International Journal of Data Mining Techniques and Applications
Volume 5, Issue 1, June 2016, Pages: 15-18
ISSN: 2278-2419

Sales Management Using Apriori Algorithm On SAP Fiori

Ketki Kulkarni, Ashwini Ghuge, Priyanka Waghmare, and Aishwarya Mali
Students of Computer Science, Pune Institute of computer technology, PICT, Pune, India
Email: aishwaryamali94@gmail.com

Abstract - Sales management is an important aspect of all Business analytics outfits. However, in today's digital world we need to find on-the-go solutions. We can find these by targeting the mobile Apps. We find that there are numerous mobile solutions offered by android Apps in the market. They are so popular because of compatibility on various mobile platforms. But a major problem faced by many outfits today is their compatibility with backend systems in use, namely the ERPs like SAP. In this paper we aim to solve this problem. We also portray a successful use of the upcoming SAP Fiori technology which works on multiple mobile platforms. Another major issue is business analytics on such mobile platforms. Using SAP ERP and the latest HANA databases, we can perform robust data mining algorithms on vast amounts of data. Infact, some data- mining essential algorithms are already a part of such systems in the form of libraries. However, these databases are expensive and are unsuitable for small outfits using SAP services. Today, there exists virtually no library for data-mining on mobile devices without a Hana database. In our experiments we aim to solve this issue. We have created a market basket-analysis system to be deployed as a mobile App on SAP Fiori from scratch without a Hana database. We implemented an Apriori algorithm, to generate rules. We found that such a system can be developed with said requirements and has very low operational delay

Keywords-Apriori algorithm, SAP Fiori, Sales management, ERP, Business analytics

I. Introduction

Sales management is an important aspect of all Business analytics outfits. Sales management is more than just tracking the business you book and providing support for your sales team. It starts with develop the required products, setting the appropriate prices and distributing in appropriate places, and continues with marketing messaging, customer service and other selling efforts. Sales reports not only provide you with information about what's selling and how much you're selling, but where you are making your sales.A sales management program evaluates your distribution methods and maximizes their use. For example, if the sales of a particular product are strong but that of another are lagging, we might find that a particular product has more relevant information on display, helping customers to buy it with confidence. To improve the sales of the other product, we may provide more in-store promotions and change our product's packaging. Some of our best-selling products, in terms of amount sold, might provide us lowest profit scopes, causing a burden on our production and administration departments. Detailed sales reports provide us with information on our inventory and production costs, cost-of-sales expenses and profit scopes. A low-profit item with high sales margin might provide a nice profit scope, making it a crucial item to keep in your catalogue. If we eliminate this item, causing a corresponding increase in sales of products with higher-sales margin, then it might be better to discontinue selling it. Sales management looks at the profit contribution, opportunity cost and impact of carrying each product on your operations. All of these efforts must be coordinated so that one doesn't interfere with the other. Setting goals, monitoring them and tracking results allow an outfit to adapt, eliminate weaknesses and take advantage of opportunities. All these can be obtained from a traditional ERP system. However, in today's digital world we need to find on-the-go solutions. We need faster insight extraction.

All managers agree on the importance of such data analysis, performed on data obtained from reporting tools. More often than not, in-house data analysis proves to be cost productive. This is further supported by the fact that internal resources have a better understanding of the business. Indeed, internal resources have the needed capability to perform data analysis in a way that would effectively serve the needs of managers in terms of data sources. It would be logical to assign data analysis to someone who is in daily contact with the business with less investment, than spending much more money on external help that would not necessarily provide managers with a fully personalized result. More importantly, companies are not willing to share data with external parties as they consider it to be a confidential key element in the business, especially when an investment has been put in place to gather the data, and they prefer to keep it as their own. We find that both the issues of optimized/faster insight extraction and keeping the secrecy of company data can be solved by targeting the mobile Apps. There are many mobile devices and platforms available today. The list is constantly growing and so is the platform support. There are hundreds of models available today, with multiple hardware and software combinations. The enterprise must select a device very carefully. But a major problem faced by many outfits today is their compatibility with backend systems in use, namely the ERPs like SAP. Many of these native Apps cannot be used on data stored at the backend of SAP systems. In an office, some business analysts or market strategist might need access to such valid corporate data but find that they can access it only from the assigned terminals. To provide data-mining solutions for all such personnel, we need to generate custom-made Apps in android or iOS. There is a conflict here in terms of security because; there exists no central hub which manages data security. Employees can then transfer data to one-another from the respective mobile devices with such Apps with impunity. The effects of such practices could be disastrous.

An easy solution is SAP Fiori. SAP Fiori provides users access only to those essential utilities of SAP ERP system from Mobile devices. It has no barrier of target devices and can be accessed effortlessly via tablets, Smartphones and desktops. They have numerous in-built Applications for managers, employees, purchasing agents and Sales representatives. These Apps are deployed to help the users who are frequently away from their desks. It provides a streamlined and homogenous experience across various devices. Fiori's responsive design is a significant overhaul in terms of efficiency. One no longer has to

wait to get in front of the computer for submitting a leave request or to track sales orders. All one needs is a device, and any device can get your job done.

From SAP Fiori one can track sales orders, generate on-the-go invoices and check availability of products. They also have superb report publishing utilities. Reports can be published using the rich data collected via these devices. We may integrate our new data with existing data via front end .We can then Apply data-mining solutions on selected data alone. It combines the advantages of a native App with that of a web App, for SAP Fiori uses HTML5. It is truly a hybrid mobile App. By using SAP Fiori we can maintain security keys for only those employees that have been cleared for access. The term User Experience or UX is used so routinely these days that sometimes we tend to undermine the presence of the U in UX. It stands for the (end) user of the system/Application and addressing the UX is about improving the experience of that system for that user. Fiori does that to the tee. Classical SAP was very transaction-oriented but Fiori focuses on usability and objective-oriented which is a huge improvement over its counterpart. Fiori has been designed such that, it targets the end user. Fiori understands that the end user may or may not be an IT professional and makes the using these Apps very easy.

Using SAP ERP and the latest HANA databases, we can perform robust data mining algorithms on vast amounts of data. Infact, some data- mining essential algorithms are already a part of such systems in the form of libraries. However, these databases are expensive and are unsuitable for small outfits using SAP services. Today, there exists virtually no library for data-mining on mobile devices, without a Hana database. In our experiments we aim to solve this issue. We have created a basket-analysis system to be deployed as a mobile App on SAP Fiori from scratch without using a Hana database. In our experiment we used an ECC database. It is a relational database. We implemented an Apriori algorithm, to generate rules. We found that such a system can be developed with said requirements and has very low operational delay

II. Methods and materials

A. Study Materials

SAP Fiori allows employees to work seamlessly across devices – desktop, tablet, or Smartphones. It provides improved user satisfaction and enables quicker Approvals and better decisions. Fiori is collection of Apps created and designed by SAP. They are available under SAP Fiori ERP for mobile devices. Front-end technology used in these Apps is SAPUI5, which includes HTML5, JavaScript and CSS where any kind of data is accessed deleted and updated from backend using Odata services through SAP Netweaver Gateway. Fiori Applications can run on mobile devices with browsers, there is a need for reverse proxies to access data from backend. Fiori Applications are example of responsive design; they can gauge the screen sizes of devices and render to them automatically. These Applications are light-weight Applications, and are not for heavy transactional Applications. However, their Apps can be extended. Fiori is built upon SAPUI5 and specifically, the sap.m library. Fiori is part operations-oriented, part responsive design. It looks and feels, like a sophisticated application and that's one of the reasons we used it to deliver our data-mining solution. Refer to this site: http://scn.sap.com/community/developer-center/front-end/blog/2014/03/04/building-sap-Fiori-like-uis-with-sapui5-in-10-exercises where one can learn about data-binding, localization, resource models, XML-based views, formatter functions, best practices for Application design & build and controls such as the responsive SplitApp, ObjectHeader & ObjectListItem, IconBar, SearchField and others. The most wonderful and appealing Apps incorporate incisive front-end and backend logic. Moreover, a good App keeps these two ends separate. SAP has realized the user interface (UI) is more susceptible to change always as ever before. User's evolving requirements and demands can easily change an Application's structure and design. With this in mind, the backend is developed and maintained separately from the frontend, and there should be a clear divide of the two. Apart from this divide, the individual UI elements of an App can and are at times developed separately, however this is Application-specific. In a nutshell, UI5 is a client UI technology based on JavaScript, CSS and HTML5. Servers come into play for deploying your Applications, storing the SAPUI5 libraries to increase reusability, and connecting to a database whichever kind it might be. Depending on the environment in which you are using SAPUI5, the libraries and your Applications are stored for instance on some kind of a Cloud Platform or another Application server. Many a times applications use a SAP HANA CLOUD. The favored means to access business data for your Application is by using the Odata model.

SAPUI5 supports the Model View Controller (MVC) concept, "a software architectural pattern for developing user interfaces". As an ABAPer, one is encouraged to use the MVC to keep the data model working, the User Interface design and the Application logic separate. This helps in enabling robust UI development in addition to giving a free reign when modifying different parts.

Model: This is the part that is responsible for the managing, retrieving, and updating of the data that is being viewed in your Application.

View: This part is responsible for understanding and representing the initial UI. The view in the context of SAPUI5 generates the presentation to the user based on changes in what is requested by the user. Views are in stored in the "view" folder and names of XML views always end with *.view.xml (as seen in the fig 1.0).

Controller: This is one of the most significant parts. This is the part that is responsible for separating the view logic from the data logic. The Controller responds to user interaction and "view events" by adjusting the view and the model. The controller is really just sending commands to the model to update its state, like editing a document in a word pad or other similar text-editing application. Similar to views, Controllers carry the same name as the related view (if there is a 1:1 relationship). Controller names always end with *controller.js (as seen in the fig 1.0).

International Journal of Data Mining Techniques and Applications
Volume 5, Issue 1, June 2016, Pages: 15-18
ISSN: 2278-2419

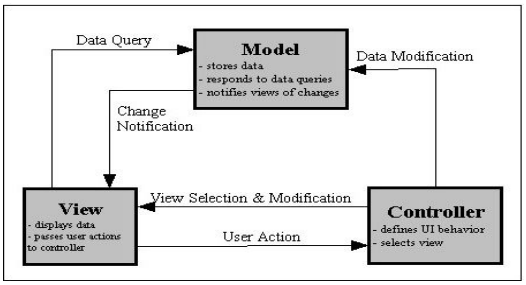

Fig 1.0

The first step in loading a SAPUI5 App is when the browser loads the web Application's index.html.

What follows it, is the then loading of the SAPUI5 toolkit core using the standard variant bootstrap. Bootstrap contains HTML and CSS-based libraries or patterns for SAPUI5's typography, forms, buttons, navigation, and other interface components. Then there is a single function calls to demonstrate a **sap.ui.core.ComponentContainer** and places it in the body of the HTML document. The sap.ui.core.ComponentContainer loads a **sap.ui.core.UIComponent** which is the independent bundle of the whole Application. The Component (which is named **sap.ui.core.UIComponent**) is defined in the **Component.js** file in the same folder as **index.html**; Our Component consists of two parts: metadata that is data about data, and a function that calls our required data when the component is initialized after the index.html file.

Following the initialization of our Component, there are a few models that are created and regulated on top of this Component. The first is the main model; this connects to our Data Source using Odata. Along with the main model, the internalization (i18) is initialized. The internalization file is where texts are loaded from a local resource (file), and device detection based on UI controls.The visible part of our imaginary App is provided by three main XML views and a fragment. When our App is presented on a device (other than a smartphone), there are two views shown - the master and the detail. At first, this is the **Master.view.xml** and the **Detail.view.xml**. Each of these views contains a sap.m.Page. There's also **NotFound.view.xml**, to be used when no view can be matched to the resource requested. Similar to that very well known"404-Not Found" page you may have come across on other websites. The sap.m.Page is nothing but a basic container for a mobile Application screen. Thus, we have used sap.m library and HTML5 toolkit of SAP Fiori to implement the front end of the mobile Application.We use Odata to connect the backend SAP Fiori to front end mobile App. We pass and initialize the data models in the front-end. We use functions called RFCs (Remote Function Call) to aggregate data and pass it to the Odata models we create at backend. These models are then passed to the front-end by simply giving a path. Odata is frequently called ODBC for internet.The main Apriori algorithm is written in ABAP programming. When making RFC we write the methods to aggregate the data and perform operations on them.

B. Apriori Algorithm

Apriori is a data –mining algorithm for frequent item set extracting and association rule learning over transactional databases. It continues by identifying the recurrent distinct items in the database and extending them to bigger and bigger item sets as long as those item sets appear satisfactorily often in the database. The recurrent item sets determined by Apriori can be used to determine association rules which highlight broad tendencies in the database. This has Applications in domains such as market basket analysis.

Association rule mining is a two-step Approach:

1. Frequent Itemset Generation – This steps generates all itemsets whose support ≥ minimum support
2. Rule Generation –This step generates high confidence rules from each recurrent itemset, where each rule is a partitioning of a recurrent itemset.

Find the frequent or recurrent itemsets: the sets of items that have minimum support.

One rule follows which is that a subset of a recurrent itemset must also be a recurrent itemset. If {XY} is a frequent itemset, both {X} and {Y} should be a frequent itemset.

We iteratively find frequent itemsets with cardinality from 1 to n (n-itemset). Finally we use the frequent itemsets to generate association rules.

Join Step: E_n is generated by joining $R_{n-1}$with itself.

Prune Step: Any (n-1)-itemset that is not frequent cannot be a subset of a frequent n-itemset

Pseudo-code:

E_n: Entrant itemset of size n.
R_n : Recurrent itemset of size n
R_1 = {frequent items};
for (n = 1; R_n != ∅; n++) do begin
E_{n+1} = entrants generated from R_n;
for each operation T in database do
increment the count of all entrants in E_{n+1}

that are contained in T
R_{n+1} = entrants in E_{n+1} with minimum_support
end
return ∪ n R_n;

III. Results And Discussion

C. Result

We use Odata to connect the backend SAP Fiori to front end mobile App. We pass and initialize the data models in the Front-end. There are two models we use, i18n and Odata model. We use functions called RFCs to aggregate data we want to process and pass it to the Odata models we create at backend. These models are then passed to the front-end by simply giving a path in the main program at front-end. Odata is frequently called ODBC for internet. The main Apriori algorithm is written in ABAP programming.

International Journal of Data Mining Techniques and Applications
Volume 5, Issue 1, June 2016, Pages: 15-18
ISSN: 2278-2419

When making RFC we write the methods to aggregate the data and perform operations on them.

The results of our experiment look like this,

Example of Rules:

{Laptop,Antivirus} → {Pendrive} (s=0.4, c=0.67)

{Antivirus,Earphones} → {Laptop} (s=0.4, c=1.0)

{Laptop,Earphones} → {Antivirus} (s=0.4, c=0.67)

{Pendrive} → {Laptop,Antivirus} (s=0.4, c=0.67)

{Laptop} → {Antivirus,Pendrive} (s=0.4, c=0.5)

{Antivirus} → {Laptop,Pendrive} (s=0.4, c=0.5)

All the above rules are binary partitions of the same itemset: {Laptop, Antivirus, Pendrive}

Rules originating from the same itemset have identical support but can have different confidence.

Thus, we may decouple the support and confidence requirements.

We can generate similar rules by taking into account brands, colors, sizes of electronic products. We have done just that in our project. We believe that Sales Management is more than just placing orders and counting invoices. We have used the data generated from these invoices and applied the Apriori Algorithm on just thoses orders that have already been completed . In doing this we ensure that our App does not face a long operational delay.

D. Discussion

We have made an App which gives us complete Sales Reports for an organization. We have provided Sales Reports that not only provide you with information about what's selling and how much you're selling, but where you are making your sales. Thus, we have Applied data-mining algorithms in inventory management.Furthermore, we have solved the issue of extracting faster insights out of available data by building a mobile App in SAP. This has helped us co-ordinate our data with the backend ERP system already in place. We have also saved cost and kept our Sales data private by building an App in-house.Finally, we have solved the problem of using an ECC database instead of an expensive HANA database by building our data-mining algorithm from scratch. HANA database is expensive and not available in India as of now. Companies today have to rely on external software to run such optimization algorithms like data-mining. Many smaller startups looking to optimize their operations face trouble in doing so. We have demonstrated that data-mining algorithms can indeed be built on databases other than HANA. We have built a market basket-analysis system for moving inventory.

IV. Conclusion

We have thus implemented a data-mining solution on a mobile SAP platform in an efficient and cost productive way. We have built the data- mining algorithm namely, Apiori from scratch using ABAP programming and Odata concept. We have not used any in-built functions available with expensive HANA databases but have used a simple relational database instead. The demonstrated result shows that Apriori Algorithm works successfully on mobile Apps with proportionately sized datasets.

Tid	Items
1	Earphones ,Antivirus
2	Earphones, Laptop, Pendrive, Mouse
3	Antivirus,Laptop,Pendrive,ipad
4	Earphones,Laptop,Antivirus,Pen-drive
5	Earphones,Antivirus,Laptop,ipad

V. Acknowledgment

We thank Mindshare Business Consulting,Pune for their continued support and dedication. They have provided us with hands-on knowledge in SAP Fiori as well as with the server and database resources. A special thanks to Mr. Mahesh Sagaonkar for believing in us and helping us achieve our said analytics goals.

REFERENCES

[1] Andris A. Zoltners, Prabhakant Sinha, Sally E. Lorimer, "Sales Force Design for Strategic Advantage."

[2] Chris Lytle, "The Accidental Sales Manager."

[3] Xie, Anne, Huang, Xiong , "Advances in Electrical Engineering and Automation." Volume 139 of the series Advances in Intelligent and Soft Computing pp 75-80

[4] http://scn.SAP.com/docs/DOC-41598, All things SAP Fiori

[5] https://SAPui5.hana.ondemand.com/sdk, Official SAP documentation.