STUDIA INFORMATICA 2009
Volume 30 Number 2B (84)

Kacper PABJAŃCZYK, Adam PELIKANT
Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych

IMPLEMENTACJA ALGORYTMÓW UŻYTKOWNIKA W ŚRODOWISKU BUSINESS INTELLIGENCE SQL SERVER 2008

Streszczenie. Artykuł porusza problem integracji algorytmów zgłębiania danych opracowanych przez użytkownika ze środowiskiem SQL Server Analysis Services. Przedstawione zostały kroki niezbędne do opracowania i wdrożenia algorytmu z zastosowaniem środowiska .NET. Pokazano wymagania formalne organizacji procedur oraz metody inkapsulacji w przypadku zastosowania języka innego niż C++. Wskazano na metody przekazywania danych do części prezentacyjnej oraz jej organizację formalną.

Słowa kluczowe: inteligencja obliczeniowa, SQL Server Analysis Services, DMPluginWrapper, zgłębianie danych, hurtownie danych

USER ALGORITHMS IMPLEMENTATION AT THE BUSINESS INTELLIGENCE SQL SERVER 2008 PLATFORM

Summary. Article propels the problem of user data mining algorithms integration with the environment SQL Server Analysis Services. The indispensable steps to work out and the initiating the algorithm with use of platform .NET were introduced. The formal requirements of organization of procedures as well as method of encapsulation were showed in case of use of language different than C++. The methods of data transfers to preventative unit were showed as well as her formal organization.

Keywords: Business Intelligence, SQL Server Analysis Services, data mining, DMPluginWrapper, data warehousing

1. Wstęp

W dobie szybkiego rozwoju technologii i informatyzacji wiele firm i instytucji zbiera powstałe w procesie funkcjonowania dane, gromadząc je w systemach baz danych. W minio-

nej dekadzie, m.in. dzięki wykorzystaniu systemów biznesowych, takich jak aplikacje finansowe, systemy zarządzania zasobami (ERP – Enterprise Resource Planning), systemy zarządzania relacjami z klientem (CRM – Customer Relationship Management) wygenerowano olbrzymie ilości danych, które przyczyniły się do powstania organizacji bogatych w dane, lecz ubogich w wiedzę. Wykładniczy postęp przyrostu zbiorów danych dodatkowo utrudniał ich analizę i praktyczne wykorzystanie. Narzędziem umożliwiającym efektywne wykorzystanie danych, przechowywanych w bazach jest data mining, czyli proces eksploracji danych [1].

W literaturze odnaleźć można wiele różnych definicji procesu eksploracji danych. Zgodnie z literaturą eksploracja danych to wykrywanie wzorców, czyli stwierdzenie, czy wśród danych można jakieś wzorce wyróżnić. Innymi słowy, jest to proces uwidaczniania informacji z dowolnie rozłożonych danych [2]. Inną definicję formułuje Cichosz Paweł ,który proces ten określa jako: „zastosowanie maszynowego uczenia się do odkrywania zależności w bazach danych". Zdaniem Cichosza do eksploracji danych i odnajdywania zależności między nimi wykorzystuje się metody z dwóch obszarów – systemów uczących się i statystyki [3].

Głównym celem data miningu jest odkrywanie z dostępnych zasobów danych najróżniejszego typu wzorców, uogólnień, reguł, prawidłowości i współzależności między danymi. Proces ten ma na celu zwiększenie znaczenia gromadzonych przez organizację danych, przez odkrywanie wartości zawartej implicite w tych zasobach, czyli przekształcenie (transfer) danych w wiedzę możliwą do interpretacji i wykorzystania przez użytkownika merytorycznego [4].

Celem poniższej pracy jest pokazanie sposobu zaimplementowanie algorytmu aproksymacji średniokwadratowej wielomianem dowolnego rzędu, wykorzystując do tego mechanizmy dostarczone wraz z Microsoft SQL Server Analysis Services (SSAS). Wybór padł na ten konkretny algorytm, ponieważ jest on stosunkowo prosty do implementacji, a zarazem można już za pomocą niego w pewnym ograniczonym stopniu zaobserwować pewne zachowania się danych. Pierwsza część pracy skupia się na dokładnym przedstawieniu sposobu implementacji algorytmu w środowisku Business Intelligence oraz rejestracji tego algorytmu, natomiast druga część pracy pokazuje, w jaki sposób napisać program, który umożliwia reprezentację graficzną wyników zwróconych przez algorytm (później dla wygody program ten nazywany jest przeglądarką). Oprogramowanie, z którego korzystano przy tworzeniu tej pracy to:

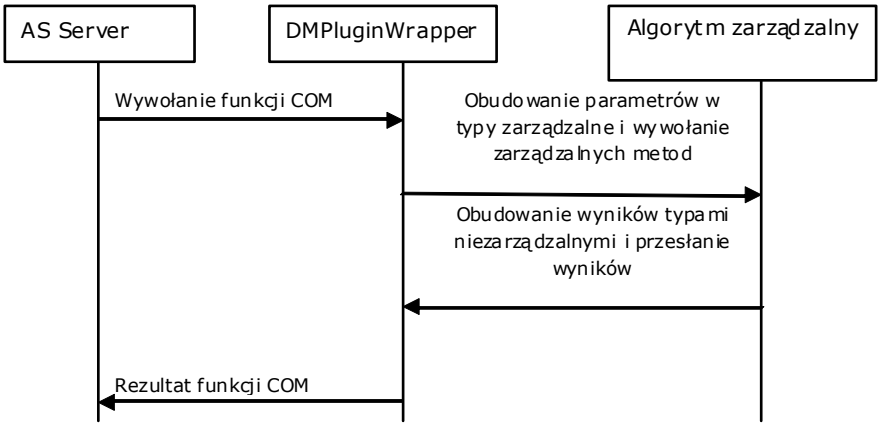
- Microsoft SQL Server 2008,
- Microsoft Visual Studio 2008 Professional Edition.

2. Implementacja algorytmu użytkownika

Microsoft SQL Server 2008 Analysis Services udostępnia możliwość integracji tego środowiska z algorytmami napisanymi przez firmy trzecie (*third party companies*) na zasadzie „wtyczek" (*plug-in*). Algorytmy takie są tworzone jako obiekty COM i mogą być pisane w języku kompatybilnym z technologią COM, np.: C++ [5]. Jednak ta technologia nie pozwala korzystać z języków środowiska .NET, czyli Visual Basic lub C#. W roku 2006 firma Microsoft udostępniła możliwość korzystania z platformy .NET przy tworzeniu algorytmów do „data miningu" przez stworzenie bibliotek, które obudowują obiekt COM obiektami .NET (*wrapper*).

Podczas implementacji algorytmu została wykorzystana biblioteka DMPluginWrapper.dll, która pozwala wykorzystywać język C# do tworzenia algorytmu. Klasa `DMPlugin-Wrapper` implementuje interfejsy COM, które są wymagane do działania algorytmu oraz „tłumaczy" je na odwołania do języka CLI (*Common Language Infrastructure* – język najniższego poziomu dla platformy Microsoft .NET odczytywalny przez człowieka).

Analysis Services używa interfejsów COM zaimplementowanych przez `DMPlugin-Wrapper` do nauki i przetwarzania danych z modelu. Na początku Analysis Services wykorzystuje plik inicjalizujący (msmdsrv.ini) do sprawdzenia, jakie algorytmy do zgłębiania danych są dostępne. W wersji pierwotnej plik ten zawiera wszystkie algorytmy wbudowane, udostępnione przez firmę Microsoft. Zasadę działania DMPluginWrappera przedstawia rys. 1 [6].



Rys. 1. Zasada działania DMPluginWrappera
Fig. 1. Principle of DMPluginWrapper working

Każdy algorytm bazuje na minimalnym wzorcu, który pozwala środowisku Analysis Services na wykorzystanie tego algorytmu.

Podstawą algorytmu są trzy klasy: klasa Metadanych, klasa Algorytmu oraz klasa Nawigatora.

Klasy te dziedziczą po klasach udostępnionych przez DMPluginWrapper.

- Klasa Metadata dziedziczy z klasy AlgorithmMetadataBase;
- Klasa Algorithm dziedziczy z klasy AlgorithmBase;
- Klasa AlgorithmNavigator dziedziczy z klasy AlgorithmNavigationBase.

Wszystkie typy są zdefiniowane wewnątrz DMPluginWrapper jako część przestrzeni nazw Microsoft.SqlServer.DataMining.PluginAlgorithm. Klasa Metadata dostarcza serwerowi Analysis Services informacji o algorytmie, takich jak: nazwa, opis, jakie typy parametrów algorytm przyjmuje itd.

Najważniejsze metody klasy Metadata to:

- `public override string GetServiceName()` – metoda zwraca nazwę implementowanego algorytmu,
- `public override string GetViewerType()` – metoda zwraca, jakiego typu przeglądarka ma być używana do wyświetlania wyników algorytmu.

Klasa Algorithm implementuje logikę algorytmu.

Najważniejsza metoda klasy Algorithm to:

- `protected override void Predict()`– metoda pozwala na dokonywanie predykcji danych.

Klasa AlgorithmNavigator implementuje możliwość poruszania się po wynikach zwróconych przez klasę algorytmu.

Implementacja samego algorytmu była trudnym zagadnieniem, ponieważ niestety nie istnieje fachowa literatura, która pokazuje precyzyjnie metodę postępowania. Dotyczy to zwłaszcza stosowania DMPluginWrappera, który jest bardzo pomocnym narzędziem, ale przez wzgląd na brak sensownej dokumentacji do większości rowiązań problemów niestety trzeba dojść samemu. Istnieją wydane przez Microsoft pomoce dotyczące implementacji zarządzalnego algorytmu, ale są one napiasne dla osób, które miały okazję pisać niezarządzalne algorytmy do data miningu w języku C++ na podstawie obiektów COM. Niestety osoba, która dobrze poznała język C#, nie będzie w stanie od razu przystąpić do implementacji algorytmu, wykorzystując DMPluginWrapper, co związane jest ze stopniem skomplikowania wymogów narzuconych przez wrappera. Także wiele problemów stwarza sama architektura SQL Server'a, z racji komunikowania się z innymi procesami, a raczej z powodu braku tej komunikacji, ale ta kwestia zostanie szerzej przedstawiona w części poświęconej implementacji przeglądarki.

Na listingu 1 przedstawiam część zaimplementowanej klasy Metadate.

Listing 1

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
using Microsoft.SqlServer.DataMining.PluginAlgorithms;
```

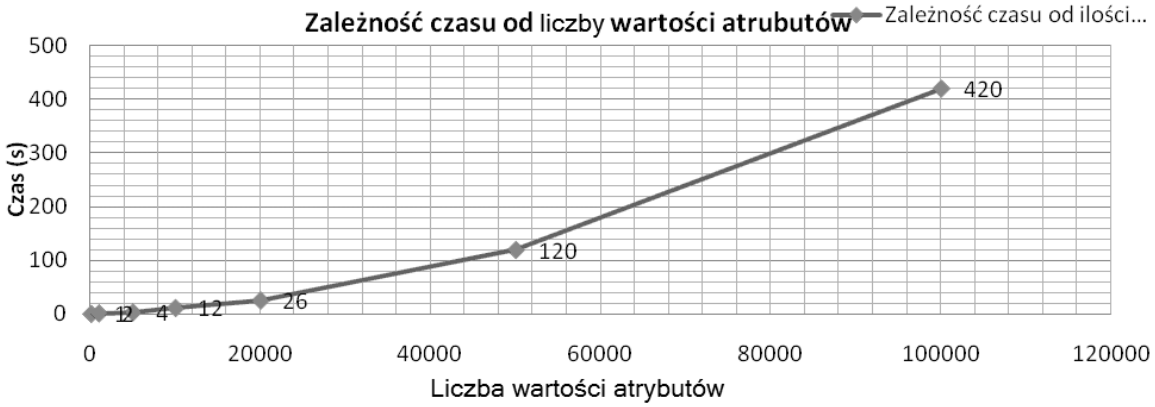
```
namespace Algorytm_Aproksymacji_Sredniokwadratowej
{
    [ComVisible(true)]
    [Guid("14C67C82-1899-4753-B257-FDA84B3EF3B4")]
    [MiningAlgorithmClass(typeof(Algorithm))]
    public class Metadata : AlgorithmMetadataBase
    {
        protected MiningParameterCollection parameters;
        internal static MiningModelingFlag MainAttributeFlag =
        MiningModelingFlag.CustomBase + 1;

        public Metadata()
        {
            parameters = DeclareParameters();
        }

        static public MiningParameterCollection DeclareParameters()
        {
            MiningParameterCollection parameters
                    = new MiningParameterCollection();
            MiningParameter param=null;
            param = new MiningParameter("StopienWielomianu",
                                "Parametr określa stopień wielomianu,
                               którym będą aproksymowane dane",
                                "1",
                                "[1, 32]",
                                true,
                                true,
                                typeof(System.Int32));
            parameters.Add(param);
            return parameters;
        }

        public override string GetServiceName()
        {
            return "Algorytm_Aproksymacji_Sredniokwadratowej";
        }

        public override string GetDisplayName()
        {
            return "Algorytm Aproksymacji Sredniokwadratowej";
        }

        public override string GetServiceDescription()
        {
            StringBuilder serviceDescription = new StringBuilder();
            serviceDescription.Append("Algorytm służący do aproksymacji metodą
                      najmniejszych kwadratów \n");
            serviceDescription.Append("aproksymujący wielomianem dowolnego rzędu
                      \n");
            return serviceDescription.ToString();
        }

        public override PlugInServiceType GetServiceType()
        {
            return PlugInServiceType.ServiceTypeOther;
        }
}
```

3. Wnioski dotyczące implementacji algorytmu

Ponieważ głównym zadaniem prac nie było opracowanie konkretnego algorytmu zgłębiania danych, a jedynie opracowanie metodyki wdrażania i integracji ze środowiskiem Analisys Services, prace były prowadzone na bazie algorytmu aproksymacji wielomianem rzędu *n*. *Notabene* algorytm ten faktycznie nie został zaimplementowany w SQL Business Intelligence. Dostępna jest tylko aproksymacja liniowa. Po wdrożeniu algorytmu zrealizowane zostały testy wydajności. Jako pierwsze przeprowadzono badania wydajności w zależność od liczby wartości atrybutów. Dla zmiennej liczby wartości atrybutów czas obliczeń kształtował się w sposób przedstawiony w tabeli 1, a zilustrowany na rys. 2. Badania były przeprowadzone dla wielomianu aproksymującego stopnia dziesiątego.

Tabela 1
Zależność czasu obliczeń od ilości wartości atrybutów

Liczba wartości atrybutów	Czas obliczeń (s)
100	1
1000	2
5000	4
10000	12
20000	26
50000	120
100000	420



Rys. 2. Zależność czasu obliczeń od liczby wartości atrybutów
Fig. 2. Dependence of calculations time of from attribute values number

Wszystkie pomiary były wykonywane w takich samych warunkach początkowych, tzn. serwer Analysis Services po każdym przeliczeniu partii danych (odpowiednio 100,1000,5000 itd. punktów) był uruchamiany ponownie, aby zapewnić, że nie będą wykorzystane wyniki pośrednie z wcześniej wykonanych obliczeń oraz że zwolnione zostały wszystkie zasoby pamięci. Jak widać na powyższym wykresie, czas obliczania wydłuża się znacząco wraz ze wzrostem liczby punktów, przyjmując charakterystykę zbliżoną do wykładniczej.

Kolejnym badaniem było sprawdzenie wydajności w zależności od stopnia wielomianu aproksymującego. Wyniki tej analizy zawiera tabela 2, a postać graficzną przedstawia rys. 3. Badania były przeprowadzone dla stałej ilości 100000 punktów.

Tabela 2

Zależność czasu obliczeń od stopnia wielomianu

Stopień wielomianu	Czas(s)
1	420
5	415
10	416
15	423
20	419
25	418
32	425

Rys .3. Zależność czasu obliczeń od stopnia wielomianu
Fig. 3. Dependence of calculations time from polynomial stages

Wszystkie pomiary były wykonywane w takich samych warunkach początkowych, tzn. serwer Analysis Services po każdym przeliczeniu partii danych (odpowiednio 1, 5, 10 itd. stopień wielomianu) był ponownie uruchamiany. Jak widać na wykresie na rys. 3, czas obliczania jest praktycznie niezależny od stopnia wielomianu, ponieważ dla tak dużej rozpiętości stopni wielomianów wyniki nie różnią się od siebie znacząco.

Na podstawie powyższych badań stwierdza się, że znaczący wpływ na szybkość działania całego procesu nie ma sam algorytm aproksymacji średniokwadratowej, tylko szybkość pobierania danych z bazy danych. Możemy tak powiedzieć, ponieważ różnice pomiędzy czasami następnych stopni wielomianu były słabo zauważalne i jedyny zaobserwowany wpływ na czasy miało zaburzenie samego serwera bazy danych. Poza tym działanie samego serwera Analysis Services było czasami zaskakujące. W trakcie badań zaobserwowano, że jeżeli wykonywało się kilka przebiegów jeden po drugim, serwer znacząco zwalniał. Najbardziej paradoksalnym przypadkiem było uruchomienie algorytmu do aproksymacji 100000

punktów wielomianem 10 stopnia, a następnie 1000 punktów także wielomianem 10 stopnia. Wynik był następujący: dla 100000 punktów obliczenia wykonywały się około 420 sekund, natomiast dla 1000 wykonywały się ok. 1100 sekund, co upewniło nas, że aby uniknąć wpływu procesów drugoplanowych, należy zapewnić pełne zwolnienie pamięci, by konieczne było ponowne uruchomienie serwera przed wykonaniem każdego z testów.

4. Implementacja okna wyświetlania wyników – Viewer

Microsoft Analysis Services Datat Minig Plug-in Framework definiuje niezbędne interfejsy oraz informacje, które należy wprowadzić do rejestru, dzięki którym istnieje możliwość budowanie programu zintegrowanego z SQL Server Business Intelligence Development Studio, służącego do reprezentacji testowej lub graficznej wyników działania algorytmów do zgłębiania danych, czyli przeglądarki. Okno wyświetlania wyników jest kontrolką napisaną w języku platformy .NET, więc może być tworzony w języku C#, Visual Basic .NET lub J#, a także może być kontrolką formantu ActiveX [7]. Jedno okno może zostać użyte do wyświetlania danych z kilku różnych algorytmów, przez co można tworzyć kompleksowe rozwiązania w szybki i efektywny sposób. W przedstawionym przykładzie przeglądarka wyników została napisana w języku C#, w środowisku Visual Studio 2008. Okno wyświetlania wyników będzie składało się z dwóch komponentów:

- Pola typu RichTextBox służącego do wyświetlania obliczonych współczynników przez algorytm aproksymacji średniokwadratowej w postaci funkcji $f(x_i) = a_i x^{i-1}$.
- Komponentu ZedGraph służącego do rysowania wykresów funkcji.

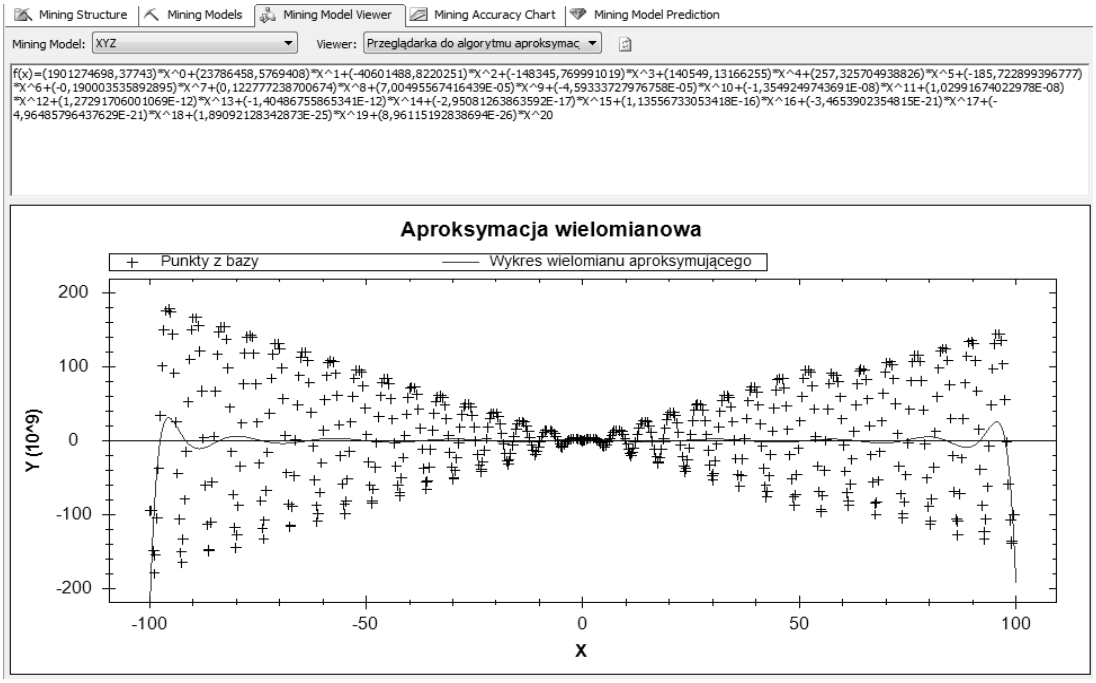
Okno przeglądania wyników jest to komponent typu UserControl, który ponadto implementuje interfejs IMiningModelViewerControl. Należy zaimplementować trzy właściwości oraz dwie metody, które zawierają się w interfejsie IMiningModelViewerControl. Zaimplementowane właściwości to:

- ConnectionString – służy on do podłączenia się przeglądarki do Analysis Services.
- MiningModelName – służy do przekazania do przeglądarki nazwy analizowanego modelu.
- ServiceProvider – służy do ustawienia przez serwer sposobu podłączenia się do Analysis Services.

Natomiast zaimplementowane metody pozwalają na:

- LoadViewerData – metoda wywoływana przez serwer za każdym razem, kiedy przeglądarka ma wyrenderować model.
- ViewerActivated – jeżeli przeglądarka posiada więcej niż jedno okno, metoda określa, czy okno przeglądarki jest aktywne.

Poniżej przedstawiono efekt działania przeglądarki.



Rys. 4. Widok przykładowej aproksymacji w oknie Viewera
Fig. 4. View of pricipal aproksimation in thr Viewer window

Okno wyświetlania wyników tak jak algorytm nie jest samodzielnym programem, tylko biblioteką dll, której metody są uruchamiane przez odpowiednie środowisko uruchomieniowe, którym jest SQL Server Business Intelligence Development Studio. Nowo dodany projekt nie będzie zawierał żadnych plików wykonywalnych tylko biblioteki dll (po skompilowaniu). Na listingu 2 przedstawiam część implementacji klasy przeglądarki.

Listing 2

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Data.OleDb;
using System.Text;
using System.Windows.Forms;
using Microsoft.DataWarehouse.Interfaces;
using System.Collections.Generic;
using System.Data.SqlClient;
using ZedGraph;
using System.Configuration;
namespace Viewer
{
    public partial class ViewerForm : UserControl, IMiningModelViewerControl
    {
        private string sqlASConnectionString = "";
        private string miningModelName = "";
        private IServiceProvider serviceProvider;

        private const int NODE_DISTRIBUTION = 16;
        private const int ATTRIBUTE_NAME = 0;
        private const int ATTRIBUTE_VALUE = 1;
```

```
public ViewerForm()
{
    InitializeComponent();
}

#region IMiningModelViewerControl Members

public string ConnectionString
{
    get
    {
        return this.sqlASConnectionString;
    }
    set
    {
        if (this.sqlASConnectionString != value)
            this.sqlASConnectionString = value;
    }
}

public bool LoadViewerData(object context)
{
    bool returnValue = false;
    try
    {
       List<NodeDescription> lNodeDescription = null;
       lNodeDescription = GetNodeDescriptionFromSchema();
       richTextBox.AppendText(ExtractFormulaFromDist(lNodeDescription));
       Start(lNodeDescription);
       return returnValue = true;
    }
    catch (Exception)
    {
    }
    return returnValue;
}

public string MiningModelName
{
    get
    {
        return this.miningModelName;
    }
    set
    {
        if (this.miningModelName != value)
            this.miningModelName = value;
    }
}

public IServiceProvider ServiceProvider
{
    get
    {
        return this.serviceProvider;
    }
    set
    {
        if (this.serviceProvider != value)
            this.serviceProvider = value;
    }
}
```

```
public void ViewerActivated(bool isActivated)
{
    throw new NotImplementedException();
}
```

5. Wnioski

Implementacja przeglądarki, tak samo jak implementacja algorytmu, była trudnym zagadnieniem wynikającym z braku fachowej literatury, która opisywałaby problem w sposób rzetelny i dokładny. Różnica w trudności polega głównie na tym, że sam kod przeglądarki jest dużo prostszy, ponieważ, jak było widać, w zaprezentowanym rozdziale implementuje on tylko jeden interfejs IMiningModelViewerControl, a nie trzy klasy wirtualne. Jednak największą trudnością, o której zostało wspomniane w rozdziale poświęconym algorytmowi, jest kwestia komunikacji pomiędzy algorytmem a oknem wyświetlania wyników. Niestety, architektura serwera SQL Server 2008 nie pozwala innym procesom korzystać z wydzielonego, współdzielonego obszaru pamięci, co uniemożliwia w prosty sposób komunikację między przeglądarką a algorytmem. Problem ten został rozwiązany w ten sposób, że wyniki, czyli w tym przypadku tablica z wartościami współczynników wielomianu aproksymującego, zostały zapisane jako statystyki węzła, a następnie zostały pobrane przez przeglądarkę wykorzystując zapytanie typu MDX. Jednak autor uważa, że przekazywanie w ten sposób danych nie jest najbardziej optymalną metodą, z racji tego, że musi zmieniać strukturę statystyk, przeznaczoną oryginalnie do innych celów. Także kwestia zarejestrowania przeglądarki może sprawiać pewne kłopoty, ponieważ jest to proces bardzo słabo udokumentowany, do którego dochodzi się metodą prób i błędów. Należy bezwzględnie pamiętać o zarejestrowaniu przeglądarki w GAC (*Global Assembly Cache*), rejestrze oraz skopiowaniu bibliotek we wskazane miejsca. Bez tych czynności okno wyświetlania wyników albo nie będzie pokazywało się na liście dostępnych przeglądarek albo, co gorsza, będzie zawieszało SQL Server Business Intelligence Development Studio. W samym procesie programowania także ważną czynnością jest debugowanie. Niestety, w przypadku pisania przeglądarki proces debugowania jest jeszcze bardziej skomplikowany i nie intuicyjny jak w przypadku implementacji algorytmu. Generalnie, aby zdebugować przeglądarkę, należy wykonać prawie te same czynności co w przypadku debugowania algorytmu, jednak przy wyborze procesu, do którego trzeba podłączyć Visual Studio 2008, wybieramy SQL Server Business Intelligence Development Studio, co sprowadza się praktycznie do tego że debugujemy Visual Studio za pomocą drugiego Visual Studio. Niestety, innej możliwości debugowania nie ma.

BIBLIOGRAFIA

1. Makhoul J., Kubala F., Schwartz R., Weischedel R.: Performance measures for information extraction. In Proceedings of DARPA Broadcast News Workshop, 1999, s. 249÷252.
2. Michalewicz Z., Fogel D. B.: Jak rozwiązać, czyli nowoczesna heurystyka. WNT, 2006.
3. Cichosz P.: Systemy uczące się. WNT, 2000.
4. Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P.: Hurtownie danych. WSIP, 2003.
5. http://msdn.microsoft.com/en-us/library/ms345112(SQL.90).aspx – dokumentacja do tworzenia algorytmu (C++).
6. http://www.sqlserverdatamining.com/ssdm/Home/Tutorials/tabid/57/Default.aspx – dokumentacja do tworzenia algorytmu w kodzie zarządzalnym.
7. http://msdn.microsoft.com/en-us/library/ms345129(SQL.90).aspx – dokumentacja do tworzenia przeglądarki.

Recenzent: Dr inż. Dariusz Mrozek

Wpłynęło do Redakcji 4 lutego 2009 r.

Abstract

To sum up this article, it can be argued that the assumptions made in the introduction has been fully completed. According to these assumptions has been implemented Least squares approximation algorithm in the Business Intelligence Sql Server 2008 enivironment.

The project is divided into two independent parts: algorithm, which is represented by the library dll and the viewer, which is also a dll library. Thus prepared, you can register in the library system, using the built-in mechanisms for Windows and then when you start SQL Server Business Intelligence Development Studio, you can analyze the data. Of course, the algorithm is very simple and there is no use in professional data mining, however, the foundation work has not been focusing on the algorithm as such, only on presentation of how to implement any complex algorithm. Because of the choice so simple algorithm, were not necessarily used all the possibilities for Analysis Services, such as full implementation of prediction, or Drill-through. Also, the viewer does not belong to a specially complicated, but the author believes that anyone who reads this article will stand up to implement their solutions.

Adresy

Kacper PABJAŃCZYK: Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, ul. Stefanowskiego18/22, 90-924 Łódź, Polska, kacper.pabjanczyk@gmail.com.

Adam PELIKANT: Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, ul. Stefanowskiego18/22, 90-924 Łódź, Polska, apelikan@p.lodz.pl.