

**МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ УКРАИНЫ
НАЦИОНАЛЬНАЯ МЕТАЛОГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ**

А.И. Михалев, Е.Ю. Островская

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению контрольных работ по дисциплине
«Компьютерные методы интеллектуальной обработки данных»
для студентів зі спеціальності 122 «Комп'ютерні науки»
галузі знань 12 «Інформаційні технології»

Днепр НМетАУ - 2019

ЗАДАНИЕ НА КОНТРОЛЬНУЮ РАБОТУ

Изучить возможности программного продукта Xelopes, который является открытой, платформенно-независимой средой для решения разнообразных задач анализа данных с помощью технологии Data Mining.

Порядок выполнения работы:

1. Ознакомится с основными возможностями и средствами GUI Xelopes.
2. Изучить принципы работы GUI Xelopes
3. Ответить на контрольные вопросы: по два вопроса на выбор из каждого уровня (всего 4 вопроса).
4. Выполнить три лабораторные работы (**ответы на контрольные вопросы и три лабораторные работы – это будет составлять одну контрольную работу**).

Контрольные вопросы:

1 уровень

1. Data Mining
2. Data mining и базы данных
3. Data mining и искусственный интеллект
4. Задачи Data mining
5. Интеллектуальный анализ данных
6. Архитектура Модели интеллектуального анализа данных
7. Определение Модели интеллектуального анализа данных
8. Свойства Модели интеллектуального анализа данных
9. Классификация данных
10. Кластеризация данных
11. Ассоциативные правила
12. Прогнозирование
13. Деревья решений
14. Области применения Data mining
15. Средства интеллектуального анализа данных

2 уровень

1. Какую статистическую информацию можно получить средствами GUI Xelopes.
2. Какие существуют типы атрибутов и их характеристики.
3. Какие mining модели можно построить средствами GUI Xelopes.
4. Какие существуют mining модели не реализованные в GUI Xelopes.
5. Какие действия можно выполнить с моделью.
6. Какие модели могут быть применены к другим данным и почему.
7. Какие проблемы возникают с исходными данными.
8. Почему для одних и тех же данных не могут быть построены все виды моделей.
9. Какие требования на исходные данные накладывают разные алгоритмы data mining.
10. Какие параметры необходимо настроить для построения ассоциативных правил. Как от них зависит результат (построенная модель).
11. Какие параметры необходимо настроить для построения дерева решений. Как от них зависит результат (построенная модель).
12. Какие параметры необходимо настроить для выполнения кластеризации. Как от них зависит результат (построенная модель).
13. Какие параметры определяются алгоритмами. Привести примеры. Как от них зависит результат (построенная модель).
14. Что такое сиквенциальный анализ и чем он отличается от поиска ассоциативных правил.
15. Что необходимо для построения модели.

16. Какие параметры должны быть установлены для построения ассоциативных правил.
17. Какие параметры должны быть установлены для построения кластеров.
18. Какие параметры должны быть установлены для построения дерева решений
19. Как можно классифицировать алгоритмы в соответствии с иерархией принятой в CWM и Xelopes.
20. Что такое формат PMML.

Библиотека Xelopes

Разработанный компанией ZSoft, программный продукт Xelopes является открытой, платформенно-независимой средой для решения разнообразных задач анализа данных с помощью технологии Data Mining.

Xelopes предназначен для встраивания в любую информационную систему, где необходима функциональность Data Mining. В этом смысле, продукт реализует идею "Embedded Data Mining".

Уникальность Xelopes заключается в его гибкой архитектуре, состоящей из трех главных частей: мощной и расширяемой библиотеки алгоритмов Data Mining, унифицированного, основанного на открытых стандартах, интерфейса прикладного программирования, а также, использующей стандарт PMML, подсистемы обмена Data Mining моделями.



Библиотека алгоритмов Xelopes, содержит все необходимые средства для решения задачи анализа рыночных корзин, сиквенциального анализа, проведения классификации методами decision tree и support vector machine, а также кластерного анализа. В настоящее время ведутся работы по добавлению в библиотеку новых алгоритмов.

Встроенная в Xelopes поддержка языка PMML (Predictive Model Markup Language – интеллектуальный язык разметки), позволяют ему обмениваться информацией с целым рядом коммерческих Data Mining продуктов.

Благодаря своей гибкости и функциональности Xelopes является оптимальным выбором для построения информационно-аналитических систем для бизнеса и науки.

Xelopes может быть использован для построения любого приложения, где необходима функциональность Data Mining.

Классификация (распознавание объектов)

Задача классификации заключается в построении моделей, которые по набору исходных данных $Z=(x,y)$, где x - входной вектор, y - отклик системы, позволяют получить прогноз отклика системы на новые входные данные.

Например, используя классификацию, можно обучить приложение распознавать поддельные денежные купюры или определять клиентов банка, имеющих высокую степень риска.

Кластеризация (выявление групп схожих объектов)

Задача кластеризации заключается в обнаружении областей сгущения в пространстве исходных данных. Другими словами, кластеризация позволяет выявлять группы схожих объектов. Использовать кластеризацию можно, например, в маркетинге для выделения групп покупателей, имеющих схожие социальные признаки или стереотипы поведения (сегментация рынка).

Ассоциативный анализ (зависимости между объектами)

Ассоциативный анализ или анализ рыночных корзин позволяет выявлять закономерности присутствующие в наборах повторяющихся данных. Такими наборами могут быть покупки, совершаемые посетителями магазина или же, изучаемые в биоинформатике, символьные последовательности с информацией о первичной структуре белков и последовательностях ДНК.

Правила установки программы XELOPESGUI

1.Инсталляция

1.1.Скопировать каталоги:

javadoc,
jgrash-2.1-java1,
XELOPES UML Sheme,
XelopesGuiLast

на ваш жесткий диск.

1.2. Убедитесь, что у вас УСТАНОВЛЕНЫ программы:

Borland JBuilder (версии выше 4)

j2sdk1.4.2_01

j2re1.4.2_01

DirectX(версии больше 8)

иначе установка бесполезна.

1.3.Скопировав каталоги из пункта 1. на свой жесткий диск откройте

каталог: ***jgrash-2.1-java1.***

В нем вы обнаружите 3 каталога: *ChartText*, *jgrash-2.1-java1.4*, *jgrash-examples-2.0*,
и 1 приложение : **java3d-1_3_1-beta-windows-i586-directx-sdk.**

Запустите это приложение.

1.4.Вам предложат установить программу: ***Java 3D 1.3.1-beta (DirectX) SDK***

Нажмите Next>.

1.5.Прочитайте и ознакомьтесь с лицензионным соглашением на программу *Java 3D 1.3.1-beta (DirectX) SDK*

Нажмите Yes.

1.6.Прочитайте текст, содержащий информацию о том, какие минимальные требования нужны для установки программы *Java 3D 1.3.1-beta (DirectX) SDK* и какие ошибки могут возникнуть при установке.

Нажмите Next>.

1.7. Место для установки программы выберите каталог

<Полный путь к каталогу>\j2sdk1.4.2_01

Нажмите Yes.

Программа установки дальше сама проведет установку программы.

2.Настройка программы XelopesGui.

2.1. Откройте каталог *XelopesGuiLast* и запустите файл *XelopesGui.jpx*.

Программа откроется с помощью ***Borland Jbuilder.***

2.2. В ***Borland Jbuilder*** на панели задач откройте *Project=>Project Properties.*

2.3. В *Project Properties* на вкладке *Paths* в поле JDK установите JDK-а **java 1.4.2_01-XXX**, если оно есть.

2.4 Если её нет откройте *Select a JDK.* Нажмите **NEW...**

2.5 В окне *New JDK Wizard* откройте поле *Existing JDK home path.*

В появившемся окне найдите директорию **j2sdk1.4.2_01** и нажмите ОК.

2.6 Подождите....

В поле *Name for this JDK* окна *New JDK Wizard* появится название JDK-а **java 1.4.2_01-XXX.** Нажмите ОК.

2.7 В окне *Select a JDK* выберите JDK-а **java 1.4.2_01-XXX.** Нажмите ОК.

2.8 В *Project Properties* нажмите ОК.

2.9 Откомпилируйте проект(Ctrl-F9) и запустите проект (F9).

Лабораторная работа №1

Знакомство с GUI интерфейсом библиотеки data mining алгоритмов

Цель работы: Ознакомиться и получить навыки работы GUI интерфейсом библиотеки data mining алгоритмов Xelopes

Задание: Получить информацию о данных из файлов `transact.arff` и `weather-nominal.arff` и построить для них задачи поиска ассоциативных правил, кластеризации и классификации.

Общие сведения

1.1. Введение

Xelopes свободно распространяемая библиотека, обеспечивающая универсальную основу для стандартного доступа к алгоритмам data mining. Она была разработана немецкой компанией ProdSys в тесном сотрудничестве со специалистами российской фирмы ZSoft. Для удобной работы с библиотекой с ней поставляется GUI интерфейс GUI Xelopes, реализованный в виде отдельного приложения. Он позволяет выполнять следующие основные функции:

- Загрузить данные представленные в виде текстового файла формата arff и просмотреть их в табличном виде.
- Получить информацию об атрибутах данных (полях таблицы)
- Получить статическую информацию об исходных данных:
- Построить модель data mining.
- Для ассоциативных правил, деревьев решений и дейтограмм визуализировать построенную модель.
- Сохранить модель и применить ее в дальнейшем.

Рассмотрим перечисленные функции более подробно.

1.2. Загрузка и просмотр исходных данных

Для загрузки исходных данных необходимо открыть диалог представленный на рис. 1.1. Это можно выполнить или нажатием кнопки **Open Mining Data** на панели инструментов или выбором пункта меню **File | Open Mining Data**. Кроме того диалог открывается при запуске программы.

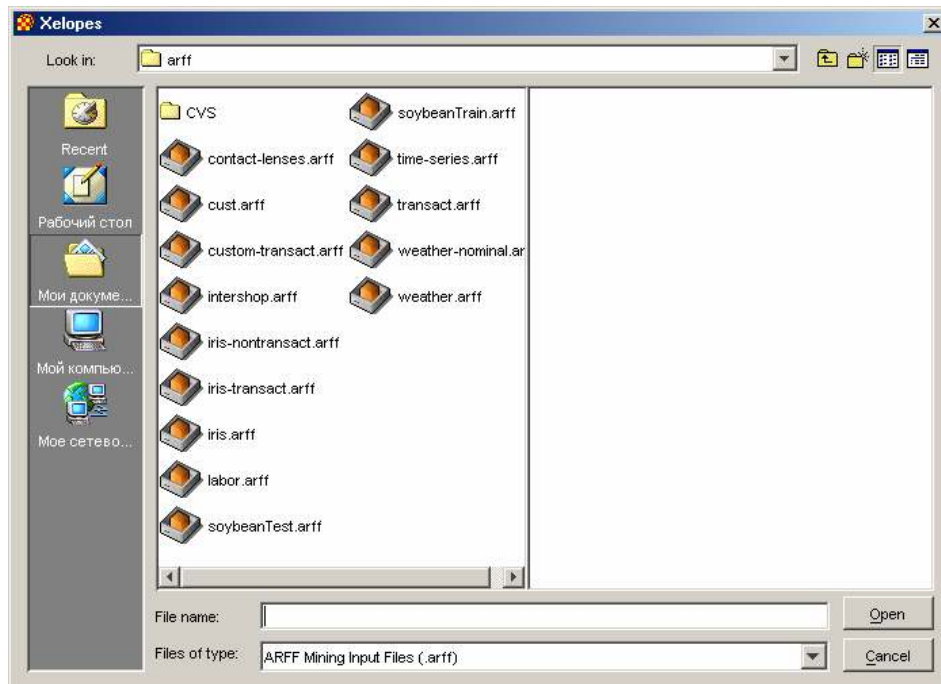


Рис. 1.1 Диалог загрузки исходных данных (в ОС Windows 2000)

Используя данный диалог необходимо выбрать текстовый файл с данными, представленными в формате arff. Нажатие на кнопку **Open** приведет к загрузке данных из выбранного файла.

После загрузки данных на панели инструментов становятся доступными следующие кнопки:

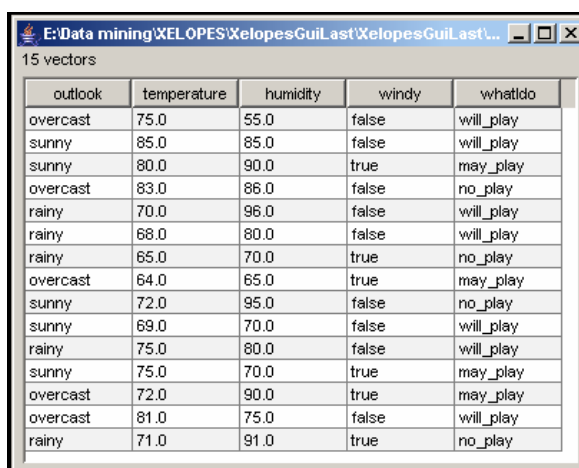
View Input Data – отображение исходных данных;

Display Data Description – получение информации о атрибутах исходных данных;

Display Descriptive Statistics – получение статистической информации об исходных данных;

Build Mining Model – генерация mining модели для загруженных исходных данных.

Для просмотра исходных данных в табличном виде необходимо нажать кнопку **View Input Data** на панели инструментов или выбрать пункт меню **File | View Data Source**. При этом открывается окно представленное на рис. 1.2. В заголовке окна отображается полный путь к файлу, из которого были загружены данные. Данные представляются в виде таблицы в которой строки соответствуют исследуемым объектам, а колонки атрибутам характеризующим их. Над таблицей можно заметить информацию об общем количестве объектов (векторов) представленных в таблице.



outlook	temperature	humidity	windy	whatdo
overcast	75.0	55.0	false	will_play
sunny	85.0	85.0	false	will_play
sunny	80.0	90.0	true	may_play
overcast	83.0	86.0	false	no_play
rainy	70.0	96.0	false	will_play
rainy	68.0	80.0	false	will_play
rainy	65.0	70.0	true	no_play
overcast	64.0	65.0	true	may_play
sunny	72.0	95.0	false	no_play
sunny	69.0	70.0	false	will_play
rainy	75.0	80.0	false	will_play
sunny	75.0	70.0	true	may_play
overcast	72.0	90.0	true	may_play
overcast	81.0	75.0	false	will_play
rainy	71.0	91.0	true	no_play

Рис. 1.2. Исходные данные в табличном виде.

1.3. Информация об атрибутах данных

Интерфейс GUI Xelopes позволяет получить подробную информацию о атрибутах загруженных данных. Для этого необходимо нажать на кнопку **Display Data Description** на панели инструментов. Информация представляется в диалоговом окне **Variables** (рис 1.3.). В верхней части окна выводится название данных (на рисунке это weather). В правой части окна представлен список атрибутов. В левой части информация о выбранном атрибуте в зависимости от его типа.

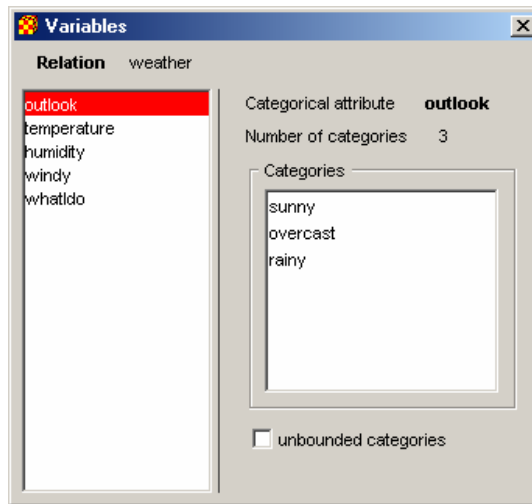


Рис. 1.3. Информация о категориальном атрибуте.

В Xelopes различают два основных типа атрибутов: категориальный и числовой. В зависимости от типа меняется и информация об атрибуте. Для любого атрибута выводится его название и тип.

Для категориальных атрибутов (рис. 1.3.) отображается информация о принимаемых им значениях (категориях): количестве (Number of categories) и списке значений (Categories). Если количество категорий не ограничено, то будет отмечен флаг unbounded categories.

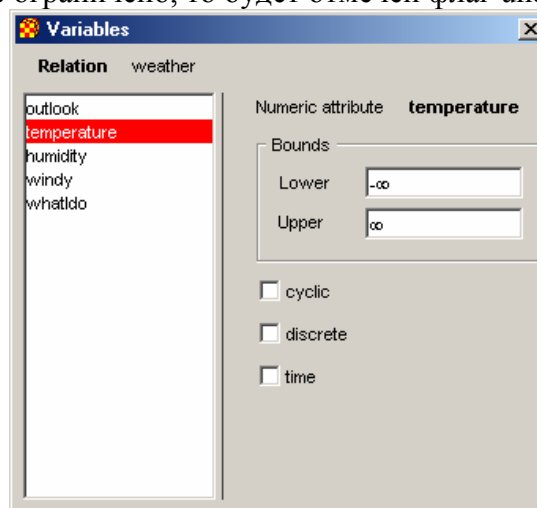


Рис. 1.4. Информация о числовом атрибуте

Для числовых атрибутов (рис. 1.4.) отображается информация о наибольшем (Upper) и наименьшем (Lower) значениях. Кроме того в зависимости от свойств атрибута могут быть установлены следующие флажки:

- Cyclic – если значения атрибута циклические (т. е. может быть определено понятие расстояния)
- Discrete – если значениями атрибута являются дискретные величины
- Time – если атрибут представляет собой время.

1.4. Статистическая информация о данных

Для получения статистической информации о данных необходимо нажать кнопку **Display Descriptive Statistics** на панели инструментов или выбрать пункт меню **File | Statistics**. В открывшемся диалоговом окне **Statistics** (рис.1.5.) необходимо выполнить настройку отображаемой информации.

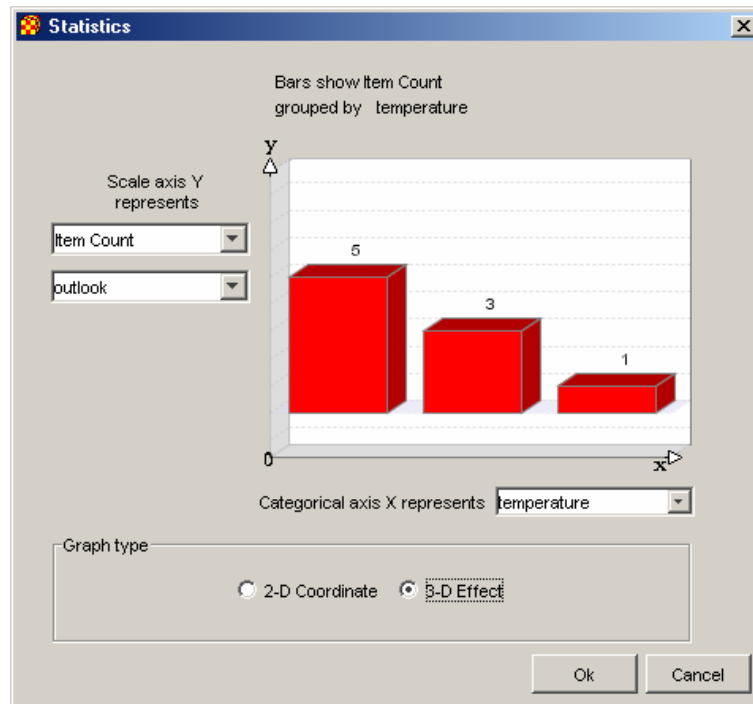


Рис. 1.5. Диалог настроек представления статистической информации по исходным данным.

Необходимо настроить следующие параметры:

- Тип отображаемой информации
- Атрибуты, откладываемые по осям X и Y
- Мерность отображаемой информации: в 2-х или 3-х мерном пространстве.

После настройки необходимых параметров нажатием на кнопку ОК можно получить статическую информацию выбранного типа. Для настроек представленных на рис. 1.5. будет открыто диалоговое окно с информацией изображенное на рис 1.6.

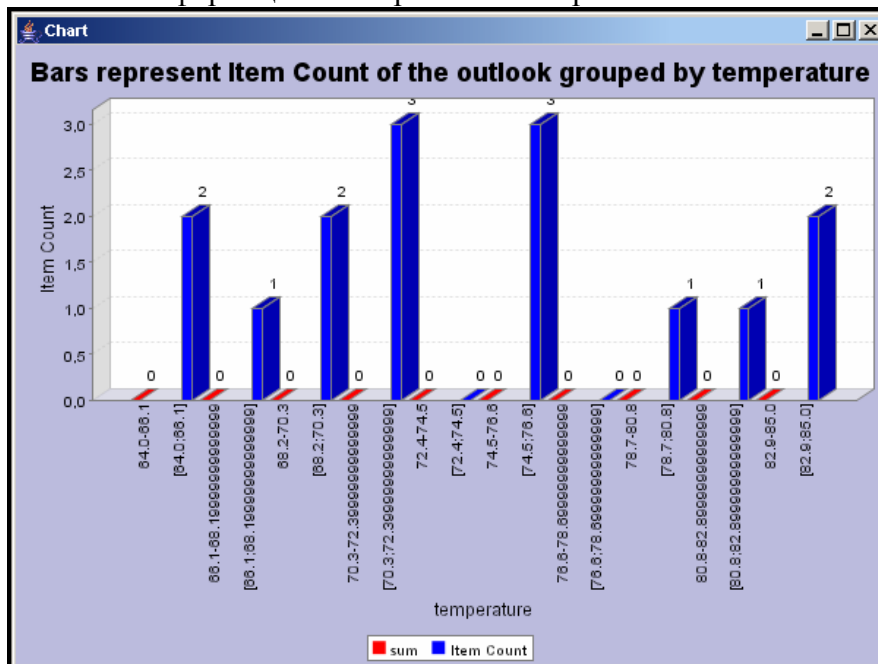


Рис. 1.6. Пример статистической информации по исходным данным.

Можно получить следующие типы информации:

- Количество объектов (Item Count)
- Минимальные (Minimal) и максимальные (Maximal) значения

- Предел (Range) значений
- Сумма (Sum) значений
- Среднее значение (Mean) др.

1.5. Построение *mining* модели

В результате применения методов data mining должна быть построена mining модель. Для этого необходимо нажать кнопку **Build Mining Model** на панели инструментов или выбрать пункт меню **Model | Build**. В результате откроется диалоговое окно предлагающее построить один из типов модели для загруженных ранее данных (рис. 1.7).

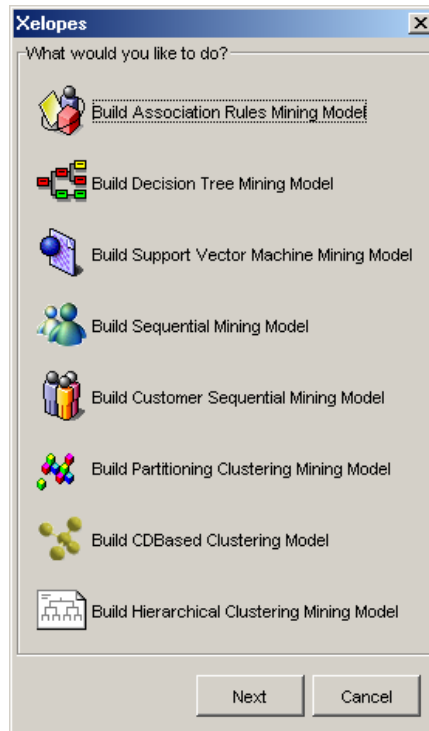


Рис. 1.6. Типы моделей создаваемых алгоритмами библиотеки Xelopes.

Для построения доступны следующие модели:

- ассоциативные правила (Association Rules Mining Model);
- деревья решений (Decision Tree Mining Model);
- математическая зависимость, построенная методом SVM (Support Vector Machine Mining Model);
- последовательности (Sequential Mining Model);
- модель сиквенциального анализа (Customer Sequential Mining Model);
- разделяемая кластерная модель (Partition Clustering Mining Model);
- центрированная кластерная модель (CDBased Clustering Mining Model);
- иерархическая кластерная модель (Hierarchical Clustering Mining Model).

После выбора строящейся модели необходимо выполнить: настройку процесса построения и алгоритм построения (рис. 1.7). Настройки процесса зависят от типа строящейся модели и выполняются на закладке **Settings** (Настройки).

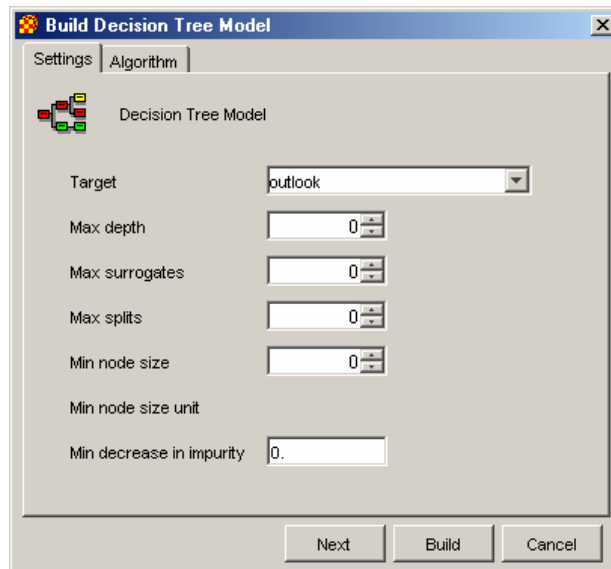


Рис. 1.7. Пример настроек для построения деревьев решений.

Выбор алгоритма выполняется на закладке **Algorithm** (алгоритм) (рис. 1.8.). Список доступных для построения модели алгоритмов зависит от типа модели. Кроме того, для некоторых алгоритмов необходимо выполнить дополнительную настройку. При их выборе в поле Algorithm Parameters появляются поля для определения специфичных для алгоритма настроек.

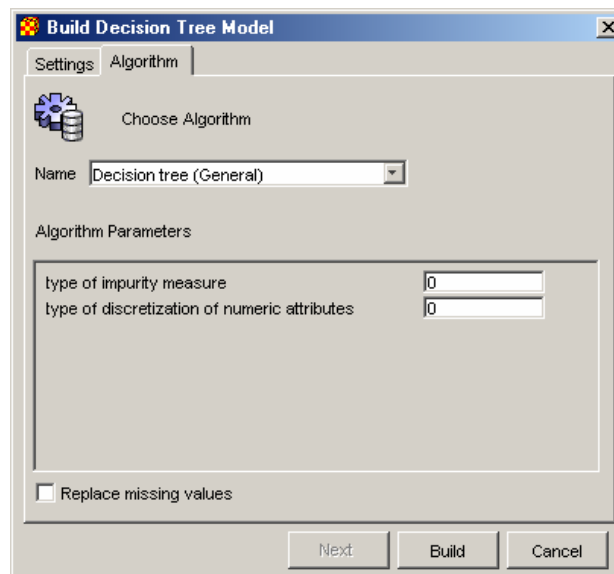


Рис. 1.8. Пример настроек алгоритма построения деревьев решений.

Для построения модели после выполнения настроек необходимо нажать на кнопку **Build** в диалоговом окне. После завершения построения модели появится диалоговое окно (рис. 1.9) предлагающее выполнить следующие действия:

- Визуализировать модель (Browse Model)
- Применить модель (Apply Model)
- Показать модель в виде PMML (View PMML Presentation)
- Записать модель в PMML формате (Save Model as PMML)

Для выполнения перечисленных действий необходимо выбрать соответствующую опцию и нажать на кнопку **Next**. Кроме того, после построения модели на панели инструментов становятся доступными соответствующие кнопки.

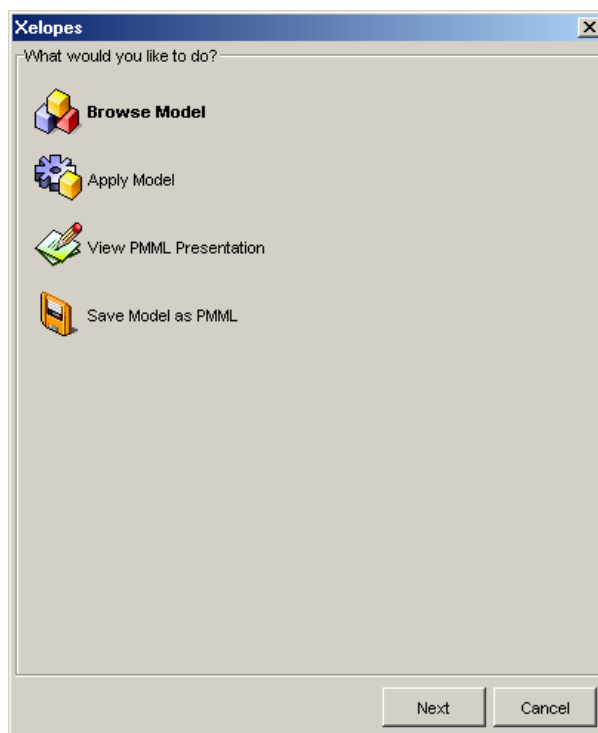


Рис. 1.9. Действия выполняемые с построенной моделью

В данной версии GUI Xelopes визуализируются только три вида моделей:

- Ассоциативные правила
- Деревья решений
- Иерархическая кластерная модель в виде дейтограмм.

Для остальных моделей при попытке визуализации происходит отображение модели в формате PMML. То есть для них действия **Browse Model** и **View PMML Presentation** будут иметь одинаковый результат.

1.7. Представление модели в формате PMML

Для представления модели в формате PMML необходимо нажать кнопку **View PMML Presentation** на панели инструментов или выбрать пункт меню **Model | View PMML** или выбрать опцию **View PMML Presentation** в диалоговом окне представленном на рис. 1.9. В результате будет открыто окно в котором будет представлена построенная модель в формате PMML в текстовом виде (1.10).

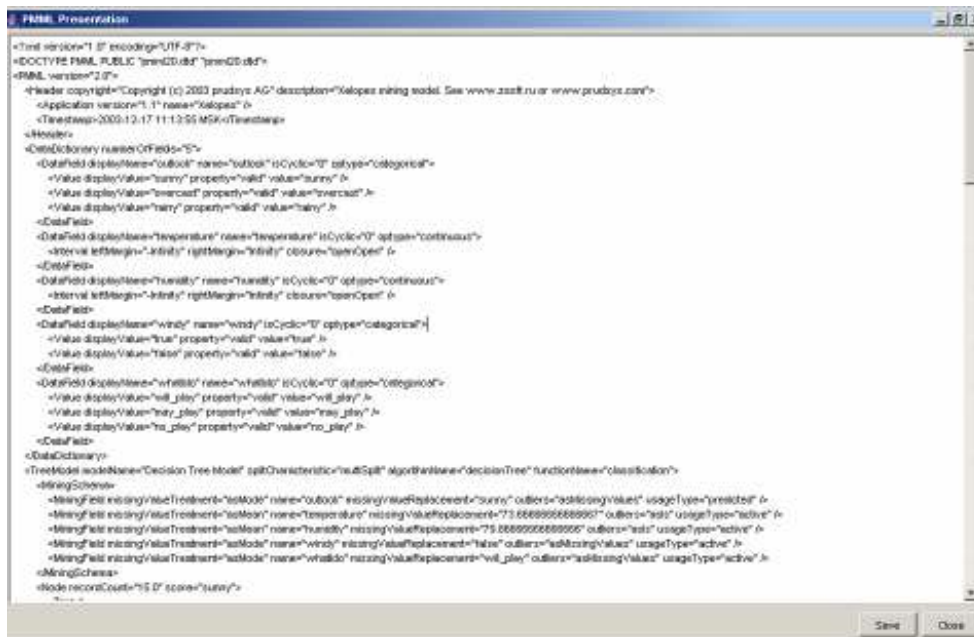


Рис. 1.10. Представление модели в PMML формате.

Представленную модель можно сохранить нажав в открытом окне кнопку Save. Кроме того, модель можно сохранить, нажав кнопку **Save Model as PMML** на панели инструментов или выбрав пункт меню **Model | Save** или опцию **Save Model as PMML** в диалоговом окне представленном на рис. 1.9.

1.8. Применение модели

Модели строящиеся для задач классификации и регрессии используются для предсказательных целей на новых данных. Следовательно они могут быть применены к другим данным. Для этого необходимо нажать кнопку **Apply Model** на панели инструментов или выбрав пункт меню **Model | Apply** или опцию **Apply Model** в диалоговом окне представленном на рис. 1.9. В результате будет предложено выбрать файл с новыми данными, записанными в формате arff (будет открыт диалоговое окно аналогичное представленному на рис. 1.1.). После выбора файла и применения построенной модели будет отображено окно в котором новые данные будут представлены в табличном виде (рис. 1.11).

outlook	temperature	humidity	windy	whatdo	predicted_outlook
overcast	75.0	55.0	false	will_play	overcast
sunny	85.0	85.0	false	will_play	sunny
sunny	80.0	90.0	true	may_play	sunny
overcast	83.0	86.0	false	no_play	overcast
rainy	70.0	96.0	false	will_play	rainy
rainy	68.0	80.0	false	will_play	rainy
rainy	65.0	70.0	true	no_play	rainy
overcast	64.0	65.0	true	may_play	overcast
sunny	72.0	95.0	false	no_play	sunny
sunny	69.0	70.0	false	will_play	sunny
rainy	75.0	80.0	false	will_play	rainy
sunny	75.0	70.0	true	may_play	sunny
overcast	72.0	90.0	true	may_play	overcast
overcast	81.0	75.0	false	will_play	overcast
rainy	71.0	91.0	true	no_play	rainy

Рис. 1.11. Результат применения модели к новым данным.

В открывшемся окне в виде таблицы будут представлены классифицированные данные. Как видно таблица алогична той же представлена на рис. 1.2. Разница заключается в новой колонке **predicted_*** описывающей результат классификации (* - заменяется на

атрибут классификации). В окне также выводится информация о степени ошибки классификации (Error rate).

Порядок выполнения работы

1. Открыть GUI интерфейс библиотеки Xelopes.
2. Загрузить исходные данные из файла transact.arff.
3. Просмотреть загруженные данные.
4. Просмотреть информацию об атрибутах данных.
5. Просмотреть статистическую информацию о данных.
6. Построить модель Association Rules Mining Model.
7. Визуализировать построенную модель.
8. Просмотреть и сохранить модель в формате PMML.
9. Выполнить пункты 2-5 для данных из файла weather-nominal.arff.
10. Выполнить пункты 6 - 8 для модели Decision Tree Mining Model.
11. Применить модель к данным из файла weather-nominal.arff
12. Выполнить пункты 6 - 8 для модели Hierarchical Clustering Mining Model.

Отчет по работе

1. Цель работы.
2. Данные из файлов transact.arff и weather.arff
3. Информация об атрибутах данных из файлов transact.arff и weather.arff
4. Статистическая информация по данным из файлов transact.arff и weather-nominal.arff
5. Скриншоты визуализации моделей Association Rules Mining Model, Decision Tree Mining Model и Hierarchical Clustering Mining Model.
6. Результат применения модели Decision Tree Mining Model к данным из файла weather-nominal.arff
7. Выводы по работе.

Лабораторная работа №2

Выполнение анализа данных методами data mining.

Цель работы: Изучить основные этапы интеллектуального анализа данных с использованием алгоритмов data mining реализованных в библиотеке Xelopes.

Задание: Для данных из файла определенных вариантом задания построить модели также в соответствии с вариантом задания помощью различных алгоритмов и объяснить результаты.

Общие сведения

2.1. Введение

Процесс интеллектуального анализа данных состоит из следующих основных этапов:

1. Подготовка данных в виде удобном для применения методов data mining
2. Настройка процесса построения mining модели
3. Построение mining модели
4. Анализ построенной mining модели
5. В случае supervised модели применение ее к новым данным

2.1 Подготовка исходных данных

Процесс подготовки предполагает сбор данных для анализа из разных источников данных и представления их в формате пригодном для применения алгоритмов data mining.

Настоящая версия Xelopes поддерживает ARFF (Attribute-Relation File Format) формат представления данных. Он разработан для библиотеки Weka в университете Waikato. ARFF файл является ASCII текстовым файлом, описывающем список объектов с общими атрибутами.

Структурно такой файл разделяется на две части: заголовок и данные.

В заголовке описывается имя данных и их метаданные (имена атрибутов и их типы).

Например,

```
@relation weather
```

```
@attribute outlook {sunny, overcast, rainy}
```

```
@attribute temperature real
```

```
@attribute humidity real
```

```
@attribute windy {true, false}
```

```
@attribute whatI do {will_play, may_play, no_play}
```

Во второй части представлены сами данные. Например,

```
@data
```

```
overcast,75,55,false,will_play
```

```
sunny,85,85,false,will_play
```

```
sunny,80,90,true,may_play
```

```
overcast,83,86,false,no_play
```

```
rainy,70,96,false,will_play
```

```
rainy,68,80,false,will_play
```

```
rainy,65,70,true,no_play
```

```
overcast,64,65,true,may_play
```

```
sunny,72,95,false,no_play
```

```
sunny,69,70,false,will_play
```

```
rainy,75,80,false,will_play
```

```
sunny,75,70,true,may_play
```

```
overcast,72,90,true,may_play
```

```
overcast,81,75,false,will_play
```

```
rainy,71,91,true,no_play
```

2.2.1 Заголовок

Заголовок содержит информацию об имени файла и метаданные о представленных в нем данных. Имя описывается в следующем формате

```
@relation <имя>
```

Имя может быть любая последовательность символов. Если имя включает пробелы то оно должно быть заключено в кавычки. Например

```
@relation weather  
@relation "weather nominal"
```

Мета данные описывают атрибуты представленных в файле данных. Информация о каждом атрибуте записывается в отдельной строке и включает в себя имя атрибута и его тип. Очевидно, что имена должны быть уникальны. Порядок их описания должен совпадать с порядком колонок описания данных. Общий формат описания атрибута следующий:

```
@attribute <имя атрибута> <тип атрибута>
```

Например,

```
@attribute outlook {sunny, overcast, rainy}  
@attribute temperature real
```

Имя атрибута должно начинаться с символа. В случае если оно содержит пробелы, то должно быть заключено в кавычки.

Значением поля <тип> может быть одно из следующих пяти типов:

- real
- integer
- <категория>
- string
- date [<формат даты>]

Типы real и integer являются числовыми. Категориальные типы описываются перечислением категорий (возможных значений). Например:

```
@attribute outlook {sunny, overcast, rainy}
```

При описании даты можно указать формат в котором она будет записываться (например, "yyyy-MM-dd").

2.2.2. Данные

Данные представляются в ARFF формате в виде списка значений атрибутов объектов после тэга **@data**. Каждая строка списка соответствует одному объекту. Каждая колонка соответствует атрибуту описанному в части заголовка. Причем порядок следования колонок должен совпадать с порядком описания атрибутов. Например:

```
@data  
overcast,75,55,false,will_play  
sunny,85,85,false,will_play  
sunny,80,90,true,may_play
```

Часто в терминологии data mining такие строки называют векторами.

Данные могут содержать пропущенные (неизвестные) значения. В ARFF они представляются символом «?», например:

```
@data  
4.4,?,1.5,?,Iris-setosa
```

Строковые данные в случае если они содержат разделяющие слова символы, должны заключаться в кавычки. Например,

```
@relation LCCvsLCSH
```

```
@attribute LCC string  
@attribute LCSH string
```


@data

AG5, 'Encyclopedias and dictionaries.;Twentieth century.'

AS262, 'Science -- Soviet Union -- History.'

AE5, 'Encyclopedias and dictionaries.'

AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'

AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'

Даты также должны быть заключены в кавычки. Если при описании соответствующего атрибута бы указан формат даты, то данные должны быть записаны в соответствии с ним:

@relation Timestamps

@attribute timestamp DATE "yyyy-MM-dd HH:mm:ss"

@data

"2001-04-03 12:12:12"

"2001-05-03 12:59:55"

2.3. Настройка процесса построения mining модели

Результатом анализа данных с помощью методов data mining являются структуры представляющие собой новые знания. Такие структуры называются моделями. Они могут быть разных видов: правила классификации, ассоциативные правила, деревья решений, математические зависимости и т.п. Вид модели во многом зависит от метода с помощью которого она была построена. Таким образом, конечный результат зависит от метода и исходных данных. Кроме того, процесс построения моделей можно настроить изменяя тем самым свойства модели (точность, глубина дерева и т.п.). Настраиваемые параметры зависят от конкретной модели.

В GUI Xelopes пользователь имеет возможность выполнить настройки для каждой строящейся модели индивидуально. Этот процесс осуществляется в диалоговом окне настроек, описанном в предыдущей лабораторной работе на закладке Settings. Далее рассмотрим более подробно настройки для каждой модели.

2.3.1. Настройки для ассоциативных правил и сиквенциального анализа

Настройки для модели представляющей ассоциативные правила выполняются в диалоговом окне изображенном на рис. 2.1.

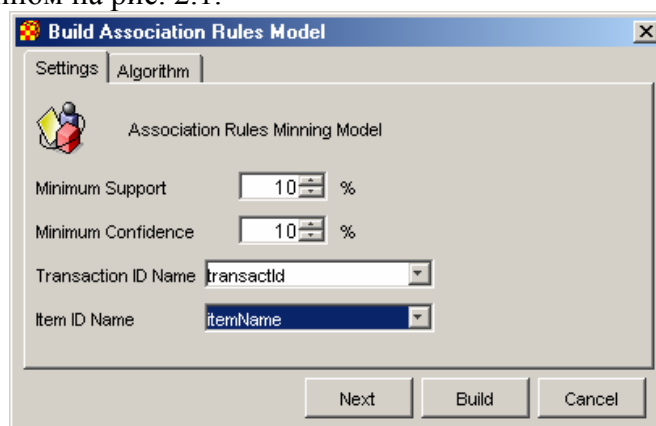


Рис. 2.1. Настройки модели ассоциативных правил

В нем выполняется настройка следующих параметров:

- **Minimum Support** – минимальное значение поддержки для искомым частых наборов и строящихся ассоциативных правил. Значение должно быть больше нуля, иначе не будет построено не одного правила.

- **Minimum Confidence** – минимальное значение доверия для строящихся ассоциативных правил. Значение должно быть больше нуля, иначе не будет построено не одного правила.
- **Transaction ID Name** – атрибут уникально идентифицирующий транзакции (ключевое поле).
- **Item ID Name** – атрибут представляющий собой имена объектов. Они используются для построения правил. От его выбора зависит степень понимания полученных результатов.

Настройки для сиквенциальной модели выполняются в диалоговом окне изображенном на рис. 2.2.

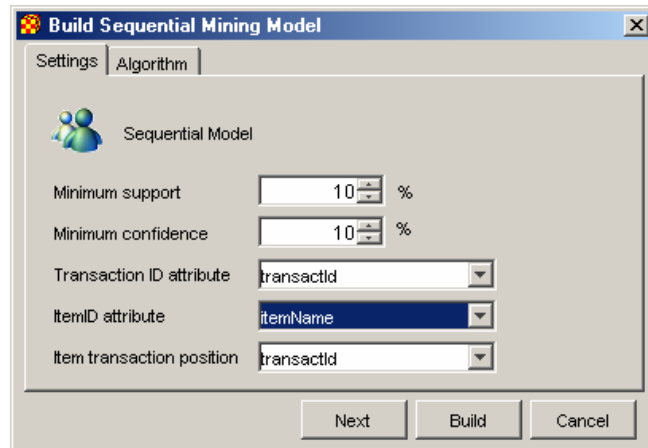


Рис. 2.2. Настройки сиквенциальной модели

В нем выполняется настройка аналогичные модели ассоциативных правил. Дополнительно появляется параметр **Item transaction position** представляющий атрибут, идентифицирующий позицию элемента в последовательности.

2.3.2. Настройки для деревьев решений (Decision Tree Mining Model)

Настройки для модели представляющей деревья решений выполняются в диалоговом окне изображенном на рис. 2.3.

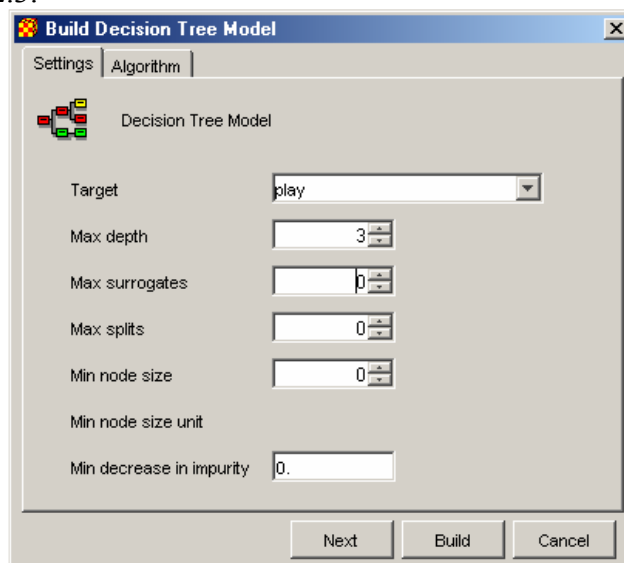


Рис. 2.3. Настройки модели деревьев решений

В нем выполняется настройка следующих параметров:

- **Target** – атрибут по которому выполняется классификация данных (независимая переменная).

- **Max depth** – максимально допустимая глубина строящегося дерева
- **Max surrogates** - максимально допустимое число замен
- **Max splits** - максимально допустимое количество расщеплений
- **Min node size** – минимальный размер узла дерева
- **Min decrease in impurity** – минимальная степень примесей

2.3.3. Настройки для математической зависимости построенной методом SVM

Настройки для модели представляющую математическую зависимость, построенную методом SVM, выполняются в диалоговом окне изображенном на рис. 2.4.

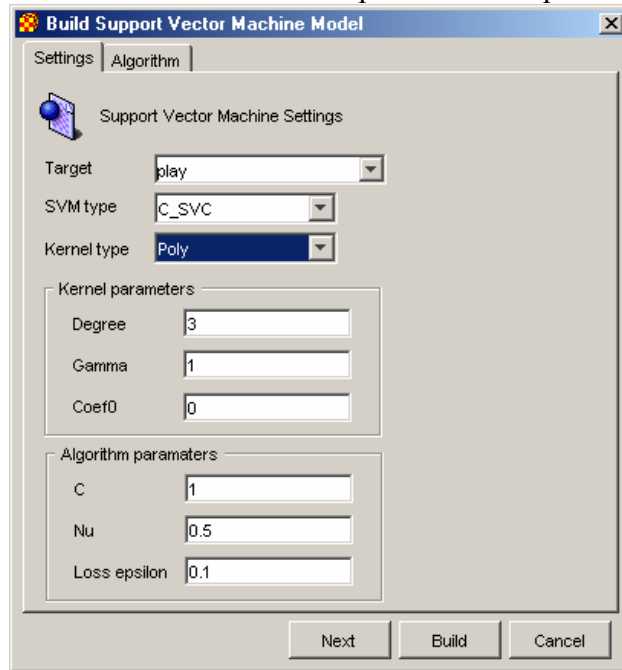


Рис. 2.4. Настройки модели SVM

В нем выполняется настройка следующих параметров:

- **Target** – атрибут по которому выполняется классификация данных (независимая переменная).
- **SVM Type** – тип модели SVM. В Xelopes могут быть построены следующие типы: C-SVC (classical SVM), Nu-SVC, one-class SCM, Epsilon-SVR (classic regression SVM), Nu-SVR. Они отличаются классификационной функцией. Так наиболее распространенная SVM для задачи регрессии Epsilon-SVR имеет функцию вида:

$$f(x, \alpha, \alpha^*) = \sum_{i=1}^M (\alpha_i^* - \alpha_i) K(x, x_i) + b$$

в то время как SVM для классификации имеет вид

$$f(x, \alpha) = \sum_{i=1}^M \alpha_i K(x, x_i) + b$$

- **Kernel Type** – вид функции $K(x, x_i)$ в классификационной функции (тип ядра). Может принимать следующие значения:
 - **Linear** - Линейная - $k(x,y) = x*y$
 - **Poly** - Полиномиал степени d - $k(x,y) = (\gamma * x*y + c_0)^d$
 - **RBF**- Базовая радиальная функция Гаусса - $k(x,y) = \exp(-\gamma ||x - y||)$
 - **Sigmoid** - Сигмоидальная $k(x,y) = \tanh(\gamma * x*y + c_0)$
- **Kernel Parameters** – параметры ядра, зависят от выбранного типа ядра.
 - Degree – степень d в ядре poly;
 - Gamma – параметр γ в последних трех видах;
 - Coef0 – коэффициент c_0 в типах poly и sigmoid.
- **Algorithm Parameters** – общие параметры алгоритмов класса SVM:

- C – инверсный регулирующий параметр $C = \frac{1}{2\lambda M}$;
- ν – параметр ν в типе Nu – SVM;
- Loss epsilon – ϵ функция потерь в типе Epsilon-SVR.

2.3.4. Настройки кластерной модели

Настройки для кластерных центрированной и иерархических моделей выполняются в диалоговом окне изображенном на рис. 2.5.

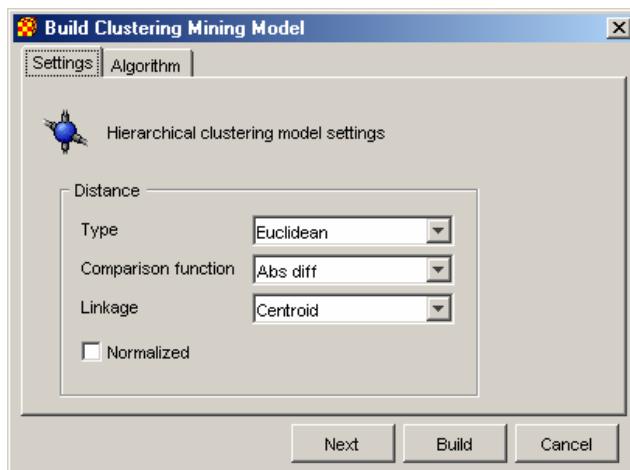


Рис. 2.5. Настройки для кластерной модели.

В нем выполняется настройка следующих параметров:

- **Maximum number of clusters** – максимальное количество построенных кластеров. Значение параметра должно быть больше нуля.
- **Distance** – параметры характеризующие функцию вычисления расстояния между объектами:
 - **Type** – тип функции расстояния. Xelopes (Евклидово – Euclidean, Чебышева – Chebyshev и др.)
 - **Comparison function** – функция сопоставления.
 - **Normalized** – использовать ли нормализацию при расчете расстояний.

Настройки для разделяемой кластерной модели выполняются в диалоговом окне изображенном на рис. 2.6.

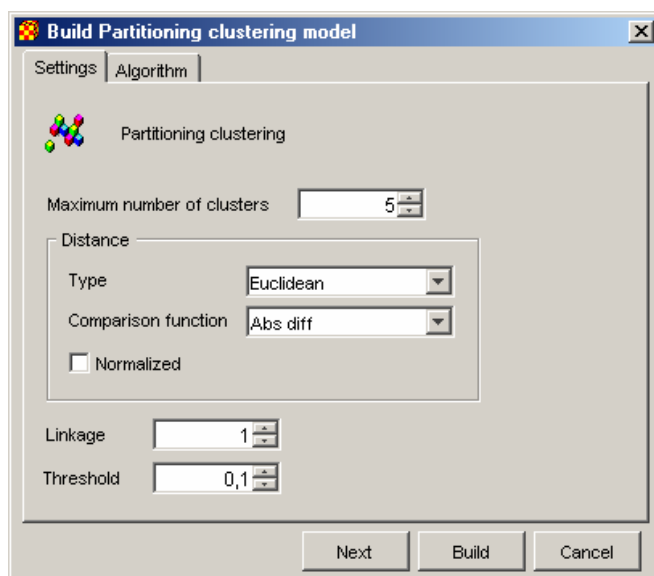


Рис. 2.6. Настройки для разделяемой кластерной модели.

В нем выполняется настройка дополнительных параметров параметров:

- **Linkage** – параметр k для алгоритма k-linkage.
- **Threshold** – предел для расстояния.

2.4. Анализ моделей

Для применения полученных с помощью методов data mining знаний необходимо проанализировать построенные модели. При анализе необходимо проверить насколько полученные знания являются логически объяснимыми, не противоречат ли они здравому смыслу, действительно ли они являются новыми и т.п. Кроме того, модели, строящиеся при решении задач ассоциативного анализа и кластеризации, являются описательными, т.е. служат для лучшего понимания самих данных. В связи с этим можно сделать вывод, что важным является представление моделей в виде удобном для их анализа человеком.

В GUI Xelopes любая модель может быть представлена в формате PMML. Это стандартизированный формат основанный на формате XML. К сожалению для визуального анализа данный формат довольно сложен. По этой причине в GUI Xelopes реализованы специальные средства визуализации для трех основных видов моделей:

- ассоциативные правила;
- деревья решений;
- дейтограммы.

2.4.1 Визуализация ассоциативных правил

Модель представляющая ассоциативные правила в GUI Xelopes представляется в виде 3-х мерных гистограмм (рис. 2.7.). По осям плоскости откладываются подмножества частых наборов. LHS – означает левую часть правил, RHS – правую. На их пересечении рисуется гистограмма. По умолчанию высота гистограмма отражает уровень поддержки правила включающего в условную и заключительную части данные наборы. Цвет от синего к красному (от меньшего к большему) уровень доверия.

Например, для правила *Если (Nut) то (Coke)* нарисована красная высокая гистограмма означающая что данное правило имеет наибольшую степень поддержки с высокой степенью доверия. Правило *Если (Water) то (Cracker, Coke)* имеет низкий уровень поддержки и низкую степень доверия.

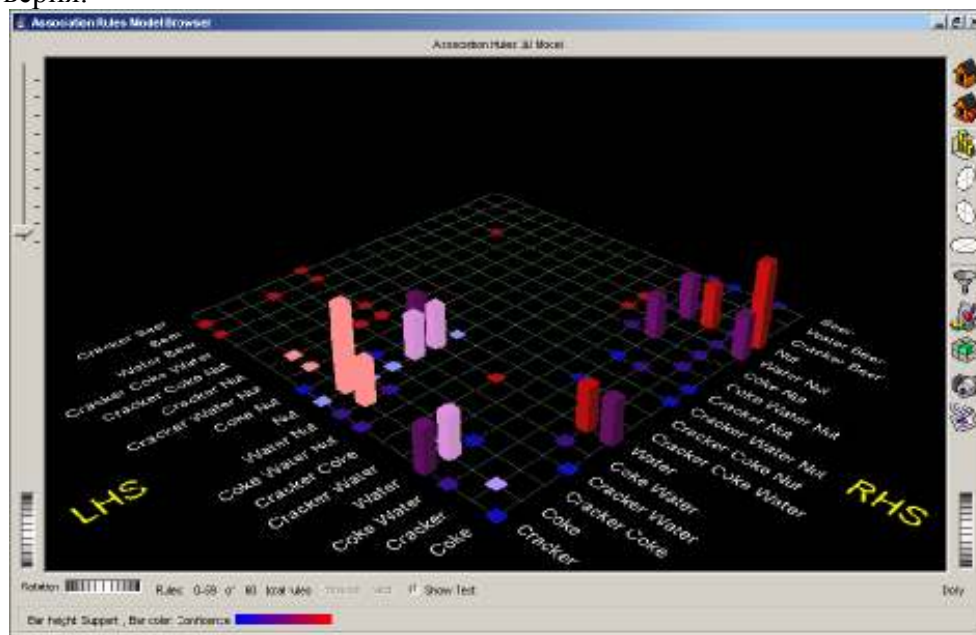



Рис. 2.7. Визуальное представление ассоциативных правил.

Для более детального изучения правил необходимо выделить гистограмму на пересечении интересующих наборов. Визуально они подсвечиваются более ярким цветом. На рис. Выделена гистограмма Nut – Coke. Для выделенных наборов можно более детально посмотреть гистограммы их оценок. Для этого необходимо на панели инструментов находящейся слева от диаграмма нажать кнопку . В результате появится диалог изображенный на рис. 2.8.

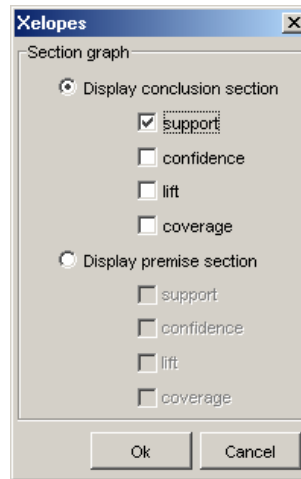


Рис. 2.8. Диалог для детализации оценок ассоциативных правил.

В нем можно выбрать какие оценки должны быть детализированы. После нажатие на кнопку ОК, появятся диаграммы (Рис. 2.9) оценок для выделенных наборов.

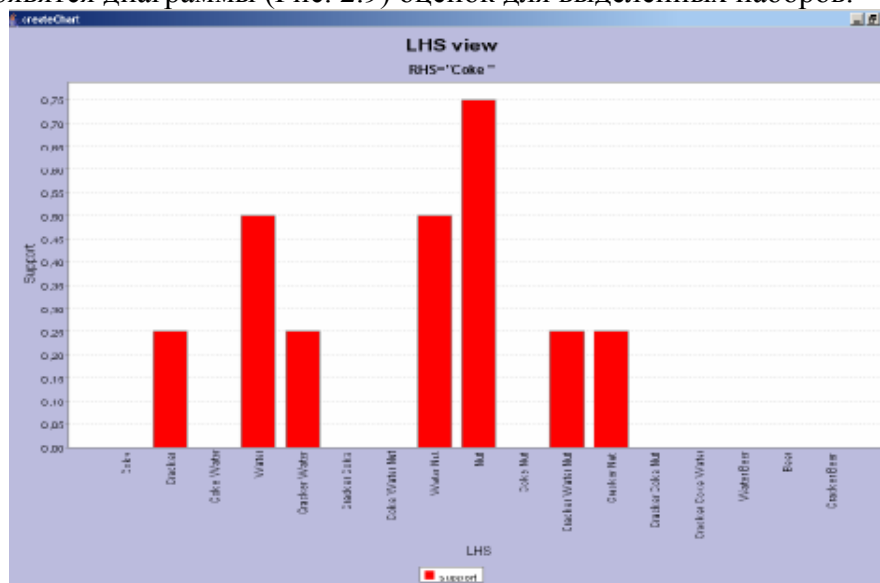


Рис. 2.9. Пример диаграммы поддержки

2.4.2. Визуализация деревьев решений

Модель представляющая деревья решений в GUI Xelopes представляется в дерева (рис. 2.10.). Узлами дерева являются выражения определяющие разбиения множества объектов на подмножества. Нижняя часть в каждом узле отображает уровень вхождения в множество соответствующее узлу объектов относящихся к разным классам. Можно заметить, что листья дерева соответствующие подмножествам содержащим объекты одного класса имеют одноцветную нижнюю часть. Подписи на ветвях дерева отражают условия перехода по этой ветви.

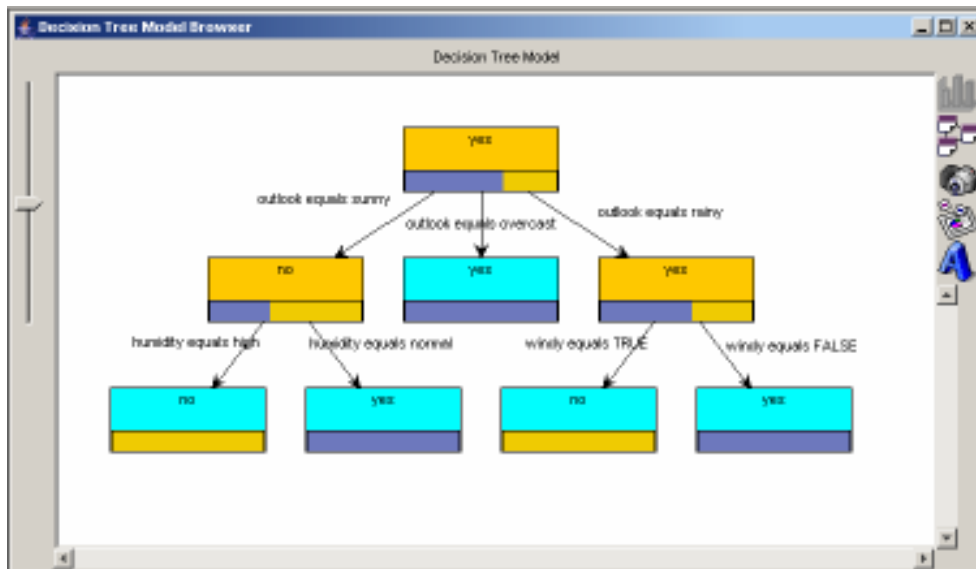


Рис. 2.10. Пример визуализации модели дерева решений

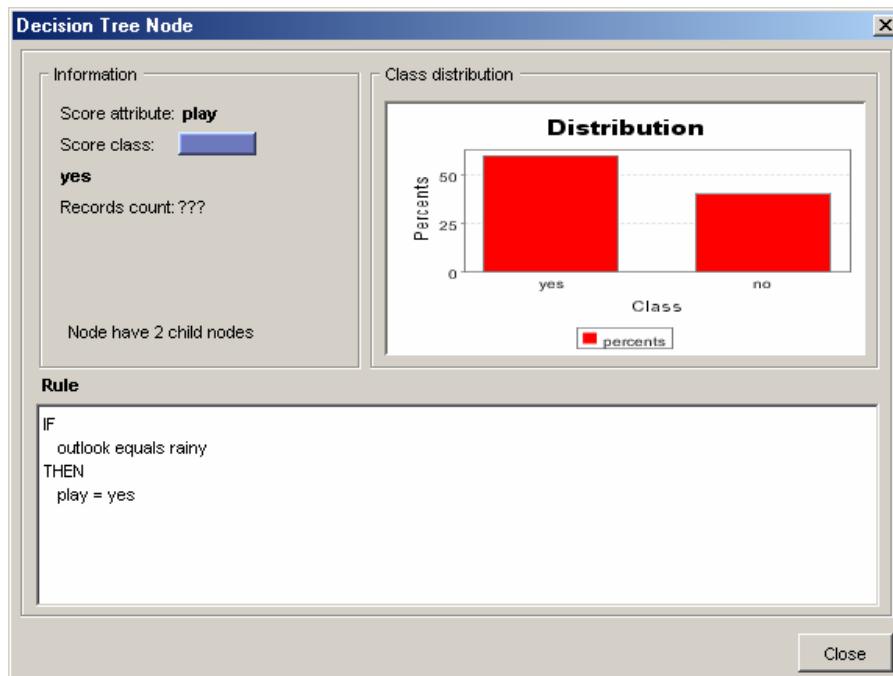


Рис. 2.11. Пример визуализации модели дерева решений

По каждому узлу дерева можно получить дополнительную информацию. Для этого необходимо выделить узел и или выбрав в контекстном меню пункт **Node Information** или нажать на кнопку **Node Info** на панели инструментов слева от диаграммы. В результате появится окно (рис. 2.11) представляющую следующую информацию об узле:

- **Information** – информация об узле:
 - **Score attribute** – сравниваемый атрибут (зависимая переменная)
 - **Score class** – значение с которым выполняется сравнение
 - **Records count** – количество объектов покрываемых узлом
 - Количество ветвей выходящих из узла.
- **Class distribution** – распределение объектов относящихся к разным классам для данного узла
- **Rule** – классификационное правило соответствующее данному узлу.

2.4.3. Визуализация иерархической кластеризации

Модель представляющая иерархическую кластеризацию решений в GUI Xelopes представляется в виде дейтограммы (рис. 2.12.). Верхний узел представляет собой кластер соответствующий всему множеству объектов. Листья соответствуют кластерам содержащим по одному элементу из исходного множества.

С помощью мыши можно задать уровень кластеризации. На дейтограмме он представляется в виде линии. При этом будет выводиться информация о среднем расстоянии между кластерами. Объединяемые кластеры при заданном уровне выделяются красным цветом.

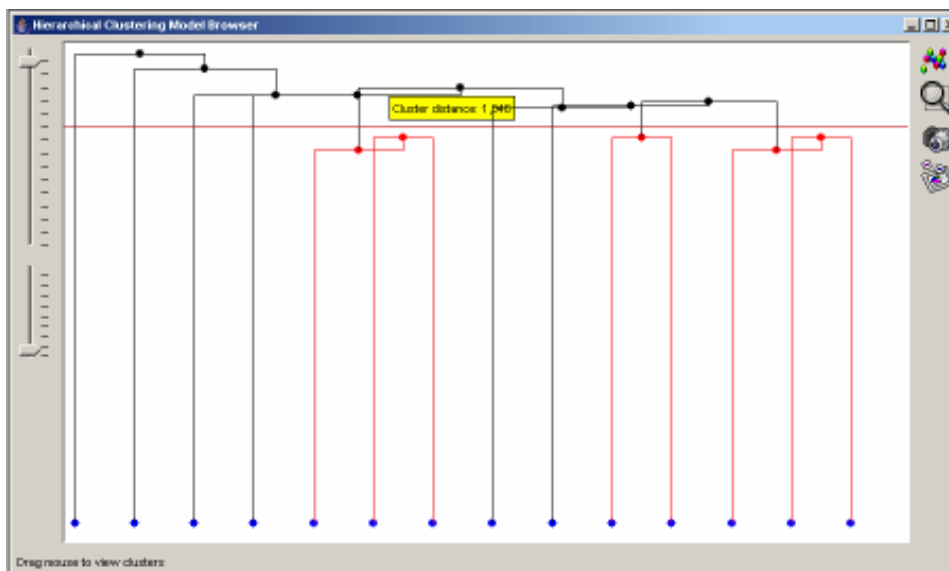



Рис. 2.12. Пример визуализации дейтограммы

По кластеризации можно получить более детальную информацию, нажав на кнопку  на панели инструментов слева от диаграммы. В результате появится окно представленное на рис. 2.13.

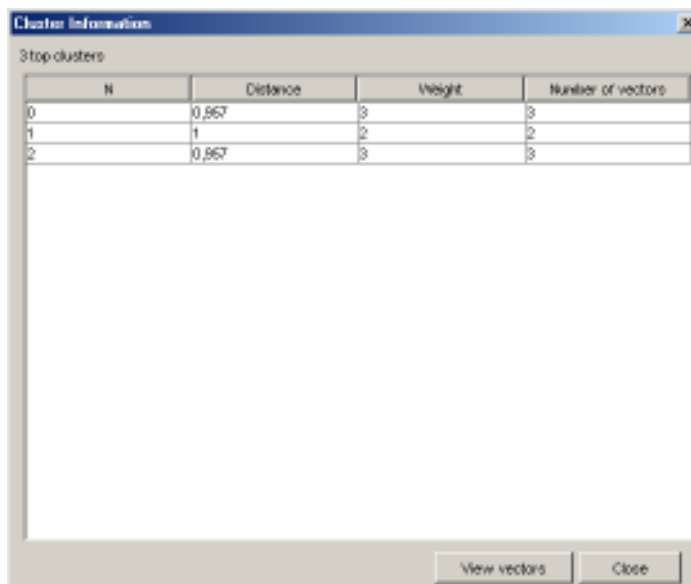


Рис. 2.13. Детализированная информация о кластерах.

В окне в табличном виде отображается информация о кластерах для заданного уровня. Над таблицей отображается информация о количестве кластеров. Колонки в таблице содержат следующую информацию:

- **N** – номер кластеров
- **Distance** – расстояние между кластерами.

- **Weight** – вес кластера (в данном случае количество объектов попавших в кластер)
- **Number of vectors** - количество объектов попавших в кластер.

Нажав на кнопку View vectors можно просмотреть информацию об исходных данных.

Порядок выполнения работы

13. Подготовить новые данные в формате ARFF имеющую ту же структуру, что и данные в файле заданном вариантом задания без значений независимой переменной.
14. Открыть GUI интерфейс библиотеки Xelopes.
15. Загрузить исходные данные из файла указанного в варианте задания.
16. Просмотреть загруженные данные.
17. Просмотреть информацию об атрибутах данных.
18. Просмотреть статистическую информацию о данных.
19. Поочередно попытаться построить модели определенные вариантом задания каждым из доступных алгоритмов для разных параметров настройки.
20. Визуализировать и сравнить модели, построенные разными алгоритмами.
21. Просмотреть и сохранить построенные модели в формате PMML.
22. Применить модели типа supervised к данным, подготовленным на шаге 1.

Варианты задания

Вариант	Файл	Модели
1	contact-lenses.arff	Sequential Mining Model, Decision Tree Mining Model, Hierarchical Clustering Mining Model
2	weather.arff	Customer Sequential Mining Model Support Vector Machine Mining Model, CDBased Clustering Mining Model
3	weather-nominal.arff	Association Rules Mining Model Decision Tree Mining Model Hierarchical Clustering Mining Model
4	iris.arff	Association Rules Mining Model Support Vector Machine Mining Model Partition Clustering Mining Model
5	iris-transact.arff	Customer Sequential Mining Model, Support Vector Machine Mining Model, Hierarchical Clustering Mining Model
6	iris-nontransact.arff	Association Rules Mining Model Decision Tree Mining Model Partition Clustering Mining Model
7	transact.arff	Association Rules Mining Model, Support Vector Machine Mining Model CDBased Clustering Mining Model
8	custom-	Association Rules Mining Model

	transact.arff	Decision Tree Mining Model CDBased Clustering Mining Model
--	---------------	--

Отчет по работе

1. Цель работы.
2. Данные из файла определенного вариантом задания и информация о них.
3. Список моделей, которые не удалось построить для данных с пояснениями почему.
4. Для каждой модели список алгоритмов, которые не построили модель для данных с пояснениями почему.
5. Каждую модель, построенную разными алгоритмами с описанием различий между ними и пояснениями.
6. Модели, построенные одним алгоритмом при разных параметрах настройки с описанием различий и пояснениями.
7. Результат применения модели типа supervised к новым данным.
8. Выводы по работе.

Лабораторная работа №3

Создание программ анализа данных с использованием алгоритмов data mining.

Цель работы: Изучить основные принципы создания систем интеллектуального анализа данных с использованием алгоритмов data mining.

Задание: Реализовать программу, выполняющую анализ данных представленных в формате ARFF с помощью алгоритма data mining и строящую модель, заданную вариантом задания.

Общие сведения

3.1. Введение

Системы поддержки принятия решений (СППР) используют методы data mining для анализа данных и получения новых знаний. Полученные результаты могут носить как описательный характер, позволяющие лучше понять данные, так и предсказательный, позволяющие в дальнейшем предсказывать значение каких либо параметров на основании найденных закономерностей. К первому виду относятся методы выполняющие поиск ассоциативных правил и кластеризацию. Ко второму методы строящие функции классификации и регрессии.

Библиотека Xelopes в своем составе имеет алгоритмы обоих типов. Она может служить основой для построения СППР. Структура и поддержка основных стандартов библиотекой делают ее использование достаточно простым.

Основной подход решения задач data mining с помощью библиотеки Xelopes не зависит от вида или используемого метода.

Первоначально создается экземпляр класс *MiningInputStream* для загрузки исходных данных, например, из файла формата ARFF.

```
MiningInputStream inputData = new MiningArffStream("data.arff");
```

Затем выделяются метаданные загруженных данных.

```
MiningDataSpecification metaData = inputData.getMetaData();
```

Далее создается экземпляр класса *MiningAlgorithm*. Для настройки процесса построения модели создаются экземпляры классов *MiningSettings* и *MiningAlgorithmSpecification*. У данных экземпляров устанавливаются необходимые параметры. Создание конкретных экземпляров классов зависят от используемого метода и решаемой задачи.

Сделанные настройки должны быть проверены с помощью метода: `verifySettings()`:

```
miningSettings.verifySettings();
```

Необходимо заметить, что настройки специфичные для алгоритма могут быть выполнены как непосредственно в коде, так и загружены из конфигурационного файла `algorithms.xml` по имени алгоритма.

Созданному экземпляру алгоритма передаются исходные данные и настройки.

```
algorithm.setMiningInputStream( inputData );  
algorithm.setMiningSettings( miningSettings );  
algorithm.setMiningAlgorithmSpecification(  
    miningAlgorithmSpecification );
```

Затем вызывается метод *buildModel()* построения модели, который возвращает построенную модель в виде экземпляра класса *MiningModel*.

```
MiningModel model = algorithm.buildModel();
```

Любая построенная модель может быть сохранена в формате PMML

```
FileWriter writer = new FileWriter("example.xml");  
model.writePmml(writer);
```

или в текстовом формате

```
writer = new FileWriter("example.txt");  
model.writePlainText(writer);
```

Если построенная модель является экземпляром класса *SupervisedMiningModel*, то следовательно она может быть применена к новым данным с целью определения значения зависимой переменной.

```
MiningVector vector = inputData.read();  
double predicted = model.applyModelFunction(vector);
```

Далее рассмотрим особенности решения задач поиска ассоциативных правил, кластеризации и классификации.

3.2. Поиск ассоциативных правил

Для настройки процесса поиска ассоциативных правил создается экземпляр класса *AssociationRulesSettings*.

```
AssociationRulesSettings miningSettings = new AssociationRulesSettings();
```

Ему передаются метаданные загруженных данных:

```
miningSettings.setDataSpecification( metaData );
```

Для поиска ассоциативных правил важно указать какие из атрибутов идентифицируют элементы, а какие транзакции, в которые они входят. Например,

```
CategoricalAttribute categoryItemId =  
(CategoricalAttribute) metaData.getMiningAttribute("itemId");  
miningSettings.setItemId( categoryItemId );
```

```
CategoricalAttribute categoryTransactId =  
(CategoricalAttribute) metaData.getMiningAttribute("transactId");  
miningSettings.setTransactionId( categoryTransactId );
```

Также для поиска необходимо определить минимальную поддержку и минимальную степень доверия для искомым правил. Например,

```
miningSettings.setMinimumConfidence( 0.30 );  
miningSettings.setMinimumSupport( 0.5 );
```

Для решения задачи поиска ассоциативных правил можно использовать алгоритм Apriori. В библиотеке Xelopes он реализован в классе *com.prudsys.pdm.Models.AssociationRules.Algorithms.AprioriSimple*. Для его

использования необходимо сделать специфичные для него настройки. Для этого создается экземпляр класса *MiningAlgorithmSpecification*:

```
MiningAlgorithmSpecification miningAlgorithmSpecification = new  
    MiningAlgorithmSpecification();
```

Указываются имя, функция, используемый алгоритм, версия алгоритма и класс его реализующий. Например:

```
miningAlgorithmSpecification.setName("AprioriSimple");  
miningAlgorithmSpecification.setFunction("AssociationRules");  
miningAlgorithmSpecification.setAlgorithm("associationRules");  
miningAlgorithmSpecification.setClassname("com.prudsys.pdm.Models.AssociationRules.Algorith  
    ms.AprioriSimple.Apriori");  
miningAlgorithmSpecification.setVersion("1.0");
```

Далее устанавливаются дополнительные параметры с помощью массива экземпляров класса *MiningAlgorithmParameter*:

```
MiningAlgorithmParameter[] miningAlgorithmParameter =  
    new MiningAlgorithmParameter[3];
```

Устанавливается минимальный размер набора. Например,

```
miningAlgorithmParameter[0] = new MiningAlgorithmParameter();  
miningAlgorithmParameter[0].setName("minimumItemSize");  
miningAlgorithmParameter[0].setType("int");  
miningAlgorithmParameter[0].setValue("1");  
miningAlgorithmParameter[0].setMethod("setMinimumItemSize");  
miningAlgorithmParameter[0].setDescr("Minimum size for large items");
```

Устанавливается минимальный размер набора. Например,

```
miningAlgorithmParameter[1] = new MiningAlgorithmParameter();  
miningAlgorithmParameter[1].setName("maximumItemSize");  
miningAlgorithmParameter[1].setType("int");  
miningAlgorithmParameter[1].setValue("-1");  
miningAlgorithmParameter[1].setMethod("setMaximumItemSize");  
miningAlgorithmParameter[1].setDescr("Maximum size for large items");
```

Устанавливается параметр позволяющий генерировать ассоциативные правила, а не только частые наборы. Например,

```
miningAlgorithmParameter[2] = new MiningAlgorithmParameter();  
miningAlgorithmParameter[2].setName("generateRules");  
miningAlgorithmParameter[2].setType("boolean");  
miningAlgorithmParameter[2].setValue("true");  
miningAlgorithmParameter[2].setMethod("setGenerateRules");  
miningAlgorithmParameter[2].setDescr("Allow to generate  
association rules");
```

Все дополнительные параметры должны быть добавлены в спецификацию алгоритма:

```
miningAlgorithmSpecification.setInputAttribute(  
    miningAlgorithmParameter );
```

Далее должен быть создан экземпляр алгоритма выполняющего построение модели ассоциативных правил:

```
String className = miningAlgorithmSpecification.getClassName();  
if( className == null )  
    throw new MiningException( "className attribute expected." );
```

```
Class algorithmClass = Class.forName( className );  
Object algorithm = algorithmClass.newInstance();  
AssociationRulesAlgorithm miningAlgorithm = (AssociationRulesAlgorithm)algorithm;
```

После создания необходимо вызвать метод *buildModel()* для построения модели класса *AssociationRulesMiningModel*.

```
AssociationRulesMiningModel model =  
    (AssociationRulesMiningModel) algorithm.buildModel();
```

Обработать построенную модель, например, для вывода ассоциативных правил в консоль можно следующим образом.

Получить список ассоциативных правил и частых наборов:

```
Vector rules = ruleModel.getAssociationRules();  
Vector LITS = ruleModel.getLargeItemSets();
```

Получить атрибуты идентифицирующие элементы и транзакции:

```
CategoricalAttribute itemId =  
    (CategoricalAttribute)((AssociationRulesSettings)ruleModel.getMiningSettings())  
    .getItemId();  
CategoricalAttribute transactId = (CategoricalAttribute)((AssociationRulesSettings)  
    ruleModel.getMiningSettings()).getTransactionId();
```

Определить количество правил, частых наборов и транзакций

```
int nLITS = LITS.size();  
int nRules = rules.size();  
int itemNumber = itemId.getCategoriesNumber();  
int transactsNumber = transactId.getCategoriesNumber();
```

Вывести все ассоциативные правила

```
System.out.println();  
System.out.println("Number of association rules found: " + nRules);  
for (int i = 0; i < nRules; i++) {  
    // Новое правило:  
    System.out.print(i + ": ");  
  
    // Показать правило:  
    RuleSet rs = (RuleSet) rules.elementAt(i);  
    int itemSize = rs.getSize();  
  
    // условная часть правила:  
    ItemSet is = rs.getPremise();
```

```

int nprem = rs.getPremise().getSize();
for (int j = 0; j < nprem; j++) {
    int pN = is.getItemAt(j);
    Category cat = (Category) itemId.getCategory(pN);
    System.out.print(cat.getValue() + " ");
};
System.out.print("=> ");

// Заключительная часть правила:
for (int j = nprem; j < itemSize; j++) {
    int pN = rs.getConclusion().getItemAt(j-nprem);
    Category cat = (Category) itemId.getCategory(pN);
    System.out.print(cat.getValue() + " ");
}

```

3.3. Задача кластеризации

Для настройки процесса кластеризации создается экземпляр класса *ClusteringSettings*.

```
ClusteringSettings miningSettings = new ClusteringSettings();
```

Ему передаются метаданные загруженных данных:

```
miningSettings.setDataSpecification( metaData );
```

Для кластеризации можно указать имя атрибута определяющего идентификацию кластера. Например,

```
miningSettings.setClusterIdAttributeName("ItemID");
```

Для решения задачи кластеризации можно использовать алгоритм KMeans. В библиотеке Xelopes он реализован в классе *com.prudsys.pdm.Models.Clustering.CDBased.Algorithms.KMeans.KMeans*. Для его использования необходимо выполнить настройку специфичных параметров. Покажем как это можно сделать с помощью загрузки из файла *algorithms.xml*.

```

MiningAlgorithmSpecification miningAlgorithmSpecification =
    MiningAlgorithmSpecification.getMiningAlgorithmSpecification("KMeans
");

```

Далее должен быть создан экземпляр алгоритма выполняющего кластеризацию:

```

String className = miningAlgorithmSpecification.getClassName();
if( className == null )
    throw new MiningException( "className attribute expected." );

```

```

Class algorithmClass = Class.forName( className );
Object algorithm = algorithmClass.newInstance();
ClusteringAlgorithm miningAlgorithm = (ClusteringAlgorithm)algorithm;

```

После создания необходимо вызвать метод *buildModel()* для построения модели класса *ClusteringMiningModel*.

```
ClusteringMiningModel model =
```

```
(ClusteringMiningModel) algorithm.buildModel();
```

Обработать построенную модель, например, для вывода кластеров в консоль можно следующим образом.

```
System.out.println("number of clusters: " + clustModel.getNumberOfClusters());  
Cluster[] clust = clustModel.getClusters();  
for (int i = 0; i < clust.length; i++)  
    System.out.println("Clust["+i+"]: " + clust[i].toString() );
```

3.4. Задача классификации

Для настройки процесса классификации например с помощью деревьев решений создается экземпляр класса *SupervisedMiningSettings*.

```
SupervisedMiningSettings miningSettings = new SupervisedMiningSettings();
```

Ему передаются метаданные загруженных данных:

```
miningSettings.setDataSpecification( metaData );
```

Для классификации необходимо указать атрибут соответствующий независимой переменной. Например,

```
MiningAttribute targetAttribute = (MiningAttribute)metaData.getMiningAttribute( "contact-lenses"  
);  
miningSettings.setTarget(targetAttribute);
```

Для построения дерева решений можно использовать алгоритм ID3. В библиотеке *Xelopes* он реализован в классе *com.prudsys.pdm.Models.Classification.DecisionTree.Algorithms.Id3.ID3Algorithm*. Для его использования необходимо выполнить настройку параметров, которые могут быть загружены из файла *algorithms.xml*.

```
MiningAlgorithmSpecification miningAlgorithmSpecification =  
    MiningAlgorithmSpecification.getMiningAlgorithmSpecification(  
        "Decision Tree (ID3)" );
```

Далее должен быть создан экземпляр алгоритма выполняющего классификацию:

```
String className = miningAlgorithmSpecification.getClassName();  
if( className == null )  
    throw new MiningException( "className attribute expected." );  
Class algorithmClass = Class.forName( className );  
Object algorithm = algorithmClass.newInstance();  
DecisionTreeAlgorithm miningAlgorithm = (DecisionTreeAlgorithm)algorithm;
```

После создания необходимо вызвать метод *buildModel()* для построения модели класса *DecisionTreeMiningModel*.

```
DecisionTreeMiningModel model =  
    (DecisionTreeMiningModel) algorithm.buildModel();
```


Обработать построенную модель, например, для вывода в консоль можно с помощью рекурсивной процедуры.

```
private static void showTreeRecursively(DecisionTreeNode node) {

    // Loop over all childs:
    for (int i = 0; i < node.getChildCount(); i++) {
        DecisionTreeNode child = (DecisionTreeNode) node.getChildAt(i);
        System.out.println( "parent: " + node.toString() + " ==> child: " + child.toString());

        // Get child's childs:
        showTreeRecursively(child);
    };
}
```

Она может быть вызвана следующим образом:

```
DecisionTreeNode root = (DecisionTreeNode) model.getClassifier();
showTreeRecursively(root);
```

Деревья решений используются для классификации данных. Следовательно построенная модель может быть применена к новым данным с целью их классификации. Покажем, как это может быть выполнено на практике:

```
int i = 0;
int wrong = 0;
while (inputData.next()) {
    // классифицировать вектор:
    MiningVector vector = inputData.read();
    double predicted = ((SupervisedMiningModel) model).apply(vector);

    double realTarCat = vector.getValue( targetAttribute.getName() );
    Category tarCat = ((CategoricalAttribute)targetAttribute).getCategory(realTarCat);
    Category predTarCat = ((CategoricalAttribute)targetAttribute).getCategory(predicted);
    System.out.println(" " + ++i + ": " + vector + " -> " + predTarCat);
    if (! predTarCat.equals( tarCat ) )
        wrong = wrong + 1;
};
System.out.println("classification rate = " + (100.0 - ((double) wrong / i)*100.0) );
```

Порядок выполнения работы

1. Изучить основные принципы анализа данных с использованием библиотеки Xelopes.
2. Реализовать программу выполняющую построение модели заданной вариантном задания.
3. Построить модели для всех файлов с исходными данными и разными параметрами.
4. Сохранить построенные модели в текстовом формате и формате PMML.
5. Сравнить полученные модели с моделями, построенными с помощью GUI Xelopes для тех же данных и тех же настройках.
6. Если существует разница объяснить ее.

Варианты задания

Вариант	Модели
1	<i>StatisticsMiningModel</i>
2	<i>AssociationRulesMiningModel</i>

3	<i>SequentialMiningModel</i>
4	<i>CustomerSequentialMiningModel</i>
5	<i>ClusteringMiningModel</i>
6	<i>SupportVectorMiningModel</i>
7	<i>SparseGridsMiningModel</i>
8	<i>DecisionTreeMiningModel</i>

Отчет по работе

1. Цель работы.
2. Блок схема программы
3. Результаты применения к исходным данным из разных файлов с разными параметрами.
4. Сравнение с моделями, построенными с помощью GUI Xelopes.
5. Выводы по работе.

ЛИТЕРАТУРА

1. А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод Методы и модели анализа данных: OLAP и Data Mining. БХВ-Петербург; 2004г.
2. А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP. БХВ-Петербург; 2007г.