II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

EKSPLORACJA DANYCH W SYSTEMACH BAZ DANYCH

Jarosław Gramacki

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: j.gramacki@iie.uz.zgora.pl

STRESZCZENIE

W artykule omówiono przykładowe rozwiązanie integrujące technologie związane z kopalniami danych (ang. *data mining*) z relacyjnymi bazami danych. W pracy ograniczono się do przedstawienia rozwiązań firmy Oracle. Do tej pory baza danych uważana była za miejsce jedynie do składowania danych. Ich obróbka statystyczną zajmowano się wykorzystując najczęściej narzędzia zewnętrzne. Obecnie narzędzia te są coraz częściej bardzo ściśle zintegrowane z bazą danych. Począwszy od wersji bazy 10g, firma Oracle zgromadziła pod wspólna nazwą Oracle Data Mining (ODM) mechanizmy oraz narzędzia wspomagające działania związane z kopalniami danych (używane jest tu też pojęcie eksploracji danych).

1. WPROWADZENIE

Techniki związane z eksploracją danych[1] w większości oparte są o metody statystyczne i / lub metody uczenia maszynowego (ang. *machine learning*). Ich zastosowanie znacznie zwiększa możliwości analizy posiadanych danych, a co za tym idzie umożliwia odkrywanie nowych zależności miedzy nimi (tzw. odkrywanie wiedzy). Do tej pory zależności te pozostawały ukryte w ogromnej ilości pozornie niezwiązanych ze sobą danych. Chodzi więc o wyszukiwanie czytelnych schematów, wzorców, reguł, które nie były wcześniej znane (a przynajmniej jawnie widoczne) a mogą być użyteczne przy np. podejmowaniu decyzji. Obecnie nie jest praktycznie żadnym problemem gromadzenie dowolnie dużych ilości danych w systemach OLTP (ang. *On Line Transaction Processing*). Dużo więcej problemów pojawia się, gdy należy sensownie z nich skorzystać w kontekście wspomnianej ich eksploracji. Stosowane obecnie narzędzia, zarówno darmowe jak i produkty komercyjne [1, 2, 3], w większości nie są w żaden sposób związane z miejscem przechowywania danych (z bazą danych). Zwykle mają one potężne możliwości analizy danych. Dostęp do danych wymaga jednak „warstwy pośredniej", która zapewnia mechanizmy umożliwiające fizyczne podłączenie się do źródła danych.

Dużo lepszym rozwiązaniem jest implementacja algorytmów eksploracji danych bezpośrednio w bazie danych (czyli maksymalnie blisko miejsca przechowywania danych). Rozwiązanie takie bardzo ułatwia realizowanie zadań typowych dla data mining w ogólnie pojętych aplikacjach bazodanowych. Przykładem rozwiązania implementującego algorytmy eksploracyjne bezpośrednio w bazie danych jest Oracle Data Mining (ODM) [4, 5]. Korzystając z ODM wszystkie etapy procesu ekspolarcji danych mają miejsce w bazie danych i wykonywane są tymi samymi narzędziami, które wykorzystywane są do obsługi baz relacyjnych. Przykładowo pisząc aplikację bazodanową w języku PL/SQL (uznany standard firmy Oracle) istnieje możliwość wywoływania w niej odpowiednich funkcji, również PL/SQL-owych, które implementują wybrany algorytm eksploracyjny. Wyniki tak przeprowadzanych analiz przechowywane są dalej w zwykły dla bazy danych sposób czyli w postaci tabel relacyjnych. Stamtąd też mogą być łatwo pobierane do dalszej analizy i / lub prezentacji wyników.

2. MODELE EKSPLORACYJNE W BAZIE DANYCH

Nadrzędnym celem osadzenia algorytmów eksploracyjnych wprost w bazie danych jest umożliwienie aplikacji bazodanowej budowanie modelu eksploracyjnego (ang. *data mining model*) bezpośrednio z danych źródłowych pobieranych z tejże bazy. Poniżej wymieniono dostępne w ODM modele wraz z krótką ich charakterystyką oraz stosowanymi algorytmami. Z uwagi na powszechnie używaną terminologię angielską, postanowiono pozostać przy oryginalnym nazewnictwie. Dostępne w ODM algorytmy podzielono na dwie grupy: algorytmy uczenia z nadzorem (ang. *supervised learning algorithms*) i algorytmy uczenia bez nadzoru (ang. *unsupervised learning algorithms*). Odpowiadające tym dwóm grupom algorytmów modele eksploracyjne nazywane są odpowiednio modelami predykcyjnymi (ang. *Predictive Data Mining Models*) i modelami opisowymi (ang. *Descriptive Data Mining Models*).

Tab. 1. Metody i algorytmy eksploracyjne dostępne w ODM

Algorytmy uczenia z nadzorem, modele predykcyjne		
Funkcja	**Algorytm(y)**	**Krótki opis**
Classification	Naive Bayes (NB) Adaptive Bayes Network (ABN) Support Vector Machine (SVM)	Budowa modelu (np. drzewo decyzyjne, reguła logiczna) na bazie danych historycznych (tzw. dane treningowe). Utworzony model wykorzystywany jest następnie do klasyfikacji nowych napływających danych według założonych reguł lub do głębszego rozumienia istniejących w nim zależności. Przykład reguły logicznej: IF WIEK>30 AND DOCHOD = Wysoki THEN 90% Pewności, że ...
Regression	Support Vector Machine (SVM)	Model podobny do klasyfikatora. Operuje na danych o atrybutach numerycznych i ciągłych (klasyfikacja wymaga atrybutów o wartościach zkategoryzowanych i dyskretnych). Najczęściej używana jest regresja liniowa, w której wyliczany jest odcinek prosty, który możliwie najlepiej (w

[1] W literaturze często pojęcia kopalnie danych oraz eksploracja danych stosowane są zamiennie.

		ustalonym sensie) dopasowuje się do danych.
Attribute Importance	Minimal Descriptor Length	Szukanie wśród dostępnych danych tzw. atrybutów predykcyjnych, które posiadają największy wpływ na analizowany atrybut celowy (ang. *target attribute*). Na przykład jakie dane osobowe klienta (atrybuty predykcyjne) najbardziej wpływają na decyzję klienta o zakupie samochodu marki VW (atrybut celowy: kupił / nie kupił).
Algorytmy uczenia bez nadzoru, modele opisowe		
Clustering	Enhanced k-means Orthogonal Clustering (O-Cluster)[2]	Szukanie naturalnych grup wśród posiadanego zbioru danych. Na przykład grupowanie klientów banku jako potencjalnych zainteresowanych danymi usługami.
Association	Apriori Algorithm	Szukanie związków pomiędzy występowaniem grup elementów w zadanych zbiorach danych. Na przykład tzw. analiza koszykowa. Klienci, którzy kupują piwo i kiełbaski grillowe prawdopodobnie kupują też węgiel drzewny.
Feature Extraction	Non-Negative Matrix Factorization	Szukanie mniejszego zbioru atrybutów, które wystarczająco opisują np. cechy klienta normalnie opisywanego większa liczbą atrybutów. Na przykład wiedząc o kliencie A, B, C i D nie potrzebujemy już analizować E, F i G.
Inne modele		
Text Mining	-	Szukanie wiedzy w zbiorze niestrukturalnych danych (np. plikach poczty elektronicznej). Budowa modeli typu klasyfikatory, klastry na bazie takich danych.

Po zbudowaniu odpowiedniego modelu, można przystąpić do właściwego zadania eksploracyjnego. Model budowany jest poprzez zastosowanie odpowiedniego algorytmu eksploracyjnego w stosunku do wybranego zbioru danych treningowych w przypadku uczenia z nadzorem (ang. *training data set*). Zbudowany model jest przechowywany w bazie, w zwykłych strukturach relacyjnych (tabelach). Można więc z niego wielokrotnie korzystać, powtarzając zadanie eksploracyjne. Zauważmy, że choć wiele wymienionych wyżej modeli ma strukturę „nie-tabelaryczną" - przykładem niech będą drzewa decyzyjne o typowej strukturze grafowej, to jednak przechowywane są one efektywnie w relacyjnym repozytorium ODM.

3. NARZĘDZIA EKSPLORACYJNE W ODM

ODM udostępnia dwie grupy funkcji do budowy aplikacji zawierających analizy eksploracyjne:

- ODM PL/SQL API,
- ODM Java API.

W artykule używana będzie biblioteka oparta na języku PL/SQL. Analogiczną funkcjonalność uzyskać można stosując drugie podejście oparte na języku Java.

Zgodnie z filozofią PL/SQL wspomniane narzędzia umieszczono w dwóch pakietach: `DBMS_DATA_MINING` (właściwa eksploracja danych) oraz `DBMS_DATA_MINING_TRANSFORM` (procedury pomocnicze przygotowujące dane). Ponieważ język PL/SQL używany był w bazie Oracle „od zawsze", więc pisząc aplikację bazodanową wzbogaconą o elementy data mining, programista

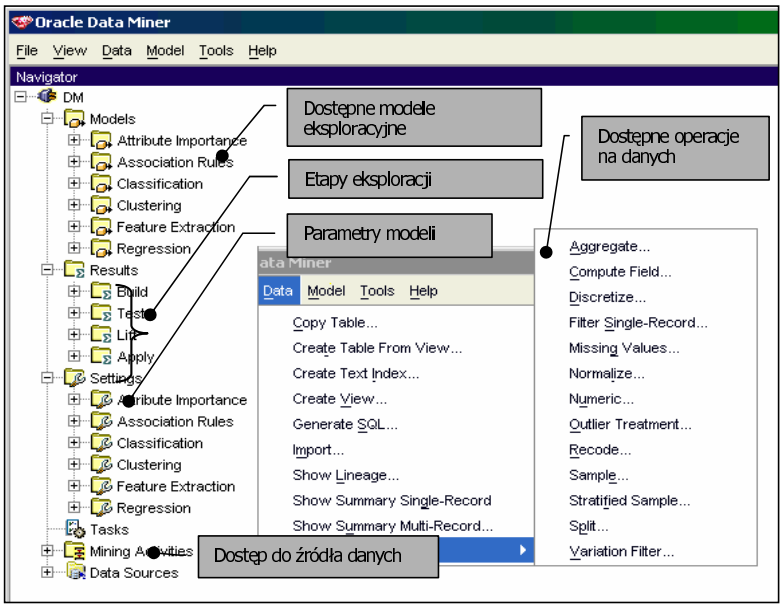
może skupić się na merytorycznej stronie zagadnienia, pracując w dobrze sobie znanym środowisku. Poniżej w Tabeli 2 pokazano wszystkie główne procedury ze wspomnianych pakietów, które użyte zostaną w przykładowej aplikacji pokazanej w dalszej części artykułu. W całym pakiecie ODM jest ich dużo więcej. Ich nazwy odpowiadają terminologii stosowanej w eksploracji danych; zaczynamy od budowy modelu a kończymy na ocenie dokonanej eksploracji. Przykład dotyczył będzie klasyfikacji z wykorzystaniem algorytmu Naive Bayes (NB).

Tab. 2. Podstawowe procedury PL/SQL używane w procesie eksploracji danych

Procedura	Zastosowanie
`CREATE_MODEL`	Budowa modelu eksploracyjnego
`APPLY`	Zastosowanie modelu dla testowych (ocena modelu) lub rzeczywistych danych (właściwa eksploracja).
`RANK_APPLY`	Ocena jakościowa poprzedniego kroku (APPLY).
`COMPUTE_CONFUSION_MATRIX`	Obliczenie tzw. macierzy pomyłek (ang. *confusion matrix*). Wartości macierzy są pewnego rodzaju wskaźnikiem dokładności zbudowanego modelu.

4. GRAFICZNE NARZĘDZIE ODM CLIENT DATA MINER

Narzędzie Data Miner jest w swojej istocie jedynie rodzajem graficznej nakładki na wspomniany zbiór funkcji API. Prowadzi ono użytkownika poprzez wszystkie procesy eksploracji danych -- począwszy od przygotowanie danych aż po analizę wyników przeprowadzonej analizy. Jakkolwiek jest to ciekawe narzędzie, to wydaje się, że jego przydatność ogranicza się do edukacji w dziedzinie eksploracji danych.



Rys. 1. Wygląd ogólny narzędzia Data Miner

[2] Tylko w ramach ODM Java API

5. PRZYKŁAD ANALIZY Z WYKORZYSTANIEM ODM

Celem niniejszego opracowania jest ogólne zaprezentowanie czytelnikowi potencjału pakietu ODM. Szczegóły wymagają odwołania się do materiałów referencyjnych [4, 5]. Pokazany zostanie jednak przykład ilustrujący jeden rzeczywisty problem analizy eksploracyjnej. Analiza oparta będzie o jeden z najlepiej poznanych algorytmów: naiwny klasyfikator Bayes'a (ang. *Naive Bayes*, *NB*). Zastosowano go do przykładowej tabeli zawierającej przykładowe dane historyczne firmy handlowej. Algorytm NB, analizując dane historyczne, wylicza prawdopodobieństwa warunkowe zajścia określonego zdarzenia opisanego tzw. atrybutem celowym (ang. *target attribute*) poprzez obserwację innych atrybutów (ang. *predictive attributes*).

W przykładzie używane będą tabele o nazwach `MINING_DATA_*`[3] pobranych z oryginalnego schematu demonstracyjnego `DM`, instalowanego wraz z pakietem `ODM`[4].

Zilustrowane zostaną wszystkie podstawowe kroki procesu eksploracji danych:

- przygotowanie danych (ang. ***preparing*** *the data*),
- budowa modelu (ang. ***building*** *a model*),
- testowanie zbudowanego modelu (ang. ***testing*** *the model*),
- zastosowanie modelu do nowych danych (ang. ***applying*** *the model to new data, scoring the data*).

Sformułowanie zadania

Załóżmy, że pewna firma handlowa zamierza wprowadzić karty rabatowe dla swoich klientów. Akcją zamierza jednak objąć tylko tych klientów, którzy „rokują", że dzięki otrzymaniu tej karty zwiększą ilość zakupów robionych w tej firmie. Pojawia się więc pytanie, którzy klienci firmy odpowiednio „rokują" dla firmy. Przeprowadzono więc kampanię testową, a jej wyniki zgromadzono w tabelach `MINING_DATA_*` pod postacią pewnej liczby atrybutów opisujących klienta. Są tam dane mające charakter demograficzny (na przykład kolumny `WORKCLASS`, `OCCUPATION`, `EDUCATION`), dane charakteryzujące profile aktywności klienta (kolumny `NO_DIFFERENT_KIND_ITEMS`, `AVARAGE_ _ITEMS_PURCHASED`), dane charakteryzujące „wartość rynkową" klienta (kolumna `BULK_PURCH_AVE_AMT`).

Zgromadzone dane testowe (traktowane jako dane historyczne) użyte zostaną do zbudowania *modelu*, który następnie użyty będzie w stosunku do wszystkich klientów w bazie firmy w celu oszacowania (*przewidzenia)* ew. korzyści wynikających z przyznania klientowi karty rabatowej. Występujący w tabeli atrybut (celowy) `AFFINITY_CARD`, ma wartość 1 dla

[3] Różne przyrostki w nazwach tabel oznaczają tabele z danymi używanymi na różnym etapie procesu eksploracji danych jak: budowa modelu (`MINING_DATA_build`), testowanie modelu (`MINING_DATA_test`), zastosowanie modelu (`MINING_DATA_apply`) do danych "operacyjnych".

[4] Z uwagi na ograniczony rozmiar pracy usunięto wiele kolumn (atrybutów) w oryginalnych tabelach, pozostawiając jednak te, które są niezbędne do demonstracji możliwości pakietu `ODM`.

klientów, których ilość zakupów w firmie zwiększyła się o 10% w trakcie trwania kampanii testowej. W innych przypadkach atrybut ma wartość 0. Tabela 3 ilustruje strukturę oraz wielkość tabel `MINING_DATA_*`.

*Tab. 3. Atrybuty tabel MINING_DATA_**

```
CREATE TABLE MINING_DATA_BUILD (
 AGE                        NUMBER (10),          GENDER                        VARCHAR2 (18),
 WORKCLASS                  VARCHAR2 (21),        AFFINITY_CARD                 NUMBER (1),
 ANNUAL_INCOME              NUMBER (10),          AVERAGE___ITEMS_PURCHASED     NUMBER,
 EDUCATION                  VARCHAR2 (21),        NO_DIFFERENT_KIND_ITEMS       NUMBER,
 MARITAL_STATUS             VARCHAR2 (21),        BULK_PURCH_AVE_AMT            NUMBER,
 OCCUPATION                 VARCHAR2 (21),        PROMO_RESPOND                 NUMBER (10),
 YRS_RESIDENCE              NUMBER,               ID                            NUMBER (10)    )
```

```
SELECT * FROM MINING_DATA_BUILD                  SELECT * FROM MINING_DATA_TEST
...                                              ...
1500 rows selected                               1500 rows selected
-- Ids from 101501 to 103000                     -- IDs from 103001 to 104500

SELECT * FROM DM.MINING_DATA_APPLY
...
1500 rows selected
-- Ids from 100001 to 101500
```

Uwaga na przyrostki _BUILD, _APPLY oraz _TEST

Przygotowanie danych, grupowanie (dyskretyzacja) danych

Oryginalna tabela z danymi zgromadzonymi w kampanii marketingowej została zmodyfikowana celem dostosowania jej do wymogów użytej metody NB. Wprowadzone zmiany to (patrz też Tabela 4):

- jako atrybut celowy (inna stosowana nazwa to zmienna zależna) wybrano kolumnę `AFFINITY_CARD`. Pozostałe kolumny są tzw. predyktorami (zmienne niezależne),
- klucz tabeli (kolumna ID) został usunięty. Kolumna ta ma oczywisty charakter „administracyjny" i jako taka nie wpływa w żaden sposób na wyniki eksploracji,
- zastosowano procedurę grupowania (ang. *binning*) danych zarówno numerycznych (ang. *numerical*) jak i kategorycznych (ang. *categorical*). Patrz Tabele 5, 6. Procedura ta zwana jest też czasami procedurą dyskretyzacji danych.

Tab. 4. Tabela przygotowana do zastosowania algorytmu NB (kolumny posortowano alfabetycznie)

Name	Data Type	Mining Type	Usage	Auto Binning
AFFINITY_CARD	Int	Categorical	**Target**	no
AGE	Int	Numerical	Active	yes
ANNUAL_INCOME	Int	Numerical	Active	yes
AVERAGE___ITEMS_PURCHASED	Float	Numerical	Active	yes
BULK_PURCH_AVE_AMT	Float	Numerical	Active	yes
EDUCATION	String	Categorical	Active	yes
GENDER	String	Categorical	Active	yes
MARITAL_STATUS	String	Categorical	Active	yes
NO_DIFFERENT_KIND_ITEMS	Int	Categorical	Active	yes
OCCUPATION	String	Categorical	Active	yes
PROMO_RESPOND	Int	Categorical	Active	yes
WORKCLASS	String	Categorical	Active	yes
YRS_RESIDENCE	Int	Numerical	Active	yes

Tab. 5. Ilustracja grupowania (ang. bining) danych numerycznych. Utworzenie granic (ang . bin boundaries). Kolumna `AGE` – 5 bins, kolumna `YRS_RESIDENCE` – 4 bins

COLUMN_NAME	LOWER_BOUNDARY	UPPER_BOUNDARY	BIN_ID	DISPLAY_NAME
AGE	17	31,6	1	[17-31.6]
AGE	31,6	46,2	2	(31.6-46.2]
AGE	46,2	60,8	3	(46.2-60.8]
AGE	60,8	75,4	4	(60.8-75.4]
AGE	75,4	90	5	(75.4-90]
YRS_RESIDENCE	0	3.5	1	(0-3.5]

YRS_RESIDENCE	3.5	7	2	(3.5-7]
YRS_RESIDENCE	7	10.5	3	(7-10.5]
YRS_RESIDENCE	10.5	14	4	(10.5-14]

Tab. 6. Ilustracja grupowania (ang. bining) danych kategorycznych. Kolumna WORKCLASS *– 5 bins*

COLUMN_NAME	CATEGORY	BIN_ID	DISPLAY_NAME
WORKCLASS	Private	1	Private1
WORKCLASS	SelfENI	2	SelfENI
WORKCLASS	Loc-gov	3	Loc-gov
WORKCLASS	?	4	?
WORKCLASS	SelfEI	5	SelfEI

Przykładowe kody w języku PL/SQL

Przykład oparto na skrypcie *nbsample.sql* [5]. Analogiczny przykład łatwo „odtworzyć” w środowisku ODM Data Miner opisywanym wyżej. Poniższe kody, w stosunku do oryginału, został maksymalnie okrojone. W praktyce wiele argumentów zastosowanych procedur mogłoby nie przybierać jedynie wartości domyślnych [5]. Kodów nie należy traktować jako działającą aplikację, ale raczej jako ilustrację zasady, wygody, prostoty, etc. pracy z pakietem ODM PL/SQL API.

```
----------------- DROP MODEL ---------------
  dbms_data_mining.drop_model('Naive_Bayes_Sample'); --  stored in ODM repository

--------------- PREPARE DATA ---------------
-- Prepare MINING_DATA_BUILD data as appropriate

-- tables used
-- nb_sample_num_boundary, table            nb_sample_build_cat, final view
-- nb_sample_cat_boundary, table            nb_sample_build_prepared, temp. view


-- Make a numerical bin boundary table. See Tables 4, 5.
  dbms_data_mining_transform.create_bin_num (
    bin_table_name     => 'nb_sample_num_boundary');

  -- Create boundaries for (5 bins)
  -- In practice, one can iteratively call this routine several times with
  -- different specifications for number of bins for a given input table.
  -- For each iteration, one can selectively exclude attributes (that is,
  -- column names) using the exclude_list parameter

  dbms_data_mining_transform.insert_bin_num_eqwidth (
    bin_table_name    => 'nb_sample_num_boundary', --> resulted table
    data_table_name   => 'MINING_DATA_BUILD',      --> source table
    bin_num           => 5,
    round_num         => 0);


  -- Make a categorical bin boundary table. See Tables 4, 6.
  dbms_data_mining_transform.create_bin_cat (
    bin_table_name     => 'nb_sample_cat_boundary');

  dbms_data_mining_transform.insert_bin_cat_freq (
    bin_table_name    => 'nb_sample_cat_boundary', --> resulted table
    data_table_name   => 'MINING_DATA_BUILD',      --> source table
    bin_num           => 5,
    default_num       => 0);

  -- BUILD STAGE.
  -- Creates the view that performs categorical and numerical binning
  -- See BIN_ID columns in Table 5 and Table 6
  -- In comparison with MINING_DATA_BUILD, a final view contains bins
  -- instead of original values
  dbms_data_mining_transform.xform_bin_cat (
    bin_table_name  => 'nb_sample_cat_boundary',-- source binning
    data_table_name => 'MINING_DATA_BUILD',     -- source table
    xform_view_name => 'nb_sample_build_cat');  -- view1 created
```

```
  dbms_data_mining_transform.xform_bin_num (
    bin_table_name  => 'nb_sample_num_boundary',
    data_table_name => 'nb_sample_build_cat',       -- view1 as input
    xform_view_name => 'nb_sample_build_prepared'); -- final view created
```

```
---------------- CREATE MODEL ---------------- Create Naive Bayes Model

  dbms_data_mining.create_model(
    model_name          => 'Naive_Bayes_Sample',
    mining_function     =>  dbms_data_mining.classification,
    data_table_name     => 'nb_sample_build_prepared', -- source view (not table)
    case_id_column_name => 'id',
    target_column_name  => 'affinity_card',
    settings_table_name => 'nb_sample_settings');
```

```
---------------- TEST MODEL -----------------

-- Tables used:
-- nb_sample_test_apply, table              nb_sample_test_cat, temp. view
-- nb_sample_confusion_matrix, table        nb_sample_test_targets, final view
--                                          nb_sample_test_prepared, view

-- TEST STAGE.
  dbms_data_mining_transform.xform_bin_cat (
    bin_table_name  => 'nb_sample_cat_boundary',
    data_table_name => 'MINING_DATA_TEST',
    xform_view_name => 'nb_sample_test_cat');

  dbms_data_mining_transform.xform_bin_num (
    bin_table_name  => 'nb_sample_num_boundary',
    data_table_name => 'nb_sample_test_cat',
    xform_view_name => 'nb_sample_test_prepared');  -- final view created

---------------- APPLY ON TEST DATA -----------------

  dbms_data_mining.apply(model_name => 'Naive_Bayes_Sample',
                data_table_name     => 'nb_sample_test_prepared', --> source view
                case_id_column_name => 'id',
                result_table_name   => 'nb_sample_test_apply');   --> resulted table

-- Create Test targets view
CREATE VIEW nb_sample_test_targets AS
SELECT id, affinity_card  FROM nb_sample_test_prepared;

-- Confusion matrix with accuracy in 'v_accuracy'.
  v_accuracy NUMBER;
  dbms_data_mining.compute_confusion_matrix (
    accuracy                  => v_accuracy,
    apply_result_table_name     => 'nb_sample_test_apply',
    target_table_name           => 'nb_sample_test_targets',
    case_id_column_name         => 'id',
    target_column_name          => 'affinity_card',
    confusion_matrix_table_name => 'nb_sample_confusion_matrix');

-- Confusion_matrix created in nb_sample_confusion_matrix
-- Accuracy: 0,8093

SELECT * FROM nb_sample_confusion_matrix;
ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE      VALUE
------------------- ---------------------- ----------
                  0                      0        937
                  0                      1        217
                  1                      0         69
                  1                      1        277
```

```
---------- APPLY ON NEW DATA ----------
-- Tables used:
-- nb_sample_apply_result , table           nb_sample_apply_prepared, final view
-- nb_sample_apply_ranked,  table           nb_sample_apply_cat, temp. view

-- PREPARE APPLY DATA.
  dbms_data_mining_transform.xform_bin_cat (
    bin_table_name  => 'nb_sample_cat_boundary',
```

```
    data_table_name => 'MINING_DATA_APPLY',
    xform_view_name => 'nb_sample_apply_cat');

  dbms_data_mining_transform.xform_bin_num (
    bin_table_name  => 'nb_sample_num_boundary',
    data_table_name => 'nb_sample_apply_cat',
    xform_view_name => 'nb_sample_apply_prepared');  -- final view created

  dbms_data_mining.apply(
    model_name                    => 'Naive_Bayes_Sample',
    data_table_name               => 'nb_sample_apply_prepared', --> source view
    case_id_column_name           => 'id',
    result_table_name             => 'nb_sample_apply_result');  --> resulted table
```

```
----------------- DISPLAY APPLY RESULT -----------------

SELECT id, prediction, Round(probability,4) probability
FROM nb_sample_apply_result ORDER BY id, prediction

        ID PREDICTION PROBABILITY
---------- ---------- -----------
    100001          0       ,9996
    100001          1       ,0004
    100002          0       ,6452
    100002          1       ,3548
    ...           ...       ...
    101500          0        1
    101500          1        0
```

```
----------------- RANK THE APPLY RESULTS -----------------
dbms_data_mining.rank_apply (
    apply_result_table_name     => 'nb_sample_apply_result',  --> source view
    case_id_column_name         => 'id',
    score_column_name           => 'PREDICTION',
    score_criterion_column_name => 'PROBABILITY',
    ranked_apply_table_name     => 'nb_sample_apply_ranked',  --> resulted table
    top_n                       => 1);

-- Display rank result
SELECT id, prediction, Round(probability,4) probability, rank
FROM nb_sample_apply_ranked
ORDER BY id ,rank;

        ID PREDICTION PROBABILITY  RANK
---------- ---------- -----------  ----------
    100001          0       ,9996          1
    100002          0       ,6452          1
    100003          0        ,951          1
    ...           ...         ...        ...
    101499          1       ,9998          1
    101500          0           1          1
```

Analiza przeprowadzonej eksploracji

Dane eksploracyjne pobrano ze standardowej tabeli relacyjnej, w której każdy wiersz reprezentował jeden przypadek (ang. *case*) a każda kolumna jeden atrybut eksploracyjny (ang. *mining attribut*). W przykładzie, z uwagi na ograniczoną objętość pracy, celowo ograniczono liczbę atrybutów eksploracyjnych do zaledwie kilku. ODM obsługuje ich maksymalnie 999. Im jest ich więcej tym, (przynajmniej teoretycznie) wyniki eksploracji (tu: klasyfikacji) lepiej sprawdzą się w praktyce.

Po koniecznym przygotowaniu danych na potrzeby algorytmu NB (grupowanie atrybutów numerycznych i kategorycznych), utworzono stosowny model eksploracyjny, przetestowano go na danych testowych i w końcu zastosowano go do danych rzeczywistych. Jakość modelu

oceniono z użyciem macierzy pomyłek (ang. *confusion matrix*)[5]. Jej interpretacja jest następująca: wartość 217 oznacza, że utworzony model *niewłaściwie* zaklasyfikował 217. klientów jako potencjalnych kandydatów do otrzymania karty rabatowej, gdyż nie osiągnęli oni założonego 10% wzrostu wartości dokonywanych zakupów w firmie. Wartość 937 oznacza, że utworzony model *właściwie* zaklasyfikował 937. klientów w sensie jak powyżej.

Dane rzeczywiste (operacyjne) przeanalizowano z użyciem utworzonego modelu (etap *apply*). Użyteczność modelu pokazano w postaci odpowiednich wartości prawdopodobieństwa i rangi (ang. *rank*).

6. WNIOSKI

Udostępnienie narzędzi eksploracyjnych i metod data mining bezpośrednio w bazie danych znacząco ułatwia ich stosowanie w ogólnie pojętych aplikacjach bazodanowych. Stosowane do tego narzędzia programistyczne, jak na przykład język PL/SQL, są powszechnie znane i cenione wśród programistów. Jest to niewątpliwą zaletą. Wszystkie elementy procesu eksploracji takie jak: przygotowanie danych, budowa modelu, wyniki analizy, przechowywane są w tym samym miejscu - w bazie danych.

Ponieważ rzeczywiste problemy eksploracji danych niejako z definicji operują na wielkich ilościach danych, dlatego celowe wydaje się prowadzenie prac nad zrównoleglaniem algorytmów i metod data mining. Obecnie podejście takie nie jest stosowane w aktualnych wersjach komercyjnych baz danych.

Innym problemem jest prowadzenie eksploracji danych zgromadzonych w wielu tabelach. Tu kluczowym zagadnieniem byłaby szybkość działania algorytmów w kontekście złączeń tabel relacyjnych. Na tym polu system Oracle oferuje jednak pewne udogodnienia związane z możliwością „zrównoleglania" złączeń w jednym z oferowanych wariantów bazy danych (tzw. opcja równoległa - ang. *parallel option*).

LITERATURA

[1] Data Mining, Knowledge Discovery, Genomic Mining, Web Mining. http://www.kdnuggets.com/

[2] SAS: http://www.sas.com/technologies/analytics/datamining/

[3] StatSoft: http://www.statsoftinc.com/products/dataminer.htm

[4] Oracle Documentation: Oracle Data Mining Concepts, 10g Release 1

[5] Oracle Documentation: Oracle Data Mining Application Developer's Guide, 10g Rel. 1

[6] Oracle Documentation: Application Developer's Guide - Fundamentals, 10g Rel. 1

[7] Oracle Documentation: Oracle OLAP Application Developer's Guide, 10g Rel. 1

[8] Oracle Documentation: Oracle Application Server Containers for J2EE User's Guide10g

[5] W pracy pominięto inne, dostępne w ODM, miary jakości modeli.