 , средние и мелкие компании, но с высокой, порой даже очень высокой рентабельностью страховых операций.

Любому из кластеров можно задать пользовательскую метку (**Кластеры ► Переименовать кластеры**).

Ассоциативные правила

Ассоциативные правила позволяют находить закономерности между связанными событиями. Примером такого правила служит утверждение, что покупатель, приобретающий 'Хлеб', приобретет и 'Молоко' с вероятностью 75%. Впервые эта задача была предложена для поиска ассоциативных правил для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis).

Транзакция – это множество событий, произошедших одновременно. Пусть имеется база данных, состоящая из покупательских транзакций. Каждая транзакция – это набор товаров, купленных покупателем за один визит. Такую транзакцию еще называют рыночной корзиной.

После определения понятия транзакция можно перейти к определению ассоциативного правила. Пусть имеется список транзакций. Необходимо найти закономерности между этими событиями. Как в условии, так и следствии правила должны находиться элементы транзакций.

Пусть $I = \{i_1, i_2, \ldots, i_n\}$ – множество элементов, входящих в транзакцию. D – множество транзакций.

Ассоциативным правилом называется импликация $X \Rightarrow Y$ (читается «X дает Y» или «из X следует Y»), где $X \subset I$, $Y \subset I$ и $X \cap Y = \varnothing$.

Правило $X \Rightarrow Y$ имеет поддержку s (support), если s% транзакций из D содержат $X \cup Y$, $\text{supp}(X \Rightarrow Y) = supp(X \cup Y)$.

Достоверность правила показывает, какова вероятность того, что из X следует Y. Правило $X \Rightarrow Y$ справедливо с достоверностью (confidence) c, если c% транзакций из D, содержащих X, также содержат Y, $conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$.

Лифт – это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом: $lift(X \Rightarrow Y) = conf(X \Rightarrow Y)/supp(Y)$. Значения лифта, большие единицы, показывают, что условие появляется более часто в транзакциях, содержащих и следствие, чем в остальных.

Покажем на конкретном примере:

75% транзакций, содержащих хлеб, также содержат молоко. 3% от общего числа всех транзакций содержат оба товара сразу.

75% – это достоверность (confidence) правила, 3% это поддержка (support) или

Если *Хлеб*, то *Молоко* с вероятностью 75%.

Другими словами, целью анализа является установление следующих зависимостей: если в транзакции встретился некоторый набор элементов X, то на основании этого можно сделать вывод о том, что другой набор элементов Y также должен появиться в этой транзакции. Установление таких зависимостей дает нам возможность находить очень простые и интуитивно понятные правила.

Алгоритмы поиска ассоциативных правил предназначены для нахождения всех правил $X \Rightarrow Y$, причем поддержка и достоверность этих правил должны быть выше некоторых заранее определенных порогов, называемых, соответственно минимальной поддержкой (minsupport) и минимальной достоверностью (minconfidence). Аналогично, поддержка и достоверность ограничиваются сверху порогами максимальной поддержки (maxsupport) и максимальной достоверности (maxconfidence). В результате получаются два окна, в которые должны попасть поддержка и достоверность правила, чтобы оно было предъявлено аналитику.

Значения для параметров минимальная (максимальная) поддержка и минимальная (максимальная) достоверность выбираются таким образом, чтобы ограничить количество найденных правил. Если поддержка имеет большое значение, то алгоритмы будут находить правила, хорошо известные аналитикам или настолько очевидные, что нет никакого смысла проводить такой анализ. С другой стороны, низкое значение поддержки ведет к генерации огромного количества правил, что, конечно, требует существенных вычислительных ресурсов. Большинство интересных правил находится именно при низком значении порога поддержки, хотя слишком низкое значение поддержки ведет к генерации статистически необоснованных правил. Ассоциативные правила с высокой поддержкой могут применяться для формализации хорошо известных правил, например, в автоматизированных системах для управления процессами или персоналом. Надо отметить, что понятия «высокая» и «низкая» поддержка или достоверность очень сильно зависят от предметной области. Например, в торговле 1% вероятности совместного приобретения хлеба и молока не значит ничего, в то время как вероятность в 1% отказа двигателя самолета совершенно неприемлема, и такое правило становится чрезвычайно важным.

Поиск ассоциативных правил совсем не тривиальная задача, как может показаться на первый взгляд. Одна из проблем – алгоритмическая сложность при нахождении часто встречающих

наборов элементов, т.к. с ростом числа элементов экспоненциально растет число потенциальных наборов элементов.

Обычные ассоциативные правила – это правила, в которых как в условии, так и в следствии присутствуют только элементы транзакций и при вычислении которых используется только информация о том, присутствует ли элемент в транзакции или нет. Фактически все приведенные выше примеры относятся к обычным ассоциативным правилам.

Для поиска обычных ассоциативных правил в программе служит обработчик «Ассоциативные правила».

Настройки

Для начала необходимо указать, что является идентификатором (ID) транзакции, а что – элементом транзакции. Например, идентификатор транзакции – это номер чека или код накладной. А элемент – это наименование товара в чеке или накладной.

Затем следует настройка параметров поиска правил. Всего четыре параметра:

- *Минимальная и максимальная поддержка*. Ассоциативные правила ищутся только в некотором множестве всех транзакций. Для того чтобы транзакция вошла в это множество, она должна встретиться в исходной выборке количество раз, больше минимальной поддержки и меньше максимальной. Например, минимальная поддержка равна 1%, а максимальная – 20%. Количество элементов «Хлеб» и «Молоко» столбца «Товар» с одинаковым значением столбца «Номер чека» встречаются в 5% всех транзакций (номеров чека). Тогда эти две строки войдут в искомое множество.
- *Минимальная и максимальная достоверность*. Это процентное отношение количества транзакций, содержащих все элементы, которые входят в правило, к количеству транзакций, содержащих элементы, которые входят в условие. Если транзакция – это заказ, а элемент – товар, то достоверность характеризует, насколько часто покупаются товары, входящие в следствие, если заказ содержит товары, вошедшие во всё правило.

Пример

Транзакция (номер чека)	Элемент (товар)
1	Булочка
4	Спички
2	Сигареты
2	Зажигалка
3	Молоко
3	Кефир
3	Булочка
1	Кефир
4	Сигареты

Дана таблица транзакций (см. выше). Список транзакций будет выглядеть так:

Номер транзакции	Элементы, вошедшие в транзакцию
1	Булочка, Кефир
2	Сигареты, Зажигалка
3	Молоко, Кефир, Булочка
4	Спички, Сигареты

Всего исходные данные состоят из 4 транзакций. Полный список множеств для поиска правил выглядит так:

Множество элементов	Встречается раз в списке транзакций (количество)	Встречается раз в списке транзакций (%)
Булочка	2	50
Кефир	2	50
Сигареты	2	50
Зажигалка	1	25
Молоко	1	25
Спички	1	25
Булочка и кефир	2	50
Сигареты и зажигалка	1	25
Молоко и кефир	1	25
Молоко и булочка	1	25
Спички и сигареты	1	25

Если установить минимальную поддержку 30%, а максимальную – 60%, то останется только часть списка множеств, так называемые *часто встречающиеся множества*:

Множество элементов	Встречается раз в списке транзакций (количество)	Встречается раз в списке транзакций (%)
Булочка	2	50
Кефир	2	50
Сигареты	2	50
Булочка и кефир	2	50

Правила будут искаться именно в этом последнем списке часто встречающихся множеств. Первые три множества в таблице одноэлементные, а последнее – двухэлементное. Ассоциативное правило можно построить только на основе 2-х и более элементного множества.

Соответственно, если будут найдены только одноэлементные множества, то количество ассоциативных правил будет равно нулю. В этом случае следует уменьшить минимальную поддержку и/или увеличить максимальную. Тогда список множеств будет увеличен и, возможно, в него попадут двух и более элементные множества.

Выявление действительно интересных правил – это одна из главных подзадач при вычислении ассоциативных зависимостей. Для того чтобы получить действительно интересные зависимости, нужно разобраться с несколькими эмпирическими правилами:

- Уменьшение минимальной поддержки приводит к тому, что увеличивается количество потенциально интересных правил, однако это требует существенных вычислительных ресурсов. Одним из ограничений уменьшения порога минимальной поддержки является то, что слишком маленькая поддержка правила делает его статистически необоснованным.
- Уменьшение порога достоверности также приводит к увеличению количества правил. Значение минимальной достоверности также не должно быть слишком маленьким, так как ценность правила с достоверностью 5% чаще всего настолько мала, что это и правилом считать нельзя.
- Правило со слишком большой поддержкой с точки зрения статистики представляет собой большую ценность, но, с практической точки зрения, это, скорее всего, означает то, что либо правило всем известно либо товары, присутствующие в нем, являются лидерами продаж, откуда следует их низкая практическая ценность.
- Правило со слишком большой достоверностью практической ценности в контексте решаемой задачи не имеет, т.к. товары, входящие в следствие, покупатель, скорее всего, уже купил.

Если значение верхнего предела поддержки имеет слишком большое значение, то в обнаруженных правилах основную часть будут составлять товары – лидеры продаж. При таком раскладе не представляется возможным уменьшить минимальный порог поддержки до того значения, при котором могут появляться интересные правила. Причиной тому является просто огромное число правил и, как следствие, нехватка системных ресурсов. Причем получаемые правила процентов на 95 содержат товары – лидеры продаж.

Варьируя верхним и нижним пределами поддержки, можно избавиться от очевидных и неинтересных закономерностей. Как следствие, правила, генерируемые алгоритмом, принимают приближенный к реальности вид.

При большом ассортименте товара важно отобразить построенные правила в удобном виде. Для этого в Deductor Studio служат четыре визуализатора: «Правила», «Популярные наборы», «Дерево правил» и «Что-если».

Визуализатор «Правила» – это таблица, в которой отражены номера правил, а так же условия и следствия входящие в него.

Вернемся к примеру.

Правил: 63 из 63 | Фильтр: Без фильтрации

№	Номер правила	Условие	Следствие	Поддержка		Достоверность	Лифт
				Кол-во	%		
1	60	Клей - ж. гвозди Шпатлёвка	Герметики Пена монтажная	2	4.55	40.00	2.933
2	57	Герметики Пена монтажная	Клей - ж. гвозди Шпатлёвка	2	4.55	33.33	2.933
3	59	Герметики Шпатлёвка	Клей - ж. гвозди Пена монтажная	2	4.55	40.00	2.514

В нем правилу с номером 60 в условии присутствуют два элемента: *Клей-ж. гвозди* и *Шпатлёвка*. Это правило показывает, что человек, купивший *Клей-ж.гвозди* и *Шпатлёвка* с вероятностью 40% купит ещё и *Герметики* и *Пену монтажная*.

Визуализатор «Популярные наборы» – таблица, в которой представлены часто встречающиеся предметные наборы с поддержкой больше либо равной заданного порога.

Множеств: 32 из 32 | Фильтр: Без фильтрации

№	Номер множества	Элементы	Поддержка		Мощность
			Кол-во	%	
1	1	Герметики	14	31.82	1
2	7	Герметики Клей - ж. гвозди	10	22.73	2
3	21	Герметики Клей - ж. гвозди Пена монтажная	4	9.09	3

Например, элемент *Герметики* содержится в 31,82% транзакций, а двухпредметный набор *Герметики* и *Клей-ж. гвозди* в 22,73%. Мощность показывает количество элементов в наборе.

Визуализатор «Дерево правил» – это всегда двухуровневое дерево. Оно может быть построено либо по условию, либо по следствию. При построении дерева правил по условию на первом (верхнем) уровне находятся узлы с условиями, а на втором уровне – узлы со следствием.

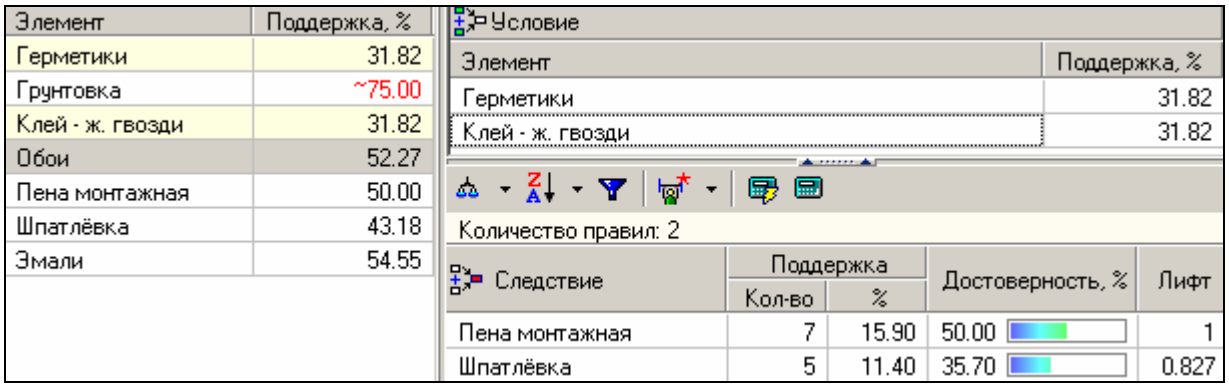
В примере узлы *Герметики*, *Клей-ж. гвозди*, *Пена монтажная* находятся на верхнем уровне дерева и представляют собой условия. А *Герметики*, *Пена монтажная*, *Шпатлёвка* и т.д. – следствия. Это означает, что человек, купивший *Клей-ж. гвозди*, купит еще и *Герметик* с достоверностью 71,40%, пену монтажную с достоверностью 50,00% и т.д. В окне слева расположен список со следствиями для конкретного узла с условием. Для каждого следствия указана поддержка, достоверность и лифт. Например, в исходной выборке данных герметики встретились в 10 транзакциях (чеках).

Второй вариант дерева правил – дерево, построенное по следствию. Здесь на первом уровне располагаются узлы со следствием.

Например, для того чтобы человек приобрел *Герметик*, он должен купить хотя бы один предмет из следующего списка: *Клей-ж.гвозди*, *Пена монтажная*, *Шпатлёвка* и т.д. И для каждого из этих правил отображены поддержка, достоверность и лифт.

Чтобы перестраивать дерево по условию или по следствию служат две кнопки на панели инструментов: **Группировать по условию** и **Группировать по следствию**.

Наиболее удобным и оперативным инструментом использования ассоциативных правил является анализ «Что-если». Внешний вид формы для проведения такого анализа представлен на рисунке.



Слева находится список всех элементов транзакций, то есть весь ассортимент товара, который фигурирует в часто встречающихся множествах. Красным цветом выделены элементы поддержка для которых превысила заданное пороговое значение. Для каждого элемента указана поддержка – количество транзакций (чеков), в которых встречался данный элемент. Предположим, что клиент купил Герметики и *Клей-ж. гвозди*. Двойным щелчком мыши по элементу он переносится в список условий. Используя ассоциативные правила, можно предложить этому человеку сопутствующий товар, который приобретался совместно с тем, что он заказал. Этот товар отображается в списке следствий в правом нижнем окне, т.е. этому человеку можно предложить купить еще и пену монтажную или шпатлёвку. Список следствий можно отсортировать по следствию, поддержке или достоверности либо отфильтровать, оставив в нем часть следствий. Для вычисления следствий по условиям служит кнопка **Обновить правила** на панели инструментов списка следствий. В связи с тем, что список элементов может быть очень большим, для быстрого поиска нужного элемента можно отсортировать список или воспользоваться поиском. Для этого нужно воспользоваться кнопками **Порядок сортировки** (позволяет провести сортировку по порядку, следствию, поддержке, достоверности и по значению лифта) и **Направление сортировки** (позволяет провести

сортировку по возрастанию и убыванию параметра). Для нахождения лучшего правила можно воспользоваться кнопкой **Тип определения лучшего правила**.

Для последующей обработки в сценарии правил, полученных после обработчика «Ассоциативные правила» становится доступным для использования так называемый *зависимый* обработчик «Правила» (см. раздел «Зависимые обработчики»).

Номер правила	Номер элемента	Условие	Следствие	Поддержка		Достоверность	Лифт
				Кол-во	%		
57	1	Герметики	Клей - ж. гвозди	2	4.55	33.33	2.93
57	2	Пена монтажная	Шпатлёвка	2	4.55	33.33	2.93
60	1	Клей - ж. гвозди	Герметики	2	4.55	40	2.93
60	2	Шпатлёвка	Пена монтажная	2	4.55	40	2.93
58	1	Клей - ж. гвозди	Герметики	2	4.55	28.57	2.51
58	2	Пена монтажная	Шпатлёвка	2	4.55	28.57	2.51
59	1	Герметики	Клей - ж. гвозди	2	4.55	40	2.51
59	2	Шпатлёвка	Пена монтажная	2	4.55	40	2.51
1	1	Герметики	Клей - ж. гвозди	10	22.73	71.43	2.24

В примере правилу 57 соответствует два условия – *Герметики* и *Пена монтажная*, а следствием для этого правила являются *Клей-ж. гвозди* и *Шпатлёвка*.

Декомпозиция

Одна из основных целей анализа временных рядов – это возможность построения прогноза. Наиболее простой метод прогнозирования значения ряда – это экстраполяция. Однако она возможна лишь при достаточной закономерности развития изучаемого явления. Реальные экономические процессы редко в достаточной степени удовлетворяют этому требованию. Поэтому исходный временной ряд представляют как совокупность нескольких компонент, к каждой из которых применяется свой метод прогнозирования в соответствии с тенденциями установленными в прошлом.

Целью декомпозиции временного ряда является выделение и изучение сезонной составляющей и тренда. Известно, что функцию исходного ряда можно представить в следующем виде:

$$y(t) = x(t) + s(t) + z(t),$$

где $x(t)$ – тренд, явно выраженная тенденция изменения значений временного ряда,

$s(t)$ – сезонная составляющая, периодически повторяющаяся компонента временного ряда,

$z(t)$ – остаток, нерегулярная (флуктуационная) составляющая временного ряда.

Выводы, полученные в ходе анализа временного ряда, можно использовать в качестве основного материала для определения общей динамики, прогнозирования. Выделение тренда и сезонной компоненты позволяет более точно оценить поведение процесса в будущем.

Для декомпозиции временного ряда на составляющие и исследования предназначен обработчик Deductor под таким же названием.

Настройка параметров декомпозиции временного ряда

Декомпозиция временного ряда может проводиться только на непрерывных упорядоченных данных, иначе полученный результат будет некорректным. Тип сезонности выбирается аналитиком на основе имеющегося временного ряда (если отчетные периоды указаны в месяцах, то нельзя выявить недельную сезонность).

При выявлении тренда предлагаются следующие модели: линейная, квадратичная, кубическая, степенная, логарифмическая, и экспоненциальная.

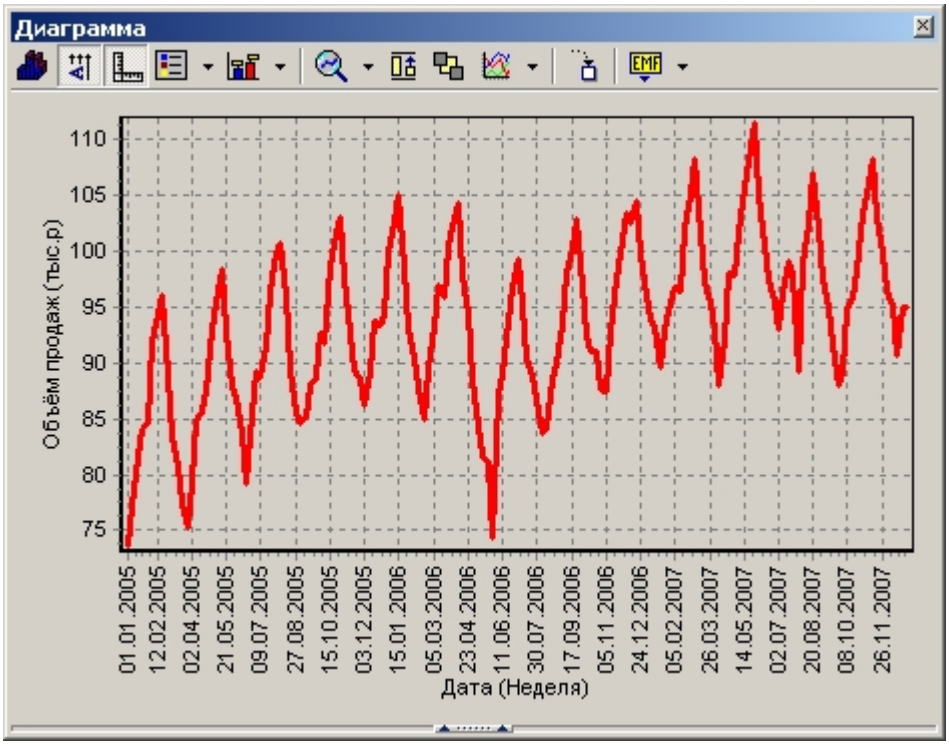
Выбор модели осуществляется на основе оценки средней относительной ошибки. По умолчанию предлагается линейная модель тренда.

Пример

Пусть имеются помесячные данные продаж в розничной торговой сети. Ниже показан элемент исходной таблицы.

Дата (неделя)	Объём продаж (тыс. руб.)
01.01.2005	73,637
08.01.2005	77,136
15.01.2005	81,481
…	…

Импортируем набор данных в Deductor и отобразим зависимость объёма продаж от даты в визуализаторе «Диаграмма».

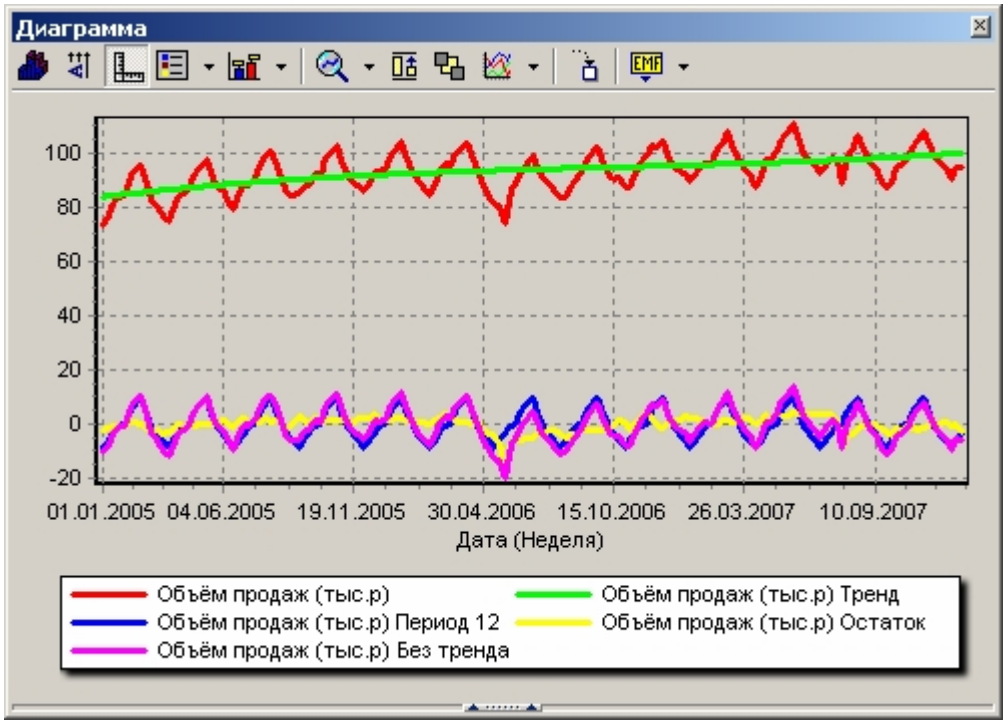


Из графика видно, что продажи имеют явно выраженные квартальные сезонные колебания и восходящий тренд.

Выделим с помощью обработчика «Декомпозиция временного ряда» сезонную, трендовую и нерегулярную компоненты.

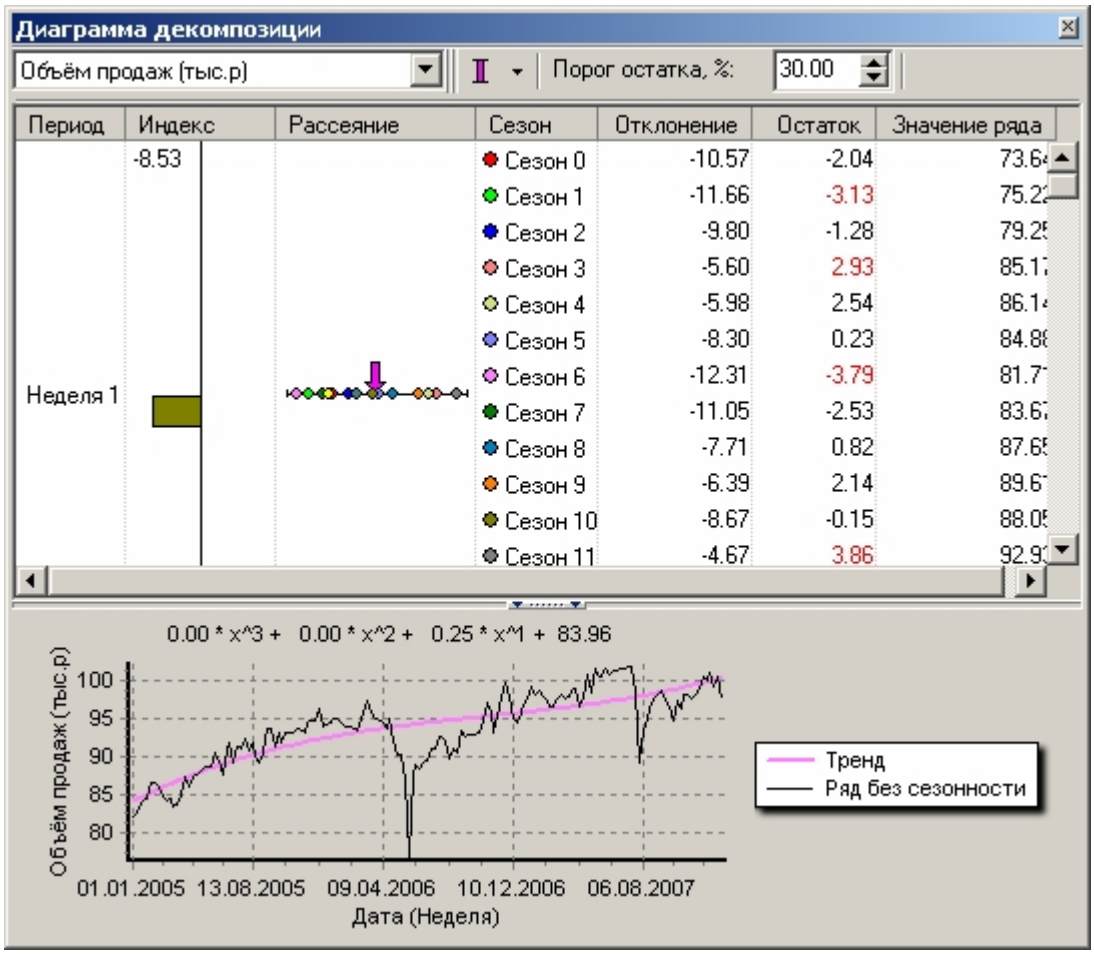
В настройках мастера обработки тип сезонности укажем равным 12 неделям. Из предложенных мастером моделей тренда выберем кубическую, так как для данного ряда она даёт наименьшую ошибку.

Результаты анализа удобно оценивать с помощью двух визуализаторов: «Диаграмма» и «Диаграмма декомпозиции».



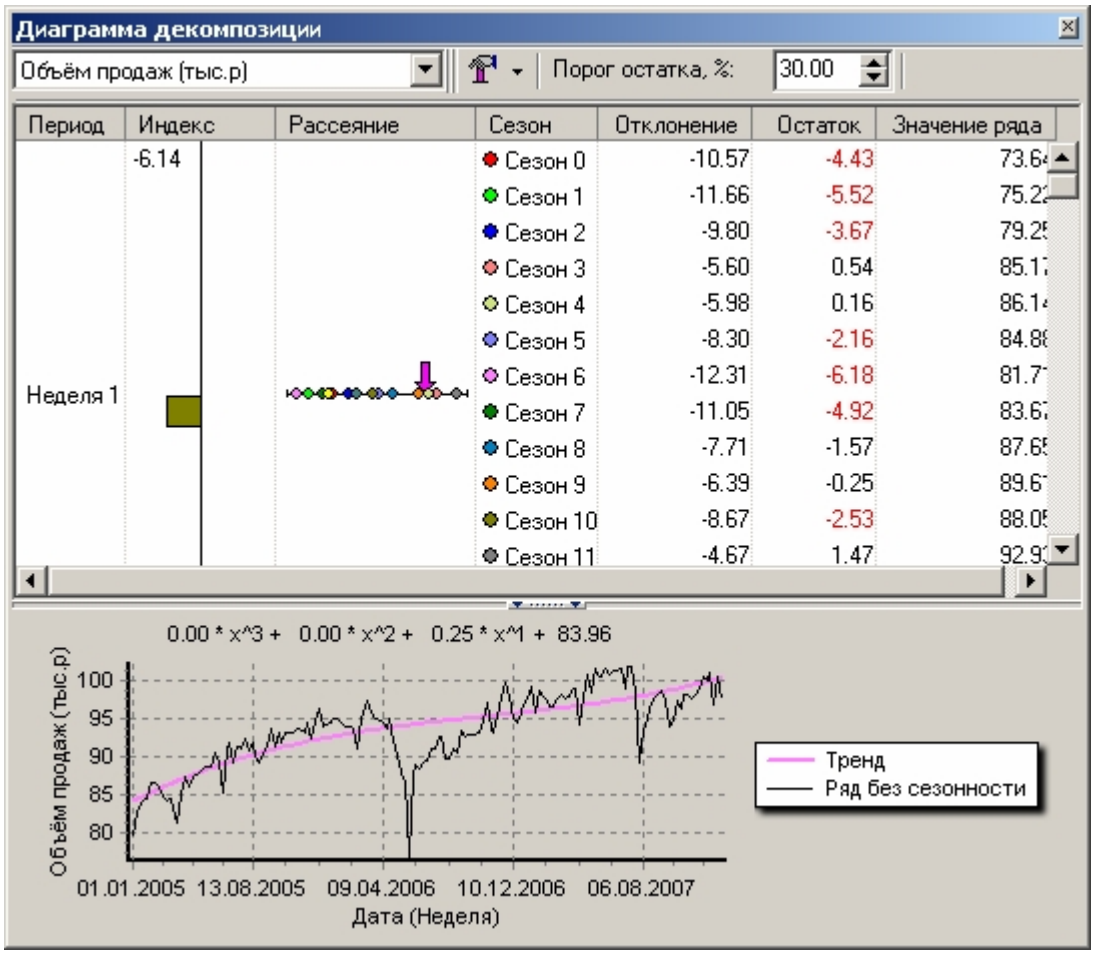
Исключение из исходного ряда тренда и сезонности позволяет получить более гладкую зависимость – остаток (желтая линия на диаграмме). По этой линии можно оценить влияние на исходный ряд пропусков, аномальных значений.

Если результаты анализа не дают желаемого результата, то при помощи корректировки сезонных индексов и модели тренда в визуализаторе «Диаграмма декомпозиции» можно изменить алгоритм обработки временного ряда.



Сезонные индексы показывают, насколько продажи различаются в разное время года. Их можно изменить, тем самым отредактировав функцию сезонных колебаний. Так же здесь можно применить другую модель для формирования тренда или подкорректировать коэффициенты в уже имеющейся.

Одним из методов изменения сезонных индексов является перемещение стрелки в поле *Рассеяние*. Например, увеличим значение индекса за первый период. Результирующая таблица представлена ниже.



Остаток – величина, показывающая разность между значением отклонения и сезонным индексом в определённом временном интервале. В столбце *Остаток* красным цветом выделяются значения, превышающие значение сезонного индекса на величину большую порога остатка, соответственно, при изменении значения сезонного индекса изменяется и количество значений, выделенных красным.

На полученных в результате декомпозиции данных может быть построен прогноз.

Пользовательские модели

В процессе анализа данных встречаются такие ситуации, когда сложные модели, основанные на линейной регрессии и нейронных сетях, не могут описать предметную область. Например, в тех случаях, когда объем исходной выборки мал либо ее качество недостаточно для того, чтобы обучить нейронную сеть. Кроме того, иногда заранее известны модели некоторых процессов, т.е. описывающие их формулы и оценки коэффициентов. Например, в задачах *оценки рисков* существуют несколько хорошо известных моделей, в которые эксперту требуется лишь подставить коэффициенты, описывающие конкретную ситуацию. Для того чтобы дать аналитику возможность самому строить модели, основанные на экспертных оценках, в Deductor включен обработчик «Пользовательские модели».

Пользовательская модель позволяет на основании исходных данных и формул, указанных аналитиком, построить произвольную модель и работать с ней так же, как с моделями на основе, скажем, нейросетей. Примерами такой модели может служить модель прогноза на основе авторегрессии с коэффициентами, задаваемыми экспертом либо прогноз на основании скользящего среднего.

При создании пользовательской модели нужно пройти, в основном, те же шаги, что и при настройке других моделей Deductor.

Нормализация значений полей

Для полей, подаваемых на входы и выход дерева решений, настройки нормализации недоступны. Можно лишь посмотреть параметры нормализации, используемые по умолчанию, и гистограмму распределения выборки (описание в разделе по нейросети).

Задание модели

Задание формулы, по которой рассчитывается модель, производится в окне Калькулятора. В Deductor предусмотрены несколько стандартных моделей, включая скользящее среднее, авторегрессию, ARMA, ARIMA и др., которые можно использовать на этом этапе. Для этих моделей достаточно указать необходимые коэффициенты. Для того чтобы ввести в Калькуляторе формулу одной из стандартных моделей, следует с помощью кнопки **Функция** вызвать окно выбора функции. В разделе «Модели» будут представлены стандартные модели с описанием применения, формул и требуемых коэффициентов. Здесь нужно выделить название нужной модели и добавить ее в окно Калькулятора, нажав кнопку **Ok**. Теперь в формулу модели следует добавить нужные коэффициенты, и на этом создание модели закончено.

В списке доступных функций имеется большой набор стандартных математических, статистических, строковых и прочих функций. Любую из этих функций, а также стандартные математические операции можно использовать при построении пользовательской модели.

В окне Калькулятор можно задавать формулы не только для полей, назначение у которых является выходным, но и добавлять новые поля. Разница состоит в том, что для выходных полей будут доступны визуализаторы группы Data Mining, предназначенные для оценки качества модели, – диаграмма рассеяния и таблица сопряженности. Создание модели с использованием выходных полей аналогично обучению нейросети с учителем, есть эталонные значения выходов и есть выходы, рассчитанные моделью. Чем они ближе, тем лучше модель описывает исходную выборку данных. Диаграмма рассеяния и таблица сопряженности позволяют оценить степень близости эталонных и рассчитанных выходов, а, следовательно, качество модели. При добавлении же новых полей эталонных выходов нет, есть только рассчитанные моделью, и оценка качества модели с помощью подобных инструментов невозможна.

Визуализация

Для пользовательских моделей доступны три вида визуализаторов из группы Data Mining: диаграмма рассеяния, «Что-если» и таблица сопряженности. Первый из них позволяет оценить качество построенной модели по тому, насколько точно она описывает имеющиеся данные, если выходное полу является непрерывным. Второй дает возможность анализировать модель по принципу «что будет, если» и позволяет исследовать ее поведение при подаче на вход тех или иных данных. С помощью таблицы сопряженности можно сравнить значения дискретных выходных полей, рассчитанные моделью, с выходными значениями полей исходной выборки и определить, насколько точно выходы модели соответствуют эталонным значениям.

Пример

В качестве примера рассмотрим построение прогноза объемов продаж на основании короткого временного ряда. Пусть задан следующий временной ряд объемов продаж товара.

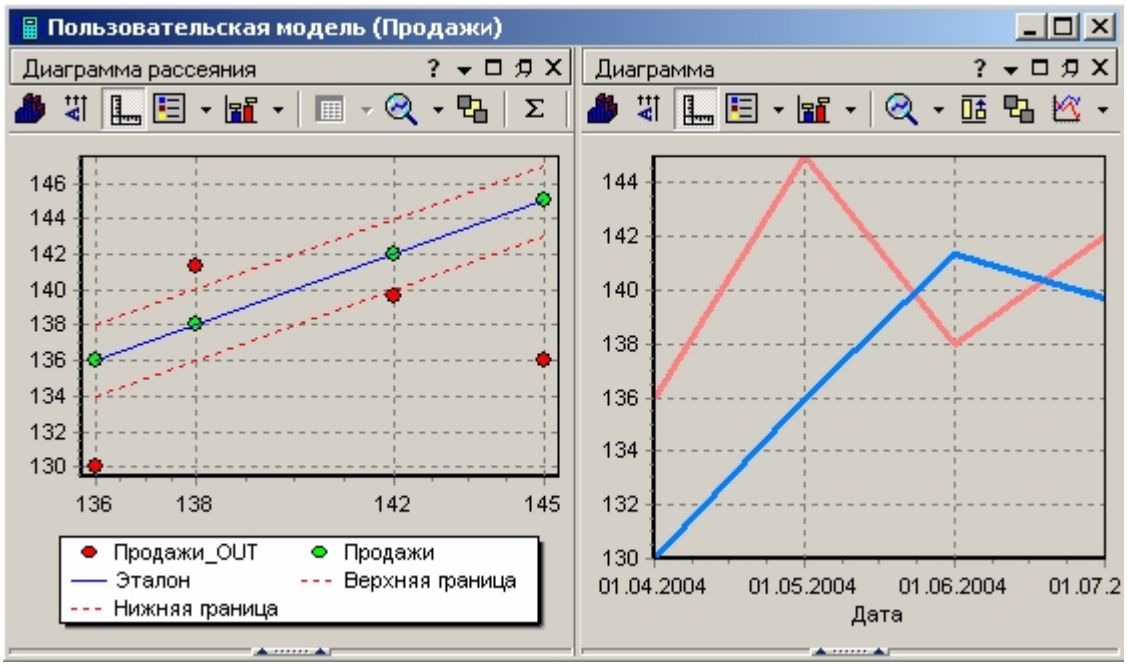
Дата	Продажи
01.01.04	118
01.02.04	129
01.03.04	143
01.04.04	136
01.05.04	145
01.06.04	138
01.07.04	142

Необходимо построить прогноз продаж на следующий месяц. Объем выборки очень мал, и построить качественный прогноз по ней нельзя. Тем не менее, даже не самый лучший прогноз зачастую бывает лучше, чем ничего. Воспользуемся для прогнозирования моделью скользящего среднего. Для начала преобразуем данные скользящим окном, указав глубину погружения 3. Таким образом, мы будем строить прогноз на основании данных за последние три месяца. На этом шаге получим следующие результаты:

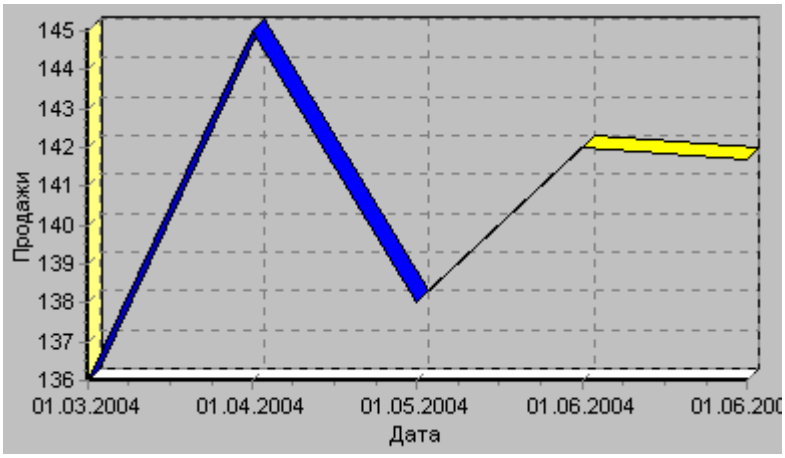
Дата	Продажи-3	Продажи-2	Продажи-1	Продажи
01.03.2004	118	129	143	136
01.04.2004	129	143	136	145
01.05.2004	143	136	145	138
01.06.2004	136	145	138	142

Теперь построим пользовательскую модель. В качестве формулы, по которой будет рассчитываться модель, укажем следующую: *Продажи = MOVINGAVERAGE(COL2B1;COL2B2;COL2B3)*.

Диаграмма рассеяния и диаграмма модели показаны на следующем рисунке.



Как видим, качество построенной модели значительно ниже по сравнению с моделями на основе нейронных сетей или линейной регрессии. Тем не менее, учитывая столь маленький объем выборки, даже такая модель прогноза представляет определенный интерес. Диаграмма прогноза, построенного на один месяц, показана на рисунке ниже.



Прогнозируемое значение объема продаж составило 141,67, т.е. средний объем продаж за последние три месяца.

Вспомогательные методы обработки

Помимо перечисленных выше алгоритмов в программе есть обработчики, которые трудно отнести к очистке, трансформации или методам Data Mining. Но эти механизмы обработки могут быть важной частью при создании сценариев для анализа данных.

Скрипт

Скрипты предназначены для автоматизации процесса добавления в сценарий однотипных ветвей обработки. По сути скрипт представляет собой динамическую копию выбранного участка сценария. Скрипт является готовой моделью, и поэтому входящие в него узлы не могут быть изменены отдельно от исходной ветки сценария. Тем не менее, на скрипте отражаются все изменения, вносимые в ветку, на которую он ссылается, т.е. при переобучении или перенастройке узлов этой ветки все сделанные изменения будут внесены в работу скрипта.

Аналогом скрипта является функция или процедура в языках программирования. Ветвь обработки строится один раз, а затем скриптами выполняются определенные в ней действия.

Для настройки скрипта достаточно указать начальный и конечный узлы, находящиеся на одной ветви обработки. Тогда при выполнении узла скрипта будут последовательно выполнены начальный, все промежуточные узлы между начальным и конечным и конечный узел.

Если исходный набор данный имеет меньшее число полей, чем начальный компонент цепочки, то система выдаст следующее сообщение: «*Количество столбцов начального компонента цепочки не должно быть больше, чем количество столбцов исходного набора данных*». При этом в момент обработки скрипта будет предпринята попытка выполнить с имеющимся набором полей. В случае, если какое-то из отсутствующих полей является критичным для любого узла, содержащегося в скипрте, то обработка будет остановлена с выдачей сообщения об ошибке.

Под исходным набором данных подразумевается тот набор данных, к которому применяется обработчик «Скрипт», начальный компонент цепочки – набор данных, на который настраивается «Скрипт».

Нажав кнопку **Ок**, можно продолжить выполнение данного компонента**.**

В случае, когда исходный набор данных содержит большее количество полей, чем набор данных, являющийся начальным компонентом цепочки, предусмотрена возможность настройки информационных полей. В данном случае под информационными полями понимаются те поля, которые не будут использоваться в модели, но которые будут помещены в результирующий набор данных, полученный после применения рассматриваемого компонента.

Возможна ситуация, когда столбцам начального компонента цепочки нет сопоставимых столбцов в исходном наборе данных. В такой ситуации система выдаст следующее сообщение: «*Столбцам начального компонента цепочки нельзя сопоставить столбцы исходного набора данных*». При этом в момент обработки скрипта будет предпринята попытка выполнить с имеющимся набором полей. В случае, если какое-то из отсутствующих полей является критичным для любого узла, содержащегося в скрипте, то обработка будет остановлена с выдачей сообщения об ошибке.

Столбцы исходного набора данных, по которым не было установлено соответствие полям в скрипте, идентифицируются программой как «Информационные» и могут быть при желании добавлены в выходной набор данных. Необходимо иметь в виду, что эти поля появятся в результирующем наборе только если в скрипте не используются следующие обработчики: группировка, разгруппировка и расчет автокорреляции.

Нажав кнопку **Ok**, можно продолжить выполнение данного компонента**.**

Основное назначение скриптов – применение готовой модели (фрагмента сценария) к различным наборам данных в рамках одного проекта.

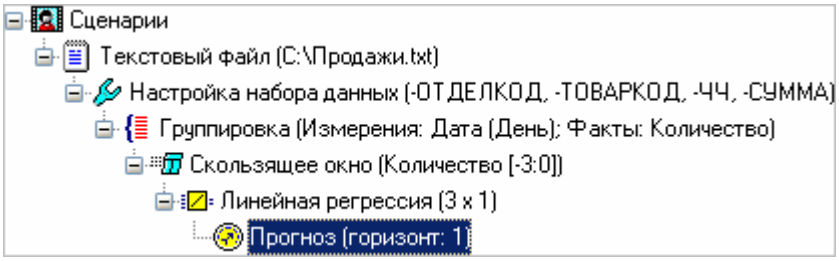
Пример

Пусть даны две таблицы с объемами продаж товара. В первой таблице продажи заданы в разрезе по дням, во второй – по месяцам, причем вторая таблица имеет дополнительное поле «Код товара». Требуется построить прогноз объемов продаж товара на один месяц по обеим таблицам, поскольку неизвестно, какая из них будет подаваться на вход в будущем.

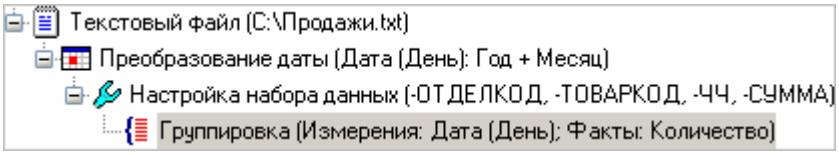
Дата (День)	Количество
17.09.2000	3
17.09.2003	38
18.09.2003	52
19.09.2003	43
20.09.2003	84
21.09.2003	60
22.09.2003	78
23.09.2003	63
24.09.2003	77
25.09.2003	78
…	…

Количество	Дата (Месяц)	Код товара
3	01.09.2000	11698
3384	01.10.2003	10
4915	01.11.2003	10
7301	01.12.2003	10
6228	01.01.2004	10
6497	01.02.2004	10
…	…	…

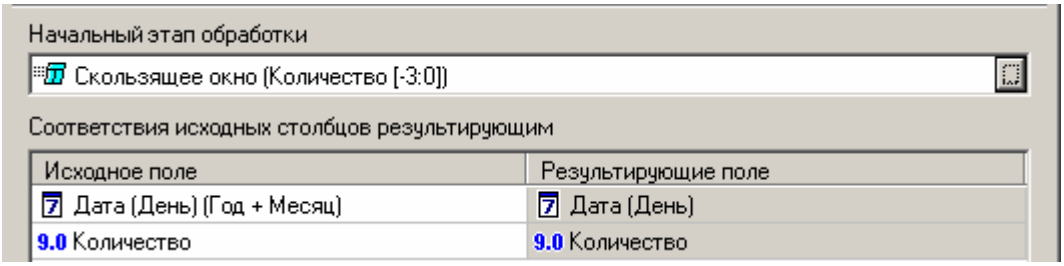
Построим ветвь прогноза объема продаж по таблице с данными в месячном разрезе.



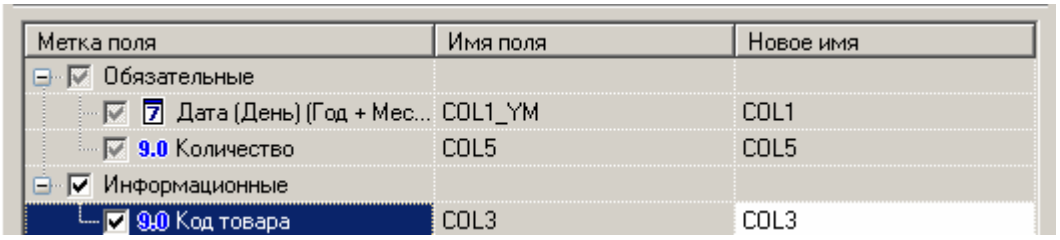
Теперь импортируем данные из второй таблицы и преобразуем дату к представлению по месяцам/



После узла группировки добавим обработчик «Скрипт» со следующими настройками начального узла.

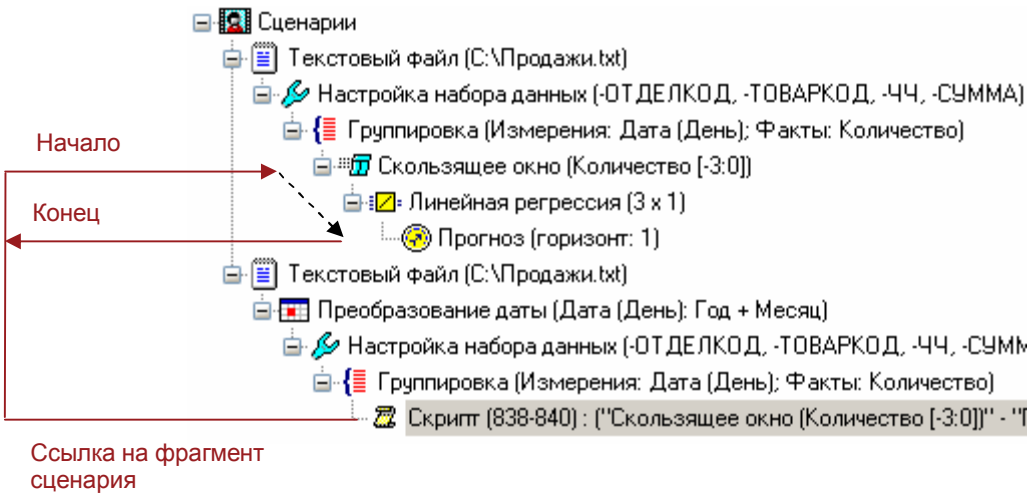


Здесь узел «Скользящее окно» – начальный компонент цепочки, а узел «Группировка» – исходный набор данных. Поскольку набор данных, к которому применяется обработчик «Скрипт», имеет большее количество полей, чем начальный компонент цепочки, то в мастере обработки можно указать, какие поля в качестве информационных будут включены в результирующий набор данных. В качестве информационного в данном случае выступает поле *Код товара*.



Конечным узлом укажем обработчик «Прогноз».

Нам не пришлось строить модель прогноза для второй таблицы, вместо этого мы воспользовались уже готовой моделью, применив ее в другой ветви сценария. Полностью выполненный сценарий имеет ниже представленный вид:



Стрелками на рисунке отображена последовательность выполняемых обработчиков в данном скрипте.

Групповая обработка

Обработчик «Групповая обработка» является аналогом обработчика «Скрипт». Он позволяет выполнять готовую ветвь сценария по отношению не ко всему исходному набору, а к отдельным группам, задаваемыми пользователем. При подготовке его работы указываются начальный и конечный узел сценария и при выполнении «Групповой обработки» первоначальный сценарий выполняется от узла к узлу. При этом необходимо учитывать, что построение результата идёт по заранее заданному алгоритму, а следовательно выполняются только узлы исходного сценария. Изменения, вносимые в первоначальный сценарий, влекут за собой изменения в обработке групп. Все настройки исходных и новых данных такие же как и для обработчика «Скрипт»

Пример

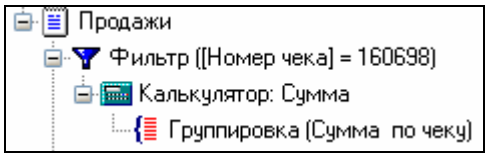
Пусть имеются данные по продажам по чекам.

№ Чека	Товар	Цена	Количество
160698	Обои	150	8
160698	Эмали	45	8
160698	Грунтовка	41	8
160747	Эмали	45	7
160747	Пена монтажная	24	7
160747	Грунтовка	41	7
…	…	…	…

Необходимо рассчитать сумму покупки по каждому чеку. Для этого с помощью обработчика «Фильтр» выделяются элементы таблицы, относящиеся к одному чеку.

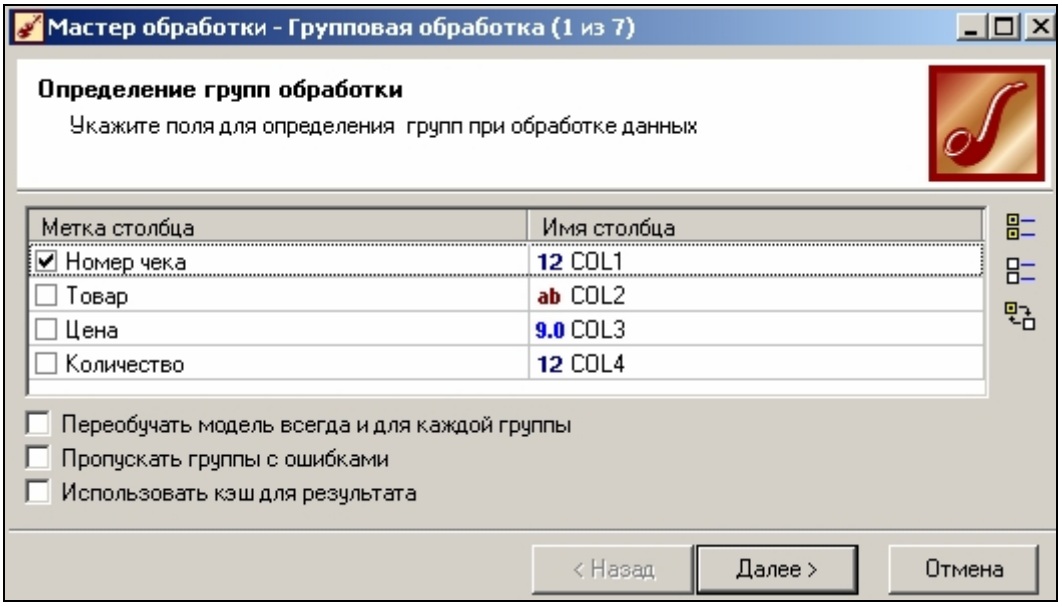
Номер чека	Товар	Цена	Количество
160698	Обои	150	8
160698	Эмали	45	8
160698	Грунтовка	41	8

Затем строится сценарий, при помощи которого вычисляется сумма покупки по чеку.



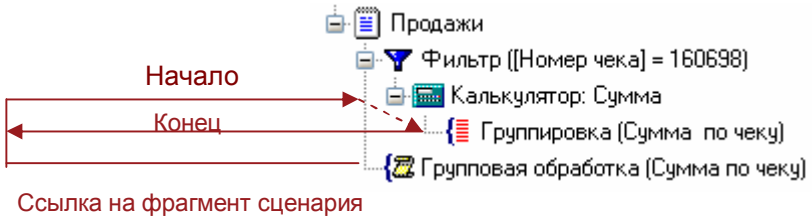
Для того чтобы посчитать сумму по каждому чеку необходимо к узлу «Продажи» добавить узел «Групповая обработка» со следующими настройками:

Агрегация данных будет проводиться по номеру чека



В качестве начального компонента цепочки указывается «Калькулятор», в качестве конечного – «Группировка».

Законченный сценарий имеет следующий вид.



Стрелками на рисунке отображена последовательность выполняемых обработчиков в данной «Групповой обработке»

Калькулятор

С помощью компонента «Калькулятор» в исходную выборку могут быть добавлены поля, значения которых вычисляются по формуле из значений других полей.

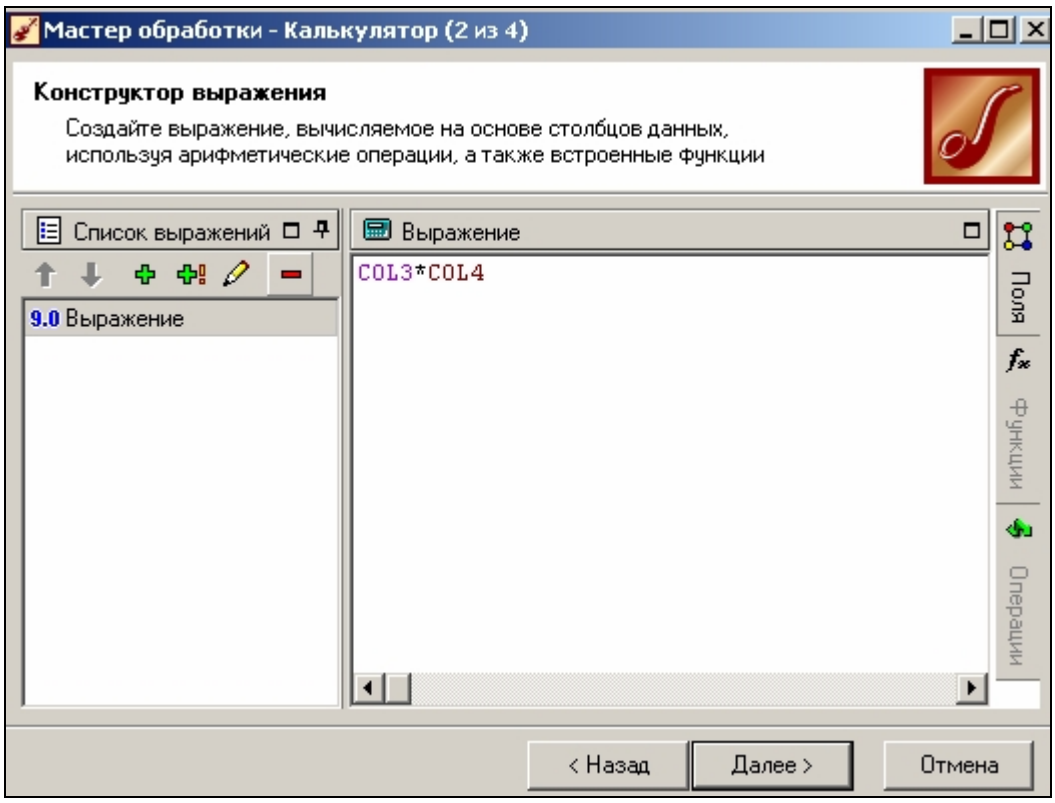
Для создания выражения нужно выполнить следующие действия.

В поле *Название выражения* ввести метку, под которой это выражение будет видно пользователю. По умолчанию для нового выражения назначается метка «Выражение_N», где N – номер, обеспечивающий уникальность. При необходимости можно назначить для выражения (и соответственно, для вычисляемого с его помощью поля) более информативное имя, например, *Сумма*, *Доля* и так далее. Справа от имени выражения необходимо указать формулу, по которой будет рассчитываться это поле. Правила составления выражений соответствуют общим правилам составления выражений в математике, в частности число открывающих скобок должно равняться числу закрывающих.

Выражение может содержать:

- Числа в явном виде;
- Переменные в виде имен столбцов;
- Скобки, определяющие порядок выполнения операций;
- Знаки математических операций и отношений;
- Имена функций;
- Даты в формате «ДД.ММ.ГГ», указываемые в двойных кавычках. Такой способ указания даты может оказаться непереносимым между разными компьютерами, поэтому лучше использовать для этой цели функцию StrToDate.
- Строки, обязательно указываемые в кавычках.

Выражение можно ввести вручную с клавиатуры, однако, удобнее выбирать функции, переменные и знаки операций с помощью мыши. В поле *Выражение* всегда отображается то выражение, которое в данный момент выделено в списке слева. Для его редактирования достаточно щелкнуть в поле мышью, вызвав курсор, а затем редактировать как обычное текстовое поле.



Условие

С помощью компонента «Условие» можно обеспечить ветвление в логике выполнения веток сценария в зависимости от того или иного условия. В зависимости от истинности или ложности выполнения заданного условия ветвь сценария будет либо выполняться дальше либо это выполнение прекратится.

Для создания условия необходимо в мастере обработки указать список условий выполнения ветки сценария.

Параметры условия задаются в виде списка, который содержит следующие столбцы:

- *Операция* – позволяет установить функцию отношения «И»/«ИЛИ» между задаваемыми списками условий выполнения ветви сценария. Построение условия выполнения ветви сценария возможно по нескольким спискам одновременно. Тогда функция в поле «Операция» устанавливает отношение между этими множествами. Если используется отношение «И», то дальнейшее выполнение ветви сценария зависит от выполнения условий по задаваемым спискам. Если используется отношение «ИЛИ», то дальнейшее выполнение ветви сценария зависит от выполнения хотя бы одного из заданных условий. Установка отношений возможна, только если настроены два или более списка условий. Для выбора операции следует дважды щелкнуть левой кнопкой мыши в столбце «Операция» для соответствующего условия и из списка, открываемого кнопкой, выбрать нужную функцию отношения. По умолчанию устанавливается отношение «И».
- *Имя поля* – позволяет выбрать поле, по значениям которого должно быть построено условие выполнения ветви сценария. Для этого нужно дважды щелкнуть в столбце «Имя поля» и с помощью кнопки открыть список полей текущей выборки, указав нужное поле. Одно и то же поле может быть использовано в нескольких условиях.
- *Агрегация* – указывается одна из возможных функций агрегации, которая будет применяться к значениям в столбце, указанном в поле «Имя поля». Возможны следующие функции агрегации:

- Сумма – суммирование значений в указанном столбце.
- Максимум – выбирается максимум из значений в указанном столбце.
- Минимум - выбирается минимум из значений в указанном столбце.
- Среднее - считается среднее арифметическое значений в указанном столбце.

- *Условие* – указывается условие, по которому нужно выполнить проверку для данного поля. Для выбора условия достаточно дважды щелкнуть мышью в соответствующей ячейке и в списке условий, открываемом кнопкой, выделить нужное условие. Доступны следующие условия:
  - *`= (равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), <> (не равно)* – условие принимает значение «истина», если для результата заданной функции агрегации значений столбца, указанного в поле «Имя поля», и заданного в поле «Значение», выполняется одна из указанных операций отношения.
  - *Пустой* – условие принимает значение «истина», когда результат заданной функции агрегации значений в указанном столбце принимает пустое значение. В этом случае поле «Значение» не используется.
  - *Не пустой* – условие принимает значение «истина», когда результат заданной функции агрегации значений в указанном столбце принимает непустое значение. В этом случае поле «Значение» не используется.
  - *Содержит* – условие принимает значение «истина», если результат заданной функции агрегации значений в указанном столбце содержит указанное значение.
  - *Не содержит* – условие принимает значение «истина», если результат заданной функции агрегации значений в указанном столбце не содержит указанное значение.
  - *Начинается на, не начинается на* – для строковых полей; условие принимает значение «истина», когда результат заданной функции агрегации значений в указанном столбце начинается (не начинается) на введенную последовательность символов.
  - *Заканчивается на, не заканчивается на* – для строковых полей; условие принимает значение «истина», когда результат заданной функции агрегации значений в указанном столбце заканчивается (не заканчивается) на введенную последовательность символов.
- *Значение* – указывается значение, по которому будет производиться проверка выполнения условия. Способ ввода значения будет различным в зависимости от типа данных и типа условия. Допустим, в качестве операции отношения выбрана одна из следующих операций сравнения: «=», «<>», «>» и т.д. Если данные в поле являются непрерывными (т.е. числовыми), то достаточно дважды щелкнуть мышью в соответствующей ячейке, чтобы появился курсор, затем ввести значение (число). Если поле, по которому осуществляется проверка выполнения условия, имеет тип «строка» (т.е. является дискретным), то в результате двойного щелчка в столбце «Значение» появится кнопка выбора, которая откроет окно «Список значений», где будут отображены все значения поля и количество упоминаний их в наборе данных. Чтобы выбрать значение для условия отбора, достаточно выделить его и щелкнуть **Ok**, либо просто выполнить двойной щелчок.

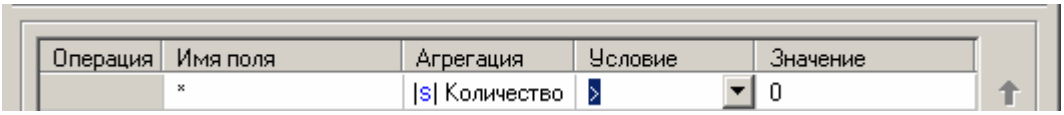
Применение компонента «Условие» может быть полезным в случае, когда в зависимости от выполнения некоторого заранее заданного условия нужно продолжать или не продолжать выполнение ветви сценария. Особенно это важно, когда ветвь содержит узлы, оперирующие с большими массивами данных, и ненужное выполнение этой части ветви может привести к большим затратам аппаратных и временных ресурсов.

Пример

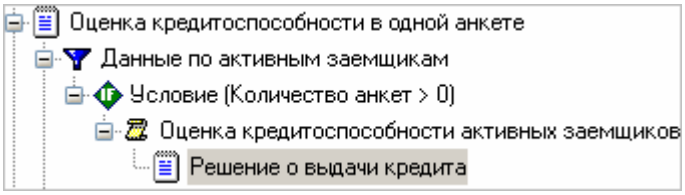
Допустим, необходимо в зависимости от количества анкет выполнять или не выполнять оценку кредитоспособности двух категорий заемщиков: активных заемщиков и пенсионеров. В случае,

когда анкеты по той или иной категории будут отсутствовать, выполнение ветви сценария по оценке кредитоспособности не осуществлять.

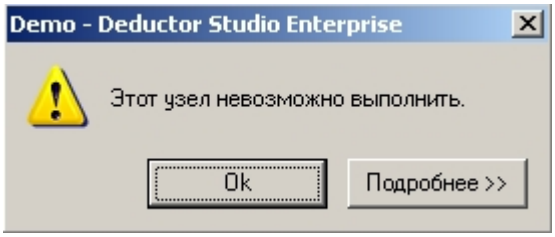
Рассмотрим настойку компонента «Условие». На первом шаге в Мастере обработки необходимо указать список условий дальнейшего выполнения ветки сценария. В столбце «Имя поля» указывается значение «*», в столбце «Агрегация» указываем функцию агрегации, в столбце «Условие» указывается условие, по которому нужно выполнить проверку для данного поля. В данном примере в качестве условия задается операция сравнения «>», поскольку нам необходимо продолжать дальнейшее выполнение сценария, когда количество анкет больше нуля, в столбце «Значение» указываем значение равное нулю. Таким образом, условие будет выглядеть следующим образом:

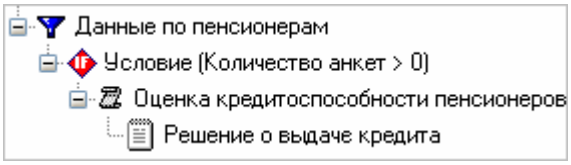


Это означает «выполнять следующие узлы сценария, если количество записей в выборке больше 0». Настройка данного узла закончена. Если заданное условие выполняется, то происходит дальнейшее выполнение ветви сценария:



Если же условие не выполняется, то при попытке вызвать идущий после условия узел, значок обработчика «Условие» в сценарии будет выделен красным цветом и система выдаст следующее сообщение:





Команда ОС

Назначение

С помощью узла «Команда ОС» становится возможным формирование и запуск различных команд операционной системы. Так, например, иногда бывает необходимо запустить то или иное приложение из Deductor Studio, это возможно при использовании компонента «Команда ОС».

Важной особенностью применения данного обработчика является не сам факт выполнения команды операционной системы или запуска приложения, а то, что это действие становится элементом сценария, т.е. встраивается в «конвейер» обработки. Кроме того, т.к. команду можно формировать, используя переменные и любые операции работы со строками, то можно не просто вызывать стороннее приложение, но и передавать ему в качестве параметров сведения из имеющегося набора данных.

Настройка

Настройка данного обработчика включает в себя следующие шаги:

- Выбор источника командной строки. На данном этапе указывается источник получения командной строки. При этом возможны следующие способы задания командной строки:
  - *Ввод команды вручную*. В данном случае команда будет выполнена один раз для всего набора данных.
  - *Команды из набора данных*. В данном случае выполнение команды или запуск внешнего приложения будет осуществляться для каждой строки из набора данных.
- Указать параметры выполнения команды:
  - *Максимальное время ожидания завершения выполнения команды*. Необходимо указать время (время указывается в секундах), в течение которого программа будет ожидать завершение выполнения заданный команды. В случае, когда время ожидания будет превышено, Deductor Studio продолжит свое выполнение, завершив при этом запускаемую команду. Если указать максимальное время ожидания 0 секунд, то Deductor Studio будет бесконечно ожидать завершение вызываемой команды.
  - *Добавление столбца с результатами в выходной набор данных*. При этом в качестве результата в набор данных добавляется код возврата вызываемого приложения. Обычно данный код позволяет судить о корректности выполнения команды. Однако возможна ситуация, когда в качестве кода возврата будет передаваться результат выполнения заданной команды.

Если на предыдущем шаге был выбран способ задания командной строки «Ввод команды вручную», то следующим шагом настройки указанного компонента является задание команды операционной системы, которая и будет выполняться. Например, если в качестве команды задать «Calc», то при выполнении этой команды будет запущен стандартный калькулятор, поставляемый с Windows; если в качестве команды указать выражение «md D:\Directory», то на локальном диске **D** будет создан каталог с именем Directory.

Если на первом шаге в Мастере обработки в качестве способа задания командной строки был выбран «Команды из набора данных», то следующим шагом в Мастере обработки будет конструктор выражения. В данном конструкторе с помощью арифметических операций и встроенных функций можно создать выражение, которое будет вычисляться на основе столбцов данных. При использовании данного способа задания командной строки, формируемую команду необходимо задавать в виде *строкового выражения*. Пример задания команды таким способом выглядит следующим образом:

"C:\Program Files\BaseGroup\Deductor\Bin\DStudio.exe "+" C:\prog_models.ded "+"/Run"

Deductor поддерживает набор параметров командной строки, предназначенных для управления его работой в пакетном режиме. Формат указания параметров следующий:

<путь к Deductor>DStudio.exe [<файл-проекта>] [параметры]

где:

- <путь к Deductor> – путь к каталогу, в который установлен Deductor; по умолчанию: C:\Program Files\BaseGroup\Deductor\Bin\;
- [<файл-проекта>] – имя файла проекта с расширением ***.ded** с полным путем;

- Параметры
  - /HELP или /? – выдать справку по синтаксису командной строки;
  - /SYSFILE=<sys-файл> – загрузить параметры приложения (параметры подключений) из файла <sys-файл>. Если этот параметр не указан, то используются установки, сделанные в окне «Настройка»;
  - /RUN – выполнить сценарии проекта в пакетном режиме.

Режим работы *run* предназначен для исполнения сценария в пакетном режиме. Будет производиться выполнение только узлов, у которых был установлен флаг **Выполнить** в меню **Статус пакетной обработки**. Кроме того, если у узла сброшен этот флаг, то его потомки также исполнены не будут. Пакетное выполнение предназначено для расчета выходных данных построенной модели и их экспорта во внешние приемники данных.

Полный список параметров запуска Deductor Studio можно посмотреть в справочном руководстве к программе.

По окончании пакетного выполнения сценария Deductor Studio возвращает вызвавшему его приложению статус окончания обработки. При корректно выполненном сценарии возвращаемое значение будет 0, а в случае возникновения ошибки – ненулевое. Проверка статуса окончания обработки часто требуется при вызове Deductor из bat-файлов или других приложений.

Примеры использования параметров командной строки

Выполнить сценарии проекта <D:\Мои документы\demo.ded> в пакетном режиме, параметры подключений взять из файла <c:\deductor.sys>:

```
"C:\Program Files\BaseGroup\Deductor\Bin\DStudio.exe" "D:\Мои документы\demo.ded" /RUN
/SYSFILE=c:\deductor.sys
```

Переобучить модели проекта <c:\demo.ded> в пакетном режиме. Подробно фиксировать в журнале регистрации выполнение каждого узла. Для ведения журнала регистрации использовать файл <c:\demo.log>. Добавлять новые записи в конец журнала регистрации:

```
DStudio c:\demo.ded /TEACH /LOG /LOGFILE=c:\demo.log /LOGMODE
```

На этом настройка компонента «Команда ОС» закончена, после чего можно запускать на выполнение заданную команду.

Замечание

> *Возможность работы с обработчиком «Команда ОС» имеется только в Deductor Enterprise.*

Сценарий Deductor

Назначение

Обработчик «Сценарий Deductor» предназначен для вызова из сценария Deductor другого сценария, сделанного тоже в Deductor, в режиме OLE-Automation сервера. В отличие от способа запуска bat-файла с командой пакетного запуска Deductor при помощи узла «Команда ОС», этот способ позволяет выполнять многократный запуск одного и того же сценария с разными параметрами без потерь времени на запуск Deductor и загрузку сценария в ОЗУ.

Настройка

Настройка данного обработчика включает в себя следующие шаги:

- Выбор сценария, списка подключения и файла переменных окружения. На данном этапе указывается источник получения сценария. При этом нужно задать тип его пакетной обработки – *Выполнение* или *Переобучение*. Кроме того, есть возможность указать:
  - *Имя файла подключений*. Файл имеет расширение *.sys. Если он не указан, то будет браться файл по умолчанию, заданный в окне **Сервис ►Настройка►Основные параметры**.
  - *Имя файла параметров окружения*. Файл тоже имеет расширение *.sys. В нем хранятся переменные окружения. Если он не указан, то будет браться файл по умолчанию, заданный в окне **Сервис ►Настройка►Основные параметры**.
- Способ и параметры выполнения сценария. Здесь нужно выбрать одну из двух опций:
  - *Однократное выполнение сценария*.
  - *Выполнить сценарий для каждой строки исходного набора данных с возможностями: игнорировать ошибки при выполнении сценария; добавить поле, содержащее признак успешного выполнения сценария; добавить поле, содержащее текст ошибки выполнения сценария.*

Замечание

> *Возможность работы с обработчиком «Сценарий Deductor» имеется только в Deductor Enterprise, при этом Deductor Studio должен быть зарегистрирован в качестве OLE Automation сервера (см. «Руководство администратора»).*

Переменные

В Deductor Studio для настройки переменных существует окно, которое можно выбрать из пункта меню **Сервис ► Переменные**. В этом окне настраиваются глобальные и локальные переменные, которые можно использовать в обработчике, где допустимо написание функцией, например, «Калькулятор».

Имеются 3 страницы для настройки переменных:

- *Проект* – список переменных текущего открытого проекта. Значения переменных хранятся внутри файла сценариев (***.ded**).
- *Приложение* – список переменных приложения Deductor Studio/Viewer. Значения переменных хранятся в файле Environment.sys рядом с exe-файлом приложения.
- *Система* – переменные окружения операционной системы. Считываются из настроек системы. Переменные системы добавлять невозможно, можно только использовать их.

Приоритет использования переменных

Переменная с одинаковым именем может быть заведена одновременно в нескольких группах. Для исключения конфликтов используется приоритет:

1. Проект
2. Приложение
3. Система

Т.е. сначала ищется значение в переменных проекта, затем - в переменной приложения и в последнюю очередь – в переменных системы. Значения переменных можно переопределить на этапе выполнения сценария в пакетном режиме при помощи параметров командной строки.

Интерпретация результатов

Одним из важных этапов анализа данных является интерпретация его результатов. Под интерпретацией результатов понимается их анализ экспертом предметной области. Результаты должны быть описаны на языке предметной области. Так как невозможно построить идеальную модель, необходимо оценить, удовлетворяет ли построенная модель запросам, проверить ее качество.

Например, после кластеризации (сегментации) клиентов по частоте и периодичности приобретения услуг, а также количеству приобретаемых услуг, каждому кластеру (сегменту) может быть сопоставлена степень важности или ценности входящих в него клиентов. Это и есть интерпретация полученных сегментов клиентов. Для решения задачи сегментации обычно используется история предоставления услуг за некоторый промежуток времени. Поэтому точность отнесения клиентов к тому или иному сегменту будет зависеть от объема выборки.

В программе есть средства для оценки качества построенной модели. К ним относятся таблица сопряженности, диаграмма рассеяния; в деревьях решений и ассоциативных правилах можно посмотреть поддержку и достоверность по каждому узлу и по правилу целиком.

Для того чтобы оценить качество *классификации* данных, обычно используют таблицу сопряженности. Для решения задачи классификации используется таблица, в которой уже есть выходной столбец, содержащий класс объекта. После применения алгоритма добавляется еще один столбец с выходным полем, но его значения уже вычисляются, используя построенную модель. При этом значения в столбцах могут отличаться. Чем больше таких отличий, тем хуже построенная модель классификации. Пример таблицы после классификации.

Входные поля					Выходное поле	Выходное поле после классификации
Сумма кредита	Возраст	Образование	…	Срок проживания	Давать кредит	Давать кредит
7000	37	Специальное	…	22	Да	Да
7500	38	Среднее	…	12	Да	Нет
14500	60	Высшее	…	30	Нет	Нет
15000	28	Специальное	…	21	Да	Да
32000	59	Специальное	…	29	Да	Да
5000	57	Специальное	…	34	Да	Нет
61500	29	Высшее	…	18	Нет	Нет
13500	37	Специальное	…	28	Нет	Да
25500	68	Высшее	…	30	Нет	Нет
…	…	…	…	…	…	…

Ошибки классификации выделены красным цветом.

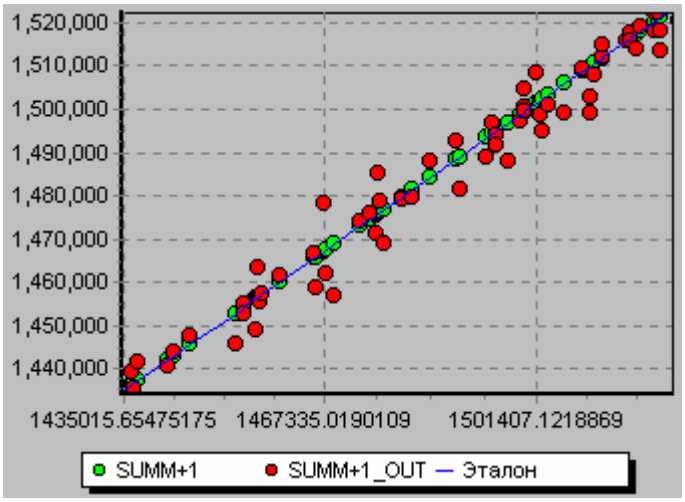
Таблица сопряженности выглядит так.

	Классифицировано		
Фактически	Да	Нет	Итого
Да	50	9	59
Нет	27	63	90
Итого	77	72	149

На главной диагонали на зеленом фоне показано количество правильно классифицированных примеров. В данном примере всего два класса, поэтому таблица сопряженности имеет размер 2x2. По нажатию кнопки **Детализация** на панели инструментов откроется таблица, содержащая примеры, отнесенные к этой ячейке. Например, если выбрать первую ячейку, то откроется таблица с правильно классифицированными примерами, попавшими в класс *Да*.

Если количество неправильно классифицированных примеров довольно велико, это говорит о плохо построенной модели, нужно изменить параметры построения модели, увеличить обучающую выборку либо изменить набор входных полей. Если же количество неправильно классифицированных примеров мало, это может говорить о том, что данные примеры являются аномалиями. В этом случае можно посмотреть, чем же характеризуются такие примеры и, возможно, добавить новый класс для их классификации.

Для оценки качества моделей прогноза какой-либо непрерывной величины, т.е. при решении задачи *регрессии* можно воспользоваться диаграммой рассеяния. На этой диаграмме отображается отклонение прогнозного значения величины от ее истинного значения.



Слишком большое отклонение величины от ее истинного значения говорит о плохо построенной модели и необходимости увеличения обучающей выборки либо предобработки данных. Например, удалить аномалии, убрать шумы, изменить набор входных параметров. Слишком точное совпадение прогнозных значений на обучающей выборке с эталонными может говорить о переобучении модели, т.е. модель «запомнила» все примеры, используемые при ее обучении. В таком случае модель выдает отличные результаты именно на этих данных, но совершенно бесполезна на каких-либо других, например, на данных за другой промежуток времени. Это может произойти, если, например, для построения модели использовалась нейросеть с большим числом слоев.

Для любой точки на диаграмме рассеяния можно посмотреть детализацию. Для этого на панели инструментов следует нажать кнопку , и под диаграммой появится таблица детализации. Если теперь левой кнопкой мыши щелкнуть на интересующей точке диаграммы, в таблице появится пример выборки, которому она соответствует. С помощью детализации можно понять, что за строка набора данных скрывается за точкой с большим отклонением и определить свои дальнейшие действия – проигнорировать этот пример, исключить его из выборки или изменить параметры модели.

ROC-анализ

ROC-анализ позволяет провести оценку качества модели-классификатора, сравнить прогностическую силу нескольких моделей, определить оптимальную точку отсечения для

отнесения объектов к тому или иному классу. При этом предполагается, что у классификатора имеются дополнительные параметры, позволяющие уже после проведенного обучения варьировать соотношение ошибок первого и второго рода. В частности, *логистическая регрессия* (см. одноименный раздел настоящего Руководства) удовлетворяет таким требованиям, т.к. модель на ее основе имеет выходное поле рейтинга, которое можно интерпретировать как вероятность положительного исхода интересующего события.

В основе ROC-анализа лежит построение графиков – *ROC-кривых*. ROC-кривая (Receiver Operator Characteristic) – кривая, которая наиболее часто используется для представления результатов бинарной классификации в машинном обучении. Название пришло из систем обработки сигналов. Поскольку классов два, один из них называется классом с положительными исходами, второй – с отрицательными. ROC-кривая показывает зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров. В терминологии ROC-анализа первые называются истинно положительным, вторые – ложно отрицательным множеством. Как уже говорилось выше, у классификатора имеется некоторый параметр, варьируя который, мы будем получать то или иное разбиение на два класса. Этот параметр часто называют *порогом*, или *точкой отсечения* (cut-off value). В зависимости от него будут получаться различные величины ошибок I и II рода.

Для понимания сути ошибок I и II рода рассмотрим четырехпольную таблицу сопряженности, которая строится на основе результатов классификации моделью и фактической (объективной) принадлежности примеров к классам.

	Фактически	
Модель	положительно	отрицательно
положительно	TP	FP
отрицательно	FN	TN

- *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).

Что является положительным событием, а что – отрицательным, зависит от конкретной задачи. Например, если мы прогнозируем вероятность наличия заболевания, то положительным исходом будет класс «Больной пациент», отрицательным – «Здоровый пациент». И наоборот, если мы ходим определить вероятность того, что человек здоров, то положительным исходом будет класс «Здоровый пациент» и так далее.

При анализе чаще оперируют не абсолютными показателями, а относительными – долями (rates), выраженными в процентах:

Доля истинноположительных примеров (*True Positives Rate*): $TPR = \frac{TP}{TP + FN} \cdot 100\%$.

Доля ложноположительных примеров (*False Positives Rate*): $FPR = \frac{FP}{TN + FP} \cdot 100\%$.

Введем еще два определения: чувствительность и специфичность модели. Ими определяется объективная ценность любого бинарного классификатора.

Чувствительность (Sensitivity) – это и есть доля истинно положительных случаев:

$$Se = TPR = \frac{TP}{TP + FN} \cdot 100\%.$$

Специфичность (Specificity) – доля истинно отрицательных случаев, которые были правильно идентифицированы моделью:

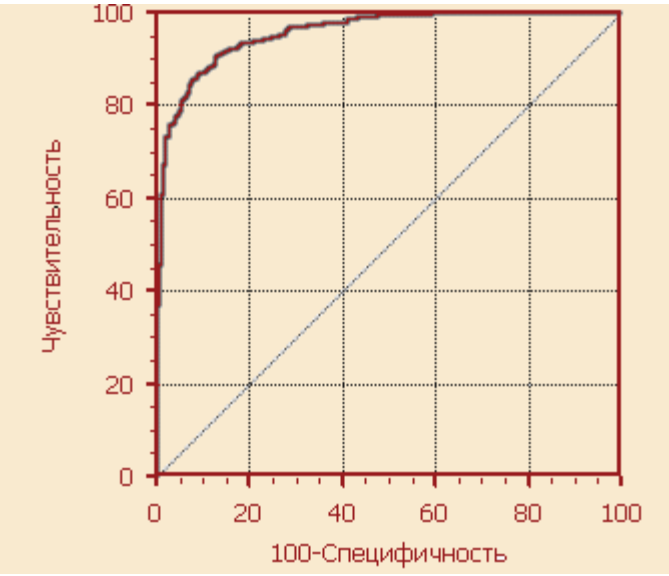
$$Sp = \frac{TN}{TN + FP} \cdot 100\%.$$

Модель с высокой чувствительностью часто дает истинный результат при наличии положительного исхода (обнаруживает положительные примеры). Наоборот, модель с высокой специфичностью чаще дает истинный результат при наличии отрицательного исхода (обнаруживает отрицательные примеры). Если рассуждать в терминах медицины – задачи диагностики заболевания, где модель классификации пациентов на больных и здоровых называется диагностическим тестом, то получится следующее:

- чувствительный диагностический тест проявляется в гипердиагностике – максимальном предотвращении пропуска больных;
- специфичный диагностический тест диагностирует только доподлинно больных. Это важно в случае, когда, например, лечение больного связано с серьезными побочными эффектами и гипердиагностика пациентов не желательна.

ROC-кривая получается следующим образом: для каждого значения порога отсечения, которое меняется от 0 до 1 с шагом dx (например, 0.01) рассчитываются значения чувствительности Se и специфичности Sp. Строится график зависимости: по оси Y откладывается чувствительность Se, по оси $X - (100\% - Sp)$ (сто процентов минус специфичность).

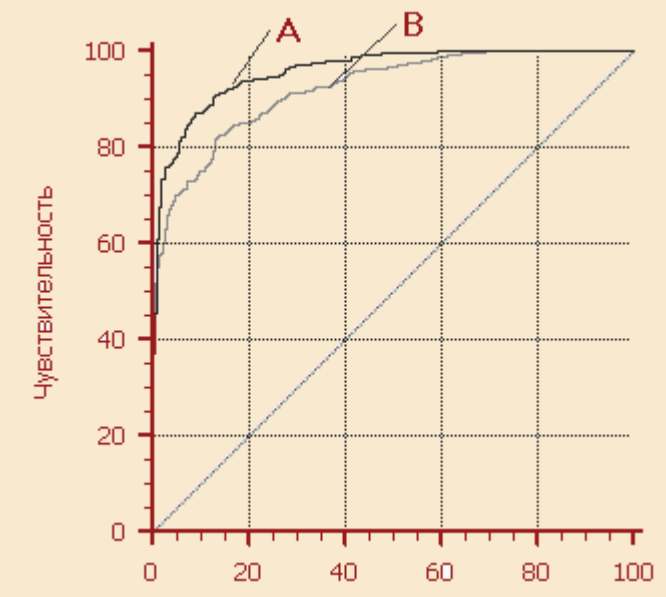
Пример ROC-кривой приведен на рисунке. График часто дополняют прямой $y = x$.



Для идеального классификатора график ROC-кривой проходит через верхний левый угол, где доля истинно положительных случаев составляет 100% или 1,0 (идеальная чувствительность), а

доля ложноположительных примеров равна нулю. Поэтому чем ближе кривая к верхнему левому углу, тем выше предсказательная способность модели. Наоборот, чем меньше изгиб кривой и чем ближе она расположена к диагональной прямой, тем менее эффективна модель. Диагональная линия соответствует «бесполезному» классификатору, т.е. полной неразличимости двух классов.

При визуальной оценке ROC-кривых расположение их относительно друг друга указывает на их сравнительную эффективность. Кривая, расположенная выше и левее, свидетельствует о большей предсказательной способности модели. Так, на рисунке ниже две ROC-кривые совмещены на одном графике. Видно, что модель «A» лучше.



Визуальное сравнение ROC-кривых не всегда позволяет выявить наиболее эффективную модель. Своеобразным методом сравнения ROC-кривых является оценка площади под кривыми. Теоретически она изменяется от 0 до 1,0, но поскольку модель всегда характеризуется кривой, расположенной выше положительной диагонали, то обычно говорят об изменениях от 0.5 («бесполезный» классификатор) до 1,0 («идеальная» модель). Эта оценка может быть получена непосредственно вычислением площади под многогранником, ограниченным справа и снизу осями координат и слева вверху экспериментально полученными точками. Численный показатель площади под кривой называется AUC (Area Under Curve). Вычислить его можно, например, с помощью численного метода трапеций. Если $AUC \geq 0{,}8$, то можно говорить о том, что модель обладает высокой прогностической силой.

Идеальная модель обладает 100% чувствительностью и специфичностью. Однако на практике добиться этого невозможно, более того, невозможно одновременно повысить и чувствительность и специфичность модели. Компромисс находится с помощью порога отсечения, т.к. пороговое значение влияет на соотношение Se и Sp. Можно говорить о задаче нахождения *оптимального порога отсечения* (optimal cut-off value).

Порог отсечения нужен для того, чтобы применять модель на практике: относить новые примеры к одному из двух классов. Для определения оптимального порога нужно задать критерий его определения, т.к. в разных задачах присутствует своя оптимальная стратегия. Существует по крайней мере, 2 варианта:

- Требование максимальной суммарной чувствительности и специфичности модели, т.е.

$$\text{Cut\_off}_{\text{o}} = \max_{k}\left(Se_k + Sp_k\right).$$

- Требование баланса между чувствительностью и специфичностью, т.е. когда $Se \approx Sp$:

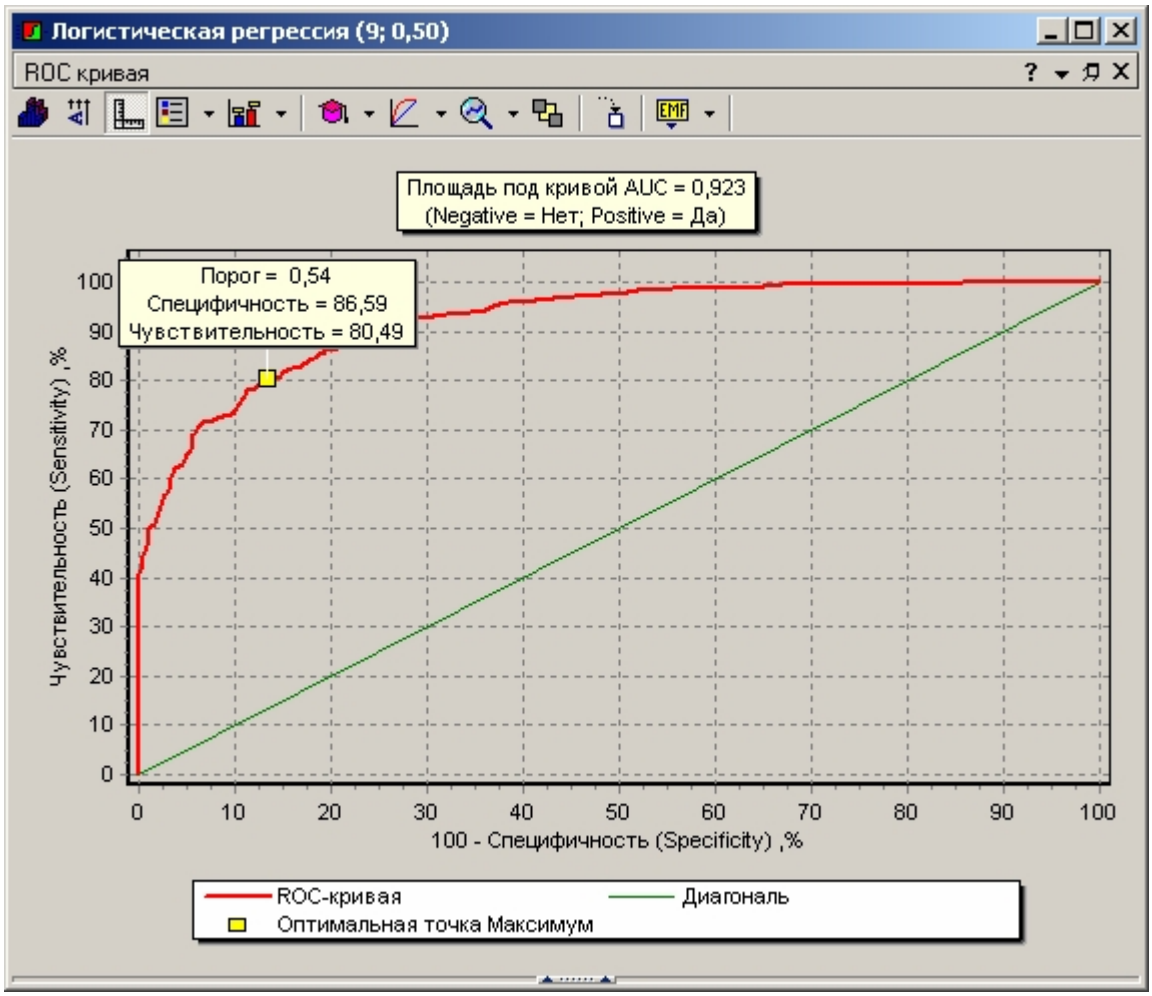
$$\text{Cut\_off}_o = \min_k \left| Se_k - Sp_k \right|.$$

Пример

В Deductor ROC-анализ является визуализатором, доступным после обработчика «Логистическая регрессия». Проведем ROC-анализ балльной модели оценки кредитоспособности физических лиц, построенной в разделе «Логистическая регрессия».

Проецируя определения чувствительности и специфичности на оценку кредитоспособности (скоринг), можно заключить, что скоринговая модель с высокой специфичностью соответствует *консервативной кредитной политике* (чаще происходит отказ в выдаче кредита), а с высокой чувствительностью – *политике рискованных кредитов*. В первом случае минимизируется кредитный риск, связанный с потерями ссуды и процентов и дополнительными расходами на возвращение кредита, а во втором – коммерческий риск, связанный с упущенной выгодой.

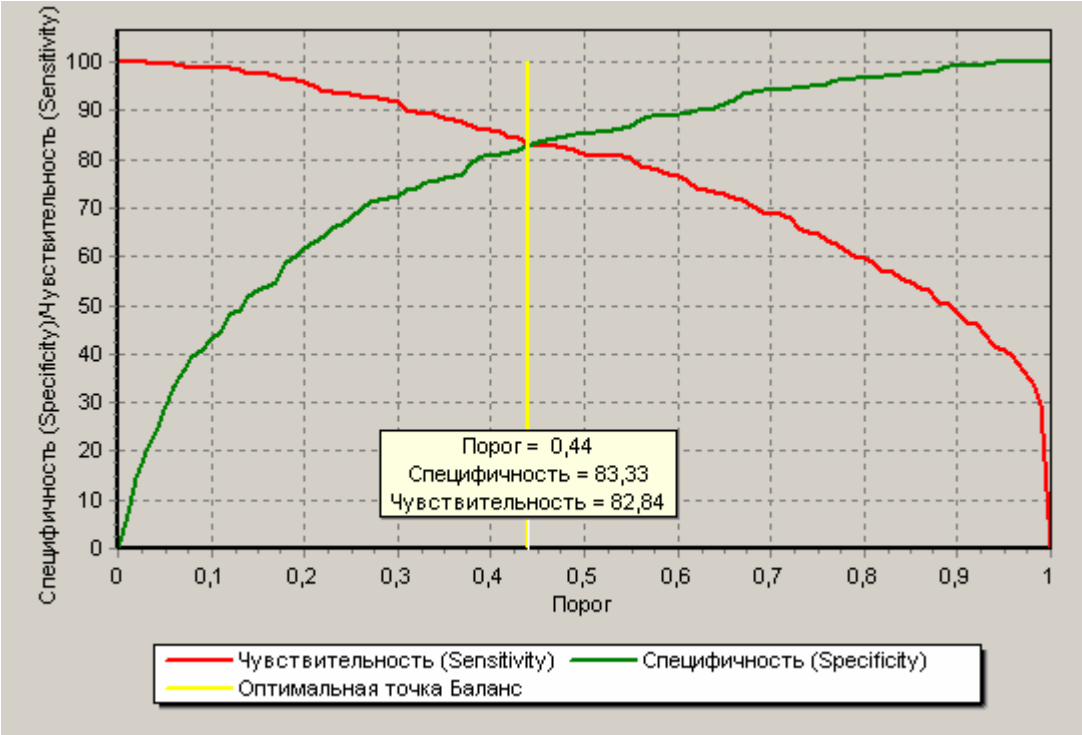
При первоначальном открытии визуализатора в Deductor рисуется график ROC-кривой, вычисляется оптимальная точка по способу максимума чувствительности и специфичности («точка максимума») и площадь под кривой AUC.



Оптимальный порог («точка максимума») в нашем случае оказался равен 0,54, а не 0,5, каким он устанавливается по умолчанию. В этой точке чувствительность превышает 80%, что означает следующее: 80% благонадежных заемщика будут подтверждены моделью. Специфичность равна 86,6%, следовательно, 13,4% недобросовестных заемщиков получат одобрение в выдаче кредита (кредитный риск). Это хороший показатель, свидетельствующий о том, что скоринговая

модель соответствует умеренной кредитной политике, но если такая ситуация не устраивает, то можно увеличить величину порога или найти такую точку на ROC-кривой, в которой будет достигаться требуемое соотношение чувствительности и специфичности.

Баланс между чувствительностью и специфичностью («точка баланса») получается в точке 0,44 (Se и Sp около 82%). Точку баланса можно наблюдать на графике кривой баланса.



Для переключения между графиками используйте кнопки и . Для переключения между критерием оптимальности для точки (максимум, баланс, текущая) нужно нажать кнопку на панели.

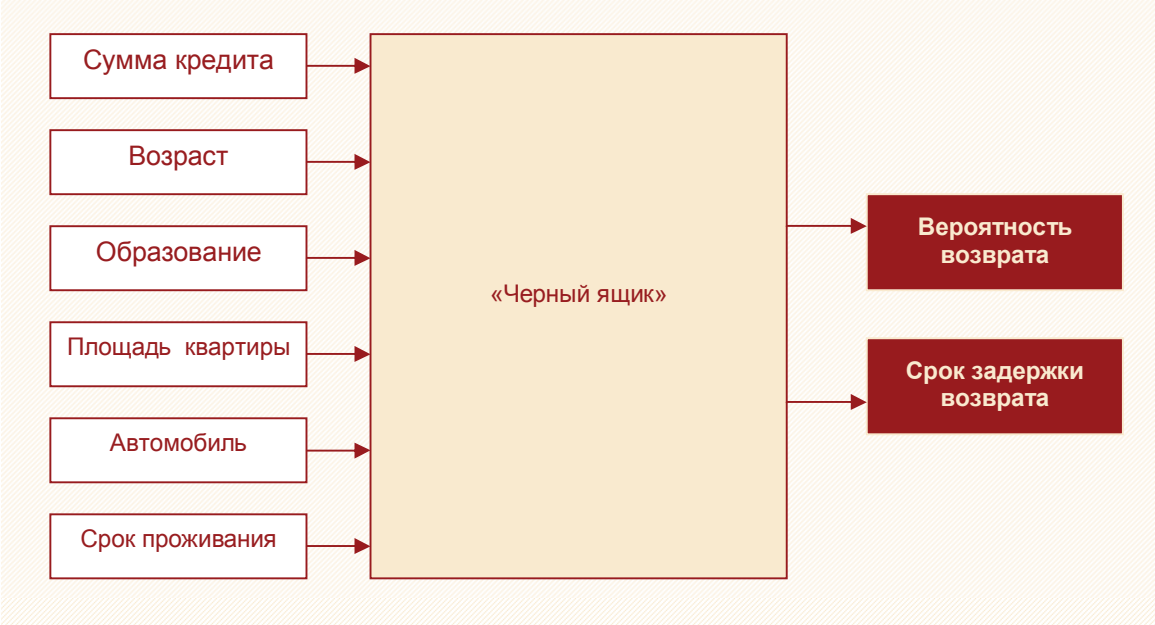
Тип события при построении ROC-кривой берется из установок нормализации логистической регрессии и выводится в легенде графика под значением AUC. В данном примере отрицательному исходу (Negative) соответствует «плохой» заемщик (займ не был возвращен), а положительному (Positive) – займ возвращен.

Определять значение оптимального порога отсечения *имеет смысл на обучающемся или общем множестве*, т.е. на тех данных, которые принимали участие в расчете коэффициентов. Поэтому на тестовом множестве, если оно рассматривается отдельно (кнопка), оптимальные точки не рассчитываются и не показываются на ROC-кривой и в детализации.

Анализ «Что-если»

Анализ «Что-если» означает ответ на вопрос: «**Что** получится, **если** задать такие значения факторов?» Здесь подразумевается, что есть некоторая величина (а в общем случае может быть несколько величин), которая зависит от различных входных факторов. При изменении входных факторов будут изменяться и значения зависимых величин. Решение обратной задачи, то есть поиск значений входных факторов для получения желаемого результата, является задачей оптимизации.

Рассмотрим пример. Есть зависимость между вероятностью возврата кредита и характеристиками кредитора. Эту зависимость можно представить в виде «черного ящика».



Прежде, чем приступать к анализу, нужно построить модель этой зависимости. Для этого в программе есть такие инструменты, как нейронные сети, деревья решений и прочие. Для анализа совершенно безразлично, каким методом была построена модель. Главное, что она имитирует работу «черного ящика», то есть может по входным параметрам вычислить значение выходного. Сделать это можно с помощью таблицы «Что-если».

Таблица «Что-если»

В этой таблице перечислены все входные и выходные поля, указан диапазон значений для числовых полей и количество значений для строковых. В колонке «Значение» для каждого входного поля можно указать значения, по которым требуется вычислить выходное поле.

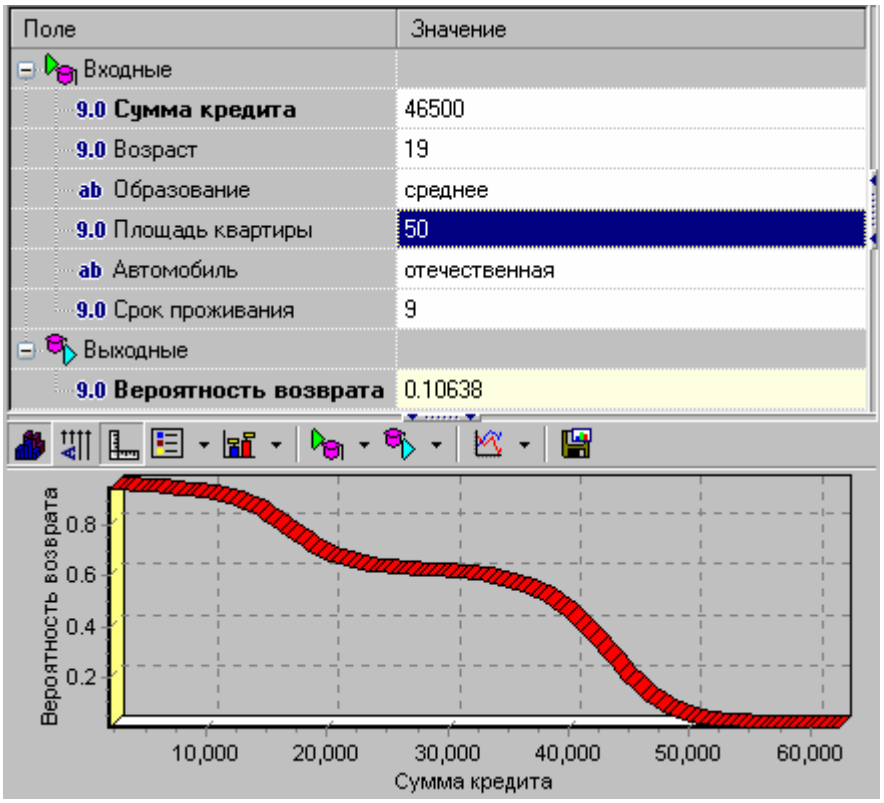
Поле	Значение
Входные	
9.0 Сумма кредита	10000
9.0 Возраст	20
ab Образование	среднее
9.0 Площадь квартиры	37
ab Автомобиль	отечественная
9.0 Срок проживания	22
Выходные	
9.0 Вероятность возврата	0.63465

Для вычисления значения служит кнопка **Рассчитать выходы** на панели инструментов. При вводе в таблицу строковых значений предлагается выбрать значение из списка. Например, для поля *Образование* может быть список: *Среднее*, *Специальное*, *Высшее*. При вводе числовых значений можно ввести любое число. Однако, желательно вводить числа из диапазона «Минимум» и «Максимум», так как модель была построена на значениях именно из этого диапазона.

Таким образом, таблица «Что-если» имитирует работу «черного ящика». Зная значения входных полей, можно вычислить значения выходных. Это имеет практическую ценность, когда известны значения всех входных полей. Например, при решении задачи оценки кредитоспособности человека.

Диаграмма «что–если»

Часто возникает ситуация, когда необходимо подобрать значение одного из входных полей для получения желаемого значения выходного поля. Например, человеку необходимо взять кредит на определенную сумму. Вероятность возврата, вычисленная в таблице «Что-если», получилась низкой. Это может говорить о слишком высокой сумме кредита, которую он запросил. Возникает вопрос, на какую сумму может рассчитывать этот человек? Ответ на него дает диаграмма «Что-если». Эта диаграмма показывает зависимость выходного поля от одного из входных при фиксированных значениях остальных полей. В данном случае нас интересует зависимость вероятности возврата от суммы запрашиваемого кредита.

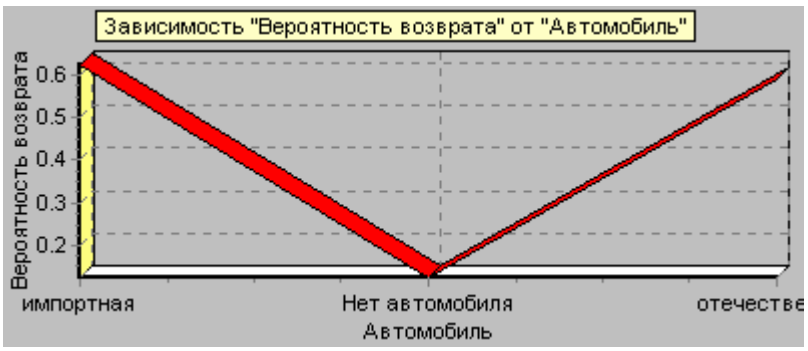


Рассмотрим, например, такую ситуацию.

Человек возрастом 19 лет со средним образованием, квартирой 50 кв. метров, отечественным автомобилем, проживающий 9 лет в данной местности запросил кредит на сумму 46500. Анализ по таблице «Что-если» показал очень низкую вероятность возврата кредита – примерно 1%. Такой риск является неприемлемым. А диаграмма «Что-если» показывает, что для людей с такими характеристиками кредит более 20 000 слишком большой и вероятность возврата низкая (менее 60 %). А вот сумму меньше 20 000 можно выдать.

Чтобы выбрать поле, по которому строить зависимость, нужно использовать кнопку **Вход** на панели инструментов диаграммы. Нажимая на нее, перебираются все входные поля. А нажав треугольник рядом с кнопкой, можно выбрать необходимое поле из списка.

Например, зависимость вероятности возврата кредита от наличия и типа автомобиля может быть такой.



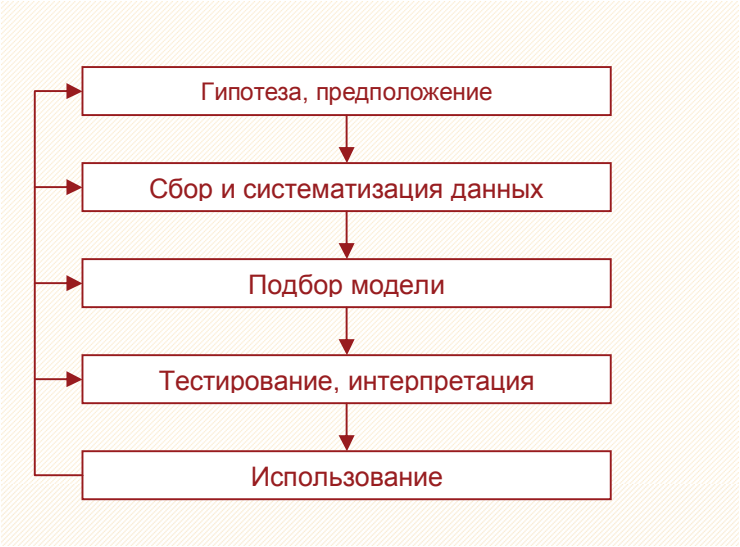
То есть если придут два человека с абсолютно одинаковыми характеристиками и запросят одну и туже сумму кредита, то его выдача будет зависеть от наличия у них автомобиля.

Подготовка данных для анализа

Методика анализа с использованием механизмов Data Mining базируется на различных алгоритмах извлечения закономерностей из исходных данных, результатом работы которых являются модели. Таких алгоритмов довольно много, но несмотря на их обилие, использование машинного обучения и т.п., они не способны гарантировать качественное решение. Никакой самый изощренный метод сам по себе не даст хороший результат, т.к. критически важным становится вопрос качества исходных данных. Чаще всего именно качество данных является причиной неудачи.

Ниже описана методика, следуя которой можно подготовить качественные данные в нужном объеме для анализа. В этой последовательности действий все достаточно просто и логично, но несмотря на это пользователи почти всегда допускают одни и те же тривиальные ошибки.

Общая схема использования методов Data Mining состоит из следующих шагов.



Эта последовательность действий не зависит от предметной области, поэтому ее можно использовать для любой сферы деятельности.

Выдвижение гипотез

Гипотезой в данном случае будем считать предположение о влиянии определенных факторов на исследуемую нами задачу. Форма этой зависимости в данном случае значения не имеет, т.е. мы можем сказать, что на продажи влияет отклонение нашей цены на товар от среднерыночной, но при этом не указывать, как, собственно, этот фактор влияет на продажи. Для решения этой задачи – поиска зависимости и используется Data Mining. Автоматизировать процесс выдвижения гипотез не представляется возможным, по крайней мере, на сегодняшнем уровне развития технологий. Эту задачу должны решать эксперты – специалисты в предметной области. Полагаться можно и нужно на их опыт и здравый смысл. Нужно постараться максимально использовать их знание о предмете и собрать как можно больше гипотез/предположений. Обычно для этих целей хорошо работает тактика мозгового штурма. На первом шаге нужно собрать и систематизировать все идеи, их оценку будем производить позже. Результатом данного шага должен быть список с описанием всех факторов.

Например, для задачи прогнозирования спроса это может быть список следующего вида: сезон, день недели, объемы продаж за предыдущие недели, объем продаж за аналогичный период прошлого года, рекламная компания, маркетинговые мероприятия, качество продукции, бренд, отклонение цены от среднерыночной, наличие данного товара у конкурентов и т.д.

После подготовки таблицы с описанием факторов нужно экспертно оценить значимость каждого из факторов. Эта оценка не является окончательной, она будет отправной точкой. В процессе анализа вполне может оказаться, что фактор, который эксперты посчитали очень важным, таковым не является, и, наоборот, незначимый с их точки зрения фактор может оказывать значительное влияние.

Формализация и сбор данных

Далее необходимо опередить способ представления данных, выбрав один из 4-х видов: число, строка, дата, логическая переменная (да/нет). Определить способ представления, т.е. формализовать, некоторые данные просто, например, объем продаж в рублях – это определенное число. Но довольно часто возникают ситуация, когда непонятно, как представить фактор. Чаще всего такие проблемы возникают с качественными характеристиками. Например, на объемы продаж влияет качество товара. Качество – это довольно сложное понятие, но если этот показатель действительно важен, то нужно придумать способ его формализации. Например, определять качество по количеству брака на тысячу единиц продукции либо экспертно оценивать, разбив на несколько категорий – отлично/хорошо/удовлетворительно/плохо.

Необходимо оценить стоимость сбора нужных для анализа данных. Дело в том, что некоторые данные легко доступны, например, их можно извлечь из существующих информационных систем. Но есть информация, которую непросто собрать, например, сведения о конкурентах. Поэтому необходимо оценить, во что обойдется сбор данных.

Сбор данных не является самоцелью. Если информацию получить легко, то, естественно, нужно ее собрать. Если данные получить сложно, то необходимо соизмерить затраты на ее сбор и систематизацию с ожидаемыми результатами.

Есть несколько методов сбора, необходимых для анализа данных:

1. Получение из учетных систем. Обычно в учетных системах есть различные механизмы построения отчетов и экспорта данных, поэтому извлечение нужной информации из них чаще всего относительно несложная операция.
2. Получение сведений из косвенных данных. О многих показателях можно судить по косвенным признакам, и этим нужно воспользоваться. Например, можно оценить реальное финансовое положение жителей определенного региона следующим образом. В большинстве случаев имеется несколько товаров, предназначенных для выполнения одной и той же функции, но отличающихся по цене: товары для бедных, средних и богатых. Если получить отчет о продажах товара в интересующем регионе и проанализировать пропорции, в которых продаются товары для бедных, средних и богатых, то можно предположить, что чем больше доля дорогих изделий из одной товарной группы, тем более состоятельны в среднем жители данного региона.
3. Использование открытых источников. Большое количество данных присутствует в открытых источниках, таких как статистические сборники, отчеты корпораций, опубликованные результаты маркетинговых исследований и прочее.
4. Приобретение аналитических отчетов у специализированных компаний. На рынке работает множество компаний профессионально занимающиеся сбором данных и предоставлением их клиентами для последующего анализа. Собираемая ими информация обычно предоставляется в виде различных таблиц и сводок, которые с успехом можно применять при анализе. Стоимость получения подобной информации чаще всего относительно невысокая.
5. Проведение собственных маркетинговых исследований и аналогичных мероприятий по сбору данных. Это может быть достаточно дорогостоящим мероприятием, но в любом случае такой вариант сбора данных возможен.
6. Ввод данных «вручную», когда данные вводятся по различного рода экспертным оценкам сотрудниками организации. Этот метод наиболее трудоемкий.

Стоимость сбора информации различными методами существенно отличается по цене и необходимому для этого времени, поэтому нужно соизмерять затраты с результатами. Возможно,

от сбора некоторых данных придется отказаться, но факторы, которые эксперты оценили как наиболее значимые нужно собрать обязательно, несмотря на стоимость этих работ либо вообще отказаться от анализа. Очевидно, что если эксперт указал на некоторый фактор как важный, то не учитывать его просто нельзя, т.к. мы рискуем провести анализ, ориентируясь на второстепенные малозначащие факторы. И, следовательно, получить модель, которая будет давать плохие и нестабильные результаты. А такая модель не представляет практической ценности.

Представление и минимальные объемы необходимых данных

Для анализируемых процессов различной природы данные должны быть подготовлены специальным образом.

Упорядоченные данные

Такие данные нужны для решения задач прогнозирования, когда необходимо определить, каким образом поведет себя тот или иной процесс в будущем на основе имеющихся исторических данных. Чаще всего в качестве одного из фактов выступает дата или время, хотя это и не обязательно. Речь может идти и о неких отсчетах, например, данных, с определенной периодичностью собираемых с датчиков.

Для упорядоченных данных (обычно это временные ряды) каждому столбцу соответствует один фактор, а в каждую строку заносятся упорядоченные по времени события с единым интервалом между строками. Не допускается наличие группировок, итогов и прочее, нужна обычная таблица.

№ п/п	Дата	Частота закупок	Объем продаж (руб.)
1	01.05.2004	256	459 874,00
2	02.05.2004	278	515 687,00

Если для процесса характерна сезонность/цикличность, необходимо иметь данные хотя бы за один полный сезон/цикл с возможностью варьирования интервалов (понедельное, помесячное…), т.к. цикличность может быть сложной, например, внутри годового цикла квартальные, а внутри кварталов недельные, то необходимо иметь полные данные как минимум за один самый длительный цикл.

Максимальный горизонт прогнозирования зависит от объема данных:

- Данные на 1,5 года – прогноз максимум на 1 месяц.
- Данные за 2-3 года – прогноз максимум на 2 месяца.

В общем случае максимальный горизонт прогнозирования (время, на которое можно строить достаточно достоверные прогнозы) ограничивается не только объемом данных. Мы исходим из предположения, что факторы, определяющие развитие процесса, будут оказывать влияние и в будущем примерно такое же, что и на текущий момент. Данное предположение справедливо не всегда. Например, в случае слишком быстрого изменения ситуации, появления новых значимых факторов и т.п. это правило не работает. Поэтому в зависимости от задачи требования к объему могут сильно изменяться. Использование слишком большого объема данных для анализа также нецелесообразно, т.к. в этом случае мы будем строить модель по старой истории, и, следовательно, возможно, будем учитывать факторы, уже утратившие свою значимость.

Неупорядоченные данные

Такого рода данные нужны для задач, где временной фактор не имеет значения, например, оценка кредитоспособности, диагностика, сегментация потребителей. В таких случаях мы

считаем ситуацию статичной, и поэтому информация о том, что одно событие произошло раньше другого, значения не имеет.

Для неупорядоченных данных каждому столбцу соответствует фактор, а в каждую строку заносится пример (ситуация, прецедент). Упорядоченность строк не требуется. Не допускается наличие группировок, итогов и прочее, нужна обычная таблица.

Номер прецедента	Стаж работы	Наличие автомобиля	Сумма кредита (руб.)
1	Больше 5 лет	Да	150 000,0
2	Меньше 5 лет	Нет	125 000,0

Количество примеров (прецедентов) должно быть значительно больше количества факторов (минимум в 2 раза). В противном случае высока вероятность, что случайный фактор окажет серьезное влияние на результат. Если нет возможности увеличить количество данных, то придется уменьшить количество анализируемых факторов, оставив наиболее значимые.

Желательно, чтобы данные покрывали как можно больше ситуаций реального процесса, и пропорции различных примеров (прецедентов) должны примерно соответствовать реальному процессу. Мы пытаемся построить модели на основе предложенных данных, поэтому, чем ближе данные к действительности, тем лучше. Необходимо понимать, что система не может знать о чем-либо, что находится за пределами собранных для анализа данных. Например, если при создании системы диагностики больных подавать только сведения о больных, то система не будет знать о существовании в природе здоровых людей. И соответственно, любой человек с ее точки зрения будет обязательно чем-то болен.

Транзакционные данные

Транзакционные данные используются в алгоритмах поиска ассоциативных правил. Под транзакцией подразумевается несколько объектов или действий, сгруппированных в логически связанную единицу. Очень часто данный механизм используется для анализа покупок (чеков) в супермаркетах. Но в общем случае речь может идти о любых связанных объектах или действиях, например, продажа туристических туров с набором сопутствующих услуг (оформление виз, доставка в аэропорт, услуги гида и прочее). Используя данный метод анализа, находятся зависимости вида, «если произошло событие А, то с определенной вероятностью произойдет событие Б».

Транзакционные данные для анализа необходимо подготовить в следующем виде:

Код транзакции	Товар
10200	Йогурт «Чудо» 0,4
10200	Батон «Рязанский»
10201	Вода «Боржоми» 0,5
10201	Сахарный песок, пачка 1 кг.
10201	Хлеб «Бородинский»

Код транзакции соответствует коду чека, счета, накладной. Товары с одинаковым кодом входят в разовую покупку.

Описанного представления данных достаточно для работы обычных ассоциативных правил, где находятся связи между каждым объектом в отдельности, например, «Если купили Йогурт Чудо 0,4, то приобретут и Батон Рязанский».

Анализ транзакций целесообразно производить на большом объеме данных, иначе могут быть выявлены статистически необоснованные правила. Алгоритмы поиска ассоциативных связей способны быстро перерабатывать огромные массивы информации, т.к. основное их достоинство заключается в масштабируемости, т.е. способности обрабатывать большие объемы данных.

Примерное соотношение между количеством объектов и объемом данных:

- 300-500 объектов – более 10 тыс. транзакций;
- 500-1000 объектов – более 300 тыс. транзакций.

При недостаточном количестве транзакций целесообразно уменьшить количество анализируемых объектов, например, сгруппировав их.

Построение моделей – анализ

В целом, можно дать следующие рекомендации при построении моделей, не зависящие от конкретного алгоритма обработки:

- Уделить большое внимание очистке данных. Собрав данные в нужном объеме, нельзя быть уверенным, что они будут хорошего качества. Чаще всего качество данных оставляет желать лучшего, поэтому необходимо проводить предобработку. Для этого есть множество методов: удаление шумов, сглаживание, редактирование аномалий и прочее;
- Комбинировать методики анализа. Это позволяет шире смотреть на проблему. Более того, использование различных методов для решения одной и той же задачи может навести на ценные идеи;
- Не гнаться за абсолютной точностью и начинать использование при получении первых приемлемых результатов. Все равно идеальный результат получить невозможно. Если мы получили результат, пусть неидеальный, но лучше, чем был ранее, то есть резон начать его использование. Во-первых, это позволяет быстрее получить практическую отдачу. Во-вторых, только на практике можно действительно оценить полученный результат. В-третьих, можно и нужно параллельно работать над совершенствованием модели с учетом полученных на практике результатов;
- При невозможности получения приемлемых результатов вернуться на предыдущие шаги схемы. К сожалению, ошибки могут быть допущены на любом шаге: может быть некорректно сформулирована первоначальная гипотеза, могут возникнуть проблемы со сбором необходимых данных и прочее. К этому нужно быть готовым. При возникновении такого рода проблем возвращаться на предыдущие пункты и рассмотреть альтернативные варианты решения.

Для оценки адекватности полученных результатов необходимо привлекать экспертов в предметной области. Интерпретация модели, так же как и выдвижение гипотез, может и должна делаться экспертом, т.к. для этого нужно более глубокое понимание процесса, выходящее за пределы анализируемых данных. Кроме того, нужно воспользоваться и формальными способами оценки качества модели: тестировать построенные модели на различных выборках для оценки их обобщающих способностей, т.е. способности давать приемлемые результаты на данных, которые не предоставлялись системе при построении модели. Некоторые механизмы анализа могут «запоминать» предъявленные ей данные и на них демонстрировать прекрасные результаты, но при этом полностью терять способность к обобщению и на тестовых (из неизвестных системе ранее) данных выдавать очень плохие результаты. При формальной оценке можно отталкиваться от идеи, что если на тестовых данных модель дает приемлемые результаты, значит, она имеет право на существование.

Оптимизация работы и создания сценариев

Анализ реальной бизнес-информации зачастую связан с обработкой больших объемов данных. В связи с этим появляются проблемы оптимизации работы готового сценария. На быстродействие создаваемой аналитической системы могут значительно повлиять особенности разработанных сценариев анализа, поэтому рассмотрим подробнее пути повышения производительности при обработке сценариев. Кроме того, рассмотрим методы создания сценариев, которые позволяют значительно упростить и ускорить разработку.

Какие источники использовать

Deductor может одинаково успешно работать с большим количеством разнообразных источников данных. Однако скорость извлечения данных во многом определяется типом используемого источника. При создании законченного решения этот фактор становится очень важным, так как оперативность получения аналитической информации имеет большое значение на практике.

Самыми быстрыми источниками данных являются хранилище Deductor Warehouse, базы данных, подключаемые напрямую (MS SQL, Oracle) или через драйверы OLE DB или ODBC, и прямой доступ к текстовым файлам и файлам dbf. Остальные источники работают значительно медленнее.

Использовать для получения данных источники, подключаемые через интерфейс ADO, следует только в крайнем случае, при отсутствии других возможностей доступа, как, например, к таблицам MS Excel и MS Access. На больших объемах данных доступ к текстовому файлу через ADO значительно медленнее прямого доступа. Кроме того, при прямом доступе можно указать большое число настроек разбора текстового файла. Подключение к базам данных через ADO приводит к тому, что загрузка данных в процесс хранилища из таблицы DBF занимает в 10-12 раз больше времени по сравнению с прямым доступом к этой таблице.

Очень медленным источником данных является 1С:Предприятие. Это связано с тем, что доступ к данным осуществляется не напрямую, а через OLE сервер 1С:Предприятие, который выполняет большое количество вспомогательных действий для получения данных. Тем не менее, удобство работы с ней напрямую часто важнее высокого быстродействия. Для того чтобы ускорить загрузку в Deductor данных из 1С:Предприятия, следует пользоваться механизмом внешних отчетов 1С. В этом случае подготавливается отчет, в котором нужные данные выгружаются в таблицы dbf, а уже из них производится загрузка данных в Deductor. Если для загрузки данных в хранилище применять пакетную загрузку, настроив ее запуск на ночное время, то создание промежуточных отчетов может и не потребоваться. Объем данных, хранимых в конфигурациях 1С:Предприятия, обычно не очень велик (сотни тысяч строк в документах, тысячи в регистрах и справочниках), вдобавок к этому документы будут загружаться только за последний период, и это не займет много времени. Справочники можно загружать изредка, только при появлении в них изменений. Поэтому наибольшее время займет загрузка регистров. По опыту работы 1С:Предприятие с базами на основе MS SQL работает при прямой выгрузке данных быстрее, чем при использовании баз dbf.

В случае неизбежности получения данных из медленных источников (ADO, 1С:Предприятие) обычно следует их загружать в хранилище Deductor Warehouse. При активной работе с данными это позволит гораздо меньше задумываться о вопросах производительности.

Кэширование

Значительно увеличить скорость выполнения сценариев может установка в «узких» местах кэша данных в оперативной памяти. Кэширование данных включается при установке в узле «Настройка набора данных» флажка **Кэшировать данные столбца**. В результате данные выбранных столбцов, получаемые на выходе этого узла, будут полностью загружены в память. Обращение к ним будет происходить гораздо быстрее, но за счет дополнительных затрат памяти.

Стоит кэшировать не все столбцы, а только реально используемые при обработке с последующих узлах сценария.

По умолчанию Deductor работает с данными таким образом, чтобы минимизировать объем используемой памяти. Каждый раз, когда какому-нибудь узлу требуется новая порция данных, он обращается к родительскому узлу с запросом на ее предоставление. Если в родительском узле данные не закэшированы, он в свою очередь отправляет запрос вверх по иерархии и так далее, пока данные не будут получены. В предельном случае узлом, предоставляющим данные, может стать узел импорта данных. Таким образом, в Deductor применен подход уменьшения требуемого объема памяти за счет роста количества вычислений, необходимых для получения каждой порции данных.

Обычно внутренняя организация работы программы не имеет особого значения. Тем не менее, могут возникать такие ситуации, когда хранение полной копии данных в некотором узле может радикально повысить скорость выполнения сценария. Для примера рассмотрим следующую ситуацию.

В узле «Калькулятор» (или любом другом) производятся сложные и объемные расчеты, причем данные из этого узла и его потомков активно используются при визуализации в интерактивном режиме. В этом случае каждый раз при отображении данных могут возникать задержки, связанные с тем, что каждый раз в «Калькуляторе» производятся расчеты. Установка узла найтройки набора данных и кэширование нужных столбцов после «Калькулятора» позволит работать при каждом обращении к данным с единожды рассчитанными результатами.

Узел может являться родительским для нескольких ветвей обработки. В этом случае каждая ветка будет независимо от остальных запрашивать данные у родительского узла. Ему, в свою очередь, потребуется обращение на верхние уровни. Таким образом, одни и те же данные будут многократно запрашиваться и обрабатываться вышестоящими узлами. В этом случае удобней было бы после этого узла установить узел кэширования и уже из него наследовать ветви последующей обработки.

Существуют некоторые ситуации, в которых кэширование данных приведет лишь к избыточным ненужным затратам памяти:

- Кэширование данных после узла импорта не имеет смысла, так как в узле импорта уже присутствуют все данные внешнего источника. Кэшировать стоит только после применения какого-либо обработчика.
- Не имеет смысла кэшировать данные после каждого узла обработки. Это не даст большого прироста скорости, но на больших объемах данных очень быстро исчерпает имеющиеся ресурсы памяти. Кэширование следует применять только в «узких» местах сценария, где это действительно дает заметный прирост производительности.
- Не имеет смысла кэширование данных перед узлами, выполняющими нормализацию и кодирование данных. Это обработчики «Нейросеть», «Карты Кохонена», «Ассоциативные правила», «Дерево решений», «Линейная регрессия» и «Пользовательская модель». При нормализации в любом случае осуществляется преобразование данных, после которого они кэшируются в память специальным образом.

Следует отметить, что кэшировать имеет смысл относительно небольшие объемы данных, размер которых не превышает свободные ресурсы оперативной памяти. Если в результате установки кэша данные не поместятся полностью в оперативную память, операционная система начнет выгружать их на жесткий диск в файл подкачки. В результате обращений к жесткому диску работа сценария может замедлиться даже по сравнению с работой в отсутствие кэша.

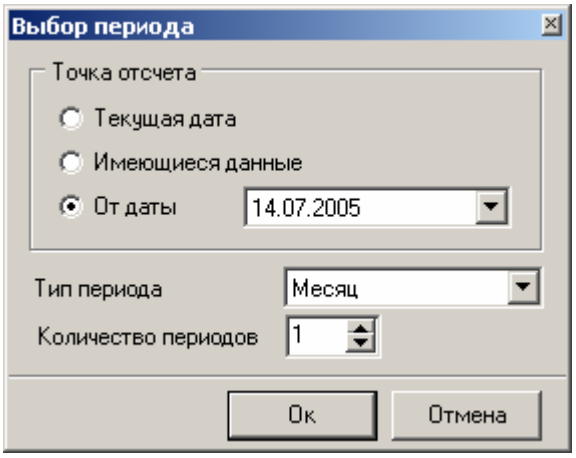
Динамические фильтры

Ключевым моментом, сказывающимся на скорости получения результатов, служит объем анализируемых данных. В некоторых случаях существует возможность значительно уменьшить объем обрабатываемых данных благодаря использованию возможностей динамической фильтрации, которая имеется в Мастере импорта из Deductor Warehouse.

При построении отчета часто возникает ситуация, когда пользователю требуется просмотреть его в некотором узком разрезе. Например, его интересует информация по конкретному поставщику и товару, или продажи за последний месяц, или отчет по работе одного дилера. Так как заранее предсказать конкретный запрос пользователя и создать на каждый запрос отдельный отчет физически невозможно, появляется необходимость в динамической генерации отчетов. Отчасти такую проблему решает использование OLAP-кубов. При этом можно посмотреть доступные данные в любом разрезе, отфильтровав ненужное. Минусом такого подхода является то, что из хранилища данных в этом случае выбирается большой объем избыточной информации. Если пользователю нужен отчет *по одному* поставщику, из хранилища все равно будут выбраны данные *по всем* поставщикам. При активной работе с хранилищем нескольких пользователей это может значительно замедлить получение ими затребованной информации. Кроме того, избыток измерений в кубе усложняет работу с ним и увеличивает усилия, затрачиваемые на получение каждого отчета.

Для решения этой проблемы в Deductor Studio введен механизм *динамических* (или *пользовательских*) фильтров. Эти фильтры становятся доступными при импорте данных из хранилища. При создании узла импорта на странице настройки среза есть возможность для каждого фильтра установить флаг **Определить при выполнении**. Его установка включает пользовательский фильтр. Когда узел импорта будет выполняться в следующий раз, на экран будет выведено окно настройки среза с предложением указать нужный разрез извлекаемых данных. Пользователь имеет выбор оставить разрез, предлагаемый аналитиком по умолчанию, или задать свои настройки выбираемых данных. После задания разреза и закрытия этого окна программа начнет импорт данных из хранилища. При этом будут извлекаться только те данные, которые удовлетворяют условиям фильтрации, заданным пользователем. Если пользователь выбрал одного поставщика, то будут получены данные только по этому поставщику, благодаря чему нагрузка на сервер хранилища данных и сеть заметно уменьшаются. Сценарий затем обрабатывает полученные данные в обычном режиме и выдает на выходе отчет в ранее определенном виде, но уже только в том разрезе, в котором он затребован пользователем.

Другой механизм создания динамических отчетов заключается в использовании фильтров за период. Например, часто возникает задача сравнить два соседних отчетных периода (месяца, квартала или года). В этом случае при импорте из хранилища или в обработчике «Фильтрация» по измерению «*Дата*» устанавливается фильтр, в котором указывается условие – отобрать два последних месяца от имеющихся данных. На выходе фильтра получим выборку, содержащую только эти два периода. Фильтр «*от имеющихся данных*» позволяет работать только с фактически имеющимися в фильтруемом измерении периодами времени. Если в измерении «*Дата*» присутствуют данные за период с 10 января 2005 по 27 мая 2005, а сегодняшнее число 14 июля 2005, то в предыдущем примере получим данные за апрель и май 2005 года. Указав тот же период, но *от текущей даты*, получим данные уже за июнь и июль. В приведенном примере, так как информации за эти месяцы еще нет в хранилище, то на выходе получим пустой набор данных. Фильтр «*от даты*» позволяет использовать в качестве точки отсчета конкретную дату.



При построении прогноза часто бывает так, что данные за последний и первый периоды не полные. Например, при месячном прогнозе есть данные только за период с 14 числа первого месяца по 10 число последнего. Если учитывать их при построении модели прогноза, то получим резкое падение продаж за последний месяц, а это может сильно исказить результаты прогноза. Влияние первого месяца значительно слабее, но также может сказаться, например, при выявлении сезонности. Поэтому перед тем, как строить прогноз, следует удалить эти данные из выборки. Для этого можно воспользоваться фильтром при импорте данных из хранилища или обработчиком «*Фильтрация*» с динамическими фильтрами «*кроме первого периода от имеющихся данных*» и «*кроме последнего периода от имеющихся данных*».

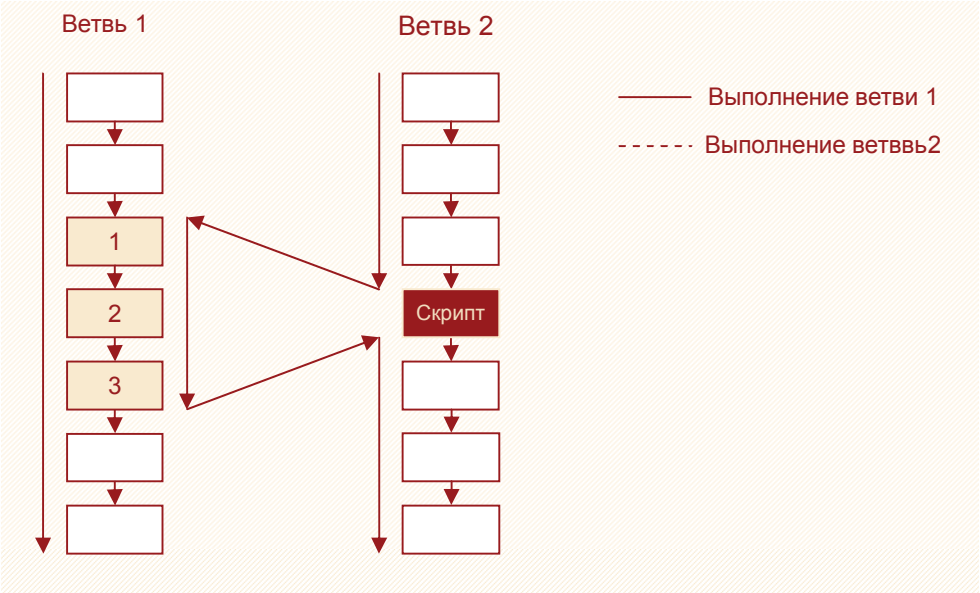
Быстрая подготовка сценариев (скрипты)

Оптимизировать требуется не только источники данных или сценарии обработки, но и работу аналитика по подготовке проектов анализа данных. Deductor включает в себя инструменты, использование которых поможет в разы ускорить создание сценариев и избавить аналитика от рутинной работы.

Ускорить разработку сценариев и предоставить возможность повторного использования однажды созданной модели призван обработчик «Скрипт». Он является аналогом функции или процедуры в языках программирования. Алгоритм работы скрипта определяется последовательностью настроенных обработчиков, входящих в его состав, а входным параметром служит поступающий набор данных.

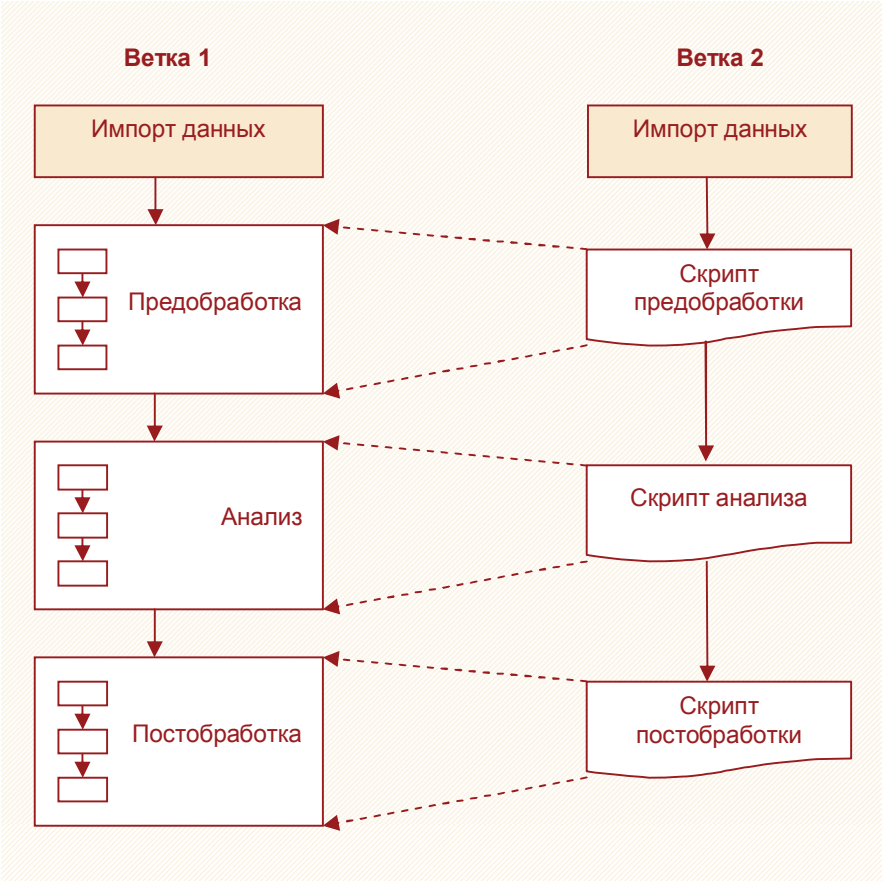
При добавлении в проект узла «Скрипт» требуется указать начальный и конечный узлы, находящиеся на одной ветви обработки. В скрипт войдут все узлы от начального до конечного включительно. В скрипте нельзя выполнить настройку отдельного узла. Скрипт является законченным блоком обработки. Изменить работу скрипта можно, только поменяв настройки узлов в его ветви-оригинале. Внесенные в нее настройки сказываются на работе скрипта. Исходя из сказанного, переобучать нейронные сети, входящие в состав скрипта, также можно только в ветке-оригинале. В состав скрипта может войти любой узел, кроме узлов импорта и экспорта данных, то есть он может включать и другие скрипты. Импорт данных не может входить в состав скрипта по тем соображениям, что основное назначение скрипта состоит в обработке нового набора данных на уже существующей модели. Поэтому не имеет смысла пропускать через него те же данные, что и в оригинальной модели.

На рисунке показана схема выполнения ветви со скриптом, включающим три узла из другой ветви сценария. Сначала (до узла со скриптом) последовательно выполняются узлы второй ветви. Затем осуществляется переход на начальный узел скрипта, находящийся в *Ветвь 1*. Далее последовательно выполняются уже узлы первой ветки, пока не будет достигнут конечный узел скрипта. После этого осуществляется возврат к *Ветвь 2* на следующий после скрипта этап обработки, и выполнение продолжается. На ход выполнения первой ветви скрипт при этом не оказывает никакого влияния.



На основе скриптов могут быть реализованы три этапа обработки данных: предобработка, анализ и постобработка.

После импорта данных в программу осуществляется их предобработка (1 этап). Предобработка может быть одинаковой для различных наборов данных (например, сглаживание, очистка, фильтрация, сортировка и т.д.), поэтому в других ветвях можно выполнить ее с помощью скрипта. Затем данные проходят через построенную аналитическую модель, например, строится прогноз на базе линейной регрессии (2 этап). Если данные подчиняются одним и тем же закономерностям, то аналитическую модель можно построить только для одного набора данных, а для остальных использовать готовую обработку на основе скриптов. После окончания этапа анализа данные подвергаются постобработке (3 этап) для того, чтобы перевести их на язык предметной области и представить пользователю в удобном виде. Этот этап также может быть реализован в виде скриптов. Схема возможного включения скриптов в ветви сценария показана на рисунке.



Любой из этих скриптов может отсутствовать, а между двумя скриптами могут находиться произвольные узлы обработки.

Дополнительное преимущество, даваемое применением скриптов в проектах, состоит в возможности избежать ошибок и легкости модернизации сразу всех моделей обработки данных. При обнаружении ошибки достаточно исправить ее в одном месте, в ветке-оригинале, и она автоматически исправляется во всех остальных ветвях, где используется скрипт. Аналогично изменение исходной модели синхронно скажется на всех ветвях обработки, построенных на скриптах.

Использование переменных

В Deductor Studio имеется возможность использовать в формулах, например, в «Калькуляторе» переменные. Эти переменные задаются в специальном окне настройки **Сервис ► Переменные**.

Данный механизм удобно использовать для случаев, когда необходимо выполнить одни и те же сценарии, но с различными параметрами. Можно настроить сценарий таким образом, чтобы в нем использовалась определяемая переменная, потом задать эту переменную и «прогнать» сценарий, ввести новое значение переменной, закрыть узлы ветки и еще раз провести обработку, но уже с новым параметром.

Переменные можно передавать программе в командной строке. Эту возможность удобно использовать при написании пакетных (***.bat**) файлов. Можно несколько раз вызвать один и тот же сценарий с различными переменными.

Обработка сценариев при помощи Deductor Server

Имеется возможность оптимизировать скорость работы при помощи Deductor Server. Deductor Server – служба Windows, способная выполнить обработку сценариев и переобучение моделей.

Делается это следующим образом.

Сначала строится сценарий при помощи Deductor Studio. Для того чтобы можно было выполнить на сервере обработку, сценарий должен содержать узлы экспорта данных. Deductor Server не интерактивное приложение, в нем отсутствует визуальная часть, поэтому получить результаты при его применении можно единственным способом – экспортировав их в какой-нибудь приемник данных. Если будет проводится серверное переобучение, то необходимо, чтобы в сценарии присутствовали обработчики допускающие переобучение, например, деревья решений или самоорганизующиеся карты.

После этого построенный файл проекта переносится на оборудование, где установлен Deductor Server. Теперь можно его использовать. Для управления и работы с сервером предназначен специальная библиотека – Deductor Client. Его применение требует написания кода программистами. Информация по этому поводу описана в SDK, поставляемый с Deductor Enterprise. Работа аналитика заключается только в написании сценариев, остальные операции должны выполнять сотрудники IT служб.

Оптимизация производительности при применении Deductor Server происходит по следующим причинам:

- Используется более производительное серверное оборудование.
- Поддерживается многопоточная обработка данных.
- Применяется специальное кэширование данных для уменьшения времени на загрузку проектов в память и повторный импорт.
- Отсутствует визуализация, обычно отнимающая значительную часть вычислительных ресурсов.

Пример создания законченного аналитического решения

Рассмотрим пример создания аналитического решения для организации, занимающейся розничной торговлей. Оно будет решать следующие задачи: консолидацию данных, то есть сбор данных о продажах из распределенных баз (например, находящихся в разных магазинах), построение аналитической отчетности, анализ потребительской корзины, прогнозирование объемов продаж, поиск оптимальной наценки и сегментацию клиентов.

Создание хранилища данных

Для анализа нам потребуется информация о продажах товаров. Будем собирать информацию, представленную следующими измерениями и фактами.

Измерения:

- *Дата* – дата покупки;
- *Клиент* – покупатель;
- *Товар* – приобретаемый товар;
- *Номер чека* – для анализа потребительской корзины.

Факты:

- *Цена товара* – закупочная цена товара – для расчета скидки и наценки в процентах;
- *Сумма наценки* – общая сумма наценки;
- *Сумма скидки* – общая сумма скидки;
- *Количество товара* – количество приобретенного за один раз товара.

Эта информация должна быть представлена тремя таблицами.

Таблица «Клиент»

N клиента	Наименование
Ключевое поле	Атрибут

Таблица «Товар»

N товара	Наименование	Группа
Ключевое поле	Атрибут	Атрибут

Таблица «Продажи»

Дата	N клиента	N товара	Номер чека	Цена товара	Сумма наценки	Сумма скидки	Количество
Измерения				Факты			

Таблицы «Клиент» и «Товар» содержат информацию о соответствующих измерениях. Необходимость в них возникла из-за наличия дополнительных атрибутов у этих измерений. Таблица «Продажа» содержит информацию о соответствующем процессе, если говорить в терминах хранилища Deductor Warehouse.

Откроем вкладку **Подключения** и создадим новое хранилище, указав путь к файлу хранилища и метку (его название, которое будет отображать при экспорте и импорте данных).

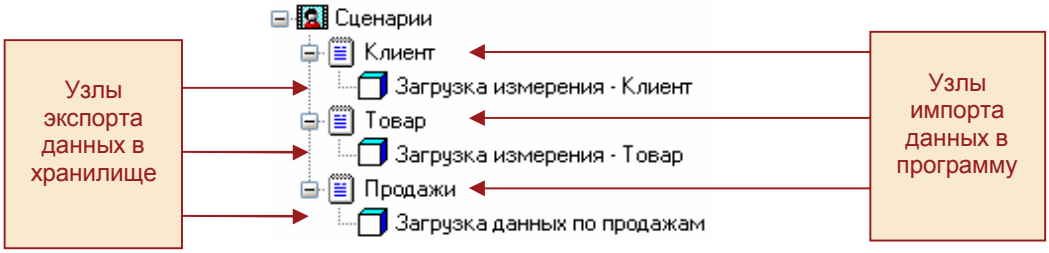
Вторым шагом будет проектирование структуры хранилища данных. Создадим при помощи редактора метаданных 4 измерения (причем 2 из них – *Клиент* и *Товар* – участвуют в иерархии) и 1 процесс.

Третьим шагом является загрузка данных в спроектированное хранилище. Для этого переместимся на вкладку **Сценарии** и вызовем мастер импорта. Предположим, что исходные таблицы содержатся в текстовых файлах с разделителем. В мастере импорта нужно будет указать, что первая строка таблицы является заголовком, указать разделитель столбцов, а также разделитель дробной части в числах. Последнее зависит от настроек операционной системы и может быть либо точкой, либо запятой. Импортируем все три таблицы в программу. Узлы импорта назовем соответственно *Клиент*, *Товар* и *Продажа*.

Четвертым шагом будет загрузка измерений *Товар* и *Клиент* в хранилище. Для этого для узлов *Клиент* и *Товар* поочередно вызовем мастер экспорта, выбрав в нем источник «Deductor Warehouse». На следующее шаге зададим измерение, в которое будем загружать данные. Далее укажем, что является уникальным идентификатором, т.е., измерением, а что атрибутами, как показано в таблицах выше.

Пятым шагом будет загрузка данных о продажах в процесс хранилища. Для этого для узла «Продажа» вызовем мастер экспорта, выбрав в нем источник «Deductor Warehouse». Выберем единственный определенный в хранилище процесс, куда планируется загружать информацию – «Продажа». Далее укажем, что является измерениями, что фактами, как показано в таблице выше. Зададим создание вспомогательной таблицы для ускорения извлечения данных. Чтобы ускорить будущие загрузки данных в хранилище, установим флажок **Удалять из хранилища, используя измерение**, и на следующем шаге мастера экспорта укажем это измерение – *Дата*.

Таким образом, мы создали сценарий загрузки данных в хранилище.



Созданный сценарий можно сохранить и использовать при загрузке новых данных в хранилище.

Для анализа данных, находящихся в хранилище, создадим новый сценарий.

Прогнозирование объемов продаж

Прогнозирование является неотъемлемой частью решения задач оптимизации. Торговые организации стремятся свести к минимуму время, которое товар лежит на складе, а также место, которое он там занимает. С другой стороны, необходимо, чтобы на складе всегда лежал требуемый в настоящее время товар. Для решения этой непростой задачи необходимо иметь данные о прогнозируемом спросе.

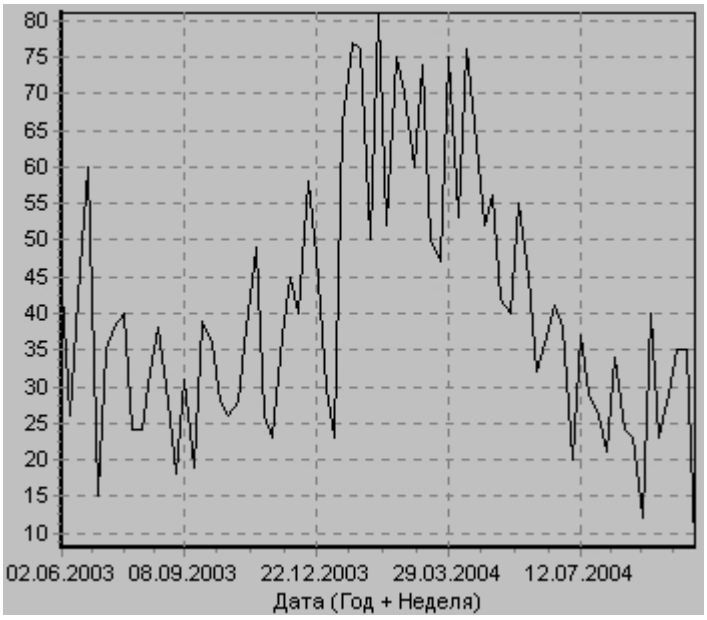
Важными сведениями для построения прогноза спроса является статистика продаж за предыдущие периоды. Такая информация есть в нашем хранилище, на основе ее и будем строить прогноз.

Для начала нужно импортировать данные из хранилища в Deductor Studio. Нас интересуют не все данные, а только количество продаваемого товара в разрезе даты и товара. Вызовем Ммстер импорта и выберем в нем источник «Deductor Warehouse». Отметим загружаемые измерения и факты. После импорта озаглавим узел «*Дата, товар, количество»*.

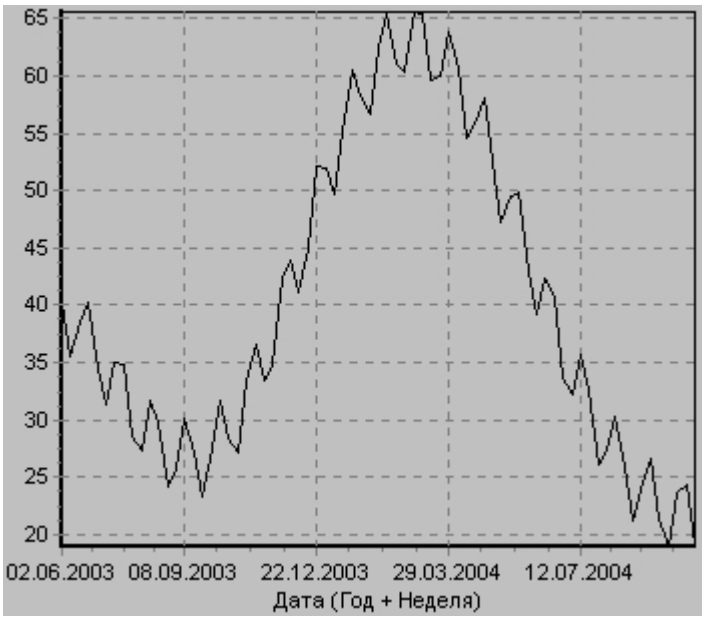
Прогнозировать мы будем понедельно. Рассматривать продажи по дням не имеет смысла, так как каждый день может очень сильно отличаться от другого по объему продаж. Но продажи по неделям разбросаны не так сильно, поэтому вторым шагом будет выделение из поля *Дата* недели. Для этого воспользуемся обработчиком «Дата и время». В таблице появится новый столбец *Дата (Год+Неделя)*.

Прогнозировать объемы по разным группам товаров не имеет смысла, так как тенденция продаж разных групп товаров может очень сильно отличаться. Поэтому сделаем фильтрацию по каждой группе товара. Для этого к узлу с преобразованием даты по неделям нужно применить обработку «Фильтрация», указав условие, например, «Группа = Группа 1».

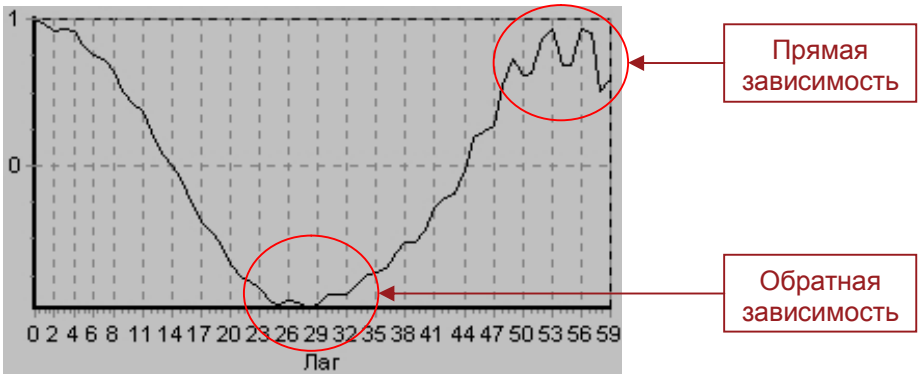
Затем следует сгруппировать количество объемов продаж по неделям. Для этого применим обработку «Группировка» к узлу с фильтрацией по группе товара. В качестве измерения укажем поле *Дата (Год+Неделя)*, в качестве факта – поле *Количество*. Таким образом, будет получена таблица с продажами товара группы 1, сгруппированные по неделям. Кривую продаж можно посмотреть на диаграмме.



Кривая продаж может содержать шумы и выбросы, которые необходимо удалить для получения более качественного прогноза. Для этого нужно воспользоваться «Парциальной обработкой» Вызовем мастер парциальной обработки для узла с группировкой. В настройках укажем для поля *Количество* вычитание шума с большой степенью вычитания. Результат такого преобразования можно посмотреть на диаграмме.



Теперь есть достаточно хорошо подготовленные для построения модели данные. Но нам необходимо определить сезонность продаж данной группы товара. Для этого служит обработчик «Автокорреляция». Применим ее к узлу с парциальной предобработкой. Будем искать зависимости в поле *Количество*. Такую зависимость следует искать, например, в течение 60 недель, то есть немного больше одного года. Для этого укажем количество отсчетов равное 60. Полученную автокорреляционную функцию можно посмотреть на диаграмме.



На ней видна обратная зависимость в течение примерно 28 недель и прямая зависимость в течение 53 недель, что соответствует половине года и году. Для построения модели будем использовать годовую зависимость объемов продаж.

При построении модели будем основываться на продажах за две предыдущие недели для того, чтобы учесть общее развитие рынка, и на продажах за две недели за аналогичный период прошлого года. Таким образом, необходимо подготовить такую выборку, чтобы в ней содержались поля.

Дата (Год+Неделя)	Количество-53	Количество-52	Количество-2	Количество-1	Количество
Неделя	Объем продаж год назад		Объем продаж предыдущие две недели		Текущий объем продаж

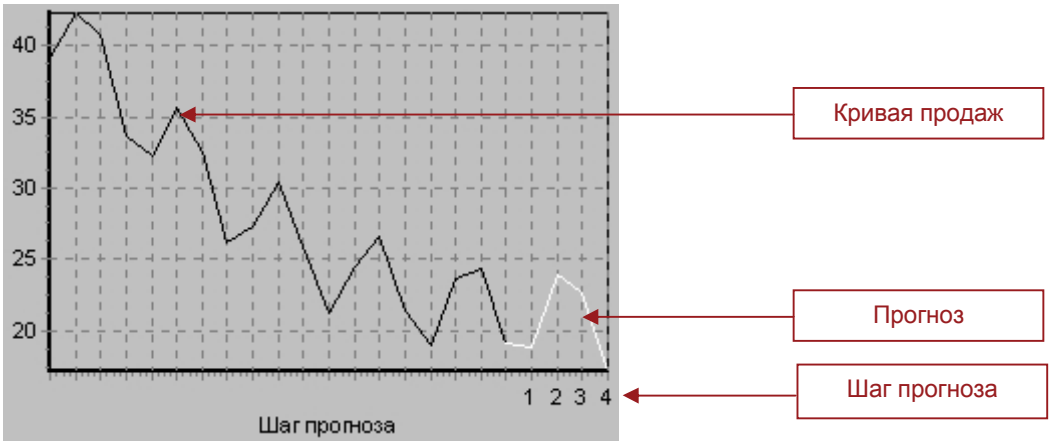
Это не что иное, как преобразование столбца «Количество» к скользящему окну. Вызовем обработку «Скользящее окно» к узлу с парциальной обработкой. Укажем столбец *Количество* используемым и установим для него глубину равной 53. Полученная таблица будет содержать столбцы не только указанные выше, но и столбцы *Количество-3» … «Количество-51*.

Построим модель, используя линейную регрессию. Вызовем после узла с преобразованием к скользящему окну обработку «Линейная регрессия». Поле *Дата (Год+Неделя)* укажем информационным, чтобы оно не участвовало в построении модели, но присутствовало в результирующей таблице. Поля *Количество-53*, *Количество-52*, *Количество-2* и *Количество-1* укажем входными. Поле *Количество* – выходным. А поля *Количество-3* … *Количество-51* укажем информационными, т.е. их можно просмотреть, но участвовать в построении модели они не будут. Пройдем дальше по мастеру обработки, оставив все параметры по умолчанию, и запустим процесс построения модели. На этом настройка линейной регрессии завершена. После построения модели можно посмотреть на диаграмме рассеяния качество построенной модели.

Теперь у нас есть модель, и можно с ее помощью узнать прогнозные значения объемов продаж на небольшой промежуток времени. Для этого вызовем обработчик «Прогнозирование» для узла линейной регрессии и укажем в ней горизонт прогноза, равный четырем, т.е. построим прогноз на 4 недели вперед. После построения прогноза результат можно посмотреть в таблице либо на диаграмме прогноза. В таблице мы обнаружим следующее.

Дата (Год + Неделя)	Количество-53	Количество-52	Количество-2	Количество-1	Количество	Шаг прогноза
20.09.2004	23,4086	27,6052	21,4422	18,9691	23,6201	
27.09.2004	27,6052	31,6763	18,9691	23,6201	24,4068	
04.10.2004	31,6763	28,3353	23,6201	24,4068	19,0879	
04.10.2004	28,3353	27,1191	24,4068	19,0879	**18,8843**	1
04.10.2004	27,1191	33,4255	19,0879	18,8843	**23,8759**	2
04.10.2004	33,4255	36,6289	18,8843	23,8759	**22,5807**	3
04.10.2004	36,6289	33,3729	23,8759	22,5807	**17,1315**	4

В поле *Количество* в последних четырех строках содержатся прогнозные значения количества продаж на четыре недели просуммированные по группе товара с кодом 1. На диаграмме прогноза можно посмотреть будущую тенденцию продаж.



Было бы интересно узнать прогноз объемов не по группе товара, а по каждому товару в группе. Это можно сделать в предположении, что пропорциональный вклад каждого товара в объемы продаж в прошлом сохранится и в будущем. Применим к узлу прогнозирования обработчик «Разгруппировка». Укажем все поля измерениями, а поле *Количество* – фактом. В поле *Исходный столбец* выберем *Количество*, в поле *Восстановить* выберем *Наименование*. Установим переключатель **По последним значениям**. В поле *Управляющий столбец* выберем *Дата (Год+Неделя)*, а в поле *Количество значений* укажем 5, т.е. будем считать пропорциональный вклад каждого товара в общее количество за последние 5 недель и использовать эту пропорцию для восстановления количества каждого товара группы.

Наглядно результаты разгруппировки можно посмотреть с помощью куба. Выберем визуализатор **Куб** и в его настройках укажем измерения *Наименование*, *Шаг прогноза* и факт *Количество*.

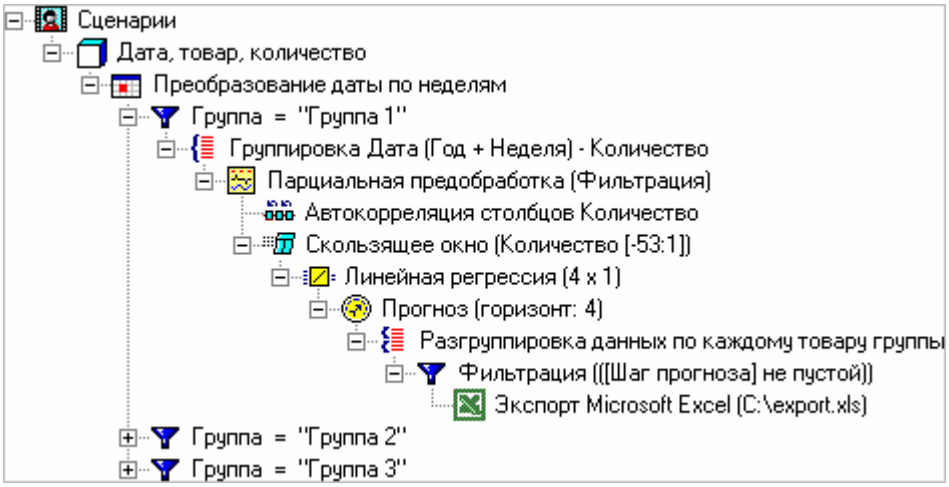
В результате получим таблицу.

	Шаг прогноза				
Наименование	1	2	3	4	Итого:
Товар 1	0.71	0.90	0.85	0.65	3.11
Товар 10	1.51	1.91	1.81	1.37	6.60
Товар 11	1.79	2.26	2.13	1.62	7.80
Товар 12	1.50	1.90	1.79	1.36	6.55
Товар 2	1.45	1.83	1.73	1.31	6.32
Товар 3	1.52	1.93	1.82	1.38	6.65
Товар 4	1.44	1.82	1.72	1.30	6.28
Товар 5	1.71	2.16	2.04	1.55	7.46
Товар 6	2.10	2.65	2.51	1.90	9.16
Товар 7	1.46	1.85	1.75	1.33	6.39
Товар 8	2.00	2.53	2.39	1.82	8.74
Товар 9	1.70	2.14	2.03	1.54	7.41
Итого:	18.89	23.88	22.57	17.13	82.47

Как видно, значения «Итого» для каждого шага прогноза совпадают со значениями из таблицы прогноза. Теперь можно сделать вывод, какой товар необходим на складе в течение следующих четырех недель. Это важный шаг на пути решения задачи оптимизации складов. Однако сам Deductor Studio не решает задачи оптимизации. Это необходимо делать в других программах, которые используют прогнозные значения для решения задачи оптимизации. Таким образом,

возникает необходимость экспортировать результаты прогноза во внешний файл. Но при прогнозировании мы добавили в результирующую выборку исходные данные. Это те строки таблицы, где значение поля «Шаг прогноза» равен пустому значению. Вызовем для узла разгруппировки обработку фильтрации и укажем в ней условие *«Шаг прогноза не пустой»*. В результате будет получена таблица, готовая для экспорта. Вызовем для этого последнего узла фильтрации мастер экспорта и выберем источник, например, «Microsoft Excel». Далее следует указать экспортируемые поля таблицы. Укажем поля: «Количество», «Шаг прогноза» и «Наименование». В итоге будет сформирован файл Excel с прогнозными значениями количества продаваемого товара на следующие 4 недели.

Такую процедуру можно повторить на остальных группах товара. В результате всех этих действий будет получен сценарий прогнозирования объемов продаж.



Поиск оптимальной наценки

Предоставление скидки покупателям является стимулом для увеличения объемов закупок. Чем больше продается некоторого товара, тем больше прибыль. С другой стороны, чем больше предоставляется скидка, тем меньше наценка на товар, и тем меньше прибыли приносят продажи этого товара. Для нахождения оптимальной скидки необходимо построить модель зависимости спроса от процентной ставки скидки.

В нашем хранилище отсутствуют данные о скидке в процентах и прибыли, зато есть информация, с помощью которой их можно вычислить.

Импортируем в программу из хранилища все факты в разрезе измерений: *Дата*, *Клиент*, *Товар*, *Номер чека*.

Применим к узлу обработку «Калькулятор». Назовем выражение «Скидка» и в окно формулы введем выражение соответствующее формуле:

Сумма скидки / (Сумма скидки + Цена).

Вводить выражение можно с помощью мыши, выбирая соответствующие поля в окне со списком столбцов. Затем добавим еще одно выражение, назвав его «Прибыль». В окне формулы необходимо ввести выражение, соответствующее формуле

Сумма наценки – Сумма скидки.

В формулах нельзя использовать метки полей, так как они описаны выше, нужно применять имена полей. Эти имена указаны рядом с метками в списке доступных полей. Таким образом формула

Сумма скидки / (Сумма скидки + Цена)

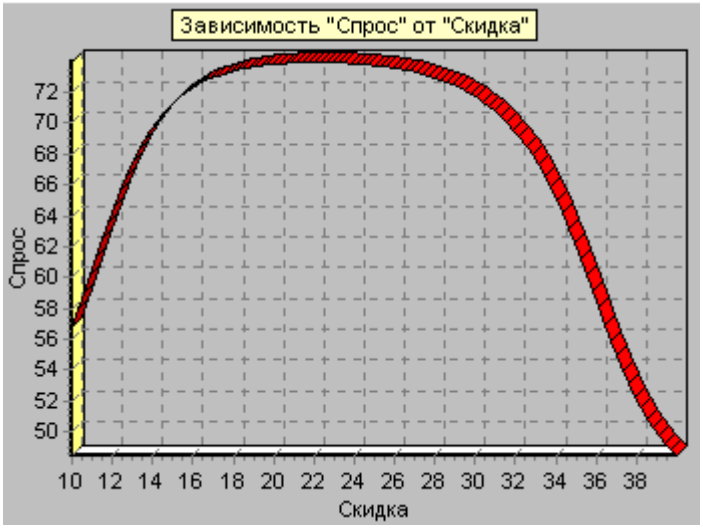
будет написана как

Discount_Sum / (Discount_Sum + Price).

В результате работы обработки в таблицу будет добавлено два поля *Скидка* и *Прибыль*.

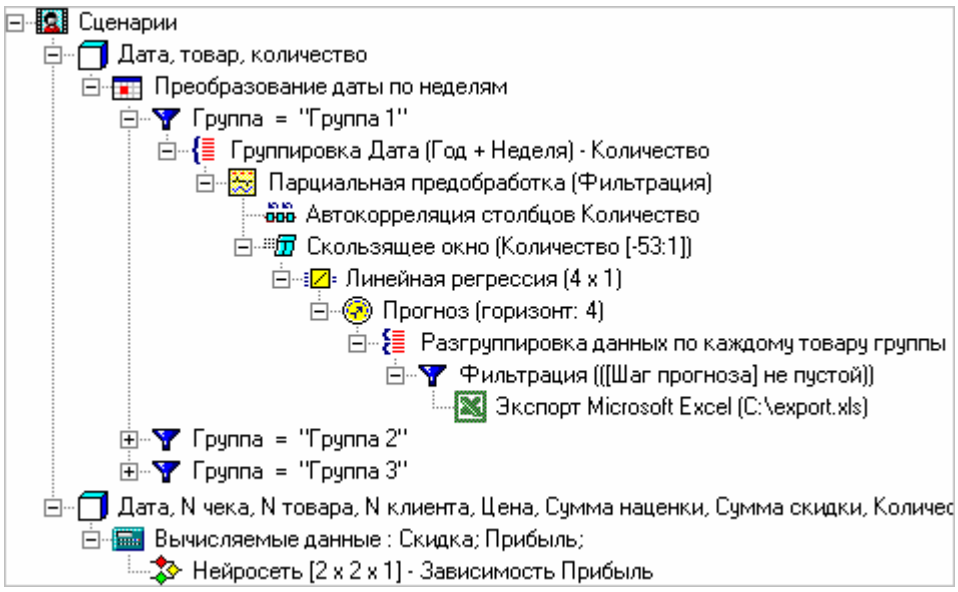
Будем проверять гипотезу, что спрос зависит от *абсолютной цены товара* и *процента скидки*. Так как зависимость спроса от цены и скидки может быть нелинейной, воспользуемся нейронными сетями. Применим эту обработку к узлу с вычисляемыми данными. Укажем поле *Скидка* и *Цена* входным, а поле *Прибыль* – выходным. Остальные поля сделаем неиспользуемыми. Запустим алгоритм, оставив все остальные настройки без изменения. Оценить качество построенной зависимости можно с помощью диаграммы рассеяния. Если прогнозируемое алгоритмом значение прибыли разбросано относительно истинных значений, то необходимо изменить настройки нейросети, например, увеличив число слоев или нейронов в слое или изменив функцию активации.

Нас же интересует результат, представленный диаграммой «Что-если».



Эта диаграмма наглядно показывает, при каком значении скидки с заданной ценой товара достигается максимум прибыли.

В результате сценарий будет выглядеть следующим образом.



Анализ потребительской корзины

Обычно клиенты покупают не один товар, а несколько. Причем эти товары могут быть каким-то образом взаимосвязаны. Например, человек, приобретающий дверь, скорее всего, купит еще и дверные ручки. А если он приобретает дверь конкретного вида, то и ручки он купит соответствующие этой двери. Зачем нужна такая информация? Например, для размещения товара на прилавках в виде, более удобном для покупателя. Зачем идти в другой конец магазина, чтобы посмотреть, какие ручки подойдут к понравившейся двери? Или другой вариант – покупатель приобрел дверь, но забыл про ручки, тогда продавец, зная такие правила приобретения товара, может сам их предложить.

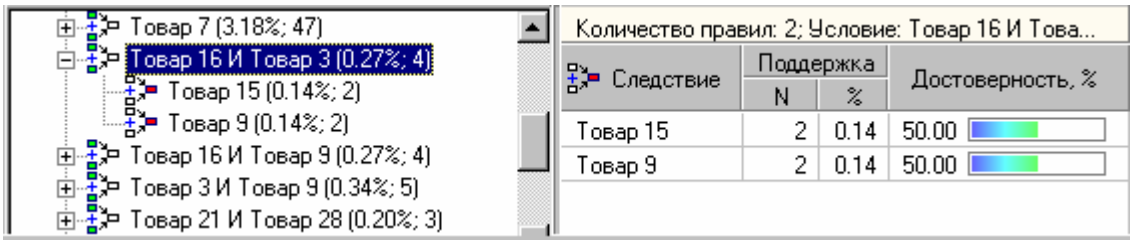
Информация о товарах, приобретаемых совместно, содержится в нашем хранилище в поле «Номер чека». Найдем правила совместного приобретения товаров с помощью инструмента «Ассоциативные правила».

Импортируем в программу из хранилища количество товара в разрезе номера чека и товара. Применим к этому узлу ассоциативные правила. Полю «Номер чека» выберем назначение «Транзакция», а полю наименование – назначение «Элемент».

Выполним сначала алгоритм, не изменяя других настроек. После его выполнения возможны следующие варианты:

- Найдены только одноэлементные множества. Это факт видно на диаграмме на странице обучения в Мастере. На диаграмме будет отображен только один столбец. В этом случае нужно вернуться на шаг назад и уменьшить минимальную поддержку либо увеличить максимальную поддержку и так до тех пор, пока не будут получены двух и более элементные множества.
- Количество правил равно нулю, что отображается в соответствующем поле на странице обучения Мастера. Тогда нужно вернуться на шаг назад и уменьшить минимальную достоверность либо увеличить максимальную.

В результате будут получены ассоциативные правила, которые можно посмотреть с помощью визуализаторов «Правила», «Популярные наборы», «Дерево правил» и «Что-Если». Например, дерево правил может выглядеть так.



Если человек приобрел вместе *Товар 16* и *Товар 3*, то он скорее всего купит еще и *Товар 15* с достоверностью 50% и/или *Товар 9* с достоверностью 50%.

Аналитическая отчетность

С помощью OLAP-куба можно быстро получать аналитические отчеты на основе данных, содержащихся в хранилище.

Проведем в качестве примера ABC-анализ клиентов. Для этого нам понадобится информация о продажах в разрезе клиентов. Вызовем мастер визуализации для узла «Вычисляемые данные: Скидка; Прибыль», созданный в разделе «Поиск оптимальной наценки». Выберем визуализатор «Куб». В настройках куба укажем, что поле *Наименование клиента* является измерением, а поле *Прибыль* –фактом. Разместим измерение в строках. Полученный куб будет выглядеть так:

Наименование	Прибыль
Клиент 11	11500
Клиент 14	10700
Клиент 2	12357
…	…

Эта таблица содержит всех клиентов. Чтобы оставить только клиентов группы A, нужно вызвать «Селектор», указать в нем фильтрацию факта *Прибыль* по измерению *Наименование* и выбрать «Сумма» в списке функций. В поле *Условие* выбрать «Доля от общего», а в поле *Значение* указать 50. После этого в таблице останутся только клиенты, приносящие в сумме 50% прибыли. Это и есть клиенты группы A. Если сделать то же самое, указав значение 80%, то будет получена таблица с клиентами групп A и B.

Аналогичный ABC-анализ можно провести и для товаров, то есть выделить наиболее выгодные товары.

Иногда перед построением куба требуется некоторая предобработка данных. Например, нужно отследить постоянных, новых и утерянных клиентов. Это можно сделать, основываясь на информации о количестве обращений клиентов в месяц. Например, если клиент на протяжении нескольких месяцев не обращался в организацию, то его будем считать утерянным. Если клиент обращается примерно одинаковое число раз каждый месяц, то его можно считать постоянным.

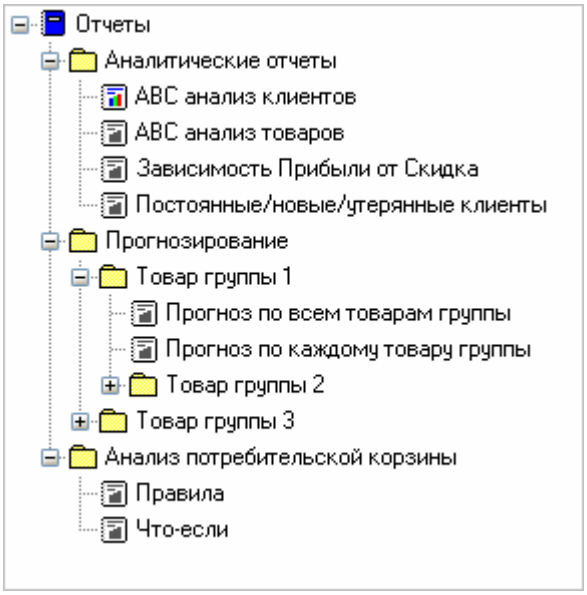
Применим к узлу «Вычисляемые данные: Скидка; Прибыль» преобразование даты и заменим дату покупки месяцем этой даты. Для нового узла выберем визуализатор «Куб». Укажем в нем измерения *Дата (Год+Месяц)*, *Наименование*, а факт – *Номер чека*. Для факта сразу выберем функцию агрегации *Количество*. Разместим *Наименование* в строках, а *Дата (Год+Месяц)* – в столбцах. Полученная таблица будет иметь вид.

Наименование	Дата (Год+Месяц)			
	01.06.2003	01.07.2003	01.08.2003	01.09.2003
Клиент 1				2
Клиент 2	2	3	1	2
Клиент 3	3			

Клиент 1 – новый клиент на сентябрь 2003 года. *Клиент 2* – постоянный клиент. *Клиент 3* – ушедший.

Создание отчетности

Весь сценарий, построенный выше, обычно готовит аналитик организации. Конечным же пользователям необходимо быстро получить доступ к результатам анализа. Для этого в программе предназначена панель **Отчеты**. Откроем эту панель и создадим на ней дерево отчетов с папками «Аналитические отчеты», «Прогнозирование», «Анализ потребительской корзины». В папки добавим ссылки на нужные узлы сценария обработки. В результате получим дерево отчетов.



Для каждого отчета настраивается свой способ отображения. Это удобно, так как одному узлу дерева сценариев может соответствовать несколько узлов дерева отчетов.

На этом создание решения закончено, и оно готово к использованию.

Выше был приведен всего лишь один из вариантов решения поставленной задачи. В анализе данных всегда существует много путей достижения одной цели. Тем не менее, базовые вещи остаются неизменными, и при создании большинства проектов будут пройдены те же самые этапы, что и в этом примере.

Что делать при возникновении ошибок

В ходе обработки данных с использованием Deductor могут возникать различные ошибки. Иногда определить причину возникновения той или иной ошибки нелегко, но существуют некоторые рекомендации, которые помогут определить место и причину появления ошибки и исправить ее. Локализация и устранение ошибок в равной мере ложатся на плечи аналитика и администратора системы.

В процессе выполнения сценария обработки возможно появление различных ошибок, о которых Deductor будет сообщать пользователю. Рассмотрим способы решения некоторых из них.

Первое, что следует сделать при появлении окна с сообщением об ошибке – это установить источник сообщения. Если сообщение об ошибке исходит от операционной системы, значит, возникли серьезные неполадки в ее работе или работе программы. К сообщениям операционной системы, в частности, относятся все, связанные с ошибками доступа к памяти (access violation). В этом случае следует перезапустить программу, возможно, с перезагрузкой компьютера. Если ошибка не исчезает, отправьте описание своих действий, текст сообщения об ошибке, конфигурацию используемых аппаратных и программных средств и, по возможности, набор входных данных разработчикам программы по адресу deductor@basegroup.ru. Описывать действия, приведшие к появлению ошибки, следует как можно более подробно, чтобы разработчики смогли их повторить и установить причину возникновения. Информация о способах устранения ошибки будет выслана в ответном письме.

Эти же действия следует выполнить, если Deductor некорректно самостоятельно завершает работу («падает») или зависает.

Если сообщение об ошибке исходит от Deductor, следует определить место и причину ее возникновения. Ошибка, как правило, возникает при выполнении какого-либо определенного узла. В интерактивном режиме этот узел можно определить вручную, последовательно выполняя все узлы сценария. В пакетном режиме – с помощью подробного лога, в который записывается информация о выполнении каждого узла. После определения места возникновения ошибки следует внимательно ознакомиться с текстом сообщения. В нем обычно есть краткая информация о причинах ошибки. Например, сообщение «*Столбец XXX должен существовать в исходном источнике данных*» говорит о том, что столбец, обрабатываемый в узле, был удален из набора данных, либо у него просто поменялось имя. Такая ошибка возникает при перенастройке вышестоящих узлов или появлении изменений в источнике данных. Для ее устранения нужно либо откатить внесенные перенастройкой изменения, либо перенастроить узел, вызывающий ошибку, на работу с новым набором данных.

Часто ошибки вызываются тем, что в наборе данных появляются пустые значения (NULL-значения). Многие обработчики не способны работать с полями, содержащими пустые значения. Это касается, прежде всего, обработчиков группы Data Mining. Пустые значения могут появиться из-за изменений в источнике данных, перенастройки узлов, особенностей обработки некоторых граничных значений в узлах и т.д. От пустых значений следует избавляться с помощью фильтрации или табличной замены. Фильтрация полностью убирает из набора данных строки, содержащие пустые значения в указанных полях. С помощью же табличной замены можно поменять пустые значения на другие, нейтральные для дальнейшей обработки, например, на ноль.

При импорте данных из текстового файла и экспорте данных в хранилище по окончании операции возможно появление лог-файла, открываемого в текстовом редакторе. При импорте такой лог создается в случае появления ошибок преобразования типов. Например, в качестве разделителя целой и дробной части числа была указана точка, в то время как в действительности им является запятая. В результате Deductor не сможет выполнить преобразование прочитанного из файла числа к вещественному типу и добавит сообщение в лог. При экспорте в хранилище лог создается при наличии в наборе данных NULL-значений или в случае, когда тип загружаемых данных не соответствует типу данных объекта хранилища.

Так как Deductor активно работает с графической информацией, то в системе может наступить дефицит графических ресурсов. Это вызовет появление бесконечного числа окон с сообщениями

об ошибках. В такой ситуации следует по возможности закрыть окна визуализаторов и другие приложения, выполняющиеся в системе. После этого сообщения об ошибках пропадут. В критических случаях поможет принудительная перезагрузка системы.

Часто возникают ситуации, связанные с ошибкой доступа к источнику данных. Например, когда недоступен нужный сетевой ресурс, локальный источник данных был перемещен в другой каталог, переименован настроенный источник данных. В сообщении об ошибке в этом случае обычно находится достаточно информации для локализации и устранения ошибки. При получении сообщения об ошибке вида «*Хранилище данных с именем XXX не найдено*», «*Файл XXX не найден*» и т.п. следует проверить указанный источник данных на существование и доступность и внести соответствующие изменения в настройки источника.

При работе сценария с хранилищем Deductor Warehouse для импорта данных используются идентификаторы объектов хранилища. Идентификаторы создаются автоматически при создании новых объектов, но их можно заменить на другие. В тех случаях, когда структура хранилища создается заново, у объектов с теми же параметрами и назначениями, что и в предыдущем хранилище, могут оказаться другие идентификаторы. В результате созданные ранее сценарии не смогут работать с новым хранилищем, выводя сообщения об ошибках при импорте. Таким образом, для корректной работы сценария необходимо, чтобы одинаковые объекты у нового и старого хранилища имели одинаковые имена. Для внесения изменений в существующие объекты хранилища, в частности их переименования следует использовать Редактор метаданных.

Заключение

Выше рассмотрены вопросы, связанные с построением аналитических систем.

На рынке программного обеспечения существует множество разрозненных приложений, предназначенных для консолидации и анализа данных. Построение аналитической системы в этом случае происходит следующим образом. Для очистки данных используется отдельная программа. Данные для нее готовятся в специальном формате. После очистки, обработанные данные сохраняются в какой-либо файл или базу данных. Далее, для построения моделей используются другие программы, для которых данные также должны быть представлены в определенном формате и сохранены в отдельный файл. Обычно эти программы содержат собственные способы визуализации результатов. Но чтобы получать различные сводки данных необходимо опять использовать отдельную программу для построения OLAP куба. Причем для каждой программы необходимо где-то сохранять настройки. Как видно такой подход к построению аналитической системы достаточно неудобен и требует много времени. Нередко не обходится без написания дополнительных программ, обеспечивающих взаимодействие различных аналитических модулей.

Deductor позволяет пройти все шаги построения аналитических систем в рамках единой платформы. Имеется возможность интеграции с различными источниками данных: файлы различных форматов, базы данных, собственное хранилище данных. Результаты обработки представляются в виде таблиц и к ним снова можно применять различные методы обработки. Нет необходимости использовать промежуточные источники данных. Архитектура построения сценариев позволяет произвести практически любой анализ. Deductor является универсальной платформой и никак не привязан к какой-либо предметной области. Он позволяет пройти все этапы общего для любой предметной области алгоритма извлечения полезных знаний из данных.

Наличие самообучающихся алгоритмов позволяет легко адаптировать построенную систему к специфике работы предприятия. Возможность переобучить модель на новых данных обеспечивает переносимость сценария в филиалы организации.

Широкий спектр визуализаторов позволяет получать результаты анализа в удобном для восприятия виде. OLAP-куб, как один из методов визуализации данных, позволяет получать разнообразные срезы данных и сам предоставляет аналитические возможности. Он строится на основе таблиц данных, поэтому его можно получить на любом шаге сценария. Анализ «что-если» позволяет протестировать построенную модель на новых данных.

Экспорт данных дает возможность интегрировать программу с существующими корпоративными учетными системами.

Наличие в Deductor всех необходимых инструментов анализа, минимальные сроки разработки готовых решений, возможность наращивать и адаптировать созданное решение гарантируют быстрое получение качественного результата и превосходную отдачу в будущем.

Дополнительные источники

1. http://www.basegroup.ru – большое количество статей по вопросам анализа данных и применяемым при этом алгоритмам, примеры эффективного использования методов анализа данных в бизнесе, доступные для скачивания библиотеки компонентов для анализа данных.
2. http://edu.basegroup.ru – модульный дистанционный учебный курс «Корпоративные аналитические системы».
3. http://forum.basegroup.ru – форум, посвященный проблемам прогнозирования и анализа данных при помощи современных технологий.
4. http://glossary.basegroup.ru – глоссарий по терминологии анализа данных, в котором можно уточнить значение непонятного термина.
5. http://www.kdnuggets.com – англоязычный портал по всем вопросам Data Mining, Knowledge Discovery, Genomic Mining и Web Mining.

Контакты

Адрес:

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

Телефоны: +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 – дистанционное обучение