Софийски университет „Св. Климент Охридски“
гр. София
Факултет по математика и информатика

Курсова проект по

Извличане на знания от данни

на тема

Анализ на потребителската кошница

Изготвил
Ангел Вардин

Y

Съдържание

Мотивация

Въведение в тематиката на проблема

Анализът на потребителската кошница е доста използван набор от техники за откриване на зависимости в пазаруването на потребителите. Чрез него могат да се открият асоциативни правила или иначе казано покупката на определена група продукти е доста вероятно (или малко вероятно) да бъдат закупени продукти и от друга група. В последствие на база на събраната информация се направят планове как да се увеличат приходите на продавача като се дава промоция определени продукти, създават се пакети от стоки, които могат да се купят заедно и др. Затова този анализ е от много голямо значение както за малки фирми така и за големи вериги магазини, в които се инвестират голям капитал за проучвания в тази насока и се използват множество големи и сложни.

Цели на курсовия проект

В настоящия проект имам за цел да имплементирам малка и лесна за използване програма, която да извършва повърхностен анализ на потреблението за даден магазин или склад, която ще има следната функционалност:

- да групира продуктите на базата на тяхната цена, брой купувания и отстъпката, която е направена за тях. На базата на това групиране могат да се определят продуктите, които носят най – големи приходи и обратното. Това е постигнато чрез K-means клъстеризация
- да определи асоциативни правила между продуктите т.е. покупката на определена група продукти е доста вероятно (или малко вероятно) да бъдат закупени продукти и от друга група. Това се постига чрез алгоритъма Apriory

- да извежда получената информация от приложените алгоритми по разбираем начин за потребителят, който я използва

Реализация

Програмата е реализирана на езика C# и .net 4. За достъпа до данните се използва Entity Framework, а за графичния интерфейс – Windows Presentation Foundation.

Програмата е разделена на три модула:

- DataLayer – това е слоят, в който е имплементирана логиката за връзката с БД, която използва проекта. В него се съдържат ентитита, които съответстват на таблиците в БД и контекст, чрез който те се достъпват. Базата данни, която е използвана е на MSSQL Server 2012 и е Northwind.
- LogicLayer - това е слоят, в който е имплементирана логиката на приложението.
- PresentationLayer – това е слоят, който отговаря за потребителският интерфейс.

Тъй като има разделение на отговорностите в програмата при извършване на промени в програмата те ще засегнат по – малък брой компоненти, което позволява лесна замяна на БД.

Алгоритмична част

***K-means* алгоритъма е реализиран по следния начин:**

Първо от БД се взима информация за единичната цена на продуктите, колко пъти са купени и каква отстъпка им е направена, чрез класа **GenetatefromDb**. Тази информация се съхранява в обект от тип **ClusterPointCollection**, който съдържа лист от тип **ClusterPoint** и **ClusterPoint Centroid** което представлява центърът на клъстера. Той има методите

- **AddPoint(ClusterPoint p)** - добавяне на нова точка в клъстера
- **RemovePoint(int index), RemovePoint(ClusterPoint p)** - премахване на точки от клъстера
- **GetDistanceToCentroid()** – взимане на Евклидовото разстоянието на точките до центъра на клъстера. Използва се за оценка при извършването на Random Restarts.
- **UpdateCentroid()** променяне на координатите на центъра на клъстера.

След това се изпълнява метода **DoKMeans** в класа **KMeans** с параметри **ClusterPointCollection** и броя на клъстерите **clusterCount**. Той действа по следния начин:

първо от всички точки на случаен принцип се избират центровете на клъстерите и после се инициализират като се попълват с точки (**InitializeClusters**). След това се изпълняват два вложени цикъла по всички клъстери и по всички точки в тях . За всяка точка се намира нейния най - близък клъстер чрез метода **FindNearestCluster** . Ако този клъстер е различен от сегашния, в който се намира точката се извършва преместване, при което се актуализират координатите на центровете на двата клъстера. Това продължава докато спре да има движение на точки между клъстерите. Накрая този алгоритъм се повтаря няколко пъти като на всяко изпълнение се изчислява оценка на намереното решение, която е средно аритметично на оценките изчислени за всички клъстери (взето от резултата на метода **GetDistanceToCentroid()** на класа **ClusterPointCollection**). Като окончателно решение се взима това с най-малка оценка.

Освен чрез **DoKMeans** съм имплементирал алгоритъма и в метода **DoKMeansSecondVariant** , който също клъстеризира данните, чрез **KMeans** алгоритъма, но има различна имплементация. При него в началото също на случаен принцип се избират центровете на клъстерите и после се инициализират. След това се въвежда константата **MIN_DISPLACEMENT**, която съхранява минималната дистанция на отместване на центровете на клъстерите при актуализиране на координатите им. После се изпълнява **do-while** цикъл, чието условие за продължаване е минималната дистанция на отместване на центровете на клъстерите при актуализиране на координатите (съхранява се в променливата **displacement**) им да е по – голяма от **MIN_DISPLACEMENT.** В тялото на цикъла първо се изтриват всички от всички клъстери, после за всяка точка се намира нейния най - близък клъстер чрез метода **FindNearestCluster** и се добавя в него. След приключването на предишните операции се актуализират координатите на всички клъстери като освен това се изчислява и разстоянието на отместването. Именно сбора на разстоянията на всички отмествания е текущата стойност на променливата **displacement.** Също така и тук алгоритъма се изпълнява няколко пъти като накрая като резултат се взима най-доброто решение.

***Apriory* алгоритъма е реализиран по следния начин:**

Първо от БД се взима информация за имената на всички продукти , и транзакциите, в които участват чрез класа **GenetatefromDb**. Тази информация се съхранява в обект от тип **ItemsetCollection**, който съдържа лист от тип **Itemset** и има метод **FindSupport** , който изчислява поддръжката (Support) на колекцията от продукти. След това се изпълнява метода **DoApriori**, който има за параметри транзакциите на всички продукти (**ItemsetCollection db**) и процента на поддръжка (Support). При този метод първоначално запълва с всички продукти списъка с кандидатите (**ItemsetCollection Ci**) за първата колекция от продукти (**ItemsetCollection Li**). После се изпълнява цикъл докато **Ci** не е празен, който съдържа следните стъпки

1. Изчислява се поддръжката (Support) на всеки елемент от **Ci** и ако тя е по голяма от зададената в параметъра на функцията се добавя в **Li** и резултатния списък с продукти **L** (всички често срещани множества)
2. После **Ci** се изпразва и се попълва с новите кандидати като се взимат всички k елементи подмножества на продукти, където **k** е текущата итерация

Като резултат от метода се връща крайният списък с продукти **L**

После се изпълнява метода **Mine** с параметри транзакциите на всички продукти (**ItemsetCollection db**) крайният списък с множества от продукти (**ItemsetCollection L**) и процента на достоверност (**double confidenceThreshold**), при който за всяко множество от продукти от **L** се взимат неговите подмножества чрез метода **FindSubsets** в класа **Bit** ги се изчисляват процентите им на достоверност и ако за някое подмножество е изпълнено, че изчислената стойност е по- голяма от зададената се конструира ново асоциативно правило (тези правила са от тип **AssociationRule**) . Този алгоритъм продължава до изчерпване на всички елементи в **L.**

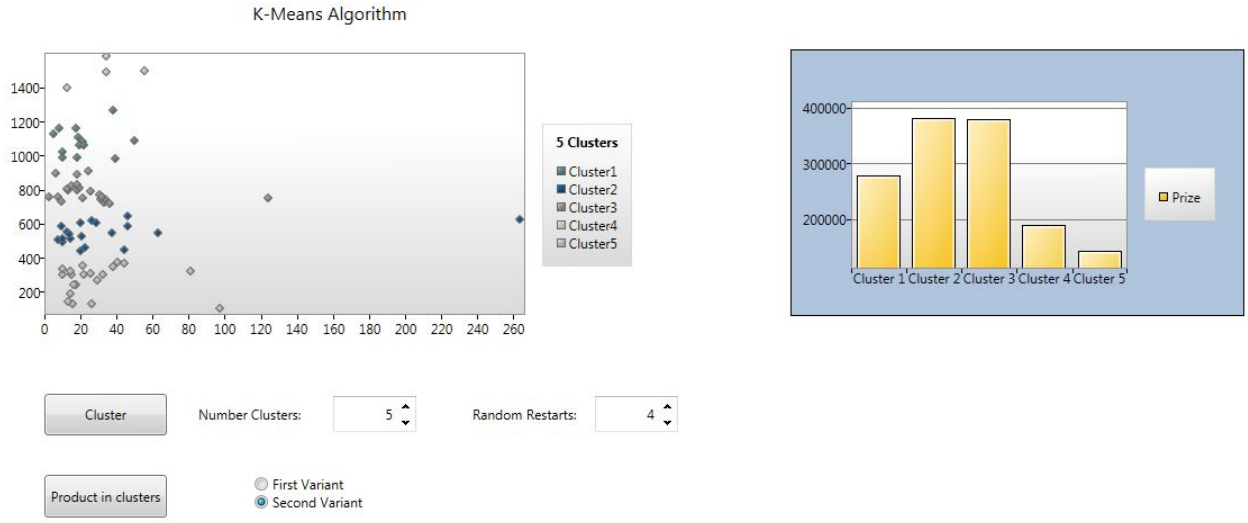
Потребителски интерфейс

Потребителският интерфейс както е обяснено по отгоре използва Windows Presentation Foundation. Той има 4 прозореца:

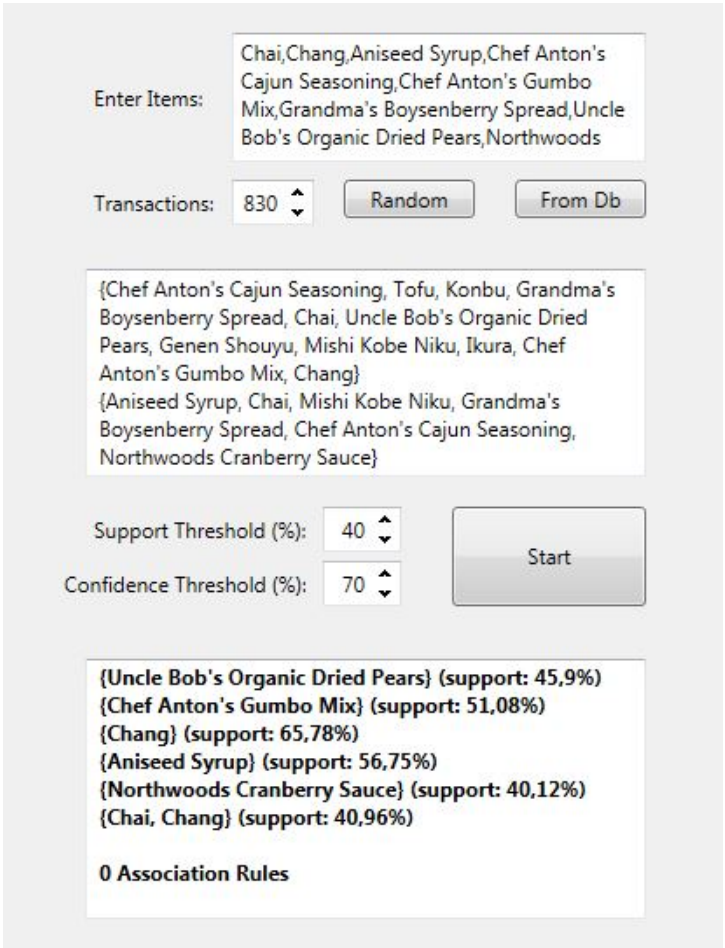
- начален прозорец, чрез който се навигира между останалите
- прозорец за клъстеризацията, на който се изобразяват резултатите от нея чрез графики за точките в отделните клъстери и за печалбата от отделните клъстери. Предвидена е възможност за смяна на броя на клъстерите и на повторните стартирания (Random Restarts)
- прозорец показващ по-детайлна информация за продуктите в клъстерите като точната печалба от тях, продуктите включени в клъстера и техните единични цени
- прозорец за алгоритъма Apriory. Той има възможност за въвеждане на процент на поддръжката и достоверност и зареждане на данните от БД

За изчертаването на графиките и числовите текстови полета са използвани библиотеките :

System.Windows.Controls.DataVisualization.Toolkit.dll – за графиките
Xceed.Wpf.Toolkit.dll – за числовите полета



фиг.1 Прозореца, показващ резултатите от клъстеризацията.



фиг.2 Прозореца, показващ резултатите от Apriory.

Резултати

След тестване на програмата се установиха следните закономерности:

- не се намериха никакви асоциативни правила и също така липсваха често срещани артикулни множества.
- при тестването на клъстеризацията се откроиха две групи от продукти от които се генерира най-голяма печалба: в първата група е на стоки, който са със сравнително ниска цена и попадат в графата често използвани от всички стоки и продукти и във втората група са на продукти и стоки с висока цена. Това най - отчетливо се вижда при клъстеризация с 5 клъстера. При увеличаване на броя клъстери втората група на скъпите продукти остава устойчива, а първата се разделя на няколко нови групи.
- разликата при клъстеризацията при двата имплементирани варианта на K-means е малка и е от порядъка на две – три точки, като тя намалява с при задаване на по-малка стойност на константата на изместването на центровете на клъстерите.

Заключение

Анализът на потребителската кошница е често използвано средство за увеличаване на печалбите на търговците, като той освен за стоки и продукти бита в магазини и хипермаркети все повече се използва и в онлайн търговията, където за потребителите може да се събере много повече информация и съответно да се извлекат по-точни закономерности в потреблението на потребителите. Например в Amazon благодарение на събраната лична информация за теб и използвайки извлечени закономерности за потребителите, който са с профил подобен на твоя профил ти предлагат артикули.

Чрез програмата, която реализирах се извършва сравнително прост анализ, чрез който се откриват асоциативни правила между продуктите и също така ги групира по печалбата, която генерират. При смяна на Базата Данни с реална и при извършване на още доработки, приложението ще може да се използва успешно в малки магазини и хипермаркети.

Използвани библиотеки

- System.Windows.Controls.DataVisualization.Toolkit.dll – за изчертаване на графиките
- Extended WPF Toolkit 2.1.0 – за визуализация на числовите полета
- Entity Framework 6.0.2 – за достъпване на Базата Данни

Използвани източници

- https://wpftoolkit.codeplex.com - официалната страница на Extended WPF Toolkit
- http://www.c-sharpcorner.com/uploadfile/mahesh/charting-in-wpf/ - информация за създаване на графики в WPF и библиотеката System.Windows.Controls.DataVisualization.Toolkit.dll
- http://www.cs.sunysb.edu/~cse634/lecture_notes/07apriori.pdf/ информация за Apriory алгоритъма
- http://www.cs.uoi.gr/~arly/papers/PR2003.pdf/ - информация за K-means алгоритъма
- http://en.wikipedia.org/wiki/K-means_clustering/ - сайта на Wikipedia

Декларация за липса на плагиатство

Декларирам, че:

- Тази курсова работа е моя работа, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
- Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.

Ангел Кънчев Вардин

Подпис: