Association Rule Analysis

Abstract

The aim of this project is to perform 'market basket analysis' on groceries' transaction data and output the association rules found in it. 'Association rule mining' is the data mining task that analyses the transactions given in the data and finds the set of items that are likely to be bought based on the items that appear together in the transactions. Knime 3.2.1 tool has been chosen for this task.

Introduction

The correlations and associations in large sets of data can be found through frequent items mining. One such mining is 'Market Basket Analysis', which analyses the transactions made by the customer and finds the relationships between the different items that the customers select. These relationships are depicted in the form of association rules and their likelihood of occurrence, which tell us about what items are likely to be bought together. This helps in many business decision making processes like cross marketing and customer shopping behaviour analysis.

Market Basket Analysis: This is a process which analyses the habits of the buyer to find a relationship between different items on their shopping cart (market basket). The discovery of these relationships can help the seller to develop a sales strategy to consider items frequently purchased together by customers.

Association rules: Association rule is a procedure which is looking for a relationship among an item with other items. Association rule is usually used "if" and "then" such as "if A then B and C", this shows if A then B and C. To determine the Association's rules, it needs to be specified the support and confidence to restrict whether the rule is interesting or not.

Problem Statement

Find association rules in groceries data by using Apriori and FPGrowth algorithms and test the strength of those rules by using the measures, Support, Confidence, Interestingness, Comprehensibility, Lift, Leverage, and Conviction.

Dataset Description

The groceries data set consists of 1000 transactions and 11 columns each indicating one grocery item. Each column is of type boolean, i.e. has the value 0 or 1. Hence each transaction is a combination of 1's and 0's where 1 indicates the presence of that item in that particular transaction and 0 indicates the absence.

Overview of Algorithms

APRIORI

Apriori is a seminal algorithm that uses prior knowledge of frequent itemset properties. Apriori employs an iterative approach known as level-wise search where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item and collecting those items that satisfy minimum support. The resulting set is denoted by L1. Next, L1 is used to find L2, the set of frequent of 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of database. This algorithm uses Apriori property which says that 'all non-empty subsets of a frequent items must also be frequent'.

When written in the form of pseudo code, Apriori algorithm is as follows

L_1 := { large 1-itemsets };
$k := 2$; // k represents the pass number
while $(L_{k-1} \neq \emptyset)$ **do**
begin
    C_k := New candidates of size k generated from L_{k-1}; (*apriori_gen*)
    forall transactions $t \in \mathcal{D}$ **do**
        Increment the count of all candidates in C_k that are contained in t;
    L_k := All candidates in C_k with minimum support;
    $k := k + 1$;
end
Answer := $\bigcup_k L_k$;

While the pseudo code of the formation of joint candidate item set is given below

(1) Join Step
insert into candidate k-itemset
select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}$
from large $(k-1)$-itemset p, large $(k-1)$-itemset q
where $p.\text{item}_1 = q.\text{item}_1, \ldots, p.\text{item}_{k-2} = q.\text{item}_{k-2}, p.\text{item}_{k-1} < q.\text{item}_{k-1}$;

(2) Prune Step
forall itemsets $c \in$ candidate k-itemset **do**
    forall $(k-1)$-subsets s of c **do**
        if $(s \notin \text{large}(k-1)-\text{itemset})$ **then**
            delete c from candidate k-itemset;

FPGrowth

FPGrowth algorithm adopts a divide-and-conquer strategy. First, it compresses the database representing frequent items into a FP-tree, which retains the items association information. It then divides the compressed database into a set of conditional databases, each associated with one "pattern fragment" and mines each database separately. For each of these, only its associated data sets need to be examined.

Simply a two-step procedure

A. Build a compact data structure called the FP-Tree

Pass 1:

Scan data and find support for each item.

- Discard infrequent items.
- Sort frequent items in decreasing order based on their support.

Use this order when building the FP-Tree, so common prefixes can be shared.

Pass 2:

- FP-Growth reads 1 transaction at a time and maps it to a path.
- Fixed order is used, so paths can overlap when transactions share items (when they have the same prefix).

Increment counter

- Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines). The more paths that overlap, the higher the compression. FP-Tree may fit in memory.
- frequent item sets extracted from the FP-Tree

B. Extracts frequent item sets directly from the FP-Tree

- FP-Growth extracts frequent item sets from the FP-Tree
- Bottom-up algorithm — from the leaves towards the root
- Divide and conquer: first look for frequent item sets ending in e, then de, etc…. then d, then cd, etc ……
- First, extract prefix path sub-trees ending in an item (set).
- Each prefix path sub-tree is processed recursively to extract the frequent item sets. Solutions are then merged.

Measures

1. Support

For the rule x => Y, the support value of X with respect to set of transactions T is defined as the proportion of transactions in the database which contains the item-set X.

$$\text{Support} = \frac{\text{number of transactions that contain antecedents}}{\text{total number of transactions}}$$

2. Confidence

The confidence value of a rule, X⇒Y , with respect to a set of transactions T, is the proportion the transactions that contains X which also contains Y.

$$\text{Confidence} = \frac{support(X\ U\ Y)}{support(X)}$$

3. Conviction

Conviction compares the probability that X appears without Y if they were dependent with the actual frequency of the appearance of X without Y

$$\text{Conviction}(X \rightarrow Y) = \frac{1 - support(Y)}{1 - confidence(X \rightarrow Y)}$$

4. Lift

Lift measures how many times more often X and Y occur together than expected if they where statistically independent.Lift is not down-ward closed and does not suffer from the rare item problem. Also lift is susceptible to noise in small databases. Rare itemsets with low counts (low probability) which per chance occur a few times (or only once) together can produce enormous lift values.

$$\text{Lift}(X \rightarrow Y) = \frac{support(X \rightarrow Y)}{support(X) * support(Y)}$$

5. Leverage

Leverage measures the difference of X and Y appearing together in the data set and what would be expected if X and Y where statistically dependent. The rational in a sales setting is to find out how many more units (items X and Y together) are sold than expected from the independent sells.

$$= support(X \rightarrow Y) - support(X) * support(Y)$$

6. Comprehensibility

The Comprehensibility measure is needed to make the discovered rules easy to understand. The comprehensibility tries to quantify the understand ability of the rule. Here |Y| and |X∪Y| are the number of attributes involved in the consequent body and the total rule respectively.

$$\text{Comprehensibility} = \frac{\log(1 + |Y|)}{\log(1 + |X\ U\ Y|)}$$

7. Interestingness

Identifies rare rules, even though they have less individual support count adds interestingness. Where D is total number of records.

$$\text{Interestingness} = \frac{support(X\ U\ Y)}{support(X)} * \frac{support(X\ U\ Y)}{support(Y)} * (1 - \frac{support(X\ U\ Y)}{D})$$

Results and Discussion

Results —

The following is the output obtained by using Apriori algorithm.

Best rules found:

1. [Cannedveg=1, Beer=1]: 167 ==> [Frozenmeat=1]: 146 <conf:(0.87)> lift:(2.89) lev:(0.1) conv:(5.3)
2. [Frozenmeat=1, Beer=1]: 170 ==> [Cannedveg=1]: 146 <conf:(0.86)> lift:(2.83) lev:(0.09) conv:(4.74)
3. [Cannedveg=1, Frozenmeat=1]: 173 ==> [Beer=1]: 146 <conf:(0.84)> lift:(2.88) lev:(0.1) conv:(4.37)
4. [Beer=1]: 293 ==> [Frozenmeat=1]: 170 <conf:(0.58)> lift:(1.92) lev:(0.08) conv:(1.65)
5. [Frozenmeat=1]: 302 ==> [Cannedveg=1]: 173 <conf:(0.57)> lift:(1.89) lev:(0.08) conv:(1.62)
6. [Cannedveg=1]: 303 ==> [Frozenmeat=1]: 173 <conf:(0.57)> lift:(1.89) lev:(0.08) conv:(1.61)
7. [Beer=1]: 293 ==> [Cannedveg=1]: 167 <conf:(0.57)> lift:(1.88) lev:(0.08) conv:(1.61)
8. [Frozenmeat=1]: 302 ==> [Beer=1]: 170 <conf:(0.56)> lift:(1.92) lev:(0.08) conv:(1.61)
9. [Cannedveg=1]: 303 ==> [Beer=1]: 167 <conf:(0.55)> lift:(1.88) lev:(0.08) conv:(1.56)
10. [Confectionery=1]: 276 ==> [Wine=1]: 144 <conf:(0.52)> lift:(1.82) lev:(0.06) conv:(1.48)

The following is the output obtained by using FPgrowth algorithm.

Best rules found:

1. Cannedveg=1 Beer=1 167 ==> Frozenmeat=1 146 <conf:(0.87)> lift:(2.89) lev:(0.1) [95] conv:(5.3)
2. Frozenmeat=1 Beer=1 170 ==> Cannedveg=1 146 <conf:(0.86)> lift:(2.83) lev:(0.09) [94] conv:(4.74)
3. Cannedveg=1 Frozenmeat=1 173 ==> Beer=1 146 <conf:(0.84)> lift:(2.88) lev:(0.1) [95] conv:(4.37)
4. Beer=1 293 ==> Frozenmeat=1 170 <conf:(0.58)> lift:(1.92) lev:(0.08) [81] conv:(1.65)
5. Frozenmeat=1 302 ==> Cannedveg=1 173 <conf:(0.57)> lift:(1.89) lev:(0.08) [81] conv:(1.62)
6. Cannedveg=1 303 ==> Frozenmeat=1 173 <conf:(0.57)> lift:(1.89) lev:(0.08) [81] conv:(1.61)
7. Beer=1 293 ==> Cannedveg=1 167 <conf:(0.57)> lift:(1.88) lev:(0.08) [78] conv:(1.61)
8. Frozenmeat=1 302 ==> Beer=1 170 <conf:(0.56)> lift:(1.92) lev:(0.08) [81] conv:(1.61)
9. Cannedveg=1 303 ==> Beer=1 167 <conf:(0.55)> lift:(1.88) lev:(0.08) [78] conv:(1.56)
10. Confectionery=1 276 ==> Wine=1 144 <conf:(0.52)> lift:(1.82) lev:(0.06) [64] conv:(1.48)

The rule strength measures of top 10 rules —

no	confidence	lift	leverage	conviction	support	interestingness	comprehensibility
1	0.87	2.89	0.1	5.3	0.146	0.422	1.144
2	0.86	2.83	0.09	4.74	0.146	0.413	1.145
3	0.84	2.88	0.1	4.37	0.146	0.420	1.140
4	0.58	1.92	0.08	1.65	0.17	0.326	1.111
5	0.57	1.89	0.08	1.62	0.173	0.327	1.108
6	0.57	1.89	0.08	1.61	0.173	0.327	1.107
7	0.57	1.88	0.08	1.61	0.167	0.314	1.115
8	0.56	1.92	0.08	1.61	0.17	0.326	1.105
9	0.55	1.88	0.08	1.56	0.167	0.314	1.109
10	0.52	1.82	0.06	1.48	0.144	0.261	1.137

The support of the individual items is as follows -

frozenmeat	0.302
cannedveg	0.303
beer	0.293
wine	0.287

Comparision of models -

1. The top 10 rules produced by both algorithms are same but apriori identifies lot more rules when compared to FPGrowth, 16 and 48 repectively. Apriori is capable of mining some less frequent and obvious rules too.
2. The time taken by Apriori is lot more compared to FPgrowth becuase Apriori scans the databse many times. Hence Apriori needs more space.

Interpretation of Results:

1. Among the rules mentioned above, the rule {canned veg, beer} ==> {frozenmeat} has the highest confidence. This means that the probability of frozen meat being bought when {cannedveg , beer} are bought is 0.87. But according to rule {frozenmeat} ==> {cannedveg, beer}, cannedveg and beer have the lowest probability of being bought when frozen meat is bought.
2. Having high lift means having positive effect on buying the consequent of the rule by the antecedent of the rule. This means buying cannedveg and beer are positively correlated to buying frozen meat.
3. From *Cannedveg=1 303 ==> Confectionery=1 71 <conf:(0.23)> lift:(0.85) lev:(-0.01) [-12] conv:(0.94)* we can say that connedveg has no effect on buying confectionery because the lift is nearly zero.
4. Support gives the probability of both the the antecedent and consequent occurring together. This means that cannedveg and frozen meat have the highest probability of being bought together, confectionery and wine are not very frequently bought together and, cannedveg and confectionery are almost never bought together.
5. Cannedveg, beer and frozen meat have high leverage, hence they will be bought together lot more than bought individually, even though they are dependent. Whereas, cannedveg and confectionery will seldom be bought together but may be bought individually.
6. Interestingness identifies rare rules. The first rule has the highest interestingness and it decreases gradually with rest of the rules.

Conclusions :

1. The items cannedveg, frozen meat and beer should be placed next to each other.
2. These items should be packaged with any other poorly seeing item to increase its sale.
3. Give discount on only one of cannedveg, beer or frozen meat.
4. Advertising only cannedveg and beer is enough. Advertising frozenmeat is not necessary.

References:

1.http://analyticstrainings.com

2.http://www.cs.ccsu.edu

3.https://tech.knime.org/forum/general/association-rule-mining

4.http://www.kdnuggets.com

5. https://www.r-bloggers.com