Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Макаренко Елена Николаевна
Должность: Ректор
Дата подписания: 29.07.2022 18:15:41
Уникальный программный ключ:
c098bc0c104...

Базовые навыки работы в Deductor Studio 5.2

Системы поддержки принятия решений (СППР) – компьютерные автоматизированные системы, целью которых является помощь людям, принимающим решение в сложных условиях, для полного и объективного анализа предметной деятельности. СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных.

В СППР используются разные методы: информационный поиск, интеллектуальный анализ данных, поиск знаний в базах данных, рассуждение на основе прецедентов, имитационное моделирование, эволюционные вычисления и генетические алгоритмы, нейронные сети, ситуационный анализ, когнитивное моделирование и др. Некоторые из этих методов были разработаны в рамках искусственного интеллекта. Близкие к СППР классы систем – это экспертные системы и автоматизированные системы управления.

Знание принципов построения современных систем поддержки принятия решений (СППР) на основе технологий хранилищ данных (Data Warehousing), оперативного анализа (OLAP) и добычи данных (Data Mining) для аналитической поддержки процессов принятия решений является необходимым компонентом в подготовке IT-специалистов.

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.

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.

Deductor 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 шести 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. Analytical 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;
6. Integration Server – веб-сервис, функционирующий поверх аналитической службы Deductor Analytical Server.

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

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 Analytic Server (DAS) – сервер удаленной аналитической обработки. Он позволяет выполнять на сервере операции «прогона» данных через существующие сценарии и переобучение моделей. DAS ориентирован на обработку больших объемов данных и работу в территориально распределенной системе. Для управления подключениями к DAS используется специальное приложение Deductor Admin.

Deductor Application Server (DIS) – веб-сервис, функционирующий поверх аналитической службы Deductor Analytical Server. Это механизм обмена данными со сторонними приложениями, реализации сервис-ориентированной архитектуры (SOA) и встраивания в корпоративные системы, построенные на базе сервисной шины предприятия (Enterprise Service Bus).

Обеспечивает большую гибкость интеграции, простоту масштабирования и переноса.

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

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.

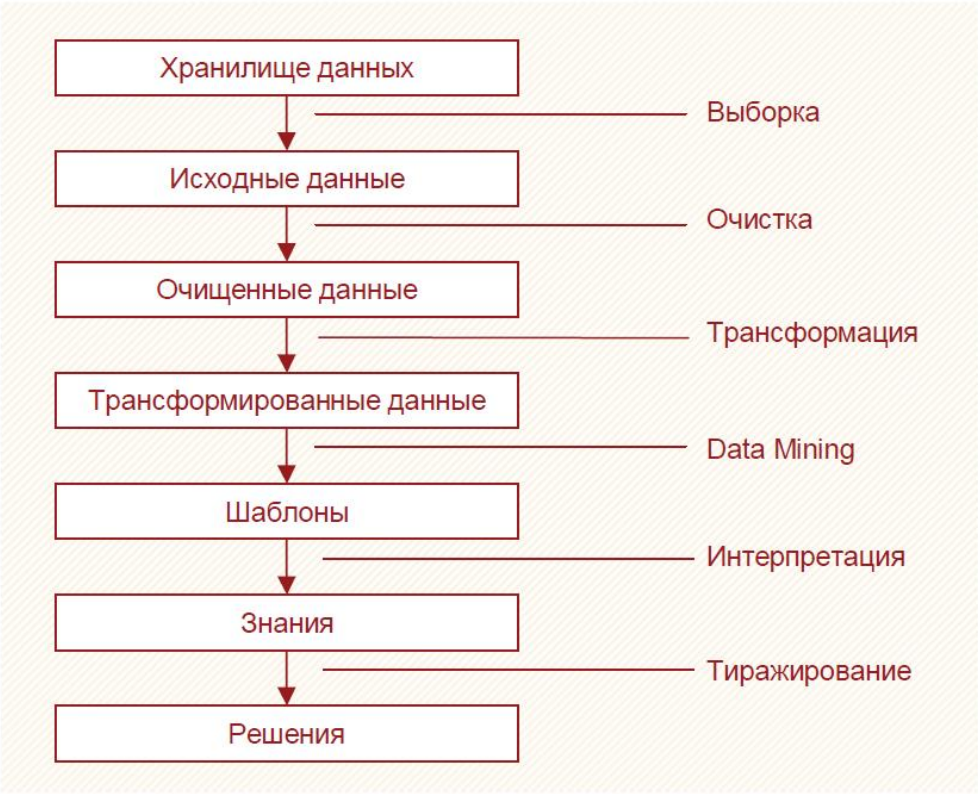
Привлекательность этого подхода заключается в том, что вне зависимости от предметной области мы применяем одни и те же операции:

1. Подготовка исходного набора данных. Этот этап заключается в создании набора данных, в том числе консолидации 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 поиска зависимостей, новых знаний, или говорят, что строятся модели. Выделяют два больших класса моделей – описательные и предсказательные. Они решают различные задачи: классификацию, регрессию, кластеризацию, установление ассоциаций и т.д. Для этого используются как классические статистические методы, так и самообучающиеся алгоритмы и машинное обучение (нейронные сети, деревья решений и др.).

5. Постобработка данных. Тестирование, интерпретация результатов и практическое применение полученных знаний в выбранной прикладной области.

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.

Одной из наиболее важных функций любой аналитической системы является поддержка процесса тиражирования знаний, то есть 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 четыре задачи:

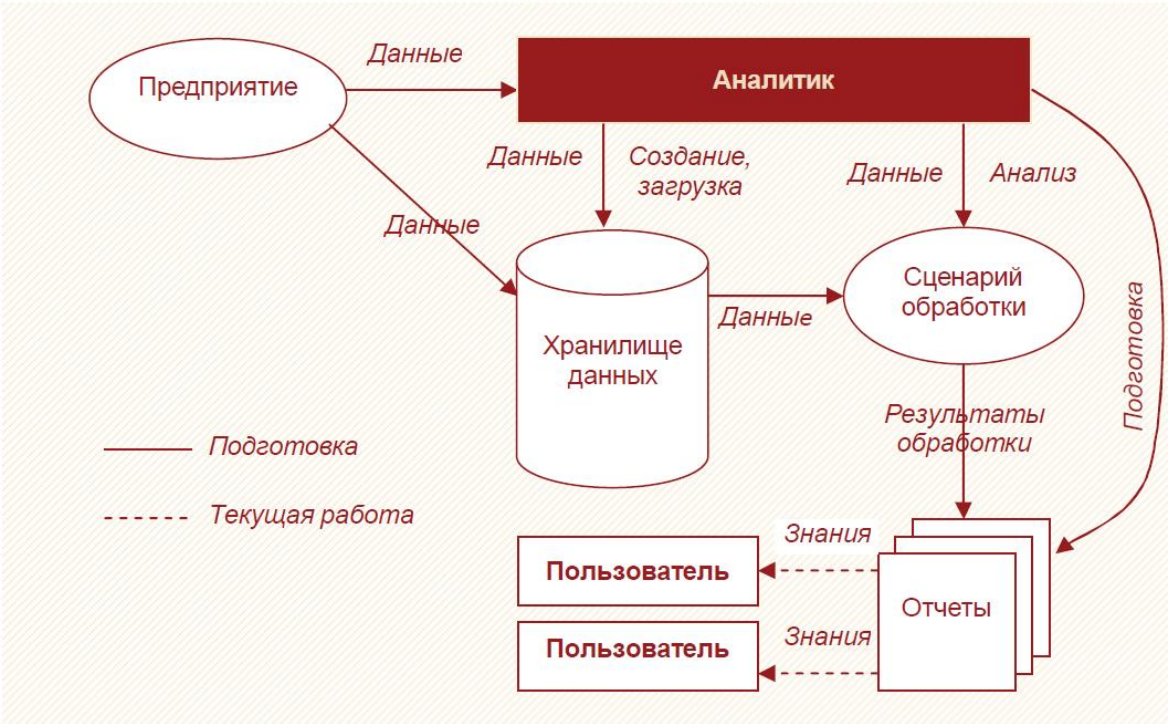
1. Аккумулировать данные, необходимые для анализа, обеспечив унифицированный простой способ получения любой информации, необходимой для анализа.
2. Формализовать знания экспертов. Трансформировать знания экспертов в модели, пригодные для автоматизированной обработки.
3. Подобрать способы визуализации 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.

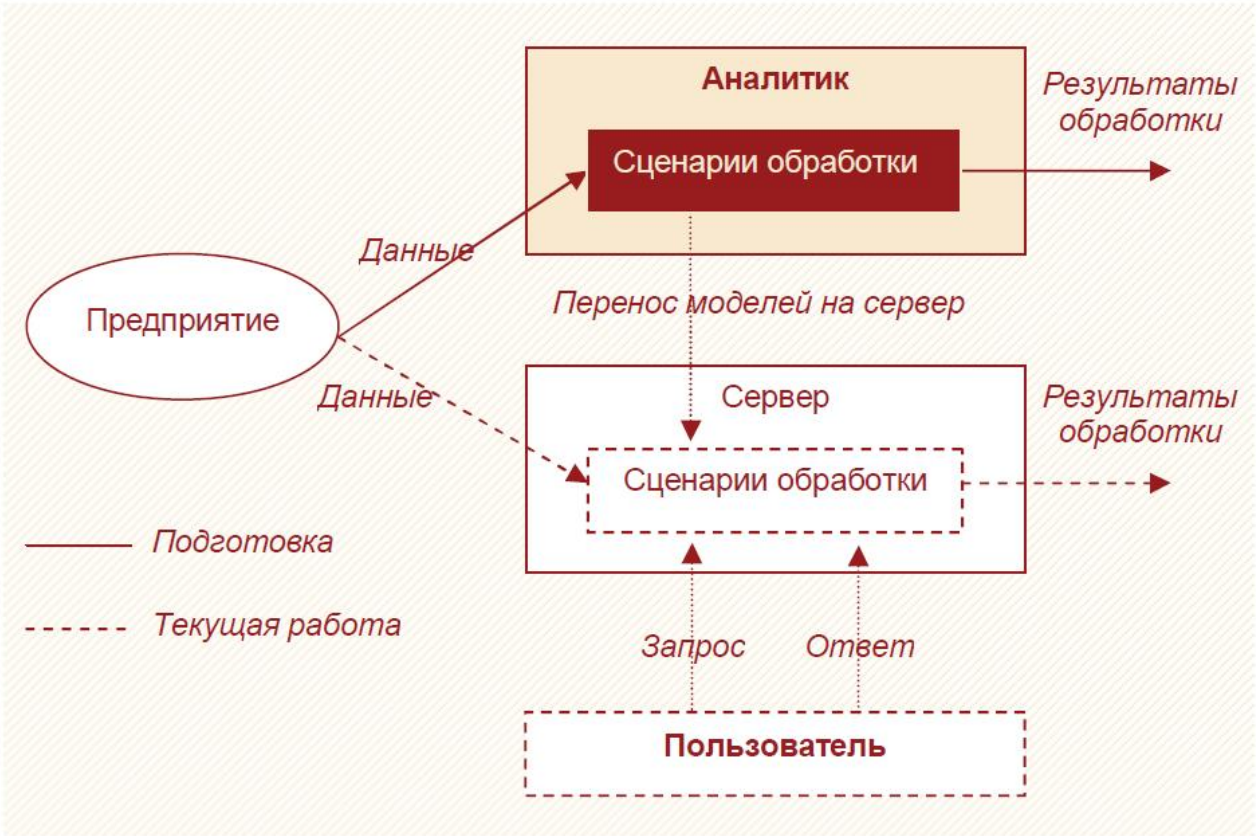
Делается это следующим образом:

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

Работа в интерактивном режиме



Работа в автоматическом режиме



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 конструирования 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 (аналитическому и/ или интеграционному) 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.

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

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

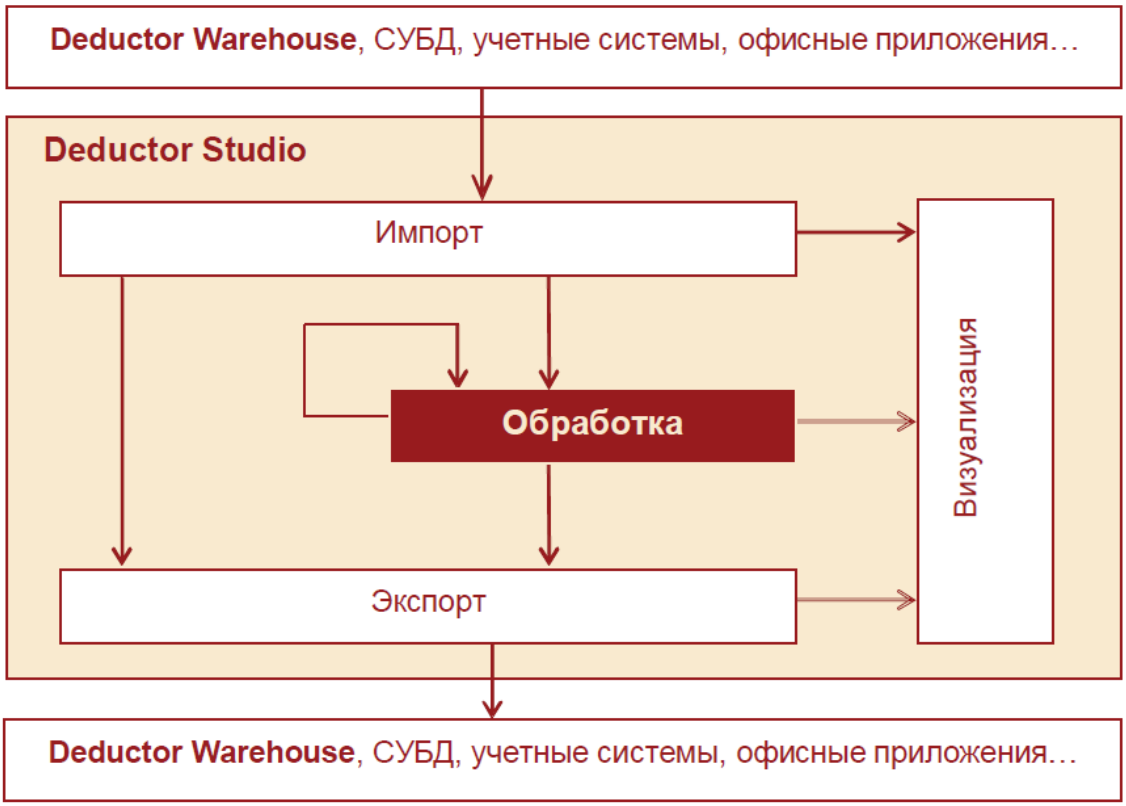


схема функционирования Deductor Studio

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 во

вне. 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.

Существует три типа варианта поставки платформы Deductor:

- Enterprise;
- Professional;
- Academic.

В зависимости от типа поставки набор доступных компонентов может различаться.

Версия Enterprise предназначена для корпоративного использования. В ней присутствуют:

- Серверные компоненты Deductor Server и Deductor Client.
- Интерфейс доступа к Deductor через механизм OLE Automation.
- Традиционное хранилище данных Deductor Warehouse на трех СУБД: Firebird, MS SQL, Oracle.
- Виртуальное хранилище данных Deductor Virtual Warehouse.

Версия Professional предназначена для небольших компаний и однопользовательской работы. В ней отсутствуют серверные компоненты, поддержка OLE, виртуальное хранилище, а традиционное хранилище данных можно создавать только на СУБД FireBird. Автоматизация выполнения сценариев обработки данных осуществляется только через пакетный режим.

Версии Professional и Enterprise требуют установки драйверов Guardant для работы с лицензионным ключом.

Версия Academi с предназначена для образовательных и обучающих целей. Ее функционал аналогичен версии Professional за исключением:

- отсутствует пакетный запуск сценариев, т.е. работа в программе может вестись только в интерактивном режиме;
- отсутствует импорт из промышленных источников данных: 1С, СУБД, файлы MS Excel, Deductor Data File;
- некоторые другие возможности.

Категории пользователей Deductor:

В процессе развертывания и использования аналитической платформы с ней взаимодействуют различные категории пользователей. Можно выделить четыре основные категории:

- аналитик;
- пользователь;
- администратор;

- программист. Функции аналитика:
- создание в Deductor Studio сценариев – последовательности шагов, которую необходимо провести для получения нужного результата.
- построение, оценка и интерпретация моделей.
- настройка панели отчетов для пользователей Deductor Viewer.
- настройка сценария на поточную обработку новых данных.

Функции пользователя:

- просмотр готовых отчетов в Deductor Viewer.

Функции администратора:

- установка компонентов Deductor на рабочих местах и сервера ключей Guardant при необходимости.
- развертывание традиционного хранилища данных на сервере.
- контроль процедур регулярного пополнения хранилища данных.
- конфигурирование сервера Deductor Server.
- настройка пакетной и/или серверной обработки сценариев Deductor.
- оптимизация доступа к источникам данных, в том числе к хранилищу данных.

Функции программиста:

- интеграция Deductor с источниками и приемниками данных.
- вызов Deductor из внешних программ различными способами, в том числе взаимодействие с Deductor Server.

Такая работа как проектирование и наполнение хранилище данных часто выполняется коллективно аналитиком, администратором и программистом. Аналитик проектирует семантический слой хранилища данных, то есть определяет, какие данные необходимо иметь в хранилище. Администратор создает хранилище данных и наполняет его данными. Программист при необходимости создает программные модули, выполняющие выгрузку информации из учетных систем в промежуточные источники (так называемые транспортные таблицы).

Установка Deductor

Установку Deductor рекомендуется проводить администратору системы, однако, при наличии прав администратора в Windows это может сделать и аналитик. Установка может быть произведена на компьютер с операционной системой MS Windows 2000 и выше. Системные требования к компьютеру изложены в справочной системе.

Для установки Deductor Professional/Academic запустите файл инсталлятора и следуйте инструкциям по установке. На странице **Выбор компонентов** программы установки предоставляется выбор, какой набор компонентов пакета Deductor необходимо установить на компьютер. В выпадающем списке можно выбрать предопределенные конфигурации установки платформы, и программа установки сама предложит нужный набор компонентов.

После установки программ серии **Professional** и **Enterprise** дополнительно потребуется настроить работу с электронным ключом защиты от копирования.

Установку и подсоединение электронного ключа осуществляет администратор.

Существуют два вида ключей – локальный и сетевой. Локальный ключ устанавливается на том же компьютере, что и Deductor, и работать с ним можно только с этой рабочей станции. Сетевой ключ устанавливается на сервере, и к нему могут подключаться несколько пользователей одновременно (количество пользователей ограничивается типом приобретаемой лицензии).

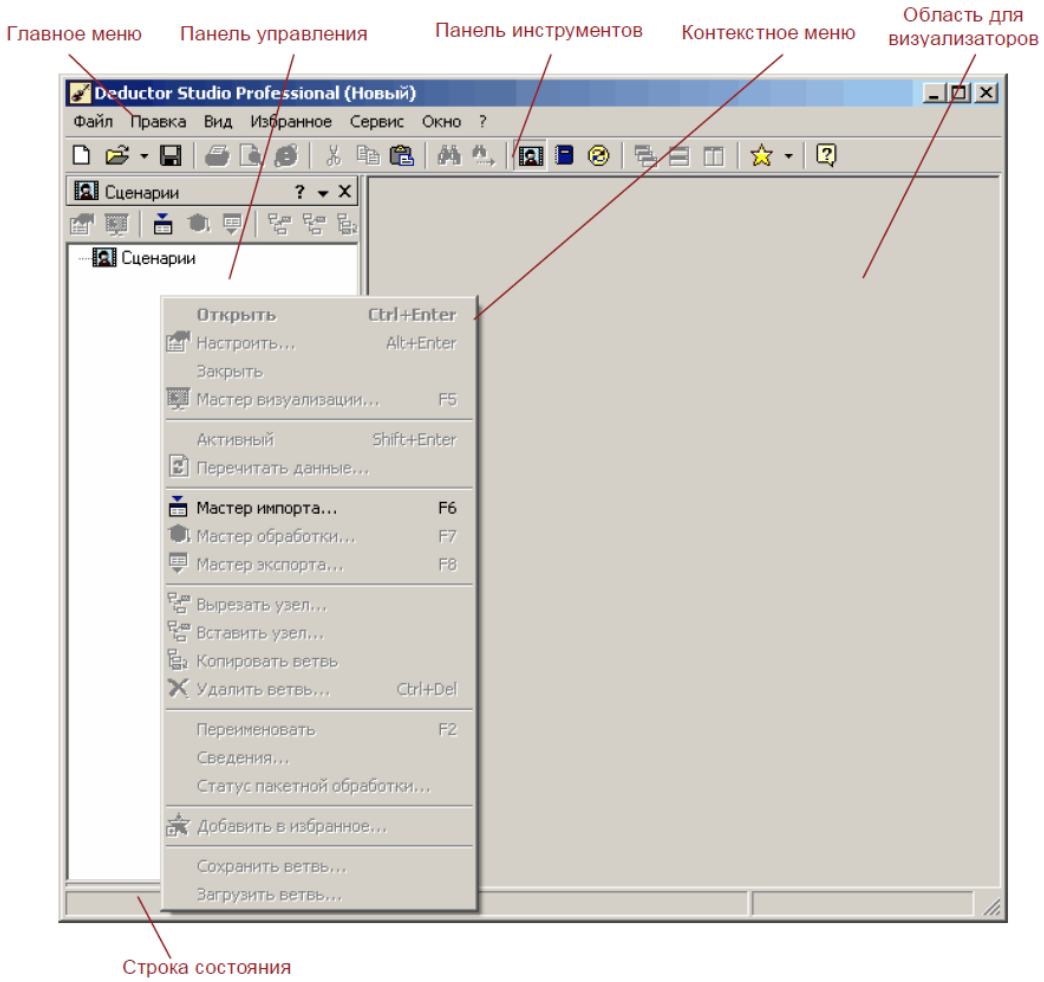
При каждом запуске Deductor пытается найти доступный электронный ключ. В случае если ключ не найден, могут появиться следующие сообщения об ошибке:

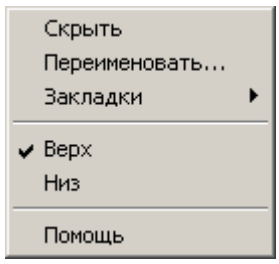
При наличии таких ошибок следует обратиться к администратору.

Начало работы с системой

Главное окно Deductor Studio

После запуска главное окно Deductor Studio выглядит следующим образом.





- Скрыть – делает вкладку невидимой;
- Переименовать – переименовывает название вкладки;
- Закладки – переключается на выбранную закладку;
- Верх/Низ – задает расположение названий вкладок: вверху либо внизу;
- Помощь – открывает раздел справки.

Понятие проекта

В Deductor Studio ключевым понятием является проект. Это файл с расширением *.ded, по структуре соответствующий стандартному xml-файлу. Он хранит в себе:

- последовательности обработки данных (сценарии);
- настроенные визуализаторы;

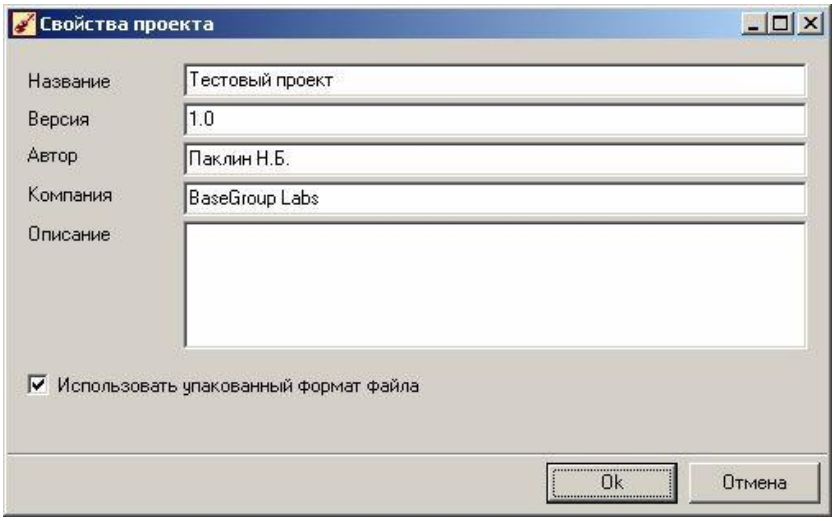
- переменные проекта и служебную информацию.

Пример фрагмента файла *.ded:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document>
    <Version>
        <Comments>Deductor Studio Enterprise</Comments>
        <CompanyName>BaseGroup Labs</CompanyName>
        <FileDescription>Deductor Studio Enterprise</FileDescription>
        <FileVersion>5.2.0.50</FileVersion>
        <InternalName>Deductor Studio Enterprise</InternalName>
        <LegalCopyright>BaseGroup</LegalCopyright>
        <LegalTrademarks>BaseGroup</LegalTrademarks>
        <OriginalFilename>DStudio.exe</OriginalFilename>
        <ProductName>Deductor Studio Enterprise</ProductName>
        <ProductVersion>5.2</ProductVersion>
    </Version>
```

По умолчанию файл проекта Deductor при сохранении запаковывается, что позволяет уменьшить его размер, поэтому просмотреть запакованный файл в виде xml невозможно. Для этого нужно снять опцию Использовать упакованный формат файла в диалоговом окне Свойства проекта (меню Файл ► Свойства проекта…)

Каждый проект имеет авторские сведения: Название, Версия, Автор, Компания, Описание. Они заполняются в диалоговом окне **С войства проекта** (меню **Файл►Свойства пр о екта …**).



Создать новый проект можно следующими способами:

- главное меню **Файл►Со зд ать** ;
- кнопка **Со зд ать но вы й проек т** на панели инструментов;
- клавиша **Ctrl + N** . Открытие существующего проекта:

- главное меню **Ф ай л► О ткр ы ть** ;
- кнопка **Открыть прое кт** на панели инструментов;
- клавиша **Ctrl + O**.

Открыть проект можно еще одним способом – в главном меню **Файл► История** найти имя проекта. Способ работает в том случае, если вы недавно открывали этот проект, и он сохранился в менеджере историй проектов.

В одной запущенной копии Deductor Studio можно открыть только один проект.

Для сохранения проекта под текущим именем нужно выбрать главное меню **Фай л ► Со хр анить** , нажать кнопку  или комбинацию **Ctrl + S**.

Для сохранения текущего проекта под другим именем: главное меню **Файл►Сохранить как…**

Мастера

В Deductor Studio вся работа ведется с использованием пяти мастеров:

- Мастер импорта;
- Мастер экспорта;
- Мастер обработки;
- Мастер визуализации;
- Мастер подключений.

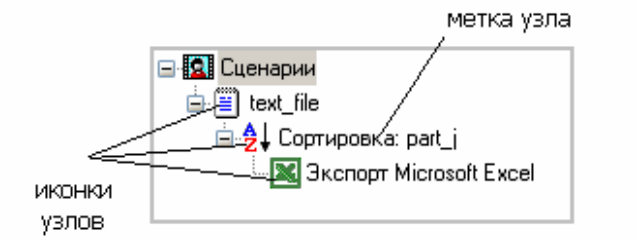
С помощью мастеров импорта, экспорта и обработки формируется сценарий. Сценарий состоит из узлов. Мастер подключений предназначен для создания настроек подключений к различным источникам и приемникам данных. Мастер визуализации настраивает визуализаторы для конкретного узла.

Визуализатором называется любое представление набора данных в каком-либо виде: табличном, графическом, описательном. Примеры визуализаторов: таблица, дерево, гистограмма, диаграмма, OLAP-куб и т.д.

Сценарии

Понятие сценария и узла обработки

В Deductor Studio для аналитика основополагающим понятием является сценарий. Сценарий представляет собой последовательность операций с данными, представленную в виде иерархического дерева. В дереве каждая операция образует узел, заголовок которого содержит: имя источника данных, наименование применяемого метода обработки, используемые при этом поля и т.д. Кроме этого, слева от наименования узла стоит значок, соответствующий типу операции.



Если узел имеет подчиненные узлы, то слева от его названия будет расположен значок «+», щелчок по которому позволит развернуть узел, т.е.

сделать видимыми все его подчиненные узлы, при этом значок «+» поменяется на «–». Щелчок по значку «–», наоборот, сворачивает все подчиненные узлы.

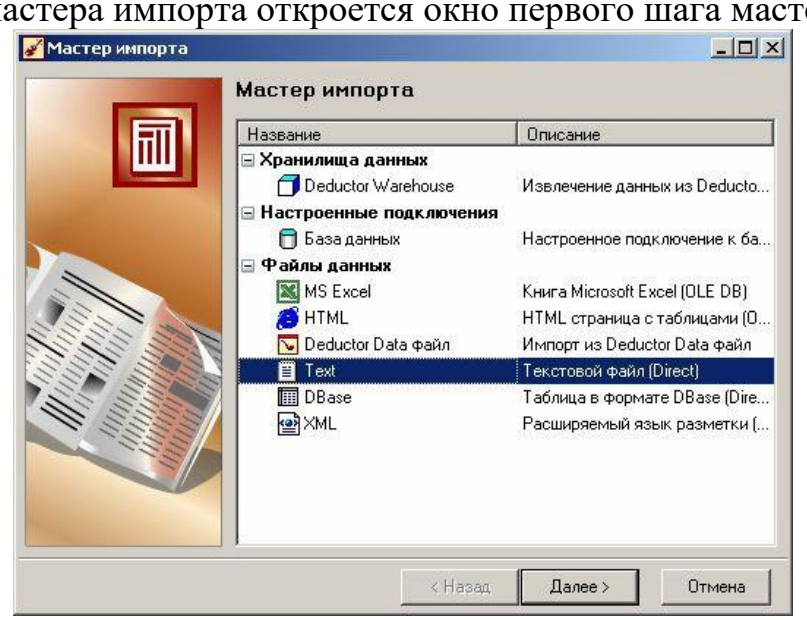
С помощью клавиш **Ctrl** + ↑ и **Ctrl** + ↓ можно перемещать узлы по дереву вверх-вниз в пределах подчинения родительскому узлу.

Сценарий состоит из *ветвей*. **Deductor** не имеет собственных средств для ввода данных, поэтому сценарий *всегда* начинается с узла импорта из какого-либо источника. Любой вновь создаваемый узел импорта будет находиться на верхнем уровне (подчиненным главному узлу Сценарии).

Создание нового узла импорта осуществляется с помощью **мастера импорта**. Вызвать мастер можно следующими способами:

- кнопка  на панели инструментов закладки **Сценарии** ;
- клавиша **F6** ;
- контекстное меню **Мастер импо рта . .**

При вызове мастера импорта откроется окно первого шага мастера.



В нем все источники данных сгруппированы по следующим четырем категориям:

- хранилища данных;
- настроенные подключения;
- файлы данных;
- бизнес-подключения.

Некоторые категории могут отсутствовать в списке. Причинами этого может быть следующее:

- Версия Deductor. Например, категории **Настроенные подключения** и **Бизнес - подключения** отсутствуют в версии Academic.
- В дереве подключений (вкладка **Подключения**) не зарегистрировано ни одного объекта из данной категории. Например, если не настроено ни одного подключения к хранилищу данных, то категория **Хранилища данных** будет отсутствовать.

- Отключена «видимость» объекта или категории объекта (подробнее об этом см. в разделе **Настройка конфигурации Deductor Studio**).

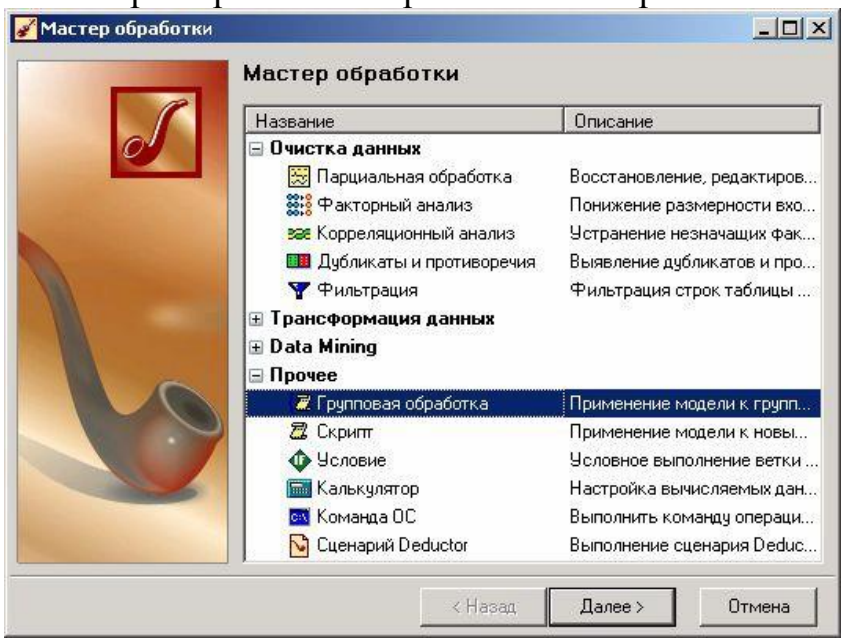
Дальнейшие шаги мастера импорта будут зависеть от того, какой объект дерева категорий был выбран аналитиком.

К любому узлу импорта можно добавить узел *обработки* или узел *экспорта,* предварительно выделив узел импорта мышью. Новый узел будет добавлен как подчиненный к узлу импорта.

Создание нового узла обработки осуществляется с помощью **мастера обработки**. Вызвать мастер можно следующими способами:

- кнопка  на панели инструментов закладки **Сценари и** ;
- клавиша **F7** ;
- контекстное меню **Мастер обработки ...**

При вызове мастера обработки откроется окно первого шага мастера.



В нем все обработчики сгруппированы по следующим четырем категориям:

- Очистка данных;
- Трансформация данных;
- Data Mining;
- Прочее.

Некоторые узлы могут отсутствовать в списке. Причины этого следующее:

- версия Deductor;
- отключена «видимость» объекта (или целой категории) объекта;
- узел «устарел» и в текущей версии Deductor его создание невозможно (допускается только его чтение и настройка).

Создание нового узла экспорта осуществляется с помощью **мастера экспорта**. Вызвать мастер можно следующими способами:

- кнопка  на панели инструментов закладки **Сценарии** ;

- клавиша **F8** ;
- контекстное меню **Мастер экспорта. . .**

В нем все приемники данных сгруппированы по следующим 5 категориям:

- хранилища данных;
- базы данных;
- файлы;
- Web-серверы;
- прочее.

Причины отсутствия некоторых объектов или категорий мастера экспорта аналогичны тем, что перечислены при описании мастера импорта.

После узла экспорта невозможно добавить ни один узел.

Базовые операции над узлами сценария

Кроме команд вызова мастеров, к каждому узлу применимы базовые операции. Операции над узлами и ветками сценария можно выполнять следующими способами:

- кнопки панели инструментов на закладке **Сценар ии** ;
- контекстное меню;
- мышь.

Список доступных операций.

1. *Открытие узла* – узел запускается на выполнение, причем выполняются все родительские узлы, а справа открываются визуализаторы, настроенные для данного узла. В интерактивном режиме для каждого узла должен быть настроен хотя бы один визуализатор, например, Таблица или Сведения. Операция вызывается:
   - двойной щелчок мышью на узле;
   - клавиши **Ctrl+ Enter** ;
   - контекстное меню Открыть.
2. *Настройка узла* – вызывается мастер импорта, мастер обработки или мастер экспорта, в зависимости от типа узла, для изменения параметров обработки, производимой в узле. Операция вызывается:
   - кнопка  ;
   - клавиши **Alt+ Enter** ;
   - контекстное меню **Настроить ….**
3. *Активация/деактивация узла* – узел может быть либо активным, либо неактивным. Если узел неактивный, то, сделав его активным, выполнится сценарий для этого узла, но визуализаторы отображены не будут. Делая узел неактивным, закрываются все визуализаторы для него и для всех подчиненных узлов, а сам узел и подчиненные узлы

превращаются в неактивные. Эта операция может быть использована для освобождения памяти. Операция активации/деактивации вызывается:

- клавиши **Shift+Enter** ;
- контекстное меню **Активный …**

4. *Перечитать данные узла* – все узлы до корневого включительно будут закрыты, а затем выполнена ветка сценария от корневого до текущего узла. Операция вызывается:
   - контекстное меню **Перечитать данные …**
5. *Вырезать узел* – удаляет текущий узел из сценария обработки. Все его потомки при этом перемещаются на один уровень вверх и начинают подчиняться родителю удаленного узла. Операция вызывается:
   - кнопка  ;
   - контекстное меню **Вырезать узел** .
6. *Вставить узел* – вставляет перед текущим узлом сценария новый узел и вызывает для него мастер обработки. Вставить узел перед узлом импорта данных нельзя. Операция вызывается:
   - кнопка  ;
   - контекстное меню **Встави ть узел** .

После вставки нового узла или удаления существующего узлы-потомки могут стать неработоспособными, в зависимости от обработки, выполняемой новым узлом.

7. *Копировать ветвь* – копирует ветвь сценария, начиная с текущего узла и включая все его потомки. Операция вызывается:
   - кнопка  ;
   - контекстное меню **Копировать ветвь** ;
   - при помощи механизма drag & drop – выделив узел, и, удерживая нажатой клавишу **Ctrl** , указать курсором мыши на новый узел, который должен стать родителем старого. При этом переносимая ветка целиком скопируется в новое место.
8. *Удалить ветвь* – удаляет узел сценария и все его подузлы. Удаленная ветвь восстановлению не подлежит, поэтому к данной операции необходимо подходить с осторожностью. Операция вызывается:
   - кнопка  ;
   - клавиши **Ctrl + Del** ;
   - контекстное меню Удалить ветвь.
9. *Перенос ветви* – переносит ветку сценария к новому узлу. Операция производится аналогично копированию ветви с помощью drag & drop без удерживания клавиши **Ctrl** .
10. Переименовать – позволяет изменить метку текущего узла. Операция вызывается:
    - клавиша **F2** ;
    - контекстное меню **Переименовать . . .**

11. *Сведения* – открывает диалоговое окно **Сведения** для текущего узла. В нем редактируется имя, метка и описание к узлу. Операция вызывается:
    - контекстное меню **Сведения . ..** ;
    - открыв скрытую панель узла с помощью кнопки  и нажать там одну из кнопок:

Имя , **Метка** или **Описание** .

Имя узла может быть задано только латинскими символами, тогда как метка – любыми. Кроме того, имя узла должно быть уникально в пределах одного сценария. Как правило, необходимости в переименовании имен узлов не возникает.

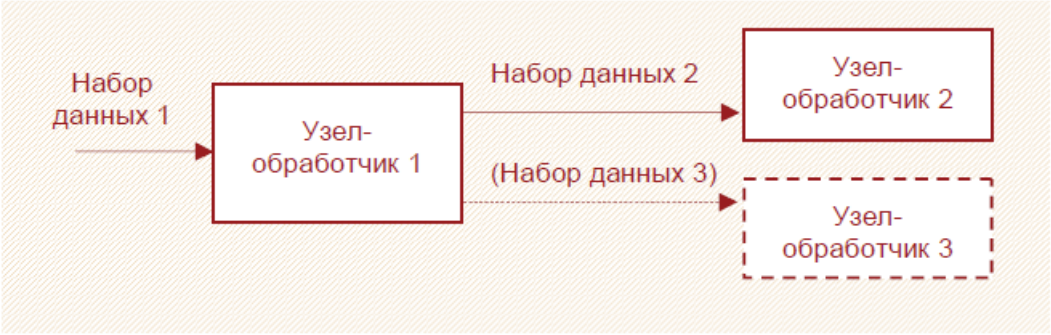
12. *Статус пакетной обработки* – устанавливает статус пакетной обработки для узла.
13. *Добавить в Избранное* – текущий узел добавляется в список избранных узлов.
14. *Сохранение ветви* – вызывается стандартный диалог Сохранение, в котором можно указать путь и имя файла для сохранения ветви сценария, начинающейся с текущего узла. Операция вызывается:
    - контекстное меню **Сохранить ветвь** .
15. *Загрузка ветви* – вызывает стандартный диалог Открытие файла, в котором можно указать путь и имя файла, хранящего ветвь сценария. Загруженная ветвь сценария станет потомком текущего узла. Ветвь, начинающаяся с узла импорта данных, будет добавлена в проект как новая корневая ветвь. Операция вызывается:
    - контекстное меню **Загрузить ветвь** .

По умолчанию ветвь сценария имеет расширение ***. deb** .

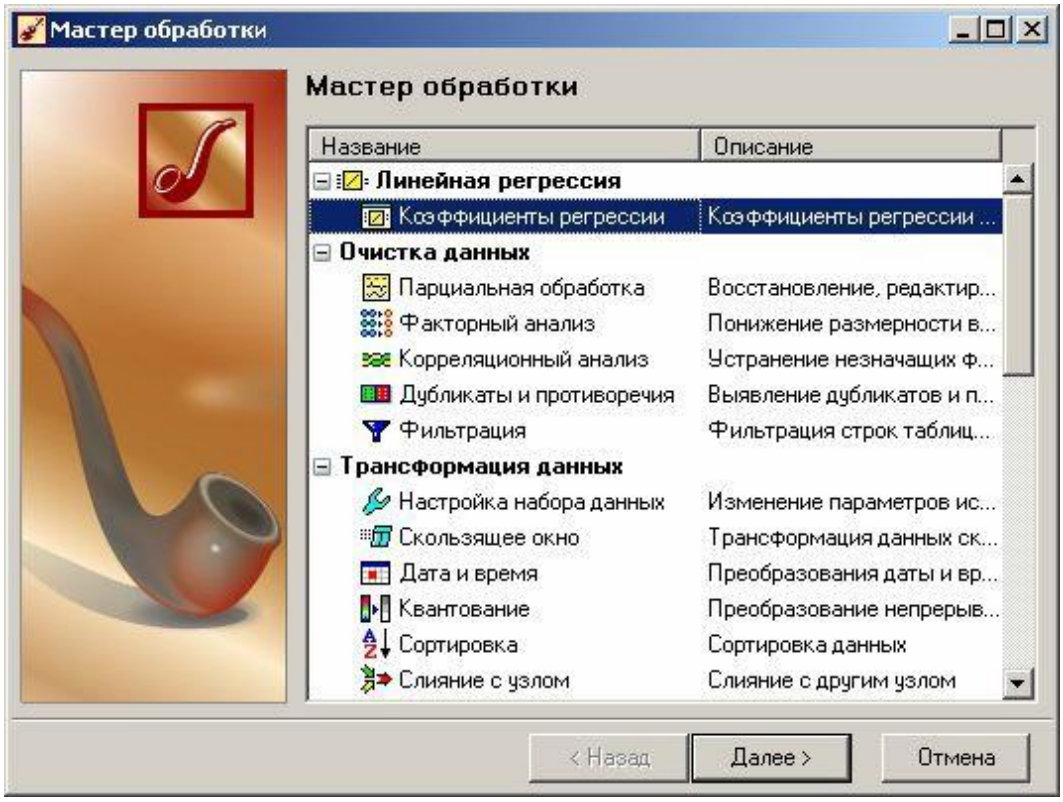
Взаимодействие узлов друг с другом

В Deductor взаимодействие узлов друг с другом спроектировано на уровне программного ядра, поэтому принцип взаимодействия един и не зависит от типа узла.

Каждый узел можно представить «черным ящиком», на вход которого подается структурированный набор данных с полями, а на выходе доступен один или несколько обработанных узлом наборов данных. Обработка может вестись любая – от простой сортировки до моделирования. Выходной набор, в свою очередь, можно снова подать на вход узла. Так конструируется сценарий.



На выходе узла может присутствовать не один набор, а несколько (на рисунке такой дополнительный набор данных обозначен пунктирной стрелкой). Например, в результате работы узла **Линейная регрессия** образуются два набора данных: один – таблица рассчитанных результатов, а другой – коэффициенты регрессии. Эти коэффициенты можно просмотреть в визуализаторе под таким же названием, но иногда нужно использовать коэффициенты в сценарии для дальнейшей обработки. Поэтому при добавлении любого узла появляется возможность «переключиться» на другой набор данных, если он присутствует в предыдущем узле. Вот как это выглядит в мастере обработки.



В Deductor Studio 5.2 узлами, которые выдают на выходе более одного набора данных, являются: **Линейная регрессия** , **Логистическая регрессия** , **Ассоциативные правила**, **Корреляционный анализ**.

Импорт из текстовых файлов с разделителями

Структурированный текстовый файл с разделителями – один из самых распространенных форматов хранения данных. Такой файл представляет собой обычный текстовый файл, столбцы данных в котором разделены однотипными символами-разделителями, например символами табуляции, пробела, точки с запятой и т.д.

Процесс импорта данных из текстового с разделителями файла в мастере импорта (категория **Текстовой файл (Direct))** содержит следующие шаги:

- указание имени файла;
- настройка параметров импорта;
- настройка импортируемых полей;
- запуск процесса импорта;
- выбор способа визуализации;
- задание сведений об узле.

На шаге **Указание имени файла** , нажав кнопку ..., необходимо выбрать имя текстового файла (расширения ***. txt** , ***. csv**), из которого следует выполнить импорт данных. После этого в поле «Имя файла» окна Мастера импорта появится имя выбранного файла и путь.

Допускается вручную ввести путь к файлу в строке поля Имя файла.

Имеется возможность использовать как абсолютные, так и относительные пути для файлов. Они указываются относительно текущей директории Deductor. При открытии Deductor текущей директорией является директория файла проекта. Поэтому, если файл проекта и текстовые файлы располагаются в одной папке, то использование относительных путей в Мастере импорта позволит не перенастраивать узлы импорта при изменении расположения папки на жестком диске.

Здесь также доступны настройки:

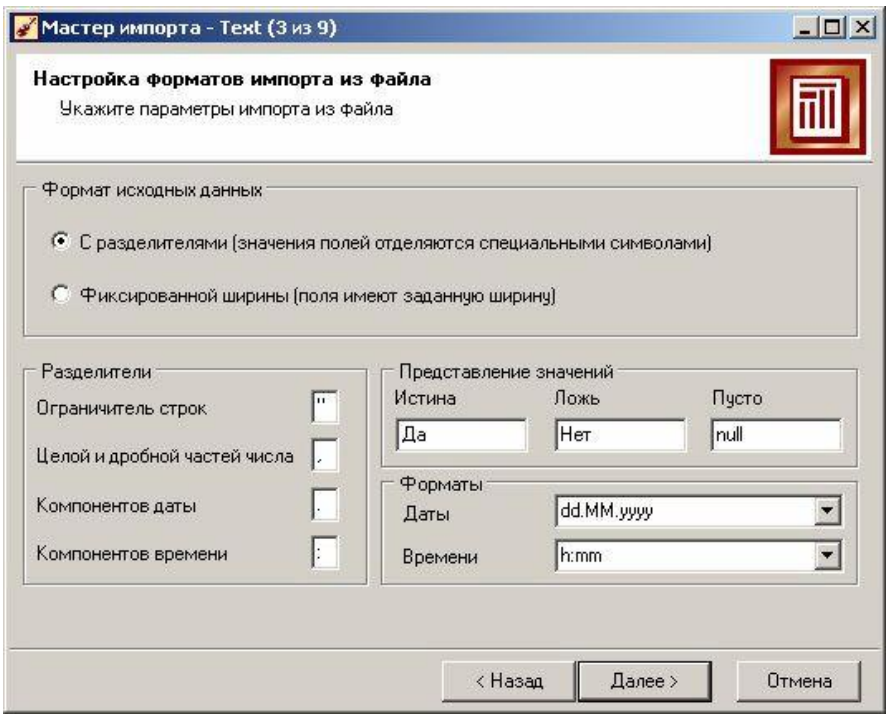
- **Начать импорт со строки** – номер строки, начиная с которой будет делаться импорт данных из файла.

- флаг **Первая строка является заголовком** – установка флажка означает, что узел будет импортировать данные с учетом того, что все записи первой строки являются заголовками столбцов.
- **Кодировка** – ANSI (Windows) или ANCII (MS DOS).

На шаге **Настройка параметров импорта** нужно настроить параметры импорта данных из текстового файла, так как существует несколько форматов структурированных текстовых файлов. Доступные опции:

- переключатель **Формат исходных данных** , который определяет символ- разделитель в файле (например: символ табуляции, пробел, запятая). Разделитель чаще всего присутствует. Если же нет, то нужно выбрать переключатель **Фиксированной ширины (поля имеют заданную шири ну)** , а позже установить ширину каждого поля.
- **Ограничите ль строк** – при задании данного параметра необходимо указать, какой именно ограничитель строкового значения нужно использовать при импорте данных из текстового файла. Обычно таким ограничителем является символ двойной кавычки " .
- **Разделитель дробной и целой части числа** – при задании данного параметра необходимо указать символ, разделяющий дробную и целую части в числовых значениях, содержащихся в файле.
- **Разделитель компонентов даты** – указывается символ, разделяющий компоненты даты в соответствующих значениях, содержащихся в файле.
- **Разделитель компонентов времени** – указывается символ, разделяющий компоненты времени в соответствующих значениях, содержащихся в файле.
- **Форматы Даты / Времени** – указываются форматы даты/времени, используемые в импортируемом файле.

Представление значений – опция для полей логического типа, которое может принимать одно из трех значений – истина (true), ложь (false) и пустое значение (null). Определяет регламент записи в эти значения. Так, при настройках по умолчанию для любого логического поля значение Да будет восприниматься как истина, Нет – как ложь.



В качестве разделителей, представлений значений и форматов по умолчанию *всегда предлагаются* системные настройки операционной системы. Поэтому при импорте необходимо обращать внимание на их соответствие формату в импортируемом текстовом файле.

Следующее окно мастера зависит от установленного переключателя в флажке **Формат исходных данных** . Если был выбран формат **С разделителями**, то появится вкладка, на которой нужно явно указать символ-разделитель (по умолчанию – табуляция). Здесь же находится флаг **Считать последовательные разделители одним** – в случае последовательно идущих символов-разделителей они будут восприниматься за один. Такое бывает, например, когда символом-разделителем выступают несколько пробелов.

Предпросмотр текстового файла в виде таблицы внизу (загружаются только первые 10 строк) позволяет убедиться в корректности выбора настроек импорта даже не запуская его.

Лабораторная работа № 1. Знакомство с Аналитической Платформой **«Deductor (АП DD)**

Целью выполнения данной лабораторной работы является:

- получение первоначальных сведений о возможностях аналитической платформы;
- изучение основных модулей; работа с мастерами импорта, экспорта, обработки и визуализации данных.

Теоретическая часть

АП «Deductor» применима для решения задач распознавания и обработки данных, таких как парциальная обработка данных (подготовка к анализу) прогнозирование, поиск закономерностей и пр. Платформа применима в задачах,

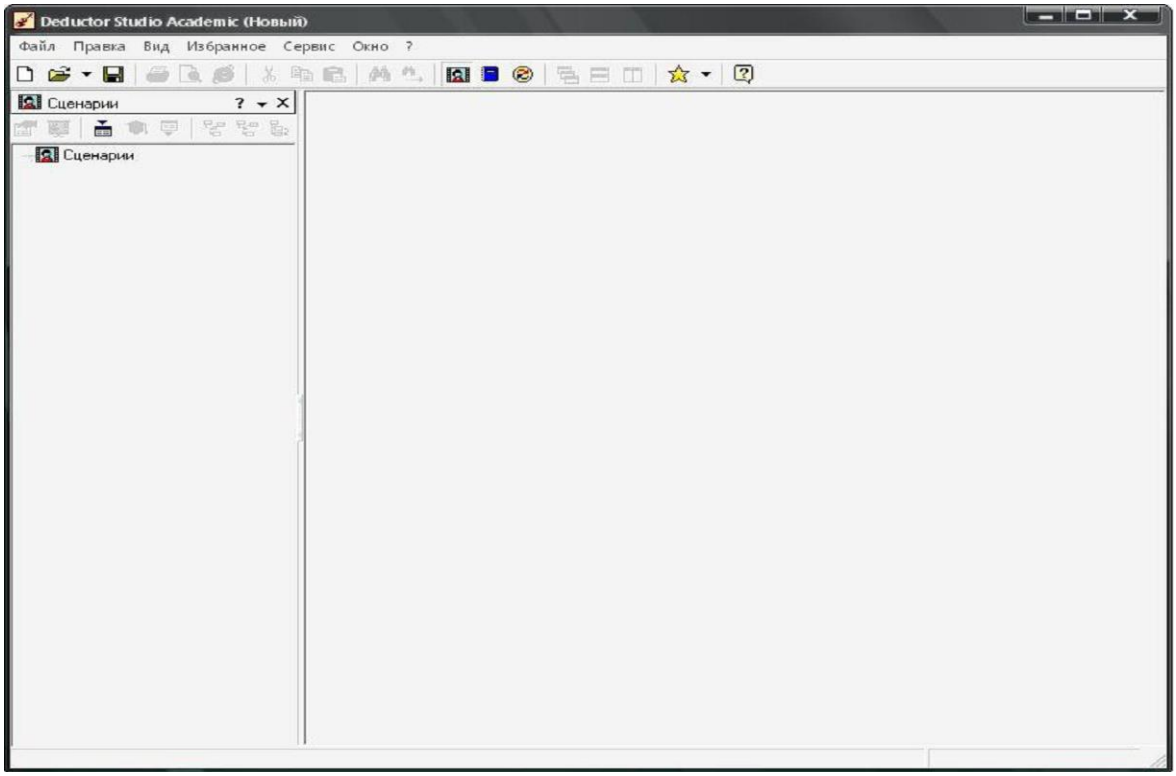
где требуется консолидация и отображение данных различными способами, построение моделей и последующее применение полученных моделей к новым данным.

Задачи, решаемые АП:

- Системы корпоративной отчетности. Готовое хранилище данных и гибкие механизмы предобработки, очистки, загрузки, визуализации позволяют быстро создавать законченные системы отчетности в сжатые сроки.
- Обработка нерегламентированных запросов**.** Конечный пользователь может получить ответ на вопросы типа "Сколько было продаж товара по группам за прошлый год с разбивкой по месяцам?" и просмотреть результаты наиболее удобным для него способом.
- Анализ тенденций и закономерностей, планирование, ранжирование. Простота использования и интуитивно понятная модель данных позволяет вам проводить анализ по принципу «Что, если...?», соотносить ваши гипотезы со сведениями, хранящимися в базе данных, находить аномальные значения, оценивать последствия принятия бизнес-решений.
- Прогнозирование. Построив модель на исторических примерах, можно использовать ее для прогнозирования ситуации в будущем. По мере изменения ситуации нет необходимости перестраивать все, необходимо всего лишь дообучить модель.
- Управление рисками. Реализованные в системе алгоритмы дают возможность достаточно точно определиться с тем, какие характеристики объектов и как влияют на риски, благодаря чему можно прогнозировать наступление рискового события и заблаговременно принимать необходимые меры к снижению размера возможных неблагоприятных последствий.
- Анализ данных маркетинговых и социологических исследований. Анализируя сведения о потребителях, можно определить, кто является вашим клиентом и почему. Как изменяются их пристрастия в зависимости от возраста, образования, социального положения, материального состояния и множества других показателей.
- Диагностика. Механизмы анализа, имеющиеся в системе Deductor, с успехом применяются в медицинской диагностике и диагностике сложного оборудования. Например, можно построить модель на основе сведений об отказах. При ее помощи быстро локализовать проблемы и находить причины сбоев.
- Обнаружение объектов на основе нечетких критериев**.** Часто встречается ситуация, когда необходимо обнаружить объект, основываясь не на таких четких критериях, как стоимость, технические характеристики продукта, а на размытых формулировках, например, найти продукты, похожие на ваши с точки зрения потребителя.

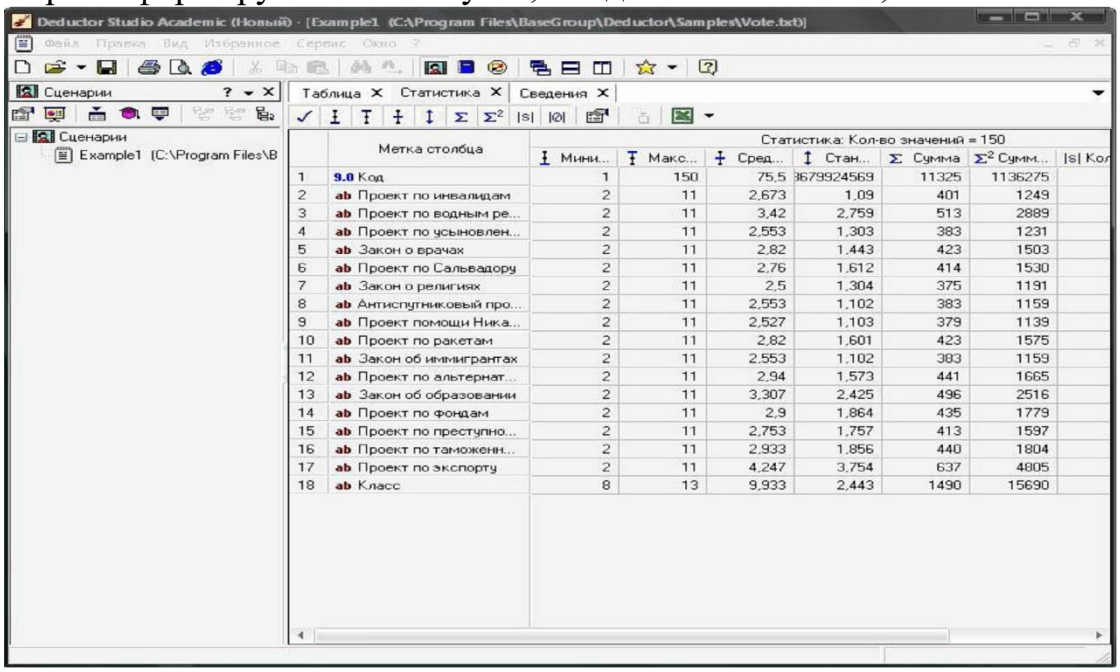
Ход работы

После запуска «Deductor Studio Academic» появится главное окно программы. Главное окно после запуска программы Deductor Studio



Главное окно после запуска программы Deductor Studio

Выполнив вышеуказанные действия по импорту данных, на панели «Сценарии» формируется новый узел, с заданными именем, меткой и описанием.



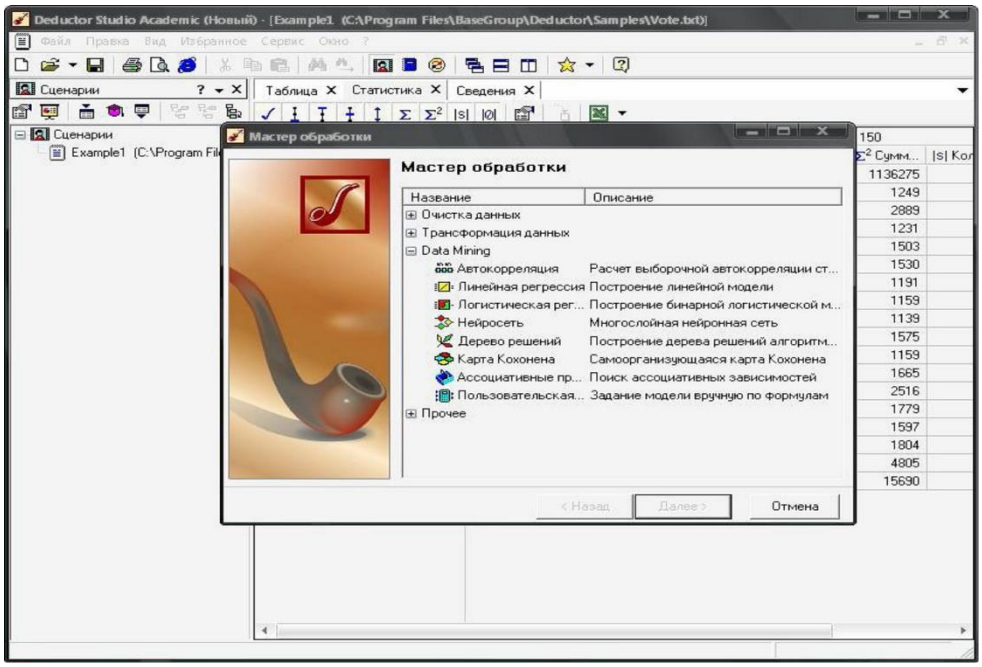
Пример создания сценария, вкладка «Статистика»

Для изучения возможности мастера обработки (кнопка в левой части главного окна либо клавиша F7). После запуска мастера обработки появится список возможных способов обработки данных.

Все способы разделены на четыре основные группы: очистка данных,

трансформация данных, Data Mining, пр. Каждый способ обработки имеет название и краткое описание. Выбор способа зависит от целей обработки данных (например, сортировка и фильтрация данных, построение дерева решений и пр.).

Мастер визуализации позволяет определить способ отображения данных, указать метки и добавить описание к проекту. Запустить его можно с помощью кнопки либо клавишей F5.



Список доступных способов обработки данных

Готовый проект можно экспортировать, воспользовавшись мастером экспорта (кнопка основного окна либо клавиша F8).

Указав параметры, проект можно перенести в один из доступных форматов.

Задание

1. Опишите назначение и возможности АП «Deductor».
2. Запустите программу «Deductor Studio Academic», ознакомьтесь с назначением кнопок и контекстным меню главного окна программы.
3. Воспользуйтесь мастером импорта данных (импортируйте файл с данными Вашей предметной области или из C:\Program Files\ BaseGroup\ Deductor\Samples\ *.txt), или из репозитория данных.
4. Ознакомьтесь с доступными способами обработки данных.
5. Изучите возможности мастера визуализации и экспорта.

Содержание отчета

1. Цель работы.
2. Краткое описание хода работы с описанием возможности Deductor дл распознавания и обработки данных и приведением скриншотов.
3. Ответы на вопросы.

Вопросы:

1. Какие существуют другие платформы для распознавания и обработки данных?
2. Какие возможности имеет АП Deductor для распознавания данных?

3. Какие возможности имеет АП Deductor для обработки данных?
4. Какие параметры доступны для мастера экспорта данных?
5. В чем заключается процедура визуализации данных?

Проектирование хранилищ данных Deductor Warehouse

Стратегические СППР, основанные на анализе большого количества информации из разных источников с привлечением сведений, содержащихся в системах, аккумулирующих опыт решения проблем, предполагают глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня является хранилище данных.

Хранилище данных – разновидность систем хранения, ориентированная на поддержку процесса анализа данных, обеспечивающая целостность, непротиворечивость и хронологию данных, а также высокую скорость выполнения аналитических запросов.

Использование концепции хранилища данных (ХД) в СППР и анализе данных способствует достижению таких целей, как:

- своевременное обеспечение аналитиков и руководителей всей информацией, необходимой для выработки обоснованных и качественных управленческих решений;
- создание единой модели представления данных в организации;
- создание интегрированного источника данных, предоставляющего удобный доступ к разнородной информации и гарантирующего получение оди наковых ответов на одинаковые запросы из различных аналитических приложений.

Первой фазой разработки ХД является системный анализ объекта принятия решений. Например, в практике СППР для бизнеса известно два подхода к такому анализу. Первый ориентируется на описание бизнес-процессов, протекающих на предприятии, которое моделируется набором взаимосвязанных функциональных элементов. Второй подход основан на первичном анализе бизнес-событий. При этом должна быть собрана информация об используемых внешних данных и их источниках; о форматах данных, периодичности и форме их поступления; о внутренних информационных системах объекта, обслуживаемого СППР, их функциях и алгоритмах обработки данных, используемых при наступлении бизнес-событий. Такой анализ, как правило, производится при проектировании любой информационной системы. Особенность анализа данных при проектировании СППР состоит в необходимости создания моделей представления информации. То, что в обычных информационных системах является вторичным понятием, а именно состав и форма отображаемых данных, в СППР приобретает особую важность, так как нужно выявить все без исключения признаки, требуемые для принятия решений.

Модель представления данных является организационно-функциональным срезом модели системы, а при ее разработке последовательно изучаются:

- распределение пользователей системы: географическое, организационное, функциональное;

- доступ к данным: объем данных, необходимый для анализа, уровень агрегированности данных, источники данных (внешние или внутренние), описа ние информации, совместно используемой различными функциональными группами пользователей;
- аналитические характеристики системы: измерения данных, основные отчеты, последовательность преобразования аналитической информации, степень предопределенности анализа, существующие или находящиеся в стадии разработки средства анализа.

По результатам анализа бизнес-процессов и структур данных отбирается действительно значимая для принятия решений информация с учетом неопределенности будущих запросов.

Следующий шаг связан с пониманием того, в каком виде и на каких аппаратных и программных платформах размещать структуру данных СППР на основе хранилищ данных.

Хранилище данных Deductor Warehouse – это специально организованная база данных, ориентированная на решение задач анализа данных и поддержки принятия решений, обеспечивающая максимально быстрый и удобный доступ к информации.

Объекты хранилища данных Deductor Warehouse:

Измерение – это последовательность значений одного из анализируемых параметров. Например, для параметра «время» – это последовательность календарных дней, для параметра «место проживания» – список названий городов. Каждое значение измерения может быть представлено координатой в многомерном пространстве процесса, например город, клиент, дата.

Атрибут – это свойство измерения (т.е. точки в пространстве). Атрибут как бы скрыт внутри другого измерения и помогает пользователю полнее описать исследуемое измерение. Например, для измерения «Код региона» атрибутом является «Регион».

Факт – значение, соответствующее измерению. Факты – это данные, отражающие сущность события. В большинстве случаев фактами являются численные значения, например сумма, количество, объем.

Ссылка на измерение – это установленная связь между двумя и более измерениями. Некоторые понятия, соответствующие измерениям в хранилище данных, могут образовывать иерархии. Например, «Недвижимость» может включать «Новостройки» и «Вторичное жилье», которые, в свою очередь, подразделяются на группы. В этом случае первое измерение содержит ссылку на второе, второе – на третье и т.д.

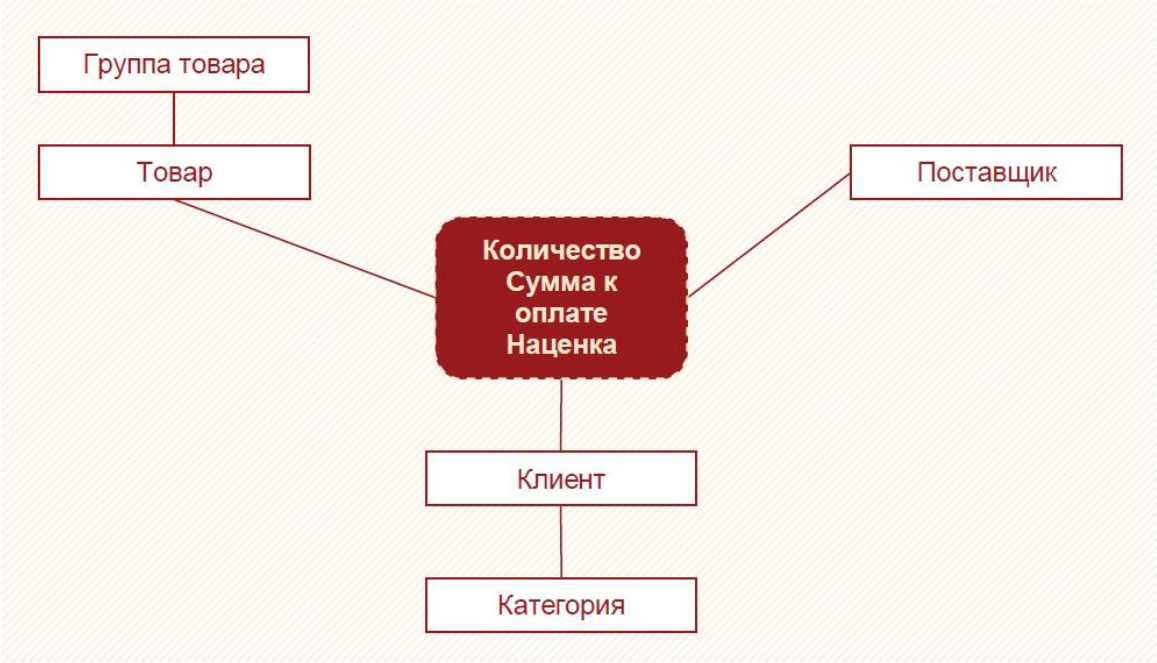
Процесс – совокупность измерений, фактов и атрибутов. Процесс описывает определенное действие, например продажа, отгрузка, мониторинг. Атрибут процесса – свойство процесса. Атрибут процесса в отличие от измерения не определяет координату в многомерном пространстве. Это справочное значение, относящееся к процессу. Значение атрибута процесса, в отличие от измерения, может быть не всегда определено.

Все загружаемые в ХД данные обязательно должны быть определены как измерение, атрибут либо факт.

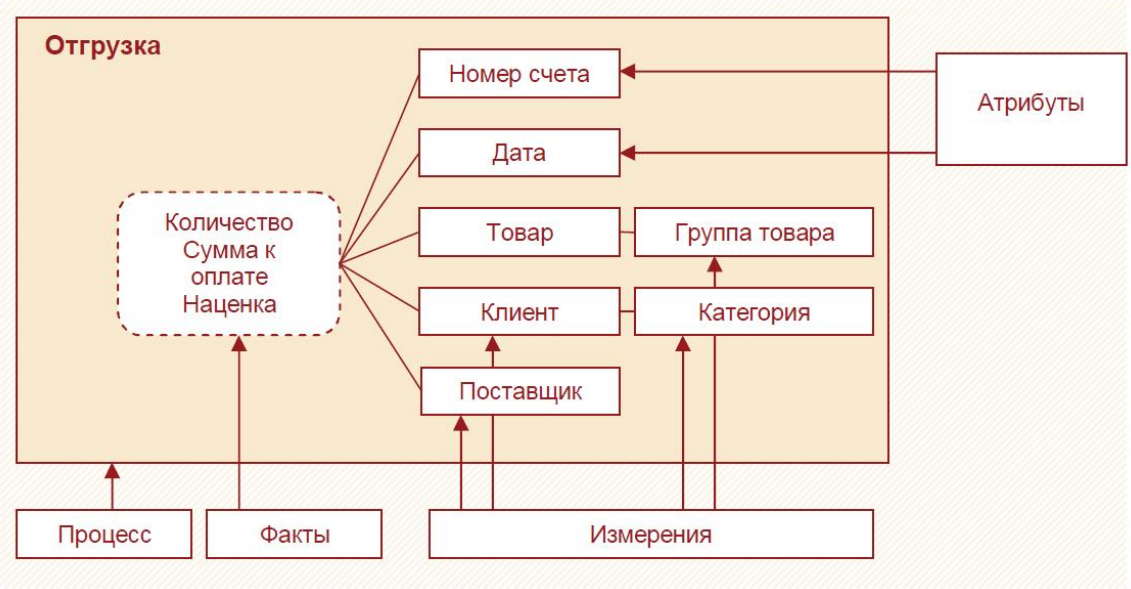
Принадлежность данных к типу (измерение, ссылка на измерение, атрибут или факт) содержится в семантическом слое хранилища.

Семантический слой – механизм, позволяющий оперировать данными посредством терминов предметной области.

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 может одновременно храниться множество процессов, имеющие общие измерения, например, Товар, фигурирующий в Поступления и в Отгрузка.

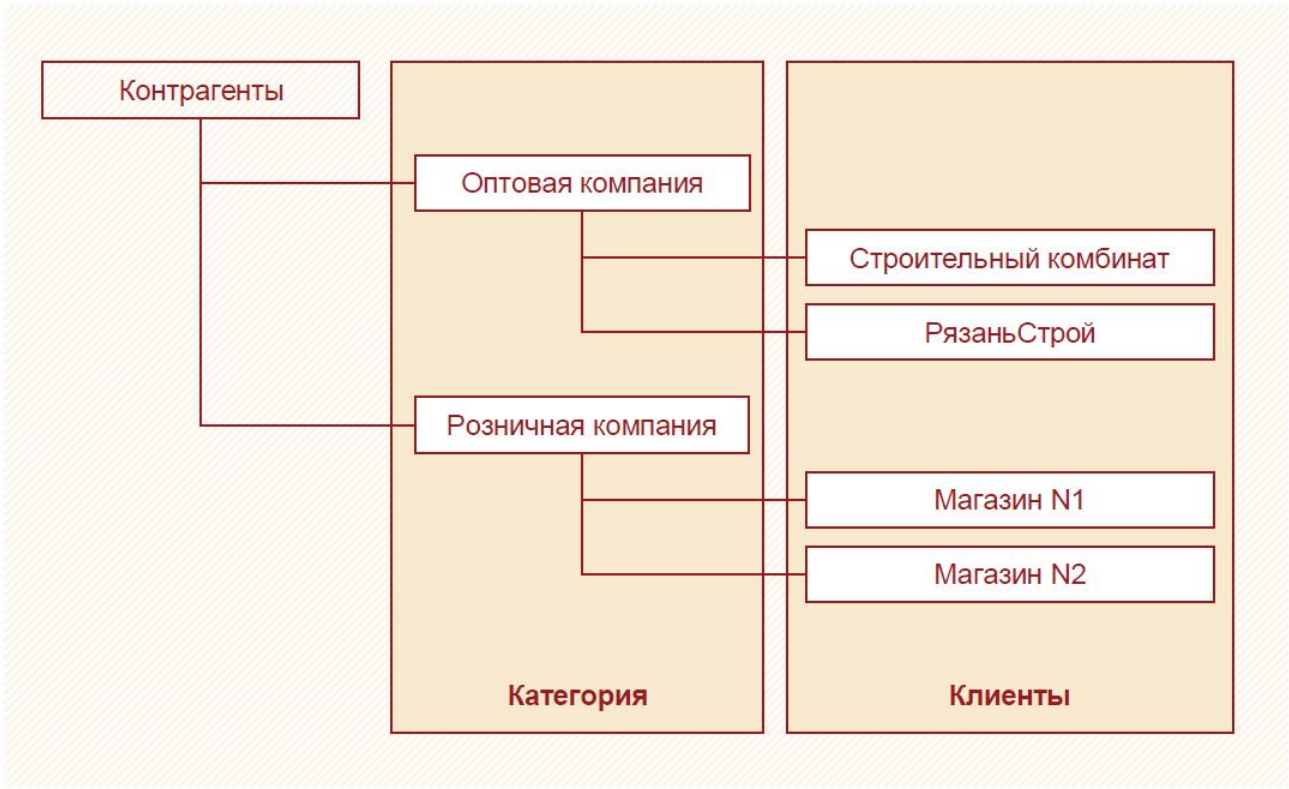


Измерения могут быть как простыми списками, например, Клиент, так и содержать дополнительные столбцы, называемые атрибутами измерений. Например, измерение Товар может состоять из Наименование товара - собственно измерение (первичный ключ) и Вес, Объем и прочее – атрибуты данного измерения.

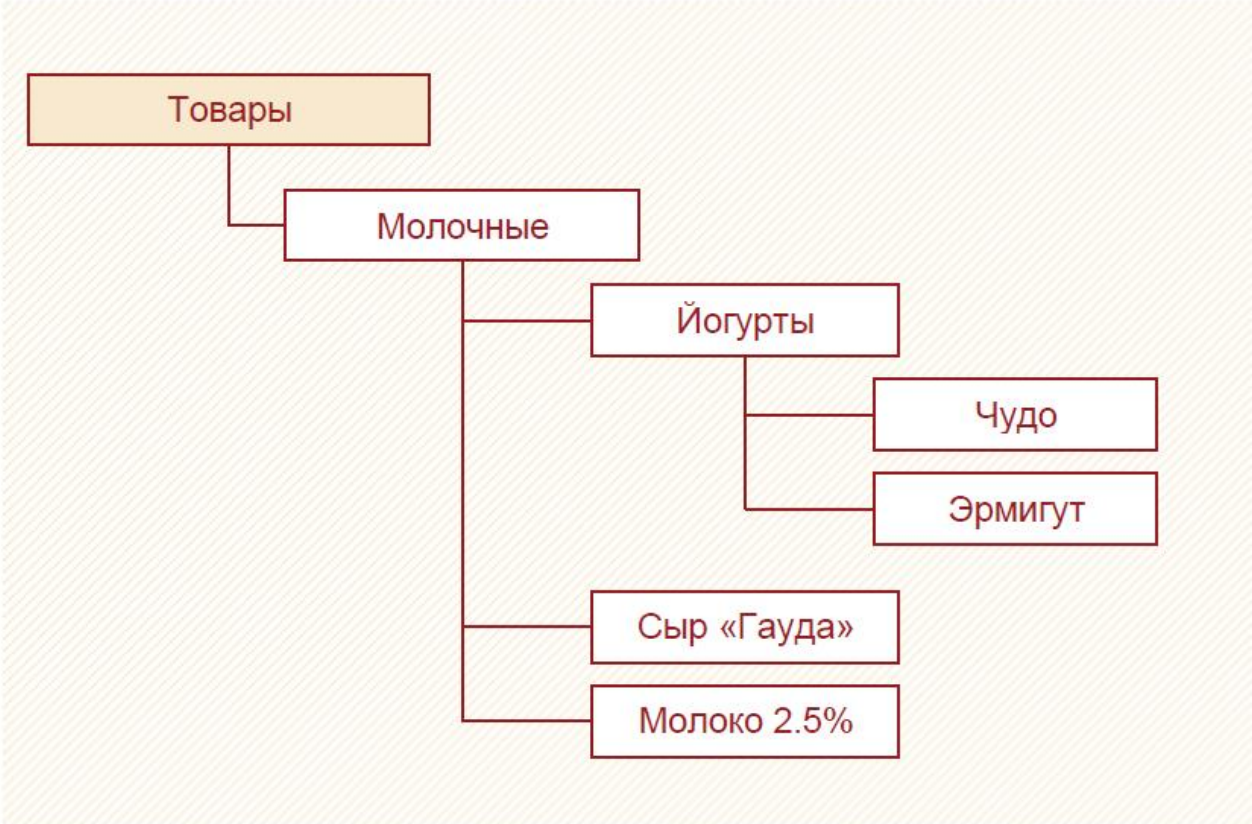
Измерение может ссылаться на другое измерение, в свою очередь следующее измерение тоже ссылаться на измерение и так далее. Таким способом организуется иерархия измерений.

Поддержку иерархий обеспечивает схема «снежинка». В Deductor Warehouse 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.

Физически Deductor Warehouse – 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. Кроссплатформенный Deductor Warehouse является удобной базой для создания распределенных хранилищ данных, витрин данных и прочее.

Deductor Warehouse 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 производится на панели **Подключения**. Открыть или скрыть эту панель можно, выбрав в главном меню **Вид ► Подключения**.

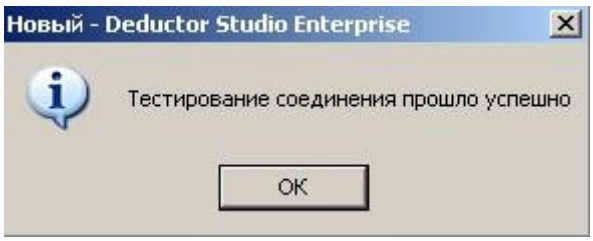
Для создания хранилища данных необходимо выполнить следующее. На панели инструментов закладки **Подключения** нажать кнопку . В результате откроется мастер подключений, в котором можно выбрать и настроить все доступные в системе источники/приемники данных:

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

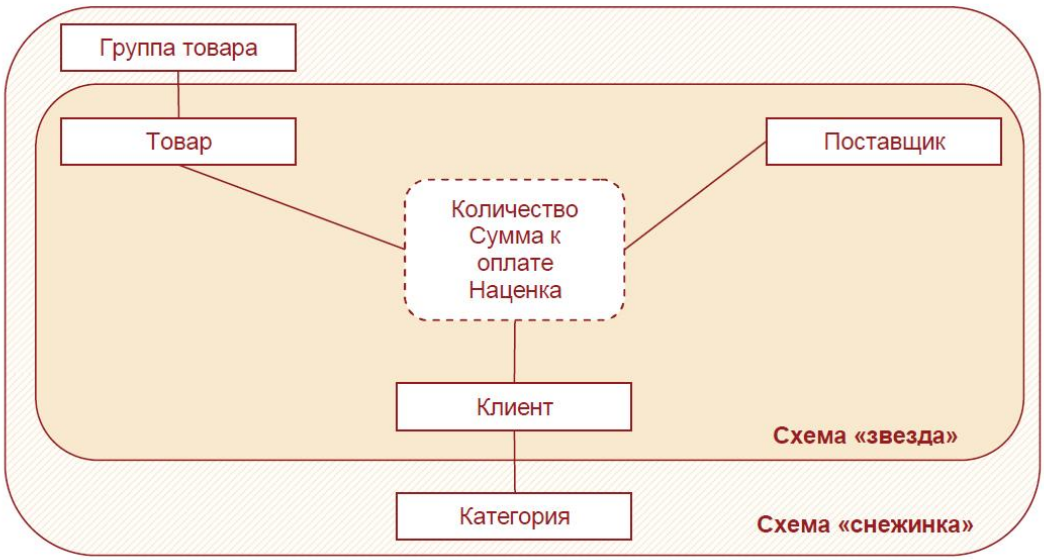
Рассмотрим работу с Deductor Warehouse (кросс-платформенный). В Мастере подключений выбираем хранилище данных «Deductor Warehouse (кросс-платформенный)». Переходим на следующий этап настройки, нажав кнопку **Далее**. На следующем этапе выбираем тип базы данных. Система поддерживает следующие СУБД: Oracle, 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» (совет: для быстрой установки этого логина и пароля щелкните ;
- *Параметры* – имеется несколько следующих параметров:
  1. Спрашивать логин/пароль при подключении;
  2. Сохранять пароль;
  3. Показывать системные таблицы;
  4. Обрамлять имена таблиц.

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:



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. 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 загружен 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.

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.

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

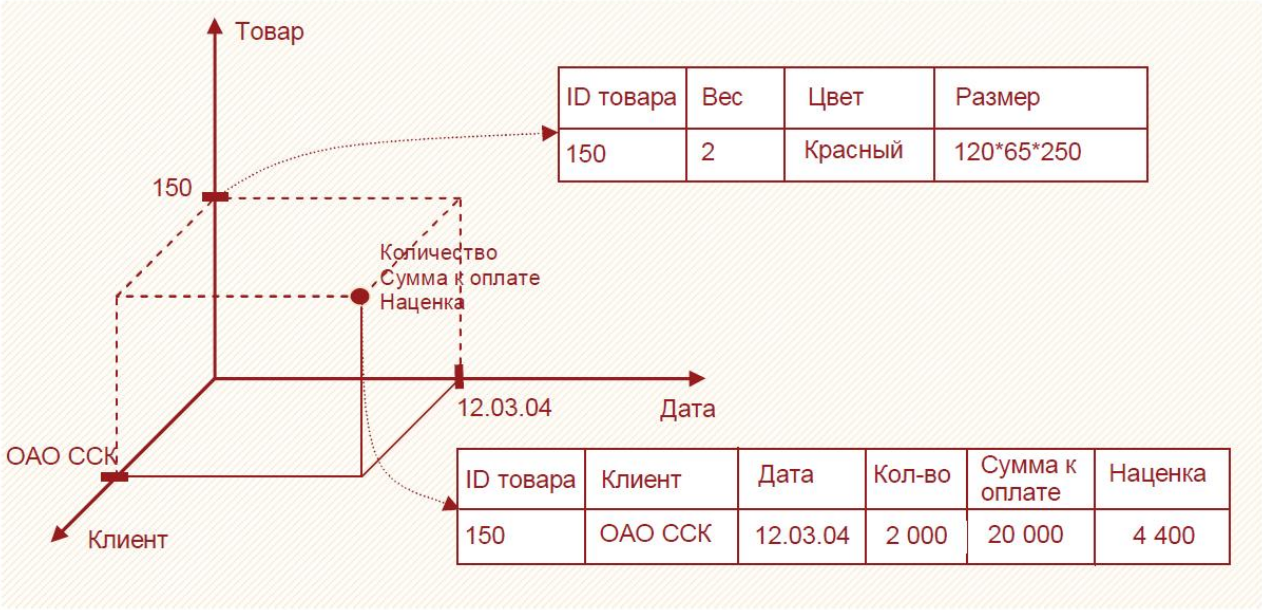
80	01.03.04	ОАО «ССК»	051/2004	1000	7500	1000
90	10.03.04	ООО «ДСК»	052/2004	1500	15000	3000
150	12.03.04	ООО «ДСК»	053/2004	900	9000	2000
150	12.03.04	ОАО «ССК»	054/2004	2000	20000	4400
160	15.03.04	ООО «ДСК»	055/2004	500	6000	1000

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:



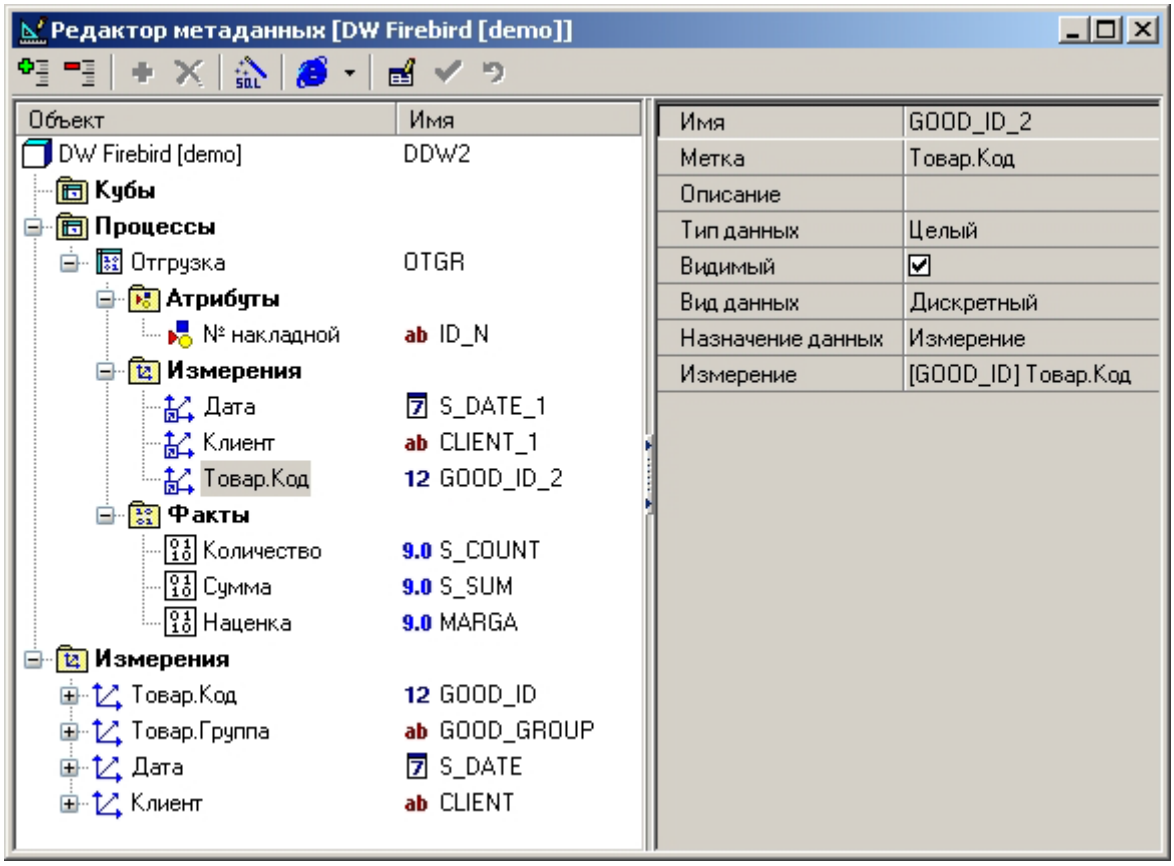
Для начала необходимо импортировать эти данные в программу. Для этого вызвать два раза Мастер импорта и загрузить (т.е. создать узел импорта) данные сначала о товаре, а затем об отгрузке.

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 **Разрешить редактировать**, выйти из этого режима – нажав кнопку **Принять изменения**.

Осталось разобраться с товарной группой. Налицо иерархия товаров: 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.

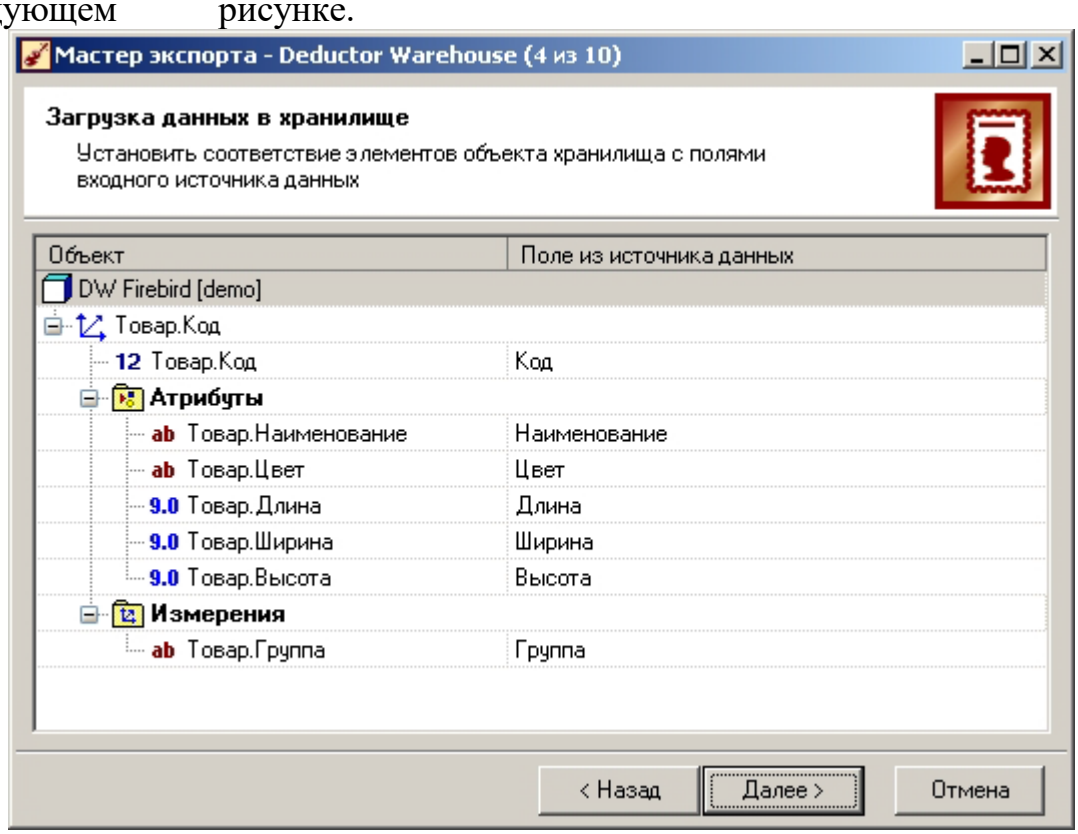


Экспорт начнем с измерений, иначе процесс *Отгрузка* успешно не загрузится. Нам необходимо загрузить 4 измерения. Причем загрузка

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.

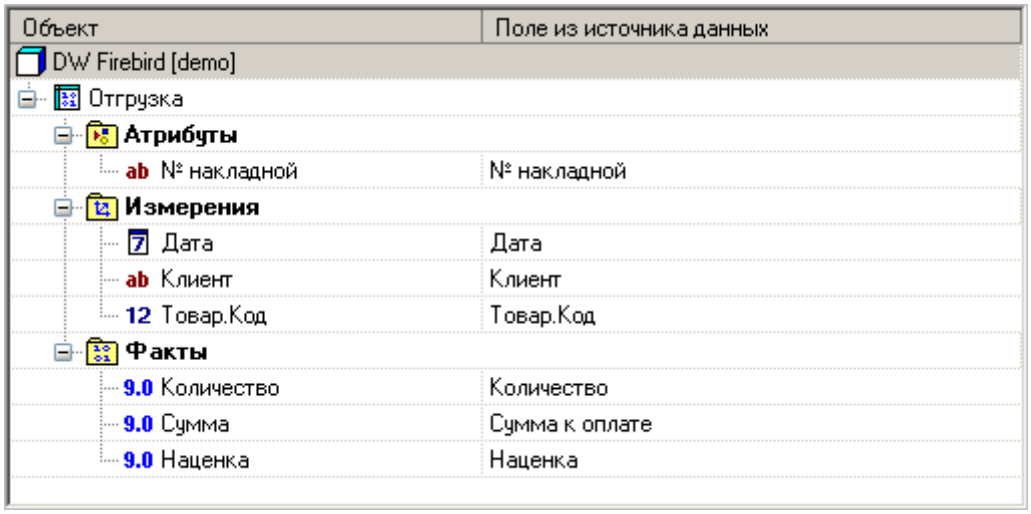
Для загрузки нужен список уникальных (неповторяющихся) значений групп. Его можно получить при помощи обработчика «Группировка», указав в качестве измерения то поле, по которому мы хотим получить список уникальных значений (в данном случае – *Группа*).

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.

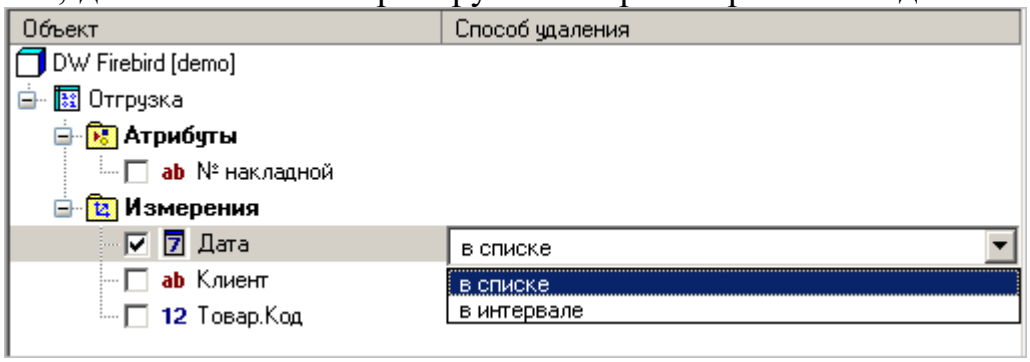


Измерения без атрибутов *Дата*, *Клиент* можно не загружать отдельными узлами сценария, а сделать это во время загрузки в процесс.

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.

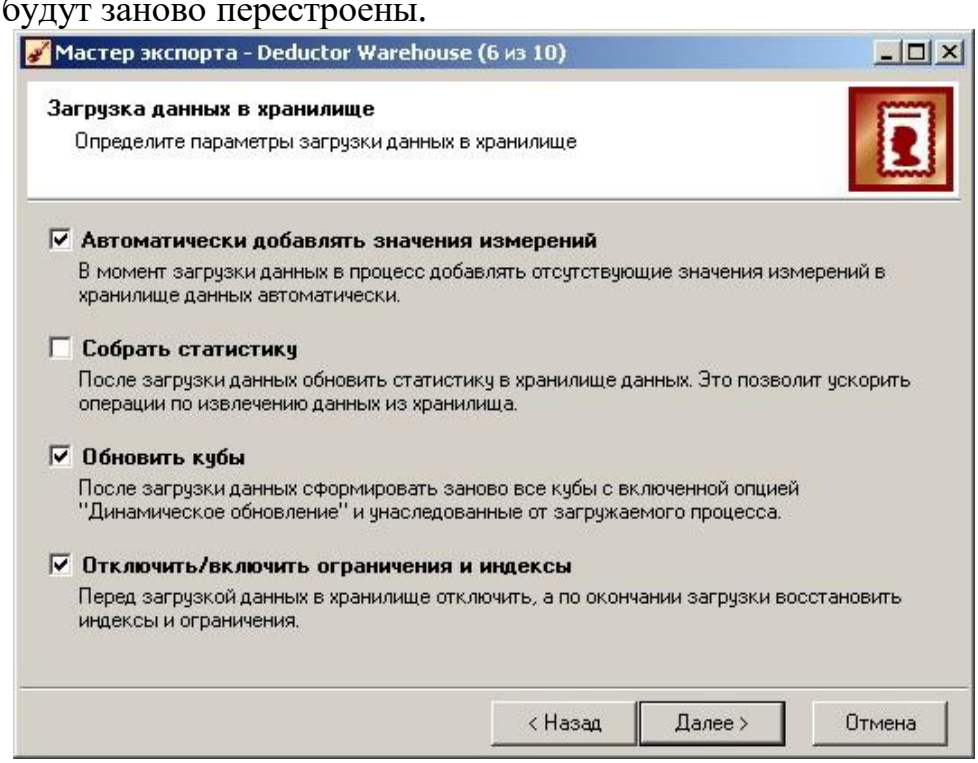


Для измерения здесь доступны две опции в колонке **Способ удаления** (по умолчанию предлагается *в списке*):

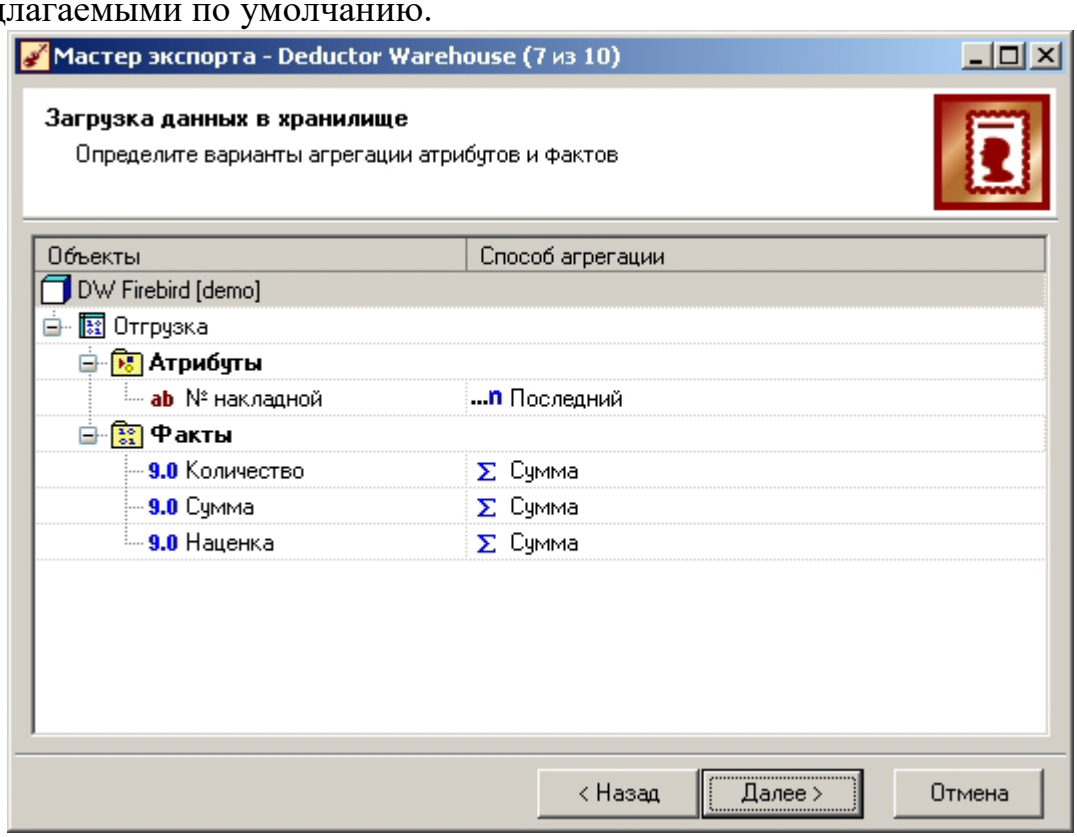
- *в списке* – при выполнении операции удаления для измерения списком перечисляются все значения;
- *в интервале* – у загружаемых значений измерения определяются минимальное и максимальное значения и из ХД удаляются все значения, лежащие внутри этого диапазона, поэтому операция производится быстрее.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

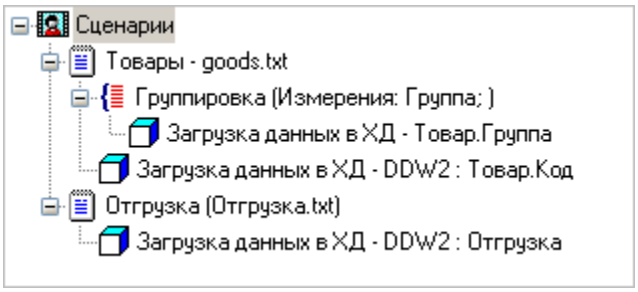
активным флаг **Обновить кубы**. Он говорит о том, что после загрузки данных в ХД имеющиеся кубы с активной опцией **Автоматическое обновление** будут заново перестроены.



Далее 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 предыдущий пример.

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 Analytical Server.

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);
- *с переменными* – значения для формирования срезов будут браться из переменных проекта Deductor или переменных приложения

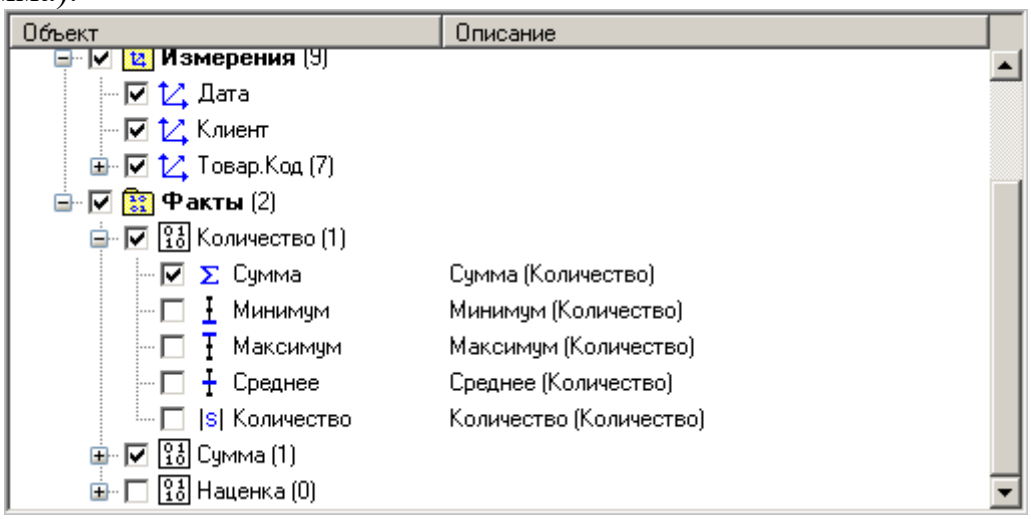
Последние две настройки позволяют строить динамические отчеты, в которых пользователю предоставляется только интересующая его информация, а конкретные условия фильтрации он выбирает в момент импорта данных, либо условия фильтрации передаются через переменные при пакетном/серверном запуске сценария.

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.

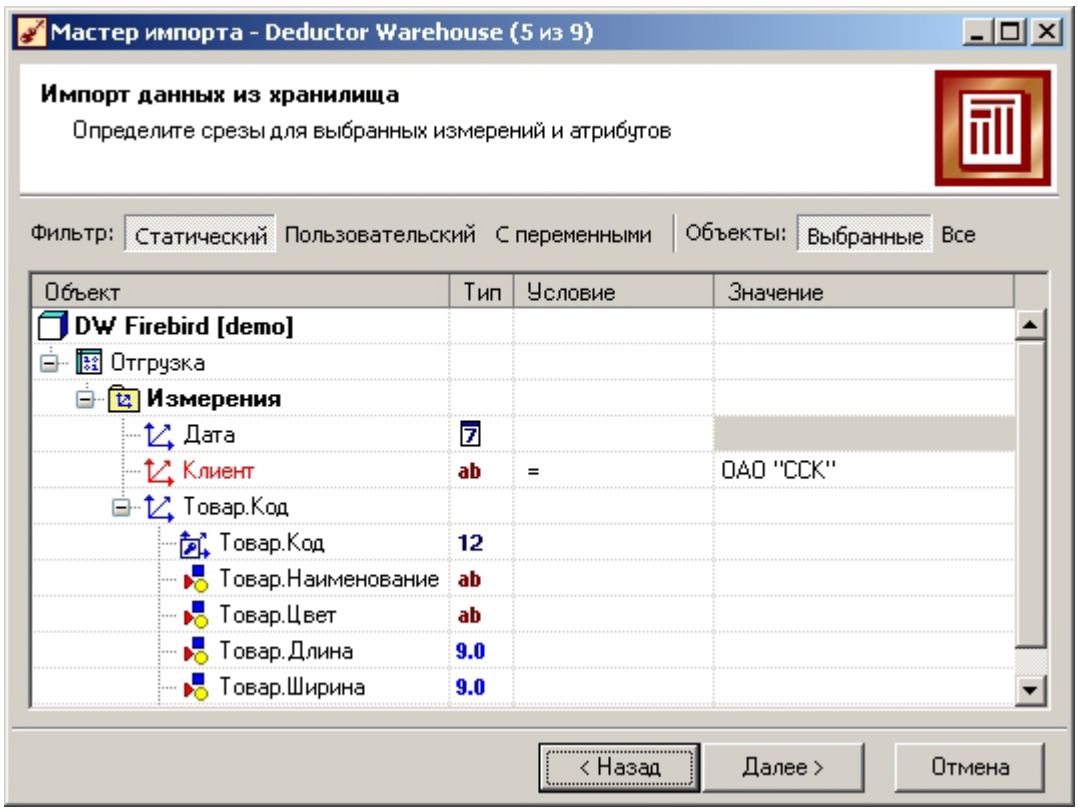
Пример

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.

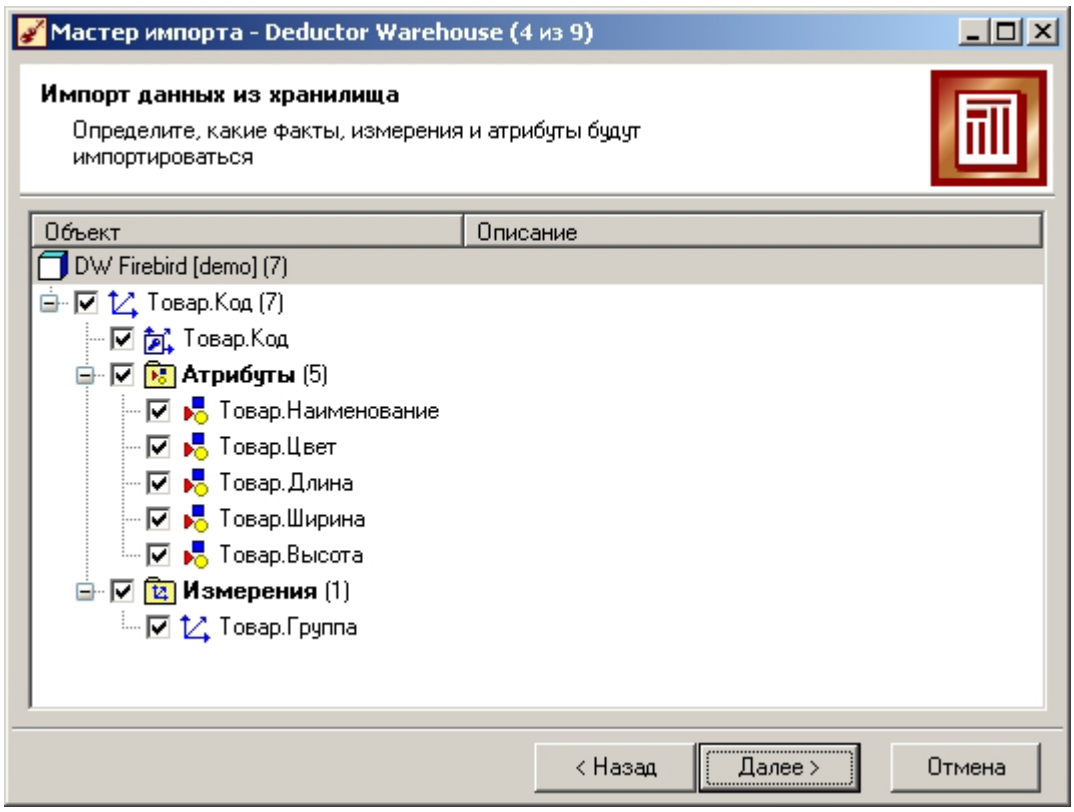
Импорт измерения

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

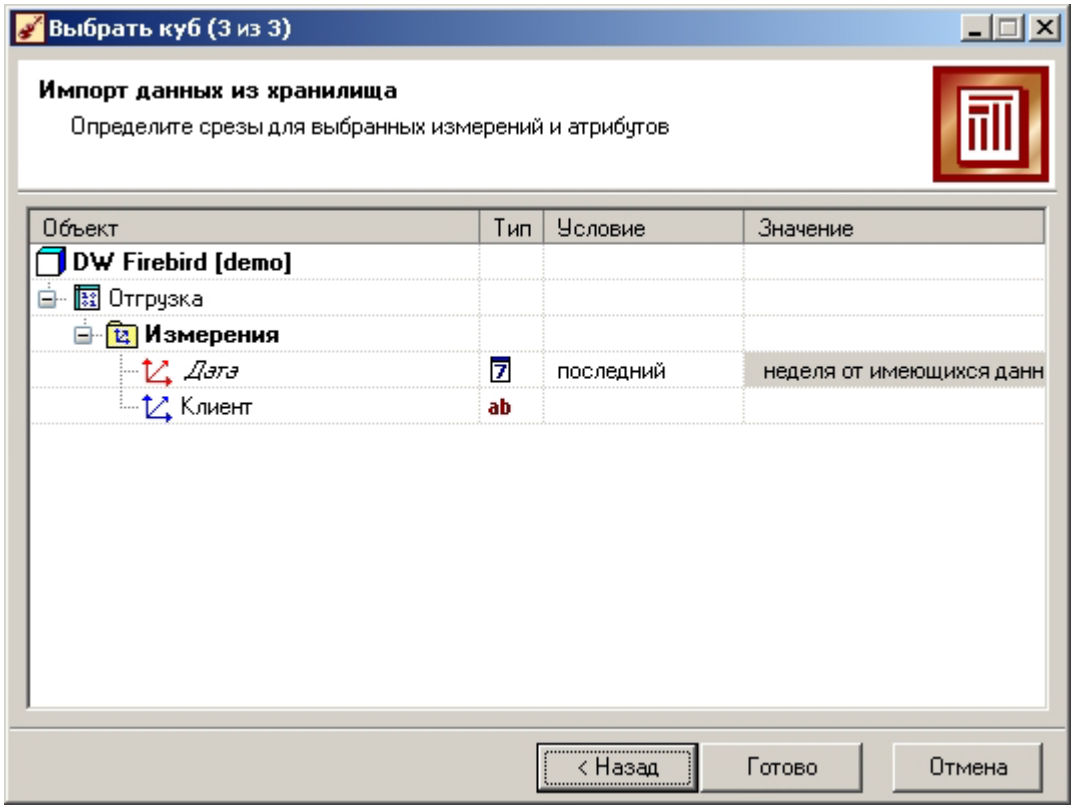
Кубы в хранилище данных

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.

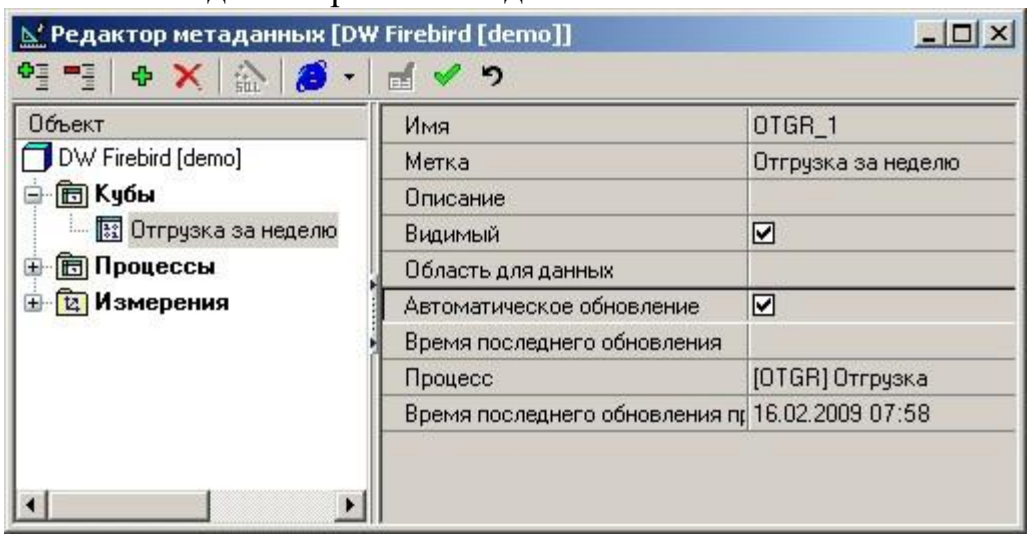
Пример

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

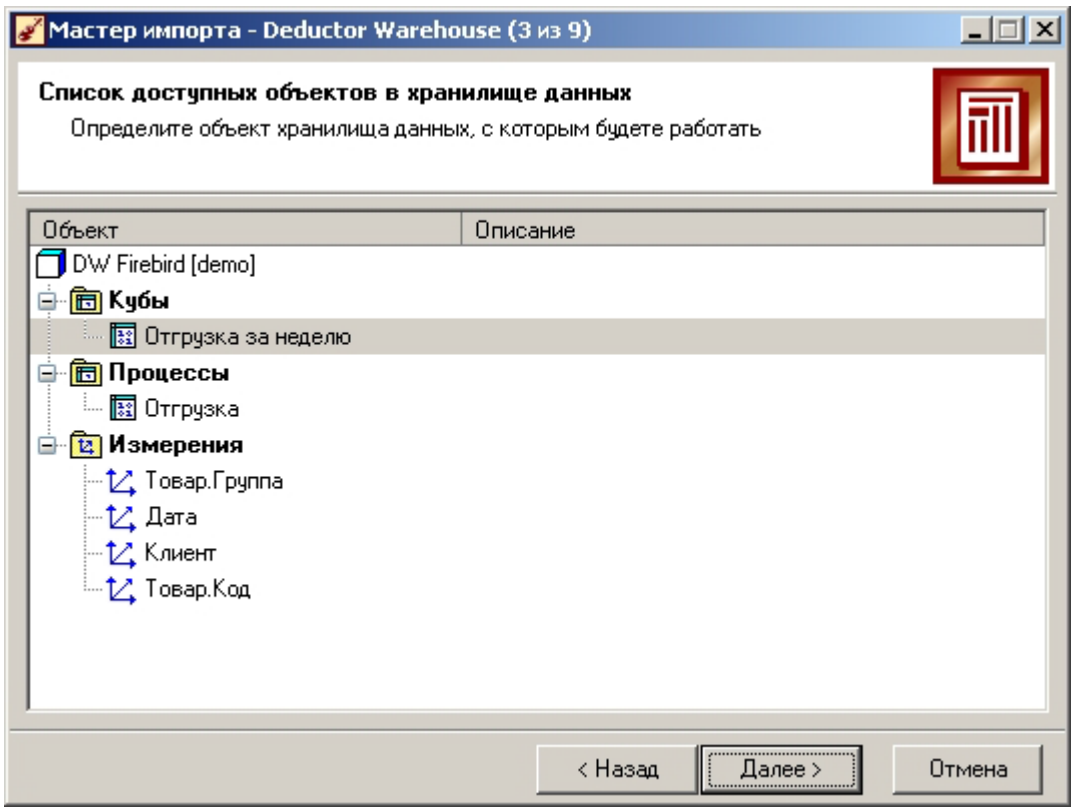
по измерению Дата настроим статический фильтр с условием «последний» и значением «неделя от имеющихся данных».



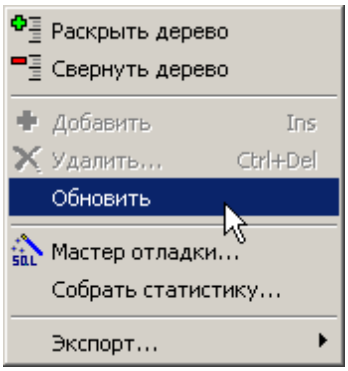
Дадим кубу название *Отгрузка за неделю*. В редакторе для него стал доступен специальный флаг **Автоматическое обновление**. Поднятый флаг говорит о том, что при попадании новых данных в процесс куб будет автоматически пересчитываться, при этом в поле «Время последнего обновления» будет заноситься дата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



Практическая работа «Создание ХД в аналитической платформе Deductor Studio Academic»

Имеется история продаж и поступлений различных товаров по дням в нескольких торговых объектах. Данные представлены текстовыми файлами, которые включают в себя выгрузку информации о поступлении товара, продаже товара и предоставляемую скидку при продаже товара, а также справочники: Артикулы, Единицы измерения, Группы клиентов, Группы товаров, Номера клиентов, Обобщенные группы товаров, Список городов.

Артикулы (фрагмент) Таблица 1

Артикул	Наименование товара	Группа товаров	Группа товаров
			Обобщенная группа товаров
108006	Балка декоративная DECOSA Рустик 120x120x20(	Потолочные покрытия	Отделочные материалы
108003	Балка декоративная DECOSA Рустик 60x90x2000,	Потолочные покрытия	Отделочные материалы
108004	Балка декоративная DECOSA Рустик 60x90x3000,	Потолочные покрытия	Отделочные материалы
108005	Балка декоративная DECOSA Рустик 60x90x4000,	Потолочные покрытия	Отделочные материалы
107902	Бамбук половина ствола, диаметр 30-40 мм, 2 м,	Стеновые покрытия	Отделочные материалы
105423	Болт 10x20 цинк, шестигранная головка, 8 штук, 8	Метизы и крепёж	Метизы и крепёж
105412	Болт 6x80 цинк, шестигранная головка, 3 штуки, 3	Метизы и крепёж	Метизы и крепёж
105413	Болт 6x90 цинк, шестигранная головка, 6 штук, 6	Метизы и крепёж	Метизы и крепёж

Группа клиентов Таблица 2

Группа клиентов
VIP клиент
Клиент
Постоянный клиент

Группа товаров (фрагмент) Таблица 3

Группа товаров	Обобщенная группа товаров
Армирующие материалы	Армирующие материалы
Гидроизоляция	Изоляционные материалы
Грунтовка	Лакокрасочные материалы и строительная химия
Изоляция	Изоляционные материалы
Краска	Лакокрасочные материалы и строительная химия
Лаки, морилки	Лакокрасочные материалы и строительная химия
Металлолом	Металлолом
Метизы и крепёж	Метизы и крепёж
Напольные покрытия	Отделочные материалы
Плитка	Отделочные материалы
Потолочные покрытия	Отделочные материалы
Профиль заказной	Металлический профиль и комплектующие
Профиль и комплектующие для вентфасадов	Металлический профиль и комплектующие PRIMET
Профиль и комплектующие для ГКЛ	Металлический профиль и комплектующие PRIMET

Единицы измерения Таблица 4

Единица измерения
кв.м
кг
л
м
уп
шт

Клиенты (фрагмент) Таблица 5

Дата продажи	№ Клиента	Артикул	Единица измерения	№ Клиента			
				Город	Город		Группа клиентов
					Экономический район	Федеральный округ	
01.03.2004	00000010	105285	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105290	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105291	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105412	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105417	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент
01.03.2004	00000010	105418	уп	Нижний Новгород	Волго-Вятский экономический район	Приволжский федеральный округ	Постоянный клиент

Клиенты (фрагмент) Таблица 5 (продолжение)

Артикул		Цена за единицу	Количество	Сумма с учетом скидки
Наименование товара	Группа товаров			
Эмаль алкидная SADOLIN Master-30 WO, 1л, ц	Эмаль	213,8	40	8124,4
Эмаль алкидная MARSHALL Pastel Yarimat 111	Эмаль	161,18	40	6124,84
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	40	4974,4
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	34	4101,3928
Эмаль алкидная ALPINA Direkt auf Rost антикс	Эмаль	184,37	100	17515,15
Эмаль алкидная MARSHALL Enamel Parlak для	Эмаль	124,36	40	4974,4
Эмаль НЦ-132П OLEKOLOR черная, 1.8 кг, шт	Эмаль	135,11	76	9754,942

Номера клиентов (фрагмент) Таблица 6

№ Клиента	Город	Группа клиента
00000003	Москва	Клиент
00000004	Тверь	Клиент
00000010	Нижний Новгород	Постоянный клиент
00000011	Балашиха	Клиент
00000014	Москва	Клиент
00000015	Нижний Новгород	Постоянный клиент
00000016	Нижний Новгород	Клиент
00000017	Москва	Клиент
00000019	Владимир	VIP клиент
00000020	Кострома	VIP клиент
00000031	Москва	Клиент
00000032	Москва	Клиент
00000038	Мытищи	Клиент

Обобщенные группы товаров Таблица 7

Обобщенные группы товаров
Армирующие материалы
Изоляционные материалы
Лакокрасочные материалы и строительная химия
Металлический профиль и комплектующие
Металлический профиль и комплектующие PRIMET
Металлолом
Метизы и крепёж
Отделочные материалы
Сыпучие и вяжущие материалы и смеси

Приход (фрагмент) Таблица 8

Дата поставки	№ Счет-фактуры	Номер поставщика	Артикул	Артикул			Цена за единицу	Количество
				Наименование товара	Группа товаров	Группа товаров		
						Обобщенная группа товаров		
01.03.2004	553	5	100429	Лак акриловый ОПТИМИСТ бесцвет	Лаки, морилки	Лакокрасочные материалы и	140,3	81
01.03.2004	553	5	105025	Краска колеровочная ПРЕМИЯ черна	Тонер, колер	Лакокрасочные материалы и	88,76	92
01.03.2004	3997	3	105412	Болт 6х80 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	9,55	225
01.03.2004	3997	3	105414	Болт 8х25 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	14,69	225
01.03.2004	3997	3	105415	Болт 8х35 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	16,73	225
01.03.2004	3997	3	105417	Болт 8х50 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	16,73	225
01.03.2004	3997	3	105419	Болт 8х70 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	21,13	226
01.03.2004	3997	3	105421	Болт 8х90 цинк, шестигранная голов	Метизы и крепёж	Метизы и крепёж	11,59	225
01.03.2004	3997	3	105422	Болт 8х100 цинк, шестигранная голо	Метизы и крепёж	Метизы и крепёж	11,59	226
01.03.2004	3997	3	105556	Дюбель с шурупом забивной WKRET	Метизы и крепёж	Метизы и крепёж	66,69	225
01.03.2004	9085	4	102438	Профиль перегородочный стоечный (	Профиль ПРОФИ	Металлический профиль и ко	22,36	137
01.03.2004	9085	4	102439	Профиль перегородочный стоечный (	Профиль СТАНДА	Металлический профиль и ко	18,86	138

Скидка (фрагмент) Таблица 9

Дата продажи	№ Клиента	Артикул	№ Клиента				Наименование товара
			Город	Город		Группа клиентов	
				Экономический район	Федеральный округ		
01.03.2004	00000081	102313	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль перегородочный стоечный
01.03.2004	00000081	102339	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль перегородочный стоечный
01.03.2004	00000081	102578	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Профиль потолочный направляющий
01.03.2004	00000081	104361	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Лента малярная KLEBEBANDER 50м
01.03.2004	00000081	113741	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113744	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113745	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113746	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113747	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113748	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2
01.03.2004	00000081	113749	Иваново	Центральный район	Центральный федеральный округ	VIP клиент	Доска паркетная TARKETT T-Lock 2

Артикул		Цена за единицу	Количество	Скидка по группе клиента %	Скидка по сумме покупки %	Общая скидка %
Группа товаров	Группа товаров					
	Обобщенная группа товаров					
Профиль укороченный	Металлический профиль	2,89	21	5	0	5
Профиль укороченный	Металлический профиль	2,6	21	5	0	5
Профиль укороченный	Металлический профиль	1,42	21	5	0	5
Армирующие материалы	Армирующие материалы	16,24	28	5	0	5
Напольные покрытия	Отделочные материалы	734,4	28	5	0	5
Напольные покрытия	Отделочные материалы	440,64	28	5	0	5
Напольные покрытия	Отделочные материалы	660,96	28	5	0	5

Список городов (фрагмент) Таб лица 10

Город	Экономический район	Федеральный округ
Архангельск	Северный экономический район	Северо -Западный федеральный округ
Астрахань	Поволжский экономический район	Южный федеральный округ
Балашиха	Центральный район	Центральный федеральный округ
Барнаул	Западно-Сибирский округ	Сибирский федеральный округ
Белгород	Центрально-Черноземный район	Центральный федеральный округ
Великие Луки	Северный экономический район	Северо -Западный федеральный округ
Владивосток	Дальневосточный экономический район	Дальневосточный федеральный округ
Владимир	Центральный район	Центральный федеральный округ
Волгоград	Поволжский экономический район	Южный федеральный округ
Вологда	Северный экономический район	Северо -Западный федеральный округ
Воронеж	Центрально-Черноземный район	Центральный федеральный округ
Екатеринбург	Уральский экономический район	Уральский федеральный округ
Зеленоград	Центральный район	Центральный федеральный округ
Иваново	Центральный район	Центральный федеральный округ
Ижевск	Уральский экономический район	Приволжский федеральный округ

Указания

При проектировании ХД необходимо учитывать следующее:

- 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;
- таблицы измерений содержат только справочную информацию (коды, наименования и т.п.) и ссылки на другие измерения при необходимости;
- таблица процесса содержит только факты и коды измерений (без их атрибутов);
- наличие возможных пропусков (необязательное поле) говорит о том, что объект лучше сделать атрибутом процесса.

Покажем, какие данные являются измерениями, какие атрибутами, а какие фактами и что представляют собой процессы.

В табл. 1 «Артикулы» измерениями являются следующие поля: Артикул, Группа товаров, Группа товаров | Обобщенная группа товаров, а поле Наименование товара является атрибутом.

Таблица «Группа клиентов» (табл. 2) содержит в себе всего 1 поле «Группа клиентов», которое является измерением.

В таблице «Группа товаров» (табл. 3) измерениями являются следующие поля: Группа товаров, Обобщенная группа товаров.

Таблица «Единицы измерения» (табл. 4) содержит в себе всего 1 поле «Единица измерения», которое является измерением.

В таблице «Клиенты» (табл. 5) измерениями являются следующие поля: Дата продажи, Номер клиента, Артикул, Единицы измерения, Номер клиента | Город, Номер клиента | Группа клиентов, Артикул | Группа товаров. Атрибутами являются поля: Номер клиента | Город | Экономический район, Номер клиента | Город | Федеральный округ, Артикул | Наименование товара, а такие поля, как Цена за единицу, Количество и Сумма с учетом скидки являются фактами. Т.е. табл. 5 является описанием процесса продажи товаров.

В таблице «Номер клиента» (табл. 6) поля Номер клиента, Город и Группа клиентов являются измерениями.

Таблица «Обобщенная группа товаров» (табл. 7) содержит в себе всего 1 поле «Обобщенная группа товаров», которое является измерением.

В таблице «Приход» (табл. 8), измерениями являются следующие поля: Дата прихода, Артикул, Артикул|Группа товаров, Артикул|Группа товаров|Обобщенная группа товаров, Номер счет-фактуры, и Номер поставщика. Поле Артикул | Наименование товара является атрибутом, а такие поля как Цена

за единицу и Количество являются фактами. Т.е. таблица 8 является описанием процесса поступления товаров.

В таблице «Скидка» (табл. 9), измерениями являются следующие поля: Дата продажи, Номер клиента, Артикул, Номер клиента|Город, Номер клиента|Группа клиентов, Артикул|Группа товаров, Артикул|Группа товаров|Обобщенная группа товаров. Атрибутами являются поля: Номер клиента|Город|Экономический район, Номер клиента|Город|Федеральный округ, Артикул|Наименование товара, а такие поля, как Цена за единицу, Количество, Скидка по группе клиента %, Скидка по сумме клиента %, Общая сумма скидки % являются фактами. Т.е. табл. 9 является описанием процесса предоставления скидки клиентам при покупке товаров.

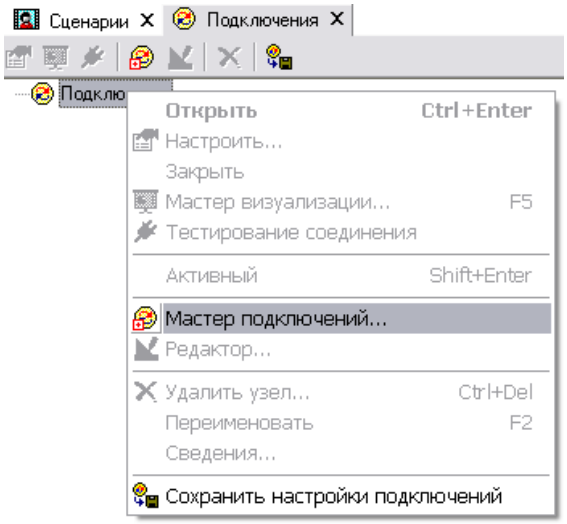
В таблице «Список городов» (табл. 10) поле Город является измерением, а Экономический район и Федеральный округ являются его атрибутами.

Стоит отметить, что таблицу «Продажи» можно объединить с таблицей «Скидки» с помощью обработки «Слияние с узлом». Целесообразность данного действия заключается в том, что в данных таблицах хранятся практически одни и те же данные, различие лишь в том, что в одной таблице есть скидки, а в другой – сумма с учетом скидок, поэтому для наглядного отображения данных решено два процесса объединить в один процесс – Продажи.

Таким образом, было выделено два основных процесса: Поступление и Продажи товаров.

Основные этапы создания ХД в АП Deductor:

1. Для создания нового хранилища данных в Deductor или подключения к существующему нужно перейти на закладку «Подключения» и запустить «Мастер подключений»

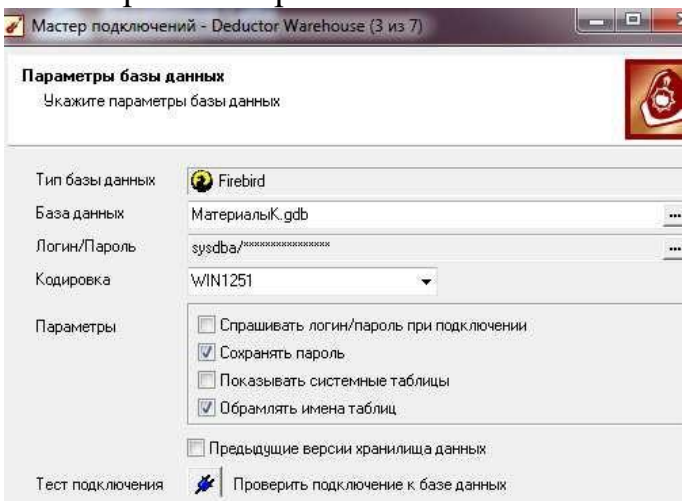


Создание (подключение) хранилища данных

2. Пройти первые два шага, выбрав тип приемника (источника) Deductor Warehouse и тип базы данных Firebird.

3. На третьем шаге нужно задать параметры базы данных, в которой будет создана физическая и логическая структура хранилища данных: база данных – материалы.gdb;

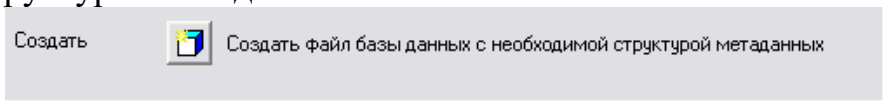
• логин – sysdba, пароль – masterkey;
• установить флажок Сохранять пароль.



Установка параметров базы данных

4. На следующем шаге выбирается версия для работы с ХД Deductor Warehouse.

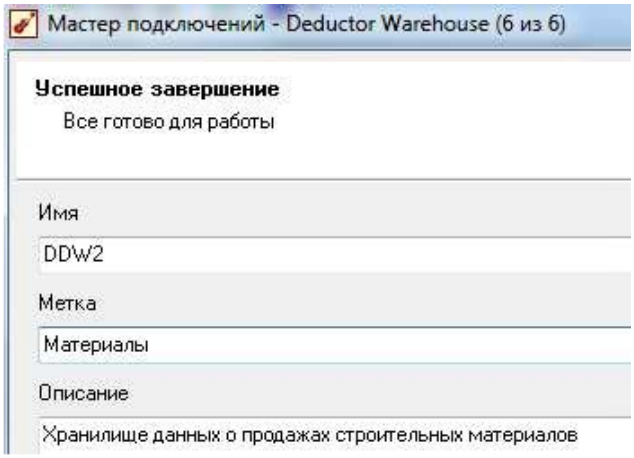
5. На пятом шаге нажать кнопку Создать файл базы данных с необходимой структурой метаданных



Вкладка Мастера подключения «Инструменты работы с ХД»

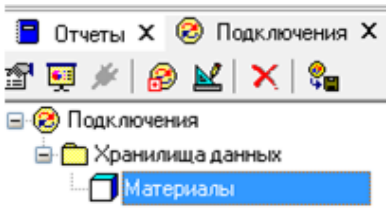
При этом выборе по указанному ранее пути будет создан файл материалы.gdb. (появится сообщение об успешном создании). Это и есть пустое хранилище данных, готовое к работе.

6. На последних двух шагах осталось выбрать визуализатор для подключения (здесь это Сведения и Метаданные) и задать для нового хранилища имя «material», метку «Материалы» и описание «Хранилище данных с информацией о продажах».



Настройка семантики узла подключения

7. После нажатия на кнопку Готово на дереве узлов подключений появится метка хранилища.



Хранилище данных «Материалы»

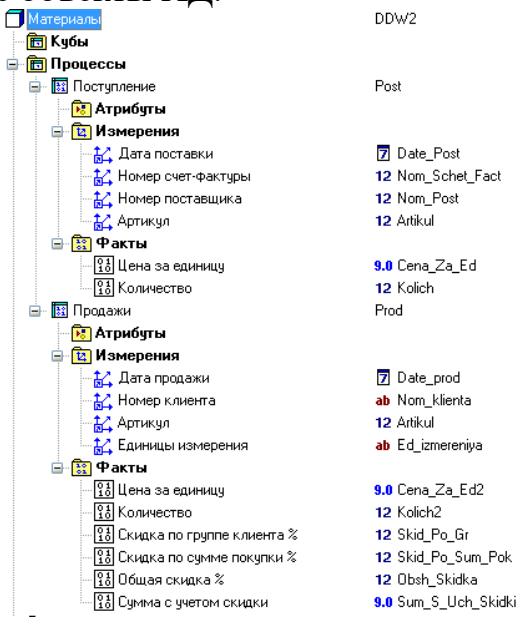
8. Для проверки доступа к новому ХД воспользуйтесь кнопкой . Если спустя некоторое время появится сообщение «Тестирование соединения прошло успешно», то хранилище готово к работе. Иначе нужно внести изменения в параметры подключения ХД, используя кнопку .

9. Сохраните настройки подключений, нажав на кнопку .

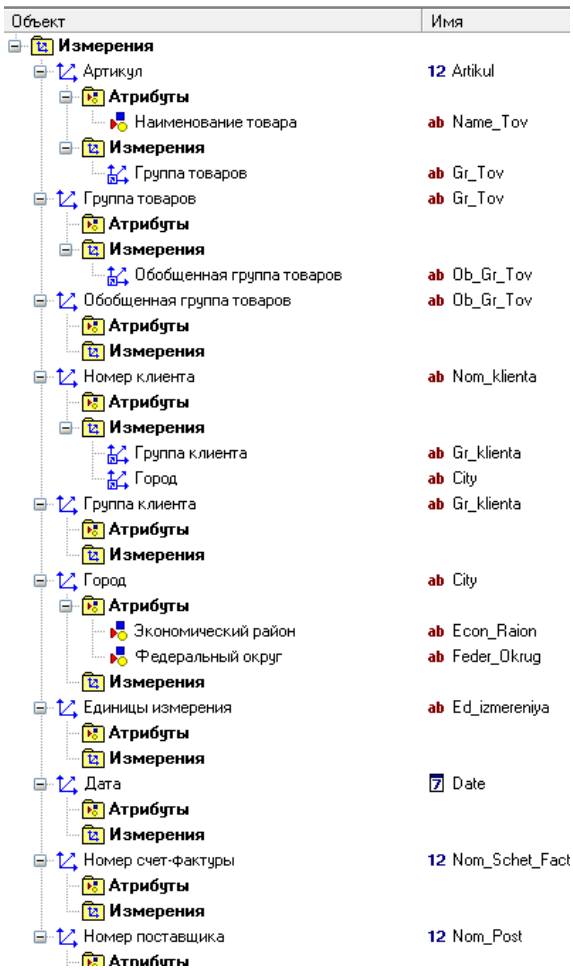
10. Для проектирования структуры ХД вызвать **Редактор метаданных** кнопкой на вкладке **Подключения**.

11. В открывшемся окне редактора нужно нажать кнопку (разрешить редактирование).

12. Встав на узле **Измерения**, при помощи кнопки **Добавить** добавьте в метаданные по очереди все объекты ХД.



Структура хранилища данных



Структура ХД

Наполнение ХД

Структура хранилища данных представляет собой «пустое» ХД Deductor Warehouse с настроенным семантическим слоем. В таком виде оно готово к загрузке в него данных из внешних структурированных источников. Для этого необходимо написать соответствующий сценарий в Deductor Studio.

Сценарий загрузки должен выполнять следующие функции:

1. Импорт данных в Deductor Studio из базы данных, учетной системы или предопределенных файлов;
2. Опциональная предобработка данных, например очистка или преобразование формата;
3. Загрузка данных в измерения и процессы хранилища Deductor Warehouse.

Исходными данными для ХД служат 10 текстовых файлов:

Артикул.txt, Группа клиентов.txt, Группа товаров.txt, Единицы измерения.txt, Клиенты.txt, Номер клиента.txt, Обобщенная группа товаров.txt, Приход.txt, Скидка.txt, Список городов.txt. Поэтому сценарий загрузки должен быть настроен на использование в качестве источников данных на эти файлы.

При создании сценария необходимо строго придерживаться следующих правил:

1. Первыми загружаются все измерения, имеющие атрибуты. Только после загрузки всех измерений загружаются данные в процессы.

2. Также имеется правило на порядок загрузки: загружать измерения нужно, начиная с самого верхнего уровня иерархии и спускаться по иерархии ниже, в противном случае иерархия не будет создана.

3. Допускается не загружать отдельно измерения, не имеющие атрибутов и не состоящие в иерархии измерений. Значения таких измерений можно при использовании специальной опции создавать во время загрузки в процесс.

В ходе наполнения ХД данными могут быть некоторые ошибки. Ниже представлено описание двух типов ошибок и пути их решения:

- ошибка 303 возникает в случае, когда длина поля в ХД не соответствует длине этого же поля в текстовом файле, т.е. получается ситуация, когда длинное название не помещается в хранилище из-за мало выделенного под это название места. Возможное решение: просмотреть текстовые файлы и найти поля, которые на первый взгляд являются очень длинными. В режиме редактирования хранилища удалить и заново добавить необходимый атрибут/измерение и увеличить длину поля со стандартно заданных 100 символов до нужного размера.
- ошибка 206 возникает, когда у наполненного данными хранилища меняют структуру и снова наполняют его, не очистив хранилище от старых данных. Возможное решение: заново добавить текстовый файл, необходимый для загрузки данных в хранилище и удалить, а затем повторно добавить необходимые измерения/атрибуты, либо в режиме редактирования структуры ХД воспользоваться командой «Очистить» и удалить ненужные старые данные из выбранного измерения в Хранилище данных.

Ниже представлена схема загрузки данных:

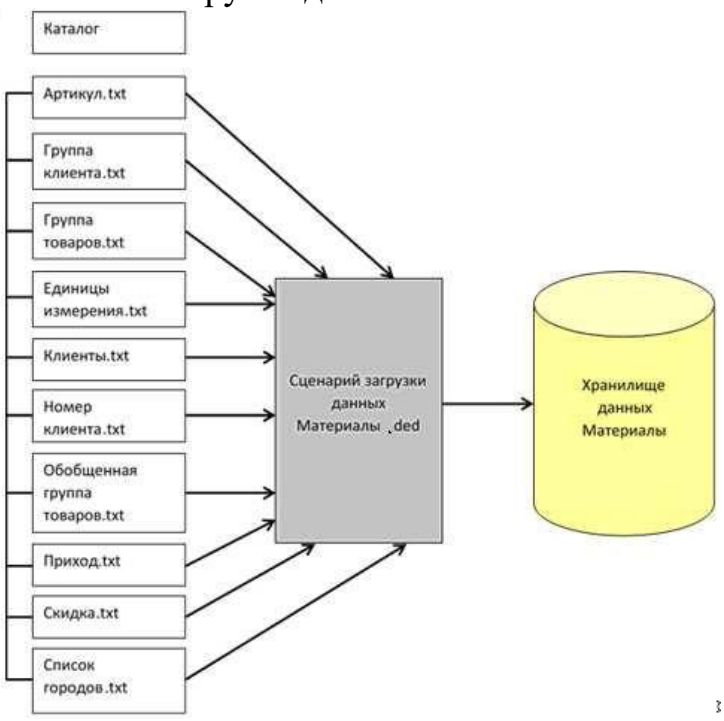
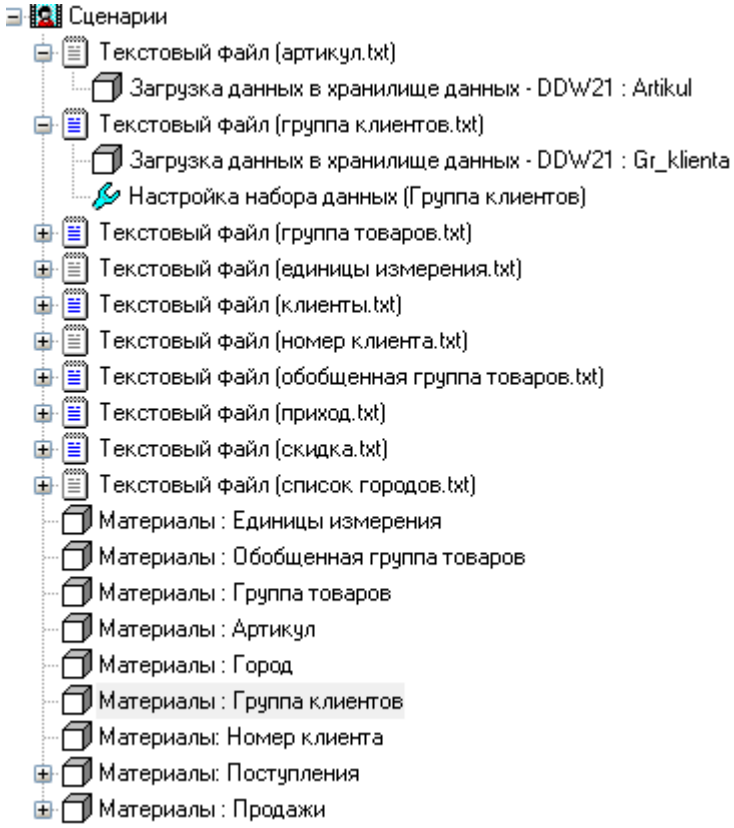


Схема загрузки данных



Сценарий загрузки данных в хранилище

Такого рода сценарий привязан не к самим данным, а только к их структуре, то есть в нем смоделирована последовательность действий, которую нужно выполнить для загрузки данных в хранилище: указаны имена файлов источников, соответствие полей и т.д. Таким образом, сценарий может использоваться неоднократно для пополнения ХД.

Созданное ХД позволяет обеспечить целостность и непротиворечивость данных, их централизованное хранение, автоматически обеспечивает всю необходимую поддержку процесса анализа данных.

В завершении работы с ХД нужно выполнить выгрузку данных из хранилища, чтобы убедиться в правильности загруженной информации и сравнить выгруженные файлы с текстовыми файлами по количеству элементов.

Из внешних источников данных информация в соответствии с некоторым регламентом должна перемещаться в ХД. Автоматическая загрузка в ХД настраивается с помощью пакетной обработки.

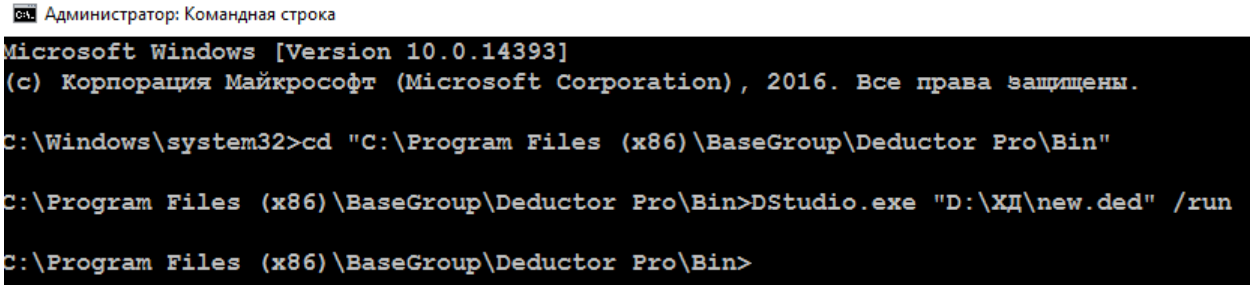
Указания

1. Чтобы запустить пакетное выполнение сценария с помощью командной строки, необходимо зайти в папку bin, которая располагается в директории установленной программы, воспользовавшись командой windows интерпретатора

–cd (change directory). Форма записи:

> cd “C:\Program Files (x86)\BaseGroup\Deductor Pro\Bin”

2. После этого запустить исполняемый файл DStudio.exe, передав в качестве аргументов полный адрес к файлу сценария (в кавычках) и команду /run для запуска пакетной обработки: > DStudio.exe "D:\ХД\new.ded" /run



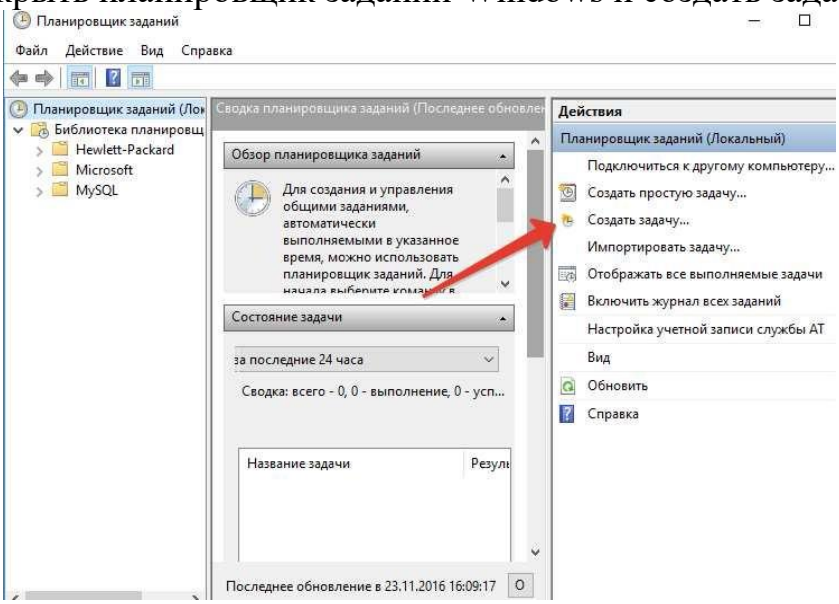
3. Если сообщения об ошибке нет, значит команда отработала. Далее необходимо проверить, появилась ли запись в ХД.

Примечание. Для корректной работы пакетной обработки необходимо, чтобы все файлы (сценарий, ХД, excel) лежали на диске D, т.к. политика безопасности диска С не всегда позволяет выполнить пакетную обработку правильно. Важно: При переносе с диска С на диск D убедитесь, что в самом сценарии для excel и ХД файлов указаны относительные пути, иначе при обработке файлы не будут найдены и новые записи в самом ХД не появятся.

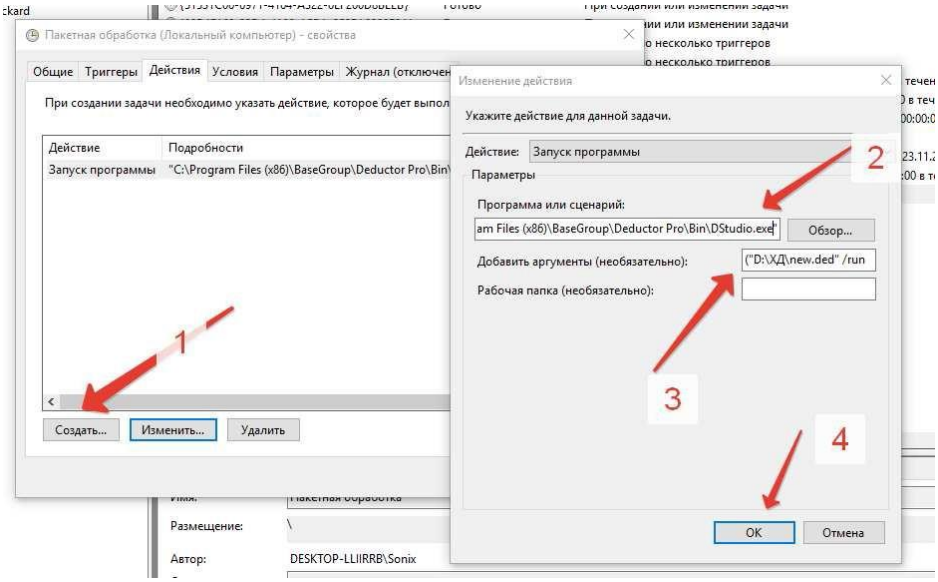
4. Пакетное выполнение настроить на запуск по расписанию с помощью планировщика заданий, например стандартного Windows Scheduler. Такая возможность удобна для автоматического запуска процесса загрузки в хранилище данных из учетной системы в нужное время. Для этого создается ярлык для файла DStudio.exe, для которого в строке «Объект» вводится командная строка запуска Deductor Studio в пакетном режиме. Затем в Windows Scheduler настраивается время запуска этого задания.

Порядок действий:

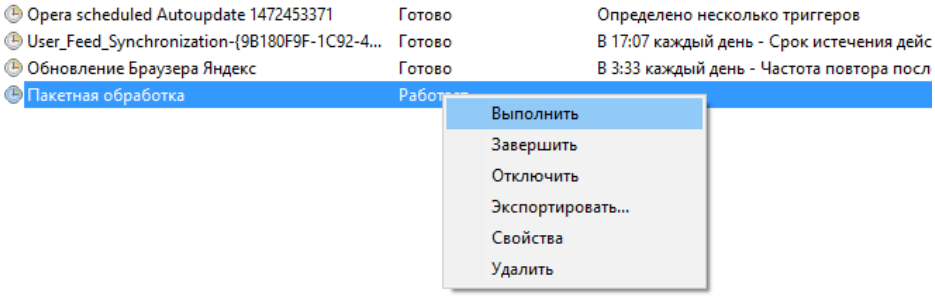
1. Открыть планировщик заданий Windows и создать задачу:



2. Ввести название задачи, перейти на вкладку действия и выбрать программу для запуска (C:\Program Files (x86)\BaseGroup\Deductor Pro\Bin\DStudio.exe);
3. В качестве аргументов передать те же самые, что и при запуске через командную строку ("D:\ХД\new.ded" /run);
4. Нажать «ок»:



5. Перейти в библиотеку планировщика, найти созданную задачу, вызвать контекстное меню и нажать «Выполнить».



Планировщик заданий необходим в том случае, когда требуется периодическая выгрузка данных в СППР, в нашем случае в АП Deductor. Данный механизм мог бы пригодиться для загрузки данных о покупках и продажах товаров в ХД.

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.

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.

Дата и время

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	Кварт	Меся	Неде	День	День недел	День недели
01.01.20	01.01.20	01.01.20	01.01.20	1	1	1	1	4	4 Четверг
09.01.20	01.01.20	01.01.20	05.01.20	1	1	2	9	5	5 Пятница
17.01.20	01.01.20	01.01.20	12.01.20	1	1	3	17	6	6 Суббота
25.01.20	01.01.20	01.01.20	19.01.20	1	1	4	25	7	7
02.02.20	01.01.20	01.02.20	02.02.20	1	2	6	33	1	1
10.02.20	01.01.20	01.02.20	09.02.20	1	2	7	41	2	2 Вт ор н
18.02.20	01.01.20	01.02.20	16.02.20	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

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	Количеств
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	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	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.

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
…	…

Квантование

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
…	…	…

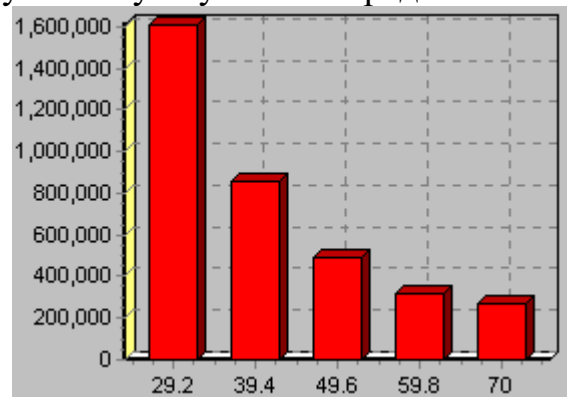
Статистика показывает, что возраст кредиторов лежит в диапазоне от 19 до 70 лет. Разобьем возраст на 5 равных интервалов, заменив возраст номером интервала.

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

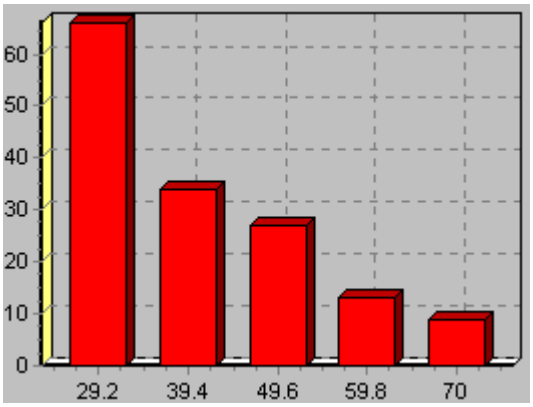
Получим таблицу.

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.

Калькулятор

С помощью компонента «Калькулятор» в исходную выборку могут быть добавлены поля, значения которых вычисляются по формуле из значений других полей.

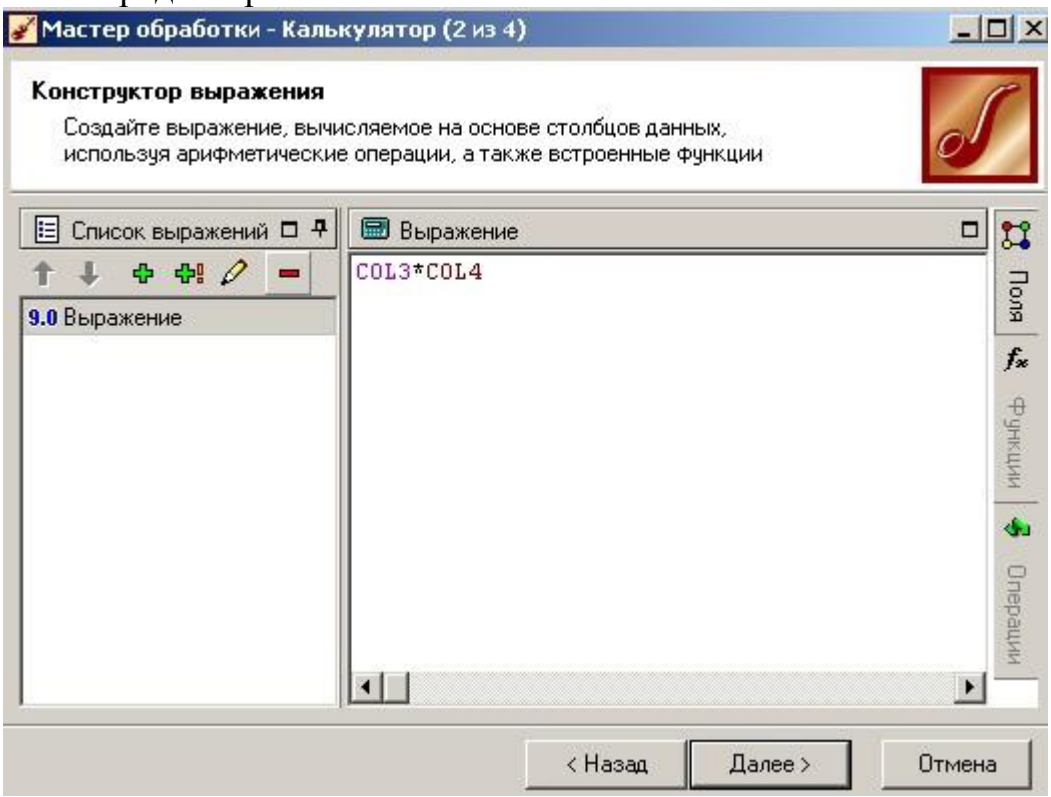
Для создания выражения нужно выполнить следующие действия.

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.



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.

Пример

Пусть дана исходная таблица.

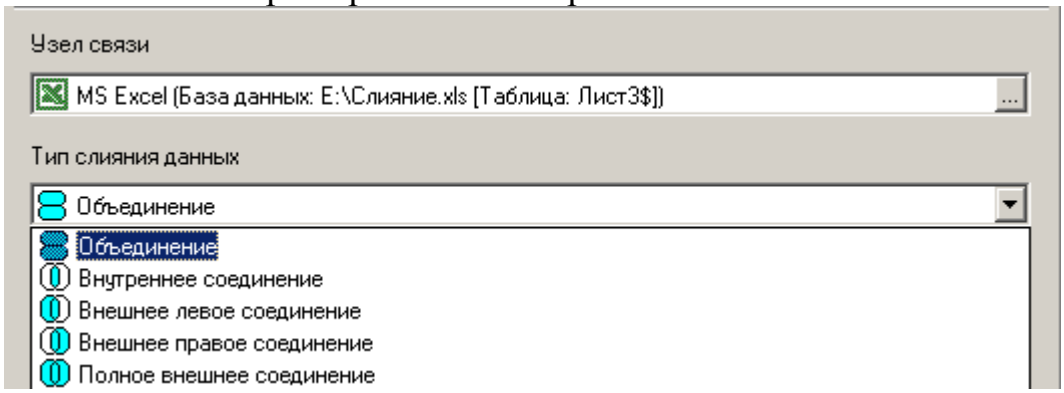
Поставщик	Товар
ЖБИ	Бетон
ЖБИ	Плита
КРЗ	Рубероид
КРЗ	Картон

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

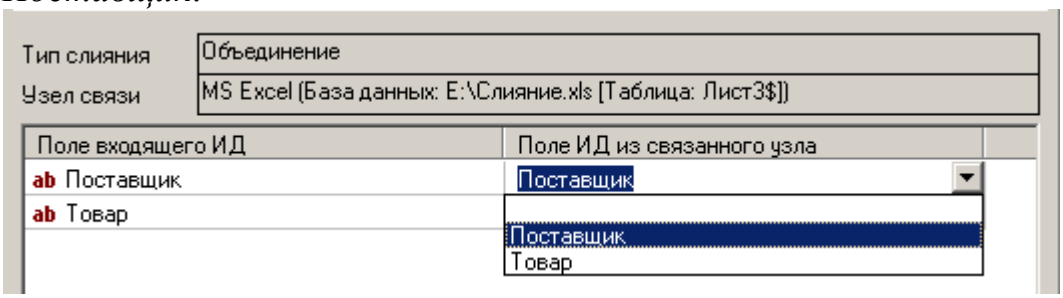
Дата	Поставщик	Товар	Количество
10.02.2004	ЖБИ	Бетон	100
10.03.2004	КРЗ	Рубероид	10
10.03.2004	КРЗ	Рубероид	20
10.03.2004	ЖБИ	Плита	5
10.03.2004	ЖБИ	Бетон	130
11.03.2004	КРЗ	Картон	20

A. Объединение

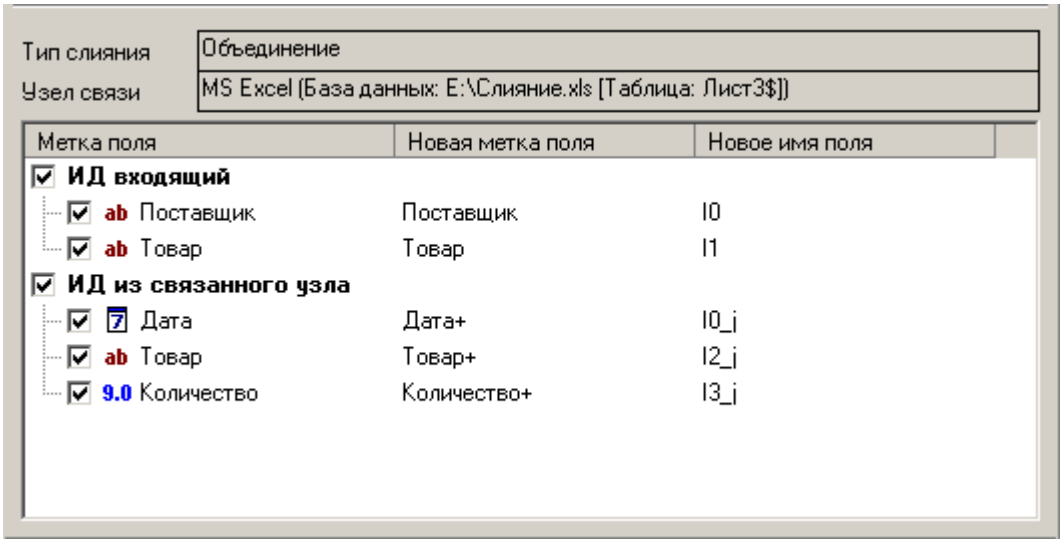
На первом шаге в Мастере обработки выбираем тип слияния «*Объединение*»:



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.

B. 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*, получим 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 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				

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 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
		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.

E. 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.

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	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

Если узел с кросс-таблицей планируется использовать в скрипте или групповой обработке, рекомендуется активировать флаг **Скользящие уникальные значения**.

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

Сэмплинг и Разбиение на множества

Назначение

На практике в бизнес- аналитике применяется подход, когда анализу подвергаются не все имеющиеся данные, а только некоторое их подмножество. Это обусловлено многими причинами, например, необходимость выделения записей, которые не будут принимать участие в построении модели, а будут использоваться в ее тестировании. Кроме того, трудоемкость некоторых алгоритмов Data Mining экспоненциально возрастает с увеличением объема данных, поэтому обработка даже относительно небольших выборок такими алгоритмами требует неприемлемо высоких временных затрат.

Узел «Сэмплинг» реализует различные способы отбора записей (единиц) в выборку из набора данных (генеральной совокупности). В Data Mining это называется *сэмплингом*.

Узел «Разбиение на множества» основан на алгоритмах узла «Сэмплинг»: процедура сэмплинга выполняется для формирования обучающего множества, а тестовое множество получается путем вычитания из исходной совокупности обучающего множества. Единственное отличие между узлами в том, что узел разбиения на множества формирует дополнительное поле логического типа «Признак тестового множества».

Принцип работы

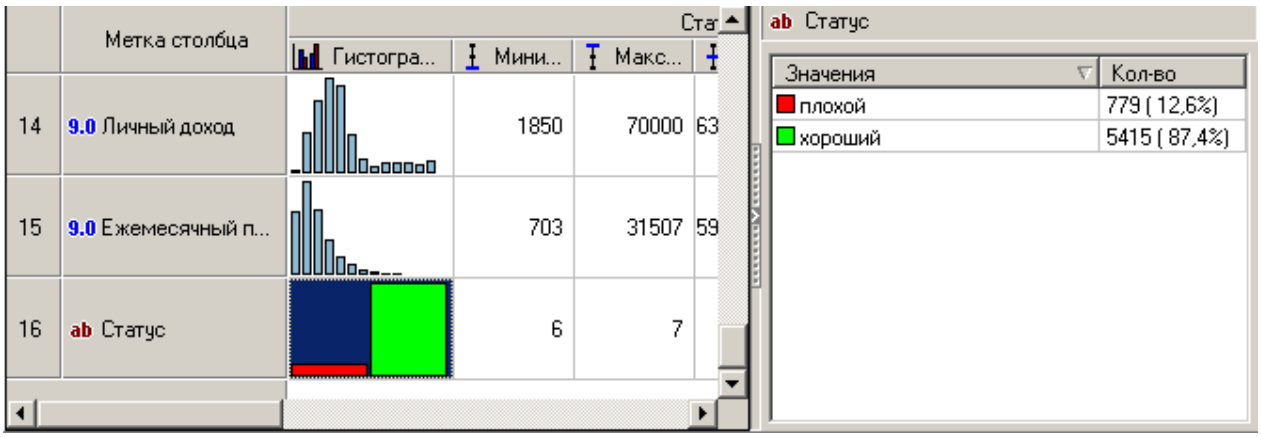
Существуют различные стратегии извлечения подмножеств наблюдений из исходных множеств данных. В узле сэмплинг реализовано пять методов сэмплинга.

- *Случайный* – выборка производится случайным образом из всей совокупности;
- *Равномерный случайный* – все записи исходной совокупности разделяются на группы, в каждой из которых содержится одинаковое число записей. Затем из каждой группы случайным образом выбирается одна запись и помещается в результирующую выборку. Выборка, полученная в результате сэмплинга, будет состоять из записей, случайным образом отобранных из каждой группы.
- *Последовательный* – выборка производится последовательным образом из всей совокупности, пока не будет достигнут требуемый объем.
- *Стратифицированный* – процесс стратифицированного сэмплинга состоит из двух шагов: формирование страт (в общем случае многомерных, тогда каждая страта образуется комбинацией уникальных характеристик столбца) и проведение случайного сэмплинга внутри каждой страты. В каждой из слоев используется так называемая выборочная доля, пропорциональная составу всей исходной совокупности.
- *Отбор со смещением* – так называемое перевзвешивание примеров, или изменение соотношения принадлежности записей к классам.

Осуществляется путем размножения или удаления примеров каждого класса. Используется в предсказательном моделировании при решении проблем редких классов.

Пример

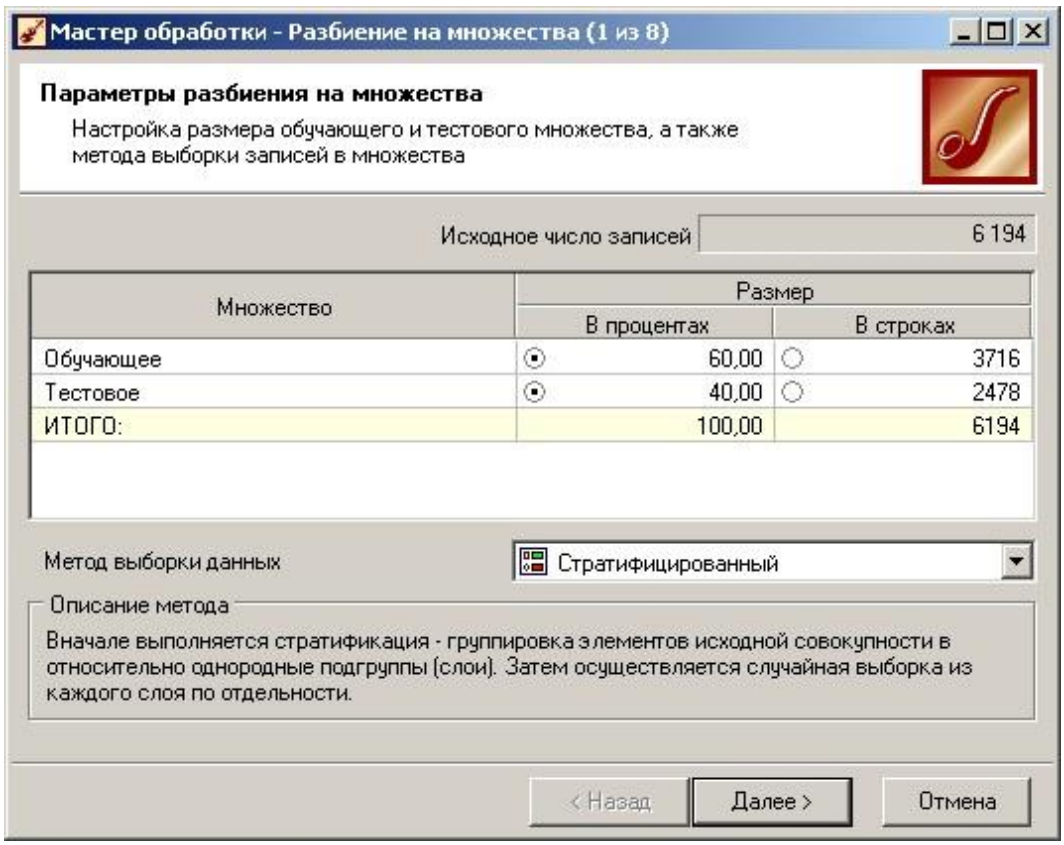
Продолжим пример с кредитным скорингом. После проведения аудита данных и корректировки выбросов число записей составило 6194. Доля плохих кредитных счетов составляет чуть более 12%.



Перед построением моделей оценки кредитоспособности нужно разделить весь набор данных на обучающее (60 %). Пусть, помимо этого, требуется сделать перевзвешивание: увеличить долю плохих счетов, но только в обучающей выборке, а в тестовом множестве – получить примерно такую же пропорцию плохих счетов, как и в исходном наборе данных.

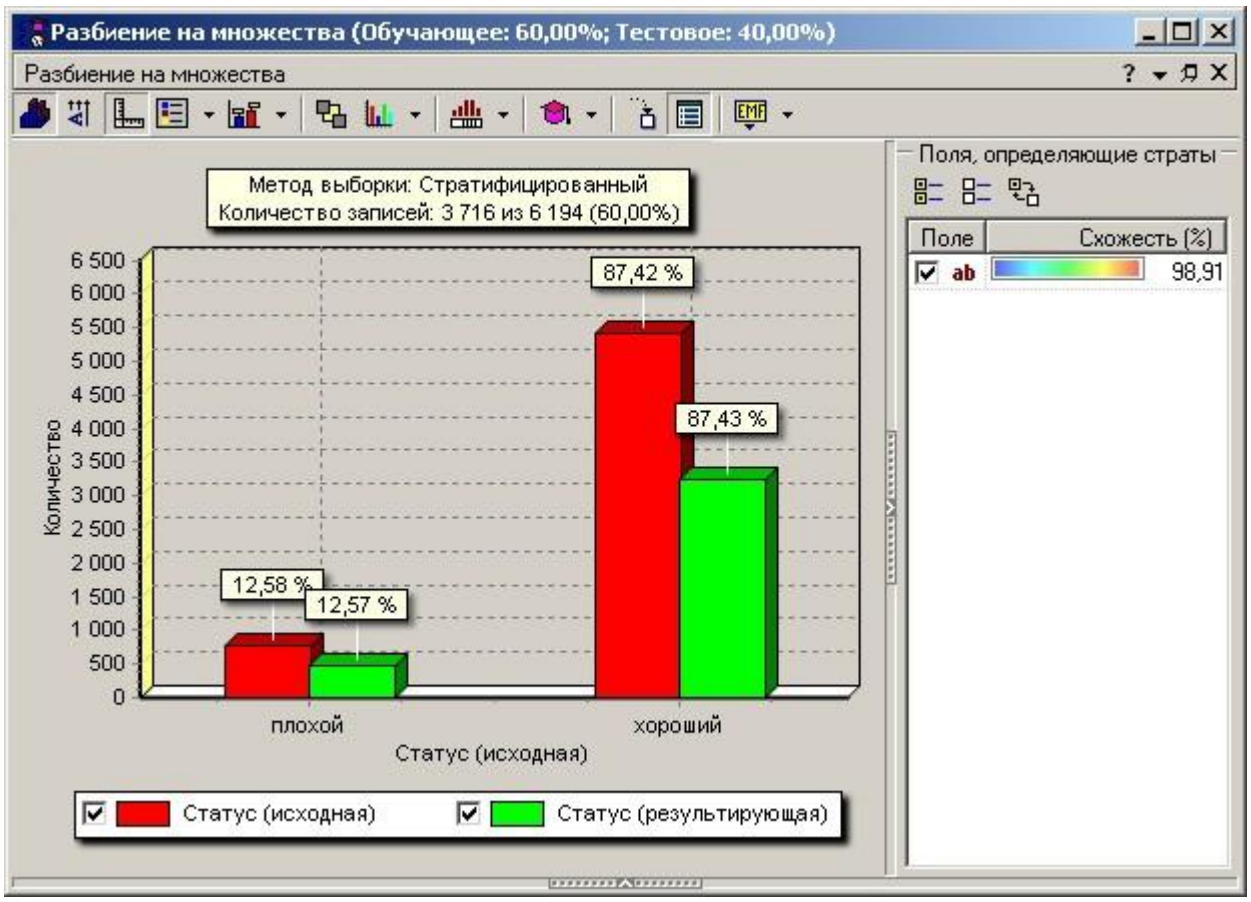
Сначала разделим набор данных на обучающий и тестовый. Откроем мастер обработки узла

«Разбиение на множества» и выставим настройки, как это показано на рисунке.



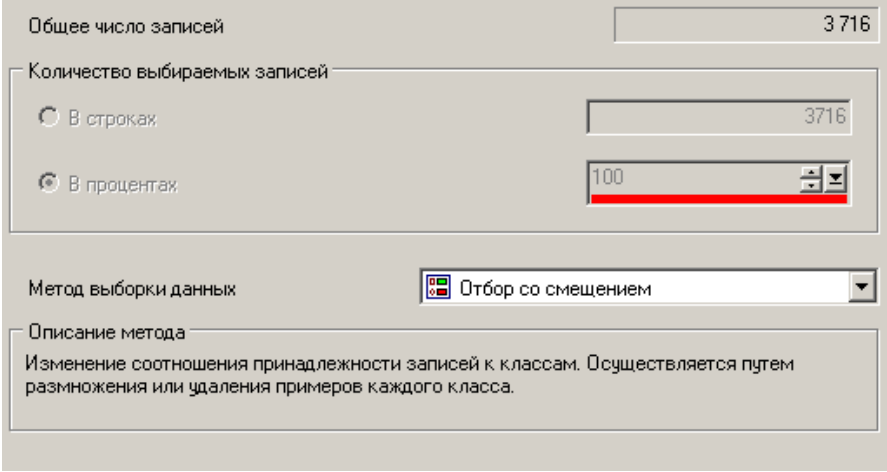
Поскольку нам нужно обеспечить после разбиения долю плохих счетов в районе 12 %, лучше выбрать стратифицированный сэмплинг, а на следующем шаге указать поле «Статус» в качестве поля, определяющего страту.

После запуска процесса обработки откроем специализированный одноименный визуализатор к этому узлу, который позволит понять, как прошло разбиение на множества, сравним долю плохих счетов в обучающем и тестовом множествах.

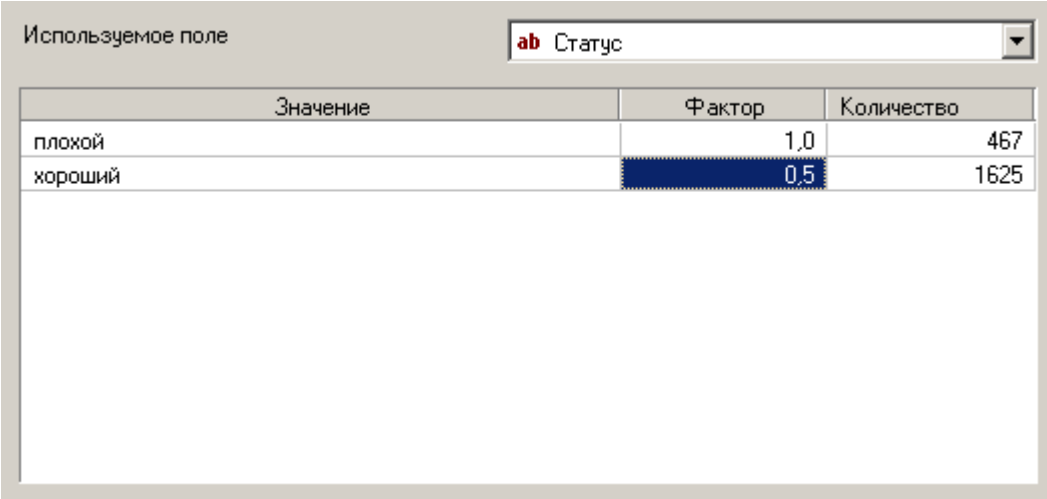


Анализ диаграммы показывает, что требуемая цель стратификации была достигнута.

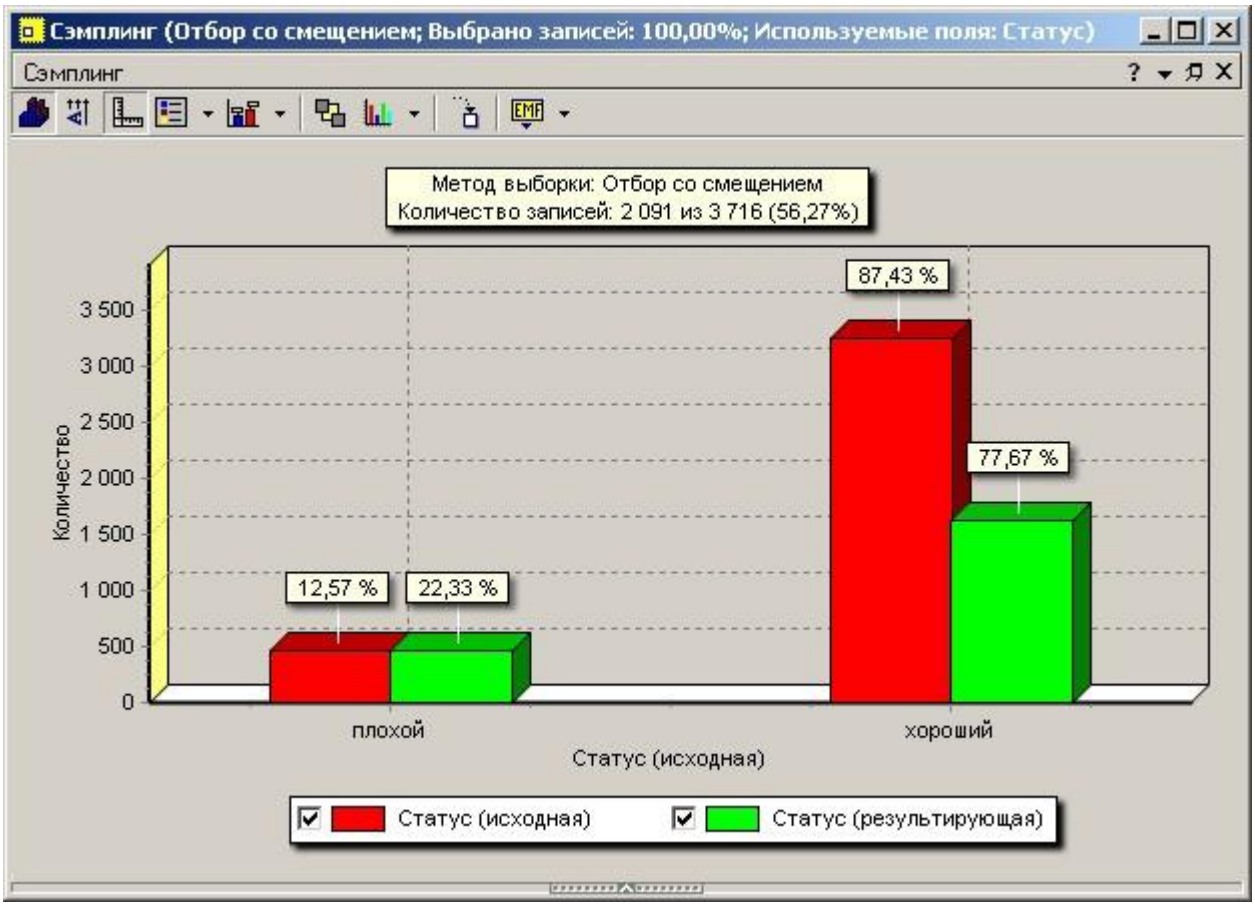
Теперь проведем перевзвешивание. При помощи узла «Фильтр» выделим только обучающее множество и к нему применим сэмплинг – отбор со смещением путем удаления каждой второй записи со статусом «хороший».



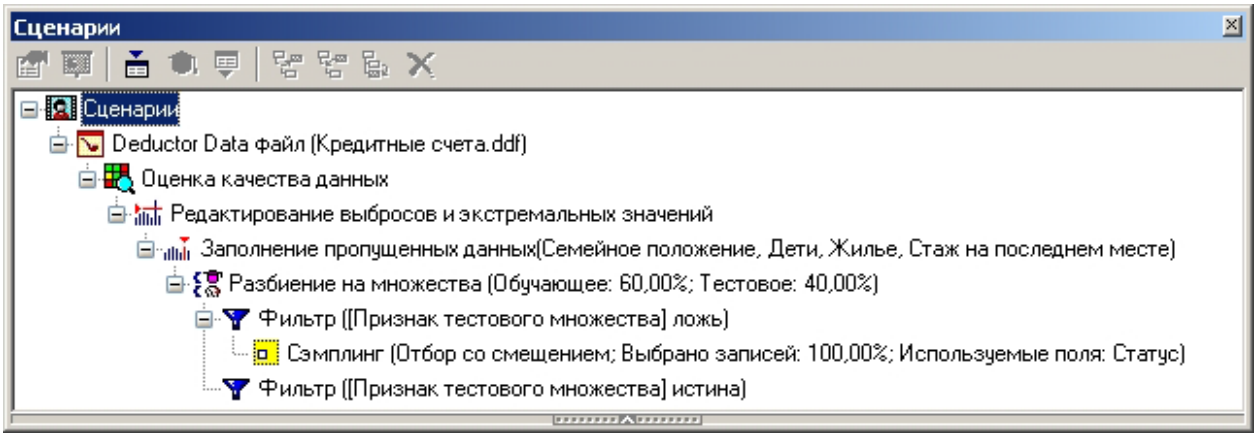
На следующем шаге мастера укажем, что по полю «Статус» у значения « хороший» фактор 0,5, то есть во сколько раз нужно изменить количество таких записей.



В результате получили новый набор данных, в котором доля плохих счетов повысилась с 12,57% до 22,33%.



В итоге сценарий будет выглядеть следующим образом.



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	Объем прода
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	Объем продаж 2	Объем продаж месяц	Объем продаж в	Объем продаж на
				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	Объем продаж 2	Объем продаж 1	Объем продаж в текущий	Объем продаж на следующий
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.

Конечные классы

Назначение

Данный обработчик предназначен для решения следующих задач**.**

1. Преобразование непрерывных и дискретных входных полей, используемых для построения моделей бинарной классификации, путем квантования на основе метода совокупности доказательств или *WoE* -анализа (weights of evidence, *WoE*). В результате каждое
2. исходное значение признака заменяется на метку интервала квантования, в который данное значение попало. Использование результатов такого преобразования для построения моделей бинарной классификации (например, логистической регрессии), позволяет повысить их точность и устойчивость к изменению входных данных.
3. Сокращение размерности данных за счет исключения признаков с низкой значимостью, снижения разнообразия значений признаков.
4. Восстановление пропусков, когда пропуски образуют отдельную метку интервала квантования, или объединяются с соседним, близким по значению WoE-индекса.
5. Борьба с выбросами и экстремальными значениями – формирование меток интервала квантования при дискретизации непрерывного поля, или объединение редких уникальных значений в одну категорию позволяет решить проблему экстремальных значений и выбросов.
6. Результатом работы обработчика «Конечные классы» является преобразование входных столбцов в последовательность интервалов, называемых *конечными классами*, каждому из которых присваивается определенная метка. Кроме этого, для каждого входного столбца может быть вычислен уровень значимости (отсутствует, очень низкая, низкая, средняя, высокая и очень высокая), на основе которого может производиться отбор переменных в модели бинарной классификации.

Замечание

В контексте данного узла понятие «класс» несколько отличается от общепринятого. Если «класс» с точки зрения задачи классификации – это совокупность объектов с близкими значениями признаков, то в данном случае, класс – это интервал значений единственного признака, в который попадают наблюдения, «близкие» именно по этому признаку. Данный вид обработки в англоязычной литературе называется «classing».

Принцип работы

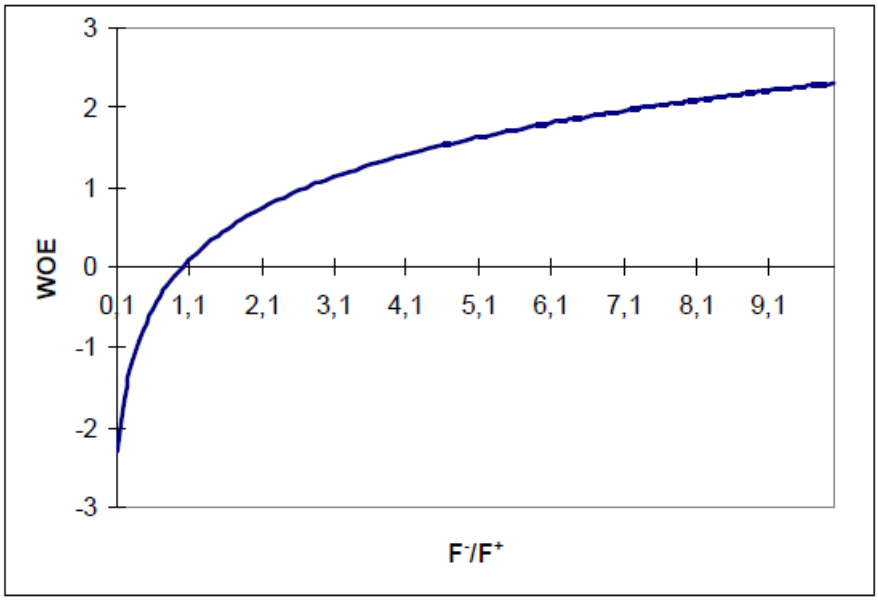
В основе идеи обработчика «Конечные классы» лежит метод *WoE* -анализа, применяющийся в различных задачах бинарной классификации в таких областях, как кредитный скоринг, медицинская диагностика, поиск полезных ископаемых и других, где требуется принятие бинарных управленческих решений.

Если имеется обучающая выборка, в которой каждому наблюдению, содержащему набор признаков, ставится в соответствие бинарная целевая переменная класса, то, в зависимости от логики задачи, одно из ее состояний считается *событием*, а другое – *не-событием*. Затем производится разбиение всего диапазона изменения того или иного признака на несколько интервалов, для каждого из которых вычисляется коэффициент *WoE*:

$$WoE_i = \ln \frac{(N_i / N)}{(P_i / P)} = \ln \frac{F^-}{F^+}$$

где *i* – индекс начального класса, *Ni* - число не-событий, попавших в интервал, *N* – общее число не событий в исходном наборе данных, *Pi* – число событий, попавших в класс, *P* – общее число событий.

Интерпретация коэффициентов *WoE* достаточно проста. В числителе отношения под логарифмом стоит относительная частота появления не событий в классе *F–*, а знаменателе – относительная частота появления событий *F+*. Если *F–* > *F+*, то логарифм их отношения также больше 0 (логарифмическая зависимость для вычисления коэффициентов *WoE* представлена на следующем рисунке).



Нетрудно увидеть, что положительные значения коэффициентов *WoE* указывают на б*о*льшую вероятность появления не-событий в интервале. Иными словами, если для некоторого нового наблюдения значение соответствующего столбца оказывается в интервале с положительным *WoE*, то наиболее вероятным для него будет наступление не-события. Напротив, если относительная частота появления событий в интервале выше относительной частоты появления не-событий, то выражение под логарифмом становится меньше 0, а коэффициент *WoE* оказывается отрицательным. Следовательно, наиболее вероятно наступление события.

На основе коэффициентов *WoE* вычисляется величина, определяющая значимость признака в модели бинарной классификации, называемая информационным индексом (от англ. information value – *IV*) по формуле:

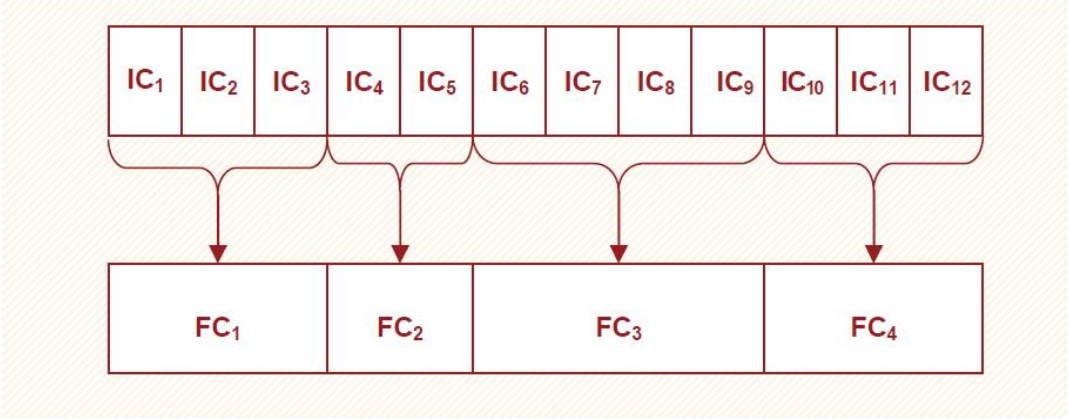
$$IV = \sum_{i=1}^{k} \left\{ \left(\frac{N_i}{N} - \frac{P_i}{P} \right) \cdot WoE_i \right\}.$$

Из формулы видно, что информационный индекс пропорционален разности относительных частот появления событий и не-событий в каждом интервале, просуммированной по всем интервалам. Информационный индекс всегда является положительной величиной. На основе информационного индекса определяется значимость признака по следующей методике:

- $IV < 0{,}02$ - отсутствует;
- $0{,}02 \le IV < 0{,}1$ - низкая;
- $0{,}1 \le IV < 0{,}3$ - средняя;
- $IV > 0{,}3$ - высокая.

Коэффициенты *WoE* и вычисленные на их основе значения информационного индекса используются в качестве критерия для формирования интервалов оптимальным образом: максимизируя значимость признака в бинарной классификационной модели или максимизируя равномерность заполнения интервалов, что обеспечивает наилучшую репрезентативность результирующей последовательности. Интервалы, полученные в результате разбиения исходного диапазона значения признака, и называются *конечными классами*.

Процедура формирования конечных классов содержит два этапа. На первом этапе формируются начальные классы, а на втором – начальные классы объединяются в конечные.



Объединение начальных классов (IC – initial class) в конечные (FC - finite class)

Для строковых и числовых столбцов способ формирования начальных классов несколько различается. В первом случае каждый начальный класс образуется единственным уникальным значением столбца, для каждого из которых определяется коэффициент *WoE*, после чего начальные классы упорядочиваются по его возрастанию.

Для непрерывных столбцов в обработчике предусмотрены два метода формирования начальных классов:

– *с предквантованием* – диапазон изменения значений в столбце разбивается на заданное число интервалов, каждый из которых формирует один начальный класс. Предквантование

может использоваться только для числовых признаков.

- *без предквантования* – конечный класс формируется каждым отдельным значением признака.

Затем конечные классы формируются путем объединения начальных классов, таким образом, чтобы максимизировать информационный индекс или равномерность, либо найти компромисс между ними.

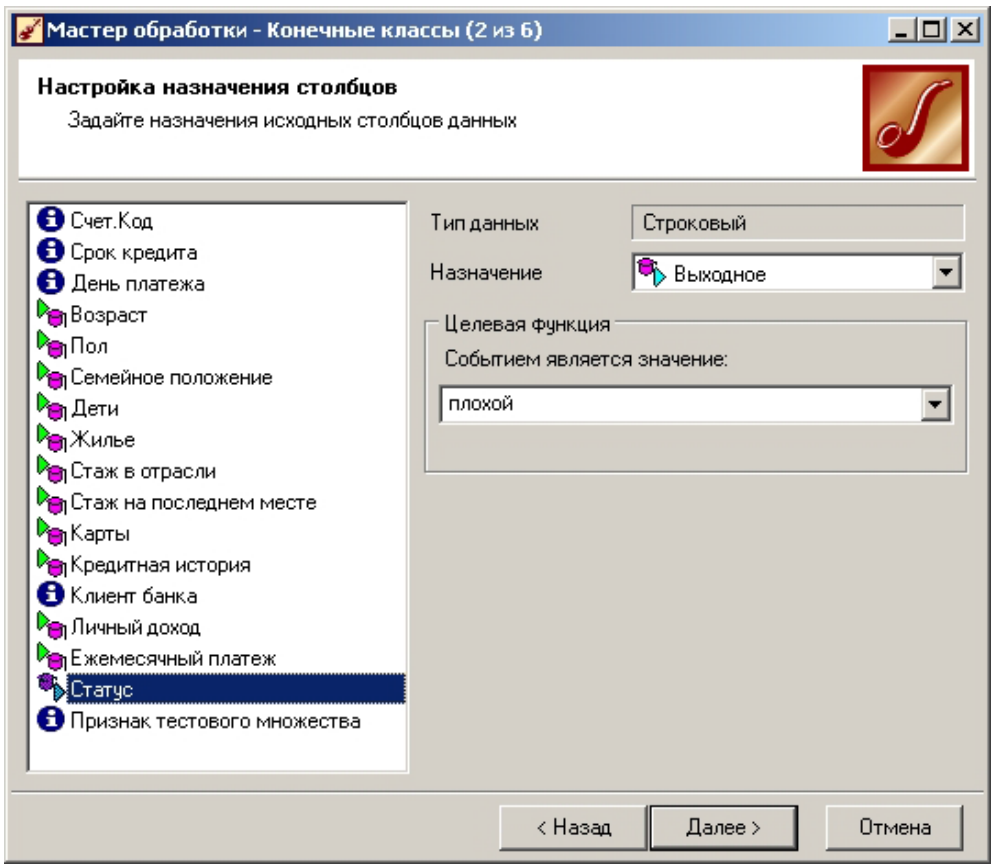
Пример

Продолжим пример с кредитным скорингом. Поскольку балльная скоринговая карта должна строиться только на предварительно квантованных полях, то лучший способ это сделать – предварительно обработать обучающую выборку узлом «Конечные классы».

Запустим мастер обработки и настроим поля, как это показано на рисунке. Выходным полем будет «Статус», со значением события «плохой».

Для каждого входного столбца задаем параметры формируемых для него конечных классов:

- *минимальный вес класса* – минимальный процент от общего числа наблюдений исходного набора данных, содержащихся в конечном классе (событий и не событий). Это нужно для того, чтобы избежать создания конечных классов, в которое попало мало наблюдений;
- *максимальное число классов* – наибольшее число конечных классов, которое может быть сформировано. Фактическое число сформированных классов может оказаться меньше заданного значения из-за ограничения по весу классов;
- *равномерность* – параметр, определяющий способ разбиения на классы. Если равномерность равна 0, то классы формируются таким образом, чтобы максимизировать значимость столбца (информационный индекс). Если равномерность равна 1, то разбиение на классы производится таким образом, чтобы в каждый из них попадало примерно одинаковое число наблюдений, что позволит обеспечить лучшую репрезентативность результирующего набора;
- *предквантование* – позволяет выбрать способ формирования начальных классов с предквантованием или без него. Если выбран способ с предквантованием, то требуется задать число уровней.



Результаты работы узла можно просмотреть в интерактивном визуализаторе **Конечные классы**, который включает в себя:

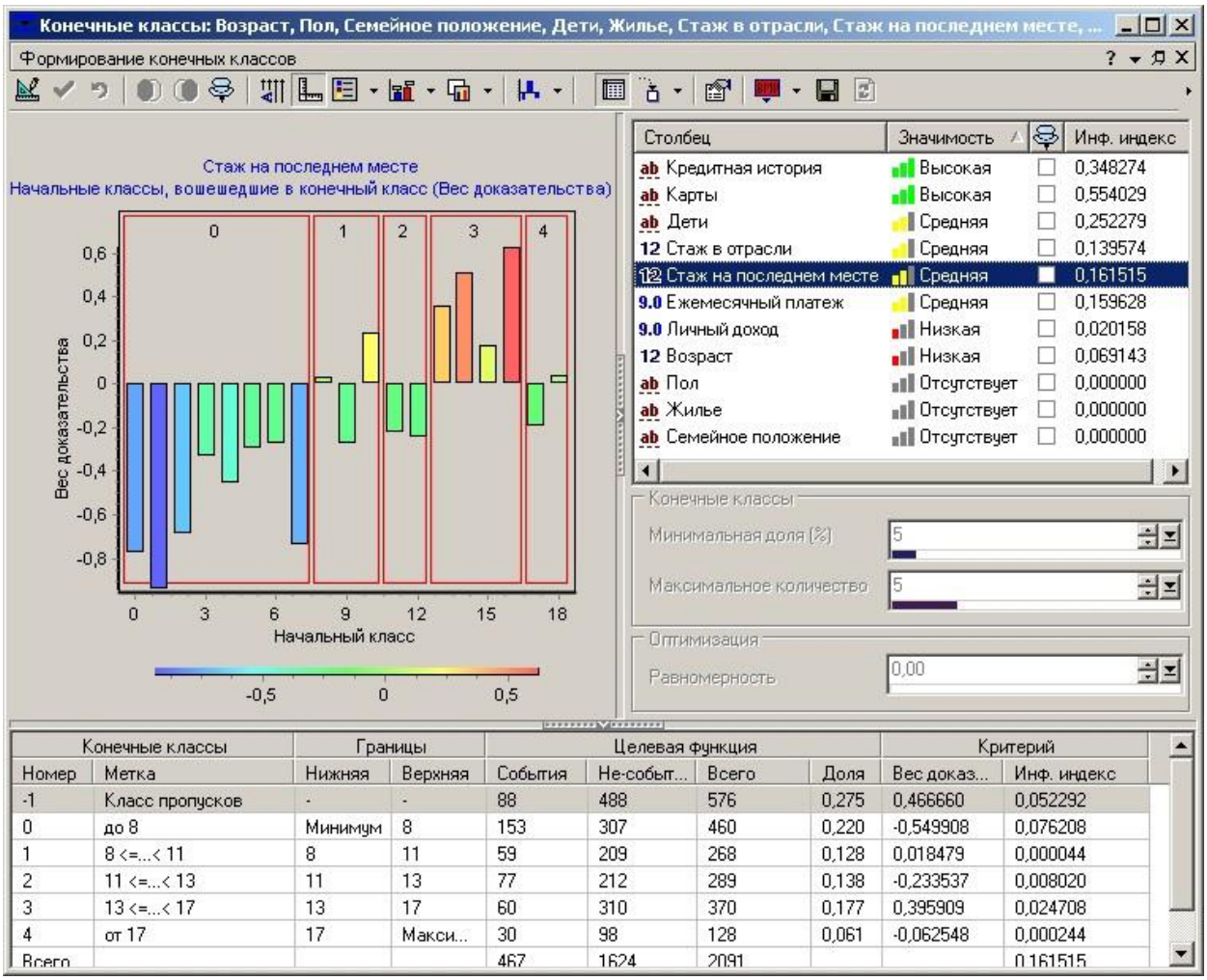
- диаграмму начальных классов;
- таблицу детализации;
- список входных столбцов, для каждого из которых указывается оцененный уровень значимости;
- секцию настройки параметров конечных классов (доступна только в интерактивном режиме).

Мы видим, что два поля имеют высокую значимость (это «Кредитная история» и «Карты»), четыре – среднюю и еще два – низкую. Остальные поля не имеют значимости, позволяющей говорить о наличии закономерностей между ими и статусом заемщика.

Наиболее ценный показатель в таблице детализации – это вес доказательства, или коэффициент *WoE*. Например, для стажа на последнем месте вычисленный *WoE* для пропущенных значений равен 0,46. Это значит, что заемщики с такими анкетами чаще склонны возвращать кредит и ведут себя похожим образом, как и заемщики со стажем от 13 до 17 лет.

Важным свойством данного визуализатора является то, что он

интерактивный, то есть позволяет изменить результаты автоматического формирования конечных классов. Это полезно в случаях, когда аналитика не устраивает квазиоптимальность алгоритма формирования классов, либо границы классов нужно сделать более интерпретируемыми в ущерб оптимальности.



Интерактивный режим включается (выключается) кнопкой на панели инструментов. В интерактивном режиме становятся доступными настройки параметров конечных классов и кнопки объединения классов . Любые внесенные изменению немедленно отражаются на диаграмме и в таблице детализации.

Переключим детализацию из таблицы на диаграмму *WoE*-индексов.

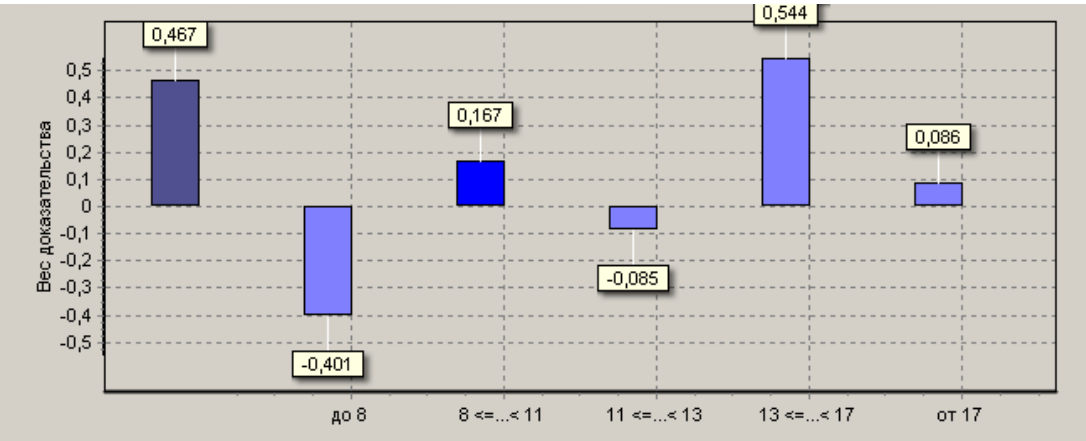
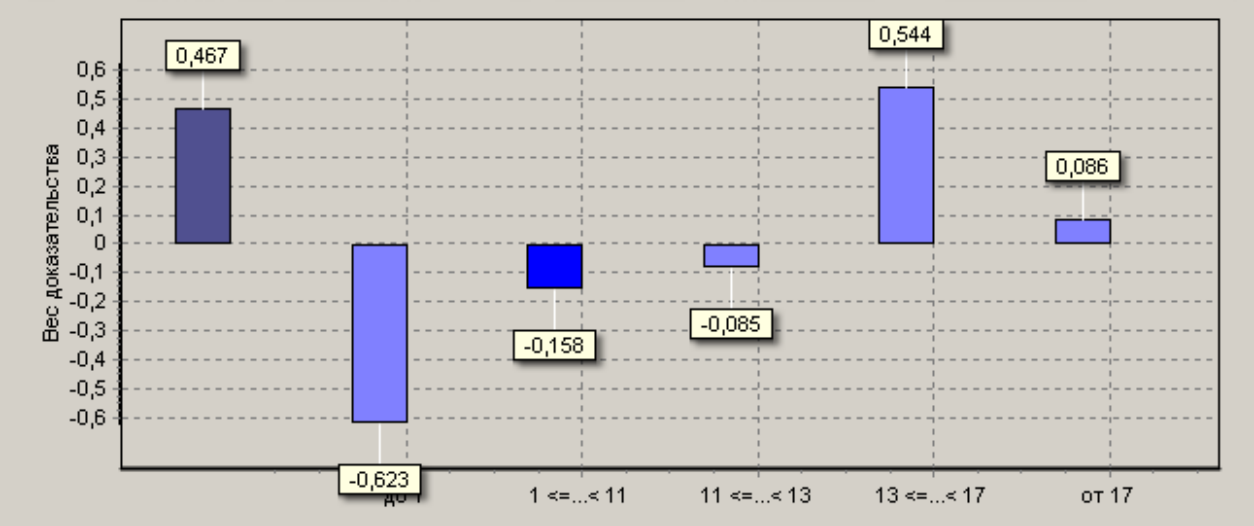


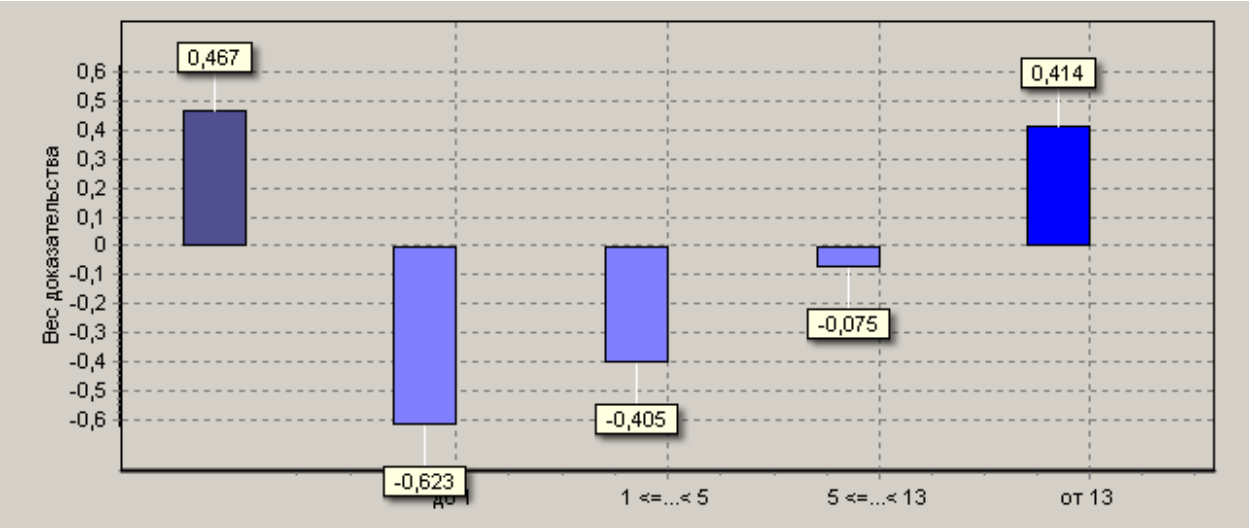
Диаграмма похоже на «пилу», четвертый и шестой конечные классы имеют значения индексов, близкие к нулю. От такой ситуации лучше уходить: классы должны максимально различаться по значениям *WoE*-индексов, а само количество классов быть минимальным.

Кроме того, пусть нас не устраивает первая граница – 8. Из практики известно, что риски потерять работу или быть уволенным на первом году достаточно высокие. Поэтому сдвинем границу влево до отметки 1.



Так и есть: *WoE*-индекс для конечного класса «до 1» стал ощутимо ниже нуля, что означает: заемщики с небольшим стажем на последнем месте представляют довольно рискованный сегмент. Однако поле «Минимальная доля» загорелось красным цветом: не выполнилось условие, что доля конечного класса должна быть не менее 5% от общего объема записей (она составила 4,8%).

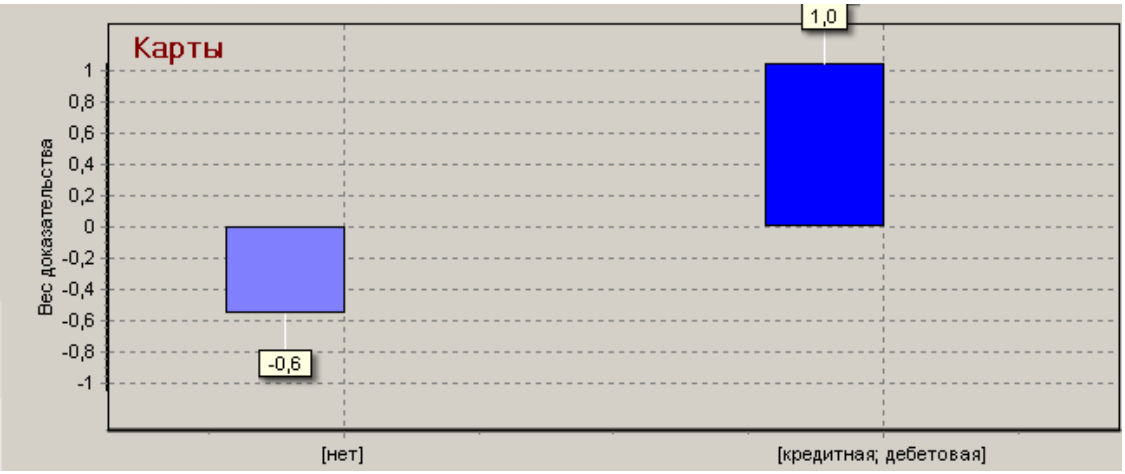
Проделаем следующие действия: вторую границу сдвинем на значение 5, а два последних – объединим. Получим новую диаграмму *WoE*-индексов.

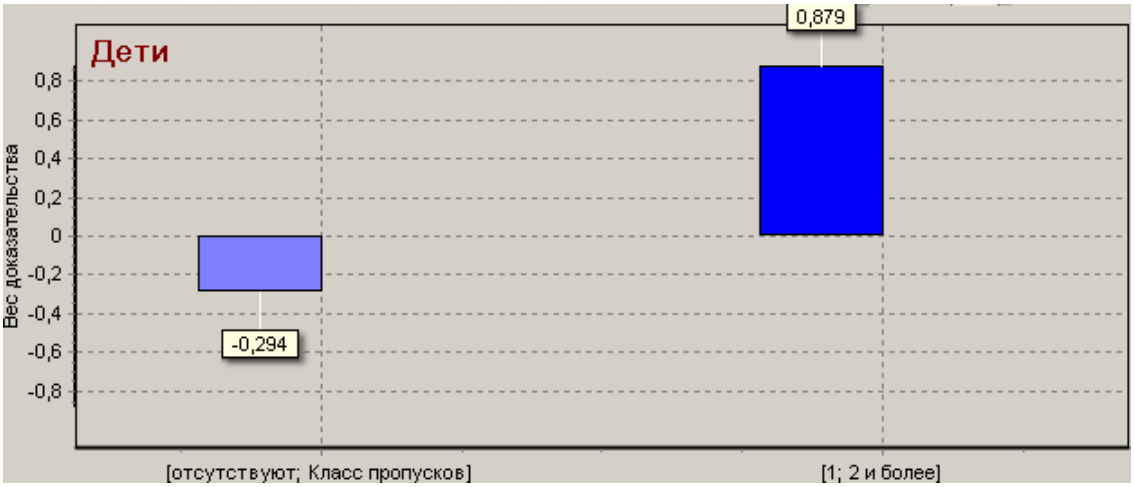


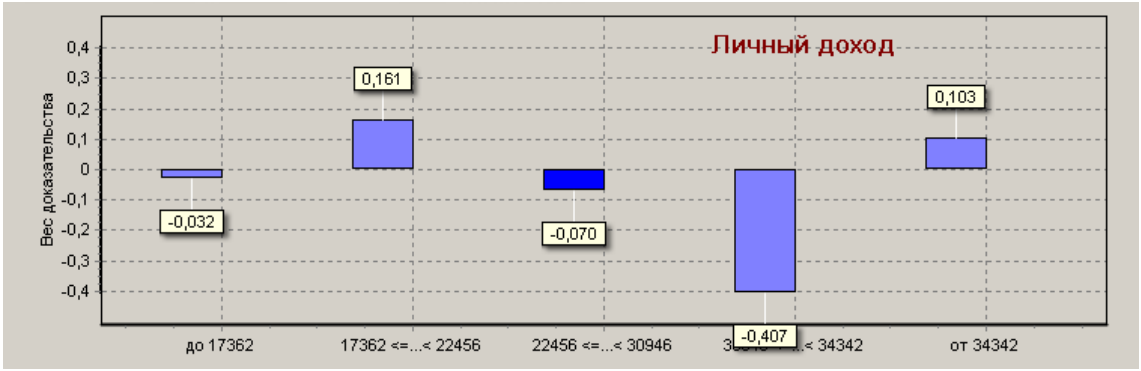
Она стала лучше с точки зрения логики и интерпретируемости и демонстрирует линейную зависимость: чем больше стаж на последнем месте, тем меньше риск возникновения проблем с обслуживанием счета. Это значит, что в скоринговой карте максимальные баллы будут начислены за стаж от 13 лет и выше. Однако информационный индекс переменной снизился с 0,16 до 0,14.

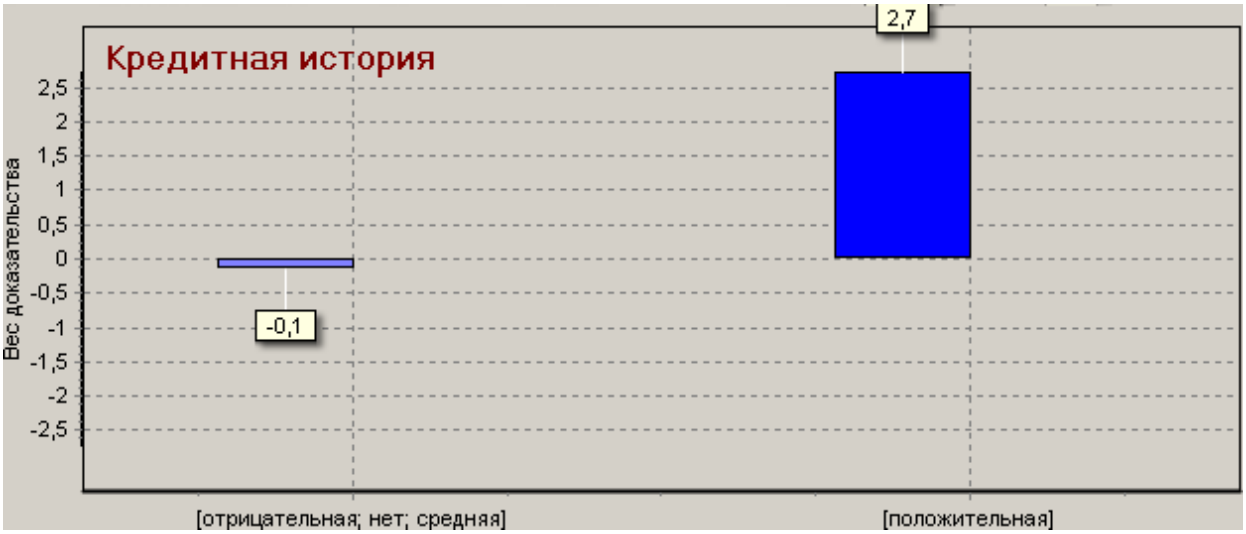
По аналогии проанализируем все оставшиеся столбцы, кроме незначимых. При необходимости вмешаемся в результаты автоматического разбиения и откорректируем их.

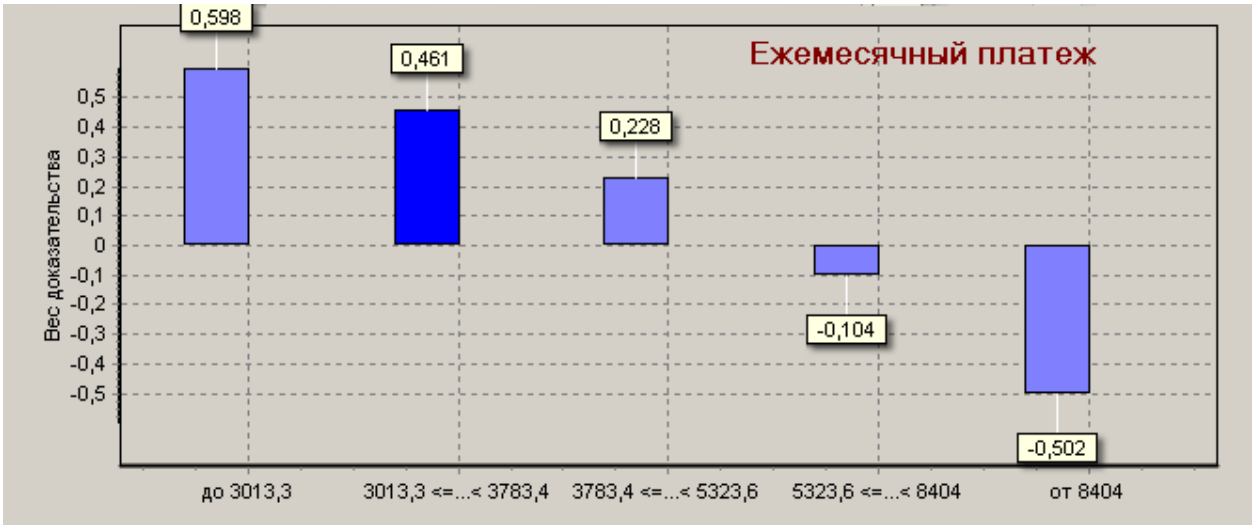
У столбцов «Карты», «Дети», «Кредитная история», «Ежемесячный платеж, «Личный доход» оставим конечные классы без изменений.



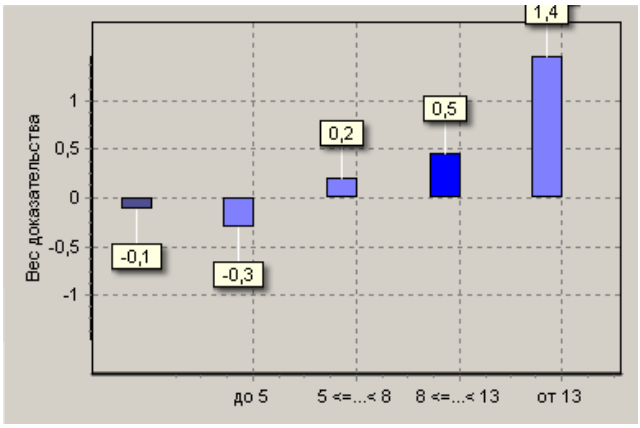

Дети
Вес доказательства
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
0,879
-0,294
[отсутствуют; Класс пропусков]
[1; 2 и более]


Личный доход
Вес доказательства
0,4
0,3
0,2
0,1
0
-0,1
-0,2
-0,3
-0,4
0,161
0,103
-0,032
-0,070
-0,407
до 17362
17362 <=...< 22456
22456 <=...< 30946
от 34342

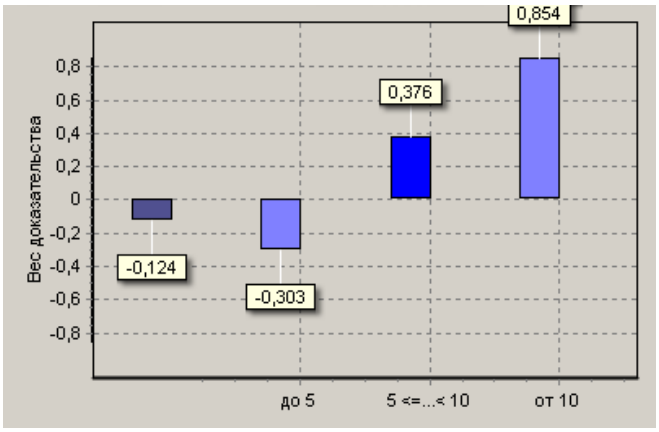

Кредитная история
Вес доказательства
2,5
2
1,5
1
0,5
0
-0,5
-1
-1,5
-2
-2,5
2,7
-0,1
[отрицательная; нет; средняя]
[положительная]



В конечные классы столбца «Стаж в отрасли» внесем следующие изменения.

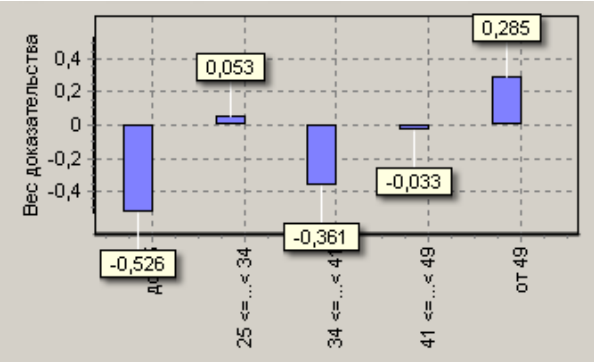


Автоматическое разбиение

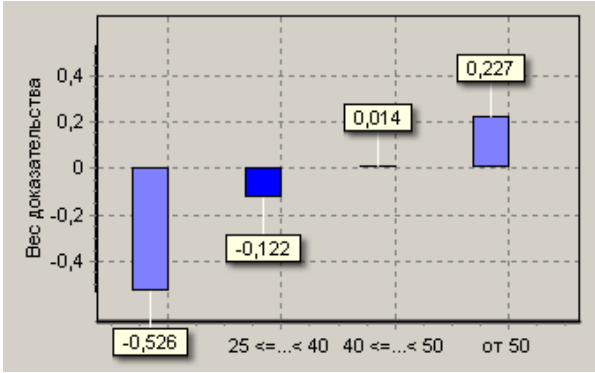


Ручная корректировка

И последний столбец, «Возраст».



Автоматическое разбиение



Ручная корректировка

В результате работы узла мы получили новый квантованный набор данных, который можно будет использовать для построения скоринговых моделей, а также список значимых столбцов.

Тема 5. ВВЕДЕНИЕ В ВИЗУАЛИЗАЦИЮ – 2 ч.

К каждому узлу сценария, который содержит структурированный набор данных, всегда предлагается несколько визуализаторов. *Мастер визуализации* (см. рис. 5.1) в интерактивном пошаговом режиме позволяет выбрать и настроить наиболее удобный способ представления данных. В зависимости от выбранного способа будут настраиваться различные параметры, а Мастер, соответственно, будет содержать различное число шагов. Первый шаг мастера визуализации будет одинаков для всех видов, поскольку на нем и производится выбор визуализатора.

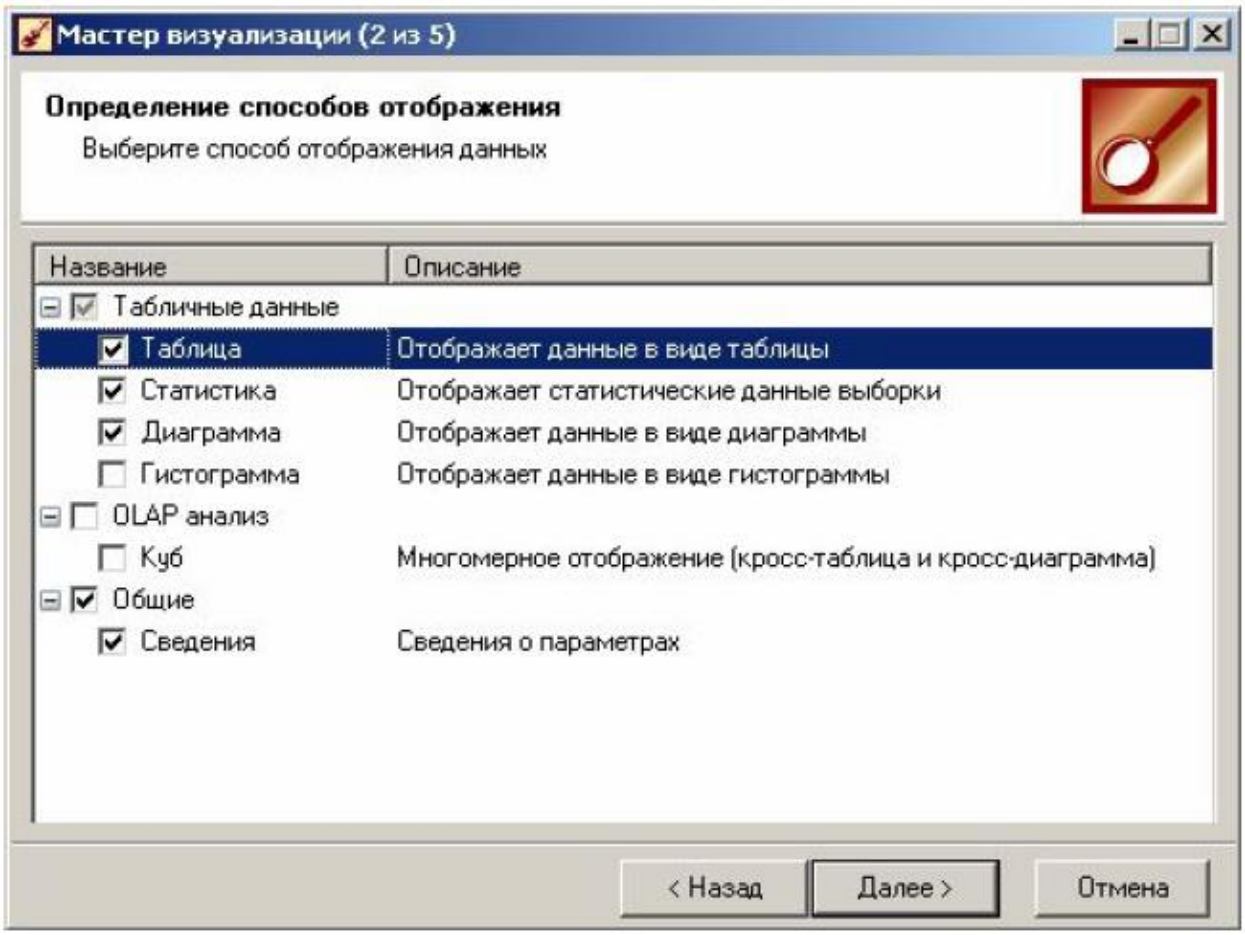


Рис. 5.1. Окно Мастера визуализации.

Для того, чтобы осуществить вызов Мастера визуализации необходимо выполнить одно из следующих действий:

- нажать на панели инструментов закладки **Сценарии** кнопку ;
- клавиша **F5**;
- контекстное меню **Мастер визуализации…**

Мастер визуализации запускается для выделенного узла сценария.

Кроме того, этот мастер всегда является продолжением мастера обработки, т.е. активизируется при создании (настройке) любого узла.

Желаемые способы отображения следует пометить флажками.

Одновременно может быть выбрано несколько визуализаторов, при этом каждый из них будет открыт в отдельном окне.

Необходимо обратить внимание, что, если на первом шаге мастера визуализации одновременно выбрано несколько способов отображения данных, то все соответствующие шаги будут последовательно включены в общую процедуру настройки. Например, если выбраны **Таблица** и **Диаграмма**, то в мастер визуализации будут последовательно включены отдельные шаги для настройки таблицы и диаграммы.

Для Deductor ьазовыми визуализаторами являются следующие:

- Таблица;
- Статистика;
- Сведения.

Визуализатор Таблица

В таблице каждое поле набора данных размещается в отдельном столбце. Столбцы озаглавлены метками полей, а если метка не была задана, то именами полей (см.рис. 5.2). Ширину и порядок столбцов можно менять при помощи мыши.

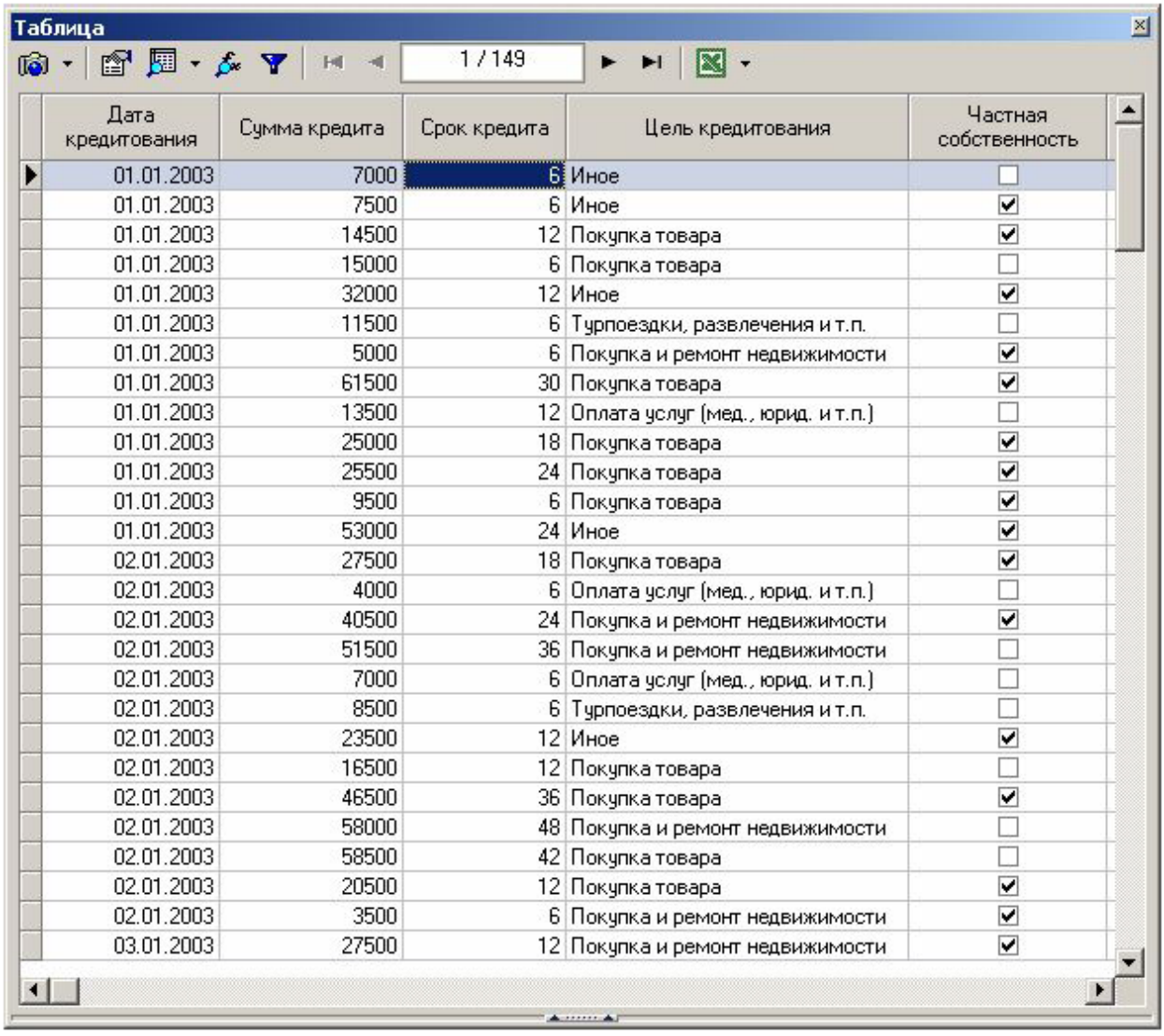


Рис. 5.2. Вариант представления визуализатора Таблица.

В таблице можно настроить объединение заголовков столбцов. Например, есть два заголовка **Продажи Сумма** и **Продажи Количество**. Если переименовать (например, с помощью обработчика **Настройка набора данных**) метку первого столбца в **Продажи|Сумма**, а второй – в **Продажи|Количество**, то получим объединение заголовка в шапке таблицы.

Символ «|» подсказывает визуализатору место в слове, где заканчивается общее название у двух заголовков.

Последовательное нажатие левой кнопкой мыши по заголовку активирует сортировку по данному столбцу в следующем порядке: сортировка по возрастанию – сортировка по убыванию – исходное состояние. Столбцы логического типа показываются в виде флажков.

В верхней части окна таблицы представлена панель инструментов, кнопки которой открывают доступ к функциям, представленным на рис. 5.3.

	Функция	Горячая клавиша	Описание
	управление конфигурациями	–	Сохранение и восстановление конфигураций отображения таблицы
	настройка полей	<F11>	Позволяет настраивать видимость полей, отображаемых в таблице, а также задавать их формат и способ выравнивания
	способ отображения	<Ctrl+F12>	Переключение между отображением данных в виде таблицы или в виде формы
	статистика	–	Позволяет посмотреть статистику по текущим данным таблицы. Аналогично визуализатору *Статистика*, но открывается внизу таблицы, а не в отдельном окне
	фильтрация	<Ctrl+D>	Позволяет выполнять фильтрацию записей в таблице по заданным условиям
	первая запись	<Ctrl+PgUp>	Переход на первую запись набора данных
	предыдущая запись	<PgUp>	Переход на предыдущую запись набора данных
28 / 149	*номер строки*	–	Индикатор текущей записи
	следующая запись	<PgDn>	Переход на следующую запись набора данных
	последняя запись	<Ctrl+PgDn>	Переход на последнюю запись набора данных
	экспорт	–	Вызывается окно выбора файла для экспорта данных из таблицы в один из доступных текстовых форматов: *MS Excel*, *RTF*, *HTML*, *TXT*, *CSV*. По умолчанию предлагается экспорт в *MS Excel*. В версии Academic доступны не все форматы экспорта.

Рис. 5.3. Панель инструментов окна Таблица.

Однажды настроенный вид таблицы (к примеру, с различными фильтрами, форматами и видимостью столбцов и т.п.) можно сохранить, чтобы впоследствии быстро вернуться к нему.

Для этого в раскрывающемся по кнопке списке нужно выбрать пункт **Сохранить конфигурацию…** и далее ввести ее название. Загрузить новую конфигурацию, можно, выбрав ее из списка конфигураций (см. рис. 5.4).

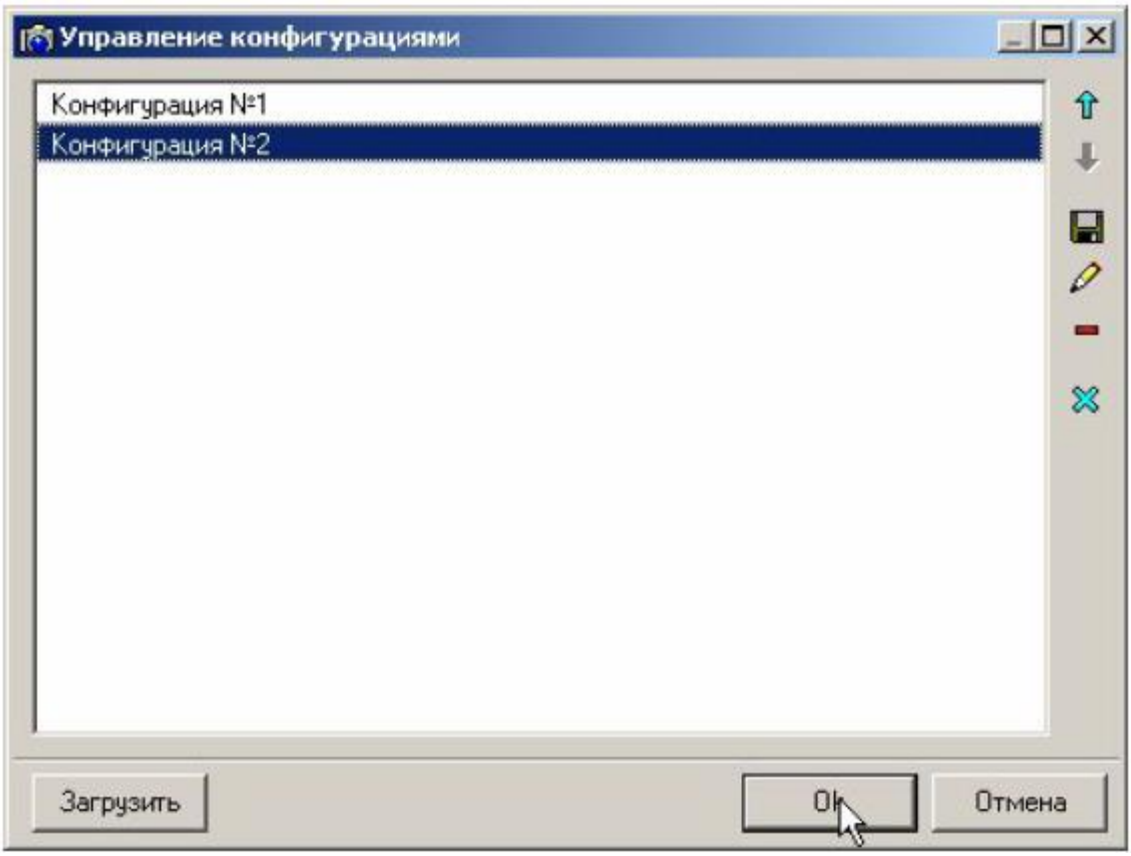


Рис. 5.4. Панель управления конфигурацией.

При вызове настройки полей появляется соответствующее диалоговое окно (рис. 5.5). В нем можно скрыть или сделать видимыми различные поля таблицы, определить способ выравнивания содержимого, ширину поля, а также задать формат отображения числовых данных и дат.

Кнопка переключает способ отображения набора данных, который может быть не только табличным, но и в виде формы. Это удобно, когда набор данных содержит большое количество столбцов.

Кнопка открывает окно настройки условий фильтрации на набор данных. При включенном фильтре цвет кнопки меняется на , а цвет заголовков столбцов, которые участвуют в фильтре, изменяется на красный (см. рис. 5.6).

Кнопка открывает визуализатор **Статистика**, но не в отдельной вкладке, а в нижней части визуализатора **Таблица**.

Визуализатор Статистика предназначен для отображения основных статистических характеристик набора данных конкретного узла (см. рис. 5.7)).

Статистические характеристики отображаются в таблице по каждому полю выборки. В верхней части окна статистики отображается общее количество записей в наборе данных. Панель инструментов окна статистики позволяет управлять отображением статистических характеристик (среднее, минимум, максимум и т.п.) с помощью группы кнопок  .

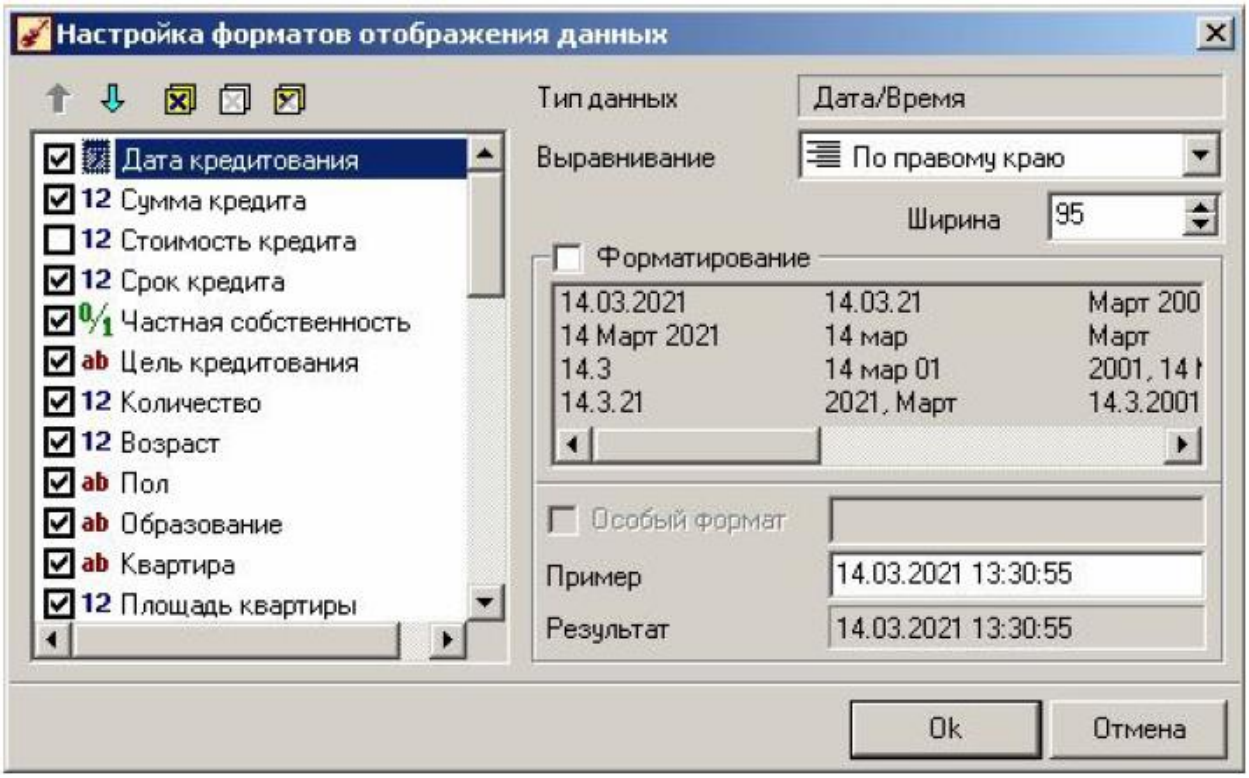


Рис. 5.5. Окно настройки форматов отображения данных.

Дата (Год + Месяц)	Количество
2002-M01	355000
2002-M02	340000
2002-M03	405000
2002-M04	452000
2002-M05	464000
2002-M06	437000

Рис. 5.6. Изменение цвета заголовков столбцов.

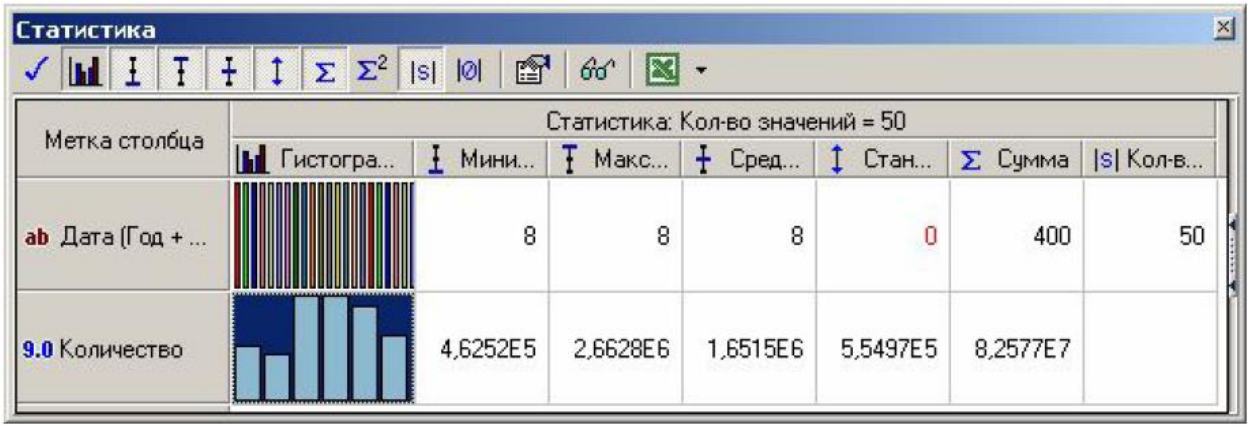


Рис. 5.7. Вариант представления визуализатора Статистика.

Для полей дискретного типа, кроме прочих, всегда рассчитываются следующие статистические показатели:

- количество уникальных значений,
- количество пустых значений.

Просмотреть список уникальных значений можно одним из двух способов:

- двойной щелчок по ячейке **Количество уникальных значений** или по ячейке **Гистограмма**;
- кнопка **Обзор статистики**.

Для поля непрерывного типа в обзоре статистики строится гистограмма распределения частот, она же в уменьшенном виде всегда показывается в соответствующем столбце (см. рис. 5.8).

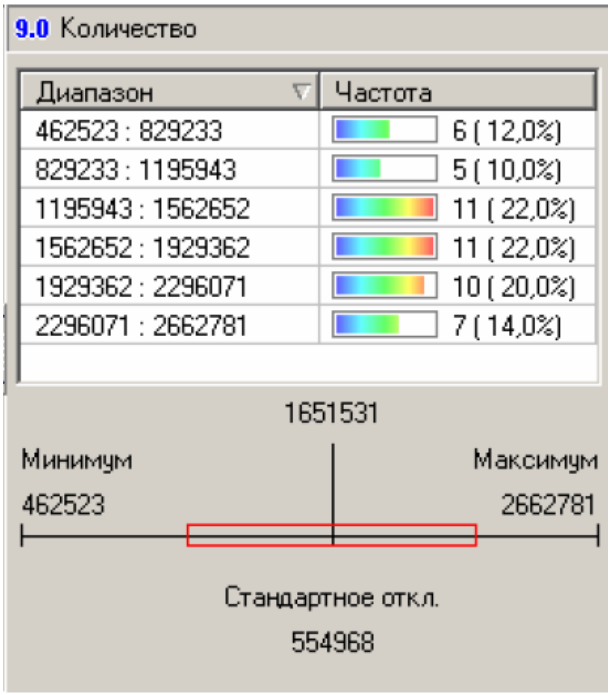


Рис. 5.8. Уменьшенный вид гистограммы распределения частот.

Визуализатор Сведения

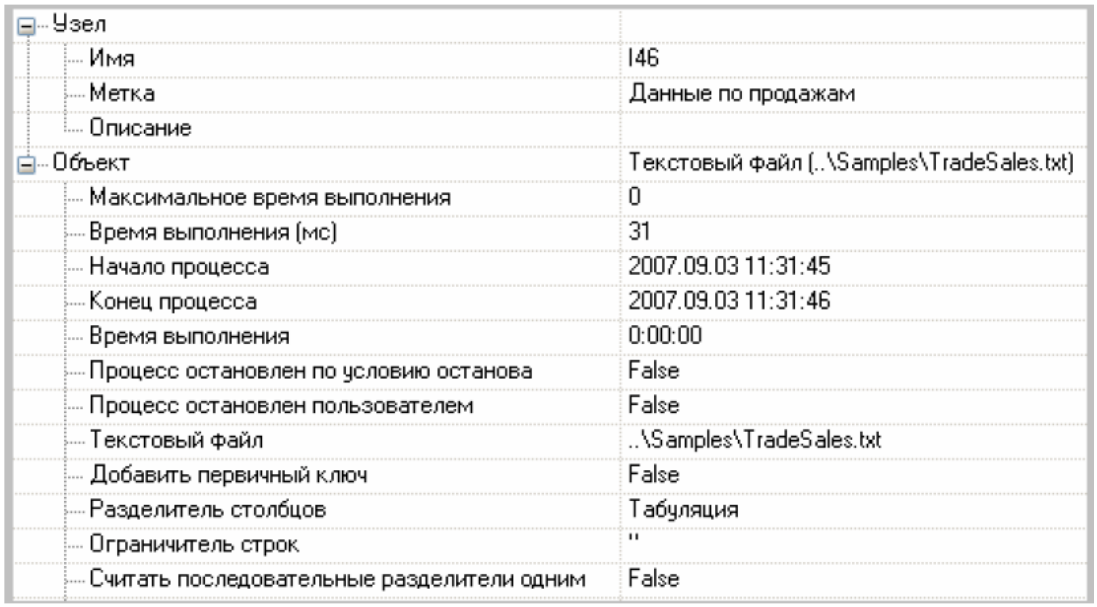
Визуализатор **Сведения** позволяет просмотреть все параметры, с которыми был выполнен тот или иной процесс преобразования данных, в результате

которого была сформирована новая выборка: импорт, обработка одним из методов или экспорт. Такими параметрами являются время и длительность выполняемого процесса, условия остановки, наличие первичного ключа, ограничители столбцов, разделители целой и дробной частей чисел, элементов даты и т.д.

Предусмотрено два вида представления описания: в виде дерева (см. рис. 5.9) и текстовый. По умолчанию устанавливается вид дерева.

Визуализатор в основном предназначен для оперативного анализа текущих настроек узлов и для поиска возможных ошибок.

Визуализатор **Сведения** является единственно доступным для узлов экспорта.



Узел	
Имя	I46
Метка	Данные по продажам
Описание	
Объект	Текстовый файл (..\Samples\TradeSales.txt)
Максимальное время выполнения	0
Время выполнения (мс)	31
Начало процесса	2007.09.03 11:31:45
Конец процесса	2007.09.03 11:31:46
Время выполнения	0:00:00
Процесс остановлен по условию останова	False
Процесс остановлен пользователем	False
Текстовый файл	..\Samples\TradeSales.txt
Добавить первичный ключ	False
Разделитель столбцов	Табуляция
Ограничитель строк	''
Считать последовательные разделители одним	False

Рис. 5.9. Вид визуализатора **Статистика** в виде дерева.

Многомерная диаграмма

Исходные данные

Рассмотрим построение многомерной диаграммы на примере данных из файла "fuel.txt". Он содержит таблицу с информацией о стоимости топлива в регионах РФ. Многомерная диаграмма позволяет увидеть различия в цене покупаемого топлива по регионам.

Выполнение настройки

Осуществим построение многомерной диаграммы в специально созданном для этого узле "Диаграмма", на основе обработчика "Настройка набора данных" (см. рис. 5.10).

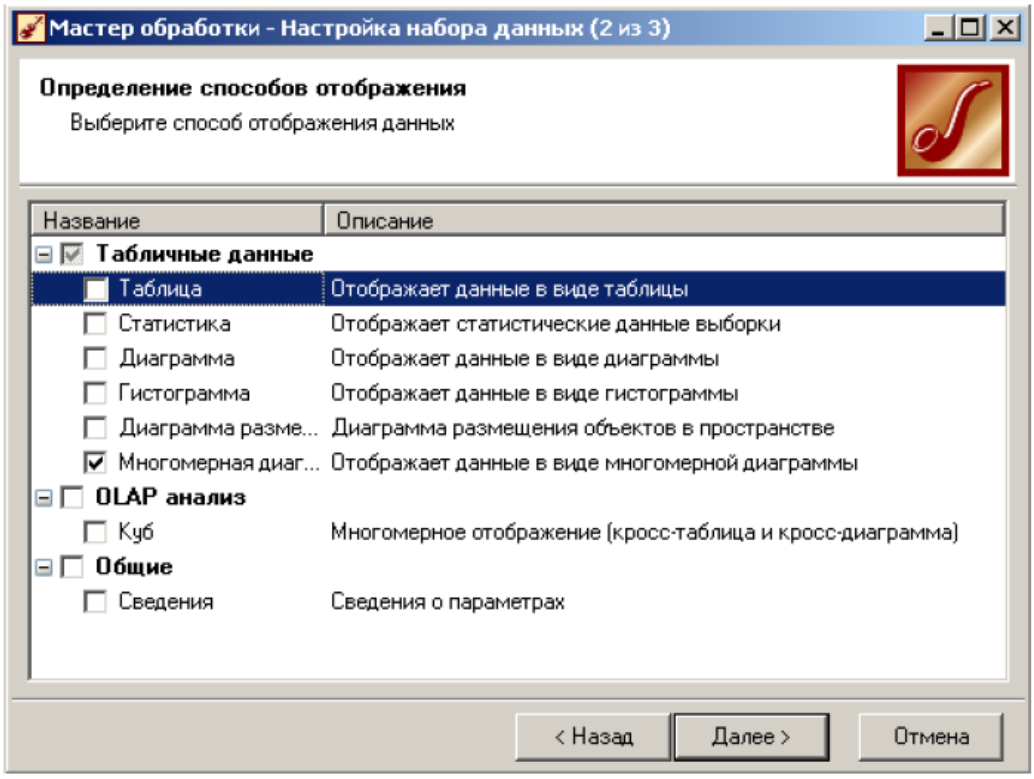


Рис. 5.10. Выполнение настройки диаграммы.

Для этого выберем в настройках отображения данных рассматриваемого узла визуализатор "Многомерная диаграмма".

На следующем шаге определим оси координат. Выберем свойства на основе которых изменяется цена в нашем случае это название регионов и наименование топлива. Обозначим координату x – "Федеральные округа", а y – " Топливо", z – "Цена" (см. рис. 5.11).

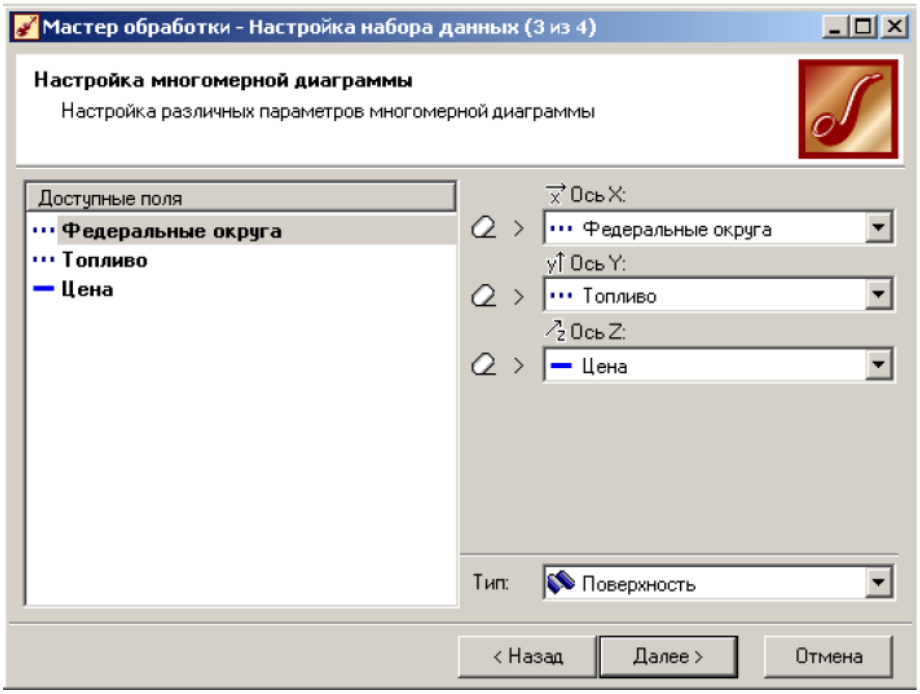


Рис. 5.11. Определение осей координат.

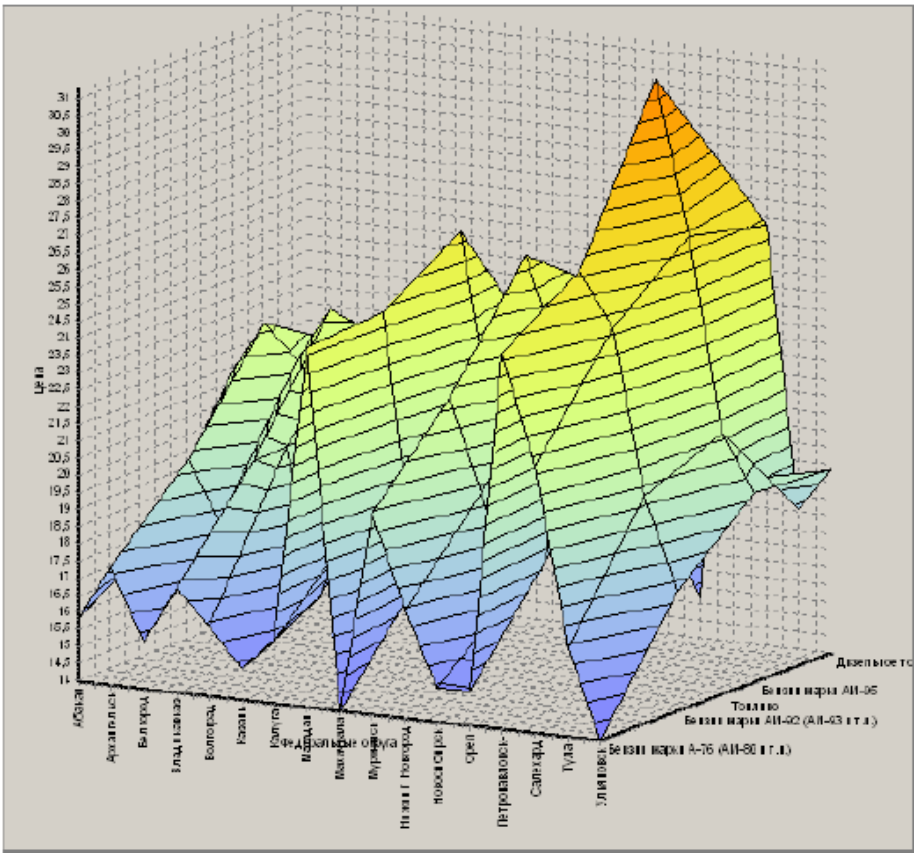


Рис. 5.12. Определение осей координат.

Результат

Выберем тип изображения диаграммы как "Поверхность", на котором будет показана объемная зависимость цены рассматриваемого топлива по регионам.

На диаграмме показаны перепады стоимости топлива в различных регионах не только графически, но и цветом (см. рис. 5.12). При необходимости можно просмотреть детализацию выбранных на графике точек, выбрав соответствующую настройку.

Самостоятельная работа:

1. Откройте проект Deductor, созданный на прошлом занятии. Настройте следующие визуализаторы к любому узлу импорта: **Таблица**, **Статистика**. Перейдите в режим формы и обратно. Имеются ли пропуски в записях?

2. В визуализаторе **Таблица** настройте, чтобы при отображении к значениям в Поле3 добавлялось слово « кг.». Сохраните конфигурацию визуализатора под названием К1.

3. Сделайте первые три столбца невидимыми. Сохраните конфигурацию визуализатора под названием К2.

4. Вернитесь к конфигурации К1.

5. В визуализаторе **Таблица** установите фильтр Поле6 = не пустой. Удалите фильтр.

6. Постройте многомерную диаграмму на примере данных из файла "fuel. txt".

Вопросы для проверки:

1. Какие характеристики набора данных показывает визуализатор **Статистика**?
2. Что означает красный заголовок столбца в визуализаторе **Таблица**?
3. Как обнаружить, имеются ли в столбце пропущенные значения?
4. Для чего предназначен визуализатор **Сведения**?
5. Как скрыть столбец в визуализаторе **Таблица**?
6. К существующему в сценарии узлу импорта необходимо добавить еще один визуализатор. Что предпринять?

Занятие 2. Создание OLAP-отчетов в Deductor Studio.

Для построения аналитической отчетности в программе Deductor предназначена вкладка **Отчеты**, которую можно вызвать последовательным выполенением команд: Вид, Отчеты, или с помощью кнопки , после нажатия на которую в рабочей части экрана появится панель **Отчеты**.

Отчеты строятся в виде древовидного иерархического списка (рис. 5.1), каждым узлом которого является отдельный отчет или папка, содержащая несколько отчетов. Каждый узел дерева отчетности связан со своим узлом в дереве сценария. Для каждого отчета настраивается свой способ отображения (таблица, гистограмма, кросс-таблица, кросс-диаграмма и т.п.). Это удобно, так как несколько отчетов могут быть связаны с одним узлом дерева сценария.

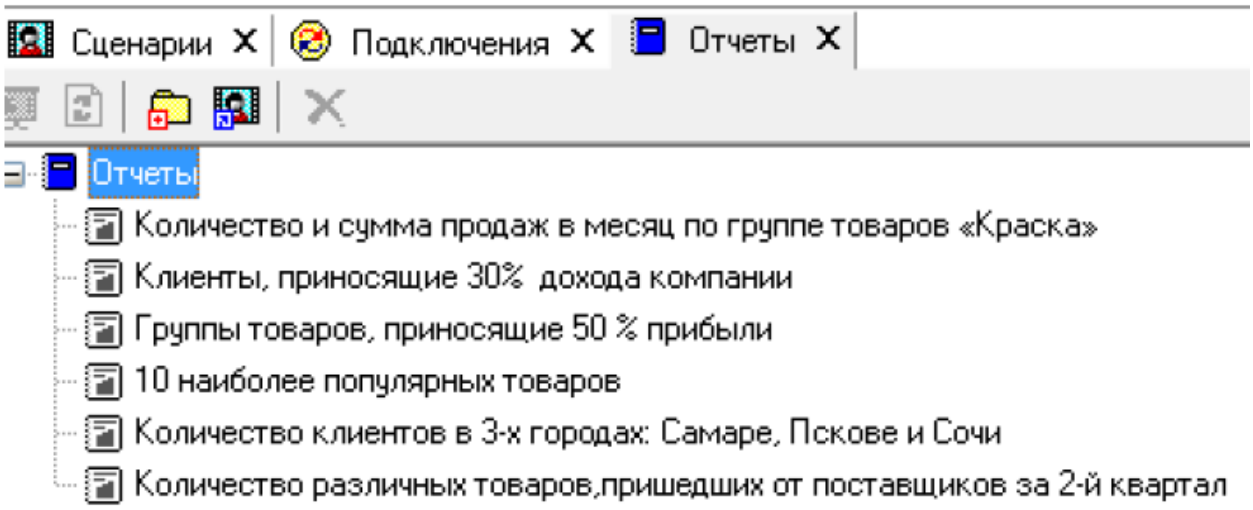


Рис. 5.1. Панель отчетов.

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.**

Следует отметить, что операция добавления нового отчета доступна, только если выделена папка или корневой пункт **Отчеты** списка отчетов. Если выделить узел, содержащий отдельный отчет, команда создания нового отчета будет недоступна.

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 полюбому узлу списка. Чтобы поместить отчет в папку, нужно перед вызовом команды **Добавить узел** выделить эту папку.

Таким образом, было построено 6 OLAP- отчетов на основе подготовленных (преобразованных) данных:

Для построения первого OLAP-отчета необходимо привести данные к нужному виду. Для этого был использован обработчик «Дата/Время» для агрегации данных в разрезе месяца. С помощью обработчика «Группировка» данные были сгруппированны по полю «Дата» и «Группы товаров», а обработчиком «Фильтр» были выбраны необходимые данные из группы товаров «Краска» (см. рис. 5.2).

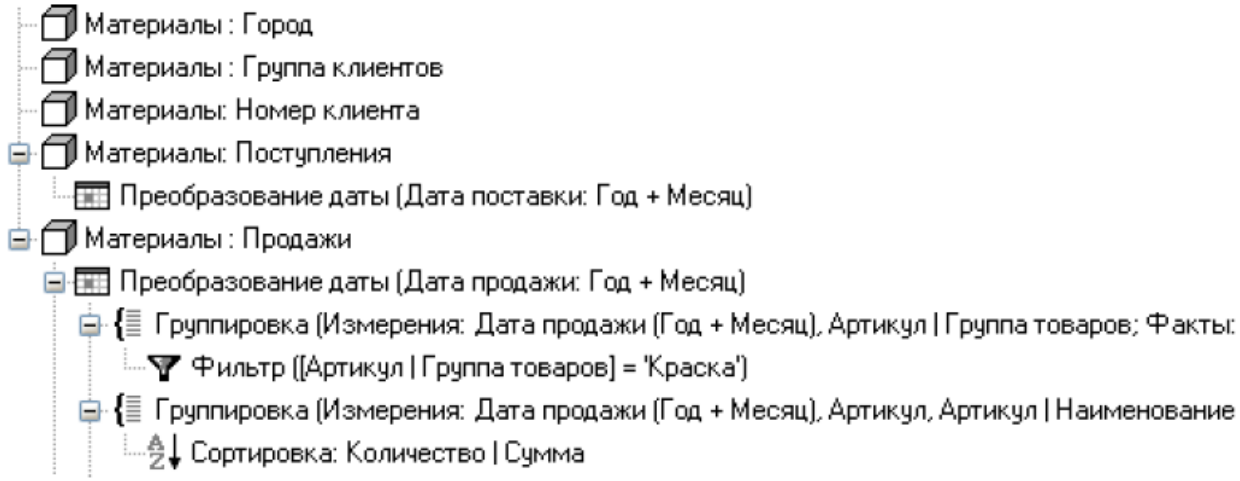


Рис. 5.2. Преобразование данных для OLAP-анализа.

Первый OLAP-отчет построен для определения количества продаж и суммы продаж помесячно по группе товаров «Краска». Он построен на основании измерений *Дата продажи* и *Группа товаров*, а также фактов «Количество» и «Сумма с учетом скидки»: рис. 5.3.

Таблица × Куб ×

	Артикул | Группа товаров			
	Краска		Итого:	
Дата продажи (Год + Месяц)	Σ Количество	Σ Сумма с учетом скидки	Σ Количество	Σ Сумма с учетом скидки
01.03.2021	23 083,00	21 361 356,81	23 083,00	21 361 356,81
01.04.2021	15 055,00	12 670 427,67	15 055,00	12 670 427,67
01.05.2021	19 107,00	17 455 930,21	19 107,00	17 455 930,21
01.06.2021	17 976,00	16 135 497,39	17 976,00	16 135 497,39
01.07.2021	28 103,00	24 842 715,88	28 103,00	24 842 715,88
01.08.2021	33 436,00	27 191 120,73	33 436,00	27 191 120,73
01.09.2021	35 831,00	32 628 371,47	35 831,00	32 628 371,47
01.10.2021	31 566,00	27 404 441,91	31 566,00	27 404 441,91
01.11.2021	28 690,00	25 592 482,67	28 690,00	25 592 482,67
01.12.2021	258,00	195 922,38	258,00	195 922,38
Итого:	233 105,00	205 478 267,10	233 105,00	205 478 267,10

Рис. 5.3. OLAP-отчет «Количество и сумма продаж в месяц по группе товаров «Краска».

По данному отчету можно сделать вывод, что в сентябре было продано максимальное количество товаров группы «Краски» на максимальную сумму. Также данный отчет можно представить в виде диаграммы, на которой наглядно

представлено, когда и какое максимальное количество товаров было продано за год: (см. рис. 5.4).

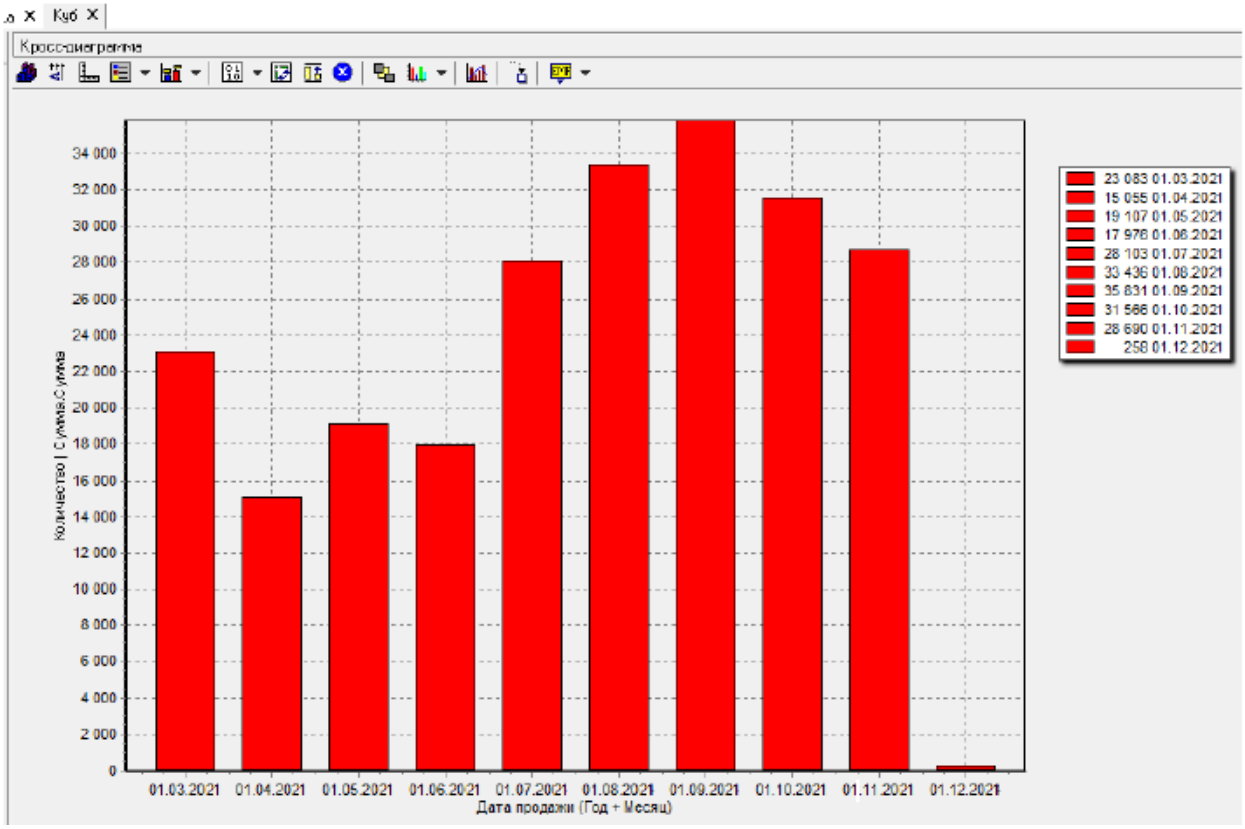


Рис. 5.4. OLAP-отчет в виде диаграммы «Количество продаж в месяц по группе товаров «Краска».

Второй OLAP-отчет построен по измерениям «Номер клиента» и «Группа товаров», а также по факту «Сумма с учетом скидки», в отчете отображены суммы проданных товаров каждым клиентом: (см. рис. 5.5).

Таблица | Куб

Номер клиента | Группа клиента				
Номер клиента	VIP клиент	Клиент	Постоянный клиент	**Итого:**
00000003		7 654 066,02		7 654 066,02
00000004		22 829,94		22 829,94
00000010			4 540 088,82	4 540 088,82
00000011		130 504,28		130 504,28
00000014		27 234,51		27 234,51
00000015			9 080 035,66	9 080 035,66
00000016		4 183 675,25		4 183 675,25
00000017		13 671,30		13 671,30
00000019	41 078 054,30			41 078 054,30
00000020	53 799 967,08			53 799 967,08
00000031		185 644,20		185 644,20
00000032		238 598,13		238 598,13

Рис. 5.5. OLAP-отчет «Суммы проданных товаров каждым клиентом».

Чтобы увидеть не все суммы проданных товаров, а только те группы товаров, клиенты которых приносят в общем 30% дохода компании, нужно в самом визуализаторе « OLAP- куб» воспользоваться фильтром. Настройки данного фильтра представлены на рис. 5.6. Внешний вид отчета показан на рис. 5.7.

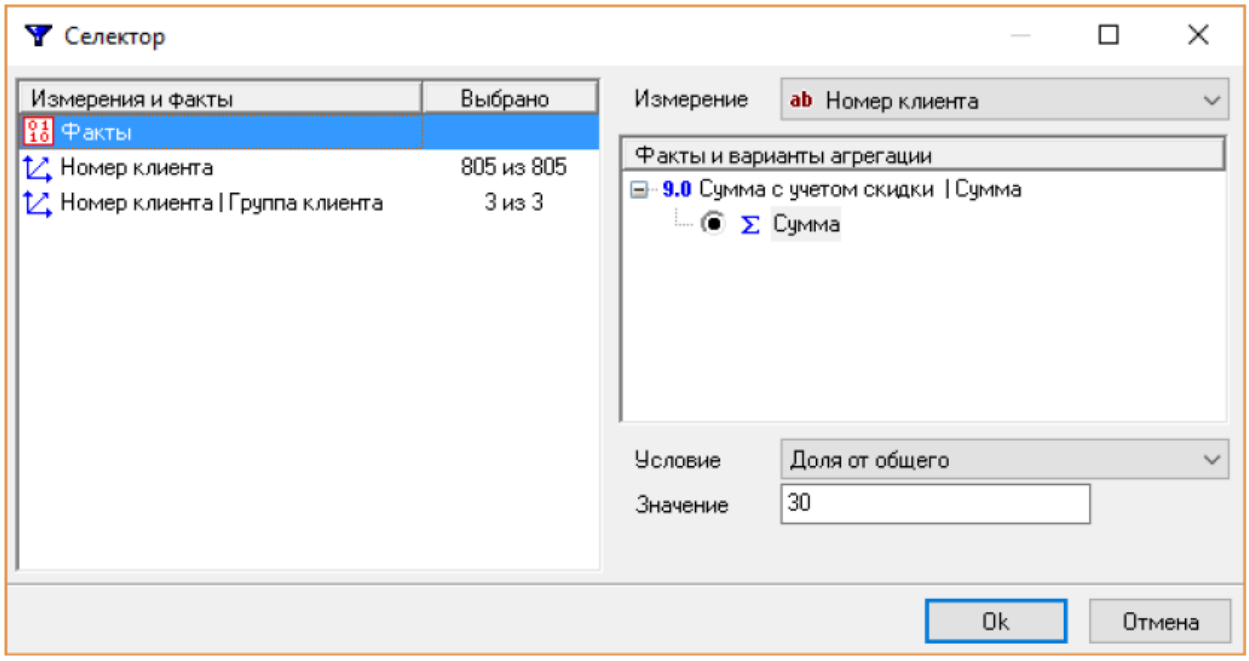


Рис. 5.6. Настройка фильтра для OLAP-отчета «Клиенты, приносящие 30% дохода компании».

Таблица ✕ Куб ✕

Номер клиента | Группа клиента				
Номер клиента	VIP клиент	Клиент	Постоянный клиент	**Итого:**
00000019	41 078 054,30			41 078 054,30
00000020	53 799 967,08			53 799 967,08
00000069			30 060 766,02	30 060 766,02
00001315	32 035 539,67			32 035 539,67
00003495		28 879 423,04		28 879 423,04
00011533			47 771 820,10	47 771 820,10
99999999	200 049 195,71			200 049 195,71
EL001176		29 710 072,18		29 710 072,18
Итого:	326 962 756,76	58 589 495,22	77 832 586,12	463 384 838,10

Рис. 5.7. OLAP-отчет «Клиенты, приносящие 30% дохода компании».

Отчет показывает, что представленные клиенты приносят 30 % дохода компании, такие клиенты очень важны для компании.

Следующий OLAP- отчет построен по измерениям « Артикул» и « Наименование товара», а также по факту « Количество», т.е. в отчете будет отображена статистика проданных товаров. Для начала преобразуем данные: с помощью обработчика « Группировка» сгруппируем данные по артикулу и наименованию товара (количество будет фактом). Затем отсортируем количество по убыванию. Сформируем OLAP- куб, используя настройку данных. Воспользовавшись фильтром, выберем первые 10 элементов (см. рис. 5.8).

Таблица ✕ Куб ✕

− + Артикул | Наименование товара ▼ Артикул	Σ Количество | Сумма
⊞ Болт 10x20 цинк, шестигранная головка, 8 штук, 8 категория, уп	41 926,00
⊞ Болт 8x100 цинк, шестигранная головка, 2 штуки, 4 категория, уп	46 545,00
⊞ Болт 8x25 цинк, шестигранная головка, 8 штук, 6 категория, уп	43 033,00
⊞ Болт 8x40 цинк, шестигранная головка, 8 штук, 8 категория, уп	41 351,00
⊞ Болт 8x50 цинк, шестигранная головка, 6 штук, 7 категория, уп	42 034,00
⊞ Болт 8x70 цинк, шестигранная головка, 6 штук, 9 категория, уп	44 005,00
⊞ Болт 8x90 цинк, шестигранная головка, 4 штук, 4 категория, уп	44 023,00
⊞ Дюбель с шурупом забивной WKRET-MET 6x60 грибок SMK, 200 штук, уп	43 798,00
⊞ Дюбель с шурупом забивной WKRET-MET 6x80 грибок SMK, 100 штук, уп	43 833,00
⊞ Шуруп универсальный 4.5x40, 50 штук, 9 категория, уп	44 486,00
Итого:	435 034,00

Рис. 5.8. OLAP-отчет «Десять наиболее популярных товаров».

Отчет, представленный на рис. 5.9, показывает десять наиболее популярных товаров, т.е. наиболее продаваемыми.

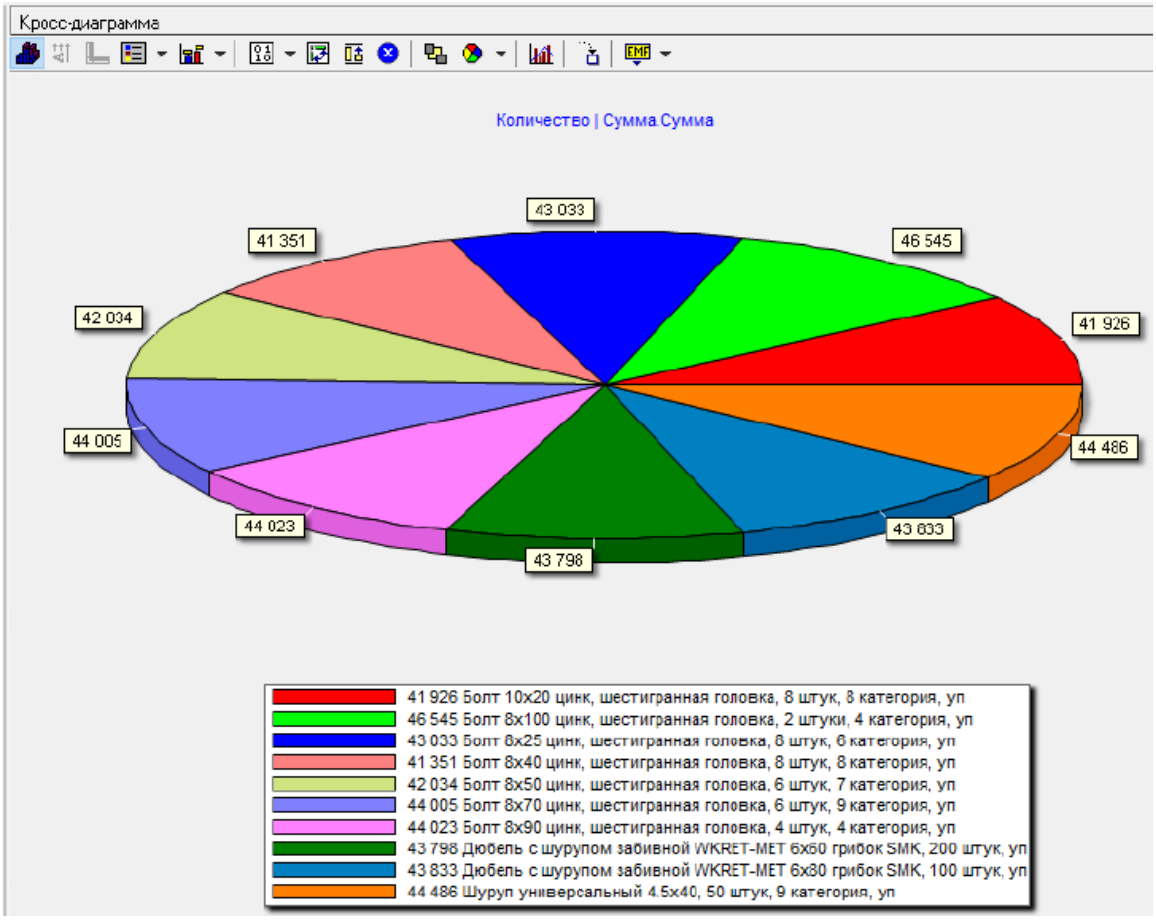


Рис. 5.9. Кросс-диаграмма OLAP-отчета «Десять наиболее популярных товаров».

Следующий OLAP-отчет построен по измерениям «Группа клиента» и « Город», а также по факту «Номер клиента», т.е. в отчете будет отображена статистика по количеству клиентов в каждом городе с детализацией по группе клиента. При настройке фактов указываем, что факт «Номер клиента» будет иметь агрегацию «Количество уникальных», таким образом будет считаться правильное количество клиентов (без повторений): (см. рис. 5.10).

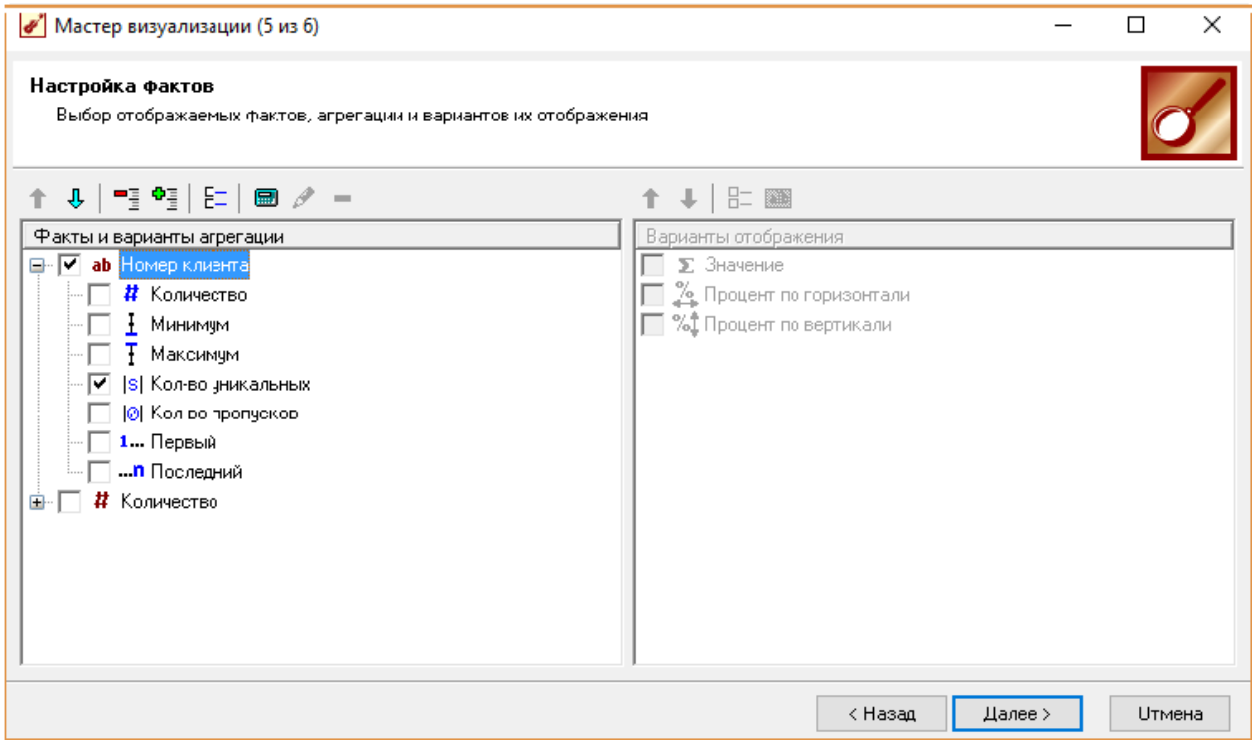
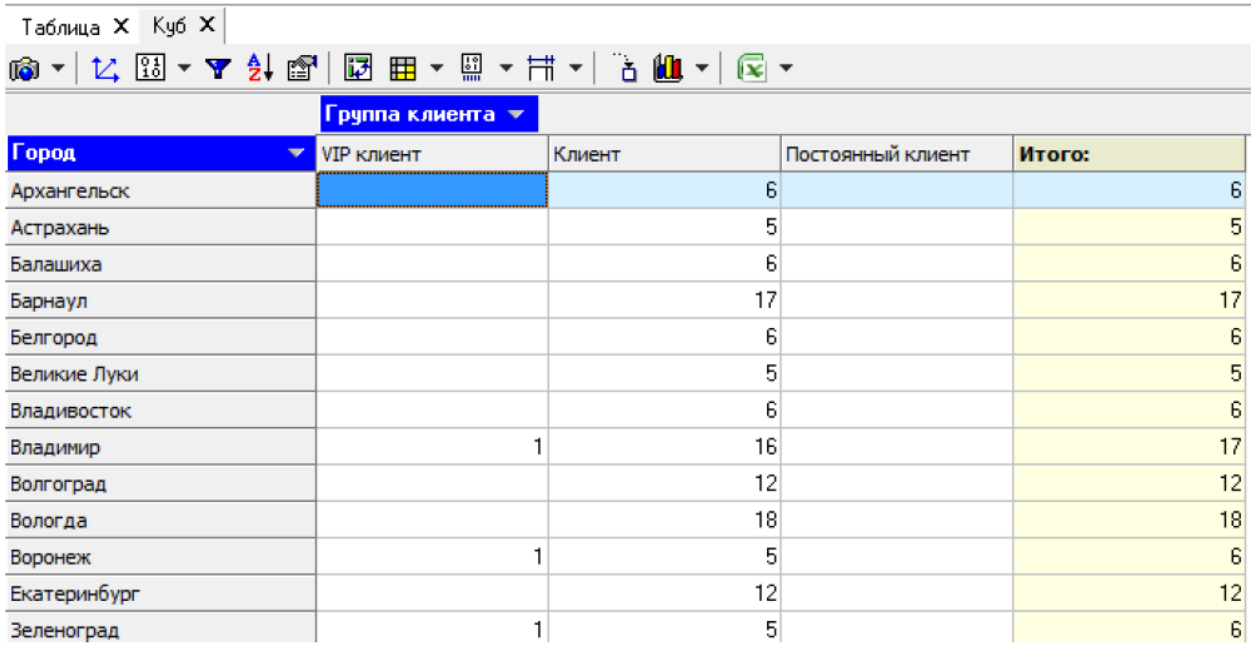


Рис. 5.10. Настройка OLAP-отчета «Количество клиентов в каждом городе».

Внешний вид OLAP-отчета «Количество клиентов в каждом городе» показан на рис. 5.11.



Город	VIP клиент	Клиент	Постоянный клиент	Итого:
Архангельск		6		6
Астрахань		5		5
Балашиха		6		6
Барнаул		17		17
Белгород		6		6
Великие Луки		5		5
Владивосток		6		6
Владимир	1	16		17
Волгоград		12		12
Вологда		18		18
Воронеж	1	5		6
Екатеринбург		12		12
Зеленоград	1	5		6

Рис. 5.11. OLAP-отчет «Количество клиентов в каждом городе».

Можно настроить фильтр по определенным городам и посмотреть, сколько в выбранных городах клиентов. Например, посмотрим, сколько покупателей есть в Пскове, Самаре и Сочи (см. рис. 5.12). Внешний вид OLAP-отчета «Количество клиентов в городах» показан на рис. 5.13.

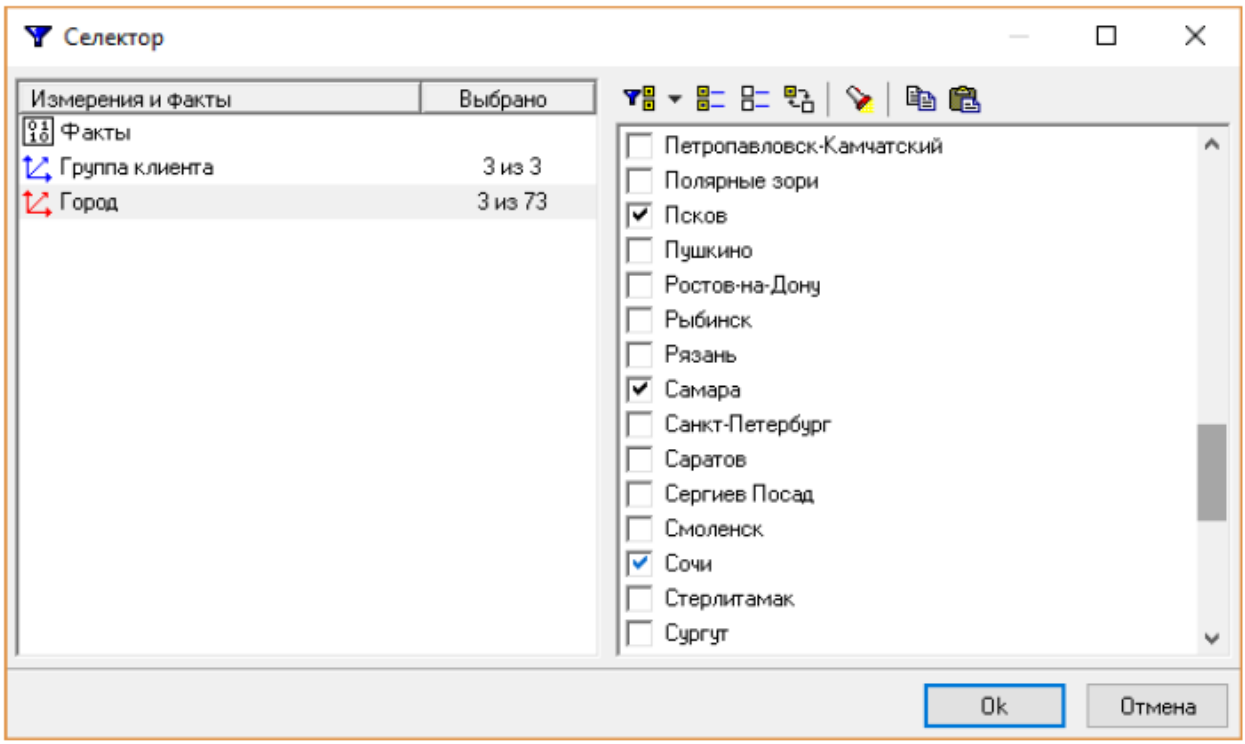


Рис. 5.12. Выбор городов.

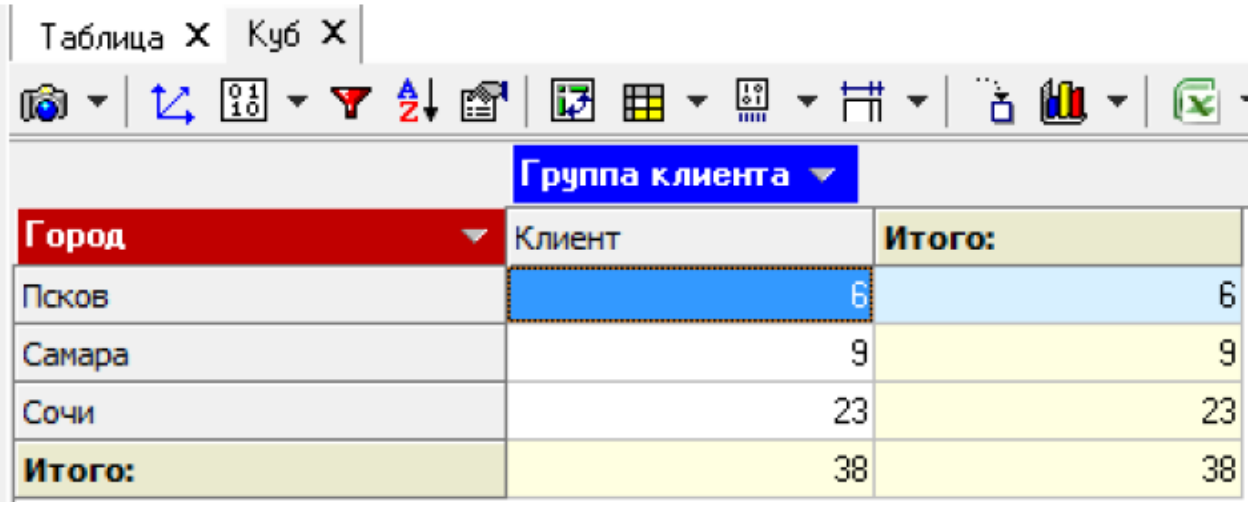


Рис. 5.13. OLAP-отчет «Количество клиентов в городах»

По данному отчету можно сделать вывод, что в Пскове есть 6 клиентов, в Самаре – 9, а в Сочи – 23, причем все они относятся к группе клиента «Клиент».

Последний OLAP-отчет построен по измерениям «Номер поставщика» и « Дата поставки», а также по факту « Наименование товара». Дата поставки преобразована с помощью обработки « Дата/ время», т.е. в отчете будет отображена статистика по номеру поставщика и дате поставки помесячно, а на пересечении столбцов и строк будет указано количество уникальных товаров. Построим OLAP-отчет за 2-й квартал по поставщикам (см.рис. 5.14).

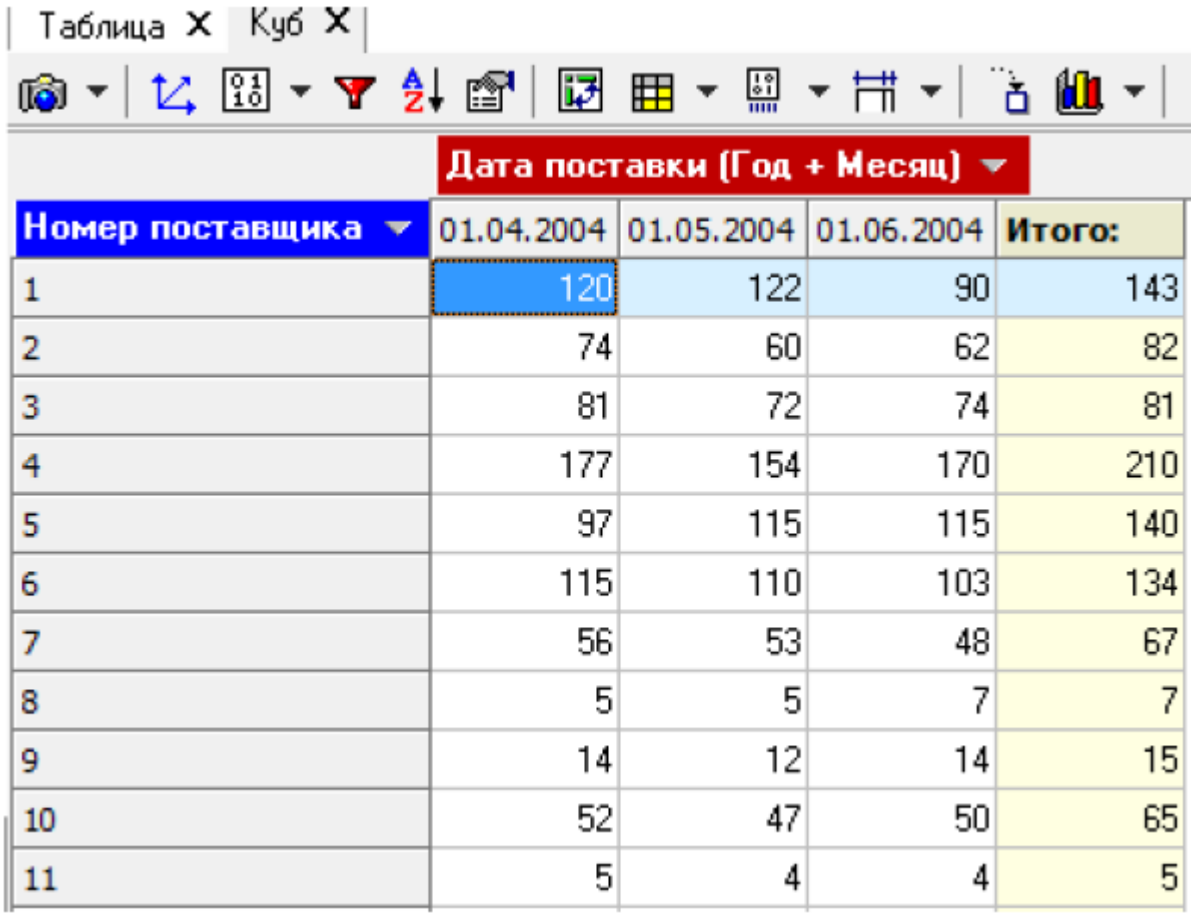


Рис. 5.14. OLAP-отчет «Отчет за второй квартал по уникальным товарам».

Данный отчет показывает, сколько различных товаров 28 поставщиков продали компании за квартал.

Самостоятельная работа:

Задания для самостоятельной работы

Задание 1. Создание хранилища данных с помошью Deductor Studio.

1. Для создания нового хранилища данных или подключения к существующему нужно перейти на закладку **Подключения** и запустить **Мастер подключений.**

2. Пройти первые два шага, выбрав тип приемника (источника) Deductor Warehouse и тип базы данных Firebird.

3. На третьем шаге нужно задать параметры базы данных, в которой будет создана физическая и логическая структура хранилища данных:

- база данных – **farma.gdb**;
- логин – **sysdba**, пароль – **masterkey**;
- установить флажок **сохранять пароль**.

4. На следующем шаге выбирается версия для работы с хранилищем данных Deductor Warehouse.

5. На пятом шаге при нажатии кнопки **Создать файл базы данных с необходимой структурой метаданных** по указанному ранее пути будет создан

файл **farma.gdb** (появится сообщение об успешном создании). Это и есть пустое хранилище данных, готовое к работе.

6. На последних двух шагах осталось выбрать визуализатор для подключения (здесь это Сведения и Метаданные) и задать для нового хранилища имя **Farma**, метку **Фармация** и описание **«Хранилище данных с информацией о продажах в аптечной сети компании Фарма-Х»**.

6. После нажатия на кнопку **Готово** на дереве узлов подключений появится метка хранилища.

7. Для проверки доступа к новому хранилищу воспользуйтесь кнопкой  . Если спустя некоторое время появится сообщение «Тестирование соединения прошло успешно», то хранилище готово к работе. Иначе нужно внести изменения в параметры подключения хранилища, используя кнопку  .

8. Сохраните настройки подключений, нажав на кнопку  .

9. Для проектирования структуры хранилища данных вызвать **Редактор метаданных** кнопкой на  вкладке **Подключения**.

10. В открывшемся окне редактора, встав на узле **Измерения**, при помощи кнопки **Добавить** добавьте в метаданные первое измерение *Код группы* со следующими параметрами:

- Имя – GR_ID;
- Метка – *Группа.Код*;
- Тип данных – целый.

11. Проделать аналогичные действия для создания всех остальных измерений, взяв параметры из таблицы:

Измерение	**Имя**	**Метка**	**Тип данных**
Код группы	GR_ID	Группа.Код	целый
Код товара	TV_ID	Товар.Код	целый
Код отдела	PART_ID	Отдел.Код	целый
Дата	S_DATE	Дата	дата/время
Час покупки	S_HOUR	Час	целый

12. К каждому измерению, кроме *ДАТА* и *ЧАС*, нужно добавить по текстовому атрибуту. Для измерения *Группа.Код* это будет *Группа.Наименование* GR_NAME, для измерения *Товар.Код* это будет *Товар.Наименование* TV_NAME, для измерения *Отдел.Код* это будет *Отдел.Наименование* PART_NAME.

13. Путем простого добавления (ссылка на измерение отображается иконкой ) установить ссылку измерения *Товар.Код* на измерение *Группа.Код*.

14. В окне редактора, встав на узле **Процессы**, при помощи кнопки **Добавить** введите в метаданные следующие параметры: Имя – SALES; Метка – *Продажи*.

15. Добавьте в процесс ссылки на 4 существующих измерения: *Дата, Отдел.Код, Товар.Код,Час* и добавьте ссылки на два факта: целочисленный – *Количество* F_COUNT и вещественный – *Сумма* F_SUM.

16. Проектирование структуры и метаданных хранилища данных закончено, можно закрыть окно редактора.

Задание 2. Создание OLAP-куба

1. Построить куб по трем измерениям (отдел, месяц года, товарная группа), в ячейках которого отображается сумма и объем (количество проданных единиц продукции) продаж за все периоды, имеющиеся в хранилище данных.

2. То же, что в п.1, но за последние три месяца от имеющихся данных.

3. Сформировать многомерный отчет и график загруженности торговых точек по дням месяца.

4. Сформировать многомерный отчет и график загруженности торговых точек по дням недели.

5. Десять самых продаваемых товаров.

6. Пять самых популярных товаров в каждой товарной группе.

7. Десять самых продаваемых товаров с 18 до 21 часа.

8. Десять самых продаваемых товаров по пятницам.

9. Пять самых популярных товаров в товарной группе «Местные анестетики»

10.Товары, дающие 80% объема продаж в летние месяцы.

11. Пять товаров, пользующихся наибольшим спросом по понедельникам до 12 часов дня.

Вопросы для проверки:

1. В чем заключается OLAP-анализ и каковы его цели?

2. Какова структура OLAP-куба?

3. Какие манипуляции с измерениями можно производить, чтобы сделать представление куба более информативным?

Тема 7. ОЧИСТКА И ПРЕДОБРАБОТКА ДАННЫХ – 2 ч.

Занятие 1. Алгоритмы обработки в Deductor Studio.

В платформу Deductor Studio встроены наиболее востребованные алгоритмы обработки данных для очистки данных. К алгоритмам обработки можно отнести [1]:

- фильтрацию;
- качество данных;
- заполнение пропусков;
- редактирование выбросов;
- дубликаты и противоречия;
- спектральная обработка;
- корреляционный анализ;
- факторный анализ.

Фильтрация

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

Фильтрация является многоцелевым средством, которое позволяет:

- выполнить очистку данных от записей, снижающих качество анализа;
- понижать размерность исходного множества данных;
- отбирать наиболее важные данные;
- упрощать визуальный анализ исходной выборки и т.п. операции.

Решение о целесообразности фильтрации и настройка ее параметров производится непосредственно пользователем, что дает возможность действовать более тонко, чем некоторые алгоритмы очистки и сокращения размерности. Кроме того, результаты фильтрации более предсказуемы и легче интерпретируются.

Обработчик **Фильтрация** предназначен для исключения из набора данных записей, не удовлетворяющих условиям фильтрации.

Обработчик **Фильтрация** находится в группе узлов **Очистка данных** мастера обработки. Параметры фильтрации задаются в виде списка условий, который содержит следующие столбцы:

1) **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. В результате фильтрации по каждому из полей или условий будет получено отдельное множество значений. 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»;

2) **Имя поля** – позволяет выбрать поле, по значениям которого должна быть выполнена фильтрация. Одно и то же поле может быть использовано в нескольких условиях;

3) **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 *29.11.2021*, то будут отобраны записи, в которых значение данного поля равно *29.11.2021, 30.11.2021, 01.12.2021* – три 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 29.11.2021, то будут отобраны записи, в которых значение данного поля равно 29.11.2021, 28.11.2021, 27.11.2021 – три предыдущих дня.

4) **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.

Флажок **Учитывать регистр** учитывает регистр символов при отборе.

Внизу окна настроек в автоматическом режиме формируется выражение для фильтрации, полученное объединением всех условий, например:

([Размер ссуды, руб] в интервале [2000..5000]) И ([Цель ссуды] = 'Покупка товара').

Иногда возникает необходимость построить фильтр для дискретного поля с условием в списке, вне списка для значений, которые не существуют в наборе данных (но предполагается, что они могут появиться в будущем). Выходом служит кнопка  **Добавить значение** в окне выбора списка значений (см. рис. 7.1). Количество записей такого «несуществующего» списочного значения всегда будет равно нулю, а строка – подкрашена светло-желтым цветом.

Практическое задание:

1. Создайте новый проект. Импортируйте в него текстовый файл **CreditSample.txt**, идущий в поставке Deductor (по умолчанию расположен в каталоге /Samples директории установки Deductor).

2. Отсортируйте этот набор данных по следующим полям в порядке возрастания: Срок ссуды, Размер ссуды, Количество иждивенцев.

3. Сделайте следующую замену (после **Сортировки**) в поле **Семейное положение**: значение Да измените на Женат/замужем, Нет – на Холост/Не замужем.

4. Сделайте следующую замену (после предыдущего узла **Замена данных**) в поле Количество иждивенцев: значение 0 – на Нет, 1 – без изменений, 2 и 3 – 2 и более. Используйте два способа – непосредственным вводом в мастере обработки и через файл таблицы соответствий. Файл подстановок предварительно создайте в любом текстовом редакторе, например, в Блокноте.

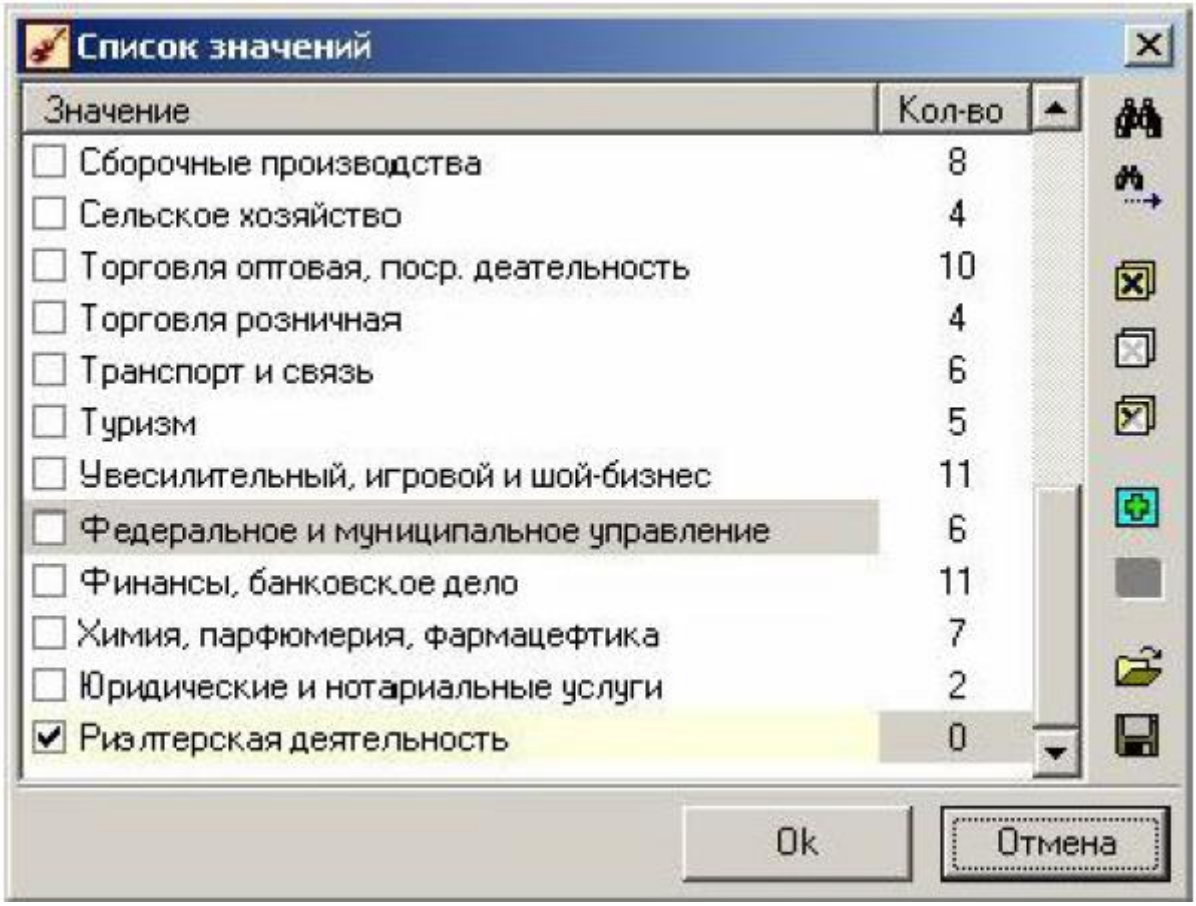


Рис.7.1. Окно «Список значений»

5. Старое поле Количество иждивенцев удалите из набора данных, а новое поле Количество иждивенцев_REPLACE переименуйте в Иждивенцы.

6. Отфильтруйте набор данных, полученный в п. 5 по полю Иждивенцы так, чтобы в выходной набор попали только строки, у которых значение в поле Иждивенцы не равно Нет. Сколько записей прошло через фильтр?

7. Отфильтруйте набор данных, полученный в п. 5 по полю Иждивенцы так, чтобы в выходной набор попали только строки, у которых значение в поле Иждивенцы не равно Н/д. Сколько записей прошло через фильтр?

8. Продолжите фильтровать набор данных, полученный в п. 6. Наложите следующий фильтр, в который попадают все записи, удовлетворяющие условиям *a* либо условиям *b*:

a) Размер ссуды – от 2000 до 5000, Цель ссуды – Покупка товара.

b. Цель ссуды – Иное.

9. Сколько записей прошло через фильтр?

10. Отсортируйте последний набор данных по полю Код.

Качество данных (Data quality) – обобщенное понятие, отражающее степень их пригодности к решению определенной задачи. В соответствии со стандартом ISO 9000:2015 основными критериями качества являются полнота, достоверность, точность, согласованность, доступность и своевременность.

Качество данных – одна из самых больших проблем бизнес-аналитики. Аналитические решения, полученные на основе некачественных данных, могут:

- могут оказаться далекими от действительности;
- исказить истинную картину исследуемых бизнес-процессов;
- показать ложные закономерности, тенденции и связи между объектами бизнеса.

Следствием этого может стать выработка неверных управленческих решений, которые нанесут ущерб бизнесу. Именно поэтому мониторингу качества данных, а также их преобразованию с целью исключения факторов, снижающих качество данных, должно уделяться особое внимание.

Оценка качества данных и действия по его повышению являются необходимым этапом любого аналитического проекта, поскольку аналитические алгоритмы или не смогут работать с некачественными данными либо будут давать некорректные результаты.

Приведение исходных «сырых» данных в соответствие с требуемыми критериями качества является важнейшей задачей Data Mining и образует целое направление, называемое предобработкой.

В качестве основных проблем, вызывающих снижение качества данных, обычно выделяют следующие:

- пропущенные значения;
- дубликаты – повторяющиеся записи одного набора данных, содержащие идентичные наборы значений всех признаков и рассматривающиеся как негативный фактор от которого необходимо избавиться в процессе очистки данных, т.к. дублирующие записи (кроме одной) не несут никакой полезной информации и бесполезны с точки зрения обучения аналитических моделей;
- противоречия – ситуация в анализе данных, при которой в двух записях множества данных одному и тому же набору значений входных атрибутов соответствуют различные наборы значений выходных;
- аномальные значения и выбросы, которые не укладываются в общую модель поведения анализируемого процесса и сильно отличающиеся от окружающих данных, вызванные как ошибками измерений, так и некорректным вводом данных, или являться результатом их сильной изменчивости;
- шум – аддитивная или мультипликативная составляющая сигнала, имеющая случайный характер и не несущая полезной информации (**белый шум** с непрерывным равномерным спектром и **цветной шум,** который локализован в ограниченных областях спектра);
- отсутствие полноты данных;
- нарушения целостности данных – не соответствие структуры и содержания базы данных предметной области;
- некорректные форматы и представления данных;

- фиктивные значения – некоторое значение, которое помещается в ячейку таблицы в случае, когда фактическое значение отсутствует или не вызывает доверия;
- ошибки ввода данных;
- нарушения структуры.

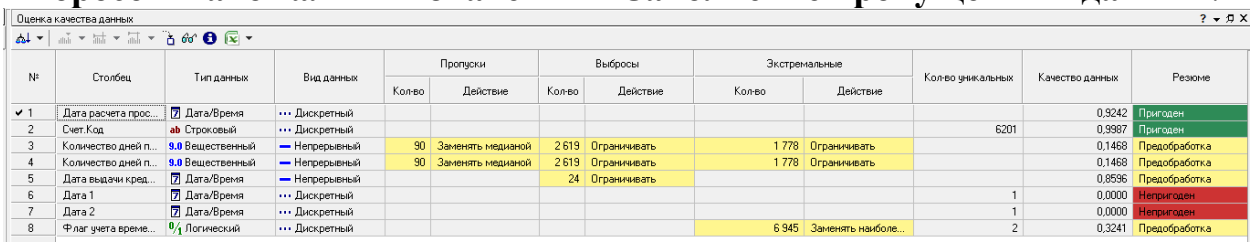
Некоторые из этих проблем являются критическими в том смысле, что они блокируют работу аналитических моделей и алгоритмов (например, пропущенные значения и нарушения структуры). Другие (например, дубликаты, противоречия, шумы) не нарушают работу алгоритмов, но порождают некорректные результаты анализа.

Независимо от того, какие факторы снижения качества присутствуют в данных, с ними необходимо бороться. Это делается в два этапа:

Профайлинг — исследование данных с целью выявления проблем и выработки стратегии их решения.

Очистка — применение различных методов для разрешения обнаруженных проблем: восстановление пропущенных значений, редактирование аномалий, обработка дубликатов и противоречий и т.д.

Обработчик **Качество данных** (см. рис. 7.2) позволяет производить комплексную оценку качества наборов данных на основе количества обнаруженных пропусков, выбросов и экстремальных значений, а также настроить параметры их дальнейшей обработки узлами **Редактирование выбросов и аномальных значений** и **Заполнение пропущенных данных**.


Оценка качества данных

№	Столбец	Тип данных	Вид данных	Пропуски		Выбросы		Экстремальные		Кол-во уникальных	Качество данных	Резюме
				Кол-во	Действие	Кол-во	Действие	Кол-во	Действие			
1	Дата расчета прос...	Дата/Время	Дискретный								0,9242	Пригоден
2	Счет.Код	Строковый	Дискретный							6201	0,9987	Пригоден
3	Количество дней п...	Вещественный	Непрерывный	90	Заменять медианой	2 619	Ограничивать	1 778	Ограничивать		0,1468	Предобработка
4	Количество дней п...	Вещественный	Непрерывный	90	Заменять медианой	2 619	Ограничивать	1 778	Ограничивать		0,1468	Предобработка
5	Дата выдачи кред...	Дата/Время	Непрерывный			24	Ограничивать				0,8596	Предобработка
6	Дата 1	Дата/Время	Дискретный							1	0,0000	Непригоден
7	Дата 2	Дата/Время	Дискретный							1	0,0000	Непригоден
8	Флаг учета време...	Логический	Дискретный					6 945	Заменять наиболе...	2	0,3241	Предобработка

Рис.7.2. Обработчик «Качество данных»

В настройках описываемого обработчика задаются критерии, по которым в данных определяются выбросы, экстремальные значения, значимость количества пропусков. По заданным полям собирается статистика, полученные значения проверяются по критериям и выдаются рекомендации о пригодности данных и необходимых действий по дальнейшей предобработке.

Заполнение пропусков – процедура, используемая при отсутствии некоторых данных в столбце в силу каких-либо причин (данные неизвестны, либо их забыли внести и т.п.). Удаление всех строк, содержащих пропущенные данные не всегда является способом решения проблемы, так как теряется информация по заполненным столбцам, либо в результате удаления данных для анализа может остаться слишком мало.

Для решения этой проблемы в Deductor Studio используется специальный обработчик **Заполнение пропусков**, который может как читать настройки,

сделанные в узле **Качество данных**, так и быть самостоятельным со своими параметрами.

В таблице 7.1 приведены все возможные методы восстановления пропусков в зависимости от типа данных и того, является ли набор данных упорядоченным или нет.

Таблица 7.1. Перечень методов восстановления пропусков в зависимости от типа данных и упорядоченности набора данных.

Метод	Неупорядоченное поле		Упорядоченное поле	
	Непрерывное	Дискретное	Непрерывное	Дискретное
Оставить без изменения	+	+	+	+
Заменять наиболее вероятным	+	+	+	+
Заменять случайными значениями	+	+	+	+
Заменять средним	+		+	
Заменять медианой	+		+	
Заменять значением *Не задано*		+		+
Интерполировать			+	
Удалять записи	+	+		

Рассмотрим указанные методы:

- *Заменять наиболее вероятным* – в случае непрерывных данных замена производится на среднее значение из наиболее вероятного интервала, число интервалов варьируется в зависимости от объема выборки – чем она больше, тем больше интервалов; в дискретном случае – выбирается значение с наибольшей вероятностью.
- *Заменять случайными значениями* – производится замена пропусков на случайное значение из распределения, параметры которого оцениваются из имеющихся значений в столбце.
- *Заменять средним* – рассчитывается среднее значением, которым заменяются все пропуски.
- *Заменять медианой* – рассчитывается медиана и ей заменяются все пропуски.
- *Заменять значением* **Не задано** – доступно только для дискретного поля, выполняется замена пропусков на значение «Не задано». Метод доступен для строковых полей.
- *Удалять записи* – строки с выявленными пропусками исключаются из набора данных. Метод недоступен для упорядоченных рядов.

- Метод *интерполяции* доступен только для упорядоченных данных, чаще всего это временные ряды. Восстановление пропусков в столбцах, значения в которых упорядочены, можно рассматривать как интерполяцию значений функции в точках, где она неизвестна. Данная задача реализуется с помощью А-сплайнов.

Редактирование выбросов

Обработчик **Редактирование выбросов** предназначен для автоматической корректировки аномальных значений в наборах данных – отклонений от нормального (ожидаемого) поведения чего-либо.

Для повышения гибкости обработки аномальных значений в узле предусмотрена возможность их разделения, поскольку они в большинстве случаев имеют различное происхождение:

- *выбросы* – это фактически имевшие место события, вызванные исключительными условиями;
- *экстремальные значения* – это, как правило, ошибки или фиктивные значения.

Для каждого типа отклонений определяется собственный порог обнаружения, что позволяет сделать процедуру очистки данных более соответствующей логике решаемой задачи (по умолчанию это 3 стандартных отклонения для выброса, 5 стандартных отклонений для экстремального значения).

Для каждого столбца исходного набора данных пользователь может выбрать подходящий метод выявления и редактирования выбросов и экстремальных значений. Алгоритмы поиска и набор методов для борьбы с выбросами в зависимости от вида данных столбца и того, считается набор данных упорядоченным или нет, приведены в табл. 7.2.

Таблица 7.2. Перечень алгоритмов поиска и методов редактирования выбросов в зависимости от типа данных и упорядоченности набора данных.

Метод	Неупорядоченное поле		Упорядоченное поле	
	Непрерывное	Дискретное	Непрерывное	Дискретное
Оставить без изменения	+	+	+	+
Удалять записи	+	+		
Ограничивать	+	+	+	+
Заменять наиболее вероятным	+	+	+	+
Заменять средним	+		+	
Заменять медианой	+		+	
Заменять заданным значением…		+		+

Сглаживать			+	

Особый интерес представляет метод **Ограничивать** – приведение экстремального значения или выброса к значению, превышение которого определяется как выброс.

Задача редактирования выбросов во многих случаях сводится к задаче заполнения пропусков – выброс заменяется пропуском, и к нему применяются те или иные методы, доступные также и в обработчике **Заполнение пропусков**.

Дубликаты и противоречия – механизм обработки, позволяющий выявить в исходной выборке данных дублирующие и противоречивые записи. Основные понятия этой операции:

- *дубликаты* – это одинаковые данные (записи), приводящие к избыточности и увеличивающие объем выборки, но при этом не повышающие информативность данных;
- *противоречивые записи* – те записи, которые содержат одинаковые наборы значений для входных признаков и различные наборы значений выходных признаков. Противоречия приводят к искажению результата анализа и снижают качество моделей, поскольку нарушают общие закономерности в данных, обнаружение которых и является целью анализа.

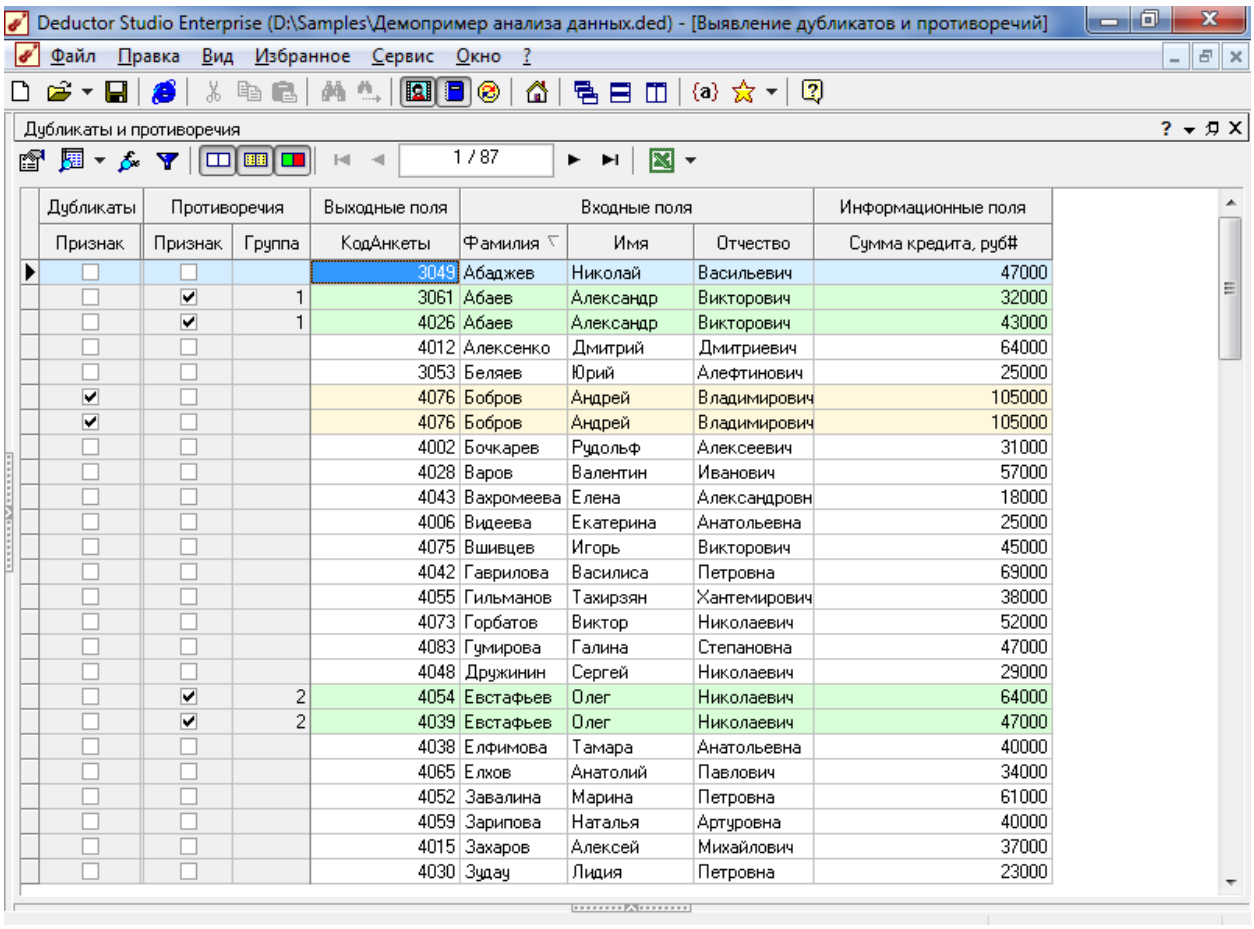
Алгоритм ищет в наборе в данных записи, для которых одинаковым входным полям соответствуют одинаковые (дубликаты) или разные (противоречия) выходные поля. На основании этой информации создаются два дополнительных поля – *Дубликат* и *Противоречие*, принимающие значения *истина* или *ложь*, и дополнительные поля *Группа дубликатов* и *Группа противоречий*. Если запись не является дубликатом или противоречием, то соответствующие поля будут пустыми (см. рис 7.3).

Спектральная обработка

Часто ряды данных содержат быстрые случайные изменения значений, которые можно рассматривать как шум, который мешает выполнить анализ данных, делает неустойчивой работу алгоритмов и не позволяет обнаруживать в данных скрытые закономерности, структуры, тенденции.

Спектральный анализ – это широкий класс методов обработки данных, в основе которых лежит их частотное представление, или спектр. Спектр получается в результате разложения исходной функции, зависящей от времени (временной ряд) или пространственных координат (например, изображения), в базис некоторой периодической функции.

Механизм **Спектральная обработка** предназначен для очистки от шумовой составляющей и сглаживания рядов данных. Сглаживание необходимо в том случае, когда ряд данных оказывается неравномерным, содержит большое количество мелких структур, препятствующих исследованию более значительных объектов и закономерностей (см. рис. 7.4 и 7.5).



Дубликаты	Противоречия		Выходные поля	Входные поля			Информационные поля
Признак	Признак	Группа	КодАнкеты	Фамилия	Имя	Отчество	Сумма кредита, руб#
☐	☐		3049	Абаджев	Николай	Васильевич	47000
☐	☑	1	3061	Абаев	Александр	Викторович	32000
☐	☑	1	4026	Абаев	Александр	Викторович	43000
☐	☐		4012	Алексенко	Дмитрий	Дмитриевич	64000
☐	☐		3053	Беляев	Юрий	Алефтинович	25000
☑	☐		4076	Бобров	Андрей	Владимирович	105000
☑	☐		4076	Бобров	Андрей	Владимирович	105000
☐	☐		4002	Бочкарев	Рудольф	Алексеевич	31000
☐	☐		4028	Варов	Валентин	Иванович	57000
☐	☐		4043	Вахромеева	Елена	Александровн	18000
☐	☐		4006	Видеева	Екатерина	Анатольевна	25000
☐	☐		4075	Вшивцев	Игорь	Викторович	45000
☐	☐		4042	Гаврилова	Василиса	Петровна	69000
☐	☐		4055	Гильманов	Тахирзян	Хантемирович	38000
☐	☐		4073	Горбатов	Виктор	Николаевич	52000
☐	☐		4083	Гумирова	Галина	Степановна	47000
☐	☐		4048	Дружинин	Сергей	Николаевич	29000
☐	☑	2	4054	Евстафьев	Олег	Николаевич	64000
☐	☑	2	4039	Евстафьев	Олег	Николаевич	47000
☐	☐		4038	Елфимова	Тамара	Анатольевна	40000
☐	☐		4065	Елхов	Анатолий	Павлович	34000
☐	☐		4052	Завалина	Марина	Петровна	61000
☐	☐		4059	Зарипова	Наталья	Артуровна	40000
☐	☐		4015	Захаров	Алексей	Михайлович	37000
☐	☐		4030	Зудау	Лидия	Петровна	23000

Рис.7.3. Выявление дубликатов и противоречий.

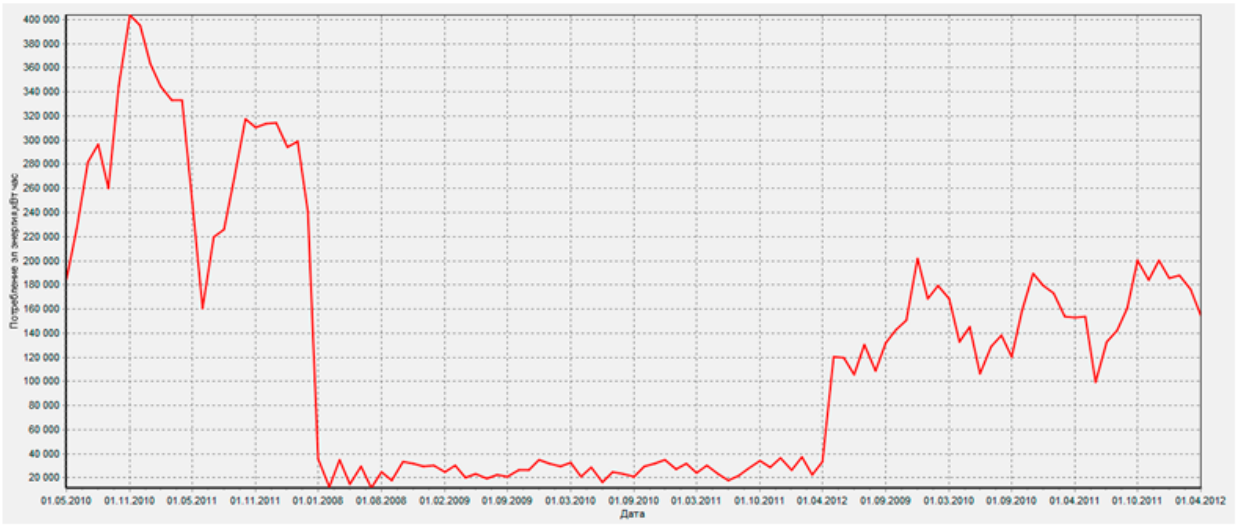


Рис.7.4. Временной ряд до обработки.

Для этих целей в узле используются два подхода: *преобразование Фурье* и *вейвлеты*, которые будут рассмотрены ниже. Принцип подобной обработки состоит в разложении исходной функции временного ряда на базисные функции с определенной частотой и амплитудой, после чего уменьшается амплитуда высокочастотных составляющих.

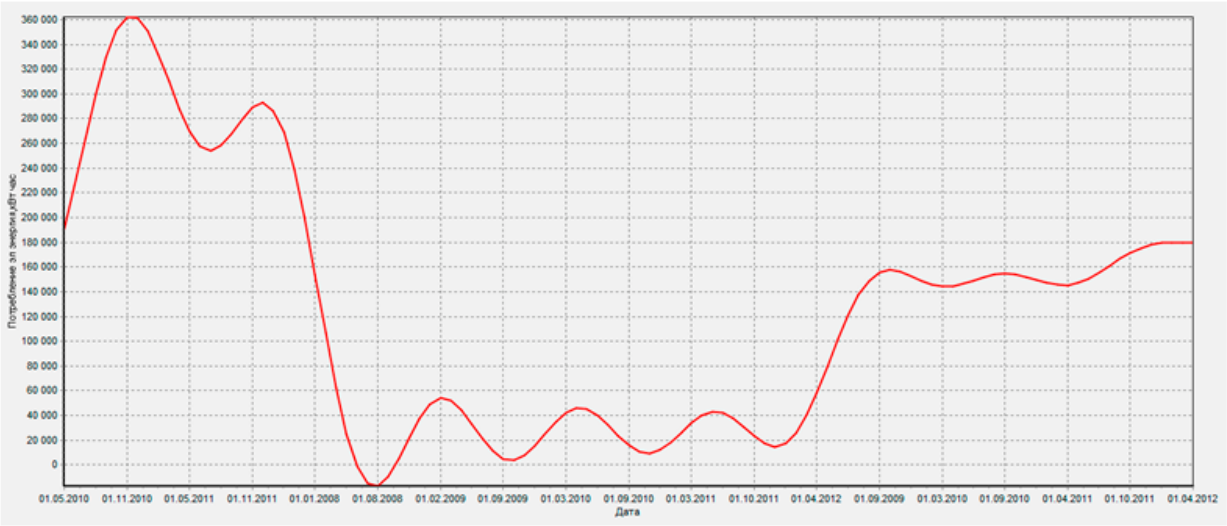


Рис.7.5. Временной ряд после обработки.

Для временного ряда, например, это означает убрать информацию об ежедневных продажах, которые сильно подвержены случайным факторам, и оставить более устойчивые тенденции, например, сезонность. Можно, наоборот, подавить составляющие с низкой частотой, что позволит убрать медленные изменения, а оставить только быстрые. В случае временного ряда это будет означать подавление сезонной компоненты.

Спектр Фурье – метод, наиболее часто используемый для спектральной обработки и получаемый на основе базиса синуса (разложение Фурье, преобразование Фурье). Основной смысл преобразования Фурье в том, что исходная непериодическая функция произвольной формы, которую невозможно описать аналитически и поэтому сложно обрабатывать и анализировать, представляется в виде набора синусов или косинусов с различной частотой, амплитудой и начальной фазой.

Иными словами, сложная функция преобразуется во множество более простых. Каждая синусоида (или косинусоида) с определенной частотой и амплитудой, полученная в результате разложения Фурье, называется **спектральной составляющей** или **гармоникой**. Спектральные составляющие образуют **спектр Фурье**.

Визуально спектр Фурье представляется в виде графика, представленного на рис. 7.6, на котором по горизонтальной оси откладывается круговая частота, обозначаемая греческой буквой «омега», а по вертикали – амплитуда спектральных составляющих, обычно обозначаемая латинской буквой А. Тогда каждая спектральная составляющая может быть представлена в виде отсчета, положение которого по горизонтали соответствует ее частоте, а высота – ее амплитуде. Гармоника с нулевой частотой называется **постоянной составляющей** (во временном представлении это прямая линия).

Даже простой визуальный анализ спектра может много сказать о характере функции, на основе которой он был получен. Интуитивно понятно, что быстрые изменения исходных данных порождают в спектре составляющие с *высокой*

частотой, а медленные – с *низкой*. Поэтому если в нем амплитуда составляющих быстро убывает с увеличением частоты, то исходная функция (например, временной ряд) является плавной, а если в спектре присутствуют высокочастотные составляющие с большой амплитудой, то исходная функция будет содержать резкие колебания. Так, для временного ряда это может указывать на большую случайную составляющую, неустойчивость описываемых им процессов, наличие шумов в данных.

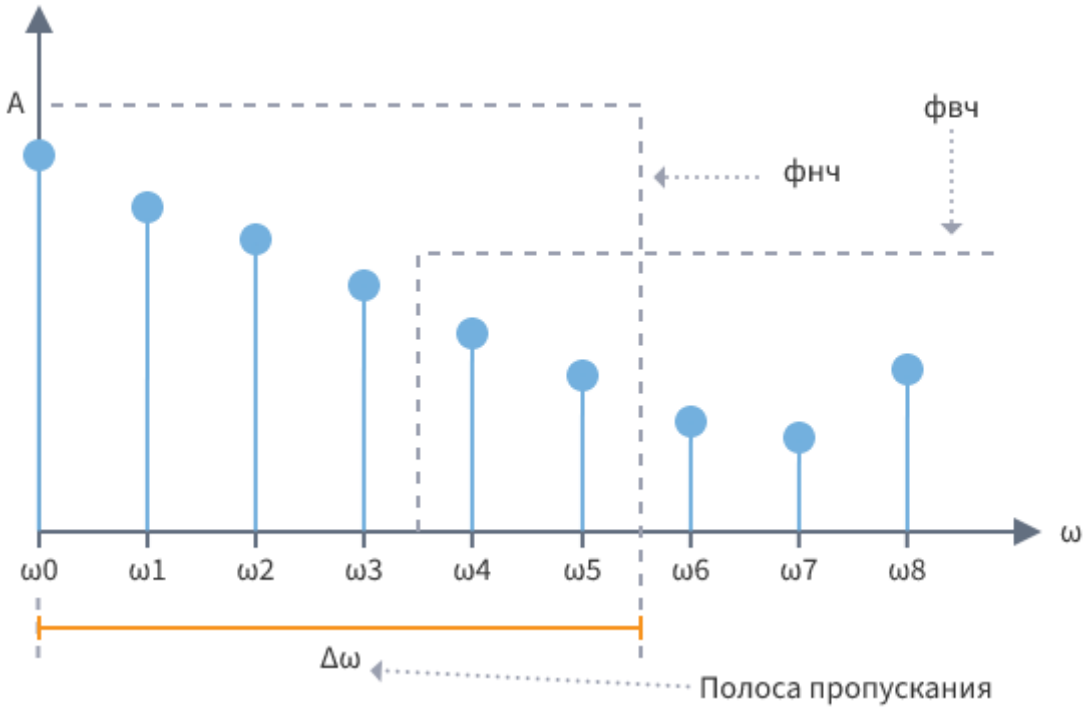


Рис.7.6. График спектра Фурье.

В основе спектральной обработки лежит манипулирование спектром. Действительно, если уменьшить (подавить) амплитуду высокочастотных составляющих, а затем на основе измененного спектра восстановить исходную функцию, выполнив обратное преобразование Фурье, то она станет более гладкой за счет удаления высокочастотной компоненты.

Для временного ряда, например, это означает убрать информацию об ежедневных продажах, которые сильно подвержены случайным факторам, и оставить более устойчивые тенденции, например, сезонность. Можно, наоборот, подавить составляющие с низкой частотой, что позволит убрать медленные изменения, а оставить только быстрые. В случае временного ряда это будет означать подавление сезонной компоненты.

Применяя спектр таким образом, можно добиваться желаемого изменения исходных данных. Наиболее часто используется сглаживание временных рядов путем удаления или уменьшения амплитуды высокочастотных составляющих в спектре.

Для манипуляций со спектрами используются фильтры – алгоритмы, способные управлять формой спектра, подавлять или усиливать его составляющие. Главным **свойством** любого **фильтра** является его амплитудно-

частотная характеристика (АЧХ), от формы которой зависит преобразование спектра.

Если фильтр пропускает только спектральные составляющие с частотой ниже некоторой граничной частоты, то он называется фильтр нижних частот (ФНЧ), и с его помощью можно сглаживать данные, очищать их от шума и аномальных значений.

Если фильтр пропускает спектральные составляющие выше некоторой граничной частоты, то он называется фильтром верхних частот (ФВЧ). С его помощью можно подавлять медленные изменения, например, сезонность в рядах данных.

Кроме этого, используется множество других типов фильтров: фильтры средних частот, заградительные фильтры и полосовые фильтры, а также более сложные, которые применяются при обработке сигналов в радиоэлектронике. Подбирая тип и форму частотной характеристики фильтра, можно добиться желаемого преобразования исходных данных путем спектральной обработки.

Выполняя частотную фильтрацию данных с целью сглаживания и очистки от шума, необходимо правильно указать полосу пропускания ФНЧ. Если ее выбрать слишком большой, то степень сглаживания будет недостаточной, а шум будет подавлен не полностью. Если она будет слишком узкой, то вместе с шумом могут оказаться подавленными и изменения, несущие полезную информацию. Если в технических приложениях существуют строгие критерии для определения оптимальности характеристик фильтров, то в аналитических технологиях приходится использовать в основном экспериментальные методы.

Спектральный анализ является одним из наиболее эффективных и хорошо разработанных методов обработки данных. **Частотная фильтрация** – только одно из его многочисленных приложений. Кроме этого, он используется в корреляционном и статистическом анализе, синтезе сигналов и функций, построении моделей и т.д.

Вейвлет (Wavelet) (*от англ. wavelet* – всплеск, маленькая волна) – это класс математических функций, которые на графике выглядят как кратковременное колебание, волнообразно увеличивающееся до максимума и так же волнообразно спадающее до нуля (см. рис. 7.7).

Вейвлеты используются в качестве базисных функций в так называемом вейвлет-преобразовании, которое, в свою очередь позволяет реализовывать кратномасштабный анализ данных.

В основе идеи использования вейвлетов в обработке данных лежит время-частотное представление данных, когда они локализуются одновременно по времени и по частоте. Основные приложения вейвлетов – сжатие данных, очистка от шума и сглаживание.

Таким образом, механизм **Спектральная обработка** наиболее часто можно применить для предварительной подготовки данных в задачах прогнозирования, т.к. позволяет сделать временной ряд более гладким, благодаря чему полученная прогнозная модель обладает высокими обобщающими качествами.

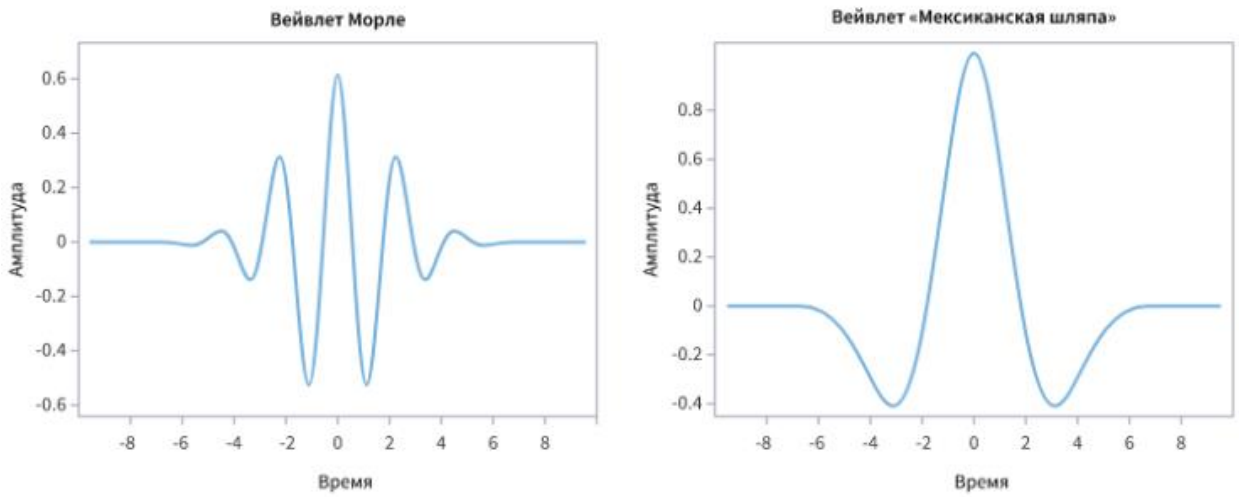


Рис.7.7. Примеры вейвлетов.

Корреляционный анализ – механизм, применяющийся для оценки степени линейной зависимости между парами факторов, выполняющийся с целью отбора и предобработки входных полей для использования в обучаемых на данных моделях. Например, наличие корреляции между входными факторами крайне отрицательно сказывается при построении линейной регрессии.

В настройках обработчика указывается входные и выходные поля. В результате получается таблица с коэффициентами корреляции для каждой возможной пары из комбинации входного и выходного факторов. Коэффициент корреляции принимает значения от -1 до 1.

Модуль коэффициента свидетельствует о степени зависимости: чем ближе его значение к 0, тем слабее линейная зависимость. Чем ближе коэффициент корреляции от 0 к 1, тем сильнее прямая линейная зависимость, чем ближе от 0 к -1, тем сильнее обратная линейная зависимость. На практике считается, что если модуль коэффициента корреляции больше 0,6, то линейная зависимость сильная, а если менее 0,3, то почти отсутствует.

Стоит заметить, что низкая степень корреляции между входным и прогнозируемым полями не означает отсутствие других, нелинейных зависимостей. Кроме того, при построении линейных моделей стоит рассмотреть такой входной фактор внимательнее, так как он может быть использован для проектирования признаков (Feature Engineering).

Пример

Пусть необходимо быстро определить товары-заменители и сопутствующие товары, имея временные ряды объемов продаж (см. табл. 7.3). У товаров-заменителей должна быть большая отрицательная корреляция, т.к. увеличение продаж одного товара ведет к спаду продаж второго. У сопутствующих товаров – большая положительная корреляция.

Таблица 7.3. Перечень товаров-заменителей и сопутствующих товаров.

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

Определим корреляцию *Товар 1* с остальными товарами (см.рис. 7.8).

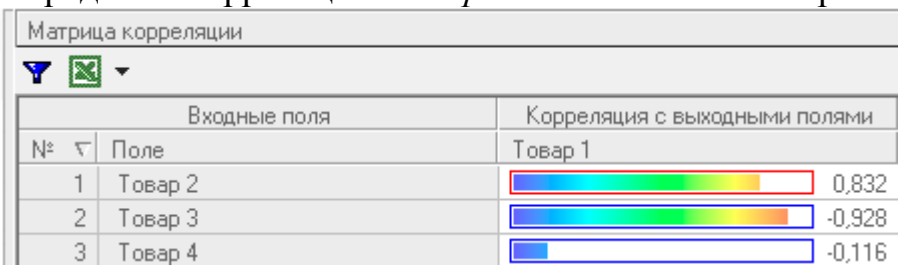


Рис.7.8. Визуализатор Матрица корреляции.

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 *Товара 1*.

Корреляция продаж *Товара 4* с *Товаром 1* является отрицательной, но при этом абсолютное значение корреляции невелико, поэтому говорить о наличии взаимосвязи между продажами *Товара 1* и *Товара 4* без проведения дополнительного анализа нельзя.

Факторный анализ

Нередко, в наборе данных объекты описываются огромным числом признаков. Однако здесь могут возникнуть следующие проблемы:

- наличие слабоинформативных и неинформативных признаков;
- мультиколлинеарность.

Анализировать большое число признаков на наличие указанных проблем крайне затруднительно. В данном случае уместнее применить метод главных компонент, который реализован в обработчике **Факторный анализ**. В результате будет получено новое пространство признаков меньшей размерности, избавленных от указанных выше недостатков.

Примеры применения:

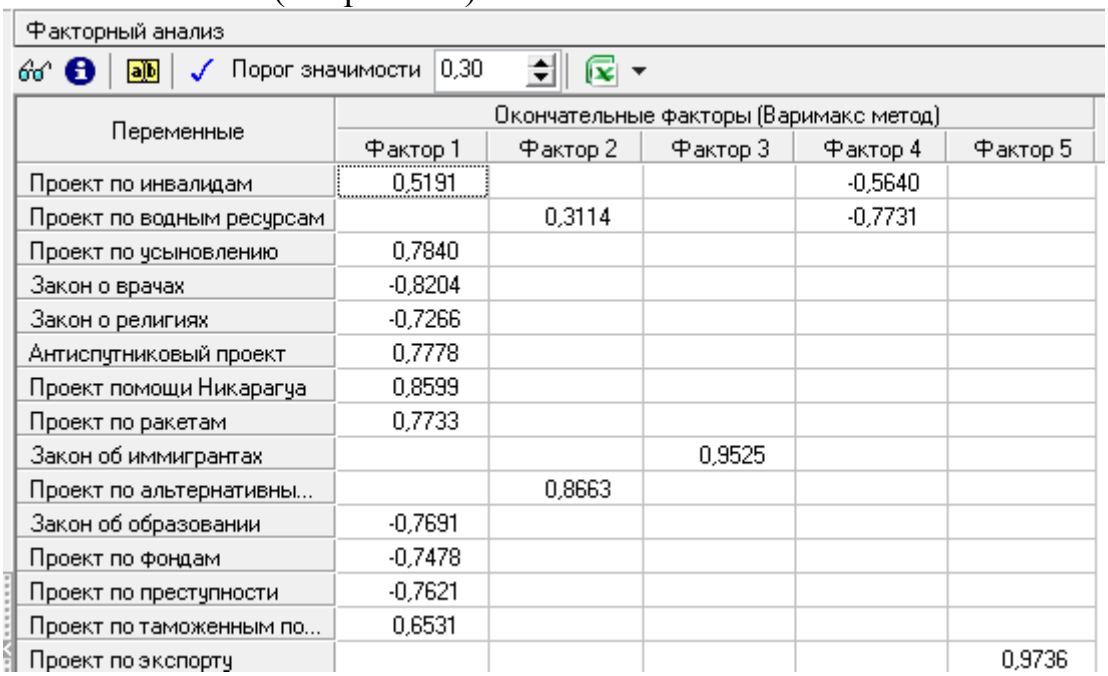
1. Сегментация клиентской базы. Для каждого клиента известно сколько покупок в той или иной группе товаров он сделал. Каждая группа – это отдельный признак. Таким образом, получаются сотни факторов. Для повышения качества кластеризации необходимо сократить размерность признакового пространства, для чего используется метод главных компонент, реализованный в обработчике **Факторный анализ**.

2. Прогнозирование распространения инфекционного заболевания. В качестве исторических данных нередко используются исторические сведения о погоде (температура воздуха, влажность). Данная информация позволяет учесть

факторы, благоприятно влияющие на размножение переносчиков заболевания. Поэтому каждый объект в выборке описывается десятками признаков. Так как изменения погодных условий, как правило, происходит постепенно, это приводит к наличию мультиколлинеарности в указанных данных. Для решения указанной проблемы в данной задаче может быть использован метод главных компонент.

Описание алгоритма

В Deductor Studio *факторный анализ* базируется на *методе главных компонент*. Алгоритм основан на преобразовании исходной матрицы и расчете собственных чисел. Геометрический смысл преобразований заключается в следующем. Координатные оси в исходном пространстве признаков подвергаются повороту. В результате чего каждая ось образует новый фактор. На основе матрицы, полученной из исходного набора данных, для каждого фактора рассчитывается значимость в новом пространстве признаков. Сокращение размерности заключается в том, что будут оставлены только новые факторы с высокой значимостью (см. рис. 7.9).



Переменные	Окончательные факторы (Варимакс метод)				
	Фактор 1	Фактор 2	Фактор 3	Фактор 4	Фактор 5
Проект по инвалидам	0,5191			-0,5640	
Проект по водным ресурсам		0,3114		-0,7731	
Проект по усыновлению	0,7840				
Закон о врачах	-0,8204				
Закон о религиях	-0,7266				
Антиспутниковый проект	0,7778				
Проект помощи Никарагуа	0,8599				
Проект по ракетам	0,7733				
Закон об иммигрантах			0,9525		
Проект по альтернативны...		0,8663			
Закон об образовании	-0,7691				
Проект по фондам	-0,7478				
Проект по преступности	-0,7621				
Проект по таможенным по...	0,6531				
Проект по экспорту					0,9736

Рис.7.9. Визуализатор «Факторный анализ».

На практике аналитикам чаще всего интересен факторный анализ с ортогональным вращением осей, когда при повороте осей координат угол между факторами остается прямым. Обработчик **Факторный анализ** реализует два метода вращения: *варимакс* и *квартимакс*:

- *Варимакс* – наиболее часто используемый на практике метод, цель которого – минимизировать количество переменных, имеющих высокие нагрузки на данных фактор, что способствует упрощению описания фактора за счет группировки вокруг него только тех переменных, которые с ним связаны в большей степени, чем с остальными.
- *Квартимакс* противоположен варимаксу, поскольку минимизирует количество факторов, необходимых для объяснения данной переменной.

Квартимакс-вращение приводит к выделению одного из общих факторов с достаточно высокими нагрузками на большинство переменных.

Вопросы для проверки:

1. Для чего предназначен узел **Фильтр**?
2. Какие условия фильтрации существуют?
3. Сколько записей будет отфильтровано в результате фильтра «([Размер ссуды, руб] в интервале [2000. . 5000]) И ([Цель ссуды] = ' Покупка товара') И ([Цель ссуды] = 'Иное')»?
4. Что делать, если нужно поставить фильтр по значению, которого в данный момент нет в рассматриваемом наборе данных?

Список литературы

1. BaseGroupLab – Технологии анализа данных. Алгоритмы обработки [Электронный ресурс] – URL: https://basegroup.ru/deductor/function/algorithm?algorytm_group%5B%5D=82 (дата обращения 10.12.2021 г.).

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 $I = \{i\ 1, i\ 2,..., in\}$ - 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 I, Y $\subset X$ $\subset =$ $\cap$ $\varnothing$

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

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 $X \Longrightarrow Y$, 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	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	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 также 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.

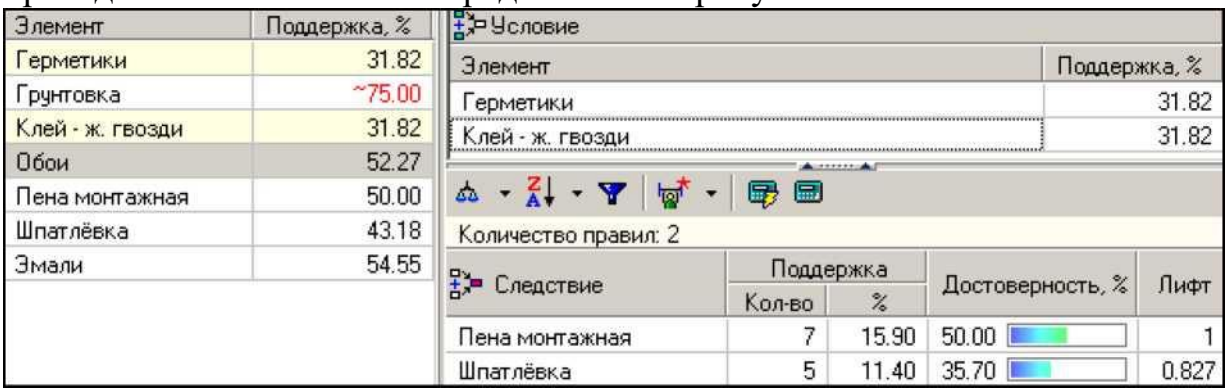
В примере узлы *Герметики*, *Клей-ж. гвозди*, *Пена монтажная* находятся на верхнем уровне дерева и представляют собой условия. А *Герметики*, *Пена монтажная*, *Шпатлёвка* и т.д. – следствия. Это означает, что человек, купивший *Клей-ж. гвозди*, купит еще и *Герметик* с достоверностью 71,40%, пену монтажную 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 использо я 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*.

Применение ассоциативных правил для оценки и выбора управленческих решений.

Ассоциативные правила - установление закономерностей между связанными событиями.

Впервые эта задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому ее еще называют анализом рыночной корзины (market basket analysis).

Ассоциация имеет место в том случае, если несколько событий связаны друг с другом. Например, исследование, проведенное в компьютерном супермаркете, может показать, что 55 % купивших компьютер берут также и принтер или сканер, а при наличии скидки за такой комплект принтер приобретают в 80 % случаев. Располагая сведениями о подобной ассоциации, менеджерам легко оценить, насколько действенна предоставляемая скидка.

Если существует цепочка связанных во времени событий, то говорят о последовательности. Так, например, после покупки дома в 45 % случаев в течение месяца приобретается и новая кухонная плита, а в пределах двух недель 60 % новоселов обзаводятся холодильником.

Примеры применения ассоциативных правил:

- выявление наборов товаров, которые в супермаркетах часто покупаются

вместе или никогда не покупаются вместе;

- определение доли клиентов, положительно относящихся к нововведениям в их обслуживании;
- определение профиля посетителей веб-ресурса;
- определение доли случаев, в которых новое лекарство оказывает опасный побочный эффект.

Пусть имеется база данных, состоящая из покупательских транзакций. Каждая транзакция - это набор товаров, купленных покупателем за один визит. Предметный набор - это непустое множество предметов, появившихся в одной транзакции.

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 возможность находить очень простые и понятные правила.

Ассоциативное правило состоит из двух наборов предметов: условие (antecedent) и следствие (consequent).

«Если условие, то следствие»; X → Y; «Из X следует Y».

Условие и следствие часто называются соответственно левосторонним (left-hand side - LHS) и правосторонним (right- hand side - RHS) компонентами ассоциативного правила.

Объективными показателями значимости ассоциативных таких правил являются ***поддержка*** и ***достоверность***.

Поддержка – количество или процент транзакций, содержащих как условие, так и следствие. Правило X → Y имеет *поддержку S* (support), если:

$$S(X \to Y) = \frac{\textit{количество транзакций, содержащих X и Y}}{\textit{общее количество транзакций}} \cdot 100\% .$$

Достоверность ассоциативного правила (confidence) C представляет собой меру точности правила и определяется, как отношение количества транзакций, содержащих условие и следствие, к количеству транзакций, содержащих только следствие:

$$C(X \to Y) = \frac{\textit{количество транзакций, содержащих X и Y}}{\textit{общее количество транзакций, содержащих X}} \cdot 100\% .$$

Пример. Пусть 75 % транзакций, содержащих хлеб, также содержат молоко, а 3 % от общего числа всех транзакций содержат оба товара, тогда 75 % - это достоверность правила, а 3 % - это поддержка.

Если поддержка и достоверность достаточно высоки, можно с большой вероятностью утверждать, что любая будущая транзакция, которая включает условие, будет также содержать и следствие.

Сильные правила - правила, для которых значения поддержки и достоверности превышают заданный порог.

Алгоритмы поиска ассоциативных правил предназначены для нахождения

всех правил вида X → Y, причем поддержка и достоверность этих правил должны находиться в рамках некоторых наперед заданных границ, называемых соответственно минимальной и максимальной поддержкой и минимальной и максимальной достоверностью.

Границы значений параметров поддержки и достоверности 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 часто статистически необоснованных правил. Тем не менее большинство интересных правил находится именно при низком значении порога поддержки. Еще одним параметром, ограничивающим количество найденных правил, является максимальная мощность часто встречающихся множеств. Если этот параметр указан, то при поиске правил будут рассматриваться только множества, количество элементов которых будет не больше данного параметра k, следовательно, любое найденное правило будет состоять не больше чем из *k* элементов.

Субъективные показатели значимости ассоциативных правил: *Лифт, Левередж, Улучшение* также могут использоваться для последующего ограничения набора рассматриваемых ассоциаций путем установки порога значимости, ниже которого ассоциации отбрасываются.

Лифт *(lift)* - это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом:

L(A → B) = C(A → B) / S(B).

Лифт является обобщенной мерой связи двух предметных наборов: при значениях лифта > 1 связь положительная, при 1 она отсутствует, а при значениях < 1 – отрицательная. Значения лифта большие, чем единица, показывают, что условие чаще появляется в транзакциях, содержащих следствие, чем в остальных.

Левередж (leverage) – это разность между наблюдаемой частотой, с которой условие и следствие появляются совместно (то есть поддержкой ассоциации), и произведением частот появления (поддержек) условия и следствия по отдельности: T(A → B) = S(A → B) – S(A)S(B).

Если Левередж≈0, то правило не значимо.

Улучшение (improvement) показывает, полезнее ли правило случайного угадывания.

Если $I(X \rightarrow Y) > I(X \rightarrow Y) = \frac{S(X \rightarrow Y)}{S(X) \cdot S(Y)}$ ероятнее предсказать наличие набора B с помощью правила, чем угадать случаино.

Поиск ассоциативных правил *алгоритм Apriori* (1994 г.)

Методика поиска:

1. Следует найти частые наборы.

Частый предметный набор - предметный набор с поддержкой больше заданного порога либо равной ему. Этот порог называется минимальной поддержкой.

При поиске частых наборов используется свойство антимонотонности: Если предметный набор Z не является частым, то добавление некоторого нового предмета A к набору Z не делает его более частым. Другими словами, если Z не является частым набором, то и набор $Z \cup A$ также не будет являться таковым.

Данное полезное свойство позволяет значительно уменьшить пространство поиска ассоциативных правил.

2. На основе найденных наборов необходимо сгенерировать ассоциативные правила, удовлетворяющие условиям минимальной поддержки и достоверности.

– Генерируются все возможные поднаборы *s*.

– Если поднабор *ss* является непустым поднабором *s*, то рассматривается ассоциативное правило *R*: *ss* → *(s – ss)*, где *s – ss* представляет собой набор *s* без поднабора *ss*. *R* будет считаться ассоциативным правилом, если будет удовлетворять условию заданного минимума поддержки и достоверности.

Данная процедура повторяется для каждого подмножества *ss* из *s*.

Пример. Пусть имеется база транзакций, представленная в табл. 12.

База данных транзакций Таблица 12.

Номер чека	Товар
160698	кетчупы, соусы, аджика, чай, макаронные изделия
160747	макаронные изделия, сыры
161343	кетчупы, соусы, аджика, макаронные изделия, сыры
161354	макаронные изделия, соусы, сыры
162915	вафли, сухари, чай
167414	чай, вафли, мед, сухари
167465	вафли, сыры, мед, сухари, чай
166474	кетчупы, сыры, макаронные изделия

Представим базу данных транзакций в нормализованном виде:

База данных транзакций в нормализованном виде Таблица 13.

№ транзакции	кетчупы	соусы	аджика	чай	сыры	макар. изделия	вафли	сухари	мед
1	1	1	1	1	0	1	0	0	0
2	0	0	0	0	1	1	0	0	0
3	1	1	1	0	1	1	0	0	0
4	0	1	0	0	1	1	0	0	0
5	0	0	0	1	0	0	1	1	0
6	0	0	0	1	0	0	0	0	1

7	0	0	0	1	1	0	1	1	1
8	1	0	0	0	1	1	0	0	0
частота	3	3	2	4	5	4	2	2	2
поддержка	0,375	0,375	0,25	0,5	0,625	0,625	0,25	0,25	0,25

Будем считать популярными наборами те, которые имеют поддержку S ≥30 %. Выпишем множество популярных однопредметных наборов:

F_1 = {кетчупы, соусы, чай, сыры, макар. изделия}.

Теперь переходим к поиску популярных двухпредметных наборов:

Двухпредметные наборы Таблица 14.

набор	кол-во	поддержка	набор	кол-во	поддержка
кетчупы, соусы	2	0,25	соусы, чай	1	0,125
кетчупы, чай	1	0,125	соусы, сыры	1	0,125
кетчупы, сыры	2	0,25	соусы, макар. изделия	3	0,375
кетчупы, макар. изделия	3	0,375	чай, сыры	1	0,125
сыры, макар. изделия	4	0,5	чай, макар. изделия	1	0,125

Выпишем множество популярных двухпредметных наборов, которые имеют поддержку $S \geq 30$ %: $\{кетчупы, макар.изделия\}$

Выпишем трехпредм$F_2 = \{сыры, макар.изделия\}$ерацию связывания популярных двухпредметных $\{соусы, макар.изделия\}$

$\{кетчупы, макар.изделия\} \{соусы, макар.изделия\} =$

$\{кетчупы, соусы, макар.изделия\}$, его поддержка S=0,25.

$\{кетчупы, макар.изделия\} \{сыры, макар.изделия\} =$

$\{кетчупы, сыры, макар.изделия\}$, его поддержка S=0,25.

Таким образом, трехпредметных популярных наборов нет, и задача поиска частных предметных наборов завершена.

Переходим к генерации ассоциативных правил на найденных популярных двухпредметных наборах.

Ассоциативные правила Таблица 15.

Правило	Под-держка	Досто-вер-ность	Лифт	Левередж	Улучшение

Если пы, то кар. изделия	3/8=0,375	3/3=1	1/0,625=1,6	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6
Если изделия, то кетчупы	3/8=0,375	3/4=0,75	0,75/0,375=2	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6
Если сыры, то макар. изделия	4/8=0,5	4/5=0,6	0,6/0,625=0,96	0,5-0,625*0,625=0,11	0,5/(0,625*0,625)==1,28
Если соусы, макар. лия	3/8=0,375	3/3=1	1/0,625=1,6	0,375-0,375*0,625=0,14	0,375/(0,375*0,625)==1,6

Рассмотрим первое правило *кетчупы макаронные изделия:* *S*=37,5%; *C*=100 %; *L*=1,6; *T*=0,14; *I*=1,6. Это означает следующее:

Так как *S*=37,5 %, ожидаемая вероятность покупки набора *кетчупы+макаронные изделия* равна 37,5 %.

Так как *C*=100 %, это значит, что если покупатель купит *кетчупы,* то с вероятностью 100 % он купит и *макаронные изделия*.

Так как лифт *L*=1,6, то покупатель, купивший *кетчупы,* в 1,6 раза чаще выбирает *макаронные изделия*, нежели любой другой товар.

Так как левередж для данного правила *T*=0,14≠0, то правило значимо.

Так как улучшение для данного правила I=1,6>1, предсказать покупку макаронных изделий вероятнее, чем угадать случайно.

Из всех рассмотренных правил незначимое одно сыры макаронные изделия, так как у него C=60 % и L=0,96.

Пример нахождения ассоциативных правил в аналитической плат форме Deductor Studio Academic.

Рассмотрим механизм поиска ассоциативных правил на примере данных о продажах товаров в некоторой торговой точке. Данные находятся в файле

« Supermarket. txt». В таблице представлена информация по покупкам продуктов нескольких групп. Она имеет всего два поля «Номер чека» и «Товар». Необходимо решить задачу анализа потребительской корзины с целью последующего применения результатов для стимулирования продаж. Для этого производится поиск товаров, присутствие которых в транзакции влияет на вероятность наличия других товаров или комбинаций товаров.

Импортируем данные из текстового файла и представим в виде таблицы (при этом тип импортируемых данных – *строковый*):

Номер чека	Товар
160698	КЕТЧУПЫ, СОУСЫ, АДЖИКА
160698	МАКАРОННЫЕ ИЗДЕЛИЯ
160698	ЧАЙ
160747	МАКАРОННЫЕ ИЗДЕЛИЯ
160747	МЕД
160747	ЧАЙ
161217	КЕТЧУПЫ, СОУСЫ, АДЖИКА
161217	МАКАРОННЫЕ ИЗДЕЛИЯ

Рис. 42. Фрагмент базы данных транзакций

Для поиска ассоциативных правил запустим мастер обработки. В нем выберем тип обработки «Ассоциативные правила». На втором шаге мастера необходимо указать, какой столбец является идентификатором транзакции (чек), а какой элементом транзакции (товар).

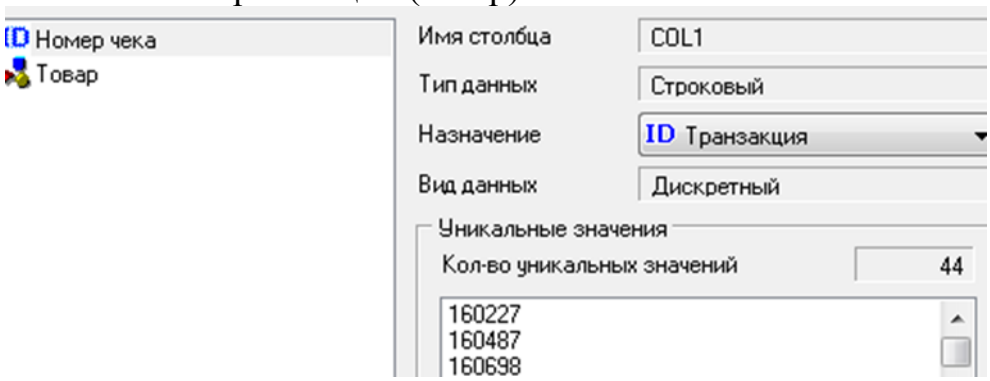


Рис. 43. Настройка назначения входных полей для решения задачи ассоциации

Следующий шаг позволяет настроить параметры построения ассоциативных правил: минимальную и максимальную поддержку, минимальную и максимальную достоверность, а также максимальную мощность множества.

По умолчанию в обработчике установлены следующие границы поддержки – 1 % и 20 %, и достоверности 40 % и 90 %, при которых для загруженных данных количество популярных наборов и ассоциативных правил равно 0. В этом случае следует увеличить максимальную поддержку.

Изменим верхнюю границу границы поддержки на 40 %.

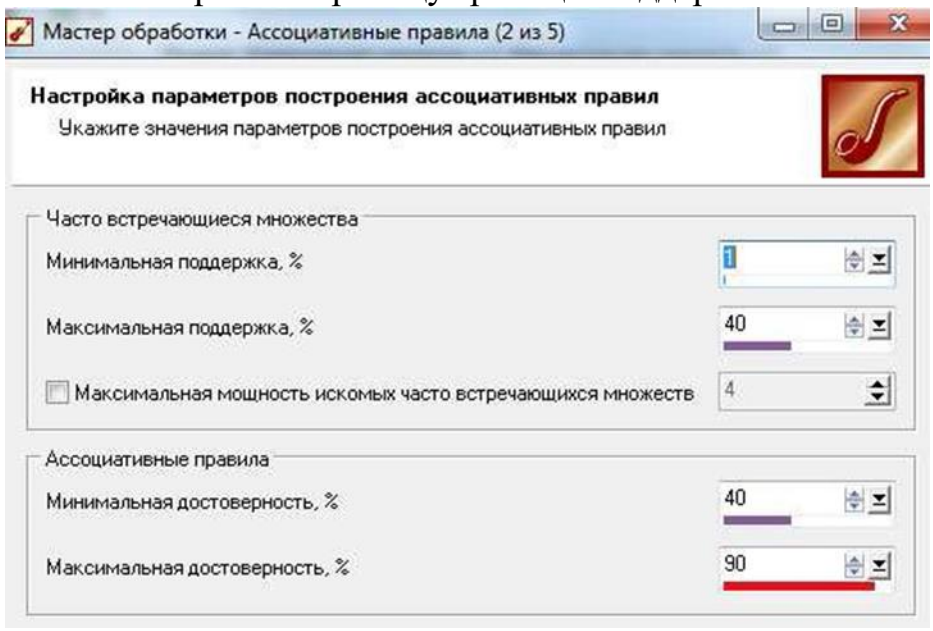


Рис. 44. Параметры алгоритма a priori

Следующий шаг позволяет запустить процесс поиска ассоциативных правил. На экране отображается информация о количестве множеств, количестве найденных правил, а также гистограмма распределения найденных часто встречающихся множеств по мощности.

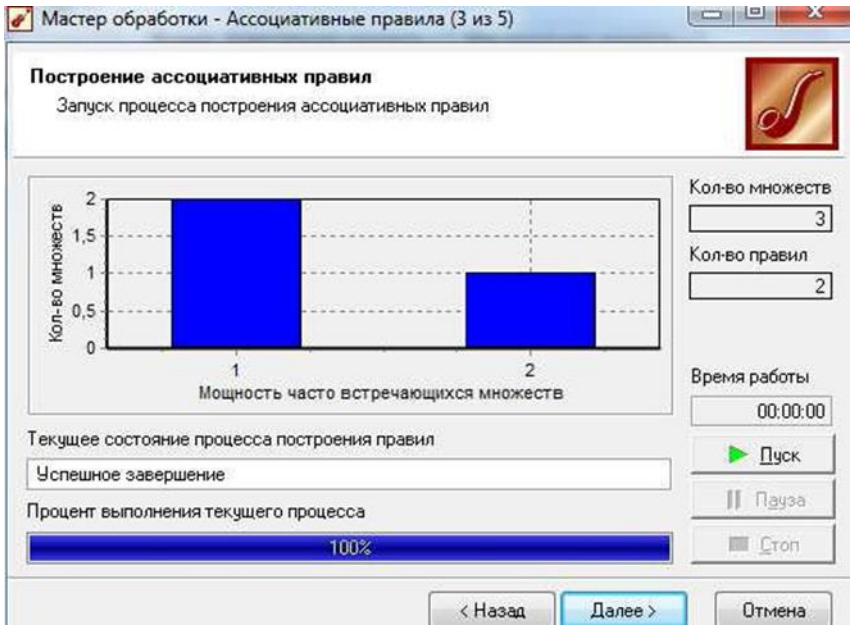


Рис. 45. Процесс выявления ассоциаций

После завершения процесса поиска полученные результаты можно посмотреть, используя появившиеся специальные визуализаторы «Популярные наборы», «Правила», «Дерево правил», «Что если». Популярные наборы – это множества, состоящие из одного и более элементов, которые наиболее часто встречаются в транзакциях одновременно. Насколько часто встречается множество в исходном наборе транзакций, можно судить по поддержке. Данный визуализатор отображает множества в виде списка.

Номер множества	Элемент	Поддержка		Мощность
		Кол-во	%	
1	ВАФЛИ	14	31,82	1
3	ВАФЛИ СУХАРИ	10	22,73	2
2	СУХАРИ	14	31,82	1

Рис. 46. Визуализатор «Популярные наборы»

Визуализатор «Правила» отображает ассоциативные правила в виде списка правил. Этот список представлен таблицей со столбцами: «номер правила», «условие», «следствие», «поддержка, %», «поддержка, количество», «достоверность».

g9

Номер правила	Условие	Следствие	Поддержка		Достоверность	Лифт
			Кол-во	%		
1	ВАФЛИ	СУХАРИ	10	22,73	71,43	2,245
2	СУХАРИ	ВАФЛИ	10	22,73	71,43	2,245
3	КЕТЧУПЫ, СОУСЫ	МАКАРОННЫЕ ИЗДЕЛИЯ	20	45,45	86,96	1,594

Рис. 47. Визуализатор «Правила»

Таким образом, эксперту предоставляется набор правил, которые описывают поведение покупателей. Например, если покупатель купил вафли, то он с вероятностью 71,4 % также купит и сухари.

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. Второй вариант дерева правил – дерево, построенное по следствию. Здесь на первом уровне располагаются узлы со следствием. Справа от дерева находится список правил, построенный по выбранному узлу дерева. Для каждого правила отображаются поддержка и достоверность. Если дерево построено по условию, то вверху списка отображается условие правила, а список состоит из его следствий. Тогда правила отвечают на вопрос, что будет при таком условии. Если же дерево построено по следствию, то вверху списка отображается следствие правила, а список состоит из его условий. Эти правила отвечают на вопрос, что нужно, чтобы было заданное следствие. Данный визуализатор отображает те же самые правила, что и предыдущий, но в более удобной для анализа форме.

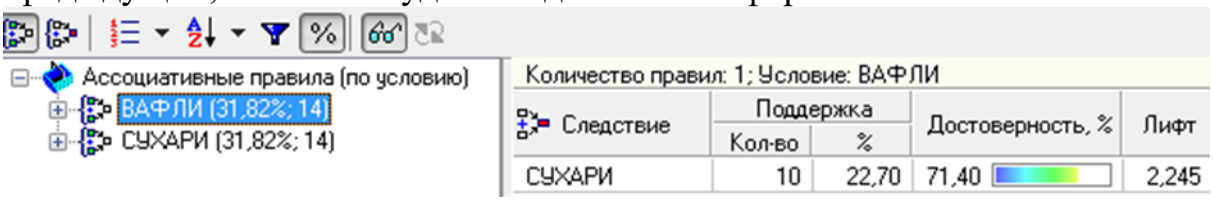


Рис. 48. Визуализатор «Дерево правил»

В данном случае правила отображены по условию. Тогда отображаемый в данный момент результат можно интерпретировать следующим образом:

Если покупатель приобрел вафли, то он с вероятностью 71 % также приобретет сухари.

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:

1. 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.

2. 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 то- вары, 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. Но ассоциативные правила с высокой поддержкой могут применяться для формализации хорошо известных правил, например в автоматизированных системах для управления процессами или персоналом.

3. 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.

4. Интерпретация ассоциативных правил. Дело в том, что ассоциативные правила сами по себе, как результат работы некоторого алгоритма, еще не готовы к использованию. Их нужно проинтерпретировать, т.е. понять, какие из ассоциативных правил представляют интерес, действительно ли правила отражают закономерности или, наоборот, являются артефактом.

Это требует тщательной работы аналитика и понимания предметной области, в которой решается задача ассоциации.

Все множество ассоциативных правил можно разделить на три вида:

1). Полезные правила содержат действительную информацию, которая ранее была неизвестна, но имеет логичное объяснение. Такие правила могут быть использованы для принятия решений, приносящих выгоду.

2). Тривиальные правила содержат действительную и легко объяснимую информацию, которая уже известна. Такие правила, хотя и объяснимы, но не могут принести какой-либо пользы, т.к. отражают известные законы в исследуемой области или результаты прошлой деятельности. При анализе рыночных корзин в правилах с самой высокой поддержкой и достоверностью окажутся товары-лидеры продаж. Практическая ценность таких правил крайне низка.

3). Непонятные правила содержат информацию, которая не может быть объяснена. Такие правила могут быть получены или на основе аномальных значений, или глубоко скрытых знаний. Напрямую такие правила нельзя использовать для принятия решений, т.к. их необъяснимость может привести к непредсказуемым результатам. Для лучшего понимания требуется дополнительный анализ.

Варьируя верхним и нижним пределами поддержки и достоверности, можно избавиться от очевидных и неинтересных закономерностей, можно найти действительно интересные и полезные правила.

В рассматриваемом примере, исходя из характера имеющихся данных, укажем границы поддержки – 13 % и 80 %, и достоверности – 60 % и 90 %. В результате количество популярных наборов и количество правил увеличится (рис. 49).

Рис. 49. Процесс выявления ассоциаций

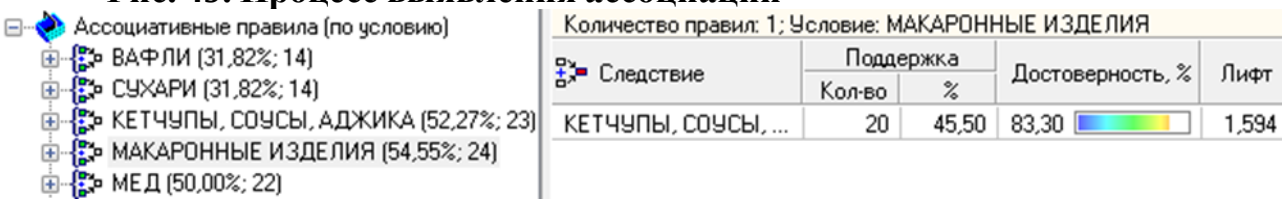


Рис.50. Визуализатор «Дерево правил»

Появились новые правила. Например, если покупатель приобрел макаронные изделия, то он с вероятностью 83,3 % также приобретет кетчупы и соусы. Это правило можно отнести к тривиальным, так как товары, присутствующие в нем, являются лидерами продаж (см. рис. 51), они имеют высокую поддержку.

Множеств: 7 из 30 | Фильтр: Минимальная мощность = 1; Максимальная мощность = 1

№	Элементы	Поддержка		Мощность
		Кол-во	%	
1	ВАФЛИ	14	31,82	1
2	КЕТЧУПЫ, СОУСЫ, АДЖИКА	23	52,27	1
3	МАКАРОННЫЕ ИЗДЕЛИЯ	24	54,55	1
4	МЕД	22	50,00	1
5	СУХАРИ	14	31,82	1
6	СЫРЫ	19	43,18	1
7	ЧАЙ	33	75,00	1

Рис. 51. Визуализатор «Популярные наборы» с фильтрацией по мощности

Наиболее удобным и оперативным инструментом использования ассоциативных правил является анализ «Что если». Внешний вид формы для проведения такого анализа представлен на рис. 52.

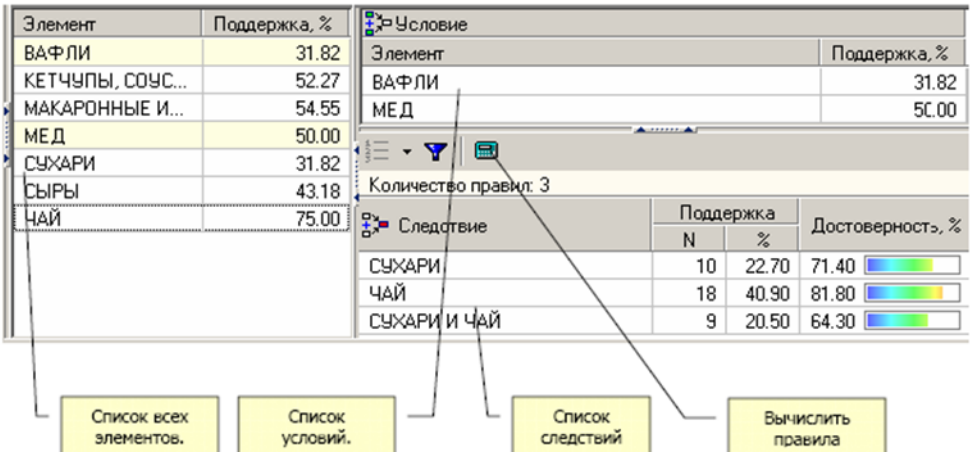


Рис. 52. Визуализатор «Что если»

Анализ «Что если» в ассоциативных правилах позволяет ответить на вопрос: «Что получим в качестве следствия, если выберем данные условия? Например, какие товары приобретаются совместно с выбранными товарами?» В окне слева расположен список всех элементов транзакций. Справа от каждого элемента указана поддержка «Сколько раз данный элемент встречается в транзакциях?» В правом верхнем углу расположен список элементов, входящих в условие. Это, например, список товаров, которые приобрел покупатель. Для них нужно найти следствие. Например, товары, приобретаемые совместно с ними, чтобы предложить человеку то, что он, возможно, забыл купить. В правом нижнем углу расположен список следствий. Справа от элементов списка отображается поддержка и достоверность.

Пусть необходимо проанализировать, что, возможно, забыл покупатель приобрести, если он уже взял вафли и мед? Для этого необходимо добавить в список условий эти товары (например, с помощью двойного щелчка мыши) и затем нажать на кнопку «Вычислить правила». При этом в списке следствий появятся товары, совместно приобретаемые с данными. В данном случае появятся «сухари», «чай», «сухари и чай». Возможно, покупатель забыл приобрести сухари или чай, или и то и другое.

Существующий в АП Deductor набор визуализаторов позволяет эксперту найти интересные, необычные закономерности, понять, почему так происходит и применить их на практике. Результаты анализа можно применить и для сегментации покупателей по поведению при покупках, и для анализа предпочтений клиентов, и для планирования расположения товаров в супермаркетах, кросс-маркетинге.

В данном примере найденные правила можно использовать для сегментации клиентов на два сегмента: клиенты, покупающие макаронные изделия и со- усы к ним, и клиенты, покупающие все к чаю. В разрезе анализа предпочтений можно узнать, что наибольшей популярностью в данном магазине пользуются чай, мед, макаронные изделия, кетчупы, соусы и аджика. В разрезе размещения товаров в супермаркете можно применить результаты

предыдущих двух анализов – располагать чай рядом с медом, а кетчупы, соусы и аджику – рядом с макаронными изделиями и т.д.

Карты Кохонена

Одной из задач, решаемой Data Mining, является кластеризация данных.

Кластеризация – это группировка объектов на основе данных, описывающих свойства объектов.

В Data Mining для кластеризации используются алгоритмы ***k-means*** и **сети Кохонена**.

k-means (иногда называемый алгоритмом *k*-средних) – наиболее популярный метод кластеризации.

Он разбивает множество элементов векторного пространства на заранее известное число кластеров *k*. Затем случайным образом выбираются начальные центры («семена») кластеров. Для каждой записи исходной выборки определяется ближайший к ней центр кластера. Производится вычисление центроидов (центров масс векторов) как среднего для значений каждого признака. Центроид становится центром кластера.

Действие алгоритма таково, что он стремится минимизировать среднеквадратичное отклонение на точках каждого кластера от центроида:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \mu_i\right)^2$$

, где *k* – число кластеров, S_i – полученные кластеры;

μ_i – центры масс векторов S_i, $x_j \in S_i$

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Сети Кохонена предназначены для решения задач кластеризации. По становка задачи кластеризации. Дано: $X[n \times m]$, n – количество объектов, m – количество признаков; или Дано: множество векторов $X = \{X^1, X^2, ..., X^p, ..., X^n\}$, где $X^p = \{X_1{}^p, ..., X_m{}^p\}$

Заранее известно количество *k* будущих кластеров, но само разбиение на кластеры не известно.

Необходимо: 1) найти *k*-ядер (центроидов) кластеров, т.е. вектор

$C^k = \{C^1, C^2, ..., C^l, ..., C^k\}$, $C^l C_1{}^l, ..., C_m{}^l\}$

2) разбить множество векторов X на k кластеров: $X^p X_1{}^p, ..., X_m{}^p$ на $K\{C^l\}$.

Иными словами, задача заключается в том, чтобы найти некоторую функцию $I(p)$, которая позволяет определить номер кластера I по номеру входного объекта P, что и является решением задачи кластеризации. При этом эта функция должна удовлетворять следующему критерию: минимизации рас стояний между объектами и ядрами кластеров.

$$D = \sum_{p=1}^{n} \sum_{l=1}^{k} d(X^p, C^l) \to \min$$

Алгоритм кластеризации:

1) задаются начальные значения ядер кластеров $\{C^l\}$.

Способы задания: а) случайными числами; б) некоторыми равными числами; в) эвристическими правилами (метод главных компонент), основанными на некоторой закономерности.

2) Фиксируются ядра кластеров $\{C^l\} \to \text{const}$. Находим $l(p)$ – разбиение $\{X^p\}$ с целью $D \to \min$.

3) Корректируется $\{C^l\}$ таким образом, чтобы в пределах каждого кластера суммарное расстояние между объектом этого кластера и ядром было минимальным.

$$D_l = \sum_{p \in S_l} d(X^p, C^l) \to \min,$$

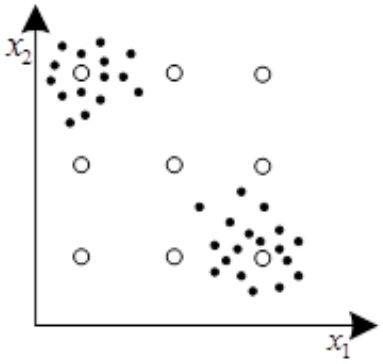
результат $\{C^l\}$.

Алгоритм обучения без учителя для сетей Кохонена

I. Инициализация весов:

1) малые случайные числа;

2) равномерная инициализация.



Для случая $n=k$ обучение не проводится. Для нормальной работы $n>>k$.

3) специальные методы инициализации.

II. Обучающая выборка: $X = \{X^p\} = \{X^1, X^2, ..., X^n\}$

III. Рассчитывается l_0 – прямой проход по сети.

IV. Корректировка весов сети Кохонена

1) Традиционный способ: корректируются веса только нейрона- победителя:

$W_{l_0}^{\text{нов}} = W_{l_0}^{\text{стар}} + \alpha(X^p - W_{l_0}^{\text{стар}})$, где α – скорость обучения ($0<\alpha<1$) монотонно убывает.

Смысл корректировки весов заключается в попытке приблизить координаты нейрона- победителя к входному образу вектора. Обучение продолжается до тех пор, пока координаты весов не перестанут изменяться. По окончанию обучения каждый нейрон отвечает за определение кластера.

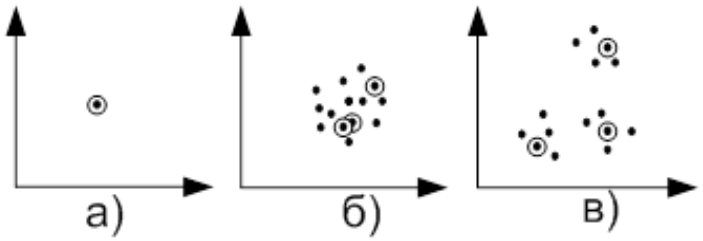
2) метод выпуклой комбинации;

3) модифицированные алгоритмы.

В процессе обучения вектора расходятся от начальных точек к своим истинным значениям. При этом особенность обучения заключается в следующем: в процессе движения к своим истинным значениям обучающие вектора захватывают ядра

кластеров пропорционально своей плотности распределения.

Графическая интерпретация:



Трудности при использовании алгоритма это выбор исходных центров кластеров и необходимость знать число кластеров заранее. Также алгоритм плохо работает в случае, когда один кластер значительно больше остальных и они находятся близко друг от друга. Возникает эффект «расщепления» большого кластера.

Самоорганизующаяся карта Кохонена строится на основе нейросети Кохонена.

Сеть Кохонена является одной из разновидностей соревновательной (конкурентной) нейронной сети с обучением без учителя. Ее основной целью является преобразование сложных многомерных данных в более простую структуру малой размерности. Сеть состоит из узлов, которые объединяются в кластеры.

Сигнал в сеть Кохонена поступает сразу на все нейроны, веса соответствующих синапсов интерпретируются как координаты положения узла, и выходной сигнал формируется по принципу «победитель забирает все» — то есть ненулевой выходной сигнал имеет нейрон, ближайший (в смысле весов синапсов) к подаваемому на вход объекту. В процессе обучения веса синапсов настраиваются таким образом, чтобы узлы решетки «располагались» в местах локальных сгущений данных, то есть описывали кластерную структуру облака данных; с другой стороны – связи между нейронами соответствуют отношениям соседства между соответствующими кластерами в пространстве признаков.

Карта Кохонена состоит из ячеек, центром каждой из которых является нейрон выходного слоя. В ячейку попадает один или несколько объектов, векторы весов которых оказываются ближе к вектору весов данного нейрона.

Изначально самоорганизующаяся карта представляет сетку из узлов, соединенных между собой. Кохонен рассматривал два варианта соединения узлов в прямоугольную и гексагональную сетку. Отличие состоит в том, что в прямоугольной сетке каждый узел соединен с четырьмя соседними, а в гексагональной – с шестью ближайшими узлами. Шестиугольные ячейки более корректно отражают расстояния между объектами на карте, т.к. у них равны расстояния между центрами смежных ячеек.

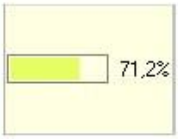
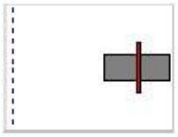
Самоорганизующаяся карта Кохонена является методом проецирования многомерного пространства в пространство с более низкой размерностью (чаще всего двухмерное). Двухмерная карта имеет цветную раскраску. Интенсивность цвета в определенной точке зависит от данных, которые туда попали. Ячейки, в которые попали элементы с минимальными значениями или не попало вообще ни одной записи, окрашиваются синим цветом, а ячейки с максимальными значениями окрашиваются красным. Раскраска карты позволяет оценить и результаты кластеризации. Если ячейки с одинаковой расцветкой образуют обособленные области, то результаты кластеризации хорошие. Если ячейки разных цветов разбросаны вперемешку по всей карте, то результаты плохие.

Недостатком карты Кохонена является эвристический характер метода. Начальная инициализация

карты, т.е. задание весов нейронов, является произвольной, что может привести к потере однозначности результата. Если обучать карту Кохонена несколько раз, то получаются непохожие итоги.

Кластеры отображают группы векторов, расстояние между которыми меньше, чем расстояние до соседних групп. Иными словами, все элементы карты, попавшие в область одного цвета (кластер), имеют сходные признаки.

Результаты кластеризации алгоритмом Кохонена можно увидеть не только на карте, но и на специальном визуализаторе *Профили кластеров*. Это кросстаблица 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.

Пример использования карт Кохонена для решения задачи сегментации данных по продажам некоторой компании на кластеры

В Deductor Studio сети и карты Кохонена реализованы в обработчике *Карта Кохонена*, где содержатся сам алгоритм Кохонена и специальный *визуализатор Карта Кохонена*.

Алгоритм Кохонена применяется к сети Кохонена, состоящей из ячеек, упорядоченных на плоскости. По умолчанию размер карты равен 16 х 12, что соответствует 192 ячейкам (нейронам).

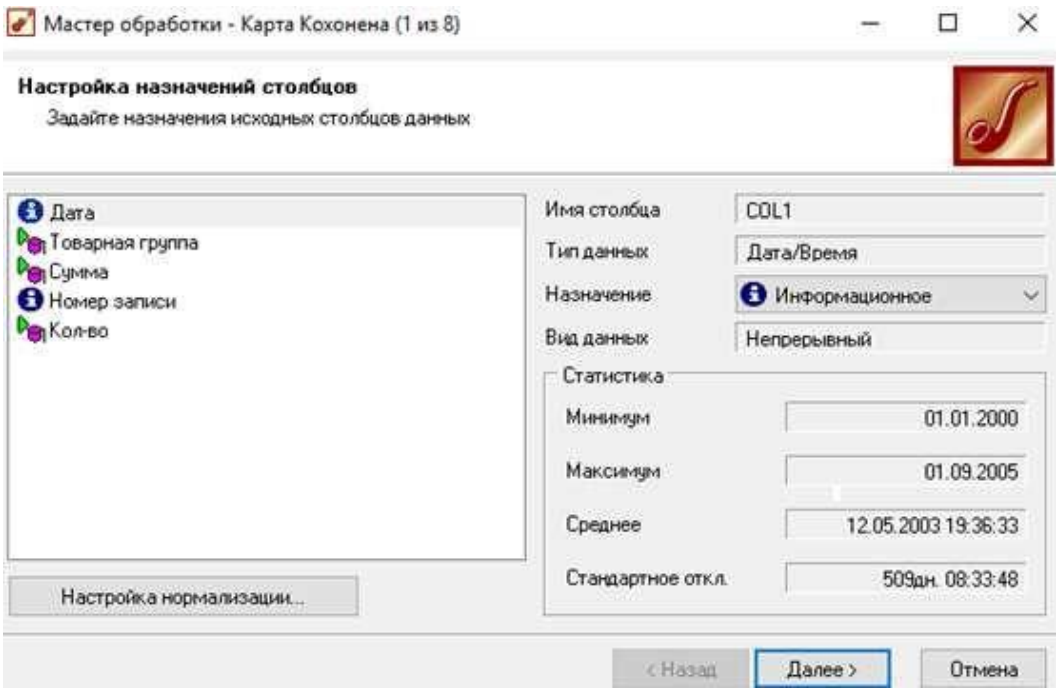
Не существует строгих правил для выбора числа нейронов в карте Кохонена. Каждый раз аналитик должен сам определять (возможно, методом проб и ошибок) оптимальное количество нейронов, исходя из решаемой задачи и особенностей данных. Например, если разброс значений признаков в обучающей выборке сильный (то есть векторы объектов в пространстве признаков разрежены), то, возможно, следует уменьшить число нейронов карты, чтобы

избежать большого количества пустых ячеек. И наоборот, если векторы объектов расположены в пространстве признаков плотно, то для получения лучших результатов можно увеличить число нейронов.

Ограничения использования карт Кохонена в Deductor: в Deductor Studio алгоритм Кохонена ориентирован на работу преимущественно с числовыми типами данных, а также с упорядоченными (ординальными) типами. Обработка данных в полях, значения которых нельзя упорядочить, будет приводить к некорректным результатам. Упорядочивание ординальных типов осуществляется на вкладке *Настройка нормализации*.

Импортируем в Deductor набор данных из файла *Продажи 2*. Запустим мастер обработки и выберем узел *Карта Кохонена*.. В качестве входных параметров используются:

- Товарная группа;
- Сумма;
- Количество.



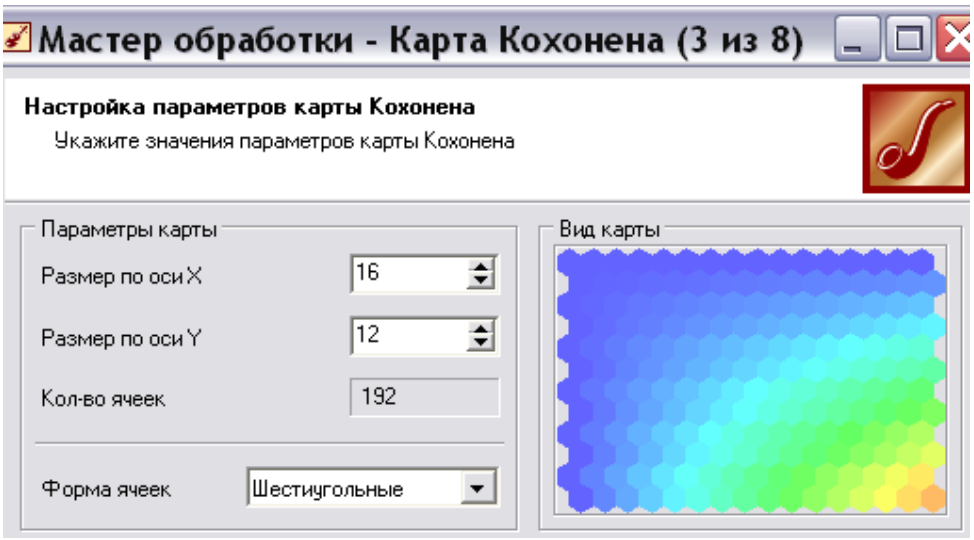
Настройка назначений полей

На этой же вкладке при нажатии кнопки « Настройка нормализации» откроется окно, где можно задать значимость каждого входного поля. Оставим значимость одинаковой для всех полей без изменений.

Замечание. В обработчике «карта Кохонена» допускается задавать и выходное поле. Несмотря на такое название, это поле не будет принимать участие в алгоритме Кохонена, однако после построения по этому полю будет собрана статистика. Это открывает возможности для решения картой Кохонена задачи классификации или регрессии.

Затем разбиваем входящее множество на обучающее и тестовое. Поскольку любой метод кластеризации, в том числе и алгоритм Кохонена, субъективен, смысл в выделении отдельного, тестового множества, как правило, отсутствует. Оставим в обучающем множестве 100 % записей.

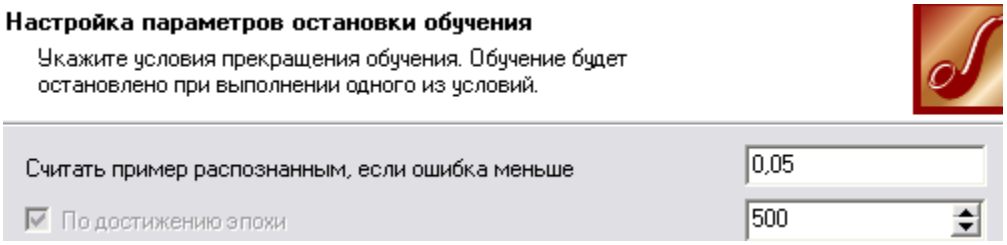
На третьей вкладке задаются размер и форма карты Кохонена. Согласимся с настройками по умолчанию – шестиугольные ячейки, размер 16x12.



Параметры будущей карты Кохонена

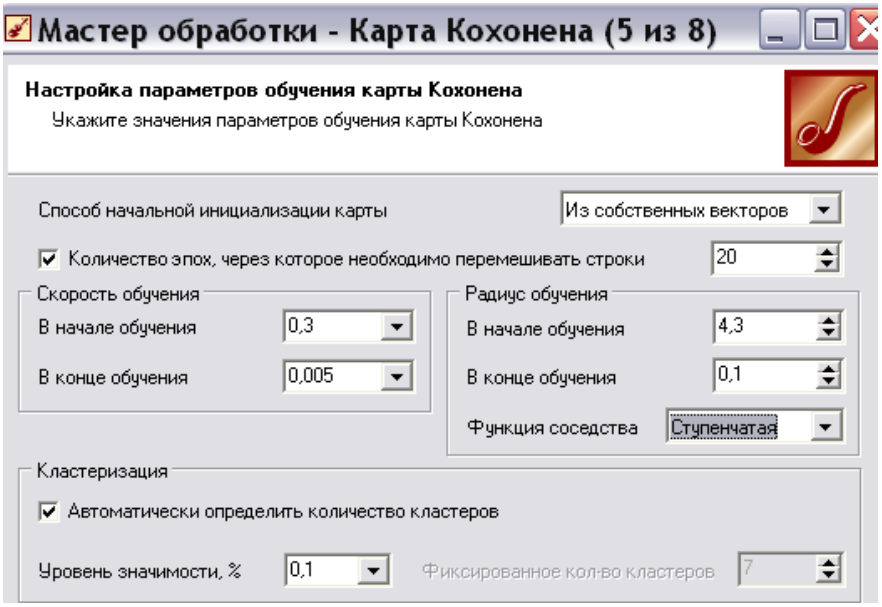
Замечание: Обычно строят несколько моделей с разными настройками. Например, карта с увеличенным масштабом 24x18 может оказаться лучше, так как позволит « разглядеть» кластер, который не удавалось обнаружить при размере карты 16x12. Поэтому здесь универсальных рецептов нет. Понять, лучше или хуже карта Кохонена, можно, только сравнив ее с картами при других настройках, сравнив матрицы ошибок квантования и матрицы плотности попадания.

Важным этапом является настройка параметров остановки обучения. Устанавливаем параметры остановки обучения и устанавливаем значение эпохи, по достижению которой обучение будет прекращено.



Настройка параметров остановки обучения

Наконец, на последнем шаге, предшествующем обучению, настраиваются параметры обучения карты Кохонена



Параметры обучения карты Кохонена

Здесь задаются следующие опции:

Способ начальной инициализации карты определяет, как будут установлены начальные веса нейронов карты. Удачно выбранный способ инициализации может существенно ускорить обучение и привести к получению более качественных результатов.

Доступны три варианта:

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

При выборе способа начальной инициализации следует руководствоваться следующей информацией:

- объемом обучающей выборки;
- количеством эпох, отведенных для обучения;
- размером карты.

Между указанными параметрами и способом начальной инициализации существует много зависимостей. Выделим несколько главных.

1. Если объем обучающей выборки значительно (в 100 и более) превышает число ячеек карты и время обучения не играет первоочередной роли, то лучше выбрать *инициализацию случайными значениями*, так как это даст меньшую вероятность попадания в локальный минимум ошибки кластеризации.

2. Если объем обучающей выборки не очень велик, время обучения ограничено или необходимо уменьшить вероятность появления после обучения пустых ячеек, в которые не попало ни одного экземпляра обучающей выборки, то следует использовать *инициализацию примерами из обучающего множества*.

3. *Инициализацию из собственных векторов* можно использовать при любом стечении обстоятельств. Единственное замечание: вероятность появления пустых ячеек после обучения выше, чем при инициализации примерами из обучающего множества. Именно этот способ лучше выбирать при первом ознакомлении с данными.

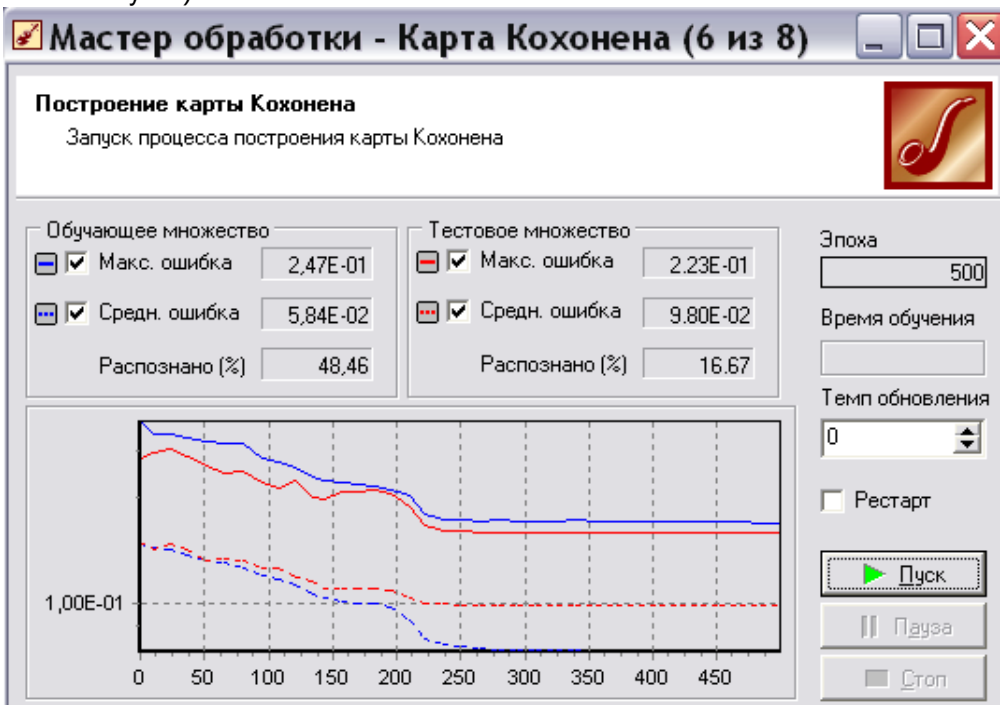
Скорость обучения – задается скорость обучения в начале и в конце обучения карты Кохонена. Рекомендуемые значения: 0,1–0,3 в начале и 0,05– 0,005 в конце обучения.

Радиус обучения – задается радиус обучения в начале и в конце обучения карты Кохонена. Радиус в начале должен быть достаточно большой – примерно половина или меньше размера карты (максимальное линейное расстояние от любого нейрона до другого любого нейрона), а в конце – достаточно малым, примерно 1 или меньше. Начальный радиус в Deductor подбирается автоматически в зависимости от размера карты.

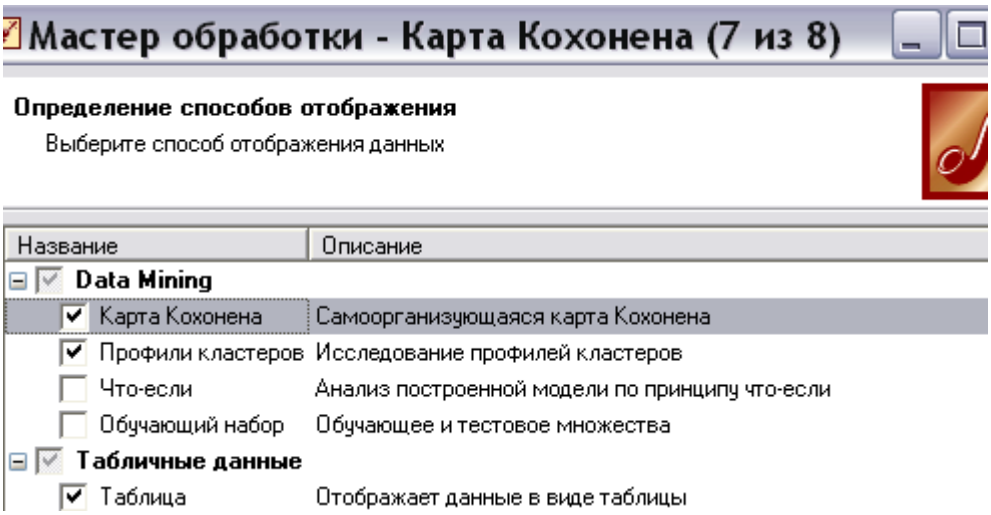
В этом же блоке задается **Функция соседства**: Гауссова или Ступенчатая. Если функция соседства *Ступенчатая*, то «соседями» для нейрона-победителя будут считаться все нейроны, линейное расстояние до которых не больше текущего радиуса обучения. Если используется Гауссова функция соседства, то «соседями» для нейрона-победителя будут считаться все нейроны карты, но в разной степени полноты. При использовании Гауссовой функции соседства обучение проходит более плавно и равномерно, так как одновременно изменяются веса всех нейронов, что может дать немного лучший результат, чем если бы использовалась ступенчатая функция. Однако времени, необходимого на обучение, требуется немного большее по причине того, что на каждой эпохе корректируются все нейроны.

Кластеризация – в этой области указываются параметры алгоритма *kmeans* (*G-means*), который запускается после алгоритма Кохонена для группировки ячеек карты. Здесь нужно только определить, позволить алгоритму автоматически определить число кластеров (*G-means*) или сразу зафиксировать его (*k-means*). Следует знать, что автоматически подбираемое число кластеров не всегда приводит к желаемому результату – число кластеров может предлагаться слишком большим, поэтому рассчитывать на эту опцию можно только на этапе исследования данных.

После окончания ввода параметров запускаем процесс обучения– необходимо нажать на кнопку **Пуск** и дождаться окончания процесса обучения. В открывшемся окне можно будет увидеть динамику процесса обучения карты Кохонена. По умолчанию алгоритм делает 500 итераций (эпох). Если предварительно установить флаг **Рестарт**, то веса нейронов будут проинициализированы согласно выбранному на предыдущем шаге способу инициализации, иначе обучение начнется с текущих весовых коэффициентов (это справедливо только при повторной настройке узла).

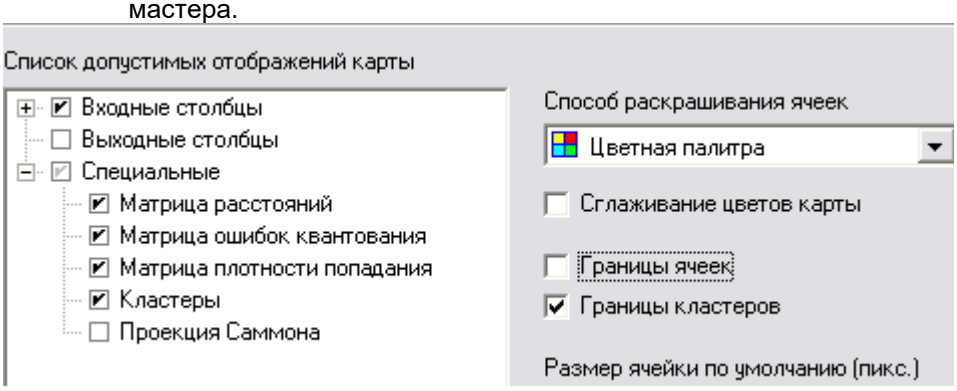


Обучение карты Кохонена



Выбор способа отображения результатов кластеризации

К обученной сети Кохонена предлагаются специализированные визуализаторы – **Карта Кохонена** и **Профили кластеров**. Параметры карты задаются на специальной вкладке мастера.



Настройки визуализатора карты Кохонена

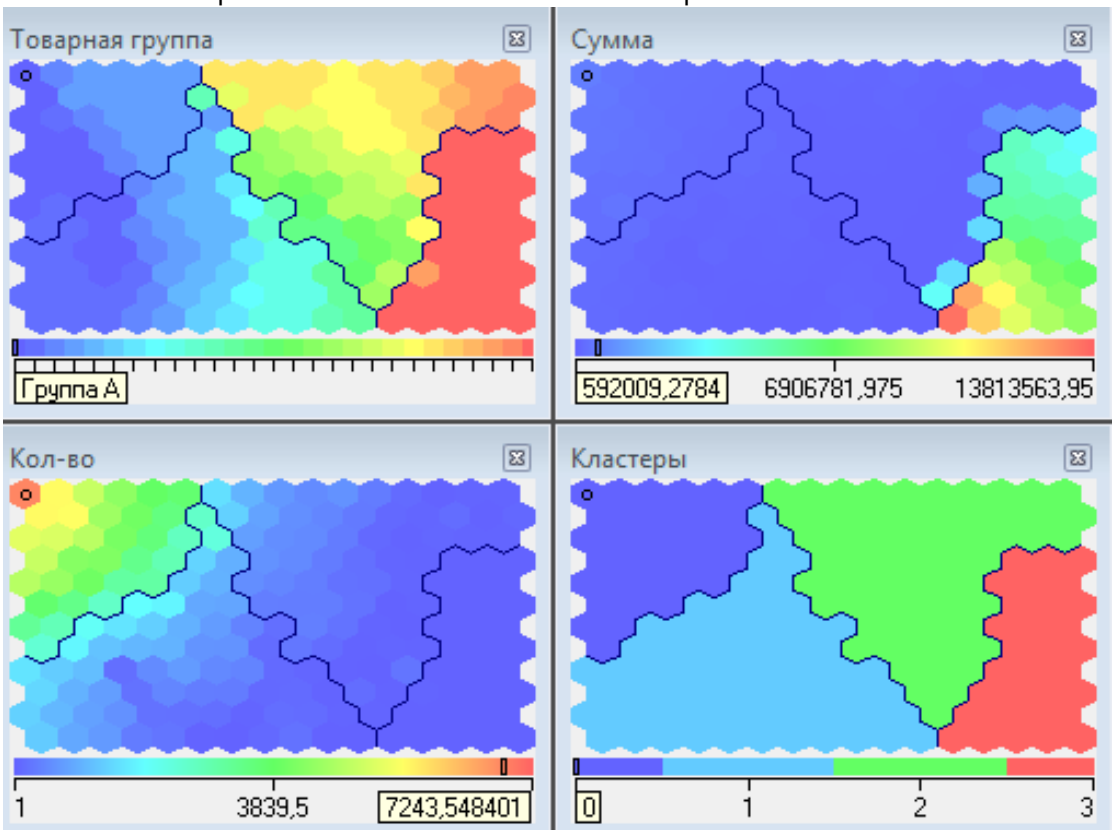
Список допустимых отображений карты содержит три группы – входные поля, выходные поля и специальные. Последние не связаны с каким-либо полем набора данных, а служат для анализа всей карты.

- *Матрица расстояний* применяется для визуализации структуры кластеров, полученных в результате обучения карты. Большое значение говорит о том, что данный нейрон сильно отличается от окружающих и относится к другому классу.
- *Матрица ошибок квантования* отображает среднее расстояние от расположения примеров до центра ячейки. Расстояние считается как евклидово расстояние. Матрица ошибок квантования показывает, насколько хорошо обучена сеть Кохонена. Чем меньше среднее расстояние до центра ячейки, тем ближе к ней расположены примеры и тем лучше модель.
- *Матрица плотности попадания* отображает количество объектов, попавших в ячейку.
- *Кластеры* – ячейки карты Кохонена, объединенные в кластеры алгоритмом *k-means*.
- *Проекция Саммона* – матрица, являющаяся результатом

проецирования многомерных данных на плоскость.

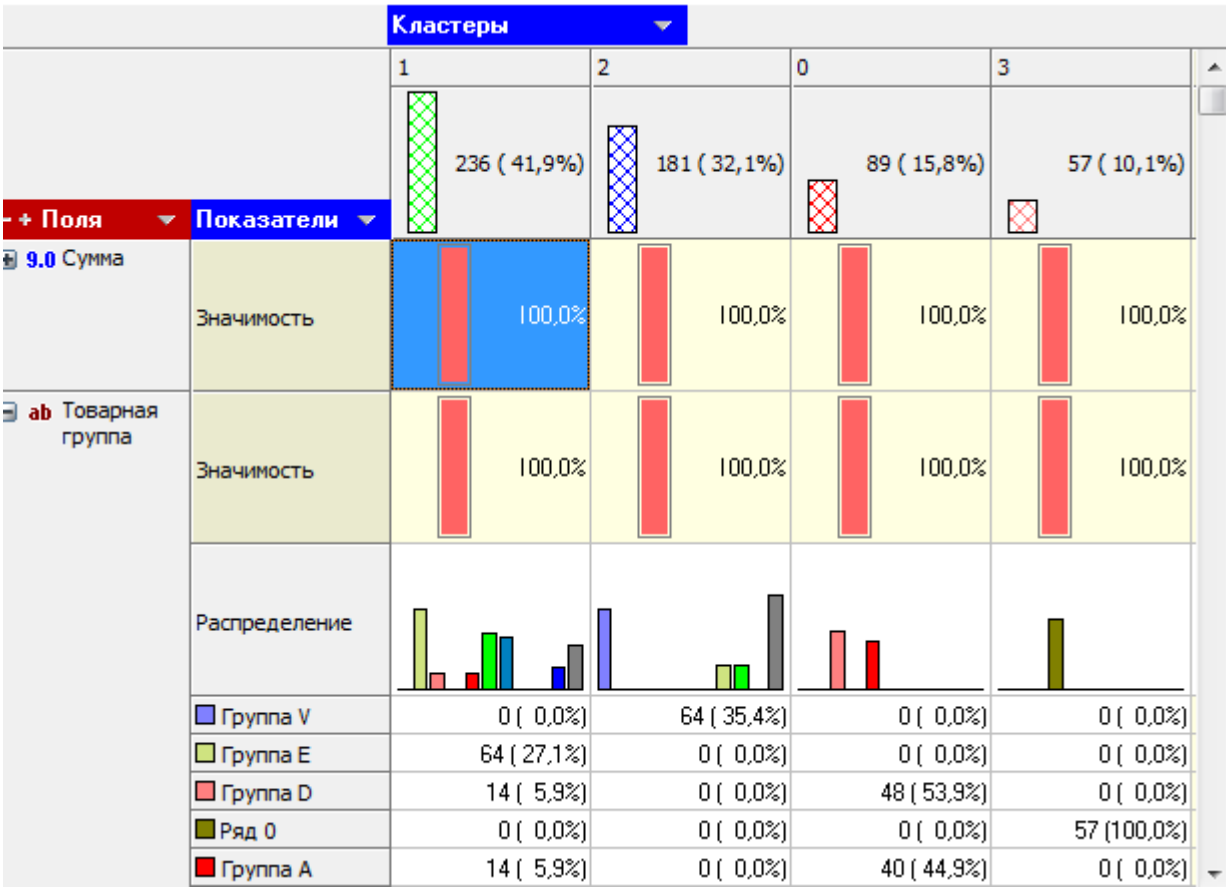
Выбрав по окончании обучения в списке визуализаторов *карту Кохонена*, увидим, что в результате кластеризации получилось четыре кластера.

При анализе карт входов используем сразу несколько карт (это зарплата, расход, доход). Например, на одной из карт выделяем область с наибольшими значениями показателя (выделена красным цветом) и изучаем эти же нейроны на других картах. При работе с картой доступны операции, выполняемые с помощью кнопок на панели инструментов визуализатора или контекстного меню, вызываемого правой кнопкой мыши в любом окне карты.



Карта Кохонена для сегментации продаж по товарным группам

Результат анализа раскраски карт соответствующих показателей и их статистических характеристик, используя визуализатор *Профили кластеров* позволил дать каждому кластеру описание.



Визуализатор *Профили кластеров*

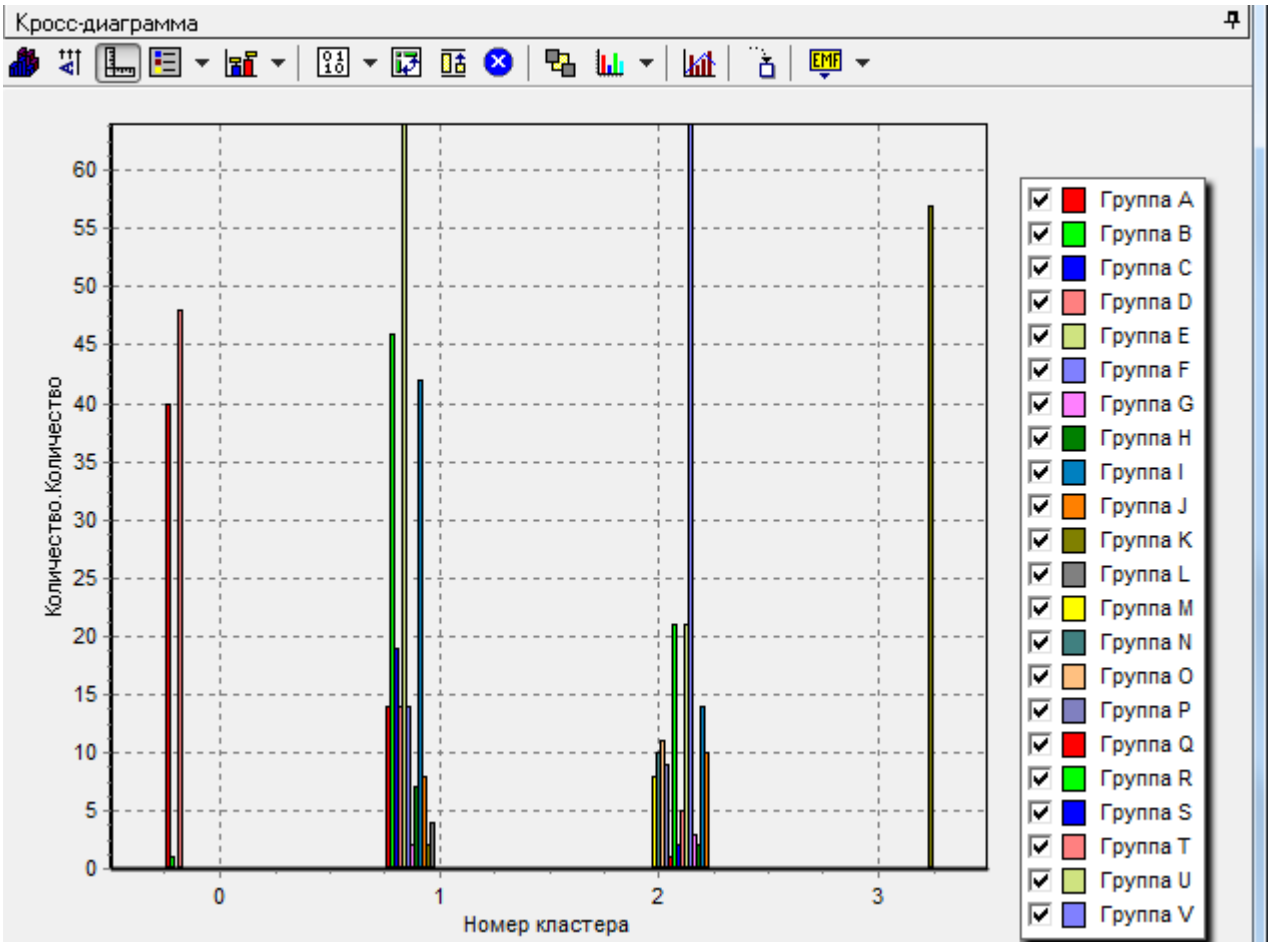
Кластер	Средняя прибыль	Колво	Колво, %	Прибыль	Процент от общей
0	319573,2	89	15,8	28442014,8	6
1	79673	236	41,9	18802828	4
2	30329,5	181	32,1	5489639,5	2
3	6900086	57	10,1	393304924,8	88
Итого		563		446039407,1	100

Кластер 0. (Мощность 15,8 %). Содержит в основном товары товарных групп A и D, дают компании 6 % прибыли.

Кластер 1. (Мощность 41,9 %). Кластер, который содержит товары почти всех товарных групп, но преобладают товары групп B, E, I. Продажа товаров этого кластера приносит компании 4 % прибыли.

Кластер 2. (Мощность 32,1 %). В этом кластере присутствуют товары разных товарных групп, но преобладают товары группыV. Товары, входящие в данный кластер, приносят самую меньшую прибыль компании (1 %).

Кластер 3. (Мощность 10,1 %). Товарная группа, по которой заключаются самые крупные сделки в компании. Это чистый сегмент – нет товаров других групп. По кластеру было совершено немного продаж, но зато сделки были самые крупные, которые приносили максимальную прибыль компании (88 %).



Кросс-диаграмма (количество товаров по каждому кластеру)

Пример решения задачи

Программное обеспечение, позволяющее работать с картами Кохонена, сейчас представлено множеством инструментов. Это могут быть как инструменты, включающие только реализацию метода самоорганизующихся карт, так и нейропакеты с целым набором структур нейронных сетей, среди которых - и карты Кохонена; также данный метод реализован в некоторых универсальных инструментах анализа данных.

Пусть имеется база данных коммерческих банков с показателями деятельности за текущий период. Необходимо провести их кластеризацию, т.е. выделить однородные группы банков на основе показателей из базы данных, всего показателей - 21.

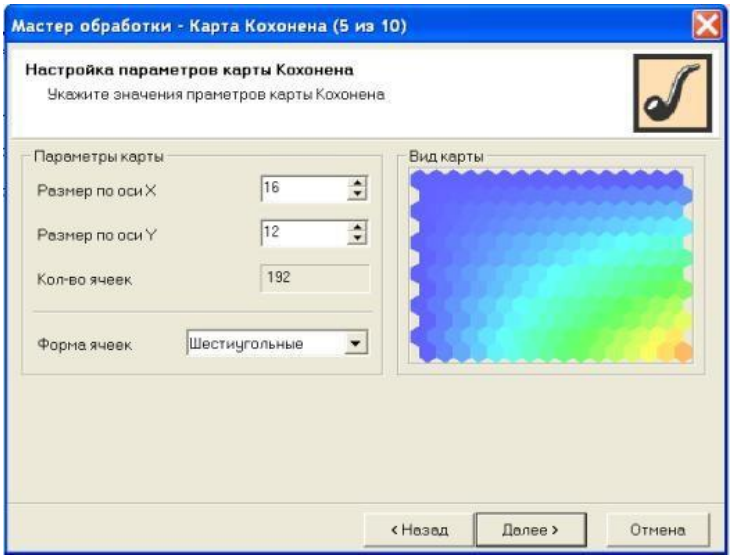
Исходная таблица находится в файле "banks.xls". Она содержит показатели деятельности коммерческих банков за отчетный период.

Сначала импортируем данные из xls-файла в среду аналитического пакета.

На первом шаге мастера запускаем мастер обработки и выбираем из списка метод обработки " Карта Кохонена". Далее следует настроить назначения столбцов, т.е. для каждого столбца выбрать одно из назначений: входное, выходное, не используется и информационное. Укажем всем столбцам, соответствующим показателям деятельности банков, назначение " Входной". "Выходной" не назначаем.

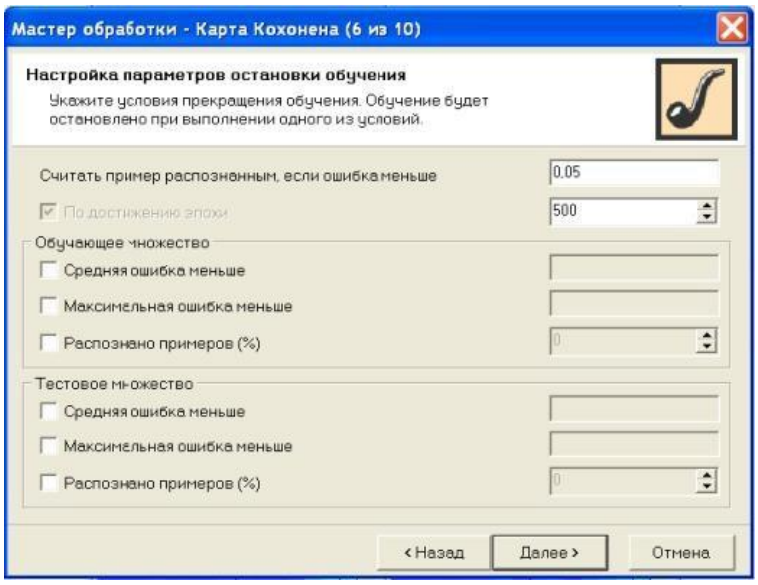
Следующий шаг предлагает разбить исходное множество на обучающее, тестовое и валидационное. По умолчанию, программа предлагает разбить множество на обучающее - 95% и тестовое - 5%.

На шаге № 5 предлагается настроить параметры карты: количество ячеек по X и по Y их форму (шестиугольную или четырехугольную).



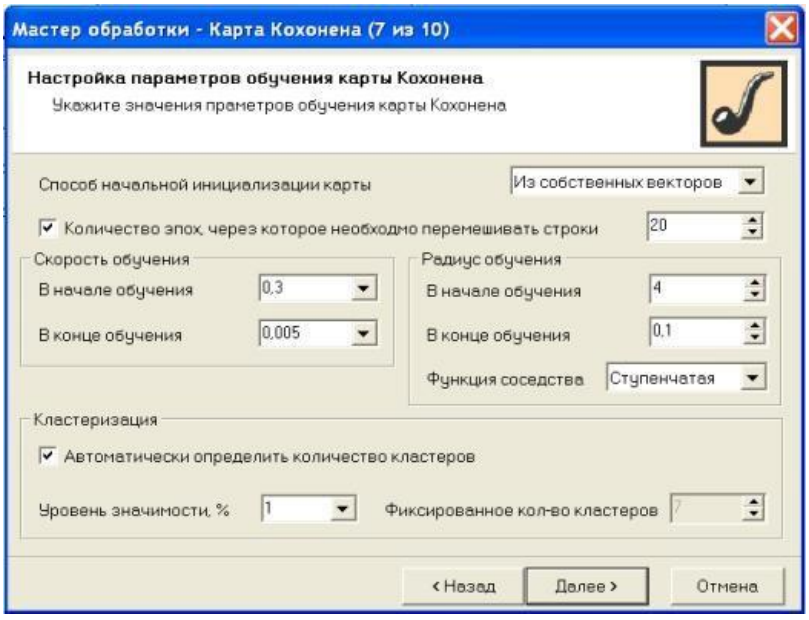
Шаг № 5 "Настройка параметров карты Кохонена"

На шестом шаге " Настройка параметров остановки обучения", проиллюстрированном на рис. 12.5, устанавливаем параметры остановки обучения и устанавливаем эпоху, по достижению которой обучение будет прекращено.



Шаг № 6 "Настройка параметров остановки обучения"

На седьмом шаге, настраиваются другие параметры обучения: способ начальной инициализации, тип функции соседства. Возможны два варианта кластеризации: автоматическое определение числа кластеров с соответствующим уровнем значимости и фиксированное количество кластеров (определяется пользователем). Поскольку нам неизвестно количество кластеров, выберем автоматическое определение их количества.

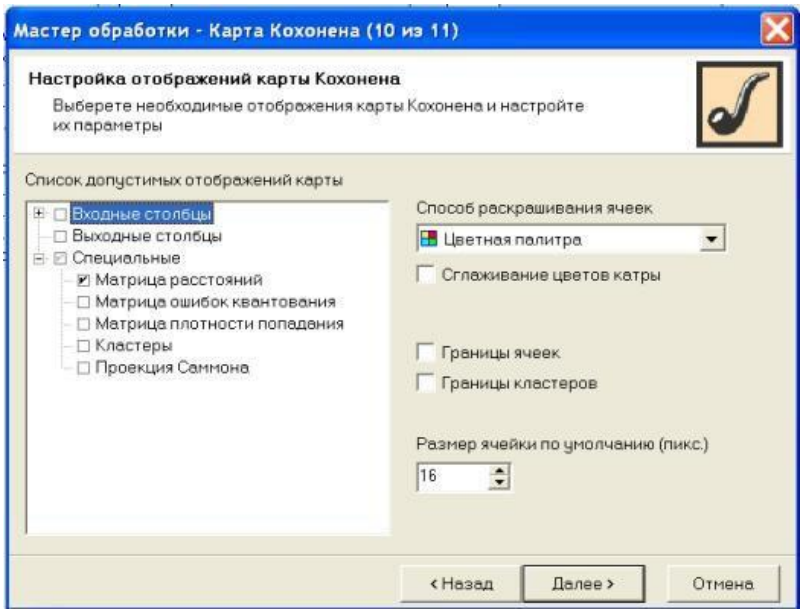


Шаг № 7 "Настройка параметров остановки обучения"

На восьмом шаге запускаем процесс обучения сети - необходимо нажать на кнопку " Пуск" и дождаться окончания процесса обучения. Во время обучения можем наблюдать изменение количества распознанных примеров и

текущие значения ошибок. Этот процесс аналогичен тому, что мы рассматривали при обучении нейронных сетей в предыдущей лекции.

По окончании обучения в списке визуализаторов выберем "Карту Кохонена" и визуализатор "Что-если". На последнем шаге настраиваем отображения карты Кохонена.

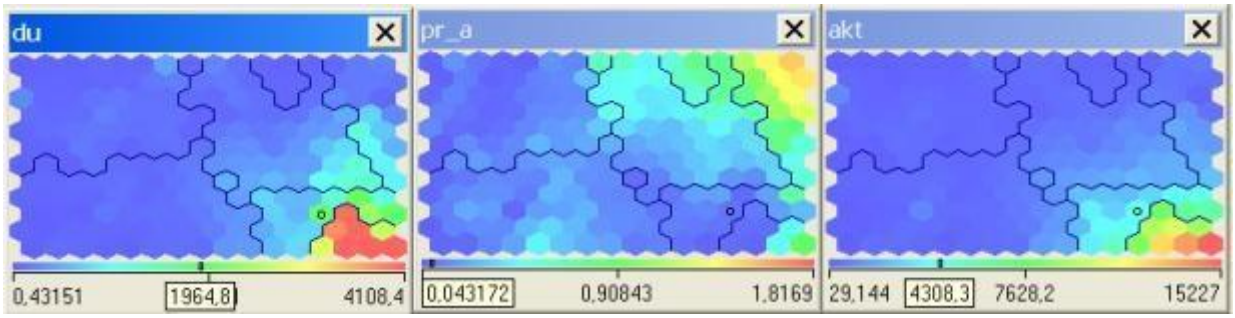


Шаг № 10 Настройка отображений карты Кохонена

Укажем отображения всех входных, выходных столбцов, кластеров, а также поставим флажок "Границы кластеров" для четкого отображения границ.

Карты входов

При анализе карт входов рекомендуют использовать сразу несколько карт. Исследуем фрагмент карты, состоящий из карт трех входов:



Карты трех входов

На одной из карт выделяем область с наибольшими значениями показателя. Далее имеет смысл изучить эти же нейроны на других картах.

На первой карте наибольшие значения имеют объекты, расположенные в правом нижнем углу. Рассматривая одновременно три карты, мы можем сказать, что эти же объекты

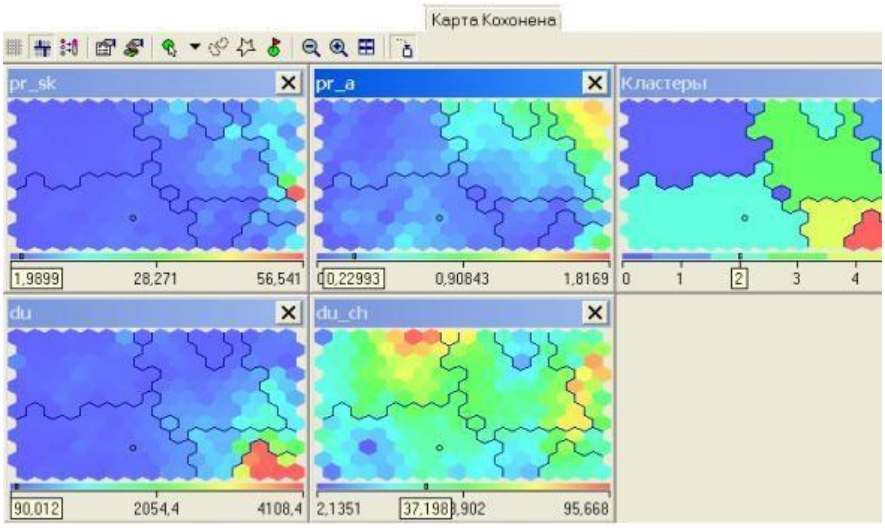
имеют наибольшие значения показателя, изображенного на третьей карте. Также по раскраске первой и третьей карты можно сделать вывод, что существует взаимосвязь между этими показателями.

Также мы можем определить, например, такую характеристику: кластер, расположенный в правом верхнем углу, характеризуется низкими значениями показателей du (депозиты юридических лиц) и akt (активы банка) и высокими значениями показателей pr_a (прибыльность активов).

Эта информация позволяет так охарактеризовать кластер, находящийся в правом верхнем углу: это банки с небольшими активами, небольшими привлеченными депозитными средствами от юридических лиц, но с наиболее прибыльными активами, т.е. это группа небольших, но наиболее прибыльных банков.

Это лишь фрагмент вывода, который можно сделать, исследуя карту.

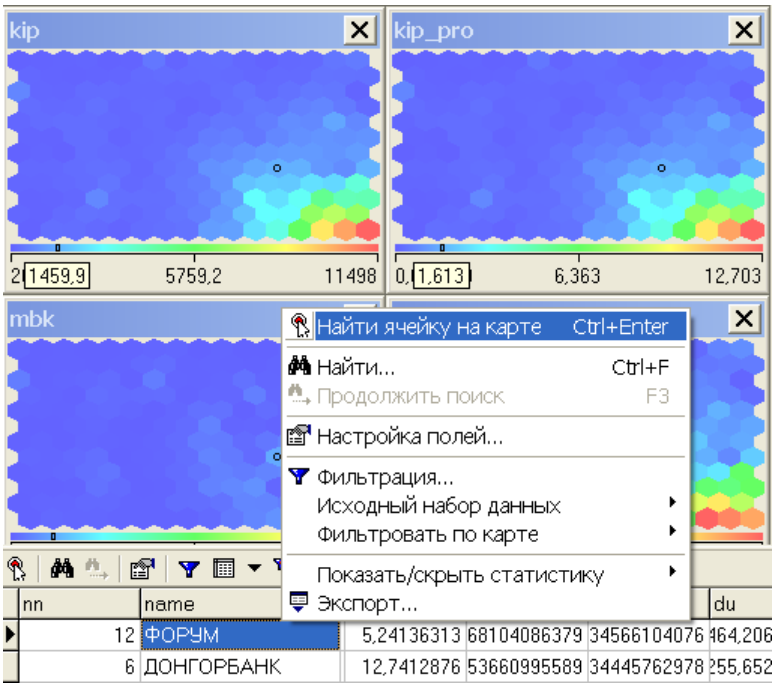
На следующем рисунке приведена иллюстрация карт входов и выходов, последняя - эта карта кластеров. Здесь мы видим несколько карт входов (показателей деятельности банков) и сформированные кластеры, каждый из которых выделен отдельным цветом.



Карты входов и выходов

Для нахождения конкретного объекта на карте необходимо нажать правой кнопкой мыши на исследуемом объекте и выбрать пункт "Найти ячейку на

карте". Выполнение этой процедуры показано на следующем рисунке. В результате мы можем видеть как сам объект, так и значение того измерения, которое мы просматриваем. Таким образом, мы можем оценить положение анализируемого объекта, а также сравнить его с другими объектами.



Ячейка на карте

В результате применения самоорганизующихся карт многомерное пространство входных факторов было представлено в двухмерном виде, в котором его достаточно удобно анализировать.

Банки были классифицированы на 7 групп, для каждой из которых возможно определение конкретных характеристик, исходя из раскраски соответствующих показателей.

Мы подробно рассмотрели такую парадигму нейронных сетей как карты Кохонена. Основное отличие этих сетей от других моделей состоит в наглядности и удобстве использования. Эти сети позволяют упростить многомерную структуру, их можно считать одним из методов проецирования многомерного пространства в пространство с более низкой размерностью. Интенсивность цвета в определенной точке карты определяется данными, которые туда попали: ячейки с минимальными значениями изображаются темно-синим цветом, ячейки с максимальными значениями - красным.

Другое принципиальное отличие карт Кохонена от других моделей нейронных сетей - иной подход к обучению, а именно - неуправляемое или неконтролируемое обучение. Этот тип обучения позволяет данным обучающей

выборки содержать значения только входных переменных. Сеть Кохонена учится понимать саму структуру данных и решает задачи кластеризации.

Логистическая регрессия в Deductor Studio

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 линейная 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.

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* (Likelihood 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*Log likelihood) либо на каждом шаге алгоритма, либо на последнем шаге.

Бинарная логистическая регрессия эквивалента построению рейтинговой или балльной модели, т.к. если признак *fj* наблюдается у объекта *x*, то к сумме баллов добавляется вес *aj*.

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 также готовится обучающая выборка. Выходное поле может быть только дискретного типа и бинарное (т.е. количество уникальных значений по нему 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 (это позволяет сделать узел «Конечные классы»), либо воспользоваться встроенными в обработчик пошаговыми методами отбора.

Нормализация значений полей, настройка разбиения и метода отбора переменных проводятся так же, как и для линейной регрессии.

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*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 долесобытий в обучающем множестве.
- *Считать событием следующее значение* – указанное значение будет кодироваться 1, а модель – продуцировать вероятность наступления события.

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

- *<Имя выходного поля>_Вероятность события* – значение выхода в уравнении логистической регрессии от 0 до 1;
- *<Имя выходного поля>_OUT* – выходное поле, полученное на основе поля с вероятностью с использованием порога отсечения k: всем примерам, большим или равным k, приписывается событие, остальным – не-событие;
- *<Имя выходного поля>_Балл* – выходное поле, полученное на основе линеаризации вероятности события.

В самом простом случае в качестве порога можно взять значение, равное частоте встречаемости события в выборке. Однако при наличии некоторого критерия качества можно определить

оптимальный порог (оптимальный балл). Это отчасти позволяет сделать специальный визуализатор «Качество классификации» (см. одноименный раздел Качество классификации).

Также доступна для визуализации таблица сопряженности.

Для выходного поля (зависимой переменной) необходимо определиться с тем, что является событием, а что – не-событием. Это зависит от конкретной задачи. Например, если мы прогнозируем вероятность наличия заболевания, то событием будет значение «Больной пациент», не-событием – «Здоровый пациент». И наоборот, если мы ходим определить вероятность того, что человек здоров, то событием будет значение «Здоровый пациент» и так далее.

Калибровка логрегрессионной модели

Часто встречается ситуация, когда пропорции событий и не-событий в обучающей выборке не соответствуют истинным пропорциям в генеральной совокупности (или ожидаемым таковыми быть). Это могло быть сделано намеренно (если использовалось перевзвешивание примеров), или быть следствием каких-то манипуляций с данными (неполная выборка вследствие удаления части примеров).

Независимо от причин, искажение пропорций приведет к тому, что вероятности событий будут переоцениваться или недооцениваться. Для того чтобы применять модель логистической регрессии на практике, требуется

провести ее калибровку с использованием знания об истинных частотах событий и не-событий. Они могут быть прочитаны из:

- узла сэмплинга, который находится выше по ветви сценария;
- тестового множества, которое не подвергалось перевзвешиванию;
- задано вручную.

Если калибровка не требуется, то следует выбрать режим «Определить из обучающего множества (модель не корректируется)».

Калибровке подвергается только константа модели.

Логистическая регрессия в задаче кредитного скоринга

Система кредитных баллов (Кредитный скоринг) – технология, которая используется кредитно- финансовыми учреждениями, для определения и оценки платежеспособности клиентов. Кредитный скоринг позволяет на основе определенных характеристик существующих клиентов, путем подсчета баллов, определить риски, связанные с кредитованием и в конечном счете разделить заемщиков на два класса – тех, кому кредит выдать можно и тех, кому нельзя.

История скоринга тесно связана с именем американского финансиста Дюрана, который впервые разработал балльную модель оценки заемщика по совокупности его имущественных и социальных параметров (возраст, пол, профессия и т.д.). Если сумма баллов превышала определенный порог, заемщик считался кредитоспособным. И хотя в настоящее время применяют куда более сложные модели, иногда под скорингом по- прежнему понимают балльную (рейтинговую) методику оценки заемщика.

Для расчета баллов скоринговой карты сейчас используют два основных метода – логистическую регрессию и деревья решений.

Логистическая регрессия и ROC-анализ

Логистическая регрессия – полезный классический инструмент для решения задачи регрессии и классификации. В последние годы логистическая регрессия получила распространение в скоринге для расчета рейтинга заемщиков и управления кредитными рисками. Поэтому, несмотря на свое « происхождение» из статистики, логистическую регрессию и ROC- анализ (аппарат для анализа качества моделей) почти всегда можно увидеть в

наборе *Data Mining* алгоритмов.

Основная цель данного метода, как и множественного регрессионного анализа вообще, состоит в выявлении связи между несколькими независимыми переменными (называемыми также регрессорами или предикторами) и зависимой переменной. Бинарная логистическая регрессия применяется в тех случаях, когда зависимая переменная может принимать только два значения. Иными словами, с ее помощью можно оценить вероятность того, что одна из двух альтернатив наступит для конкретного испытуемого (например, возврат кредита/дефолт).

Например, при множественной линейной регрессии предполагается, что зависимая переменная является линейной функцией независимых переменных, то есть:

$$y = a + b_x x1 + b2x2 + ... + b_n x_n.$$

Эту модель можно использовать для оценки вероятности исхода события. Если параметры x_i характеризуют заемщика, то, определив стандартные коэффициенты регрессии b_i и рассчитав величину y, можно интерпретировать ее как вероятность возврата долга.

Для расчета коэффициентов логистической регрессии можно применять любые градиентные методы: метод сопряженных градиентов, методы переменной метрики и другие.

ROC-кривая (Receiver Operator Characteristic) – кривая, которая наиболее часто используется для представления результатов бинарной классификации в машинном обучении.

Предположим, у нас имеется бинарная модель, предсказывающая, что заемщик с параметрами x_i вернет долг с вероятностью P. Как воспользоваться ею практически? Очевидно, нужно выдвинуть некий критерий в виде порогового уровня вероятности: например, при $P > 0{,}8$ кредит предоставляется, а в противном случае (при $P \leq 0{,}8$) в кредите будет отказано. Значение $P = 0{,}8$ будет выполнять роль так называемой *точки (порога) отсечения* (*cut- off value*), в соответствии с которым все множество потенциальных заемщиков делится на два класса: «хороших» и «плохих» клиентов банка. При этом предполагается, что имеется некоторый параметр, при изменении которого это разбиение

будет меняться. Например, если вместо критерия *P* = 0,8 мы примем *P* = 0,6, многие клиенты, которые раньше считались « плохими», перейдут в класс « хороших».

В логистической регрессии порог отсечения изменяется от 0 до 1 – это и есть расчетное значение уравнения регрессии. Будем называть его рейтингом. Для оценки качества данной модели необходимо более детально рассмотреть ошибки, которые могут возникнуть при ее использовании.

Очевидно, при любой бинарной классификации их может быть только две:

- принять положительный случай за отрицательный (например, добросовестного заемщика за банкрота) – *ошибка первого рода;*
- отрицательный – за положительный – *ошибка второго рода.*

Для понимания сути ошибок I и II рода рассмотрим четырехпольную *таблицу сопряженности* (*confusion matrix*), которая строится на основе результатов классификации моделью и фактической (объективной) принадлежностью примеров к классам.

		Фактически
Модель	***положительно***	***отрицательно***
Положительно	TP	FP
Отрицательно	FN	TN

TP (True Positives) – количество верно классифицированных моделью положительных примеров (так называемые истинно положительные случаи);

TN (True Negatives) – количество верно классифицированных от- рицательных примеров (истинно отрицательные случаи);

FN (False Negatives) – количество положительных примеров, классифицированных моделью как отрицательные (ошибки первого рода);

FP (False Positives) – количество отрицательных примеров, классифицированных как положительные (ошибки второго рода).

При оценке модели важную роль играют следующие

соотношения, выраженные в процентах:

– доля истинно положительных примеров, распознанных моделью (*True Positives Rate*):

$$TPR = \frac{TP}{TP+FN} \cdot 100\ \%;$$

– доля ложно положительных примеров (*False Positives Rate*), то есть отношение истинно отрицательных примеров, классифицированных данной моделью как положительные, к общему числу таких примеров:

$$FPR = \frac{FP}{TN+FP} \cdot 100\ \%.$$

Очевидно, что чем мягче критерий отбора «хороших» клиентов, тем больше кредитоспособных заемщиков будут признаны таковыми и тем ближе величина *TPR* к 100 %. Но одновременно будет возрастать и доля ошибок второго рода, доля неверно квалифицированных « плохих» клиентов, отражаемая показателем *FPR*.

Введем еще два определения: чувствительность и специфичность модели. Ими определяется объективная ценность любого бинарного классификатора.

Величина $Se = TPR$ называется также *чувствительностью* модели (*Sensitivity*),

$$Se = \mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \cdot 100\ \%$$

а величина $Sp = 100 - FPR$, характеризующая долю верно распознанных отрицательных случаев, – ее *специфичностью (Specificity*).

$$Sp = \frac{TN}{TN + FP} \cdot 100\ \%$$

$$FPR = 100 - Sp.$$

Эти два показателя определяют объективную ценность любого бинарного классификатора. Когда один из них стремится к нулю, другой принимает значения, близкие к 100, и наоборот (поскольку, как мы только что заметили, величины *TPR* и *FPR* всегда изменяются в одном направлении).

Сами термины *чувствительность* и *специфичность*, не очень понятные в контексте задачи кредитного скоринга, происходят из теории систем обработки сигналов; под

чувствительностью здесь понимается способность воспринимающего устройства (*Receiver*) распознать полезный сигнал, под специфичностью – отсечь бесполезный сигнал (шум). Если провести аналогию с рассматриваемой нами задачей, «полезным сигналом» для банка будет запрос на кредит со стороны добросовестного заемщика, а «шумом» – со стороны банкрота.

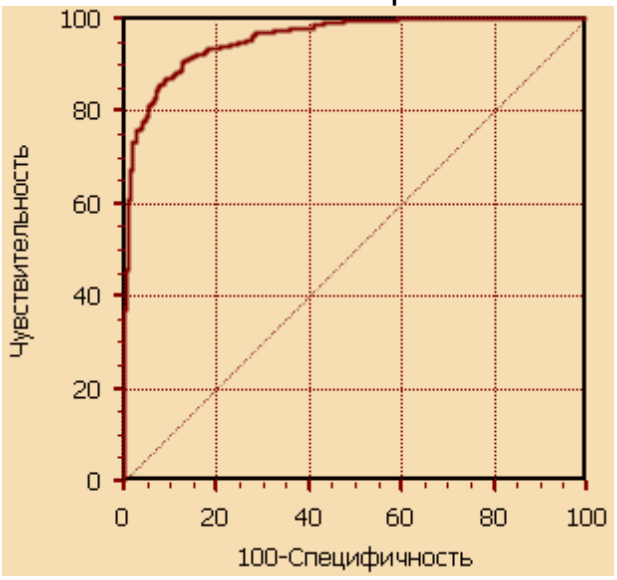
Модель с высокой *чувствительностью* часто дает истинный результат при наличии положительного исхода (обнаруживает положительные примеры).

Наоборот, модель с высокой *специфичностью* чаще дает истинный результат при наличии отрицательного исхода (обнаруживает отрицательные примеры).

Для наглядного отображения указанных характеристик используется так называемая *ROC-кривая* (*Receiver Operator Characteristic*), которая строится следующим образом.

1. Для каждого значения порога отсечения, изменяемого с шагом *dx* (например, 0,01) от 0 до 1, рассчитывают значения *чувствительности Se* и *специфичности Sp* (в качестве альтернативы можно просто взять каждое последующее значение примера в выборке).
2. Строят график зависимости *TPR* от *FPR*:
   - по оси ординат (Y) откладывают значение *чувствительности Se* = *TPR* (доля истинно положительных случаев),
   - по оси абсцисс (X) – величину *FPR* = 100 – *Sp* (доля ложно положительных случаев).

В результате вырисовывается ROC-кривая.

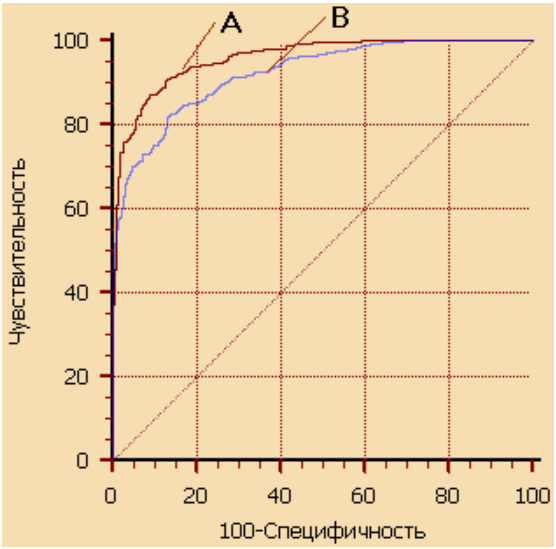


ROC-кривая

Полученный график обычно дополняют диагональной линией *TPR* = *FPR*.

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 *ложно положительных* 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.

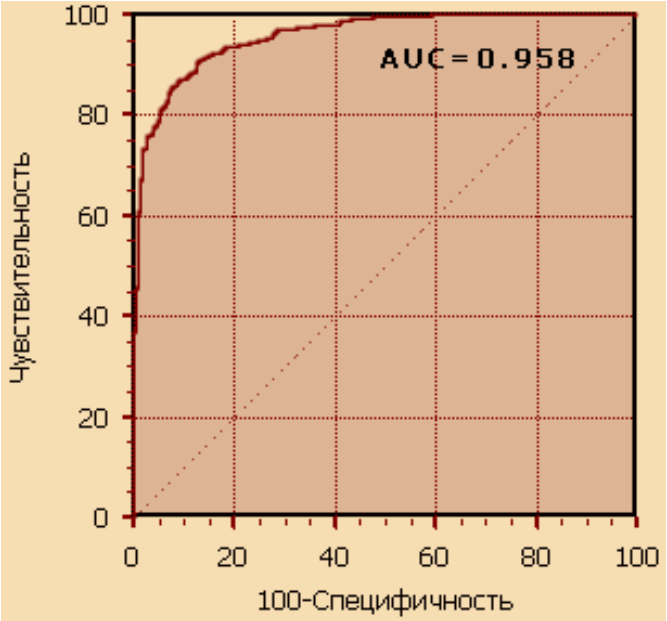
При визуальной оценке ROC-кривых расположение их относительно друг друга указывает на их сравнительную эффективность. Кривая, расположенная выше и левее, свидетельствует о большей предсказательной способности модели. Так, на рисунке две ROC-кривые совмещены на одном графике. Видно, что модель «А» лучше.



Сравнение ROC-кривых

Сравнение ROC-кривых

Иногда ROC-кривые располагаются достаточно плотно или пересекаются, и их визуальное сравнение затруднено. В этом случае для оценки эффективности модели можно использовать площадь под кривой (AUC, Area Under Curve). Поскольку ROC-кривая всегда расположена выше диагонали, треугольник под диагональю площадью 0,5 всегда входит в состав измеряемой области. Таким образом, показатель AUC может изменяться от 0,5 (« бесполезный» классификатор) до 1,0 (идеальная модель).



Площадь под кривой (показатель AUC)

В литературе иногда приводится следующая экспертная шкала для значений AUC, по которой можно судить о качестве модели:

Интервал AUC	**Качество модели**
0.9–1.0	Отличное
0.8–0.9	Очень хорошее
0.7–0.8	Хорошее
0.6–0.7	Среднее
0.5–0.6	Неудовлетворительное

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*).

В качестве критерия оптимальности могут, в частности, выступать следующие условия:

1) максимальная *чувствительность* при заданном уровне специфичности (например, при *специфичности* не ниже 80 %);

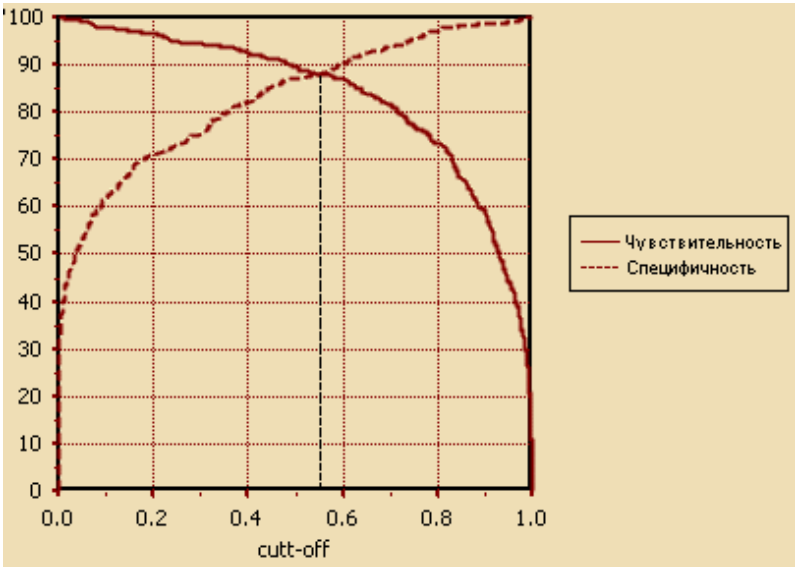
2) максимальная *специфичность* при заданном уровне *чувствительности* (например, при *чувствительности* не ниже 80 %);

3) максимальная сумма *чувствительности* и *специфичности* модели
Se + Sp → max);

4) достижение баланса между *чувствительностью* и *специфичностью (Se → Sp).*

Если принимается последний вариант, оптимальный порог отсечения (*точка баланса*) соответствует точке пересечения двух кривых (*чувствительности* и *специфичности)*.

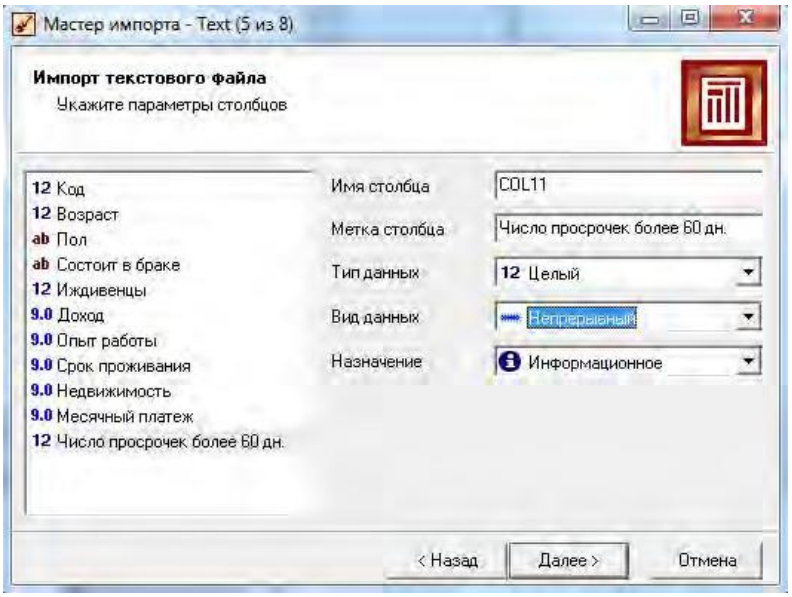
Данный вариант представлен на рисунке.



Достижение баланса между чувствительностью и специфичностью

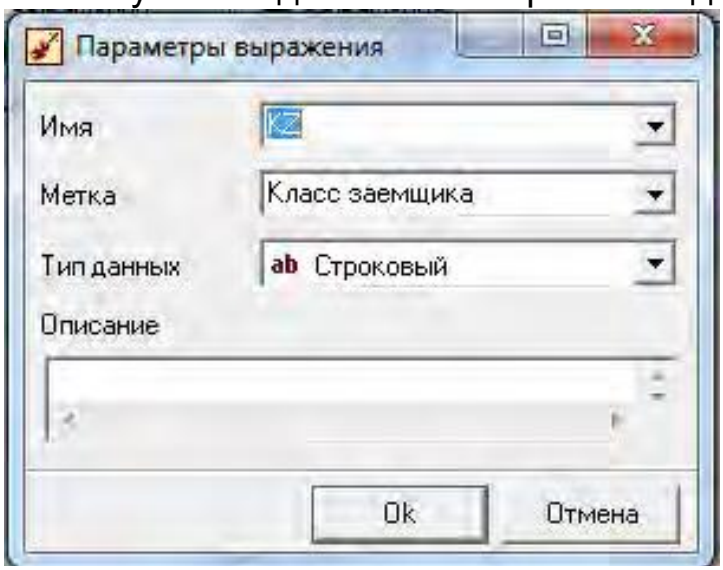
Порог – точка пересечения двух кривых, по оси X откладывается порог отсечения, а по оси Y – чувствительность или специфичность модели.

1) Импортируйте в новый проект (в файл **L6_1. ded**) данные из текстового файла **loans_ demo. txt,** в котором хранится информация о заемщиках банка. При импорте исправьте типы полей, как указано на рисунке.



Параметры типа полей

2) Необходимо уточнить, по каким правилам следует относить заемщиков к одному из двух классов («хороший» или «плохой»), используя данные о возврате ими долга.



Параметры класса заемщика

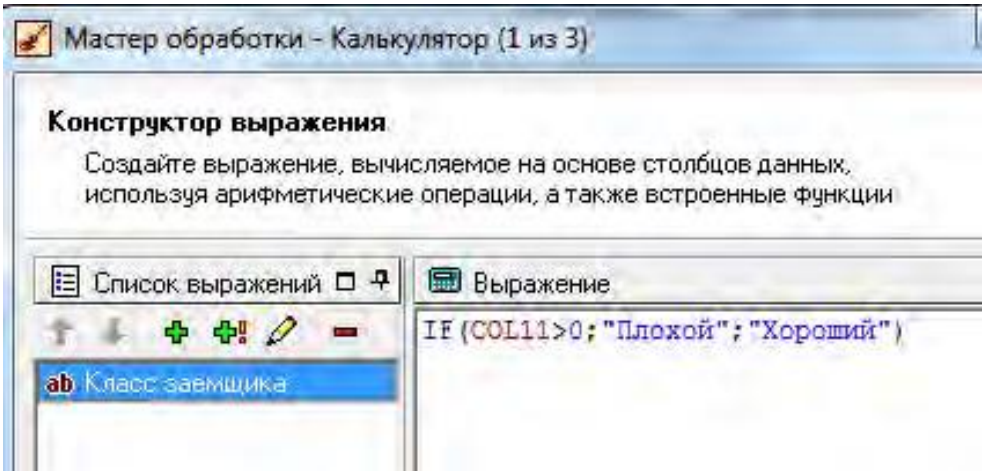
Будем считать, что для физических лиц обслуживание долга признается *плохим*, если в течение последних 180 календарных дней имеются платежи по основному долгу и (или) по процентам, просроченным более чем на 60 календарных дней.

В данном случае в файле **loans_demo.txt** последний столбец, характеризующий качество обслуживания долга

заемщиком, представлен полем *Число просрочек свыше 60 дн.* Остальные поля (кроме информационного поля *Код*) содержат социально-экономические характеристики заемщиков: их возраст, пол, доход и др.

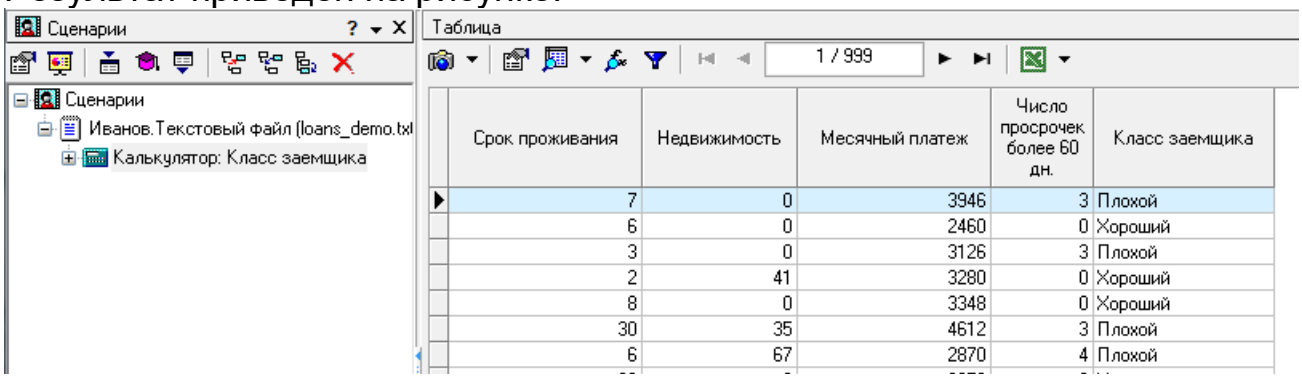
Из поля *Число просрочек более 60 дн.* получим новое поле *Класс заемщика.*

Для этого с помощью обработчика *Калькулятор* создадим *строковое* поле и в строке функции напишем условный оператор (*Если Число просрочек* >0, *то Класс заемщика – плохой*, иначе – *хороший*).



Запуск обработчика калькулятор

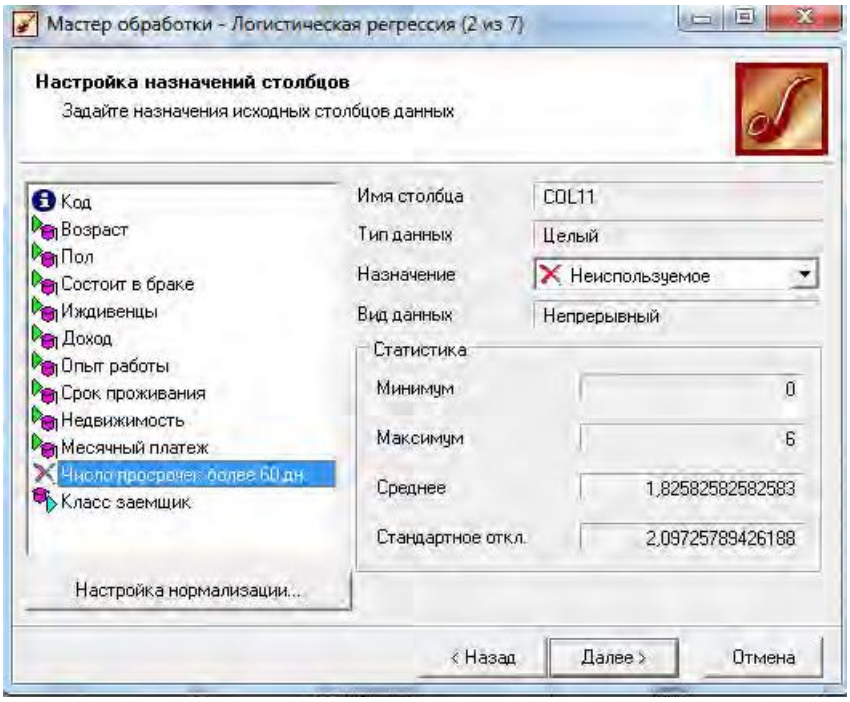
Результат приведен на рисунке.



Срок проживания	Недвижимость	Месячный платеж	Число просрочек более 60 дн.	Класс заемщика
7	0	3946	3	Плохой
6	0	2460	0	Хороший
3	0	3126	3	Плохой
2	41	3280	0	Хороший
8	0	3348	0	Хороший
30	35	4612	3	Плохой
6	67	2870	4	Плохой

Результат работы правил, к какому из двух классов относить заемщиков

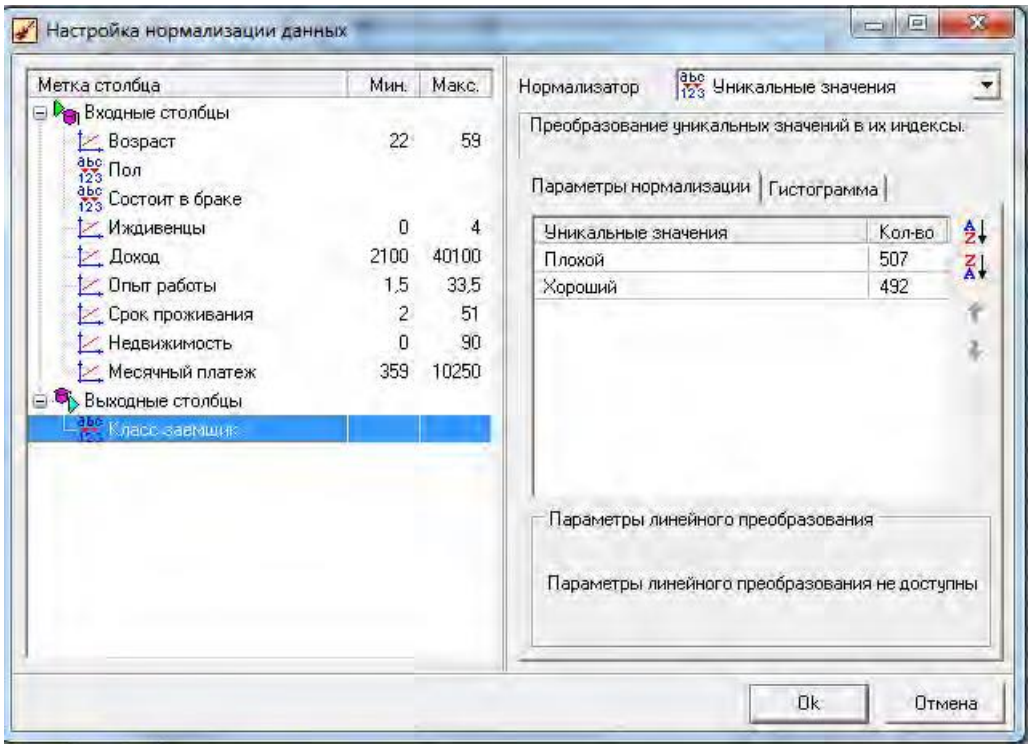
3)Добавим в ветвь сценария обработчик Логистическая регрессия и на первом шаге **Мастера** зададим входные и выходные значения столбцов.



Мастер обработки *Логистическая регрессия*

Поле *Код* будет *информационным*, *Число просрочек более 60 дн. – неиспользуемым*, *Класс заемщика – выходным*, остальные поля – *входными*.

После щелчка по кнопке *Настройка нормализации* появится диалоговое.

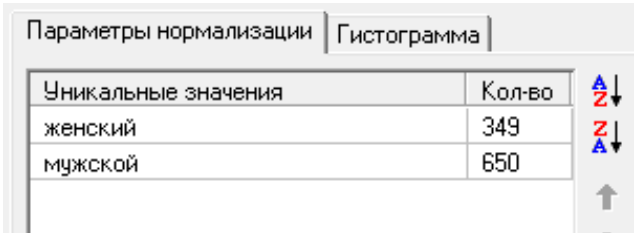


Настройка нормализации данных

Для логистической регрессии необходимо настроить:

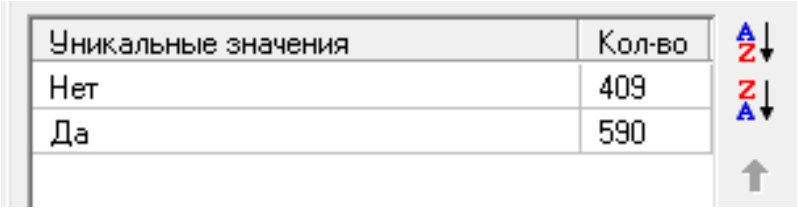
- способы кодирования дискретных входных полей (битовая маска или уникальные значения);
- значения положительного и отрицательного событий для выходного поля.

В рассматриваемом случае имеется два входных дискретных поля – *Пол* и *Состоит в браке*. Для них рекомендуется указать способ кодирования *Уникальные значения.* Порядок списка таких значений влияет на кодирование значений полей. Так, для поля *Пол* первое уникальное значение («женский») будет закодировано нулем, второе («мужской») – единицей. Это значит, что при расчете кредитного рейтинга по уравнению логистической регрессии женщинам всегда будет начисляться 0 баллов, а мужчинам – какой-либо отличный от нуля балл.



Кодирование по уникальным значениям *Пол*

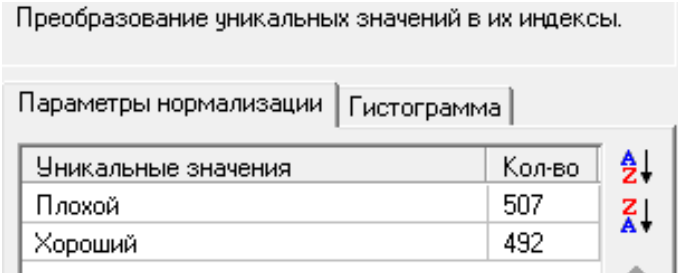
Аналогичным образом для поля *Состоит в браке* зададим кодирование по уникальным значениям в следующем порядке: «нет» (значение 0), «да» (значение 1).



Кодирование по уникальным значениям *Состоит в браке*

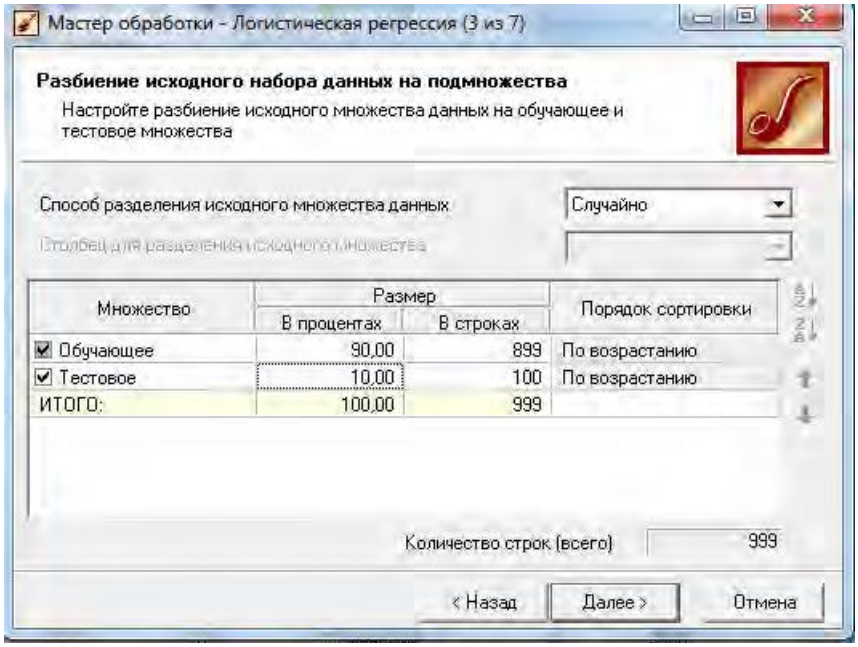
Для *выходного* поля *Класс заемщика* порядок сортировки уникальных значений (их всегда два) определяет тип события: первое – отрицательное, второе – положительное. В данном случае, чем выше рейтинг – тем выше кредитоспособность, поэтому значение «хороший» будет положительным исходом события, а «плохой» –

отрицательным.

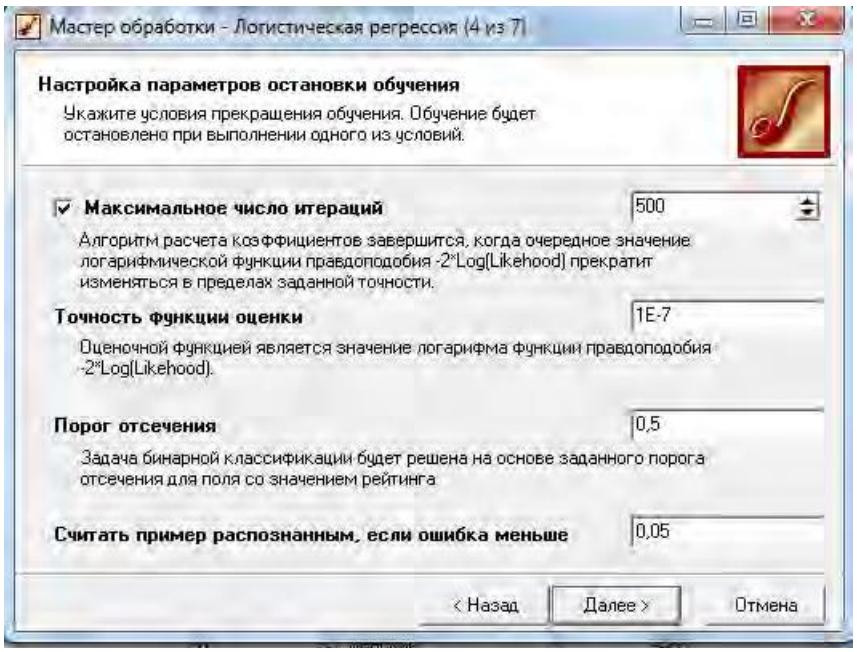


Преобразование уникальных значений

4) В следующих окнах **Мастера** будет предложено настроить обучающие и тестовые множества, а также изменить параметры алгоритма логистической регрессии. По умолчанию предлагается порог классификации, равный 0,5.

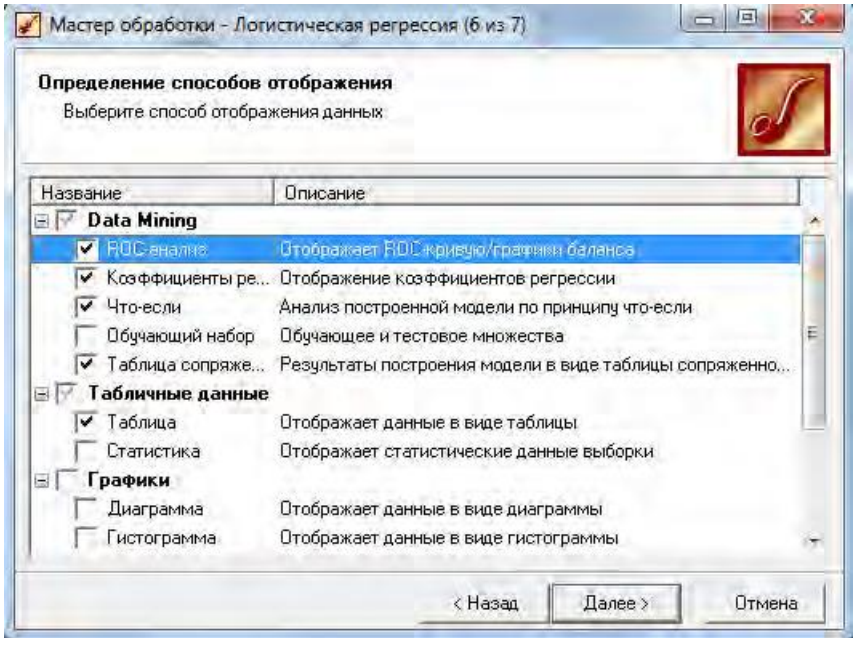


Настройка обучающего и тестового множеств



Изменение параметров логистической регрессии

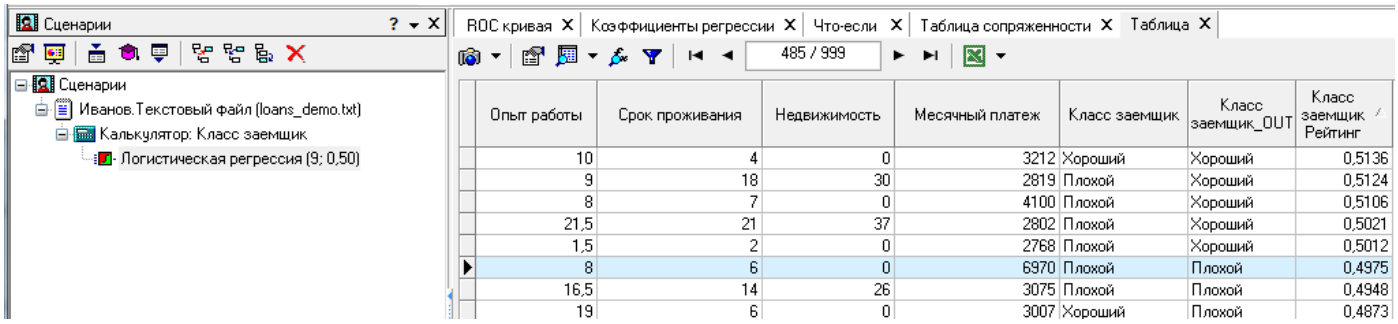
5) После щелчка по кнопке *Пуск* в последнем окне будет построена модель, и нужно будет выбрать визуализаторы узла. Отметим флажками следующие: *ROC-анализ, Коэффициенты регрессии, Что-если, Таблица сопряженности, Таблица.*



Выбор способа отображения данных

6) Визуализатор *Таблица* показывает, что после применения обработчика *Логистическая регрессия* в массиве данных появились две новые колонки: *Класс заемщика_OUT* и *Класс заемщика Рейтинг*. Рейтинг представляет

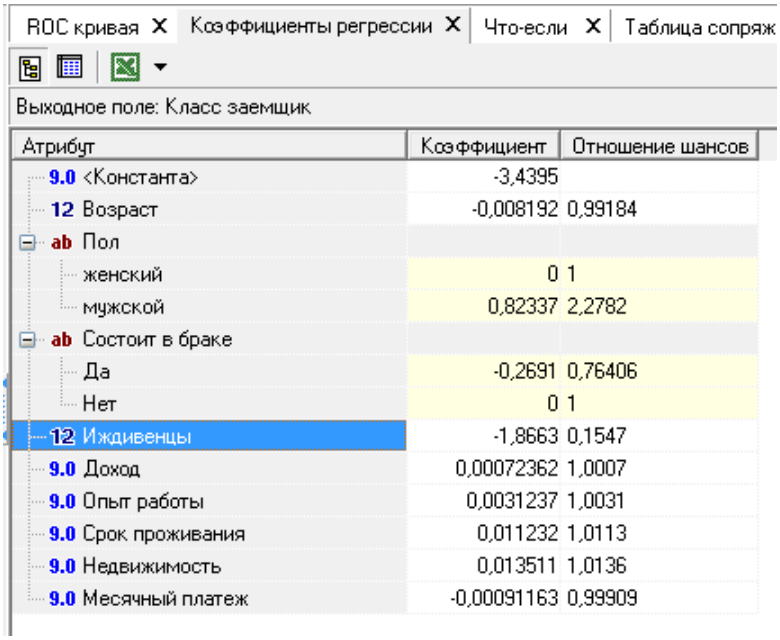
собой значение независимой переменной *y*, рассчитанное по уравнению логистической регрессии, а первое поле – принадлежность случая к тому или иному классу в зависимости от установленного порога классификации (в данном случае он равен 0,5).



Опыт работы	Срок проживания	Недвижимость	Месячный платеж	Класс заемщик	Класс заемщик_OUT	Класс заемщик Рейтинг
10	4	0	3212	Хороший	Хороший	0,5136
9	18	30	2819	Плохой	Хороший	0,5124
8	7	0	4100	Плохой	Хороший	0,5106
21,5	21	37	2802	Плохой	Хороший	0,5021
1,5	2	0	2768	Плохой	Хороший	0,5012
8	6	0	6970	Плохой	Плохой	0,4975
16,5	14	26	3075	Плохой	Плохой	0,4948
19	6	0	3007	Хороший	Плохой	0,4873

Логистическая регрессия отображена визуализатором *Таблица*

7) о влиянии факторов (входных параметров модели) на результат. Например, каждый дополнительный иждивенец уменьшает кредитный рейтинг заемщика на 1,87, а каждый дополнительный год стажа работы увеличивает его на 0,0031.

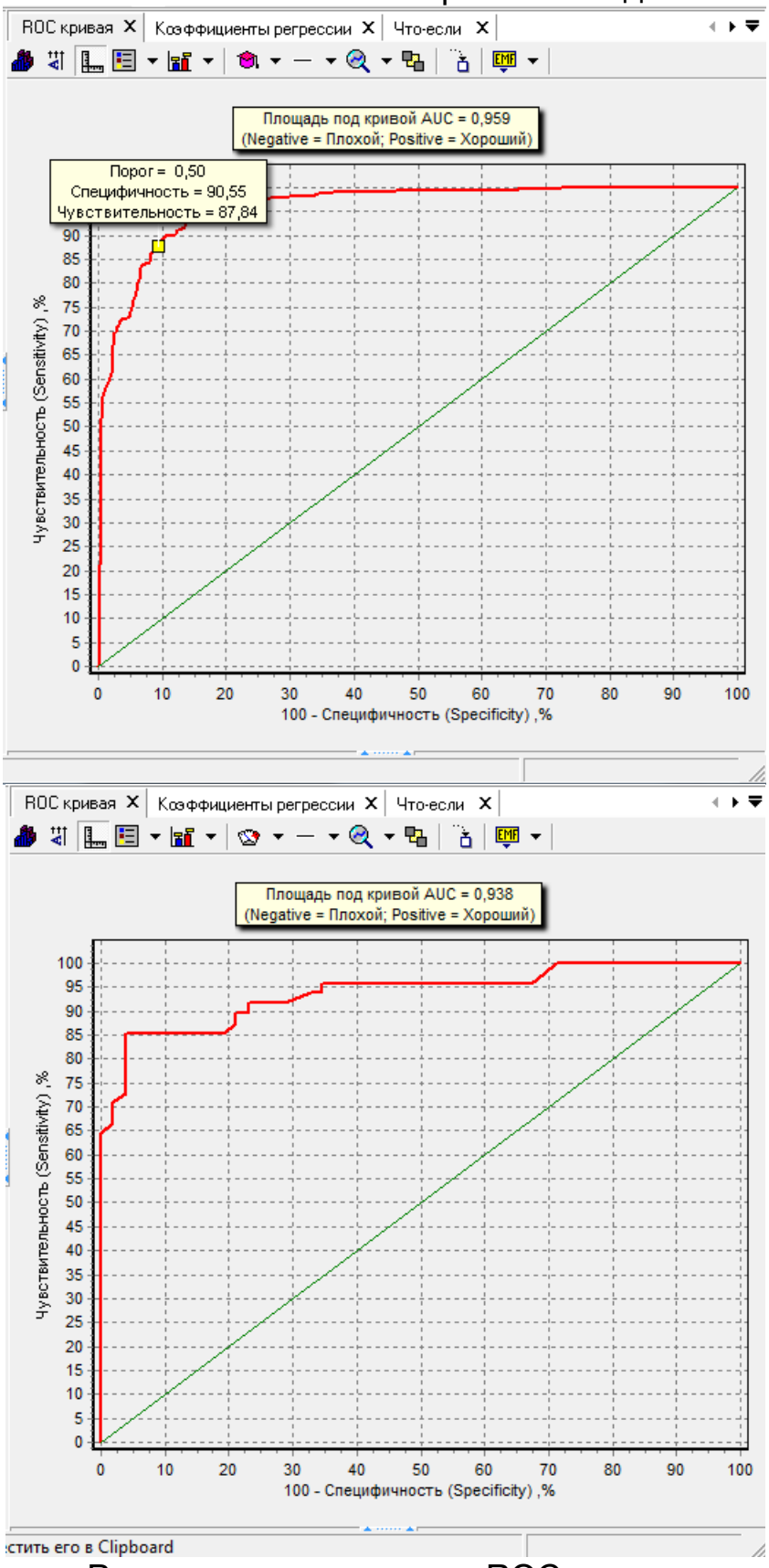


Атрибут	Коэффициент	Отношение шансов
9.0 <Константа>	-3,4395	
12 Возраст	-0,008192	0,99184
ab Пол		
женский	0	1
мужской	0,82337	2,2782
ab Состоит в браке		
Да	-0,2691	0,76406
Нет	0	1
12 Иждивенцы	-1,8663	0,1547
9.0 Доход	0,00072362	1,0007
9.0 Опыт работы	0,0031237	1,0031
9.0 Срок проживания	0,011232	1,0113
9.0 Недвижимость	0,013511	1,0136
9.0 Месячный платеж	-0,00091163	0,99909

Результат визуализации *Коэффициентом регрессии*

8)Визуализатор *ROC-анализ* выводит график ROC-кривой, на котором по умолчанию отмечается (желтым квадратным маркером) текущий порог отсечения, значения *чувствительности* и *специфичности*, показатель AUC и типы событий. В данном случае площадь под кривой AUC = 0,959 на обучающем множестве (кнопка

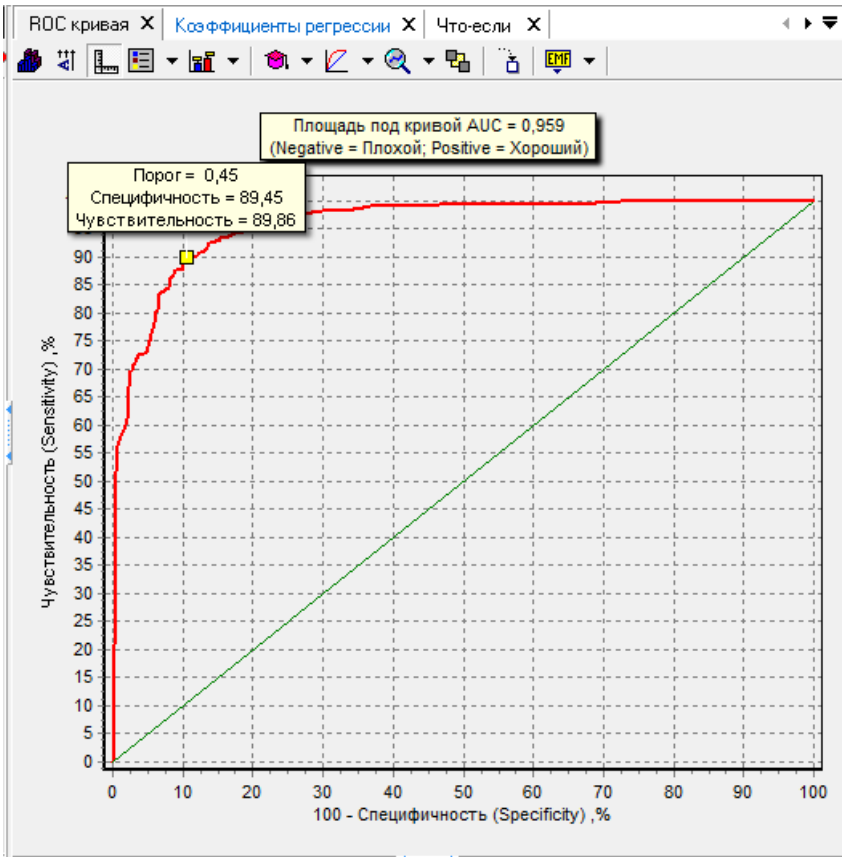
) и AUC = 0,938 на тестовом (кнопка), что говорит об очень хорошей предсказательной способности построенной модели.



Результат визуализации ROC-анализ

Оптимальный порог отсечения для данной модели не равен

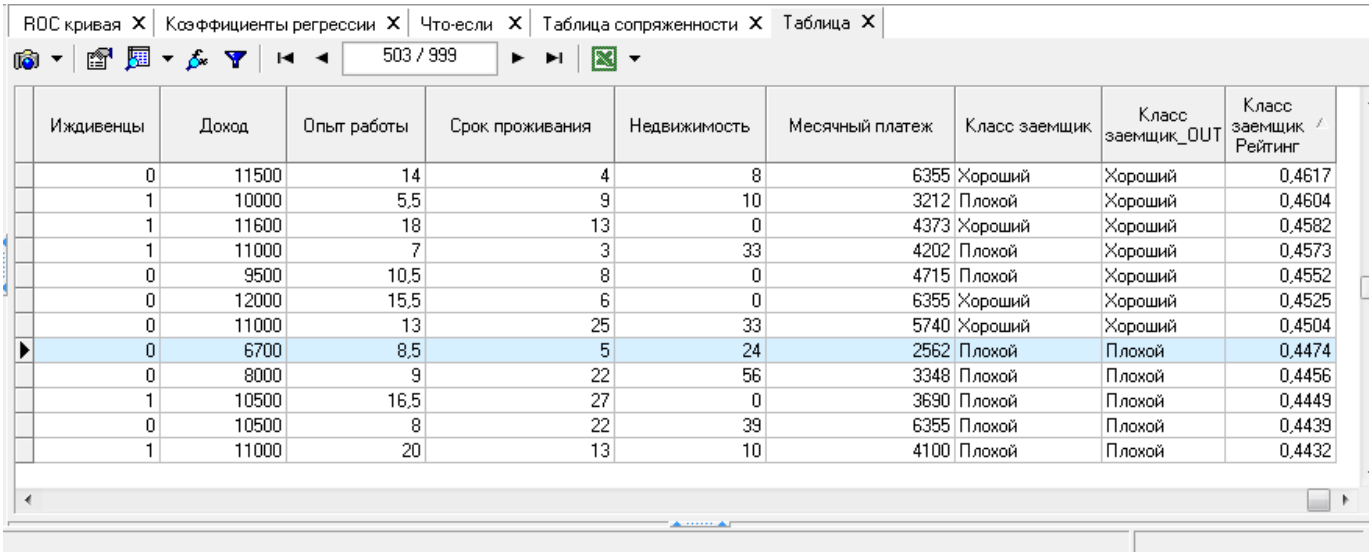
предварительно установленной величине 0,5; чтобы определить его, нужно в выпадающем меню кнопки  выбрать пункт *Максимум.*



Результат визуализации ROC-анализ с оптимальным порогом отсечения

Оказывается, что максимум суммарной *чувствительности* и *специфичности* достигается в точке 0,45, для которой *чувствительность Se* = 90 %, *специфичность Sp* = 89 %. Это означает, что 90 % благонадежных заемщиков будут выявлены классификатором, а 100 – 89 = 11 % недобросовестных заемщиков получат одобрение при запросе кредита (кредитный риск).

Для установки нового порога отсечения, равного 0,45, необходимо перенастроить узел-обработчик логистической регрессии, нажав кнопку  . Результат можно посмотреть в визуализаторе *Таблица*.



ROC кривая | Коэффициенты регрессии | Что-если | Таблица сопряженности | Таблица

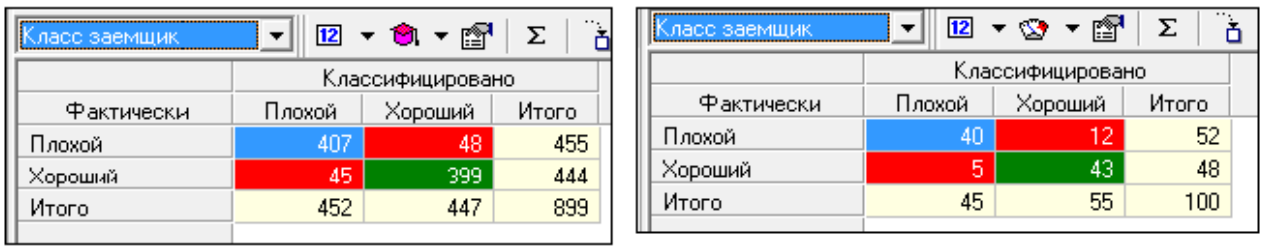
503 / 999

Иждивенцы	Доход	Опыт работы	Срок проживания	Недвижимость	Месячный платеж	Класс заемщик	Класс заемщик_OUT	Класс заемщик Рейтинг
0	11500	14	4	8	6355	Хороший	Хороший	0,4617
1	10000	5,5	9	10	3212	Плохой	Хороший	0,4604
1	11600	18	13	0	4373	Хороший	Хороший	0,4582
1	11000	7	3	33	4202	Плохой	Хороший	0,4573
0	9500	10,5	8	0	4715	Плохой	Хороший	0,4552
0	12000	15,5	6	0	6355	Хороший	Хороший	0,4525
0	11000	13	25	33	5740	Хороший	Хороший	0,4504
0	6700	8,5	5	24	2562	Плохой	Плохой	0,4474
0	8000	9	22	56	3348	Плохой	Плохой	0,4456
1	10500	16,5	27	0	3690	Плохой	Плохой	0,4449
0	10500	8	22	39	6355	Плохой	Плохой	0,4439
1	11000	20	13	10	4100	Плохой	Плохой	0,4432

Результат визуализации Таблица

9) Нетрудно видеть, 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, вызванный неплатежами, во втором – коммерческий риск, связанный с упущенной выгодой.

Это хорошо иллюстрирует визуализатор *Таблица* сопряженности. Он позволяет сравнить категориальные значения выходного поля исходной выборки (обучающей или тестовой) с рассчитанными по модели с выбранным порогом отсечения (в данном случае – 0,45).



Класс заемщик

Фактически	Классифицировано: Плохой	Хороший	Итого
Плохой	407	48	455
Хороший	45	399	444
Итого	452	447	899

а

Класс заемщик

Фактически	Классифицировано: Плохой	Хороший	Итого
Плохой	40	12	52
Хороший	5	43	48
Итого	45	55	100

б

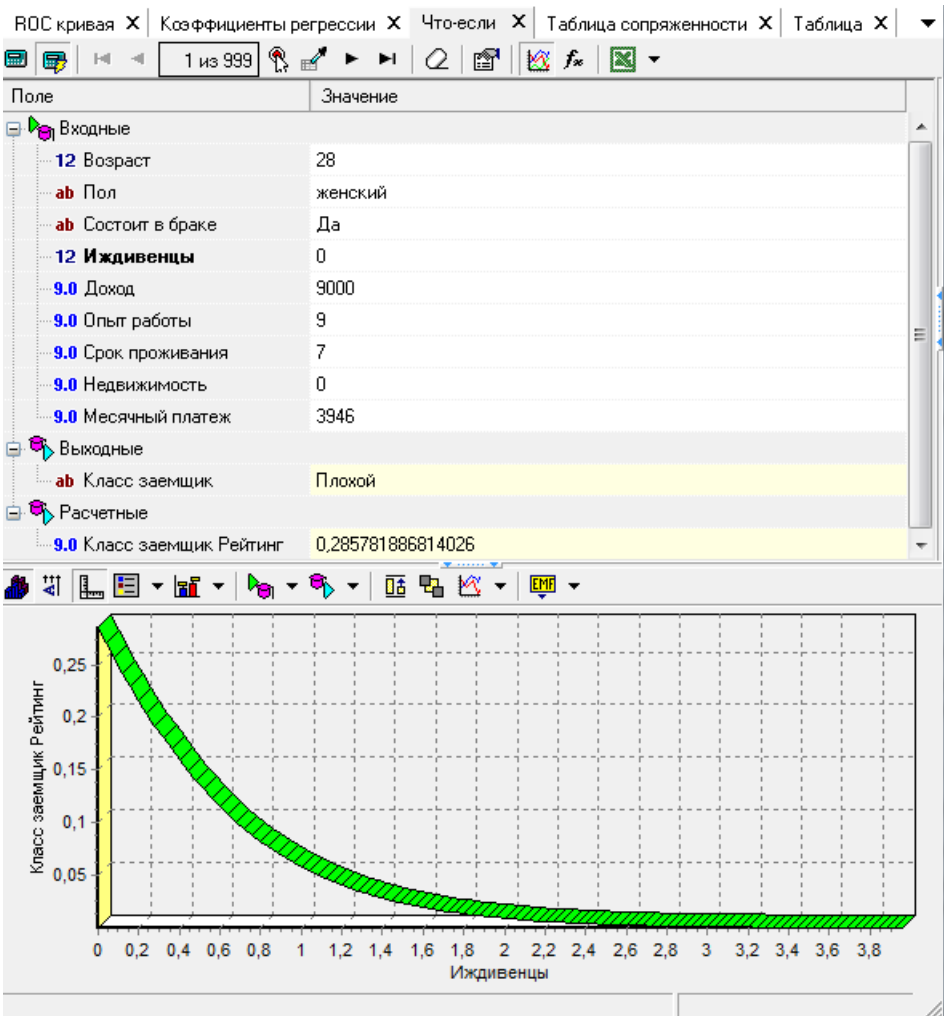
Результат визуализации *Таблица сопряженности*

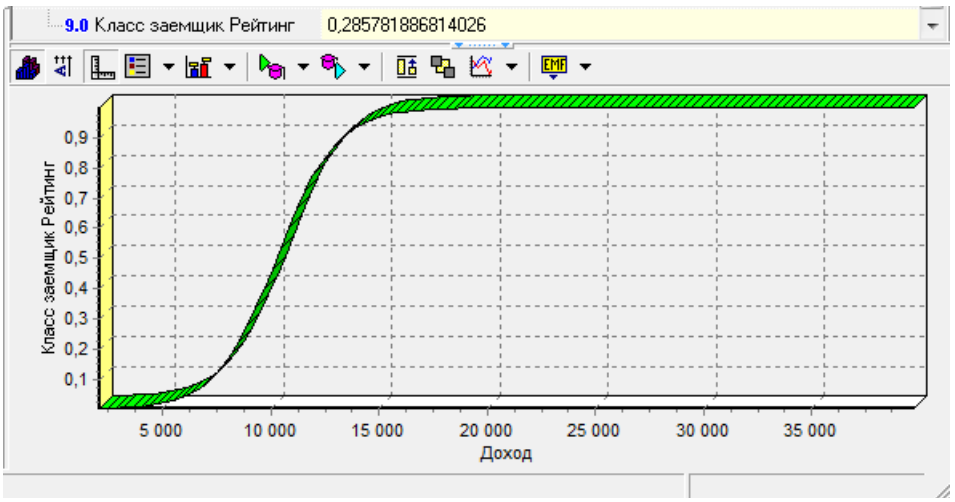
На обучающем множестве (рабочая выборка) модель реже отказывала в выдаче «хорошим» заемщикам (45 ошибочных случаев) и чаще выдавала кредит «плохим» клиентам (48); точность классификации составила 89 % (407 + 399) : 899 = 89 %). На тестовом множестве наблюдается примерно та же картина (точность классификации (40 + 43) : 100 = 83 %). Если такое решение не соответствует кредитной политике банка, можно поднять порог

отсечения и добиться того, чтобы модель чаще выдавала отрицательное решение.

10) Визуализатор *Что-если* позволяет определить, как будет вести себя построенная модель при подаче на ее вход тех или иных данных. Другими словами, проводится эксперимент, в котором, изменяя значения входных полей обучающей или рабочей выборки для модели логистической регрессии, можно увидеть, как изменяются выходные значения модели (рис. 6.23).

Окно визуализатора включает два представления – табличное и графическое, которые формируются одновременно. В верхней части таблицы отображаются входные поля, в нижней -выходные и расчетные. Изменяя значения входных полей, пользователь дает команду на выполнение расчета и может отслеживать значения, получаемые на выходе логистической регрессии.





Отслеживание значений, получаемых на выходе логистической регрессии.

Сохраните результаты в файл **L6_1.ded**.