



OUT OF STOCK PRODUCT RECOMMENDER

Solving the Peanut Butter Problem

WHITE PAPER | JULY 2020



BLUEGRANITE

Table of Contents

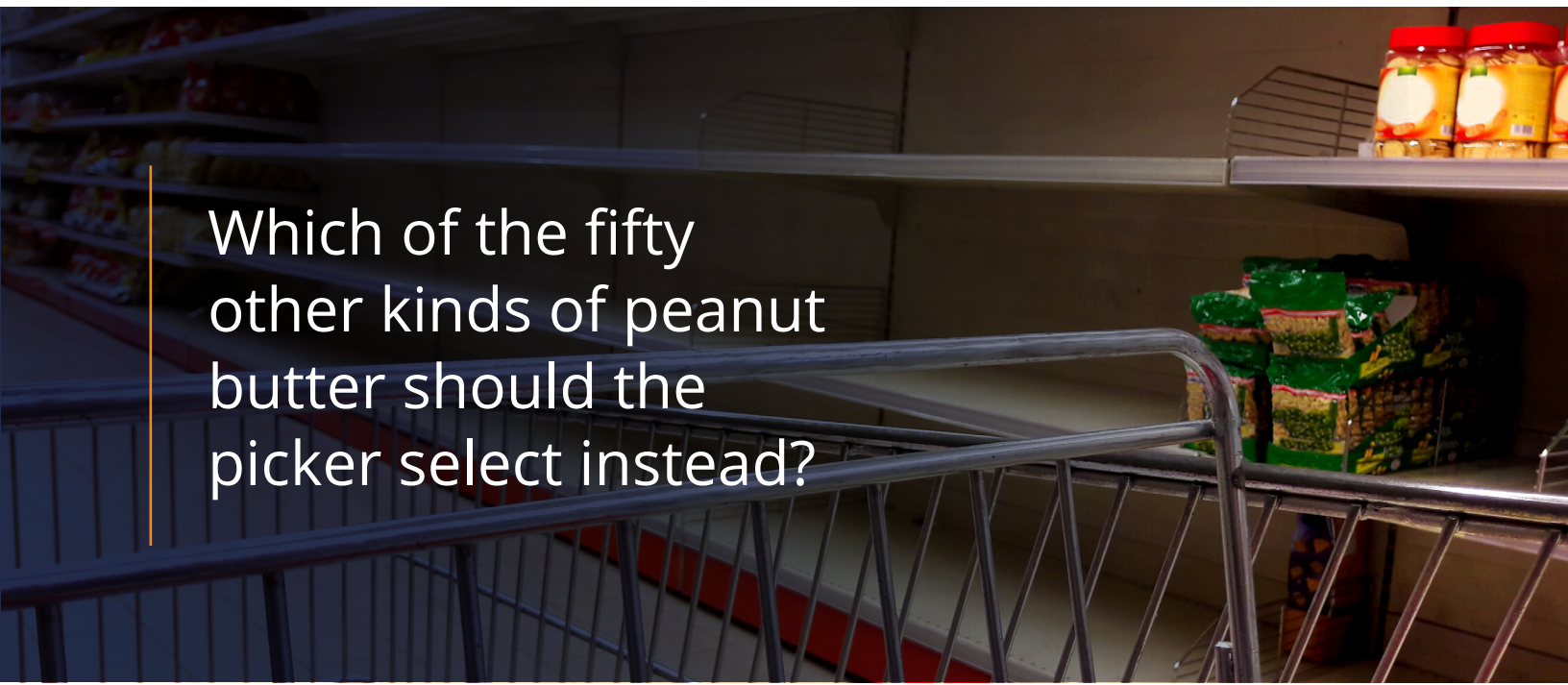
Executive Summary	3
Section 1: High Demand's Strain on E-Commerce	4
Section 2: Technical Challenges in Supply Chain Disruptions	5
2.1: Large Stores, Small Carts.....	5
Section 3: Identifying Substitute Products	6
3.1 Combining the Models	6
3.2 Model Enhancements.....	7
Section 4: How Does the Business Benefit	8
Section 5: Solution Architecture	9
6.1 Categorical Data	9
6.2 Text as Data	10
6.3 Continuous Data	11
6.4 Missing Data	11
Section 7: Customer Segmented Recommender	12
7.1 Customer Segmentation	13
7.2 Custom Weights	15
Section 8: Incorporating Continuous Customer Feedback	16
8.1 The Adjustment Process.....	16
8.2 Adjusting Weights vs. Match Scores	17
8.3 Future Implications	17
Section 9: Partner with BlueGranite.....	18

Executive Summary

The peanut butter problem occurs when a customer orders a specific kind of peanut butter to be delivered in a grocery order, but the picker fulfilling the order sees that it's not on the shelf. Which of the fifty other kinds of peanut butter should the picker select instead?

This paper lays out a framework for substituting out of stock items from the perspective of a retail store or order fulfillment platform. We first define the problem from a business perspective and argue for how this solution helps a store's bottom line. Then we propose a design for building a recommendation engine that uses available data to estimate a percent match between all products within a store. Finally, we propose an architecture for hosting the product data and recommender using a modern data warehouse.

Our proof of concept shows that the recommendation engine successfully identifies meaningful product substitutions. The design is robust to work with limited data and flexible to allow for customization at both the store level and customer level.



Which of the fifty other kinds of peanut butter should the picker select instead?

Section 1:

High Demand's Strain on E-Commerce

For grocery store chains, one of the most dramatic effects of the COVID-19 pandemic has been the sudden and massive demand for e-commerce. This demand grew out of the calls for social distancing and government regulations, pushing grocery shoppers to go online. Additional online grocery demand came from a \$100B shift from restaurants to grocery.

UNFI's Chairman and CEO, Steve Spinner, feels that this trend will continue despite some restaurants beginning to open at a lower capacity. Spinner says that consumers are still hesitant to eat out due to safety concerns, and the recession and high unemployment rate will drive sales of at-home dining for at least a year.

As grocery customers moved online, they began pushing retailers to devise ways to make the online shopping experience faster and more efficient. Recently [PWC shared insights from a survey](#) they conducted with 1,600 consumers, indicating that the majority of shoppers (64%) will continue their current rate of pantry stocking for the foreseeable future. The demand for online grocery shopping will be here to stay for some time.

The demand for online grocer shopping will be here to stay for some time.

Shifting the shopper experience online has been a bit clunky for most grocers. One of the most challenging aspects of translating an in-store experience to an online experience centers around product and inventory out of stocks. When a shopper's favorite product is out of stock, it's often not communicated to the customer until after the order is sent to be filled. While in-store, customers can see for themselves whether something they want is on the shelf, but conveying

accurate stock levels to online customers is deceptively challenging. Grocers have placed a priority on getting reliable information to customers about what is available. Determining what to offer as a substitution based on the customer's personal preferences across brand loyalty, price sensitivity, or more nuanced dimensions like dietary aversions, is very challenging and has the potential to add friction to the purchasing path.

Stores have spent decades and massive fortunes on optimizing product placement to help delight customers and influence choice. Now grocers must figure out how to give the online shopper the same warm feeling of exploration and discovery, all without overwhelming them with decisions that might trigger a cart abandonment.

The remainder of this white paper outlines one tactical solution for helping customers find preferred product replacements for out of stock items. In return, grocers will be provided with an opportunity to enrich the customer profile with a refined pattern of directional purchasing preferences.

Section 2:

Technical Challenges in Supply Chain Disruptions

The ideal solution to the stockout problem is to simply eliminate the problem in the first place, but that is easier said than done. Despite strong financial incentives and many researchers and business leaders trying to solve the out of stock problem, a [report by Gruen and Corsten](#) (2008, p. 7) point out that there has been no measurable improvement in the preceding decades. A number of out-of-stock rates will always have to be tolerated because the labor cost of continuously restocking will at some point outweigh the benefits. Furthermore, a retail store cannot fully mitigate upstream shortages in the supply chain that can occur from disruptions in areas like resource procurement, manufacturing, transportation or competing demand. Therefore, a practical solution should focus on reducing the harm from stockouts.

2.1: LARGE STORES, SMALL CARTS

It is not uncommon for a retailer to have 100,000 SKUs or more. One of the most prominent problems with modeling shopping behavior for such a retailer is the issue of high dimensionality. This arises from scenarios where a typical grocer has a large variety of products, but a typical shopper only selects a small number of products to purchase. If a shopper buys less than 1% of all items in a store, then a model will be over 99% accurate if it predicts the shopper never buys any individual item.

If a shopper buys less than 1% of all items in a store, then a model will be over 99% accurate if it predicts the shopper never buys any individual item.

Similarly, if a preferred product is out of stock, what will a customer purchase as a substitute? Such a question makes the challenge exponentially harder because now we are considering all product pairs. Consider the [regional grocery retailer, Meijer](#): it reports each typical supercenter contains more than 220,000 products (accessed 6/8/20). This means there are over 24 billion product pairs to be considered. It is doubtful that purchase history alone can be enough to estimate product relationships such as cross price elasticities.

One way to reduce the selection space is to only consider products within the same aisle or category. This has some intuition but has limitations when considering if vinegar is a food or cleaning product. Similarly, an item like corn is spread out between multiple product categories because it is packaged as produce, canned, milled, frozen, or prepared. A recommendation system that learns from consumer selections could be able to overcome this by including out-of-category products that had been substituted by previous customers.

Section 3:

Identifying Substitute Products

At the highest level, we will describe the conceptual blueprints for building a recommendation engine to identify products that are appropriate substitutes for out of stock items. However, the ability to match similar products depends on data about those products. Better data leads to better models. Even two companies that sell the same products in the same market will have a different volume, variety, and velocity of data. To allow for such heterogeneity that exists in the world, we designed a model that combines multiple, independent recommendation models individualized to the available data. Our results show that even just a few dimensions to this model can yield meaningful suggestions.

These models-within-models are “independent” in the mathematical sense: a scored match in one dimension is does not affect the scored match in another. For example, the text of the product description can be used as one dimension. When two products have the same description (e.g. a specific deodorant and then a two-pack of the same deodorant) they will have a 100% match on the dimension of product description, but this will not impact the match percent on the price dimension where they will presumably be quite different. However, the independence of models does not prevent the dimensions from being uncorrelated. Consider that Smucker’s brand peanut butter has very similar verbiage to Smucker’s strawberry preserves. If there was a dimension of brand similarity in the model, then that match would be highly correlated with product description in this instance.

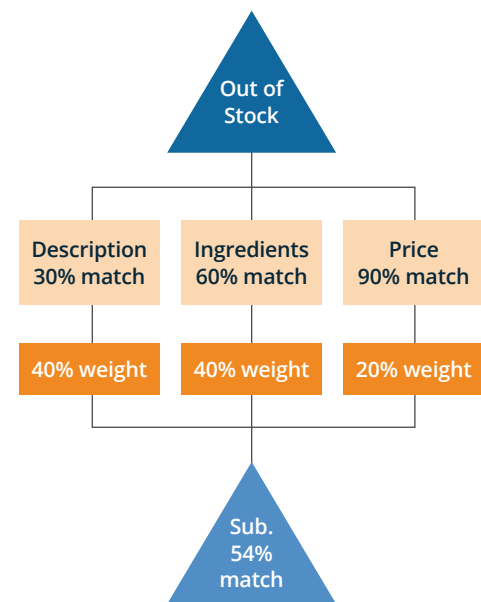


Source: harristeeter.com

The example shows that using a single dimension such as product description may yield an unwanted result, like substituting peanut butter for jelly. We improve on this by using an ensemble of models to construct a recommendation engine that is more robust to two dissimilar products matching simply because they have the same marketing department. The dimensions selected will vary by data availability and should vary by the use case. We suggest including other models like ingredient list, product price, nutritional information or anything else relevant to a specific retailer. See Section 7 for an explanation on how to build some of these individual recommenders.

3.1 COMBINING THE MODELS

The separate models are combined using a simple weighted average that yields a final match percent between two products. The diagram below shows a simplified model for two products with three dimensions of recommenders. In this example, the out of stock product and the substitute product are very similar in price, somewhat similar in ingredients, and not very similar in product description. The total match percent on its own has limited meaning; instead, it is compared against the total match percent of other products. The query can return the top n-many matches in ranked order to offer the best substitutes. The total match score is calculated for each *pair of products*, not just each product, making the problem combinatorically large.



The weights are instrumental in how well the model works and function by scaling the contribution each individual recommender has on the final match percent. Because each individual recommender is normalized to yield an output score between 0-100%, and the weights all add to 100%, then the total recommendation will also produce a total match score between 0-100%. The weight values must be determined in advance and based on domain knowledge as well as evaluating the impact on model results.

For example, we built a prototype recommender and assigned equal weights to product price and product description. Through testing, we found that price matching was not giving as much valuable information as we hoped, such as larger sizes of the same product that were not being included in the top five results because of price differences. Adjusting down the price weight resolved this problem. Finally, weights can also be adjusted to improve the profitability of the business, such as by recommending store brands with higher margins or nearby products that reduce the time to fulfill an order.

3.2 MODEL ENHANCEMENTS

Two additional ways to improve the model are to incorporate prior customer information and purchasing behavior as well as customer substitution selection from using the service. Previously, we described the model as a global recommender; however, it could also be customized down to customer type or even to the individual by using information from a customer profile or their purchasing behavior. This is explored further in Section 8.

Another model enhancement incorporates feedback into user data and improves over time. Each selection made is evidence that a substitute was satisfactory enough to continue with the purchase. That selection could then adjust either the match score or the dimension weights in future recommendations for the individual, customer type, or all users. See Section 9 for a deep dive into this concept.


Section 4:

How Does the Business Benefit

The primary benefit the business obtains is through the recommendation results of the model. By surfacing these to a personal shopper or picker when they encounter an out of stock item in a customer's order, they can make better decisions about which substitutions to offer. If the recommendations are surfaced to the customer when they are placing the order, either when an item is out of stock or when an item may soon be out of stock, the customer can select a suitable replacement. These actions will reduce the customer's frustration by giving them more of the items that they want.

Whenever a customer selects a replacement, or rates the appropriateness of a recommendation, that data enters into their customer profile. As the customer's personal profile gets richer and more complete, the business will be able to offer the customer a unique and tailored experience. This will drive loyalty through differentiation.

The customer's profile data containing past purchases, selected substitutions, and substitution ratings could offer rich insights into customer preferences. Some of these insights would include what goods that customer considers substituting, their level of brand loyalty, and their direction of preference indicating if they would replace the item with a more or less expensive alternative. This information, along with customer segmentation and product segmentation, can be used in more typical marketing targeting efforts.

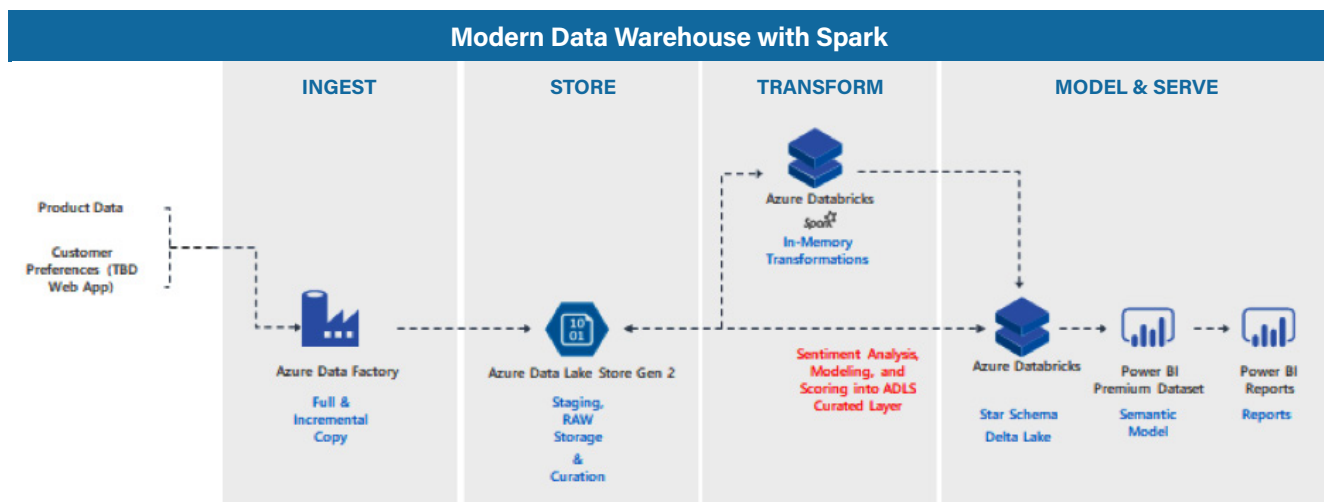


As the customer's personal profile gets richer and more complete, the business will be able to offer the customer a unique and tailored experience.

Section 5:

Solution Architecture

The solution is implemented in 4 layers: ingestion, storage, transformation/compute and modeling, and serving. The ingestion layer uses Azure Data Factory to land data into the Azure Data Lake storage layer in the Raw zone. This layer performs simple transformations and converts the data into parquet files in the staging layer for consumption in Azure Databricks. Azure Databricks picks up these files to perform more complex transformations to order data into the format needed for training the recommendation engine. Databricks is used to perform text analytics and score the similarity of products, and then output the scores into the curated layer in the Azure Data Lake. The curated layer is organized as a Star Schema and is ready to be used for analysis in the serving layer. We used Power BI to explore the data, however, it could be loaded into an Azure SQL DB for access by downstream web services or other applications.



INGESTION

Azure Data Factory was chosen for simple transformations due to its low cost and ease of implementation. Databricks is used for more complex transformations because of its flexibility using Spark SQL and Python. Databricks comes at a higher cost and is more code intensive, so it is only used for transformations not easily undertaken by Azure Data Factory.

Architecture Decisions - Ingestion		
	Azure Data Factory	Databricks
	Use REST API or Python copy activity to retrieve and copy them into ADLS	Use Databricks to execute Python code into the ADLS
Strengths	Low cost Easy to use	Flexible Can land transformed data into the Data Lake
Weaknesses	Click intensive Hard to follow	Higher cost Code intensive

STORAGE

Azure Data Lake Store is used for Raw and Staging because ADLS storage is inexpensive and these layers are not relational and would not leverage the benefits of a SQL Database. The curated star schema is landed in the ADLS because the simplicity of the model doesn't necessitate an Azure SQL DB, though a SQL or CosmosDB could be used as a serving layer for downstream applications that may have more difficulty accessing a model in the Data Lake.

Architecture Decisions - Storage		
	ADLS Raw/Stage + Azure SQL DB	ADLS Raw/Stage/Curate
	Land data in Raw/Staging zones in ADLS and transform, model, and store in Azure SQL DB	Use Azure Databricks to execute Python code into the ADLS
Strengths	Transformations with stored procedures Logging and error handling with stored procedures Ease of modeling for reporting	Lower cost Only one storage location
Weaknesses	Higher cost Additional development	More difficult to model data Transformations are potentially more costly

TRANSFORMATIONS AND ADVANCED ANALYTICS

Azure Databricks is powerful and flexible allowing the use of Spark SQL, Python, and R. These are tools our data scientist was familiar with and able to start developing immediately. The flexibility and familiarity of developing in Python notebooks is preferable in this case over Azure ML, which has a simple code light interface but has flexibility that is more difficult to leverage.

Architecture Decisions - Advanced Analytics		
	Azure ML Perform advanced analytics (modeling, sentiment analysis) in Azure ML	Databricks Use Azure Databricks to execute Python code into the ADLS
Strengths	Code light	Flexible Familiar to data scientists Spark in-memory compute
Weaknesses	Harder to leverage flexibility Click Intensive	Code intensive

MODEL AND SERVE

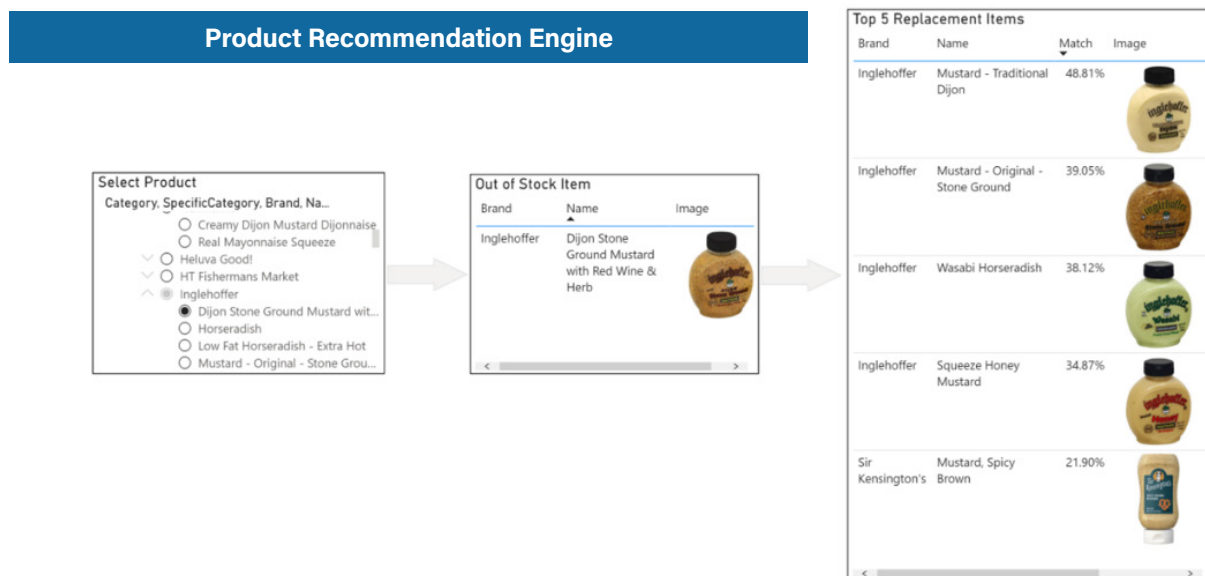
Power BI is easy to use and can be leveraged to quickly get into the data and explore the recommendation engine. To surface the data to downstream applications, an additional serving layer is required. An Azure SQL DB is a cost-effective way to allow other applications to query the data.

Architecture Decisions - Model and Serve		
	Power BI Dataflows and Datasets Use Databricks or MDF to create a star schema in ADLS to consume in Power BI via Power BI dataflows and datasets.	Azure SWL DB and Power BI Use REST API or Python copy activity to retrieve tweets and copy them into ADLS
Strengths	Low cost No server overhead Less development	Familiar modeling and transformation in SWL Easy to use Great for complex data models
Weaknesses	Fewer modeling tools Lack of established patterns and code base	Additional overhead Additional development

Section 6:

Product Attribute Recommendation Engine

The best recommendations come from an ensemble model that combines multiple recommenders together. Each independent model is built according to the format the data is in and the kind of information it conveys. We choose our variables according to our domain knowledge of retail by asking: "Do consumers consider calorie content when substituting a product?" If so, and if the data is available, we proceed to building a model for the variable that scores a match between 0-100%. Below are some of the different methods we use to go about scoring dimensions of different types and the issues that can arise in the process.



6.1 CATEGORICAL DATA

A dummy variable that takes the form of present (100% match) or absent (0% match) can form a recommender. This could include measures of whether a product is organic, non-GMO, or part of the same brand. Such a recommender on its own is not likely to be very insightful (e.g. substituting a non-organic watermelon with non-organic flour) but may prove useful if included in a model with a small weight. The model can also include constraints to consider rigid preferences like vegetarian, kosher, or gluten-free products. Constraints can be in the form of dummy variables (e.g. does or does not contain peanuts) but instead of partially adjusting the final score they will bring the match to 0% if the condition is not met. These constraints can be added to the recommendation model itself, or they can be built into the query by only calling on products that meet a defined criterion. Consider the following example model with three normal recommenders and one constraint:

$$\text{total match} = (\text{weighted same-brand match} + \text{weighted description match} + \text{weighted price match}) * \text{gluten-free match}$$

If the condition of two products is met (e.g. gluten-free match is 100%) then the total match is the weighted sum of the three normal recommenders. If the condition is not met (e.g. gluten-free match is 0%) then the total match will also be 0%. This is different from a normal dummy variable. If the two products are the same brand and both gluten-free, then the dummy variable same-brand match will be 100% and it will contribute to the total match according to the value of its weight.

6.2 TEXT AS DATA

Another class of variables will require text analysis as the basis of the recommender. In retail, this will include measures such as product descriptions or ingredient lists. Text analysis is the process of converting human language into a numeric representation for quantitative analysis. A common first step in this process is representing the text by a list of n-grams. An n-gram is an n-sized sliding window of words.

For example, a bigram (where $n=2$) of “nobody doesn’t like Sarah Lee” would become the list “nobody doesn’t”, “doesn’t like”, “like Sarah”, “Sarah Lee”. The benefit of n-gram is that it preserves the context of words by grouping them together before analysis. This can become very important for something like an ingredient list where items are listed in rank order. We take the list of n-grams and compare that against the list of n-grams for other products with text and measure their similarity. This is done by calculating their cosine similarity which measures the angle between two arrays of word counts. Cosine similarity is superior to a simple shared word frequency because it can still identify two texts of different lengths as similar in orientation. The output is a value between 0 and 1 which can become the percent match between two products.

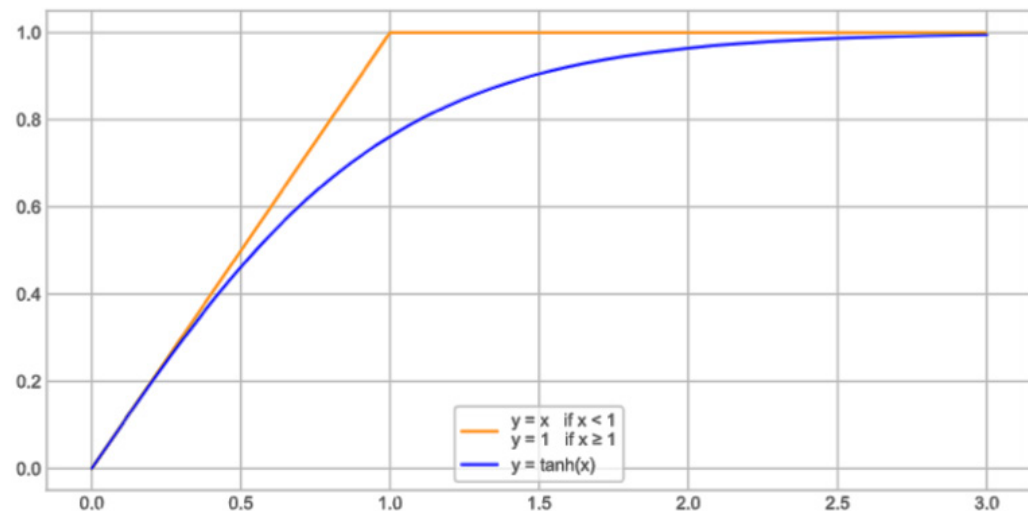
6.3 CONTINUOUS DATA

Next are continuous variables such as price, calories or macronutrients. The simplest method of comparing two products according to a continuous variable is to take the absolute percent similarity with a minimum value of 0%. This is straightforward and easy to interpret; however, it is limited to say that an item double the price (0% match) is no different than a product triple the price (still 0% match).

One solution to this problem is to take the hyperbolic tangent of the absolute percent difference. Hyperbolic tangent has some nice properties of being similar to small values between about 0-50% but then asymptotically sloping off to accommodate inputs of infinite value without ever exceeding an output of 100%. This is shown in the below graph that compares the hyperbolic tangent function and a simple $y=x$ line with a ceiling.

The table next to the graph shows similar values when price differences are less than 50% but then at 100% the hyperbolic tangent is 76.2% and at 200% difference it is 96.4%. Thus, we suggest the use of $1-\tanh$ (absolute percent difference) to calculate the match percent of continuous variables.

x	tanh(x)
0.0	0.000
0.2	0.197
0.4	0.380
0.6	0.537
0.8	0.664
1.0	0.762
1.5	0.905
2.0	0.964
3.0	0.995
4.0	0.999



6.4 MISSING DATA

It worth addressing the issue of missing product data as well. Retailers have complex databases with data that is constantly changing and often dependent on external sources. It is inevitable that a share of available products missing key variables are used in a recommendation engine like this. Another reason to use an ensemble model is to not expect complete, immutable datasets. It is not possible either to simply drop products with missing values when it is that very product that needs to be substituted.

Consider such a problem arising with Jiff brand creamy peanut butter with no ingredient list entered. If that item is out of stock, then it is not possible to match it with substitutes along the ingredient list dimension. The solution is to remove the ingredient recommender from the weighted average and renormalize the weights so they sum to 100%. This also makes the recommendation system use a broader range of products such as non-food items which do not have an ingredient list. This should occur on the individual level when a candidate substitute product is missing data.

For example, if Peter Pan brand creamy peanut butter is missing caloric count, then that measure should be excluded from its own weighted average formula and its weights renormalized. To allow such adjustments at the individual level would prevent unfairly penalizing products from consideration when driving down their match score.

Section 7:

Customer Segmented Recommender

Up to this point we discussed a recommendation model that is the same globally for all customers, but that does not have to be the case. It may be computationally expensive to individualize a recommender for each shopper, however, one solution is to define clusters of like customers and deploy a recommendation engine specific to their group type.

7.1 CUSTOMER SEGMENTATION

It is common practice in the retail space to segment customers into like groups in order to customize marketing efforts and shopper experience. We propose three types of data to use in segmenting customers: **demographic information, stated preferences and revealed preferences.**

- 1. Demographic data** contains observable characteristics about a customer before a purchase is made. These can be self-reported, such as senior or military status used for a discount program, or filled out as part of shopper loyalty membership or online profile. Demographic information can also be inferred by using their zip code from a credit card purchase to estimate their income based on neighborhood average.
- 2. Stated preferences** are what a customer claims to like about products. If a customer is interested in using a product recommender like this, then they may be willing to answer a few questions after placing an order. A customer that agrees to statements like “price is more important than quality” and “I choose stores based on the best sales” is providing useful information that can be used to improve which products they are shown as substitutes.
- 3.** A store can also segment customers according to their **revealed preferences** by identifying patterns in their purchase history. Buying behavior is often a better metric because that itself is the outcome variable. Revealed preferences also avoid the problem of when a customer is either knowingly or unknowingly being dishonest in their answers. However, purchase history can take time to accumulate and is subject to the issue of high dimensionality discussed in Section 2.1.

With the data in hand, we now must decide the method of splitting customers into groups. If the retailer already uses defined customer groups, then assigning new users to one of those groups is supervised machine learning. This is a classification problem that can be addressed with many different techniques, such as support vector machine, decision tree, and k-nearest neighbors, among others. If there are no pre-defined groups, then it will conduct unsupervised machine learning to solve a clustering problem. Here, the most widely used method is k-means, which takes all available variables to split up the customers into k many groups. The best tool will depend on the specific use case and is beyond the scope of this whitepaper to explore the trade-offs.

7.2 CUSTOM WEIGHTS

The next step after defining groups of customers is to build a recommender specific to their group. The simple approach would be to use the same model structure and variables, but also adjusting the weights according to each group. It would make intuitive sense if the model for the health-conscious customers have higher weights on matching nutritional information and the model for price-sensitive customers have higher weights on matching price per unit. Such a weighting mechanism offers greater customization for each of a store’s customer types.

Section 8:

Incorporating Continuous Customer Feedback

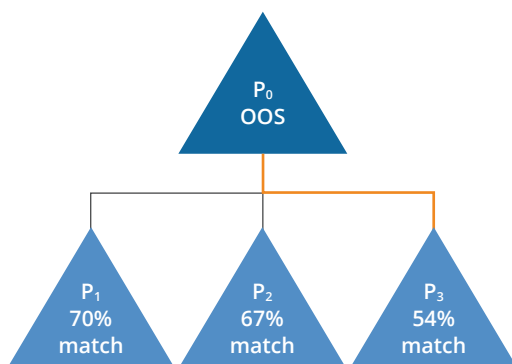
Thus far the recommendation model can be built without any specific customer data. In fact, one of the biggest strengths of this setup is how it does not require any sort of training data. This is necessary as an initial point since there will inevitably be a constant influx of new users and new products being introduced. Although this is an important starting point, the ideal recommender should be designed to improve with use. When a user selects a substitute for an out of stock product, that user is providing valuable information about the relationship between those two items. We describe how the model can be adjusted to incorporate that information in two distinct ways.

8.1 THE ADJUSTMENT PROCESS

It is useful to walk through how a hypothetical user selecting a substitute and then showing how the selection can improve the recommender model. Consider the following simplified example:

Step 1: Product P_0 is identified as out of stock (OOS) and our recommender returns three suggestions for as potential substitutes with their associated match score.

Step 2: The customer selects product P_3 as an alternative, even though it did not have the highest match score. This provides a signal that P_3 is a satisfactory substitute and that it should be rated higher going forward.

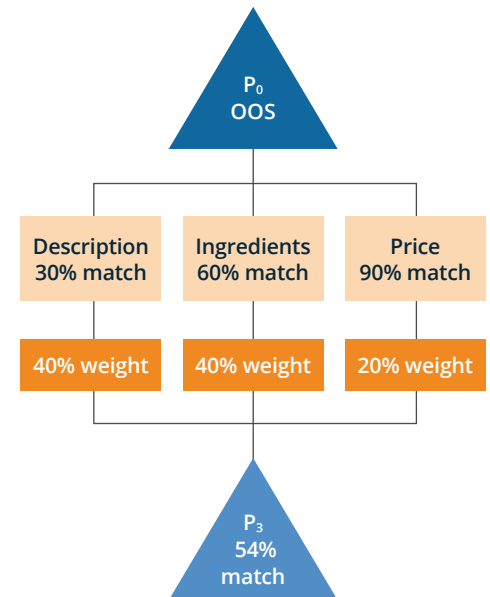


Step 3: We zoom into the relationship between P0 and P3 to see how the 54% match was calculated. In this example, the model is a weighted average of three individual recommenders.

Step 4A: In this option, we identify which dimension had the highest match and increase that weight for future matches. For example, Price should now be weighted slightly higher at 22% while Description and Ingredients should be weighted slightly lower at 39% each.

Step 4B: In this option, we increase the total match score between products P0 and P3 (for example, to 55%) so it is more likely to be selected in the future.

Note that the adjustments can occur at the global, customer type or individual level. At a minimum, once a user makes a substitute, that product should be remembered as a 100% match for that user should the stockout occur again.



8.2 ADJUSTING WEIGHTS VS. MATCH SCORES

There are benefits and challenges associated with adjusting either the weights or the total match score. The benefit of adjusting weights is that it adjusts for all other product pairs. This diffuses the information across the entire shopping experience and optimizes the parameter values over time. At the individual level, this option is much more feasible than optimizing match scores since the former is only changing one of the weights instead of changing one of all possible product pair scores.

The benefit of adjusting total match scores is that it offers more precision. Maybe, in this example, it was not price itself that made P3 a good substitute for P0 but something intangible. Changing the total match percent between these two specific products will not impact other matches but best captures the true relationship between these two products.

As an alternative to directly changing the match percent, it is also possible to add another individual recommender into the model where this information is stored. The recommender could return the percent of people that chose a different product when product P0 was out of stock.

8.3 FUTURE IMPLICATIONS

Once a system is in place that incorporates user feedback, the model may become vastly different than its original design. That is not only tolerable, but an ideal outcome! It would not be surprising if the best way to predict the optimal product substitute is by crowdsourcing the answer through a dynamic recommendation system. If there is too much reliance on cumulative user feedback, then there could be a concern with results that are globally suboptimal but locally optimal such as an ideal substitute never being shown because it never made the top five options. This can be overcome by adding some random perturbations to the total match score or randomly showing products that are decent matches but usually never shown in the top results.

Once the model is deployed and put in use, many of the specifications and assumptions included can then be tested. An A/B test easily evaluate multiple decisions based its impact on a defined objective like maximizing revenue or customer satisfaction.

Section 9:

Partner with BlueGranite

BlueGranite's Retail Industry thought leaders give our clients and partners the edge they need to win in an industry bent on change. The BlueGranite Catalyst for Analytics is our engagement approach that features our "think big, but start small" philosophy. Starting with collaborative envisioning and strategy sessions, we work with Retail and Consumer Products clients to discover, create, and realize the value of new modern data and analytics solutions using the latest technologies on the Microsoft platform.

We offer free learning resources online that can help accelerate your adoption of Modern Data and Analytics tools and strategies. Register for an upcoming in-person or virtual Power BI or Modern Data Platform training by visiting our [events calendar](#).

If your organization is ready embrace data as a strategic asset, let us help you accelerate your path to success. With our unique framework, we create roadmaps, build solutions, and deliver value across organizations. [Contact us today](#) and we'll be happy to help.

