Week 2 Topics

* Chapter 5 – Benchmarks

The prediction accuracy for the *ith* observation is defined as the difference between its actual value and its predicted y value is . A few popular numerical measures of predictive analytics are:

# Mean Absolute Error(MAE)

The mean absolute error is one of the simpler errors to understand. It takes the absolute difference between the actual and predicted values and finds the average. Finding the absolute value is important because it doesn’t allow for any form of cancellation of error values. For instance, if you were to take the average of 1 and -1 then you would have an average value of 0 because the 1 and -1 would essentially cancel each other out.

To avoid this we use the absolute value. This is how you find the MAE mathematically

The math formulae is:

Where actual value (y) and predicted value are two column in the dataset which used for prediction. represent an observation index and n represents the total number of observations

# Root Mean Squared Error (RMSE)

The root mean squared error seems somewhat similar to the MAE. They both take the difference between the actual and the forecast. However, the RMSE also then squares the difference, finds the average of all the squares and then finds the square root. Now it might seem like the action of squaring and then taking the square root may cancel each other out. This isn’t the case. The RMSE essentially punishes larger errors. Another way to phrase that is it puts a heavier weight on larger errors.

For example, let’s compare the two tables in figure 1. If you notice, the MAE and RMSE are nearly identical for both table 1 and table 2. However, the difference between the two values, even when increase in error is only 1 gets slightly larger as denoted in the first row. If the error were 5, 6, or another larger number, the difference between the RMSE and MAE would grow even larger. This is because you square the number. This creates an exponential change in the base number. Thus, an error difference of 1 has a greater effect for every increase e.g. from (3 to 4 then from 4 to 5). This is why it essentially punishes larger errors.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 1 | | | |
| Actual | Predicted | Error (absolute) | Error Squared |
| 10 | 12 | 2 | 4 |
| 12 | 11 | 1 | 1 |
| 10 | 13 | 3 | 9 |
|  |  |  |  |
|  |  | MAE | 2 |
|  |  | RMSE | 3.74165739 |
|  |  |  |  |
| Table 2 | | | |
| Actual | Predicted | Error (absolute) |  |
| 10 | 13 | 3 | 9 |
| 12 | 11 | 1 | 1 |
| 10 | 13 | 3 | 9 |
|  |  |  |  |
|  |  | MAE | 2.333333333 |
|  |  | RMSE | 4.35889894 |

Figure 1: RMSE and MAE comparison

Below is again the mathematical notation of the RMSE.

# Mean Absolute Percentage Error(MAPE)

The one issue you may run into with both the RMSE and MAE is that both values can just become large numbers that don’t really say all that much. What does a RMSE of 597 mean? How bad or good is that? Part of this is because you need to compare it to other models. Another issue is the fact that the RMSE will be based off the difference of the actual and predicted, which depending on your data could be on very different scales. For instance, if you are creating a model for a billion-dollar corporation your error will be much larger than one for a company that only grosses 6 figures.

In this case, the mean absolute percentage error is good method in the sense that it is the percentage of the error compared to the actual value. This provides more of a standardized error measure. For instance, if the error was 10 and the actual value was 100, then the percentage would be 10% compared to if the error was 100 and the actual value was 1000, the measure would still be 10%.

This provides a little more context than the RMSE and the MAE which can help better explain the model’s accuracy.

The mathematical notation is listed below

​

# Mean Absolute Scaled Error (MASE)

The mean scaled error is the last error that we will be discussing in this notes. The MASE is slightly different than the other three. It compares the MAE of your current model you are testing to the MAE of the naïve model. The naïve model just *forecast* the previous observation to the current observation. (this is just for forecasting not predicting)

​The MASE is the ratio of the MAE over the MAE of the naïve model. In this way, when the MASE is equal to 1 that means that your model has the same MAE as the naive model, so you almost might as well pick the naïve model. If the model’s MASE is .5, that would suggest that your model is about 2 times as good as just picking the previous value.

This error skips the step of running several models and instead automatically compares your model to another one. It provides a little more context than the MAE, RMSE and MAPE.

As you see the MASE is only used in forecasting.

* Chapter 5 – Gains and ROC curves

and their difference

# Introduction

When we develop statistical models for classification tasks (e.g. using machine learning algorithms), we usually need to have a way to compare the generated models to decide which model is best. Typical tools for this task are Gains, ROC or Lift charts. All of them are popular, so there arises a question what is the difference between them. This section sets out to describes these charts in a detailed manner by detailing their with illustrations.

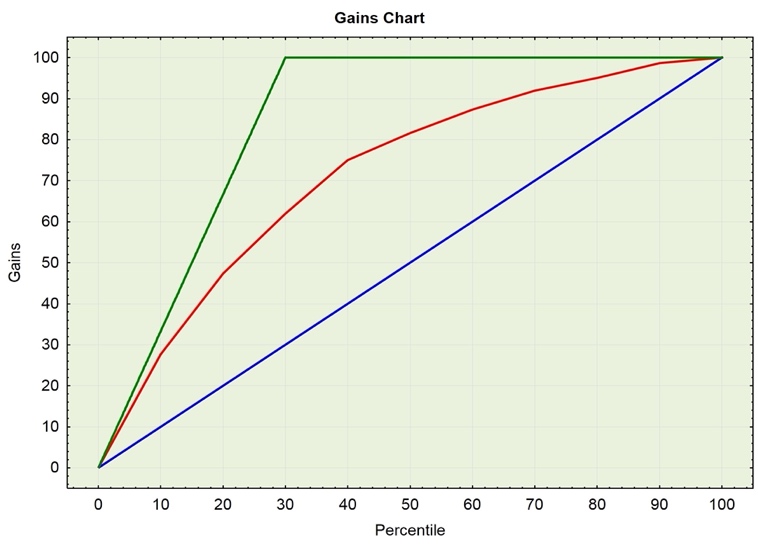
**Gains chart**

Typically called a Cumulative Gains Chart it can be simply explained by the following example:

For simplicity let's assume we have 1000 customers. If we run an advertising campaign for all our customers, we might find that 30% (300 out of 1000) will respond and buy our new product.

Marketing to all our customers could be one strategy for running a campaign. But this is not the optimum use of our marketing dollars, considering the cost of campaign for especially for large number of customers. Therefore we would like to have a better way of running this advertising campaign so that instead of targeting all our customer base, we target only to those customers with a **high probability of responding positively** to the campaign. This will, firstly lower the cost of the campaign and, secondly (and maybe more importantly) we will not disturb those customers with advertising who have no interest in our new product.

This is where predictive classification models come in. There are lots of different models such as classification tree, and neural networks, but no matter which one we use, we can still evaluate the results of our model by using Cumulative Gains Charts. As you know by now, If we have historical data on the reactions of customers to past campaigns then, we can use this data to build a model that predicts, if a particular customer will respond by buying the product or not. The results of such a model are typical, for each customer, the probability of a positive and negative reaction from the customer. We can sort customers according to the probability of a positive reaction to the campaign and run the campaign only for a percentage of customers with the highest probability.



The Gains chart is the visualization of that principle. On the X-axis we have the percentage of the customer base we want to target with the campaign. The Y-axis gives us the answer to the percentage of all positive responses customers have found in the targeted sample. In the figure 2 on the right, you can see an example of the Gains chart. (The gains chart associated with the model is the red curve):

**Figure 2: Gain chart example (red line)**

What can we read from the graph? What happens if we only target 10% of our customer base? According to the results of our model, if we will take the 10% of customers with the highest probability of a positive response, we will get 28% of all the possible positive responses .

This means we will find 84 customers with positive responses from the 100 customers reached by the campaign (84 is 28% of 300 positive response customers in our customer base).

In little more detail we can say that our model gives us 28 for 100 customers (10 on the x line and 28 on y axis). Now if out of 100 84 responded based on our predictive model then how many we should choose to be able to get the same number without a predictive model? See the purple lines and they show 300 which is: (84 x 100)/28 = 300

With an increase of targeted customers to 50%, we already have more than 80% of those who will, in a real situation, give a positive response. If this is our selected strategy for the real campaign (reaching 50% of our customers by the model), then we will have reached 80% of all the positive responses and saved 50% of our costs of running the campaign (we do not want to run the campaign to customers that are not likely to respond positively).

The choice of the percentage to be targeted in the campaign depends on the concrete costs for the campaign and the profit from the expected positive responses. The Gains chart is a display of the expected results base on the choice of the percentage targeted. Our final strategy, therefore, consists of the model and the targeted percentage (instead of the percentage we can define the cut-off value for probabilities - if the probability is above this value/threshold we will include the customer in the campaign).

It was already said that the **Red curve** represents the proposed model. The **Blue curve** represents the gains chart of a random model. In this case, we are displaying the observed results of picking customers randomly without any selection criteria, which assumes that we would get the same proportion of positive responses if we target the whole customer base. In other words, If we target 10% of all customers, we will have 10% of all the positive responses within our 10% sample. The curves are meeting at (0, 0) and (100, 100), the second point means we run the campaign to all customers, therefore the output (all those who responded positively) is the same as the observed results. When we are using a predictive model, in this case picking customers according to sorted probabilities, it does not make sense when we include all customers.

The **Green curve** is the optimal model, the best possible order for picking customers – we will first target all customers with a positive response and then those with a negative response. The slope of the first part of the green curve is 100/(percentage of all positive responses).

**X and Y values and Base line (the diagonal line from 0,0 to 100,100)**

The y-axis shows the percentage of positive responses. This is a percentage of the total possible positive responses.

The x-axis shows the percentage of customers contacted, which is a fraction of the total customers. Considering

**Baseline** (overall response rate): If we contact X% of customers then we will receive X% of the total positive responses.

**Analyzing the Charts**: Cumulative gains and lift charts are a graphical representation of the advantage of using a predictive model to choose which customers to contact. The lift chart shows how much more likely we are to receive respondents than if we contact a random sample of customers. For example, by contacting only 10% of customers based on the predictive model we will reach 3 times as many respondents as if we use no model.

**Confusion Matrix**

Test a predictive classification model using *confusion matrix*

A confusion matrix (Kohavi and Provost, 1998) contains information about actual and predicted classifications done by a classification algorithm (model). Performance of such models is commonly evaluated using the data in the matrix. The following two tables showing the confusion matrix for a two-class classifier. The first one is a over simplified version and the second one is the detailed one.

Table 1

|  |  |  |
| --- | --- | --- |
|  | Prediction POSITIVE | Prediction NEGATIVE |
| Actual POSITIVE | Count TP (Right Decision) | Count FN (Error of second kind) |
| Actual NEGATIVE | Count FP (Error of the first kind) | Count TN (right decision) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion matrix** | | |  |
|  | **Prediction Positive** | **Prediction Negative** | **Measures** |
| **Actual POSITIVE** | **TP – True Positive (Hit)** | FN – False Negative (type II error) |  |
| **Actual NEGATIVE** | FP – False Positive (type I error) | **TN – True Negative (Correct Rejection)** |  |
|  |  |  |  |

Table 2

Where:

* TP – True positive rate: The number of samples correctly marked as positive
* TN – True negative rate: The number of samples correctly marked as negative
* FP – False positive rate: The number of samples incorrectly marked as positive (aka type 1 error)
* FN – False negative rate: The number of samples incorrectly marked as negative (aka type 2 error)
* P – Condition positive: The number of real positive cases in the data

(P = TP + FN)

* N – Condition negative: The number of real negative cases in the data

(N = FP+TN)

This matrix summarizes the performance of a classification model. We apply several models on our problem and, for each trial, we build a confusion matrix. Next, we need a way to compare these matrices to find out the best model.

**Analyzing Error Type I & II**

If our model is perfect the matrix should be diagonal with only the true positive and true negative values.

It’s most likely not going to happen … so how to choose? should we favor type I or type II error, or the matrix with the lowest error rate overall?

Well the answer is: it depends. It depends on what you are trying to predict.

Let’s assume we have a classification model (a medical test for blood analysis) to predict a patient is sick or not. Then we want to find out the accuracy of the model.

In this case, you probably want the lowest possible False Negative rate. (You want to avoid letting a person with the disease untreated). That might also mean a higher False Positive rate (in this case a person without the disease but a positive test could probably find out that she’s safe with further medical investigation).

Minimizing the error II is easy: predict that all the patients are positive (sick). No more type II error but the test becomes useless as every patient now need further medical investigation.

|  |  |
| --- | --- |
| Sensitivity – True Positive Rate |  |
| Specificity – True Negative Rate |  |
| Precision – Positive Predicted Value |  |
| Negative Predicted Value |  |
| Fall-out – False Positive Rate |  |
| False Discovery Rate |  |
| Miss Rate – False Negative Rate |  |
| Accuracy |  |
| F1 Score |  |

So how to deal with it? Well, from the confusion matrix there are many indicators or scores that can be computed:

Depending on your needs you may pick up one of these scores to compare your models. In practice the accuracy, f1-score and sensitivity are the most frequently used.

For better orientation, it is common practice to display the confusion matrix in the form of the following graph (Figure 3). From this graph, we see, how many times the model predicts correctly (true negatives and true positives) and how many times we have an incorrect prediction (false positives and false negatives). The better the model, the larger the bars TP and TN in comparison to FN, and FP.

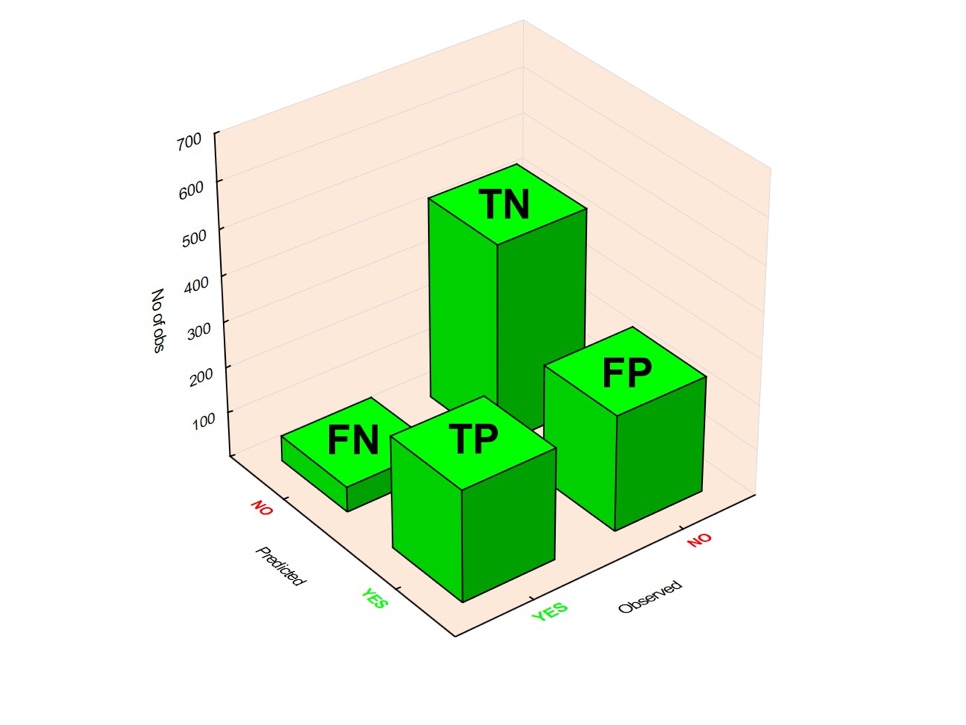


Figure 3: Confusion Matrix plot

**ROC curve**

A point on the gains chart is equivalent to Sensitivity against

Against

The second term is on the X-axis and it is a fraction of targeted customers

The Gains or Lift and ROC charts are computed based on information from confusion matrices. It is important to realize that curves are created according to a larger number of these confusion matrices for various targeted percentages/cut-off values.

in the confusion matrix:

Other terms connected with a confusion matrix are Sensitivity and Specificity. They are computed in the following way:

The ROC curve (Receiver Operating Characteristics curve) is the display of sensitivity and specificity for different cut-off values for probability (If the probability of a positive response is above the cut-off, we predict a positive outcome, if not we are predicting a negative one). Each cut-off value defines one point on the ROC curve, plotting the cut-off for the range of 0 to 1 will draw the whole ROC curve. The Red curve on the ROC curve diagram Figure 4 below is the same model as the example for the Gains chart:

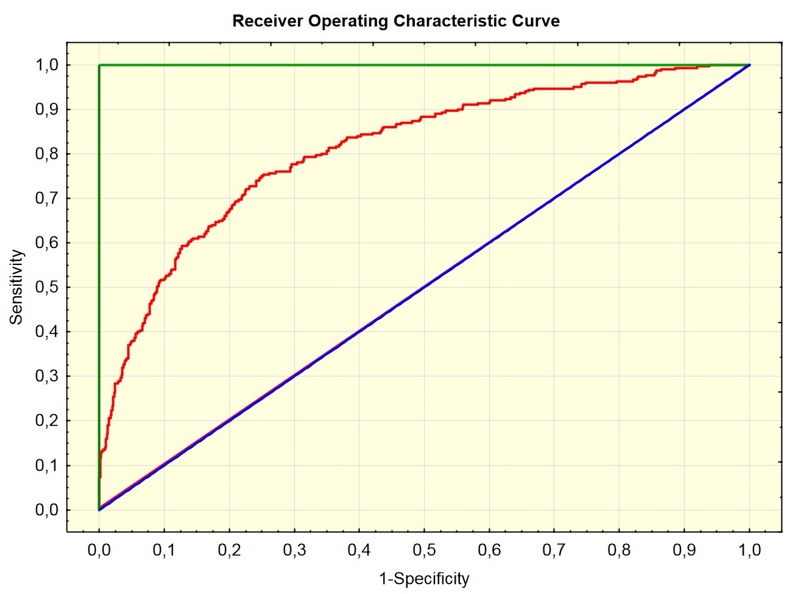
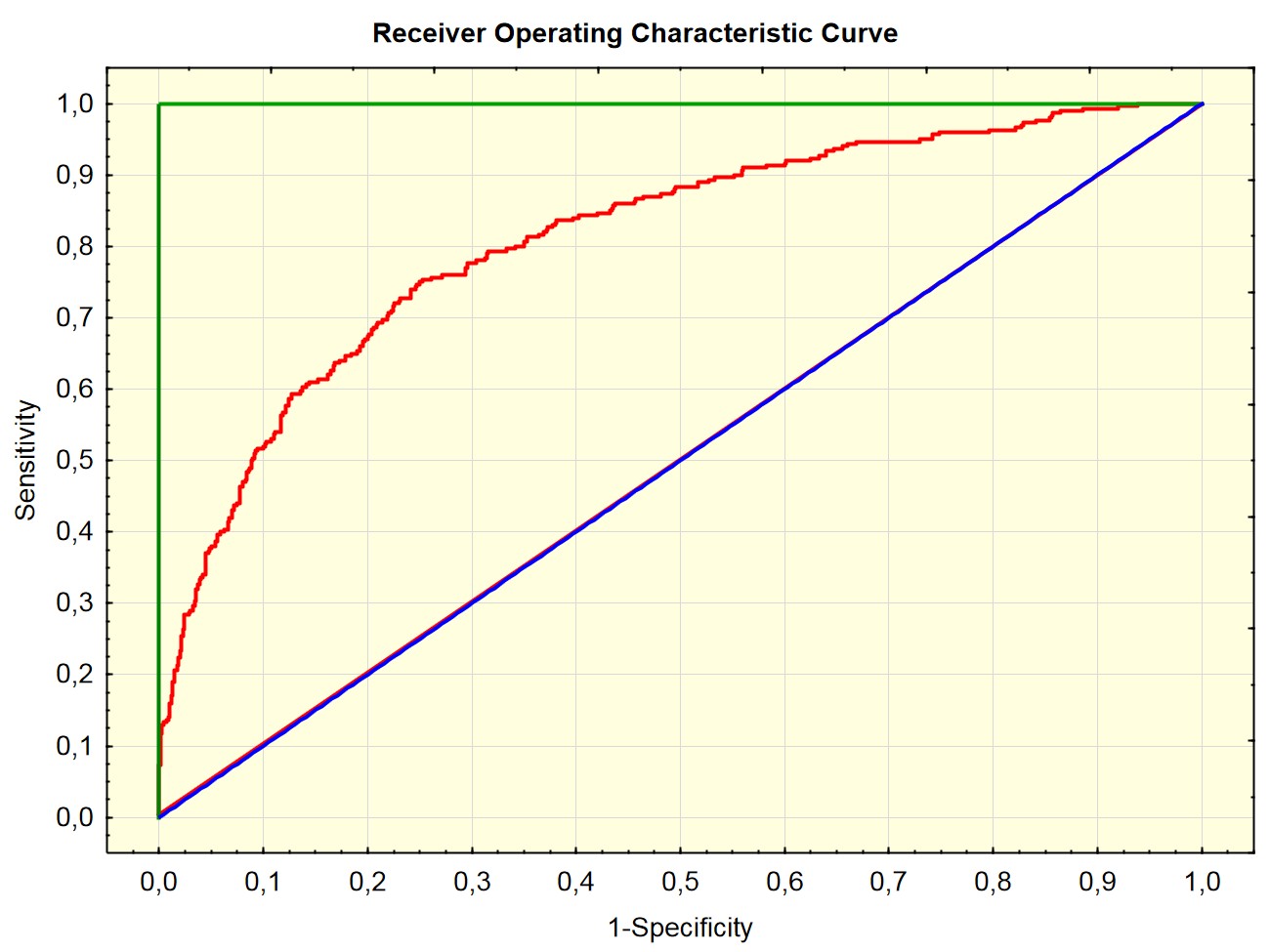


Figure 4: ROC Curve



Check the text book chapter 5 for Gain and ROC charts R code

* Chapter 15 – Affinity Analysis

Affinity analytics is a data analysis and data mining technique that discovers co-occurrence relationships among activities performed by (or recorded about) specific individuals or groups. In general, this can be applied to any process where agents can be uniquely identified and information about their activities can be recorded. In retail, affinity analysis is used to perform market basket analysis, in which retailers seek to understand the purchase behavior of customers. This information can then be used for purposes of cross-selling and up-selling, in addition to influencing sales promotions, loyalty programs, store design, and discount plans.

## Association Rule Mining

**Association rule learning** is a method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in data using some measures of interestingness. Based on the concept of strong rules, we can introduce association rules for discovering regularities between products in large-scale transactional data recorded by point-of-sale (POS) systems in stores. For example, the rule {onion, potatoes} →{burger} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements. In addition to the above example from market basket analysis, association rules are employed today in many application areas such as Web usage mining, intrusion detection, continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

In simple words, association rule mining means: “Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

**Example of Association Rule: Market-Basket transactions**

|  |  |
| --- | --- |
| TID | Items |
| 1 | Bread, Milk |
| 2 | Bread, Diapers, Beer, Eggs |
| 3 | Milk, Diapers, Beer, Coke |
| 4 | Bread, Milk, Diapers, Beer |
| 5 | Bread, Milk, Diapers, Coke |

Example of association rules

{Bread} → {Milk}

{Diaper}→ {Beer}

Table 1 : Transactions

## Association Mining Usages

Association Rule Mining is used when you want to find an association between different objects in a set, find frequent patterns in a transactional database, relational databases or any other information repository. The applications of Association Rule Mining are found in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering, and classification. It can tell you what items customers frequently buy together by generating a set of rules called **Association Rules**. In simple words, it gives you output as rules in form **if this then that**.

Outcome association rule mining can be used for numerous marketing strategies such as:

* Changing the store layout according to trends
* Customer behavior analysis
* Catalogue design
* Cross marketing on online stores
* What are the trending items customers buy
* Customized emails with add-on sales

## Problem Definition and Mathematical Description

Binary Representation Market basket data can be represented in a binary format as shown in Table 2, where each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TID | Bread | Milk | Diapers | Beer | Eggs | Coke |
| **1** | 1 | 1 | 0 | 0 | 0 | 0 |
| **2** | 1 | 0 | 1 | 1 | 1 | 0 |
| **3** | 0 | 1 | 1 | 1 | 0 | 1 |
| **4** | 1 | 1 | 1 | 1 | 0 | 0 |
| **5** | 1 | 1 | 1 | 0 | 0 | 1 |

**Table 2: A binary 0/1 representation of market basket data**

Following the original definition of the association rule mining and is defined as:

Let  (set of items) be a set of n binary attributes called items and

Let T =  (transaction set) be a set of m transactions called the dataset

Each transaction in T has a unique transaction ID and contains a subset of the items in I.

A rule is defined as an implication of the form:

* In the above example,
* In the above example, for a given set of transactions (not all possible transactions) see the right column of the table 1.

Every rule is composed by two different set of items, also known as ***itemsets***, X and Y, where X is called antecedent or left-hand-side (LHS) and Y consequent or right-hand-side (RHS).

An example rule for the supermarket could be {Bread, Milk} → {Diapers} meaning that if Bread and Milk are bought, customers also buy Diapers.

Note: this example is extremely small. *In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions*.

## Useful Definitions

* **Itemset**

A collection of one or more items selected from the set of items, I. For example: {Milk, Bread, Diapers}. K-itemset is an itemset with k items. In our example we have 3-itemset

* **Support count**

The transaction width is defined as the number of items presented in a transaction.

A transaction tj is said to contain an itemset X if X is a subset of tj. For example, the second transaction shown in Table 2 contains the itemset {Bread, Diapers} but not {Bread, Milk}. That is {Bread, Milk} is not a subset of t2 or

An important property of an itemset is its support count (or the frequency of occurrence of an itemset), which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, σ(X), for an itemset X can be stated as follows:

Where the symbol |{ }| denote the number of element in the set,

Using the table 1: σ ({Bear, Diapers, Milk}) = 2

* **Support**

Fraction of transactions that contain an itemset. For example: supp ({Milk, Bread, Diapers}) = 2/5

* **Frequent Itemset**

Is an itemset whose support is greater than or equal to a *minsup* threshold

## Association Rule Evaluation Metrics

An association rule of a set of transactions T is an implication expression of the form , where X and Y are both member of a transaction *ti* and disjoint itemsets, that is, and X ∩Y = ∅. The strength of an association rule can be measured in terms of its **support** and **confidence**. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in Y appear in transactions that contain X. The formal definitions of these metrics are

and

In order to select interesting rules from the set of all transaction’s rules (the set of all transaction rules is a subset of all possible rules), constraints on various measures of significance and interest are used. The best-known constraints are minimum thresholds on support (minsup) and confidence (minconf).

We use the following example for explanation of support and confidence values.

Consider the rule, since the support count for is 2 and the total number of transactions is 5, the rule’s support is 2/5 = 0.4. The rule’s confidence is obtained by dividing the support count for {Milk, Diapers, Beer} by the support count for {Milk, Diapers}. Since there are 3 transactions that contain milk and diapers, the confidence for this rule is 2/3 = 0.67.

* **The *suppor*t value** of X, supp(X) with respect to T is defined as the proportion of transactions in the database which contains the itemset X. In the example database (table 1), the itemset  , has a support of 2/5 = 0.4 since it occurs in 40% of all transactions (2 out of 5 transactions). The argument of supp() is a set of preconditions and thus, becomes more restrictive as it grows (instead of more inclusive).
* **The *confidence* value** of a rule, , with respect to a set of transactions T, is the proportion of the transactions that contains X which also contains Y. Confidence is defined as:

For example, the rule has a confidence of 0.4/0.6 = 0.67 in the database, which means that for 67% of the transaction containing bread and milk the rule is valid (or true) (67% of times a customer buys bread and milk, diapers are bought as well)

Note that we can rewrite the support and confidence formulas with support count. Assuming the same rule:

The Lift of a rule is defined as:

Calculating the Lift of our example rule:

Lift signifies the likelihood (chance) of the itemset *Y* being purchased when item X is purchased while taking into account the popularity of Y. The fewer is the occurrences of X and Y relative to occurrences of X and Y together indicates that the X and Y are more related to each other!

## Association Rule Mining Task

Given a set of transactions T, the goal of association rule is to find all rules having

* Support >= minsup threshold
* Confidence >= minconf threshold
* Higher Lift

## Association Rules Algorithms

## AIS

## SETM

## Apriori

## AprioriTid

## AprioriHybrid

## <https://www.saedsayad.com/association_rules.htm> (not for students)

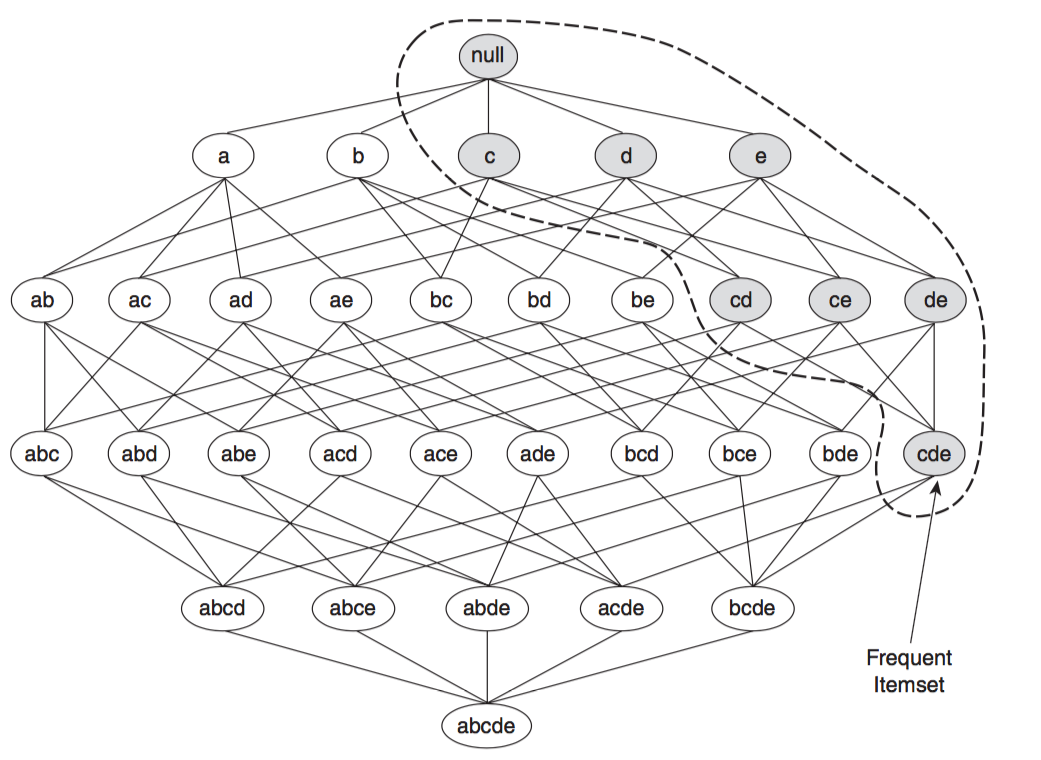
## Apriori Algorithm

***Apriori*** is an algorithm for frequent itemset mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by *Apriori,* can be used to determine association rules which highlights general trends in the dataset

### Apriori principle

This section describes how the support measure helps to reduce the number of candidate-itemsets explored during frequent itemset generation. The use of support for pruning candidate itemsets is guided by the following principle.

*Theorem (Apriori Principle). If an itemset is frequent, then all of its subsets must also be frequent.*

To illustrate the idea behind the Apriori principle, consider the itemset lattice shown in Figure 2. Suppose {c, d, e} is a frequent itemset. Clearly, any transaction that contains {c, d, e} must also contain its subsets, {c, d}, {c, e}, {d, e}, {c}, {d}, and {e}. As a result, if {c, d, e} is frequent, then all subsets of {c, d, e} (i.e., the shaded itemsets in this figure) must also be frequent.

**Figure 2. An illustration of the Apriori principle. If {c, d, e} is frequent, then all subsets of this itemset are frequent.**

### Example

We use the following set of transactions to show how the association rules are generated and selected. In this example, I, set of items is {1, 2, 3, 4, 5}

|  |  |
| --- | --- |
| TID | Items |
| 100 | 1, 3, 4 |
| 200 | 2, 3, 5 |
| 300 | 1, 2, 3, 5 |
| 400 | 2, 5 |

Remember:

* ***Frequent Itemset Property:***

*Any subset of a frequent itemset is frequent.*

* ***Contrapositive:***

*If an itemset is not frequent then, none of its supersets are frequent.*

In the context of Apriori algorithm, a supper set S is a set which contain as a subset the selected itemset. For example, set S={…, ei,…, ej,…., ek,…} is a superset for {ei}, {ei, ej}, {ei, ek}, {ej, ek}, and {ei, ej, ek}. If any of these sets are not frequent then S is not frequent.

**Algorithm Implementation**

Using the example above transaction table.

Step 1: Algorithm generates all I’s itemsets and calculate the frequency of each. Eliminating the rules (sets) with Frequency less than 2 ({4}), assuming frequency (minsup = 2) threshold is equal and greater than 2.

|  |  |
| --- | --- |
| Rule | frequency |
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {4} | 1 |
| {5} | 3 |

|  |  |
| --- | --- |
| Rule | frequency |
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {5} | 3 |

Repeating the step 1: generating 2-itemset. Remember K-itemset is an itemset with k items.

Eliminating the rule with frequency less than 2 ({1, 2} and {1, 5})

|  |  |
| --- | --- |
| Rule | frequency |
| {1, 2} | 1 |
| {1, 3} | 2 |
| {1, 5} | 1 |
| {2, 3} | 2 |
| {2, 5} | 3 |
| {3, 5} | 2 |

|  |  |
| --- | --- |
| Rule | frequency |
| {1, 3} | 2 |
| {2, 3} | 2 |
| {2, 5} | 3 |
| {3, 5} | 2 |

->

Repeating step 1: generating 3-itemset. Eliminating the rule with  
Frequency less than 2 ({1, 2, 3} and {1, 3, 5})

|  |  |
| --- | --- |
| Rule | Frequency |
| {1, 2, 3} | 1 |
| {1, 3, 5} | 1 |
| {2, 3, 5} | 2 |

->

|  |  |
| --- | --- |
| Rule | Frequency |
| {2, 3, 5} | 2 |

|  |  |
| --- | --- |
| Rule | Frequency |
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {5} | 3 |
| {1, 3} | 2 |
| {2, 3} | 2 |
| {2, 5} | 3 |
| {3, 5} | 2 |
| {2, 3, 5} | 2 |

Result: Now as you see there are no more rules to generate! Thus, algorithm output is the 1-itemset, 2-itemset, and 3-itemset with frequency greater than 2

Now, let’s look back at our original transaction set T and calculates support, confidence, and lift for each possible rule. Let’s pick the {3} -> {2, 5} rule

|  |  |
| --- | --- |
| TID | Items |
| 100 | 1, 3, 4 |
| 200 | 2, 3, 5 |
| 300 | 1, 2, 3, 5 |
| 400 | 2, 5 |

* The ***Lift*** of a rule is defined as:

Calculating the Lift of our example rule:

## Computational Complexity

The computational complexity of the Apriori algorithm can be affected by the following factors.

**Support Threshold:** Lowering the support threshold often results in more itemsets being declared as frequent. This has an adverse effect on the computational complexity of the algorithm because more candidate itemsets must be generated and counted. As the maximum size of the frequent itemsets increases, the algorithm will need to make more passes over the data set.

**Number of Items (Dimensionality)**: As the number of items increases, more space will be needed to store the support counts of items. If the number of frequent items also grows with the dimensionality of the data, the computation and I/O costs will increase because of the larger number of candidates itemsets generated by the algorithm.

**Number of Transactions:** Since the Apriori algorithm makes repeated passes over the dataset, its run time increases with a larger number of transactions.

**Average Transaction Width (defined as the number of items present in a transaction)**: For dense data sets, the average transaction width can be very large. This affects the complexity of the Apriori algorithm in two ways. First, the maximum size of frequent itemsets tends to increase as the average transaction width increases. As a result, more candidate itemsets must be examined during candidate generation and support counting.

* R – “Apriori” Algorithm

R provides all required function to conduct association rule analysis. However, we should make sure that we know the structure of our dataset prior to apply R functions. The following steps will help to conduct association rules analysis with R.

1. You should know the concept and steps of Apriori algorithm
2. You should know the meaning of Apriori algorithm measures, support, confidence, and lift.
3. Install the required R packages

library("arules")

library("arulesViz") #for visualization

1. You should know the structure of your dataset. Three types of dataset are acceptable by R’s Apriori function. these structure should be prepared into a right transactional format before given to the R’s Apriori function.

## Dataset Type and proper code to prepare

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item 1 | Item 2 | Item 3 | Item 4 | Item 5 |
| citrus fruit | semi-finished bread | margarine | ready soups |  |
| tropical fruit | yogurt | coffee |  |  |
| whole milk |  |  |  |  |
| pip fruit | yogurt | cream cheese | meat spreads |  |
| other vegetables | whole milk | condensed milk | long life bakery product |  |
| whole milk | butter | yogurt | rice | abrasive cleaner |
| rolls/buns |  |  |  |  |
| other vegetables | UHT-milk | rolls/buns | bottled beer | liquor (appetizer) |
| potted plants |  |  |  |  |

* 1. Multiple Items Transaction (without repetition) dataset (example: market.csv)

As you see transaction 1 itemset is {citrus fruit, semi-finished bread, margarine, ready soups}

* 1. Single Item Transactional (with repeating transaction number) dataset (example: grocery.csv)

|  |  |
| --- | --- |
| id | item |
| 1 | Toothbrush |
| 1 | Toothpaste |
| 1 | Moisturizer |
| 2 | Toothbrush |
| 2 | Toothpaste |
| 2 | Pen |
| 3 | Toothpaste |
| 3 | Pen |
| 4 | Toothpaste |
| 4 | Paper |
| 5 | Toothbrush |

As you see transaction 1 itemset is {Toothbrush, Toothpaste, Moisturizer} and each item is in one row!

* 1. Tabular binary dataset (example: cosmetics.csv)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trans. | Bag | Blush | Nail Polish | Brushes | Concealer | Eyebrow Pencils | Lip liner | Mascara |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

As you see transaction 1 itemset is {blush, Nail Polish, Brushes, Concealer, Lip liner}

1. Codes for transactional and binary dataset is different. Look at the data file before loading it into R. after viewing the data file, load it into R and apply correct R codes.

## R Codes for Reading and implementing the algorithm

* 1. R code for transactional data without repetition (format ***a***)

#reading the dataset as a matrix

library(arules)

basket <- read.transactions("market.csv", sep = ",")

# look at the first five transactions

inspect(basket[1:5])

'"

Sample output of the dataset ready for apriori algorithm

items

[1] {citrus fruit, margarine, ready soups, semi-finished bread}

[2] {coffee, tropical fruit, yogurt}

[3] {whole milk}

[4] {cream cheese, meat spreads, pip fruit, yogurt}

[5] {condensed milk, long life bakery product, other vegetables, whole milk}

"'

* 1. R codes for transaction dataset with repeating Transaction Number (format ***b***).

grocery.df<-read.csv("grocery.csv")

binary<-as(split(grocery.df[,2], grocery.df[,1]), "transactions")

inspect(binary)

#the sample output of the dataset ready for apriori algorithm

items transactionID

[1] {Moisturizer,Toothbrush,Toothpaste} 1

[2] {Pen,Toothbrush,Toothpaste} 2

[3] {Pen,Toothpaste} 3

[4] {Paper,Toothpaste} 4

[5] {Paper,Toothbrush,Toothpaste} 5

[6] {Pen,Toothbrush} 6

[7] {Moisturizer,Pen,Toothbrush,Toothpaste} 7

[8] {Paper,Toothpaste} 8

[9] {Soda} 9

[10] {Pen,Toothpaste} 10

* 1. R codes for binary tabular data file (format ***c***)

cosmetics.df<-read.csv("Cosmetics.csv")

cosmetics.mat <- as.matrix(cosmetics.df)[,-1] (remove the transaction number)

#Transform the matrix into a transaction list

cosmetics.trans <- as(cosmetics.mat, "transactions")

inspect(cosmetics.trans[1:5])#display the first 5 transactions

'#

#the sample output of the dataset ready for apriori algorithm

items

[1] {Blush, Nail.Polish, Brushes, Concealer, Bronzer, Lip.liner, Mascara, Eyeliner}

[2] {Nail.Polish, Concealer, Bronzer, Lip.liner, Foundation, Lip.Gloss}

[3] {Blush, Concealer, Eyebrow.Pencils, Bronzer, Lip.liner, Mascara, Eye.shadow, Foundation, Lip.Gloss, Lipstick}

[4] {Nail.Polish, Brushes, Concealer, Bronzer, Foundation, Eyeliner}

[5] {Blush, Concealer, Bronzer, Lip.liner, Mascara, Eye.shadow, Lip.Gloss, Lipstick}

#'

1. Manipulating and customizing output rules using grocery.csv dataset

The following functions provides customization. I used my grocery example. You can adjust it to your case Apriori rules

#inspecting those rules where rhs is item Pen.

penrules<-apriori(binary, parameter = list(support = 0.1, confidence = 0.1), appearance = list(default = "lhs", rhs="Pen"))

#inspect rules which confidence is greater than 0.05 and then greater than 0.5

Cg0.05rules<-grocery.rules[quality(grocery.rules)$confidence>0.05]

inspect(Cg0.05rules)

Cg0.5rules<-grocery.rules[quality(grocery.rules)$confidence>0.5]

#Get top 10 lift rules

Top.10.lift.Rules<-sort(grocery.rules, decreasing = TRUE, na.last = NA, by = "lift")

inspect(head(Top.10.lift.Rules, 10))

1. Plotting. I used my grocery rules to build interactive and static

#graph items frequency

graphitemFrequencyPlot(binary)

#scatter plot with different color

library(colorspace)

plot(grocery.rules, control = list(col=sequential\_hcl(100)))

plot(grocery.rules, col=sequential\_hcl(100))

plot(grocery.rules, method = "two-key plot")

#creating interactive plot

plot(grocery.rules, method = "graph", engine = "htmlwidget")

The following to plots are the interactive plot of grocery as well as grocery rules.

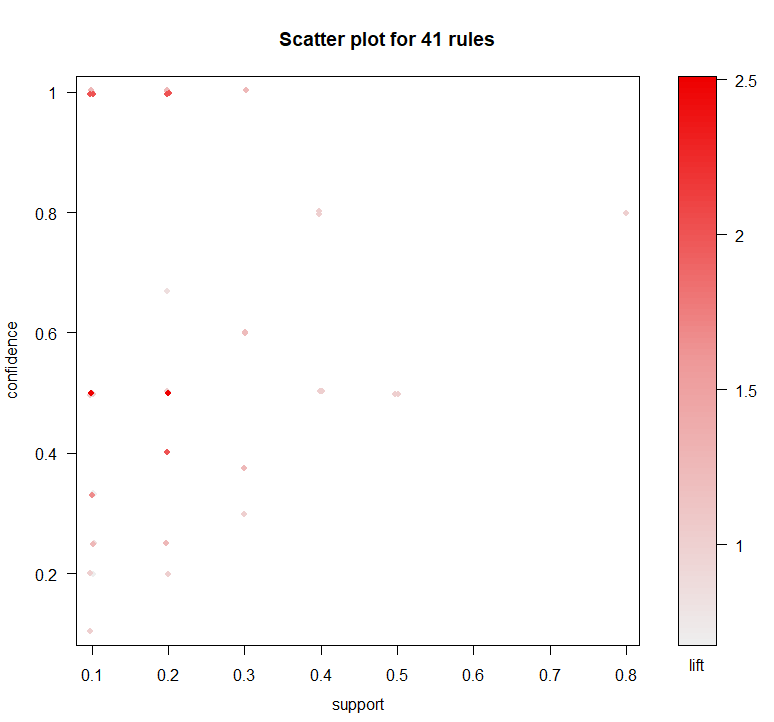


Figure 3: Grocery transactions Rules

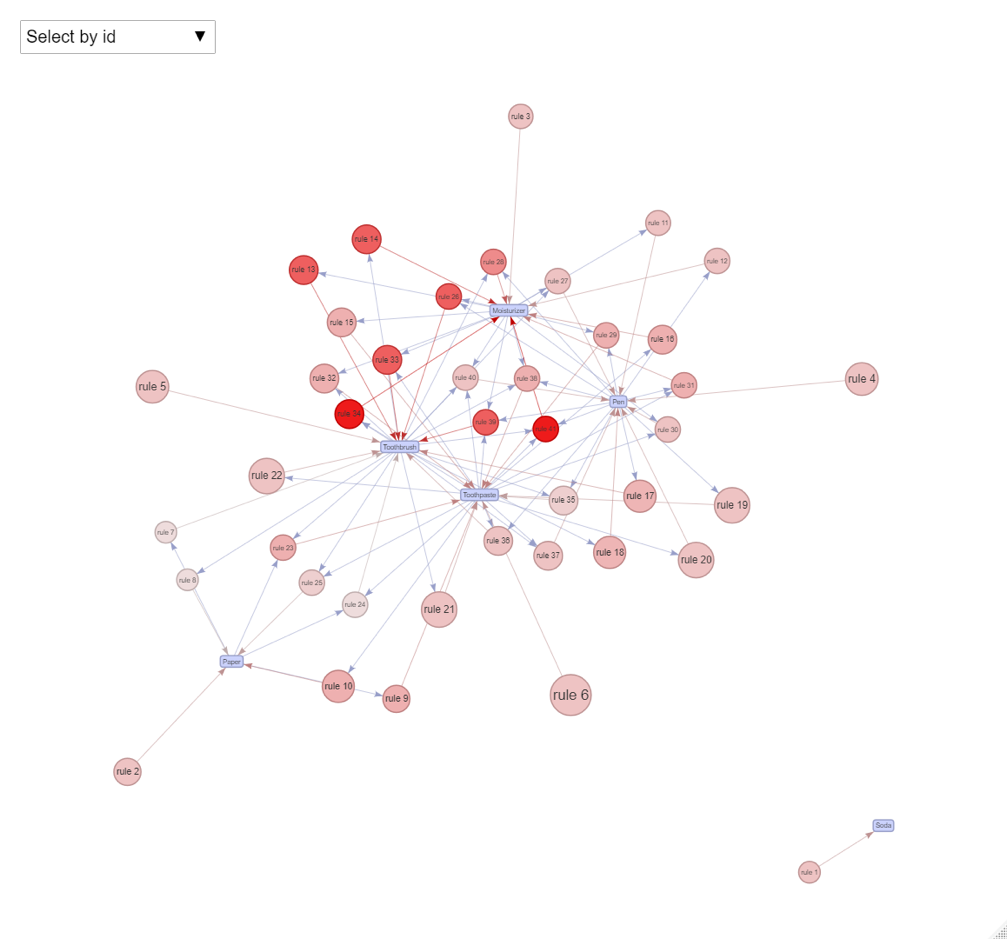


Figure 4: Grocery Transaction Rules Interactive

[

1. Output the Apriori rules into a csv file for use with other visualization tools

after inspecting the rules and their measure with the inspect code,

* write the rules and measures in a CSV file. Save the CSV as an XL file.

write.csv(inspect(cosmetics.rules), file = "<your file name>.csv")

* Prepare the XL file for Tableau (for example), by creating a new column “Rules”, the contents of this column is the contents of ***lhs column + => + rhs column***.
* Open your new XL file in Tableau and generate proper visualization.

The following is what I created. But you are free to create your own.

