**Inteligentne oprogramowanie
z wykorzystaniem SQL Server Data Mining**

Celem prezentacji jest pokazanie idei budowy inteligentnego systemu oraz na pokazanie w jaki sposób można wykonać taki system korzystając z możliwości SQL Server Data Mining.

Odbiorcami prezentacji będą głównie analitycy, inżynierowie i studenci informatyki i tym podobni. Grupy docelowe mają często całkiem sporą wiedzę odnośnie programowania w środowisku. NET oraz systemów bazodanowych (czasem też OLAP i hurtowni danych). Mają także pewną podstawową wiedzę wyniesioną ze studiów na temat metod sztucznej inteligencji. Często natomiast brakuje im trzech rzeczy: wiedzy o praktycznej przydatności metod sztucznej inteligencji w tworzeniu szerokiej gamy oprogramowania, wiedzy jakie metody i w jaki sposób powiązane w cały proces przetwarzania danych można użyć do konkretnych zagadnień oraz wiedzy o tym, jak to można osiągnąć korzystając z możliwości SQL Server Data Mining – rozwiązania, które już jest wbudowane w używany przez nich SQL Server oraz które od strony użycia programistycznego bazuje na znanych już im technologiach .NET i SQL. Tak więc wykorzystanie tej prezentacji pozwoli małym nakładem pracy i kosztów osiągnąć duże efekty.

Przez predykcję rozumiem w tym opracowaniu przewidywanie wielkości w zagadnieniach klasyfikacji, regresji, reguł asocjacyjnych, przynależności do klastrów oraz ekstrapolacja szeregów czasowych). Przez eksplorację danych rozumiem odkrywanie wiedzy w danych: reguł, trendów
i zależności. Predykcja i eksploracja danych są bardzo istotnymi sprawami, umożliwiającymi podniesienie efektywności firm poprzez wykrycie nowych prawidłowości i zastosowanie ich do poprawy procesu produkcyjnego bądź biznesowego (co autor może potwierdzić także własnym doświadczeniem z kilku wdrożeń systemów predykcyjnych w przemyśle). Prezentacja pokaże jak to można zrealizować przy pomocy narzędzi, które już mają i przy pomocy technologii w dużym stopniu już bazującej na ich wiedzy z zakresu C# i SQL, czyli jak można osiągnąć duży efekt małym nakładem pracy i bez dodatkowych inwestycji w infrastrukturę informatyczną.

Wykorzystane technologie: SQL Server 2014, Visual Studio 2013 z SQL Server Data Tools (główne technologie), dodatkowo Data Mining Add-Ins for Excel.

**Budowa modeli SQL Server Data Mining**

Podstawowe trzy elementy przy tworzeniu modelu: to źródło danych, struktura (mining structure)
i model. Źródło danych zawiera dane na których będzie działał model, choć można wybrać tylko część danych odpowiednim poleceniem SELECT. Struktura jest odpowiednikiem tabeli w bazie SQL i zawiera nazwy i typy pól modelu. Mogą być również pola (tabele) zagnieżdżone, czyli takie, gdzie pole nie jest typem prostym, tylko kolejną strukturą. Natomiast sam model zawiera informacje o tym, które pola struktury są wejściami, a które wyjściami dla algorytmu uczącego, jaki jest nasz algorytm uczący i jak dokonujemy podziału zbioru na zbiór uczący i testujący.

Mamy do dyspozycji 9 gotowych algorytmów: reguły asocjacyjne (Association rules), grupowanie (clustering), drzewa decyzyjne (Decision trees), regresja liniowa (Linear regression), regresja logistyczna (Logistic regression), Naïve Bayes, sieci neuronowe (Neural network), Sequence clustering i szeregi czasowe (Time series). Zaprezentuję tabelką z przykładowymi zastosowaniami poszczególnych algorytmów, pokazującą, że np. Clustering można użyć do analizy podobieństwa, np. grupując razem poszczególnych klientów i w ten sposób dostosowując do nich reklamę spersonifikowaną. Nie będę omawiał, jak który algorytm działa, bo nie ma na to czasu, a odbiorcy tej prezentacji pewnie trochę o tym wiedzą, a resztę w razie potrzeby się bez problemu douczą, gdyż informacja te jest dostępna w licznych źródłach i nie ma potrzeby jej powielania. Możemy również dopisać własne algorytmy - w piątej części prezentacji będzie pokazane, jak to zrobić.

Przedstawię trzy metody tworzenia modeli:    używając Data Mining Wizzard i Data Mining Designer, używając języka DMX, używając języka C# .

Data Mining Wizzard zapewnia nam interfejs graficzny, gdzie wszystko możemy sobie wybrać, przeklikać, przetestować, itd. i sprawdzić bardzo szybko różne opcje nawet bez znajomości tych opcji (bo wszystkie widać na ekranie) i bez znajomości DMX i C# oraz dodatkowo zobaczyć po nauczeniu modelu wynik jego działania. Zaletą DMX jest to, że możemy zautomatyzować tworzenie modeli poprzez odpowiednie skrypty oraz przeprowadzać łatwe modyfikacje. Zaletą tworzenia modeli w C# jest jest jeszcze większa możliwość ich dynamicznego tworzenia
i testowania w zależności od zmieniających się na bieżącą warunków, natomiast pracochłonność tego podejścia jest największa i przy tworzeniu pojedynczego modelu, jeśli nie potrzebujemy go mieć stworzonego w kodzie programu łatwiej użyć Data Mining Wizzard i Data Mining Designer.

**Tworzenie modeli w Data Mining Wizard**

Na przykładzie drzewa decyzyjnego zostaną przedstawione kolejne kroki tworzenia modelu oraz omówiona analiza modelu z wykorzystaniem Model Viewer.

**Tworzenie modeli w Data Mining Add-Ins for Excel.**

Pokażemy to na przykładzie predykcji szeregów czasowych oraz drzewa dezyzjnego. Robi się to dość podobnie, jak w Data Mining Wizard. Excel nie posiada Data Mining Engine i konieczne jest podłączenie się z Excela do SQL Servera. Excel służy tu tylko jeko end-user interface.

**Tworzenie modeli w języku DMX**

Język DMX (Data Mining eXtensions) został zaprojektowany tak, aby być maksymalnie podobnym do języka SQL. Składnia tworzenia struktur (mining structures) i modeli w DMX jest wzorowana na składni tworzenia tabel w SQL, tak więc każdy kto zna SQL będzie w stanie łatwo zrozumieć
i szybko opanować również DMX. Aby utworzyć strukturę należy uruchomić SQL Server Management Studio i zalogować się do Analytical Services Engine. Zademonstruję przykład modelu, którego zadaniem będzie zaproponowanie użytkownikowi internetowej agencji ubezpieczeniowej oferta będzie prawdopodobnie najlepsza dla niego. Użytkownik wprowadza pewne dane i na ich podstawie system ma zaproponować najlepiej dopasowaną ofertę.

CREATE MINING STRUCTURE [Klienci]

(

[UserID] LONG KEY,

[Imie] TEXT DISCRETE,

[Plec] TEXT DISCRETE,

[Wiek] LONG DISCRETIZED(Automatic,10),

[OkresEdukacji] DOUBLE CONTINUOUS,

[Zawod] TEXT DISCRETE,

[Miejscowosc] TEXT DISCRETE,

[Oferta] TEXT DISCRETE

)

Bardziej złożone struktury mogą posiadać tabele zagnieżdżone (nested tables). Następnym etapem jest dodanie modelu, gdzie zaznaczamy, które pola struktury są wejściami, które wyjściami (model może zawierać tylko podzbiór pól struktury) i jaki algorytm predykcji ma zostać użyty.

ALTER MINING STRUCTURE [Klienci]

ADD MINING MODEL [OfertaDecisionTree]

(

[UserID],

[Plec],

[Wiek],

[OkresEdukacji],

[Zawod],

 [Oferta] PREDICT

) USING Microsoft Decision Trees

W kolejnym kroku musimy załadować dane do struktury.

INSERT INTO MINING STRUCTURE [Klienci]

([UserID], [Imie], [Plec], [Wiek], [OkresEdukacji], [Zawod], [Miejscowosc], [Oferta])

OPENQUERY(UserData, ’SELECT UserID, Imie, Plec, Wiek, OkresEdukacji, Zawod, Miejscowosc FROM Klienci’)

**Predykcja z użyciem modeli SQL Server Data Mining**

Cechą inteligentnych programów jest to, że nie musimy znać żadnych reguł jakimi rządzi się dane zjawisko, aby napisać kod programu. Napisanie kodu w postaci reguł było by często bardzo uciążliwe, gdyż trzeba by stosować ogromną ilość odpowiednio zagnieżdżonych instrukcji if. Po drugie poznanie i zebranie tych reguł stanowiłoby ogromną pracę. Po trzecie na świecie wszystko się zmienia i raz napisany zestaw reguł należałoby okresowo weryfikować. Więc dlatego zastępujemy taki zakodowany w programie zestaw reguł modelem predykcyjnym, który sam się uczy na podstawie danych. Zbudowanie i nauczenie modelu zajmuje nieco czasu, ale nieporównanie mniej niż poprawne zakodowanie odpowiedniego systemu reguł.

Natomiast predykcja przy użyciu modelu jest bardzo szybka, bowiem następuje jedynie podanie aktualnego rekordu na wejście modułu i na tej postawie określenie przewidywanego wyjścia. Sam model jest bardzo mały – wielkość rzędu kilobajtów. Jedynie uczenie modelu jest bardziej czasochłonne w zależności od rozmiaru bazy danych, która może zawierać tysiące rekordów. Natomiast ze względu na wielkość modelu i bazy danych, odpytywanie modelu znacznie jest szybsze niż odpytywanie bazy danych. Predykcję można więc przeprowadzać w czasie rzeczywistym, bez zauważalnego opóźnienia.

Podam kilka przekładów zastosowań algorytmów sztucznej inteligencji. Pierwszy przykład: weryfikacja poprawności wprowadzanych danych do formularza (klasteryzacja). Drugi przykład: wspomaganie procesu produkcyjnego (sieć neuronowa). Trzeci przykład: wspomaganie diagnostyki problemów w sieci komputerowej (drzewo decyzyjne).

W części praktycznej zademonstruję jak stworzyć program, który wykorzystując modele utworzone w poprzednich częściach prezentacji pozwala na przewidywanie wartości danych. W zasadzie tylko dwie rzeczy trzeba wiedzieć, żeby napisać taki program: jak podłączyć się do modelu i jak go odpytać. Podłączamy się korzystając z ADOMD.NET, a odpytujemy model poleceniem PREDICTION JOIN (omówię to polecenie i jego opcje szczegółowo)

string query = @”SELECT

 [Oferta],

 PredictProbability([Oferta], 1) AS [Oferta = OK],

FROM [OfertaDecisionTree]

NATURAL PREDICTION JOIN

(SELECT ‘M’ AS [Plec], 45 AS [Wiek], 12 AS [OkresEdukacji],

 ‘Farmer’ AS [Zawod] As t”;

AdomdConnection connection = new AdomdConnection(connectionString);
connection.Open();

AdomdCommand queryCmd = new AdomdCommand(query, connection);

AdomdDataReader aReader = queryCmd.ExecuteReader();

if (aReader.Read())

 oferta = aReader.GetSTring(0);

.......

**Eksploracja danych OLAP**

Choć OLAP nie jest tematem tej prezentacji, występuje tu tylko w roli żródła danych do modelu predykcyjnego, to krótko przypomnę, czym jest OLAP, omawiając jego ideę i takie pojęcia jak fact, measure, dimension. Jednak zakładam, że odbiorca prezentacji ma już podstawową wiedzę (choćby z zajęć na uczelni) o tworzeniu modeli OLAP. Następnie omówię sytuacje, kiedy powinno się dokonywać predykcję wykorzystując jako uczący zbiór danych raczej dane zagregowane z kostek OLAP niż rekordy z tabeli SQL. W poniżej przedstawiono przykłady sytuacji, w których warto użyć OLAP jako źródło danych dla algorytmów predykcyjnych:

|  |  |
| --- | --- |
| **Sytuacja** | **Algorytm** |
| Pogrupowanie klientów w grupy w oparciu o produkty które kupują w poszczególnych porach roku i miejscowościach, na potrzeby optymalizacji produkcji i łańcucha dostaw | klasteryzacja |
| Przewidzenie, które punkty w sieci sprzedaży będą cię cechować największą dynamiką wzrostu lub spadku w poszczególnych działach produktów | drzewa decyzyjne |
| Identyfikacja przyczyn awarii w systemach telekomunikacyjnych w zależności od wersji hardware, software, medium transmisyjnego, wieku infrastruktury i regionu kraju.  | Naïve Bayes  |

W części praktycznej zademonstruję tworzenie kostki OLAP w projekcie Analytical Services
w Visual Studio oraz podpięcie jej jako źródła danych modelu Data Mining a następnie predykcję
z użyciem tak stworzonego modelu i interpretację wyników.

Do operacji na danych OLAP można też używać języka MDX, którego składnia jest podobna do DMX i do SQL i zamiast użycia wizzarda i designera, można wykorzystać MDX do operacji na kostkach OLAP, np. zwracania określonych zakresów danych, a DMX do budowania modelów predykcyjnych na tych danych. Jednakże podejście to nie będzie omawiane w prezentacji, a tylko
w jednym zdaniu zasygnalizuję taką możliwość, odsyłając zainteresowanych do literatury.

**SQL Server Data Mining Plug-Ins Algorithms**

Czasami możemy potrzebować jakiś szczególny algorytm, który np. dobrze radzi sobie dobrze
z zaszumionymi danymi, lub umożliwia rozpoznawanie obrazów z wyodrębnieniem różnych etapów generalizacji (np. sieć konwolucyjna), lub umożliwia generowanie reguł rozmytych (fuzzy rules), gdyż takie potrzebujemy w swoim programie. Są tu dwa wyjścia: pierwsze to włączyć istniejący program w cały proces predykcji danych (i to m. in. zastosujemy przy budowie całego procesu na końcu prezentacji), a drugie to rozszerzyć funkcjonalność SQL Server Data Mining. Wybór będzie oczywiście zależał od konkretnej sytuacji, ale to drugie rozwiązanie pozwoli na ujednolicenie dalszego sposobu postępowania z takim algorytmem.

Data Mining w SQL Server pozwala na rozszerzanie funkcjonalności poprzez tworzenie własnych algorytmów (plug-ins). Plug-in algorytm jest obiektem COM, który musi implementować zestaw odpowiednich interfejsów zawartych w SQLServer Data Mining. Natomiast korzystanie z tych algorytmów jest identyczne, jak korzystanie z algorytmów już wbudowanych w SQL Server Data Mining.

Są dwa sposoby tworzenie algorytmów plug-in: w języku C++ bezpośrednio jako obiekt COM, oraz w środowisku .NET, w języku C# z wykorzystaniem odpowiedniego wrappera “Data Mining Managed Plug-in Algorithm API” dostarczanego przez Microsoft. Zarówno korzystając z COM jak i z .NET należy zaimplementować odpowiednie interfejsy zgodnie z opisem w dokumentacji. Tworzenie algorytmów w kodzie zarządzalnym w C# może być łatwiejsze dla wielu programistów na co dzień pracujących w tym środowisku. Nie ma również różnic w funkcjonalności między dwoma podejściami, choć jest pewna różnica w wydajności na korzyść COM.

W następnym etapie należy zarejestrować obiekt COM z algorytmem i dokonać odpowiednich modyfikacji w konfiguracji Analysis Services. W przepadku korzystania z zarządzalnego API, zawiera on już projekt, który za nas sam stworzy obiekt COM z naszego algorytmu, zarejestruje go i uaktualni konfigurację Analysis Servicees. Ograniczę się tylko do podania istotnych informacji
i krótkiego zademonstrowania stworzonych już algorytmów ze zwróceniem uwagi na sposób implementacji interfejsów. Zainteresowany odbiorca znajdzie szczegółowy opis na stronie technet.microsoft.com.

**Technologie Pominięte w Prezentacji**

Dla zachowania kompletności prezentacji wspomnę, że zostały pominięte następujące technologie Microsoftu związane z SQL Server Data Mining: PowerShell, procedury składowane w Data Mining, Power BI. Pominięto także bazujące na SQL Server Business Intelligence rozwiązania firm trzecich jak Predixion czy Agile Design LLC. Zainteresowane osoby znajdą stosowne informacje w internecie. Celem prezentacji jest bowiem pokazanie skutecznej metody postępowania, a nie omówienie szczegółowe całej technologii.

**Inteligentny System**

Przedstawię przykładowy inteligentny system, zawierający kompletny proces eksploracji danych, składający się z odpowiednio połączonych różnych elementów omówionych w poprzednich częściach prezentacji. W tej części prezentacji nie będę raczej przedstawiał jak coś zrobić od strony programistycznej, bo to już zostało wszytko omówione, tylko zaprezentuję idee wraz
z uzasadnieniem oraz pokażę przykładową implementację takiego systemu zbudowanego w oparciu o SQL Server Business Intelligence Data Mining.



Rys. 1. Schemat przykładowego procesu eksploracji danych.

**Klasteryzacja** wykorzystana zostanie do podziału danych na kilka różnych obszarów, gdyż dane są na tyle zróżnicowane, że w tym przypadku będzie optymalnym podzielenie ich na kilka podzbiorów i dopiero na tych podzbiorach niezależne prowadzenie dalszych operacji.

**Selekcja informacji i redukcja szumu.** Przy ładowaniu danych ze źródła do struktury modelu w designerze można łatwo przeprowadzić selekcję atrybutów. Często jednak potrzebne jest bardziej zaawansowane przetwarzanie danych, np. selekcja wektorów, by ograniczyć zarówno rozmiar przetwarzanych danych, jak i zakłócenia, albo odpowiednie transformaty pozwalające doprowadzić dane do postaci optymalnej dla algorytmu predykcyjnego. Osobiście preferuję tu włączenie w ciąg procesu programu RapidMiner, który będzie czytać dane z bazy SQL, przetwarzać informacje i wynik także zapisywać do bazy. Nawet jeśli ktoóś uważa, że jest to najbardziej eleganckie rozwiązanie, to jego zaletą jest łatwość implementacji – nie muszę intensywnie myśleć, jak powtórnie zrobić, to zo już jest zrobione i gotowe do użycia.

**Komitety modeli.** Omówię dlaczego wynik predykcji komitetu modeli jest przeważnie lepszy niż wynik najlepszych z modeli składowych i praktycznie pokażę jak szybko i łatwo można bezpośrednio w SQL Server Data Mining skonstruować model wykorzystujący głosowanie (majority voting) algorytmów składowych (można oczywiście wykorzystać i inne schematy komitetów, ale ten będzie najprostszy do pokazania).

**Ekstrakcja reguł z danych.** Omówię dlaczego zrozumienie wyników predykcji jest czasami niezbędne i pokażę, jak to można uzyskać wprost z drzewa decyzyjnego w modelu SQL Server Data Mining oraz jak można wyciągnąć reguły logiczne z sieci neuronowej standardowo zaimplementowanej w modelu SQL Server Data Mining (w trochę bardziej złożony sposób, wykorzystujący metodę opracowaną przez R. Setiono). Nie będę dokładnie omawiał tej metody ekstrakcji reguł z sieci neuronowej, gdyż głównie chodzi tu o pokazanie na tym przykładzie sposobu dostępu do parametrów modelu i możliwości dalszego ich wykorzystania.