Mikołaj Morzy, Marek Wojciechowski:
"Integracja technik eksploracji danych z systemem zarządzania bazą danych
na przykładzie Oracle9i Data Mining"

Streszczenie

Eksploracja danych znajduje coraz szersze zastosowanie we wszystkich dziedzinach życia codziennego. W ostatnich latach obserwujemy odchodzenie od specjalizowanych, aplikacyjnie zorientowanych systemów eksploracyjnych na rzecz systemów ogólnego przeznaczenia. Systemy takie wymagają ścisłej integracji z istniejącą architekturą baz danych. Jedną z propozycji integracji systemu eksploracyjnego z relacyjną bazą danych jest moduł Oracle9*i* Data Mining. Artykuł prezentuje składniki modułu eksploracyjnego i przedstawia metodykę tworzenia aplikacji eksploracyjnych wykorzystujących to narzędzie.

Wprowadzenie

Eksploracja danych, zwana także odkrywaniem wiedzy w bazach danych (*data mining, knowledge discovery in databases*) to proces odkrywania nowych, nieznanych, pożytecznych i zrozumiałych wzorców w dużych wolumenach danych. W ostatnich latach obserwujemy wyraźne odchodzenie od dedykowanych i wyspecjalizowanych systemów eksploracyjnych i dążenie do integracji tych systemów z istniejącymi systemami zarządzania bazami danych. Integracja ta przebiega przede wszystkim w ramach magazynów danych (*data warehouses*), które stanowią doskonałe źródło danych dla różnych technik eksploracyjnych. Począwszy od wersji 9.0.1 serwera bazy danych, Oracle oferuje nowy moduł umożliwiający łatwe zanurzanie aplikacji eksploracyjnych bezpośrednio w bazie danych. Wszystkie czynności charakterystyczne dla eksploracji, takie jak budowanie i testowanie modeli, odkrywanie reguł asocjacyjnych bądź klasyfikacja danych są dostępne za pomocą zbioru klas i metod języka Java. Interfejs programowy modułu Oracle9*i* Data Mining pozwala programistom na pełną kontrolę nad procesem eksploracji danych i całkowitą integrację tego procesu z istniejącą architekturą bazy danych.

Architektura Oracle9*i* Data Mining składa się z dwóch komponentów: interfejsu programistycznego (Oracle9*i* Data Mining API, ODM API) i serwera eksploracyjnego (Data Mining Server, DMS). ODM API jest zbiorem klas i metod za pomocą których programista określa zbiór danych w których będzie przebiegać eksploracja, wybiera odpowiednią metodę, ustala wartości parametrów sterujących przebiegiem poszczególnych algorytmów, itp. DMS jest komponentem zaszytym w bazie danych. Jest to zbiór procedur PL/SQL i funkcji języka Java które dokonują eksploracji. Poza tym DMS zawiera repozytorium, w którym przechowywane są modele eksploracji i jej wyniki.

Oracle9*i* Data Mining umożliwia dwa tryby eksploracji: uczenie z nadzorem (*supervised learning*) i uczenie bez nadzoru (*unsupervised learning*). Do uczenia z nadzorem wykorzystywana jest metoda klasyfikacji. Algorytm wykorzystywany do klasyfikacji to implementacja naiwnego klasyfikatora Bayesa (*Naive Bayes*). Poza budowaniem (*applying*) klasyfikatora Oracle9*i* Data Mining pozwala również na aplikowanie modelu do danych (*scoring*), testowanie modelu na danych treningowych (*testing*) oraz stosowanie modelu do nowych danych w celu obliczenia różnych wartości statystycznych (*computing lift*). Metodą wykorzystywaną do uczenia bez nadzoru jest odkrywanie asocjacji. Oracle9*i* Data Mining wykorzystuje w tym celu algorytm o nazwie *Association Rules*. Algorytm ten pozwala na odkrywanie i przechowywanie reguł w repozytorium serwera eksploracyjnego w celu ich późniejszej analizy.

Metody eksploracji dostępne w Oracle9*i* Data Mining

Klasyfikacja za pomocą naiwnego klasyfikatora Bayesa

Naiwny klasyfikator Bayesa jest jedną z metod uczenia maszynowego, stosowaną do rozwiązywania problemu sortowania i klasyfikowania. Zadaniem klasyfikatora Bayesa jest przyporządkowanie nowego przypadku do jednej z klas decyzyjnych, przy czym zbiór klas decyzyjnych musi być skończony i zdefiniowany a priori. Do przypisania przypadku do określonej klasy klasyfikator wykorzystuje twierdzenie Bayesa.

Twierdzenia Bayesa mówi że dla dwóch niezależnych zdarzeń A i B prawdopodobieństwo wystąpienia B jeśli wystąpiło A wynosi:

$P(B|A) = (P(B) * P(A|B)) / P(A) = P(A \text{ i } B) / P(A)$,

gdzie P(A) oznacza prawdopodobieństwo że wystąpi A, P(A|B) oznacza prawdopodobieństwo że wystąpi A pod warunkiem że wystąpiło już B, zaś P(A i B) oznacza prawdopodobieństwo tego, że wystąpią zarówno A jak i B. Jak widać, naiwny klasyfikator Bayesa zakłada niezależność przykładów. W przypadku gdy poszczególne przykłady są zależne, klasyfikator dokonuje szacowania prawdziwej wartości. Eksperymenty dowodzą jednak, że szacowanie nie wpływa silnie na zdolność klasyfikatora do poprawnego przewidywania.

Klasyfikator może być wykorzystywany do klasyfikacji binarnej i wieloklasowej. Dla przykładu, model klasyfikacji binarnej został zbudowany na podstawie informacji o kredytach zaciągniętych przez klientów banku oraz o ściągalności tych kredytów. Każdy klient został przypisany do jednej z dwóch klas: ‘bezpieczny’ (klienci z tej klasy spłacili kredyty w terminie) i ‘ryzykowny’ (klienci z tej klasy ociągali się ze spłatą kredytu). Na podstawie tak zbudowanego modelu klasyfikator może określać klasę, do której należy nowy klient ubiegający się o kredyt. Oprócz określenia klasy klasyfikator określa też prawdopodobieństwo poprawności klasyfikacji (np. nowy klient należy do klasy ‘ryzykowny’ z prawdopodobieństwem 80%).

Naiwny klasyfikator Bayesa jest jedną z najbardziej efektywnych metod uczenia maszynowego służącą do klasyfikacji dokumentów tekstowych. W praktyce stosowany jest najczęściej do filtrowania dokumentów (np. w wyszukiwarkach internetowych). Każdy dokument w zbiorze treningowym jest przypisany do jednej z klas (np. rozrywka, nauka, polityka). Z każdym dokumentem związany jest wektor zawierający liczbę wystąpień słów z predefiniowanej listy słów (taka lista zazwyczaj zawiera kilkaset lub kilka tysięcy słów). Klasyfikator zbudowany na takim zbiorze danych wejściowych potrafi z bardzo dużą trafnością przypisywać nowe dokumenty do istniejących klas.

Reguły asocjacyjne

Reguły asocjacyjne służą do reprezentowania współwystępowania elementów. Najczęściej reguły asocjacyjne są wykorzystywane w analizie koszyka zakupów (*market basket analysis*), w trakcie której odkrywane są fakty współwystępowania poszczególnych towarów lub grup towarów w zakupach klientów. Przykładem reguły asocjacyjnej może być np. reguła „70% klientów którzy kupili szampana kupiło też kawior. Zakup szampana i kawioru wystąpił w 0.5% wszystkich zakupów”. Reguły asocjacyjne odkryte w bazie danych opisującej zakupy mogą być wykorzystywane do organizowania promocji, pocztowych ofert sprzedaży, konstruowania transakcji wiązanych, planowania katalogów lub planowania rozkładu towarów i półek w sklepie.

Drugim ważnym segmentem wykorzystania reguł asocjacyjnych jest automatyczna personalizacja serwisów internetowych. Reguła która głosi „50% użytkowników którzy obejrzeli strony A i B w tej samej sesji obejrzeli stronę C” może być wykorzystana do automatycznej modyfikacji strony prezentowanej użytkownikowi i dodawania skrótów do

stron, które prawdopodobnie zainteresują danego użytkownika. W przypadku technologii internetowych reguły asocjacyjne są wykorzystywane przede wszystkim do dodawania rekomendacji zakupów i dynamicznej zmiany struktury połączeń między poszczególnymi częściami serwisu.

Formalnie reguły asocjacyjne można zdefiniować w oparciu o pojęcie zbioru częstego. Niech $L=\{l_1, l_2, ...,l_n\}$ będzie zbiorem literałów zwanych elementami. Niech D będzie kolekcją transakcji, gdzie każda transakcja jest dowolnej długości i $\forall T \in D\ T \subseteq L$. Mówimy, że transakcja T wspiera element x jeśli $x \in T$. Mówimy, że transakcja T wspiera zbiór X, jeśli T wspiera każdy element $x \in X$. Wsparciem (*support*) zbioru X nazywamy stosunek liczby transakcji wspierających X do liczby wszystkich transakcji.

$$support(X, D) = \frac{|\{T \in D : T\ wspiera\ X\}|}{|D|}$$

Problem odkrywania zbiorów częstych polega na znalezieniu w danej bazie danych D wszystkich zbiorów, których wsparcie jest wyższe od zdefiniowanej przez użytkownika wartości, zwanej minimalnym wsparciem (*minsup*). Zbiór, którego wsparcie jest wyższe niż minsup nazywamy zbiorem częstym (*frequent itemset*).

Reguła asocjacyjna to implikacja postaci $X \rightarrow Y$, gdzie $X \subset L$, $Y \subset L$ i $X \cap Y = \emptyset$. Zbiór X nazywamy głową reguły a zbiór Y ciałem reguły. Z każdą regułą asocjacyjną związane są dwie miary wyznaczające statystyczne znaczenie i siłę reguły. Wsparciem (*support*) reguły $X \rightarrow Y$ w bazie danych D nazywamy stosunek liczby transakcji wspierających regułę do liczby wszystkich transakcji. Innymi słowy reguła $X \rightarrow Y$ ma w bazie danych D wsparcie s, jeśli s% transakcji w bazie danych wspiera $X \cup Y$.

$$support(X \rightarrow Y, D) = \frac{|\{T \in D : T\ wspiera\ X \cup Y\}|}{|D|}$$

Ufnością (*confidence*) reguły $X \rightarrow Y$ w bazie danych D nazywamy stosunek liczby transakcji wspierających regułę do liczby transakcji wspierających głowę reguły. Innymi słowy reguła $X \rightarrow Y$ ma w bazie danych D ufność c, jeśli c% transakcji wspierających X wspiera również Y.

$$confidence(X \rightarrow Y, D) = \frac{|\{T \in D : T\ wspiera\ X \cup Y\}|}{|\{T \in D : T\ wspiera\ X\}|}$$

Wsparcie reguły określa jej znaczenie zaś ufność reguły reprezentuje jej siłę. Na przykład, reguła „chleb→woda mineralna” ma wysokie wsparcie, ponieważ zbiór produktów {chleb, woda mineralna} jest bardzo powszechny wśród zakupów klientów. Z drugiej strony reguła ta ma niewielką ufność ponieważ zakupy chleba i wody mineralnej nie jest ze sobą ściśle związane. Reguła „szampan→kawior” ma bardzo wysoką ufność (klient kupujący butelkę szampana bardzo często kupuje też kawior) ale niskie wsparcie, ponieważ niewielu klientów pozwala sobie na zakup tak luksusowych produktów.

Problem odkrywania reguł asocjacyjnych polega na znalezieniu w danej bazie danych D wszystkich reguł asocjacyjnych, których wsparcie i ufność są wyższe od zdefiniowanych przez użytkownika wartości minimalnego wsparcia i minimalnej ufności (*minsup* i *minconf*).

Konstruowanie modelu eksploracji

Użytkownicy korzystający z Oracle9*i* Data Mining mogą tworzyć modele na dwóch poziomach szczegółowości: na poziomie funkcji i na poziomie algorytmu. Budowanie modelu na poziomie funkcji pozwala na pominięcie specyfikowania wszystkich szczegółów algorytmu i skupienie się na właściwym przedmiocie eksploracji. Podczas budowania modelu na poziomie funkcji Oracle9*i* Data Mining samodzielnie dobiera odpowiedni algorytm i wyznacza wartości wszystkich parametrów, dzięki czemu wiele technicznych szczegółów związanych z eksploracją staje się nieistotnych i niewidocznych dla użytkownika.

Użytkownicy którym nie wystarczają standardowe procedury odkrywania wzorców mogą jednak budować model na poziomie algorytmu, dostosowując proces eksploracji dokładnie do swoich potrzeb. Zbudowane modele są przechowywane w repozytorium serwera eksploracji i mogą być później wielokrotnie użyte.
Proces budowania modelu składa się z kilku następujących po sobie kroków. Na początku użytkownik musi utworzyć dane wejściowe poprzez skojarzenie obiektu reprezentującego dane (*data mining object*) z rzeczywistym źródłem danych, którym może być relacja w bazie danych lub płaski plik w systemie operacyjnym. Następnie użytkownik tworzy obiekt reprezentujący ustawienia wybranej funkcji eksploracji (*function settings object*). W kolejnym kroku obiekt funkcji eksploracji jest wiązany z obiektami reprezentującymi logiczną strukturę danych (*logical data specification*) i fizyczne wykorzystanie danych (*data usage specification*). W ostatnim kroku użytkownik wykonuje wybraną przez siebie metodę tworzenia modelu (klasyfikacja lub odkrywanie asocjacji).
Zbudowany model klasyfikacyjny może następnie zostać przetestowany. Testowanie pozwala na oszacowanie dokładności klasyfikacji. Podczas testowania model klasyfikatora jest stosowany do zbioru danych testowych, które nie były użyte do budowania modelu i dla których znana jest wartość klasyfikowanego atrybutu. W wyniku testowania użytkownik otrzymuje macierz pomyłek (*confusion matrix*) z której można odczytać, w ilu przypadkach model poprawnie sklasyfikował dane testowe a w ilu się pomylił oraz jakiego typu były to pomyłki. Analiza takiej macierzy pozwala na dokładniejsze dostrojenie modelu aż do uzyskania satysfakcjonującej jakości klasyfikacji. Dane wykorzystywane do testowania muszą odpowiadać specyfikacji logicznej struktury danych wykorzystywanych do budowania modelu.
Model klasyfikatora zastosowany do nowych danych powoduje obliczenie przewidywanej wartości lub klasy wraz z prawdopodobieństwem. Dane do których stosowany jest model muszą odpowiadać specyfikacji struktury logicznej danych wykorzystywanych do budowania modelu. Wynik zastosowania klasyfikatora jest najczęściej zapisywany do bazy danych. Model może być stosowany do zbioru rekordów odczytanych z tabeli źródłowej albo do pojedynczego rekordu (reprezentowanego wówczas przez pojedynczy obiekt). Przed zapisaniem do bazy danych wynik klasyfikacji może zostać przetworzony do postaci odpowiadającej użytkownikowi. Dla przykładu, użytkownik może dokonać klasyfikacji klientów wyliczając prawdopodobieństwo przynależności każdego klienta do jednej z klas ‘dobry’, ‘średni’, ‘zły’ i zapisać do relacji wynikowej tylko identyfikatory klientów wraz z identyfikatorem przewidywanej klasy i wyliczonym prawdopodobieństwem.
Dla binarnych modeli klasyfikacji (modeli, w których przewidywany atrybut przyjmuje tylko dwie wartości, pozytywną i negatywną) Oracle9*i* Data Mining pozwala na wyliczanie przesunięcia (*lift*). Testowane rekordy są sortowane zgodnie z prawdopodobieństwem poprawności klasyfikacji w ten sposób, że pozytywne przykłady klasyfikowanej klasy charakteryzujące się najwyższym prawdopodobieństwem (rekordy dla których klasyfikator jest najbardziej pewny poprawności ich sklasyfikowania jako pozytywnych przykładów) są umieszczane na początku, zaś najbardziej prawdopodobnie negatywne przykłady klasy są umieszczane na końcu. Tak posortowane rekordy są następnie dzielone na kwantyle. Dla każdego kwantyla Oracle9*i* Data Mining wylicza następujące statystyki:

- gęstość kwantyla (*target density*): stosunek liczby pozytywnych rekordów w kwantylu do liczby wszystkich rekordów w tym kwantylu
- skumulowana gęstość kwantyla (*cumulative target density*): średnia gęstość pierwszych n kwantyli
- przesunięcie kwantyla (*quantile lift*): stosunek gęstości kwantyla do gęstości wszystkich danych

W przypadku stosowania modelu asocjacji Oracle9*i* Data Mining pozwala na zastosowanie stworzonego modelu do danych źródłowych. Wynikiem zastosowania modelu jest zbiór reguł asocjacyjnych odkrytych w danych źródłowych. Odkryte reguły mogą być wyświetlone użytkownikowi oraz zapisane do repozytorium serwera eksploracyjnego. Reguły są składowane w repozytorium w zwykłych relacjach, dzięki czemu możliwe jest ich odczytanie i zaprezentowanie za pomocą dowolnych narzędzi analitycznych (np. Oracle Discoverer, Oracle Reports).

Klasy obiektów tworzące serwer eksploracji

Poniżej przedstawiono opisy najważniejszych klas wchodzących w skład Oracle9*i* Data Mining. Są to klasy umożliwiające programistom tworzenie aplikacji bazodanowych wykorzystujących techniki eksploracji danych.

Fizyczna specyfikacja danych

Fizyczna specyfikacja danych (*physical data specification*) to obiekt reprezentujący fizyczną strukturę danych wykorzystywanych w eksploracji. Specyfikacja określa między innymi format danych, nazwy atrybutów, role pełnione przez poszczególne atrybuty. Dane reprezentowane przez fizyczną specyfikację mogą być wykorzystane w różnych etapach eksploracji, np. podczas budowania modelu, jego testowania, obliczania przesunięcia lub podczas analizy statystycznej. Oracle9*i* Data Mining akceptuje dane wejściowe w dwóch formatach:

- transakcyjny: każdy przykład jest przechowywany w bazie danych w postaci wielu rekordów w tabeli o schemacie SEQUENCEID, ATTRIBUTE_NAME, VALUE. SEQUENCEID to liczba całkowita która służy do identyfikowania rekordów opisujących ten sam przykład, ATTRIBUTE_NAME to łańcuch znaków przechowujący nazwę atrybutu a VALUE to liczba całkowita reprezentująca wartość danego atrybutu. W przypadku atrybutów o dużej liczbie różnych wartości Oracle9*i* Data Mining udostępnia metody umożliwiające dyskretyzację takich atrybutów.
- kategoryczny: każdy przykład jest przechowywany w bazie danych w postaci jednego rekordu. Tabela przechowująca dane kategoryczne nie musi posiadać obowiązkowo klucza podstawowego, lecz jest on zalecany, np. w celu łatwego wiązania rekordów danych wejściowych z rezultatami klasyfikacji. Oracle9*i* Data Mining wymaga aby dane kategoryczne były dyskretyzowane.

W rzeczywistości dane przechowywane w formacie kategorycznym są automatycznie konwertowane przez Oracle9*i* Data Mining do postaci transakcyjnej przed wykonaniem jakiegokolwiek algorytmu. Jeśli w danych występują wartości puste to w przypadku danych kategorycznych para <atrybut, wartość> jest używana tylko wtedy, gdy wartość jest niepusta. W przypadku danych transakcyjnych jeśli którykolwiek z atrybutów SEQUENCEID, ATTRIBUTE_NAME, VALUE jest pusty to cały wiersz jest pomijany.

Specyfikacja funkcji eksploracji

Specyfikacja funkcji eksploracji (*mining function specification*) to obiekt reprezentujący najbardziej ogólne parametry budowanego modelu. W obiekcie tym przechowywana jest informacja o typie wyniku danej funkcji, o specyfikacji logicznej struktury danych wykorzystywanych do budowania modelu, o specyfikacji wykorzystania danych i opcjonalnie o algorytmie wykorzystywanym do budowania modelu (jeśli konkretny algorytm nie zostanie podany to Oracle9*i* Data Mining samodzielnie wybiera odpowiedni algorytm). Specyfikacje funkcji eksploracji są przechowywane w repozytorium serwera eksploracji.

Specyfikacja algorytmu eksploracji
Specyfikacja algorytmu eksploracji (*mining algorithm specification*) to obiekt przechowujący szczegóły dotyczące wykorzystywanego algorytmu. Użycie tego obiektu pozwala użytkownikom zaawansowanym na dokładne dostosowanie eksploracji do swoich potrzeb i dostrojenie wszystkich parametrów algorytmu. Wyraźny rozdział między specyfikacją funkcji i algorytmu pozwala na oddzielenie użytkowników zaawansowanych od użytkowników korzystających ze standardowych procedur eksploracji.

Specyfikacja struktury logicznej danych
Specyfikacja struktury logicznej danych (logical data specification) to zbiór atrybutów eksploracyjnych opisujących logiczny charakter danych wykorzystywanych podczas eksploracji. Każdy atrybut eksploracyjny opisuje domenę danych wejściowych. Atrybuty eksploracyjne mogą być kategoryczne bądź numeryczne. Aktualnie Oracle9*i* Data Mining wspiera wykorzystanie tylko i wyłącznie atrybutów kategorycznych, atrybuty numeryczne muszą być dyskretyzowane do postaci kategorycznej. Każdy atrybut wchodzący w skład specyfikacji logicznej musi posiadać unikalną nazwę. Specyfikacja logiczna danych jest też nazywana sygnaturą modelu.

Specyfikacja fizycznego wykorzystania danych
Specyfikacja fizycznego wykorzystania danych (*data usage specification*) to obiekt przechowujący informacje o tym, w jaki sposób atrybuty eksploracyjne wchodzące w skład specyfikacji logicznej są wykorzystywane w budowaniu modelu. Każdy atrybut może być

- nieaktywny (*inactive*): atrybut nie jest wykorzystywany, to domyślne wykorzystanie każdego atrybutu,
- aktywny (*active*): atrybut jest wykorzystywany do budowania modelu,
- dodatkowy (*supplementary*): atrybut jest wykorzystywany tylko jako dodatkowa informacja umieszczana razem z wynikami klasyfikacji.

Dodatkowo, dla każdego atrybutu specyfikacja fizyczna pamięta czy dany atrybut jest celem klasyfikacji (czyli czy dany atrybut stanowi klasę docelową klasyfikatora).

Model eksploracji
Model eksploracji (*mining model*) to obiekt reprezentujący wynik wykonania eksploracji przy wykorzystanie określonych ustawień (specyfikacji funkcji eksploracji). Fizyczna reprezentacja modelu zależy od algorytmu wykorzystanego do budowania modelu. Oracle9*i* Data Mining pozwala na przechowywanie modelu w repozytorium serwera eksploracji w postaci nazwanej encji.

Wynik eksploracji
Wynik eksploracji (*mining results*) przechowuje wynik wykonania jednej z następujących operacji: zastosowania modelu do danych, przetestowania modelu na danych lub obliczenia przesunięcia. Oracle9*i* Data Mining zezwala na zapisywanie wyników eksploracji jako nazwanych encji do repozytorium serwera eksploracji. Zmaterializowane wyniki eksploracji mogą być następnie wykorzystane do analiz, porównań i raportów. Obiekt ten przechowuje informacje o nazwie modelu wykorzystanego w eksploracji, czas rozpoczęcia i zakończenia operacji, nazwy i lokalizacje danych wejściowych i wyjściowych.

Macierz pomyłek
Macierz pomyłek (*confusion matrix*) to obiekt przechowujący informacje o dokładności klasyfikacji. Jest to macierz kategoryczna, w której wiersze i kolumny odpowiadają poszczególnym wartościom danej kategorii a komórki macierzy przechowują wartości

numeryczne. W macierzy pomyłek każdy wiersz odpowiada rzeczywistym wartościom klasyfikowanego atrybutu zaś każda kolumna odpowiada wartościom wyznaczonym przez model eksploracji. Na przecięciu wiersza i kolumny umieszczona jest wartość odpowiadająca liczbie zaobserwowanych przypadków. Na przykład, liczba 30 umieszczona na przecięciu wiersza „dobry klient" i kolumny „zły klient" oznacza, że klasyfikator 30 razy niepoprawnie zaklasyfikował dobrego klienta jako klienta należącego do kategorii „zły klient".

Wynik zastosowania eksploracji

Wynik zastosowania eksploracji (*mining apply output*) zawiera pewną liczbę elementów pozwalających na sformatowanie wyników eksploracji. Wynik eksploracji może przyjmować różne postacie:

- dane skalarne które są przepisywane bezpośrednio z danych wejściowych do danych wyjściowych (np. wartości atrybutów kluczowych lub atrybuty dodatkowe)
- dane wyliczone przez algorytm (np. nazwy klas, prawdopodobieństwo przynależności przykładu do klasy, wartość przesunięcia)
- dla danych w formacie transakcyjnym wartość atrybutu SEQUENCEID opisującego dany przykład

Wynik zastosowania eksploracji przedstawiany jest w formacie kategorycznym, każdemu przypadkowi wynikowemu odpowiada jeden rekord wynikowy.

Dyskretyzacja

Oracle9*i* Data Mining wymaga, aby dane wykorzystywane w algorytmach były dyskretyzowane. Dyskretyzacja pozwala na grupowanie związanych ze sobą wartości i przetwarzanie atrybutów o znacznie mniejszej liczbie wartości. Przykładowo, podczas dyskretyzacji wartości {chleb, bułka, rogalik, bagietka} zostałyby połączone w jedną wartość {pieczywo}. Dyskretyzacja może odbywać się na trzy sposoby:

- dyskretyzacja jawna: dla atrybutów kategorycznych użytkownik podaje reguły mapowania między zbiorem wartości i zredukowanym zbiorem kategorii, dla atrybutów numerycznych użytkownik podaje dolne i górne granice dla każdej kategorii
- N najczęstszych: tylko dla atrybutów kategorycznych, użytkownik podaje liczbę *n* interesujących go wartości, Oracle9*i* Data Mining automatycznie znajduje *n* najczęściej występujących wartości a wszystkie pozostałe są umieszczane w kategorii zbiorczej {others}
- podział na kwantyle: tylko dla atrybutów numerycznych, użytkownik podaje liczbę interesujących go kwantyli a Oracle9*i* Data Mining automatycznie wyznacza dolne i górne wartości atrybutów dla każdego z kwantyli

Przykładowy program wykorzystujący Oracle9*i* Data Mining

Poniżej zamieszczono szkic przykładowego programu wykorzystującego moduł Oracle9*i* Data Mining. Program odkrywa reguły asocjacyjne w relacji o schemacie DEMO_TX(SEQ_ID,ATRYBUT,WARTOŚĆ) przechowującej dane demograficzne o klientach banku. Fragment danych wejściowych został przedstawiony poniżej:

```
    SEQ_ID ATRYBUT                    WARTOŚĆ
---------- ---------------------- ----------
      6000 wiek                            4
      6000 płeć                            1
      6000 wykształcenie                   3
      6000 dochody                         2
```

```
6000 zawód                          23
6000 stan_cywilny                    4
```

Odkrycie reguł asocjacyjnych w tej bazie danych wymaga wykonania kilku kroków. Poniżej przedstawiono przykładową sekwencję operacji jakie muszą zostać wykonane.

Połączenie z serwerem eksploracji

Na początku program użytkownika musi nawiązać połączenie z serwerem eksploracyjnym. Do połączenia można wykorzystać dowolny sterownik JDBC. Serwer eksploracji jest reprezentowany w programie przez obiekt klasy *DataMiningServer*, zaś połączenie z serwerem jest reprezentowane przez obiekt klasy *Connection*.

```
oracle.dmt.odm.DataMiningServer dms = new
oracle.dmt.odm.DataMiningServer(“jdbc:oracle:thin:@dcs-mm:1521:ora9i", "mikolaj", "*");
oracle.dmt.odm.Connection dmsConnection = dms.login();
```

Utworzenie fizycznej specyfikacji danych dla danych wejściowych

Użytkownik ma do wyboru dwa rodzaje specyfikacji: transakcyjną i kategoryczną, w zależności od formatu wykorzystywanych danych.

```
oracle.dmt.odm.data.PhysicalDataSpecification pds = new NonTransactionalDataSpecification();
```

Dokonanie dyskretyzacji danych

Dyskretyzacja wymaga podania specyfikacji dla każdego atrybutu i utworzenia dwóch relacji tymczasowych do przechowywania szczegółów o dyskretyzacji atrybutów numerycznych i kategorycznych. Do dyskretyzowania danych numerycznych program wykorzystuje obiekty klasy *NumericalDiscretization*, *NumericalBin*, natomiast dyskretyzacja danych kategorycznych wykorzystuje klasy *CategoricalDiscretization* i *CategoryGroup*. Dane o dyskretyzacji są następnie zapisywane do bazy danych do relacji tymczasowych.

```
DiscretizationSpecification binningDetails[] = new DiscretizationSpecification[6];

// dyskretyzacja atrybutu WIEK
NumericalBin  bb0[] = new NumericalBin[6];
bb0[0] = new NumericalBin(1, (float)0.0, (float)18);
bb0[1] = new NumericalBin(2, (float)18, (float)29);

NumericalDiscretization bin0 = new NumericalDiscretization(bb0);
DiscretizationSpecification des0 = new DiscretizationSpecification("wiek", bin0);
binningDetails[0] = des0;

// dyskretyzacja atrybutu dochód
NumericalBin  bb1[] = new NumericalBin[6];
bb1[0] = new NumericalBin(1, (float)0, (float)1000);
bb1[1] = new NumericalBin(2, (float)1000, (float)1500);
…
NumericalDiscretization bin1 = new NumericalDiscretization(bb1);
DiscretizationSpecification des1 = new DiscretizationSpecification("dochód", bin1);
binningDetails[1] = des1;

LocationAccessData inputDataLocation =  new LocationAccessData(“demo_tx”, “mikolaj”);
Transformation.createDiscretizationTables(dmsConnection, ”mikolaj”, pds,
        binningDetails, ”num_temp”, ”cat_temp”, ”mikolaj”);
LocationAccessData resultViewLocation = new LocationAccessData(“decs_demo_tx”, ”mikolaj”);
Transformation.discretize(dmsConnection, ”mikolaj”, pds, “num_temp”, ”cat_temp”, ”mikolaj”, ”mikolaj”);
```

Zbudowanie specyfikacji funkcji eksploracji
Użytkownik tworzy nowy obiekt reprezentujący specyfikację danych wejściowych jako instancję klasy *LogicalDataSpecification*. Specyfikacja wykorzystania danych jest przechowywana w obiekcie klasy *DataUsageSpecification.* Wreszcie ustawienia funkcji są przechowywane w wystąpieniu klasy *AssociationRulesFunctionSettings* (dla algorytmu klasyfikacji jest to klasa *NaiveBayesSettings*).

```
LogicalDataSpecification lds = LogicalDataSpecification.create(
        "jdbc:oracle:thin:@dcs-mm:1521:ora9i", ”mikolaj”, “*”, ”demo_tx”, ”mikolaj”, pds);
DataUsageSpecification dus = DataUsageSpecification.create(lds.getMiningAttributes());
AssociationRulesFunctionSettings afs = new AssociationRulesFunctionSettings(0.01, 0, 2, lds, dus);
afs.store(dmsConnection, ”Function_AR”);
```

Zbudowanie modelu i odczytanie odkrytych reguł
Ostatnim krokiem wykonanym przez program jest pobranie wyników i ich zaprezentowanie użytkownikowi. Dane mogą być wczytywane do obiektów klasy *MiningRuleSet* (reguły asocjacyjne) lub *ClassificationTestResult* (klasyfikacja).

```
LocationAccessData buildDataAccessData =  new LocationAccessData(“disc_demo_tx”, ”mikolaj”);
MiningModel.build(dmsConnection, buildDataAccessData, pds, afs, ” Model_AR”);

ruleSet = AssociationRulesModel.getRules(dmsConnection, ”Model_AR”, (float) 0.01, 10);
ruleSet = AssociationRulesModel.getRulesBySupport(dmsConnection, ”Model_AR”, (float) 0.50, 10);
```

W efekcie wykonania powyższego programu w repozytorium serwera eksploracji zostają zapisane zarówno specyfikacja funkcji eksploracji jak i model eksploracji i wynik eksploracji. Poniżej przedstawiono fragmenty repozytorium.

Tabela ODM_MINING_FUNCTION_SETTINGS przechowuje obiekty specyfikacji funkcji eksploracji.

```
SQL> SELECT NAME, CREATION_TIMESTAMP, FUNCTION_NAME, ALGORITHM_NAME
SQL> FROM ODM_MINING_FUNCTION_SETTINGS;

NAME                CREATION FUNCTION_NAME        ALGORITHM_NAME
------------------- -------- -------------------- ------------------------------
Function_AR         02/05/17 associationRules     aPrioriAssociationRules
```

Obiekty prezentujące model eksploracji są przechowywane w repozytorium w relacji ODM_MINING_MODEL. Dla każdego modelu zapisywana jest jego nazwa, moment rozpoczęcia budowy modelu, moment zakończenia budowy modelu, nazwa funkcji i użytego algorytmu, lokalizacja danych wejściowych oraz nazwy relacji wynikowych.

```
SQL> SELECT NAME, STARTING_TIMESTAMP, ALGORITHM_NAME, DATA_LOCATION
SQL> FROM ODM_MINING_MODEL;

NAME                STARTING ALGORITHM_NAME                DATA_LOCATION
------------------- -------- ------------------------------ ------------------------------
Model_AR            02/05/17 aPrioriAssociationRules       mikolaj.disc_demo_tx
```

Informacje o odkrytych regułach są zapisywane do dynamicznie tworzonych tabel. Jedna relacja zawiera elementy należące do ciała reguły (poprzednika) a druga elementy głowy

reguły (następnika). Poza składnikami reguły zapamiętywane są również wartości parametrów wsparcia i ufności każdej reguły.

```
SQL> SELECT * FROM RT_0517021230148 WHERE RULE_ID = 1;

   RULE_ID ATTRIBUTE_NAME                     VALUE NUMBER_OF_ANTECEDENTS
---------- ------------------------------ ---------- --------------------
   SUPPORT CONFIDENCE ANTECEDENT_SUPPORT CONSEQUENT_SUPPORT
---------- ---------- ------------------ ------------------
         1 zawód                                14                   1
,000166694 ,001677852         ,099349892         ,000166694

SQL> SELECT * FROM AT_0517021230149 WHERE RULE_ID = 1;

   RULE_ID ATTRIBUTE_NAME                     VALUE
---------- ------------------------------ ----------
         1 dochód                                 4
         1 wiek                                   3
         1 stan_cywilny                           2
```

Podsumowanie

Oracle9*i* Data Mining jest zgodny z powstającym standardem o nazwie Java Data Mining. Standard ten jest dopiero w fazie tworzenia i obejmuje dotychczasowe propozycje zgłoszone w ramach Common Warehouse Metadata (propozycja Object Management Group), Predictive Mining Markup Language (propozycja Data Mining Group) oraz SQL/MM for Data Mining (propozycja ISO). Bez wątpienia jest to ciekawa i obiecująca propozycja, która w niedalekiej przyszłości umożliwi jeszcze szersze wykorzystanie technik eksploracji danych w ramach aplikacji bazodanowych, w szczególności integrację tych technik z magazynami danych powstającymi w oparciu o bazy danych Oracle.