CHAPTER 6
ANALYSIS OF CUSTOMER BEHAVIOR USING CLUSTERING AND ASSOCIATION RULES

6.1 INTRODUCTION

Customer Relationship Management is used to make more efficient business-customer relationships in order to maximize client satisfaction and thereby improve customer loyalty and retention, Jawed Siddiqi et al. (2002). Data mining techniques are useful to analyze the customer behavior. Data Mining grew from the persistent growth of techniques used to interrogation masses of data, Dawn et al. (2012). Prediction is done using the relevant data set taken from the database on the basis of the attributes. Customer data and customer relationships should be stored and maintained. Customer loyalty and retention are improved when customer satisfaction is more.

6.2 CLUSTERING ANALYSIS

Predictive analytics analyzes historical data to make good predictions. Such predictions rarely take the form of absolute statements, and are expressed as values that correspond to the behavior taking place in the future, Jaideep Srivastava et al. (2004). The customers with similar purchasing behavior are first grouped by means of clustering techniques. Finally, for each cluster, association rules are used to identify the products that are frequently bought by the customers. Clustering analysis is a data mining technique that maps data objects into unknown groups of objects with high similarity. Clustering is the task of segmenting a heterogeneous population into a number of more homogenous clusters, Ngai et al. (2009). Clustering algorithms are categorized as hierarchical or partitional clustering.

6.2.1 K-MEANS CLUSTERING

The k-means clustering is used to cluster observations into groups of related observations without any prior knowledge of those relationships. This algorithm aims to assign a set of \(n\) data objects to \(k\) clusters in order to achieve a high intra cluster similarity and a low inter cluster similarity. The figure 6.1 shows the clusters obtained by k-means method.
Fig 6.1: Clusters obtained by k-means Method

The algorithm clusters observations into $k$ groups, where $k$ is provided as an input parameter. It then assigns each observation to clusters based upon the observation’s proximity to the mean of the cluster. The cluster’s mean is then recomputed and the process begins again. The algorithm arbitrarily selects $k$ points as the initial cluster centers. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$
where $\|x^{(j)}_i - c_j\|^2$ is a chosen distance measure between a data point and the cluster centre, is an indicator of the distance of the n data points from their respective cluster centres. The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These are represented as initial group centroids.
2. Assign each object to the group which has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat the steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

We must define the initial seeds in the first iteration of the algorithm. We have to run $k$-means with different seeds and to choose the ones that have produced the lowest value of a criterion function, usually the square error. Concerning the definition of the number of clusters, several heuristics are available, as it is not possible to theoretically determine the optimal value of the number of clusters.

### 6.2.2 K-MEDOIDS CLUSTERING

The k-medoids algorithm is a clustering algorithm related to the k-means algorithm and the medoidshift algorithm. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses data points as centers (medoids or exemplars).

The k-medoid is a classical partitioning technique of clustering that clusters the data set of n objects into k clusters known a priori. It is more robust to noise and outliers as compared to k-means because it minimizes a sum of dissimilarities instead of a sum of squared Euclidean distances. A medoid can be defined as the object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal i.e. it is a most centrally located point in the given data set. The figure 6.2 shows the clusters obtained by k-medoids method. The most common realization of k-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follows:
1. Initialize: randomly select k of the n data points as the medoids
2. Associate each data point to the closest medoid. ("closest" here is defined using any valid distance metric, most commonly Euclidean distance, Manhattan distance or Minkowski distance)
3. For each medoid m
   1. For each non-medoid data point o
   2. Swap m and o and compute the total cost of the configuration
4. Select the configuration with the lowest cost.
5. Repeat steps 2 to 5 until there is no change in the medoid.

Fig 6.2: Clustering by k-medoids Method

6.3 ASSOCIATION RULES

Association mining, which is widely used for finding association rules in single and multidimensional databases, can be classified into intra and inter transaction association mining. Intra-transaction association refers to association in the same transaction; inter-transaction association indicates association among different transactions, Lu et al. (2000).
Association rules are useful for the analysis of customer data, Beomsoo Shim et al. (2012). An association rule can be represented in the form \( X \Rightarrow Y \), indicating that when product \( X \) is purchased, product \( Y \) is also purchased. \( X \) is known as the antecedent and \( Y \) is the consequent, such that \( X \) triggers the purchase of \( Y \).

### 6.3.1 A PRIORI ALGORITHM

The algorithms which can be used for market basket analysis are for a generic problem of association: the Apriori algorithm and Frequent-pattern growth algorithm, Han et al. (2000). These algorithms involve two stages. The first stage concerns the discovery of the products, which more frequently purchased. The Apriori algorithm is the most well known association rule mining algorithm. At first, we give the following definitions:

**Definition 1:** Given a set of items \( I = \{ I_1, I_2, \ldots, I_s \} \), and the database of transaction records \( D = \{ t_1, t_2, \ldots, t_n \} \), where \( t_1 = \{ I_{i1}, I_{i2}, \ldots, I_{ik} \} \) and \( I_{ij} \in I \), an association rule is an implication of the form \( X \Rightarrow Y \) where \( X, Y \subset I \) and \( X \setminus Y = \emptyset \).

**Definition 2:** The support (s) for an association rule \( X \Rightarrow Y \) is the percentage (%) of transactions in the database that contains \( X \cup Y \). support \( (X \Rightarrow Y) = P(X \cup Y) \), where \( P \) is the probability.

**Definition 3:** The confidence or strength (\( \Phi \)) for an association rule \( (X \Rightarrow Y) \) is the ratio of the number of transactions that contain \( X \cup Y \) to the number of transactions that contains \( X \). That is confidence \( (X \Rightarrow Y) = P(Y|X) \).

Associations among requirements with high support and confidence suggest standardization. The algorithm uses the following property: If an item set satisfies the minimum support threshold, so do all its subsets. The key of Apriori algorithm is to generate the large item sets and then to generate association rules. Association rules are adopted to discover the interesting relationship and knowledge in a large dataset, Kotsiantis and Kanellopoulo (2006). The problem is decomposed into two sub problems. One is to find those item sets whose occurrences exceed a predefined threshold in the database, which are called frequent, or large item sets. The second problem is to generate association rules from those large item sets with the constraints of minimal confidence. Suppose one of the large item sets is \( L_k \), \( L_k = \{ I_1, I_2, \ldots, I_k \} \), association rules with this item sets are generated in the following way: the first rule is \( \{ I_1, I_2, \ldots, I_{k-1} \} \Rightarrow \{ I_k \} \), by checking the confidence this rule can be determined as interesting or not. Then other rules are generated by deleting the last
items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interest of them. Those processes are iterated until the antecedent becomes empty. The Apriori algorithm finds the frequent sets \( L \) in Database \( D \).

First we can stop generating supersets of a candidate once we determine that it is infrequent, since no superset of an infrequent item set can be frequent. Second, we can avoid any candidate that has an infrequent subset. These two observations can result in significant pruning of the search space.

**Apriori Algorithm**

Find frequent set \( L_{k-1} \)

Join Step.

\( C_k \) is generated by joining \( L_{k-1} \) with itself

Prune Step.

Any \((k-1)\)-itemset that is not frequent cannot be a subset of a frequent \( k \)-itemset, hence should be removed.

where

\( (C_k: \) Candidate itemset of size \( k \) )

\( (L_k: \) frequent itemset of size \( k \) )

Apriori Pseudocode

Apriori \((T, \varepsilon)\)

\( L_1 \leftarrow \{ \text{Large 1-itemsets that appear in more than } \varepsilon \text{ transactions} \} \)

\( k \leftarrow 2 \)

While \( L_{k-1} \neq \emptyset \)

\( C_k \leftarrow \text{Generate } (L_{k-1}) \)

for transactions \( t \in T \)

\( C_t \leftarrow \text{Subset } (C_k, t) \)

For candidates \( c \in C_t \)

\( \text{count}[c] \leftarrow \text{count}[c] + 1 \)

\( L_k \leftarrow \{ c \in C_k | \text{count}[c] \geq \varepsilon \} \)

\( k \leftarrow k + 1 \)

Return \( \bigcup_k L_k \)
6.3.2 FP GROWTH ALGORITHM

The FP Growth Algorithm, proposed by Han in, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). The FP Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

This algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step, it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating in the long frequent patterns, offering good selectivity.

**FP Growth Algorithm:**
Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null).
Procedure FP-growth (Tree, α)
{
   1) if Tree contains a single path P then
      2) for each combination do generate pattern β
         a with support = minimum support of nodes in β.
   3) Else for each header ai in the header of Tree
      Do {
         4) Generate pattern β = ai U α with support = ai.support;
         5) Construct β.s conditional pattern base and then
            β.s conditional FP-tree Tree β
         6) If Tree β = null
            7) Then call FP-growth (Tree β, β)
      }
}
6.4 PROPOSED ALGORITHMS

The customer behavior has been identified by using clustering and association rules. The k-medoid is a classical partitioning technique of clustering that clusters the data set of n objects into k clusters known a priori. It is more robust to noise and outliers as compared to k-means because it minimizes a sum of dissimilarities instead of a sum of squared Euclidean distances. A medoid can be defined as the object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal i.e. it is a most centrally located point in the given data set. The k-medoid method has been applied to cluster the customers into many groups, according to the weighted RFMT values. CLV has been identified for each customer. Finally, for each cluster, the association rules have been used to identify the frequently purchased products by the customers.

6.4.1 RFMT BASED APRIORI ALGORITHM

RFMT based Apriori algorithm is used to extract important and effective rules. RFMT model is used to identify the type of customers and their loyalty. Association rules that are extracted from these important segments of customers are significantly more valuable than those that are extracted without considering the segmentation parameters recency, frequency, monetary and term.

Step 1: Standardize each customer's RFMT value.
Because each of the units of RFMT is different, standardization is necessary.

Step 2: Calculate each customer's CLV.
Multiply each customer’s RFMT value times its weight to get an integrative CLV score.

\[ C' = W_R X'R + W_F X'F + W_M X'M + W_T X'T. \]

Step 3: Calculate the CLV of each cluster.
The k-medoid method is then applied to cluster the customers into many groups, according to the weighted RFMT values. The customers’ standardized values of each cluster are added together, and then divided by the number of customers of each cluster to get the average values. Then, a CLV value can be derived after multiplying the weights as shown in below formula, respectively.

\[ C^j = W_R C^j_R + W_F C^j_F + W_M C^j_M + W_T C^j_T \]

Step 4: Association rules are used to identify the frequent item sets for each cluster.

RFMT based Apriori \((T, \varepsilon)\)
Large 1-itemsets that appear in more than $\varepsilon$ transactions

$k \leftarrow 2$

While $L_{k-1} \neq \emptyset$

$C_k \leftarrow$ Generate $(L_{k-1})$

for transactions $t \in T$

$C_t \leftarrow$ Subset $(C_k, t)$

For candidates $c \in C_t$

$count[c] \leftarrow count[c] + 1$

$L_k \leftarrow \{ c \in C_k | count[c] \geq \varepsilon \}$

$k \leftarrow k + 1$

Return $\bigcup_k L_k$

### 6.4.2 RFMT BASED FP GROWTH ALGORITHM

RFMT based FP (Frequent Pattern) Growth algorithm is used to extract important and effective rules. The FP Growth Algorithm is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern (FP) tree.

**Step 1:** Standardize each customer’s RFMT value.

Because each of the units of RFMT is different, standardization is necessary.

**Step 2:** Calculate each customer’s CLV.

Multiply each customer’s RFMT value times its weight to get an integrative CLV score

$$C'_x = W_R X'_R + W_F X'_F + W_M X'_M + W_T X'_T.$$  

**Step 3:** Calculate the CLV of each cluster.

The K-medoid method is then applied to cluster the customers into many groups, according to the weighted RFMT values. The customers’ standardized values of each cluster are added together, and then divided by the number of customers of each cluster to get the average values. Then, a CLV value can be derived after multiplying the weights as shown in below formula, respectively.
\[C^j = W_R C^j_R + W_F C^j_F + W_M C^j_M + W_T C^j_T\]

Step 4: Association rules are used to identify the frequent item sets for each cluster.

Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null).
Procedure FP-growth (Tree, \(\alpha\))

\{ 
  1) if Tree contains a single path \(P\) then
  2) for each combination do generate pattern \(\beta\)
      \(\alpha\) with support = minimum support of nodes in \(\beta\).
  3) Else for each header \(a_i\) in the header of Tree
     Do {
       4) Generate pattern \(\beta = a_i\) \(\alpha\) with support = \(a_i\).support;
       5) Construct \(\beta\).s conditional pattern base and then
           \(\beta\).s conditional FP-tree Tree \(\beta\)
       6) If Tree \(\beta\) = null
       7) Then call FP-growth (Tree \(\beta\), \(\beta\))
  } 

The input file has been generated and modified according to the requirements of RFMT based FP Growth. There are four operators used in this process. The description of the operators in sequence has been presented below:

**Operator-1- Read CSV**: This operator can read CSV files, where all the transaction details were written into one line and separated by a constant separator. The separator might be specified in the column separators parameter. The default will split the line on each comma, semicolon and blank. The first line is used for the attribute names as default, controlled by the use first row as attribute names parameter. By using this operator, an input file has been processed into rapid miner for further process.

**Operator-2- Numerical to Binominal**: It converts all numerical attributes to binary ones. If the value of an attribute is between the specified minimal and maximal value, it becomes false, otherwise true. If the value is missing, the new value will be missing. The default boundaries are both set to 0, thus only 0.0 is mapped to false and all other values are mapped to true. In original data input file, all values were in binary format, but for the implementation
of FP growth all data required to be converted in binominal (True/False) format. Therefore, we used this operator to value conversion.

**Operator-3- FP Growth:** This operator calculates all frequent items sets from a data set by building a FP Tree data structure on the transaction data base. This is a very compressed copy of the data which in many cases fits into main memory even for large data bases. From this FP Tree, all frequent item sets were derived. A major advantage of FP Growth compared to Apriori is that it uses only 2 data scans and is therefore often applicable even on large data sets.

**Operator-4- Create Association Rules:** This operator generates association rules from frequent item sets. In Rapid Miner, the process of frequent item set mining is divided into two parts: first, the generation of frequent item sets and second, the generation of association rules from these sets. The result will be a set of frequent item sets, which could be used as input for this operator. Finally this operator takes the input from the FP growth algorithm, which means that this operator only takes the list of frequent item sets generated by the previous algorithm operator.

**6.5 SUMMARY**

The analysis of customer behavior is used to maintain good relationship with customers in order to maximize the customer satisfaction. We can also improve customer loyalty and retention. RFMT based Apriori and FP-Growth algorithm have been proposed for association rule mining.