Joris Valkonet

Voorspel de toekomst

JOUW EIGEN ALGORITME IN SQL SERVER ANALYSIS SERVER

Datamining en rapportages zijn hot. Met behulp van datamining in SQL Server Analysis Services (SSAS) liggen toepassingen als het opsporen van frauduleuze transacties tot en met het ontdekken van potentiële klanten binnen handbereik. Sinds SQL Server 2005 is het ook mogelijk jouw eigen plug-in datamining-algoritme toe te voegen aan Analysis Services. In dit artikel beschrijft de auteur de implementatie van een eigen datamining-algoritme in SSAS.

Stel je eens voor dat je voor een bank werkt die leningen verstrekt aan iedereen die maar een lening wil hebben. Na een paar jaar werken voor deze bank, merk je dat er veel klanten zijn die hun lening niet terugbetalen, omdat ze failliet zijn verklaard. Dit is erg vervelend voor deze klanten en uiteraard ook voor de bank. Om in de toekomst deze ellende te voorkomen, besluit je gegevens van alle leningen die zijn verstrekt in de afgelopen jaren te verzamelen en te analyseren met behulp van je eigen datamining-algoritme in SQL Server Analysis Services. Het doel is te voorspellen of klanten failliet zullen worden verklaard. Je verzamelt de klantgegevens in een tabel van drie kolommen: de leeftijd en het salaris van de klant op het moment van het afsluiten van de lening, en een waarde die aangeeft of de klant failliet is verklaard. Zie tabel 1 voor de verzamelde gegevens.
Voor datamining zijn er vele definities in omloop, maar een van de meest gangbare definities van datamining is het zoeken naar patronen in grote hoeveelheden gegevens. Aan de hand van deze patronen kunnen voorspellingen worden gedaan. Het zoeken naar patronen in data kan op verschillende manieren en met verschillende doeleinden. In ons geval zijn we alleen geïnteresseerd in een voorspelling welke klanten failliet zullen worden verklaard. Het algoritme dat we in dit artikel gebruiken, is het klassieke perceptron-algoritme, zie de links onder aan dit artikel voor meer informatie.

Microsoft's Data Mining Framework

Voordat we beginnen met de implementatie van het algoritme, bekijken we eerst hoe we het algoritme gaan opbouwen. Het is mogelijk om je eigen datamining-algoritme toe te voegen aan SSAS met behulp van voorgedefinieerde COM-interfaces. Al vrij snel na de release van SQL Server 2005 heeft Microsoft een managed plug-in-wrapper beschikbaar gesteld die alle COM-zaken voor je regelt. Dankzij de plug-in-wrapper die we in dit artikel gebruiken, kunnen we ons op het algoritme zelf richten en laten we de COM-interfaces voor wat ze zijn.
Plug-in datamining-algoritmen in SSAS behoren drie dingen te doen:

1. Het algoritme beschrijven
2. Patronen ontdekken, opslaan en voorspellingen doen
3. De patronen beschikbaar stellen aan de buitenwereld.

Salaris	Leeftijd	Falliet
99000	40	0
60000	50	0
30000	65	0
96000	29	0
75000	60	0
51000	40	1
13000	35	1
10000	65	1
30000	35	1
60000	28	1

Tabel 1.

Bij het ontwikkelen van ons algoritme maken we gebruik van twee abstracte klassen uit de plug-in-wrapper. De eerste is AlgorithmMetadataBase, waarin de features van het algoritme kunnen worden beschreven. De tweede klasse is AlgorithmBase, waarin we verzamelde informatie inlezen, patronen ontdekken en voorspellingen afhandelen.

Informatie aanbieden aan het algoritme

Voordat data aangeboden kunnen worden aan een datamining-algoritme, moet er in SSAS eerst een datamining-structuur worden gemaakt door de eindgebruiker. In een datamining-structuur specificeer je wat de types zijn van de kolommen die je aanbiedt aan een algoritme. In het Microsoft Data Mining Framework zijn er in totaal vijf types van waardes mogelijk:

1. Discrete waardes; dit zijn voornamelijk categorieën.
2. Continue waardes; dit zijn voornamelijk meetwaardes.
3. Gediscretiseerde waardes; dit zijn continue waardes die verdeeld zijn in categorieën.
4. Een key; een primary key van de aangeboden records.
5. Een nested table; dit is een tabel in een kolom voor elke rij van een tabel.

Niet elk algoritme kan met hetzelfde type informatie omgaan. Een eigenschap van het perceptron-algoritme is dat het alleen discrete waardes kan voorspellen vanuit continue inputkolommen. De tabel met gegevens die we hebben verzameld, bestaat uit twee continue inputkolommen en een discrete predict-kolom, namelijk salaris, leeftijd en een indicatie van het faillissement. SSAS zal voor validatie van de datamining-structuur zoeken naar de specificatie van het algoritme. Om de specificatie vast te leggen, maken we een klasse Metadata als extensie van de klasse AlgorithmMetaData. Het vastleggen van het type van de inputkolommen en de predict-kolom doen we aan de hand van respectievelijk de override-methode GetSupInputContentTypes()

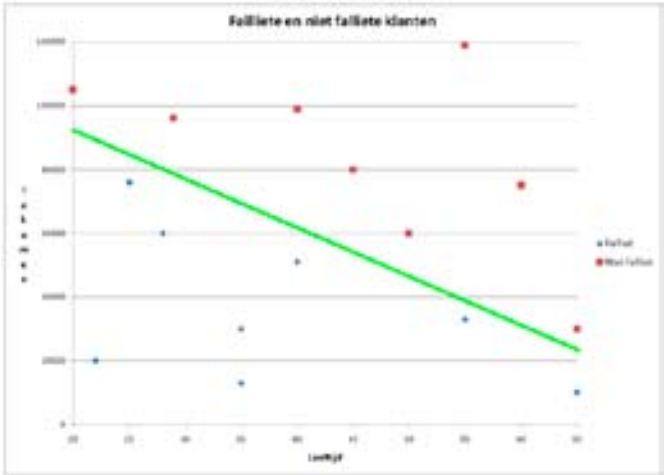
```
public override MiningColumnContent[] GetSupInputContentTypes()
{
  //specificeer de input kolommen
  return new MiningColumnContent[] {
    MiningColumnContent.Continuous
  };
}

public override MiningColumnContent[] GetSupPredictContentTypes()
{
  //specificeer de predict kolommen
  return new MiningColumnContent[] {
    MiningColumnContent.Discrete
  };
}
```

Codevoorbeeld 1.

en GetSupPredictContentTypes(); zie codevoorbeeld 1.
Na validatie van de aangeboden datamining-structuur kan het algoritme worden aangeroepen. Door de override-methode CreateAlgorithm(ModelServices model) wordt een instantie van de klasse Algorithm teruggegeven als extensie van de klasse AlgorithmBase en de parameter ModelServices is het dataminingmodel. Dit object wordt onder water gebruikt voor toegang tot de data. Nu we het algoritme hebben gespecificeerd, en we in staat zijn een instantie van de klasse Algorithm terug te geven, kunnen we gaan zoeken naar patronen in de tabel.

Informatie analyseren

Om het faillissement van potentiële klanten te voorspellen, gaan we historische data analyseren met behulp van ons algoritme. In de klasse Algorithm gaan we het algoritme implementeren. SSAS zal automatisch de methode InsertCases(PushCaseSet caseSet, MiningParameterCollection trainingParameters) aanroepen om het algoritme te starten. Als parameter krijgen we de gehele dataset mee in de vorm van het PushCaseSet-object. De tweede parameter is de collectie van parameters. In de methode InsertCases lezen we de dataset in waarna we het algoritme uitvoeren. Je kunt meerdere malen door de dataset heen lopen, maar wij cachen voor het gemak de data in twee lijsten. De eerste lijst bevat de predict-waardes en de tweede lijst bevat de inputwaardes.
Het uitlezen van de data uit het PushCaseSet-object kan door het aanroepen van het statement caseSet.startCases(ICaseProcessor). De IcaseProcessor-interface heeft als doel het uitlezen van data uit het PushCaseSet-object door middel van de methode ProcessCase(long caseId, MiningCase currentCase). Onze Algorithm-klasse verrijken we met de IcaseProcessor-interface, zodat we de data in de lijsten kunnen laden. De methode ProcessCase krijgt een rij uit onze tabel 1 aangeboden in de vorm van een MiningCase-object. Om de gegevens in de lijsten te laden, gebruiken we een while-loop om door de gegevens te lopen. Zie codevoorbeeld 2 voor details.



Afbeelding 1. Grafische weergave van de gegevens uit tabel 1

```
protected override void InsertCases(PushCaseSet caseSet,
MiningParameterCollection trainingParams)
{
  //lees de voorbeelden in de arrays
  caseSet.StartCases(this);

  trainAlgorithm() ;
}

public void ProcessCase(long caseId, MiningCase currentCase)
{
  //tellertje voor bijhouden kolom
  int aCounter = 0;
  bool bContinue = currentCase.MoveFirst();
  while (bContinue)
  {
    if (isTarget(currentCase.Attribute))
      //als de kolom predict is, sla het op in de target lijst
      targets[iteration] = currentCase.IndexValue == 1 ? 1 : 0;
    else{
      //als de waarde op in de value lijst
      values[iteration][aCounter] = currentCase.Value.Double
aCounter++;
    }
    bContinue = currentCase.MoveNext();
  }
  //ga naar de volgende rij in de tabel
  iteration++;
}
```

Codevoorbeeld 2.

Nu we de data uit tabel 1 gecached hebben in de twee lijsten, richten we ons op het algoritme zelf. Het perceptron-algoritme zoekt in de dataset naar een scheiding tussen de klanten die failliet zijn en de klanten die niet failliet zijn. De gegevens uit tabel 1 zijn grafisch weergegeven in afbeelding 1. Het perceptron-algoritme berekent een lijn die failliet verklaarden scheidt van niet-failliet verklaarden. Het patroon dat het perceptron-algoritme oplevert, is de vergelijking van de lijn dat er als volgt uitziet: w . x = c. Hierbij is w een lijst van gewichten, x de variabelen en c de berekende constante. Het algoritme berekent de waardes in w en de constante c. Met behulp van deze vergelijking wordt van nieuwe klanten voorspeld of ze failliet verklaard zullen worden. Straks bespreken we hoe deze voorspellingen precies werken. Voor meer details over de oorsprong van het perceptron-algoritme, zie de links onderaan dit artikel.

Voorspellingen doen

Stel dat het perceptron-algoritme de lijn in afbeelding 1 heeft berekend. Als we een punt tekenen in het scatter-plot aan de hand van het salaris en de leeftijd, dan kunnen we bepalen of het punt aan de kant van de failliet verklaarde of aan de kant van de niet-failliet verklaarde personen ligt. Ligt het punt aan de kant van de failliete personen, dan geven we als voorspelling terug dat deze persoon in de toekomst failliet zal worden verklaard. Ligt het punt aan de andere kant van de lijn, dan voorspellen we dat deze persoon in de toekomst niet failliet zal worden verklaard. In de klasse Algorithm kunnen we voorspellingen doen door het toevoegen van de methode Predict.
In de methode Predict krijgen we alle inputwaardes binnen via het MiningCase-object, die we ook al bij de methode ProcessCase hebben gezien. Aan de hand van dit object kunnen we alle inputwaardes uitlezen en bepalen of het punt boven of onder de lijn ligt. Bij het uitvoeren van het algoritme hebben we de w-waardes en de constante c berekend. Met behulp van deze variabelen kijken we of w . x – c groter is dan 0, want dan ligt het punt boven de

lijn en anders ligt het punt onder de lijn. Op deze manier kunnen we voorspellen of een klant failliet verklaard zal worden. Na het toevoegen van de voorspellingsfunctionaliteit is ons algoritme in grote lijnen voltooid. Om het algoritme bekend te maken aan SQL Server Analysis Services, moeten we de dll eerst toevoegen aan de GAC en het algoritme enablen in Analysis Services via een XMLA-script. Aan de code die bij dit artikel zit, is een installer toegevoegd die het installatiewerk voor jou regelt.

Excel helpt je op weg

Vlak na de introductie van Office 2007 heeft het SQL Server Datamining-team een add-in beschikbaar gesteld om in Excel 2007 te kunnen dataminen. Het is daardoor niet nodig een SQL Server-database te maken met klantgegevens, omdat we deze gegevens gewoon in een Excel-sheet kunnen opslaan. Via de datamining-ribbon in Excel, komen we met een paar muisklikken bij het scherm waar we het algoritme kunnen selecteren dat tussen de meegeleverde algoritmes van Microsoft staat. Na het selecteren van het algoritme, specificeren we de mining-structuur en voeren we het geselecteerde algoritme uit. Vervolgens bekijk je via 'Accuracy Chart' in de datamining-ribbon of het algoritme zinnige voorspellingen oplevert. Daarnaast is het ook mogelijk via de 'Query'-knop jouw eigen Data Mining Expressions (DMX) uit te voeren op SQL Server Analysis Services. Excel biedt hierdoor een makkelijke toegang tot de datamining-wereld. Als je geen Office 2007 hebt, dan kun je altijd nog DMX-queries uitvoeren in SQL Server Management Studio of jouw algoritme testen in Business Intelligence Development Studio met behulp van een Analysis Services-project.

Goede voorspellingen zijn waardevol

Stel dat je echt voor een bank werkt en dat je de gegevens uit tabel 1 hebt verzameld. Met behulp van de implementatie van het perceptron-algoritme kun je dan een hoop ellende voor klanten en de bank zelf besparen. Naast het voorspellen van failliet verklaarde klanten, kun je gemakkelijk met Excel allerlei soorten voorspellingen doen. Van het voorspellen van de omzet volgend jaar tot en met het verwachte aantal bugs in jouw volgende sharepoint-project. Als je goede historische data hebt, dan kunnen jouw voorspellingen verbazingwekkend accuraat zijn. Het ontwikkelen van je eigen datamining-algoritme is met behulp van de plug-in-wrapper erg gemakkelijk gemaakt. Dit heeft als voordeel dat de focus kan liggen op het algoritme zelf en niet op allerlei randzaken. Zelf aan de slag gaan met datamining kan als een ver-van-je-bed show klinken, maar Microsoft heeft er veel aan gedaan om het zo simpel mogelijk te maken.

Joris Valkonet is business intelligence consultant bij Avanade. Hij is gespecialiseerd in datamining en is te bereiken via Joris.Valkonet@avanade.com.

Referenties

[1] www.sqlserverdatamining.com; SSAS Data Mining. Bij tutorials staat een volledige uitleg over plug-in algoritmes. De Excel-add-in is hier ook te vinden.

[2] http://www.eee.metu.edu.tr/~alatan/Courses/Demo/AppletPerceptron.html; Een Perceptron-demo.

[3] http://en.wikipedia.org/wiki/Perceptron; Perceptron volgens Wikipedia.