Next Best Product

Next Best Offer

Next Best Action

# System recommender

1. Association rules (Apriori rules, F-P Growth)
2. Content-based filtering (Euclidean distance, Pearsons coefficient, Cosine similarity, TF-IDF, ..)
3. Collaborative filtering (Nearest neighborhood, Matrix Factorization, ..)
4. Kaggle competition – Santander product recommendation (Tom Van de Wiele)
5. Multiclassification (Xgboost, Neural Network, …)
6. Learning to Rank….

# Association rules

Association rule mining is a technique to identify underlying relations between different items. Take an example of a Super Market where customers can buy variety of items. Usually, there is a pattern in what the customers buy. For instance, mothers with babies buy baby products such as milk and diapers. Damsels may buy makeup items whereas bachelors may buy beers and chips etc. In short, transactions involve a pattern.

Apriori Algorithm for Association Rule Mining

There are three major components of Apriori algorithm:

* Support
* Confidence
* Lift

**Support**

Support refers to the default popularity of an item and can be calculated by finding number of transactions containing a particular item divided by total number of transactions. Suppose we want to find support for item B. This can be calculated as:

Support(B) = (Transactions containing (B))/(Total Transactions)

**Confidence**

Confidence refers to the likelihood that an item B is also bought if item A is bought. It can be calculated by finding the number of transactions where A and B are bought together, divided by total number of transactions where A is bought. Mathematically, it can be represented as:

Confidence(A→B) = (Transactions containing both (A and B))/(Transactions containing A)

**Lift**

Lift (A -> B) refers to the increase in the ratio of sale of B when A is sold. Lift(A –> B) can be calculated by dividing Confidence(A -> B) divided by Support(B). Mathematically it can be represented as:

Lift(A→B) = (Confidence (A→B))/(Support (B))

* If lift(A→B) = 1, then it would imply that probabilities of occurrences of itemset X and itemset Y are independent of each other, meaning that the rule doesn’t show any statistically proven relationship.
* If lift(A→B) > 1, then it would imply that probabilities of occurrences of the itemsets X and Y are positively dependent on each other. It will also tell us the magnitude of the level of dependence. The higher he lift value, the higher is the dependence, which can also be referred to as itemsets being complements to each other.
* If lift(A→B) < 1, then it would imply that the probabilities of occurrences of the itemsets X and Y are negatively dependent on each other. The lower the lift value, the lower is the dependence, which can also be referred to as itemsets being substitutes to each other.

Steps Involved in Apriori Algorithm

For large sets of data, there can be hundreds of items in hundreds of thousands transactions. The Apriori algorithm tries to extract rules for each possible combination of items. For instance, Lift can be calculated for item 1 and item 2, item 1 and item 3, item 1 and item 4 and then item 2 and item 3, item 2 and item 4 and then combinations of items e.g. item 1, item 2 and item 3; similarly item 1, item2, and item 4, and so on.

As you can see from the above example, this process can be extremely slow due to the number of combinations. To speed up the process, we need to perform the following steps:

1. Set a minimum value for support and confidence. This means that we are only interested in finding rules for the items that have certain default existence (e.g. support) and have a minimum value for co-occurrence with other items (e.g. confidence).
2. Extract all the subsets having higher value of support than minimum threshold.
3. Select all the rules from the subsets with confidence value higher than minimum threshold.
4. Order the rules by descending order of Lift.

F-P Growth

F-P Growth (or frequent-pattern growth) algorithm is another popular technique in Market Basket Analysis (first introduced by Han). It produces the same results as Apriori algorithm but is computationally faster due to a mathematically different technique (divide and conquer).

F-P Growth follows a two-step data preprocessing approach:

1. First, it counts the number of occurrences of each item in the transactional dataset.
2. Then, it creates a search-tree structure using the transactions.

Unlike Apriori algorithm, F-P Growth sorts items within each transaction by it’s frequency from largest to smallest before inserting it into a tree. This is where it has a substantial computational advantage over Apriori algorithm since it does the frequency sorting early on. Items which don’t meet minimum support (frequency) requirements (that we can set) are discarded from the tree.

Another advantage is that frequent itemsets that repeat will have the same path (unlike Apriori algorithm, where each itemset has a unique path).

Primer:



Literatura/Linkovi:

<https://pyshark.com/market-basket-analysis-using-association-rule-mining-in-python/>

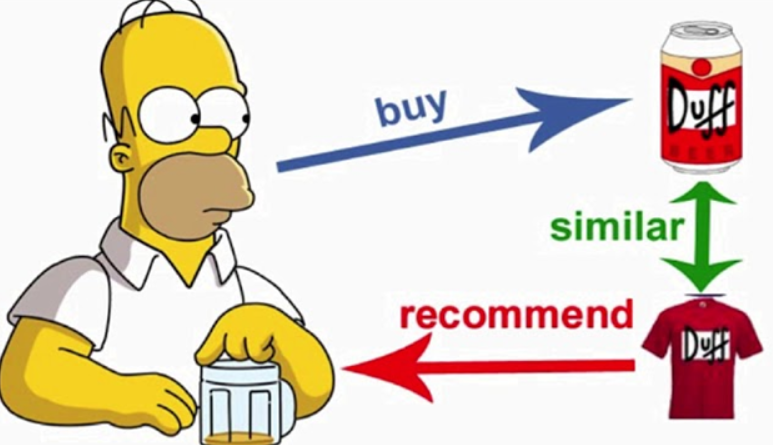
<https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/>

<https://medium.com/analytics-vidhya/association-analysis-in-python-2b955d0180c>

Agrawal, R.; Imieliński, T.; Swami, A. (1993). “Mining association rules between sets of items in large databases”.

Han (2000). *Mining Frequent Patterns Without Candidate Generation*.

# Content based filtering



A content-based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on those recommendations, the engine becomes more and more accurate.

A recommender system has to decide between two methods for information delivery when providing the user with recommendations:

* Exploitation. The system chooses documents similar to those for which the user has already expressed a preference.
* Exploration. The system chooses documents where the user profile does not provide evidence to predict the user’s reaction.

Disadvantages of Content Based Filtering

* **Limited content analysis**: If the content doesn’t contain enough information to discriminate the items precisely, the recommendation itself risks being imprecise.
* **Over-specialization:** Content-based filtering provides a limited degree of novelty, since it has to match up the features of a user’s profile with available items. In the case of item-based filtering, only item profiles are created and users are suggested items similar to what they rate or search for, instead of their past history. A perfect content-based filtering system may suggest nothing unexpected or surprising.

The model does not require any data about other users, since the recommendations are specific to one user. This makes it easier to scale it up to a large number of users.

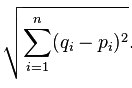
Recommending content involves making a prediction about how likely it is that a user is going to like the recommended content, buy an item or watch a movie.

There is a large amount of methods and literature available on recommender systems. Popular methods include:

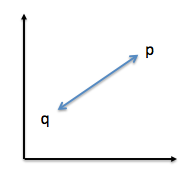
* [**Similarity-based Methods**](https://en.wikipedia.org/wiki/Similarity_measure)**(Eucledean distance, Cosine similarity, Pearson’s coefficient, TF-IDF, ….)**
* [One-class SVMs](https://en.wikipedia.org/wiki/One-class_classification)
* [Matrix Factorisation](https://en.wikipedia.org/wiki/Matrix_decomposition)
* [Supervised Learning](https://en.wikipedia.org/wiki/Supervised_learning)
* [Deep Learning](https://en.wikipedia.org/wiki/Deep_learning)

**Euclidean distance**

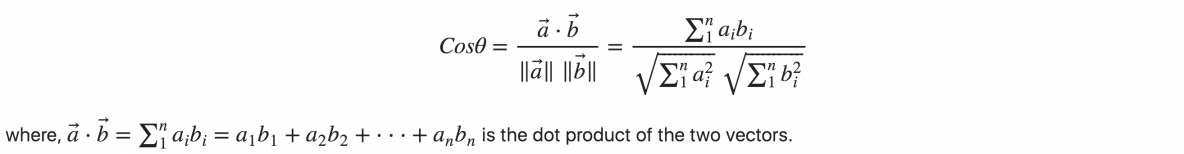
It is just a distance measure between a pair of samples p and q in an n-dimensional feature space:



For example, picture it as a “straight, connecting” line in a 2D feature space:



**Cosine similarity**



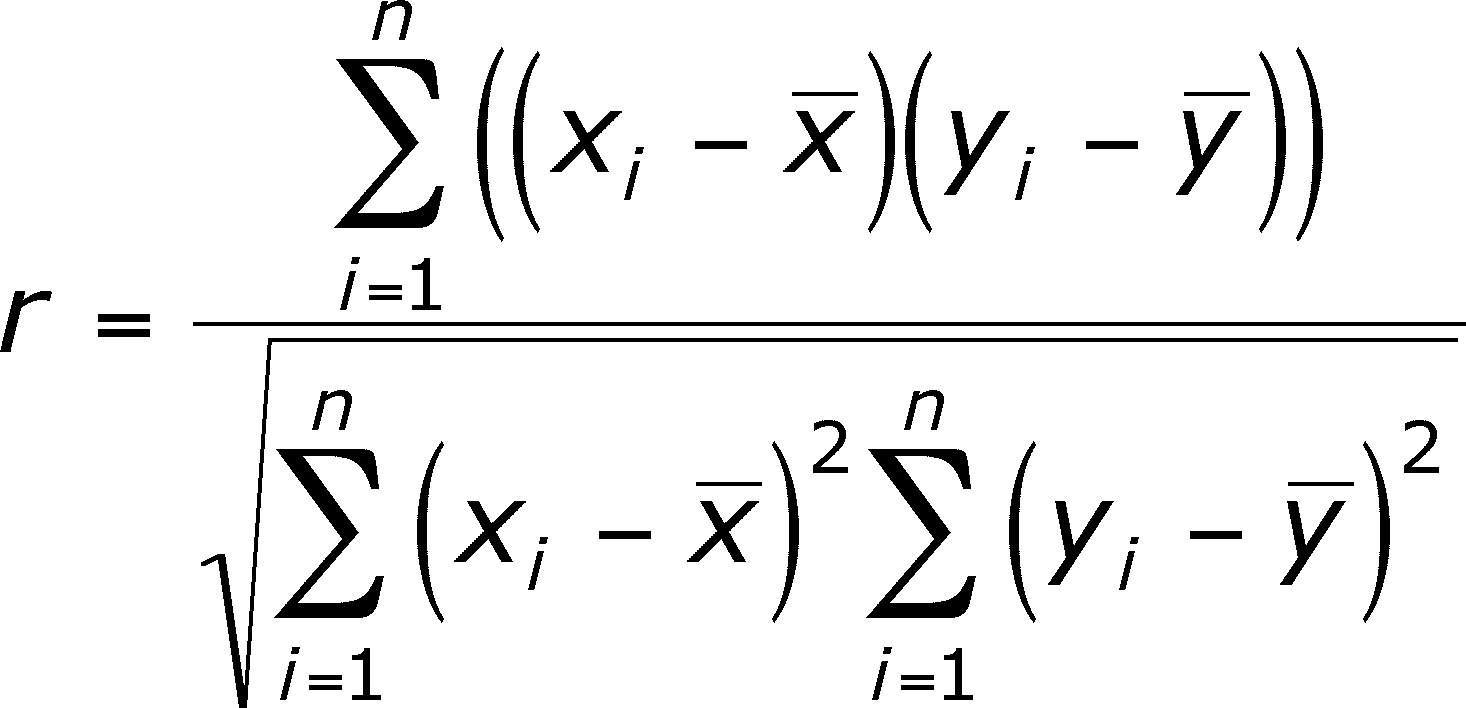
**Pearson’s coefficient**

Pearson correlation coefficient or Pearson’s correlation coefficient or Pearson’s r is defined in statistics as the measurement of the strength of the relationship between two variables and their association with each other.

In simple words, Pearson’s correlation coefficient calculates the effect of change in one variable when the other variable changes.

For example: Up till a certain age, (in most cases) a child’s height will keep increasing as his/her age increases. Of course, his/her growth depends upon various factors like genes, location, diet, lifestyle, etc.

This approach is based on covariance and thus is the best method to measure the relationship between two variables.



**TF – IDF**

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

It has many uses, most importantly in automated [text analysis](http://www.monkeylearn.com/text-analysis/), and is very useful for scoring words in machine learning algorithms for [Natural Language Processing](https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP).

TF-IDF (term frequency-inverse document frequency) was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don’t mean much to that document in particular.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

* The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.
* The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.
* So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

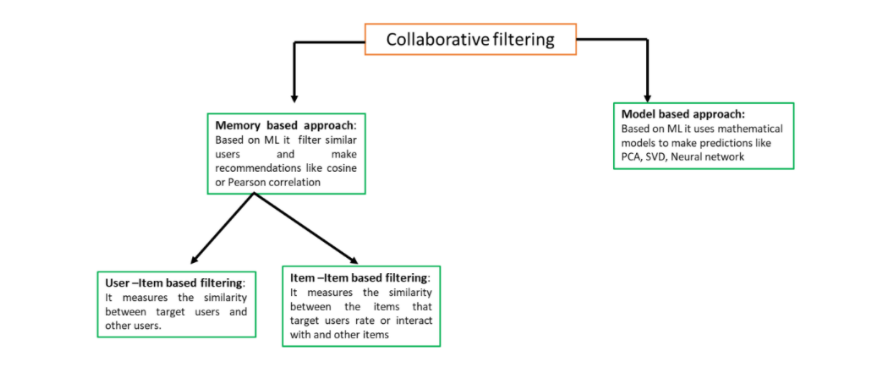
Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

<http://recommender-systems.org/content-based-filtering/>

<https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831>

# Collaborative filtering

This filtering method is usually based on collecting and analyzing information on user’s behaviors, their activities or preferences and predicting what they will like based on the similarity with other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and thus it is capable of accurately recommending complex items such as movies without requiring an “understanding” of the item itself. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. For example, if a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.



Further, there are several types of collaborative filtering algorithms:

* **User-User Collaborative Filtering:** Here, we try to search for lookalike customers and offer products based on what his/her lookalike has chosen. This algorithm is very effective but takes a lot of time and resources. This type of filtering requires computing every customer pair information which takes time. So, for big base platforms, this algorithm is hard to put in place.

<https://www.geeksforgeeks.org/user-based-collaborative-filtering/>

* **Item-Item Collaborative Filtering:** It is very similar to the previous algorithm, but instead of finding a customer look alike, we try finding item look alike. Once we have item look alike matrix, we can easily recommend alike items to a customer who has purchased any item from the store. This algorithm requires far fewer resources than user-user collaborative filtering. Hence, for a new customer, the algorithm takes far lesser time than user-user collaborate as we don’t need all similarity scores between customers. Amazon uses this approach in its recommendation engine to show related products which boost sales.

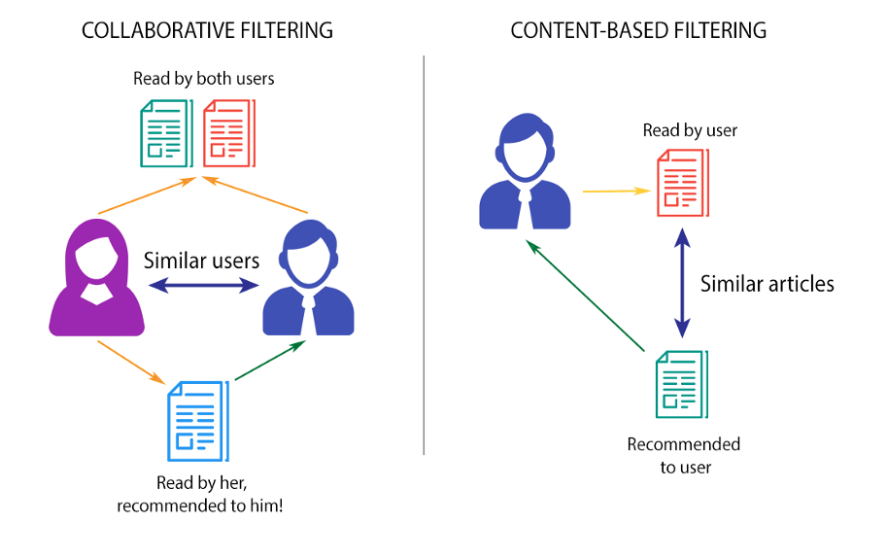
<https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/>

Advantages of collaborative filtering

* There is no dependence on domain knowledge as embedding are automatically learned.
* The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.
* To some extent, the system needs only the feedback matrix to train a matrix factorization model. In particular, the system does not require contextual features.

#### Disadvantages of collaborative filtering

* The matrix cannot handle fresh items, for instance, if a new car is added to the matrix, it may have limited user interaction and thus, will rarely occur as a recommendation.
* The output of the recommendation could be biased, based on popularity, that is, if most user interaction is towards a particular car, then the recommendation will focus on that popular car only.



<https://beckernick.github.io/matrix-factorization-recommender/>

<http://www.quuxlabs.com/wp-content/uploads/2010/09/mf.py_.txt>

<https://towardsdatascience.com/what-are-product-recommendation-engines-and-the-various-versions-of-them-9dcab4ee26d5>

<https://buomsoo-kim.github.io/recommender%20systems/2020/09/25/Recommender-systems-collab-filtering-12.md/>

<https://sifter.org/~simon/journal/20061211.html>

<https://netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf>

<https://www.youtube.com/watch?v=giIXNoiqO_U&list=PL3ZVX5cUMdLbiFgitZszhnMUZHDDEL0rS&ab_channel=ArtificialIntelligence-AllinOne>

<https://www.digitalvidya.com/blog/collaborative-filtering/>

<https://www.mygreatlearning.com/blog/matrix-factorization-explained/> !!!!!!!!!

# Kaggle competition – Santander product recommendation (Tom Van de Wiele)

<https://ttvand.github.io/Second-place-in-the-Santander-product-Recommendation-Kaggle-competition/>