## **What is an Association Rule Learning Algorithm?**

**The association rules are rule based machine learning approach to find the pattern in sale of item with other.It is usually used in large database to find interesting relationship in how and why these items are related.**

The **association rule learning algorithm** is a rule-based machine learning approach to find patterns from items that are dependent on one another and map the connections between them. It is usually used in a large database to find interesting relationships in how and why two items are connected. Association rule learning algorithm finds application in many real-life scenarios.

## **Why is Association Rule Learning Algorithm important?**

The association rules are generally useful to predict purchasing behavior and relationship among the items in the database.It is usually used for classification and discovering patterns.

The association rule learning algorithm is particularly useful in **predicting behaviors and relationships between variables in a dataset**. It is useful for classification and discovering patterns within data. It also helps to find patterns that explain the correlation between features in a dataset.

## **How do Association Rule Learning Algorithms work?**

Association rule learning algorithms work like conditional statements (ex, if A then B). In this case, A is called the **antecedent** while B is called the **consequent**.

A, or the antecedent can be an item in your data while B is the result of the combination of antecedent(s). B could take the form of an action, such as signaling a customer is likely to be a repeat buyer, or B could be another item in the dataset.

**Support** is the number of times an **item A and B** sold in combination to total number of transaction .

**Confidence:** Out of the transactions that contains **item A and B** , how many also contains **item B**. The bigger the overlap, the greater the confidence we have that people who are buying item A also buys item B.. While **Lift** is used to compare the number of times a rule was supposed to be obeyed to the number of times it actually obeyed.

#### What is the Apriori algorithm?

Apriori is an algorithm that finds all frequent items set in a dataset. **It finds items that are frequently transacted together whose support and confidence are above the minimum threshold.** In scenarios where there are so many items, Apriori helps with defining the rules for these items.

#### What’s the difference between Apriori and FP Growth?

The future pattern (FP) growth algorithm tries to find the most frequent itemset using a depth-first tree method while Apriori uses a breath first and hash tree approach. It’s also much faster than Apriori which looks at the data one at a time.

#### What algorithms can be classified under Association Rule Learning Algorithms?

* [Apriori algorithm](https://www.geeksforgeeks.org/apriori-algorithm/): Used to generate association rules
* [Eclat](https://towardsdatascience.com/the-eclat-algorithm-8ae3276d2d17#:~:text=Eclat%20stands%20for%20Equivalence%20Class,also%20regroups%20frequent%20itemset%20mining).) (Equivalent Class Transformation): Uses the current generated itemset to learn frequent itemset in the data.
* [FP Growth](https://www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/https:/www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/): Frequent Pattern algorithms aim to extract frequent patterns from the data set using a tree-based approach.

#### What are confidence, support, and lift?

* **Support** shows how frequently an item appears in a dataset and it’s useful in understanding how items connect to the whole dataset.   
    
  For a simple rule where A implies B, it can simply be calculated as the total number A and B to the total number of transactions altogether.

A picture containing diagram

Description automatically generated

### Support:

Within a dataset, i.e. a list of transactions, how many transactions contain **item A**, so it is just the probability of **item A** occurring, which we can represent as below. Statistically speaking, it is a frequentist's estimate of the probability.

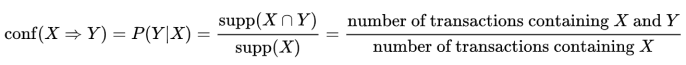
A picture containing text, athletic game, sport

Description automatically generated

### Confidence:

Out of the transactions that contains **item A**, how many also contains **item B**. The bigger the overlap, the greater the confidence we have that people who are buying item A also buys item B.  Statistically speaking, it is (estimated) conditional probably of **item B** given **item A**, i.e. **P(B|A)**.

* **Confidence** is a bit different from support in that it finds how an item connects to other items in the dataset. It is simply trying to filter out how many times another item B occurs when Item A has occurred. It works like conditional probability and it’s usually expressed in percentages.  
    
  In a rule where A implies B, it can be calculated as the ratio of the number of transactions containing A and B to the total number of transactions containing A alone.



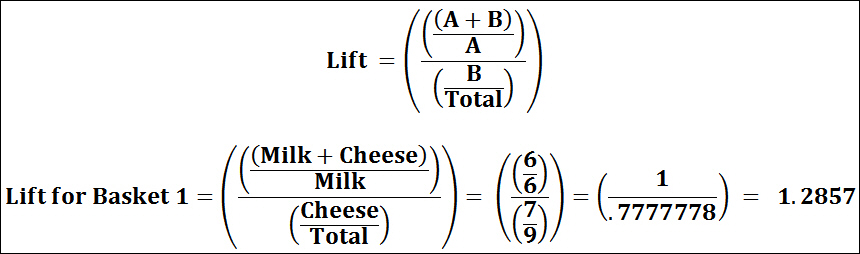
* A picture containing text, tennis, athletic game, racket

  Description automatically generated

**Lift:**

The ratio between C**onfidence of A** and S**upport B**, it is less intuitive with the description, so let's try to visualize it better. First let's see the formula below.

* Graphical user interface

  Description automatically generated
* 
* **Lift** tells us the strength of a rule over every random choice.   
    
  For a rule where A implies B, it is calculated as the support of A and B to the product of the support of A and the support of B. If the lift has a value less than 1, it means that one item is only a substitute for the other and that they can’t be bought together. If the lift has a value greater than 1, it shows you the extent to which one item depends on the other.

Text

Description automatically generated with medium confidence

#### What is market basket analysis?

Market basket analysis is a technique used by companies to find associations between products so that they can generate more revenue by presenting various related products to the customer

#### What is a frequent itemset?

Frequent itemset is a set of items whose support and confidence obey the minimum threshold rules.

#### What is the importance of pruning?

Pruning helps to remove rules that are below the minimum threshold you set. It helps you filter out irrelevant rules.

#### What are some other applications of association rule learning algorithms?

The algorithms are widely used in web data mining, intrusion detection, continuous production, and bioinformatics.

What are limitations of association rules?

* Setting the parameter and threshold is a challenge
* It can involve finding too many rules, of which some are irrelevant and thus lead to low / lower performance of the model.

The difference between association rule and sequence mining

The difference between association rule and sequence mining is that the association rule does not consider the order (or arrangement) of the information, unlike sequence mining where order matters.

What is drawback of apriori algorithm?

Suppose In a transaction of 100,if milk is 80 times occurs and tea 90 times ,support can only calculate the count but do not consider the dependency of each other

What is true dependent rule

Select one:

a. Lift = 1

b. Lift >1

c. Lift <1

d. Lift = 0.5

#### **Feedback**

Your answer is incorrect.

The correct answer is: Lift >1

Recommendation systems

Recommender systems are the systems that are designed to recommend things to the user based on many different factors.

* Content-based systems which use Characteristic information: This is information about items (keywords, categories, etc.) and users (preferences, profiles, etc.).
* They hypothesize that if a user was interested in an item in the past, they will once again be interested in it in the future. Similar items are usually grouped based on their features. User profiles are constructed using historical interactions or by explicitly asking users about their interests.
* Collaborative filtering systems, which are based on user-item interactions.
* In short, collaborative filtering systems are based on the assumption that if a user likes item A and another user likes the same item A as well as another item, item B, the first user could also be interested in the second item. Hence, they aim to predict new interactions based on historical ones. There are two types of methods to achieve this goal: memory-based and model-based.
* A **Recommendation System** is a subclass of *information filtering system* that seeks to predict the *rating* or *preference* a user would give to an item.
* **Collaborative** methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations.
* Unlike *collaborative* methods, **content-based** approaches use additional information about users and/or items.
* **Model-based Collaborative** approaches only rely on *user-item* interactions information and assume a *latent model* to explain these interactions.
* For example, *matrix factorization* algorithms consist of decomposing the huge and sparse user-item interaction matrix into a product of two smaller and dense matrices: a user-factor matrix (containing users representations) that multiplies a factor-item matrix (containing items representations).

## What are some Domain-Specific Challenges in Recommender Systems?

* In different domains, such as *temporal data*, *location-based data*, and *social data*, the context of the recommendation plays a critical role. The notion of **contextual recommender systems** was developed to address the additional side information that arises in these domains.
* This notion is used with different modifications for various types of data.
* Some *domain-specific* applications are *context-based recommender systems*, *time-sensitive recommender systems*, *location-based recommender systems*, and *social recommender systems*.

## What are the basic components of a Content-Based System?

* **Preprocessing and feature extraction:** Content-based systems are used in a wide variety of domains, such as Web pages, product descriptions, news, music features, and so on. In most cases, features are extracted from these various sources to convert them into a keyword-based vector-space representation.
* **Content-based learning of user profiles:** A content-based model is specific to a given user. Therefore, a user-specific model is constructed to predict user interests in items, based on their history of either buying or rating items.
* **Filtering and recommendation:** In this step, the learned model from the previous steps is used to make recommendations on items for specific users. This step needs to be very efficient because the predictions need to be performed in real-time.

## How are Knowledge-based Recommender Systems different from Collaborative and Content-based Recommender Systems?

* The recommendations of **content-based** and **collaborative systems** are primarily based on **historical data**.
* The recommendations of **knowledge-based** systems are based on the direct specifications by users of *what they want*.
* An important distinguishing characteristic of **knowledge-based systems** is the high level of *customization* to the specific domain. This customization is achieved through the use of a *knowledge-base* that encodes relevant domain knowledge in the form of either constraints or similarity metrics.

**1.Recommendation engine :** memory based or lazy learning

**2.Computational intensive**

**How to reduce the computational complexity?**

1.randomly sample the customers

2.Discard the infrequent buyers

3.Discard the very popular items

4clustering

5.PCA

**Runtime vs Quality recommendation**

**Recommend while the customers are buying**

**Recommend better but later after processing the need**

**Item to Item collaborative filtering**

Similar item tables are computed offline.

Recommendation is provided only those customers who had purchased items of same category earlier.

Cross selling is not possible

**Amazon uses which recommendation system?**

**Incorporating Implicit Feedback**

Item to item collaborative filtering

Various frameworks such as *asymmetric factor models* and *SVD*++ have been proposed to

incorporate implicit feedback.

Netflix uses collaborative filtering algorithms,They will ask questions and ratings to some movies.Netflix uses SVD to reduce computational complexity.

The Google News personalization system [697] is able to recommend news to users based on

their history of clicks.

Facebook uses

the recommendation of potential friends (or *links*) enables better growth and connectivity of the network. This problem is also referred to as *link prediction* in the field of social network analysis. Such forms of

recommendations are based on *structural relationships* rather than ratings data.

**Difference between Association rules and recommendation systems**

|  |  |
| --- | --- |
| **Association rules** | **Recommendation system** |
| **What is getting purchased** | **What and who is purchasing** |
| **Useful for large physical store** | **Useful for e commerce** |
| **No.of transactions are important** | **No.of transactions are not important** |

a) How long time does it take too implement recommendation system?

The process of finding suitable features is relatively slow, especially when it comes to images, videos, or music. Because there is no way to build a user profile, new users have a difficult time receiving recommendations. -Overspecialization – do not recommend items from outside the organization.

## **How Do You Create A Recommendation System?**

We can create a recommendation system by analyzing popularity data over all popular products, so here’s how to identify the most popular goods by the products most purchased: In shopping, for example we can suggest popular dresses based on the amount of purchases.

## **Who Has The Best Recommendation Engine?**

 This Netflix recommendation system is among the best out there, for sure…

 With its selection of best, and one of the only ones that I am aware of, Amazon is a pioneer in integrating recommendation systems into e-commerce.

## **Do Recommendation Engines Work?**

An enhanced user experience can be obtained by using recommendation engines for products. With the aid of advanced algorithms such as machine learning and artificial intelligence, a recommendation system can assist customers in finding the products they need.

## **How Do I Create A Content Recommendation System?**

In the model, the title and description suggest a similar book. Using cosine similarity, make a comparison of the different literatures. Based on the book title and the title and description of the book, this function returns the top five similarly recommended books.

## **Which Algorithm Is Best For Recommender System?**

You might have heard that collaborative filtering is a popular recommendation algorithm. It is also convenient for anyone who is just beginning to work with data to build their own recommendations for movies, for example, for resume writing.

## **How Are Recommender Systems Built?**

based on preferences and profiles of the users are able to make recommendations. In other words, they can try to match customers to products they like. For the purpose of judging similarity levels between items, factors such as similarities and dislikes of items are commonly taken into account.

## **How Do You Write A Simple Recommendation System?**

 The idea is to suggest similar items that may appeal to users who were previously interested and interacted with them.

 An item’s rating on the online rating system is used as an indicator of whether or not a user will be able to like it.

## **How Do You Make A Recommendation Engine?**

 A recommendation engine must start with collecting data….

 …Data storage and its importance in determining how good a recommendation could be.

 Data filtering is done

## **What Is The Best Recommendation System?**

I present the following four popular ones: Surprise: A Python Scikit framework that builds recommender databases. In Implicit, a Python collaborative filtering engine can be quickly used to filter data based on oblicants. provides feedback both implicit and explicit based on algorithms widely used in Python.

## **How Do Recommendation Systems Work?**

Using their knowledge of each product, content-based recommendation systems suggest a new product based on their knowledge of that particular product. Items that earn favorable recommendations are classified according to their attributes. In content-based recommender systems, descriptive info of the content must be disclosed before action can be performed. Customer affinity is evaluated as a measure of similarity.

## **What Are The Benefits Of Recommendation Engines?**

 Do whatever it takes to drive traffic.

 Relevant content should be provided.

 Shoppers engage with you….

 Become an expert at persuading shoppers to become clients.

 How to increase the average order value…

 You might increase order minimum requirements by more than one item…

 Take control of merchandising and inventory rules…

b) Is it possible to place the recommendations any where in the customer flow

In the context of Recommender systems. When customer data is about is the Rating matrix for the various products, for similarity measure between the customers, it is required to ignore products where one of the pair of the customers has not rated the product. What aspect of Data preprocessing is being addressed by this?

Select one:

a. Pair wise deletion

Pairwise deletion or Avalablbe case analysis is a method of keeping the missingness as is, till the point where the computation/comparison is not possible and only for those cases the missingness is deleted along with the complementing record(the other record with which the record with missingness is being compared/studied/distance computed) In a database with user ratings of products/services there bound to be a large number of missing values and absence of rating does not imply a '0' rating. therefore the pair-wise deletion is used wherein when a pair is being considered for distance/similarity measure only those products (columns) that have non-nan values for both the pairs are included in the measure and the rest are excluded.

b. Case wise deletion

c. Hot deck Imputation

d. Mode imputation

#### **Feedback**

Your answer is correct.

The correct answer is: Pair wise deletion

**Analyzing the Retention Stage**

When analyzing the retention phase, you'll want to surface where you can make your customer's experience better, so that they stay with you for longer.

You'll want to answer these questions:

* How do your customers feel about your business?
* Have you made it easy for customers to do business with you again? For example, do you include a one-click reorder button in your user portal?

**Evaluating Recommenders**

The evaluation of the recommender systems is another important step in order to

assess the effectiveness of the method. When dealing with numerical labels, as the

5-star ratings, the most common way to validate a recommender system is based

on their prediction value, i.e., the capacity to predict the user’s choices. Standard

functions such as *root mean square error* (RMSE), *precision*, *recall*, or *ROC/cost*

curves have been extensively used.

A **graph** is a common data structure that consists of a finite set of **nodes** (or **vertices**) and a set of **edges** connecting them. A pair (x,y) is referred to as an edge, which communicates that the **x vertex** connects to the **y vertex**.

Graphs are used to solve real-life problems that involve representation of the problem space as a **network**. Examples of networks include telephone networks, circuit networks, social networks (like LinkedIn, Facebook etc.).

**Graph:**

* Consists of a set of vertices (or nodes) and a set of edges connecting some or all of them
* Any edge can connect any two vertices that aren't already connected by an identical edge (in the same direction, in the case of a directed graph)
* Doesn't have to be connected (the edges don't have to connect all vertices together): a single graph can consist of a few disconnected sets of vertices
* Could be directed or undirected (which would apply to all edges in the graph)

**Tree:**

* A type of graph (fit with in the category of Directed Acyclic Graphs (or a DAG))
* Vertices are more commonly called "nodes"
* Edges are directed and represent an "is child of" (or "is parent of") relationship
* Each node (except the root node) has exactly one parent (and zero or more children)
* Has exactly one "root" node (if the tree has at least one node), which is a node without a parent
* Has to be connected

# Explain the BSF (Breadth First Search) traversing method

Breadth First Search (BFS) is the traversing method used in graphs. It uses a queue for storing the visited vertices. In this method the emphasize is on the vertices of the graph, one vertex is selected at first then it is visited and marked. The vertices adjacent to the visited vertex are then visited and stored in the queue sequentially. A node is fully explored before visiting any other node in the graph, in other words, it traverses shallowest unexplored nodes first.

The BST algorithm works as follows:

* Start by putting any one of the graph's vertices at the back of a queue.
* Take the front item of the queue and add it to the visited list.
* Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
* Keep repeating steps 2 and 3 until the queue is empty.

BST example step-by-step:

We have a graph whose vertices are A, B, C, D, E, F, G. Considering A as starting point. The steps involved in the process are:

* Vertex A is expanded and stored in the queue.
* Vertices B, D and G successors of A, are expanded and stored in the queue meanwhile Vertex A removed.
* Now B at the front end of the queue is removed along with storing its successor vertices E and F.
* Vertex D is at the front end of the queue is removed, and its connected node F is already visited.
* Vertex G is removed from the queue, and it has successor E which is already visited.
* Now E and F are removed from the queue, and its successor vertex C is traversed and stored in the queue.
* At last C is also removed and the queue is empty which means we are done.
* The generated Output is – A, B, D, G, E, F, C.

# What is difference between BFS and Dijkstra's algorithms when looking for shortest path?

What is the difference between Dijkstra's and BFS? And then why are the time complexities of these algorithms so different?

* Breadth-first search is just Dijkstra's algorithm with all edge weights equal to 1.
  + BFS basically just expands the search by one “step” (link, edge, whatever you want to call it in your application) on every iteration, which happens to have the effect of finding the smallest number of steps it takes to get to any given node from your source (“root”)
  + Breadth-first search can be viewed as a special-case of Dijkstra's algorithm on unweighted graphs, where the priority queue degenerates into a FIFO queue.
  + Operations on a regular queue are O(1).
  + BFS runs in O(E+V).
* Dijkstra's algorithm is conceptually breadth-first search that *respects* edge costs.
  + For example, in routing the distances (or weights) could be assigned by speed, cost, preference, etc.
  + Dijkstra's uses a priority queue data structure to keep track of the frontier of unvisited nodes.
  + Operations on a priority queue are O(log n).
  + Dijkstra's runs in O((V+E)\*log(V))

The process for exploring the graph is structurally the same in both cases.

**Depth-first search** (DFS) is a method for exploring a tree or graph. In a DFS, you go as deep as possible down one path before backing up and trying a different one.

Depth-first search is like walking through a corn maze. You explore one path, hit a dead end, and go back and try a different one.

Here's a how a DFS would traverse this tree, starting with the root:

We'd go down the first path we find until we hit a dead end:

Then we'd do the same thing again—go down a path until we hit a dead end:

And again:

And again:

Until we reach the end.

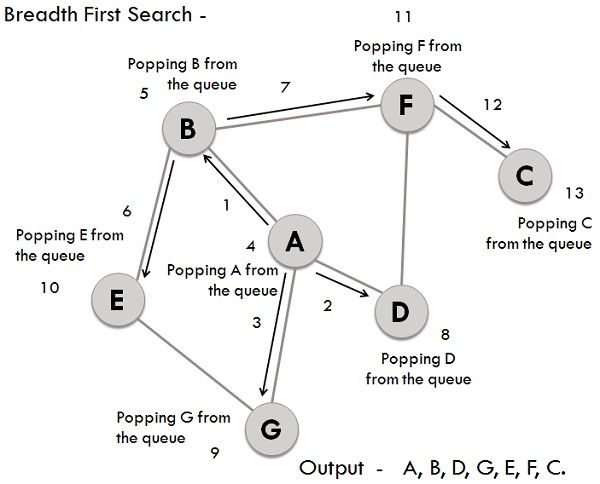
Depth-first search is often compared with **breadth-first search**.

Advantages:

* Depth-first search on a binary tree *generally* requires less memory than breadth-first.
* Depth-first search can be easily implemented with recursion.

Disadvantages

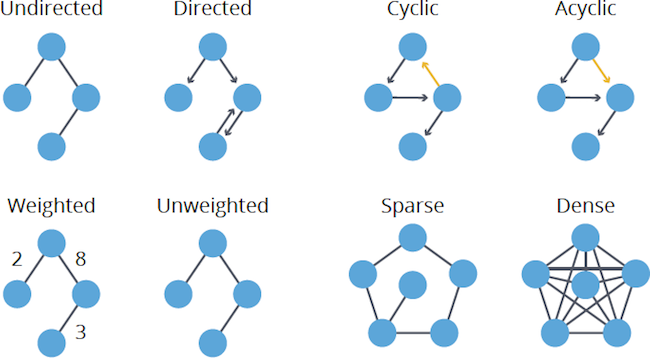
* A DFS doesn't necessarily find the shortest path to a node, while breadth-first search does.



c) How long time will it take before a newly added product appear among the recommended products  
d) How does the system monitor the visitor?  
e) Am I allowed to splittest against competing alternatives?  
f) Where does my data end up? (some companies stores the data forever)  
g) How fast can I get out of my contract? The industry is changing at a crazy rate at the moment and you have to keep up?  
h) What analytics software will I have to use to monitor the effect of the system.

# Name some common types and categories of Graphs

* In an undirected graph, nodes are connected by edges that are all *bidirectional*. For example if an edge connects node 1 and 2, we can traverse from node 1 to node 2, and from node 2 to 1.
* In a directed graph, nodes are connected by directed edges – they only go in *one direction*. For example, if an edge connects node 1 and 2, but the arrow head points towards 2, we can only traverse from node 1 to node 2 – not in the opposite direction.
* A weight is a numerical value attached to each individual edge. If edges in our graph have weights then your graph is said to be a weighted graph
* if the edges do not have weights, the graph is said to be unweighted
* A cyclic graph is a directed graph which contains a path from at least one node back to itself. In simple terms cyclic graphs contain a cycle.
* An acyclic graph is a directed graph which contains absolutely no cycle, that is no node can be traversed back to itself.
* Dense graph is a graph in which the number of edges is close to the *maximal (max)* number of edges.
* Sparse graph is a graph in which the number of edges is close to the *minimal (min)* number of edges. Sparse graph can be a disconnected graph.



* **Definition:** Degree centrality assigns an importance score based simply on the number of links held by each node.
* **What it tells us:** How many direct, ‘one hop’ connections each node has to other nodes in the network.

**Advanced graph algorithms**

If you have lots of time before your interview, these advanced graph algorithms pop up occasionally:

* **Dijkstra's Algorithm:** Finds the shortest path from one node to all other nodes in a *weighted* graph.
* **Topological Sort:** Arranges the nodes in a *directed*, *acyclic* graph in a special order based on incoming edges.
* **Minimum Spanning Tree:** Finds the cheapest set of edges needed to reach all nodes in a *weighted* graph.
* **Definition:** Betweenness centrality measures the number of times a node lies on the shortest path between other nodes.
* **What it tells us:** This measure shows which nodes are ‘bridges’ between nodes in a network. It does this by identifying all the shortest paths and then counting how many times each node falls on one.
* **When to use it:** For finding the individuals who influence the flow around a system.
* **Definition:** Closeness centrality scores each node based on their ‘closeness’ to all other nodes in the network.
* **What it tells us:** This measure calculates the shortest paths between all nodes, then assigns each node a score based on its sum of shortest paths.
* **When to use it:** For finding the individuals who are best placed to influence the entire network most quickly.
* **Definition:** Like degree centrality, EigenCentrality measures a node’s influence based on the number of links it has to other nodes in the network. EigenCentrality then goes a step further by also taking into account how well connected a node is, and how many links their connections have, and so on through the network.
* **What it tells us:** By calculating the extended connections of a node, EigenCentrality can identify nodes with influence over the whole network, not just those directly connected to it.
* **When to use it:** EigenCentrality is a good ‘all-round’ SNA score, handy for understanding human social networks, but also for understanding networks like malware propagation.
* **Definition:** PageRank is a variant of EigenCentrality, also assigning nodes a score based on their connections, and their connections’ connections. The difference is that PageRank also takes link direction and weight into account – so links can only pass influence in one direction, and pass different amounts of influence.
* **What it tells us:** This measure uncovers nodes whose influence extends beyond their direct connections into the wider network.
* **When to use it:** Because it takes into account direction and connection weight, PageRank can be helpful for understanding citations and authority.