# Customer Segmentation with Cosine similarity and Apriori Algorithm

**Lawrence Araa Odong**
**2nd year, Masters Computer Science**
**Matricola: 204736**
**Industry Intention**
**lawrence.odong@studenti.unitn.it**

**Evidence Monday**
**2nd year, Masters Computer Science**
**Matricola: 204729**
**Industry Intention**
**evidence.monday@studenti.unitn.it**

**Abstract**

This paper aims at segmenting customers in groups based on what they eat using cluster analysis on a given transaction data. The dataset provided consists of a market basket dataset as well as a dataset that provides for each recipe its ingredients. To achieve our goal, we implement two mining algorithms, cosine similarity algorithm module from the scikit-learn library as well as the apriori algorithm to cluster customers into various groups upon careful experimentations and analysis. We explain how each algorithm works with respect to the dataset being worked upon and also provide pseudocode for each algorithm implemented. Finally we evaluate the performance of our algorithm by testing them using synthetic data and real data.

**Keywords** *datasets, cosine similarity, cluster analysis, Customer Segmentation, Apriori, Cluster Analysis*

## Introduction

With the exponential growth of data and databases, it is critical to generate smart techniques that can transform data into useful information and knowledge. These techniques are essential for industries to identify relevant patterns and segmentation associated with this data. The information gathered is needed as a driving force to upscale, boost market productivity, develop new products, product recommendation, targeted advertising, category management, online marketing, stock optimization as well as customer segmentation.

Customers segmentation is the cornerstone of every business today. Businesses and retail chains have a large amount of sale data available. Proper analysis of these data is essential in order to understand the behaviour patterns of customers. Supermarkets are one of the organizations that keep records of customers day to day transactions. Other organisations that keep track of their customers are the banking sector, health organisations, government establishments etc. Customers can also be segmented based on demographic information (age, gender), geographic information (residence), etc. In this project, we aim at segmenting customers based on behavioural patterns. The behavioural data used for segmentation here is *consumption habit*. The idea is that customers are grouped based on the similarity of a chosen set variables. Customers in different segments will be treated differently strategically to achieve different marketing objectives with greater overall effect. In order to achieve this goal, we use cluster analysis.

We first examined a dataset containing lists of ingredients attributed to different cuisines. These cuisines are typically geographical representations such as 'Japanese', 'chinese', 'Italian' and so on. A cuisine can often be identified by its unique ingredients that are identifiable to that cuisine, but over the years customers explored new ingredients for cuisines in order to produce new cuisines. This raises some challenges in dealing with our problem. For example, the same ingredient may be used across different cuisines or the same cuisines having different ingredients with respect to a particular geographical location. Furthermore, a customer may buy only some of the ingredients he needs which means that it is not clear what dish a person intends to prepare.

In order to identify the customers based on the food they eat, first, we employed cosine similarity then apriori algorithm. We used cosine similarity to address the first challenge by clustering cuisines into groups of similar ingredients. We used apriori algorithm to obtain insights to a certain degree about the items that the customer might already have at home so as to narrow down what kind of dish or cuisine he is planning to prepare.

## Related Works

The purpose or our research is to cluster customers based on what they eat using a transaction dataset. Related works in this include cuisine prediction, Segmentation analysis of grocery shoppers, Retail Customer Segmentation.

### Cuisine Prediction

There have been many studies towards recipe predictions using various approach. R.M.Rahula, M.Anand et al. explored cuisine prediction based on a list of ingredients using tree boosting algorithm `XGBoost`. Here, Extreme boosting algorithm was used in creating and prediction of cuisine. Also, Random forest classifier algorithm was implemented using Sklearn python and a comparison was made between the results of both algorithms. After prediction, XGBoost happened to have perform better than Random forest classifier, having an over all accuracy of 80.417%. This is because XGBoost has both tree learning algorithm and linear model which makes it a more efficient boosting algorithm than the rest.

A full blown extension of the cuisine prediction is the recipe recommendation system. Geleijnse et al proposed a prototype for personalized recipe recommendation based on a user's past food selection. However this project goes beyond cuisine prediction to cluster customers using cosine similarity and apriori algorithm.

**Retail Customer Segmentation**
Here, Ondřej S., and Vladimír H. (2019) segmented customer based on the *recency*, *fequency* and *monetary value* (RFM) of their shopping. In their approach, individual baskets were first clustered followed by segmentation of customers using information obtained from the first step. Kmeans method was utilized for both steps. They proposed a process combining RFM and shopping mission (SM) segmentation based on long range of customer characteristics not tied to a single specific purpose. Their proposal answered the question *why a customer visits a shop* and concluded possible shopping missions to be either emergency purchase or general purchase. However, their proposal differs from the purpose of this paper because the shopping mission of customer is defined here *(prepare food)*. Hence, we aimed at clustering these customers based of the food they prepare per transaction.

**Segmentation analysis of grocery shoppers**
K.Nikolaos, P.wolson et. al in their paper segmented customers based on underlying demographics and income. KMeans clustering algorithm was implemented and the end result yielded six types of shopper profiles which were ranked according the cluster mean. The shopper profiles (clusters) include *Time Pressed Meat Eaters*, *Discriminating Leisure Shoppers*, *One Stop Socialites*, No *Nonsense Shoppers*, *Back to Nature Shoppers* and *Middle of the Road Shoppers* . Although, the experiment covered customer segmentation using KMeans algorithm but clustering was done based on demographics. However, this project focused on customer segmentation based on what customers eat.

## Problem Statement

This section gives a brief explanation of the problem and formal representation to it.
Our problem statement states, given a supermarket dataset containing a list of items bought by each customer and another dataset containing cuisines with their respective ingredients, our goal is to group customers into separate groups according to what they eat. In other words, we have to predict what cuisine the customers might prepare by analyzing the products that they bought which are the ingredients to the meals that they are planning to prepare.

**Formal Definition**
Given a set of transaction D such that $\mathcal{D} = \{T_1, T_2, ..., T_n, \}$ where $T_i$ is a single transaction made by a customer. $T_i$ =(P1,P2,.....Pn), where Pi is the set of items bought by the customer in a single transaction. And given a set of cuisines $\mathcal{C}$ such that $\mathcal{C} = \{C_1, C_2, ..., C_n, \}$, where $C_i$ is cuisine from a certain geographical location. The problem entails we identify types of customers based on the cuisine that they likely to prepare using the different ingredients they in the supermarket.
**Input:** Customer transactions and list of cuisines with their respective ingredients.
**Output:** A group of customers categorized according to what they eat.

**Solution**
To solve the problem, we used two approaches to solve our problem of trying to identify the different types of customers. These are TDIDF with cosine similarity and apriori algorithm. We divided our solutions into two phases. First, we calculate the cosine similarity between each cuisine pair in order to cluster cuisines that share similar ingredients together. In the second phase, we focused on predicting items that the customer might have already bought or items that led to the purchase of other items using apriori algorithm. To do this, we list the most frequently bought items from the market basket dataset and then individually assign them as input to the consequence section *(rhs)* of the algorithm. Finally, using the results obtained from both phases, a careful analysis was done to identify the types of customers based on what they eat. We described the implementations of both phases in details as well as their respective pseudo code and outputs in the next subsections.

Before delving into our approach to this problem, it is important to understand some preliminary techniques used. This includes TDIdf, cosine similarity and Apriori algorithm.

### Term Definitions

**Td-idf**
Tf-idf stands for term frequency-inverse document frequency [] and it is a numerical statistic that reflects the significance of a term in a document given a certain corpus. Typically, the td-idf algorithm is made up of two terms which are the *Term Frequency* and the *Inverse Document frequency*.

- Term Frequency: measures the frequency of terms in a document. That is, for every documents with different length, it finds the number of times a particular term appears. The computation is as follows:

$$TF_t = \frac{n_t}{D_t}$$

where $n_t$ is the number of times term $t$ appears in a document and $D_t$ is the total number of terms in the document

- Inverse Document Frequency: measures how important the term is. Hence, the need to weigh down the frequent terms while scale up the rare terms. A simple computation of the IDF is as follows:

$$idf_t = \log_x \frac{N}{df_t}$$

where $\mathcal{N}$ is the total number of documents $t$ appears in a document and $df_t$ is the number of documents with term $t$ in it.

**Algorithm 1** Tfidf algorithm

**Input:**
1. $\mathcal{D}$ : the documents of the training set.
2. $\mathcal{F}$ : the selected features set
3. $\mathcal{TF}$ : weight matrix from text presentation phase.

**Output:**

*TFIDFweights: tf-idf weights for selected features set*

**Procedure:**
1. reshape the columns of TF to match the selected features set
2. 
3. **for** Each term $t_i \epsilon \mathcal{F} \ldots$ **do**
4. **for** each document $d_j \epsilon \mathcal{D} \ldots$ **do**
5. **if** $TF_ij$ not equal zero then $df++$ **then**
6. **end if**
7. **end for** of document
8. $idfi = \log_x \frac{N}{df_t}$
9. **end for**for term
10. **for** each term $t_i \epsilon \mathcal{F} \ldots$ **do**
11. **for** each documents $d_j \epsilon \mathcal{D} \ldots$ **do**
12. $TFID_ij = TF_ij * idfi$
13. **end for** of Document
14. **end for** of term

**Cosine Similarity**

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically,it is the measure of the cosine angle between two vectors in a multi-dimensional space. It matches similar documents by counting the number of frequent words between the documents. For two documents A and B, the similarity between them is calculated by the use of the equation below:

$$cos(A, B) = \frac{A.B}{\|A\|\|B\|}$$

when the measure of two the two documents is close to 1 then they are identical and when their measure is close to 0, it indicates that there is nothing common between them.

**Note:** Attribute vector A and B are term frequency vectors of the documents.However, in our project each cuisine will be representing a vector in a multi-dimensional space, that is, the documents in the formula will be represented by each cuisine in our dataset.

**Phase One:** In this phase, as mentioned before, we grouped cuisines into similar groups by calculating the similarity between each cuisine pair using the pairwise metrics module under the python scikit-learn library. This module implements utilities to evaluate the pairwise distance or affinity of sets of samples.

**Phase One: TDIDF with Cosine similarity Implementation**

1. Import the dataset
2. Perform word lemmatization.
3. Generate top 15 ingredients for each cuisine using the *most-common()* function and counter object of the dictionary class.
4. Convert the raw data into a matrix of TF-IDF features using *TfidVectorizer()* function.
5. Calculate the cosine similarity of each cuisine using metrics.pairwise submodule.
6. Plot heatmap graph using season library to visualize our output.

**Observations**

For the purpose of visualization, the result of the cosine similarity was displayed using a heatmap graph. Cuisines that are similar to each other were observed to have high similarity value whereas cuisines that are dissimilar had a low similarity value. We also observed that similar cuisines were geographically located closer to each other. For example, most Asian cuisines such as Chinese, Filipino, Korean, Vietnamese, were very similar to each other same as European cuisines. This can be attributed to the diversity in culture in the US that has people from different heritage. To create clusters of cuisines that are similar to each other, we set a threshold for similarity to *0.75* and using the hierarchical method of clustering, we grouped cuisines together given that their average cosine similarity value is above our threshold value.

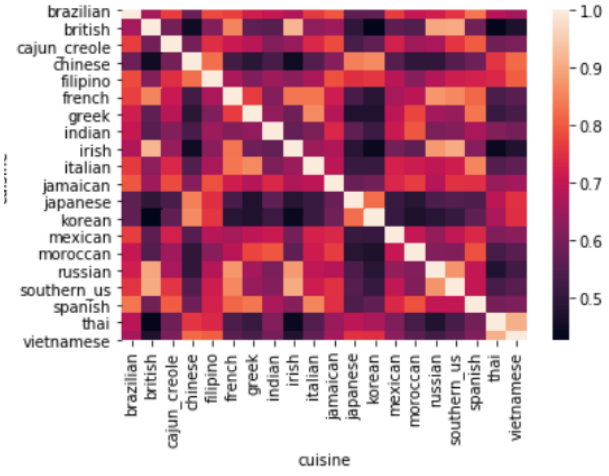


Figure 1: Cuisine similarity heat map

We also noticed that cuisines have unique identifiable ingredients. For example, 'sesame seed' is uniquely identifiable to Korean cuisine while 'feta cheese' is unique to Greek cuisine. Below is a graph showing this analogy:

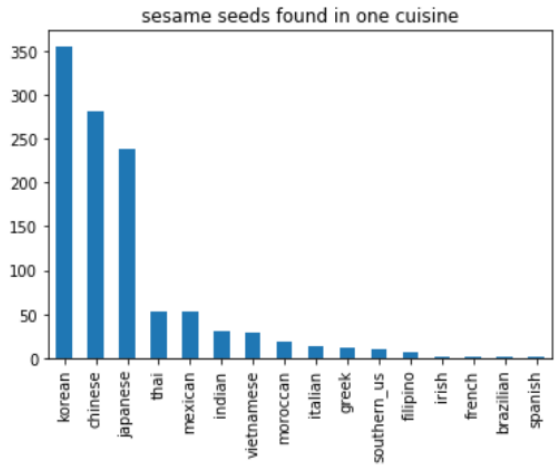


Figure 2: sesame seeds found in one cuisine

The following are the clusters that we came up with using the output of the cosine similarity and a table of frequently used ingredients in each cuisine:

1. {Chinese, Japanese, Korean}⇒ Cluster 1
2. {British, Southern-us, Russian, French}⇒ Cluster 2
3. {Italian, French, Spanish, Greek}⇒ Cluster 3
4. {Jamaican, Cajun-creole, Brazilian}⇒ Cluster 4
5. {Thai, Vietnamese, Filipino}⇒ Cluster 5
6. {Moroccan, Indian}⇒ Cluster 6

To have a better understanding of our clusters, we inspected the table containing frequent ingredient for each cuisine. Soy sauce, garlic, sesame oil/seeds, vegetable oil were the most frequent ingredients in all of the cuisines of cluster 1 indicating a more plant-based style of cooking. This is because of having only plant products in all of their frequently used ingredients. We renamed this cluster 1 to vegan-based cuisines due to the dominance of no animal ingredients used by cuisines in this cluster.
All-purpose flour, butter, sugar and eggs were all among the top spot ingredients for all cuisines in cluster 2 which are all fundamental ingredients to making pastry products. We renamed this cluster *pastry-based cuisines* due to the dominating amount of pastry recipes that its cuisines possess.

*Cluster 3* had a combination of plant and animal products such as cheese, olive oil, eggs but the top ingredient found in each of the cuisines in this cluster was grounded black pepper which is a top seasoning spice for both animal and plant-based dishes across Europe. With a wide variety of both plant and animal-based recipes, we named this cluster *Omnivores-based cuisines*.

*Cluster 4* did have cuisines dominated by tropical species such as cinnamon, black pepper, ground allspice which are all recipes for preparing spicy meals. We named this cluster *tropical cuisines*.
*Cluster 5* had similar ingredients as cuisines in *cluster 1* but distinct ingredients that differentiated them from the other Asians cuisines. Asides having a lot of vegetable and fruit ingredients such as lime juice, soy sauce, carrots and vegetable juice, they equally had sea-foods such as fish sauce which was the top ingredient for majority of the cuisines in this cluster. We named this cluster *sea-food cuisines*.
In *cluster 6*, cumin, ground cumin, ginger were the frequently used ingredients in both cuisines but in general, both cuisines have grounded spices as the base of their cuisines. Hence, we named this cluster *spice-based cuisines* due to the variety of multiple spices used by cuisines in this cluster.

| Cluster name | Cuisines belong to that cluster |
|---|---|
| Vegans | Chinese, Japanese, Korean |
| Pastry-based cuisines | British, Irish, French, Russian, Southern-us |
| Omnivores | French, Italian, Spanish, Greek |
| Tropical-spice cuisines | Jamaican, Cajun-Creole, Brazilian, Mexico |
| Sea-food cuisines | Filipino, Thai, Vietnamese |
| Spice-based cuisines | Moroccan, Indian |

Ingredients like water and salt were not considered when defining similarity between cuisines because they were frequent ingredients for almost all cuisines.

## Apriori Algorithm

Apriori is a classic algorithm for learning association rules or frequent itemsets. Association rules are statements that help find patterns in seemingly unrelated data. Association rules can be thought of as an *IF-THEN* relation with two parts, an antecedent *(if)* and a consequent *(then)*. An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. Association rules are created by analyzing data for frequent patterns using criteria of support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in a transaction database. Confidence indicates the number of times the if/then statements have been found true. A general pseudo code showing the steps for apriori algorithm is as follows:

**Algorithm 2** Apriori Algorithm

1: $C_k$: Candidate itemset of size k
2: $L_k$: Frequent itemset of size k
3: $L_1$ = $frequentitems$;
4: **for** $k = 1$; $L_k$ != 0; $k++ \ldots$ **do**
5: $\quad C_{||}$= candidates generated from $L_k$;
6: $\quad$ **for** Each transaction $t$ in database $\ldots$ **do**
7: $\quad\quad$ Increment the count of all candidates in $C_{||}$ + 1 that are contained in $t$
8: $\quad\quad$ $L_k$ + 1 = candidates in $C_k$ + 1 in min-support
9: $\quad$ **end for**
10: **end for**

Having this understanding below is the implementation of the algorithm in relation to this project:

**Algorithm 3** Apriori Implementation

1. Import *arules* package
2. Read dataset and transform it into transactional data.
3. Call *arules()* function on the new transformed transaction data specifying necessary parameters
4. Evaluate output sorted by *lift* measure.

**Phase Two:**
In this phase, we predict the transaction history of customers apriori algorithm. Before predicting the transaction history of customers, we listed the most frequently bought items in the supermarket and used the 'itemFrequency' method in $R$ to output a list of these items. We then filter out items that have been bought at least 1000 times and then created an *association rule* for each of the items that were bought at least 1000 times and sorted the output of each rule by lift. Lift indicates the strength of a rule over the random occurrence of a pair of items. We sorted the rules by lift because when sorting using lift, the directionality of rules is lost. For example, the lift of any rule, $A \Rightarrow$ B and the rule B $\Rightarrow$ A will be the same. For instance, the lift value for milk being bought as a consequence of bread being bought is the same as the lift value for bread being bought as a consequence of the milk being bought. With this, we can safely identify groups of items that are bought together with a particular ingredient and then predict what kind of meal the customer might prepare. We assumed that items that appeared on the Left Hand Side (lhs)of our output to be items that customers might already have in possession or items bought along side the frequent item.

**Phase 2: Implementation steps**

1. Import *arules* package
2. .Read dataset and transform it into transaction objects
3. Run apriori algorithm on the transaction objects specifying necessary parameters(confidence and support)
4. Adjust support and confidence parameters to obtain meaningful results.
5. Output the algorithm result sorted by lift measure.

```
> inspect(head(sort(milk_beef, by ="lift"),5))
    lhs                                rhs          support     confidence lift     count
[1] {rolls/buns,root vegetables}    => {beef}       0.004982206 0.2050209  3.907715 49
[2] {curd,domestic eggs}            => {whole milk} 0.004778851 0.7343750  2.874086 47
[3] {butter,curd}                   => {whole milk} 0.004880529 0.7164179  2.803808 48
[4] {butter,whipped/sour cream}     => {whole milk} 0.006710727 0.6600000  2.583008 66
[5] {pip fruit,whipped/sour cream} => {whole milk} 0.005998983 0.6483516  2.537421 5

 #sort by lift
> inspect(head(sort(vegetables, by ="lift"),5))
    lhs                                rhs                support     confidence lift
[1] {pip fruit,whipped/sour cream}    => {other vegetables} 0.005592272 0.6043956  3.123610
[2] {onions,root vegetables}          => {other vegetables} 0.005693950 0.6021505  3.112008
[3] {citrus fruit,root vegetables}    => {other vegetables} 0.010371124 0.5862069  3.029608
[4] {chicken,yogurt}                  => {other vegetables} 0.004880529 0.5853659  3.025262
[5] {root vegetables,tropical fruit} => {other vegetables} 0.012302999 0.5845411  3.020999

> inspect(head(sort(tropical, by ="lift"),5))
    lhs                                rhs              support     confidence lift     count
[1] {citrus fruit,pip fruit}        => {tropical fruit} 0.005592272 0.4044118  3.854060 55
[2] {other vegetables,pip fruit}    => {tropical fruit} 0.009456024 0.3618677  3.448613 93
[3] {pip fruit,yogurt}              => {tropical fruit} 0.006405694 0.3559322  3.392048 63
[4] {pip fruit,root vegetables}     => {tropical fruit} 0.005287239 0.3398693  3.238967 52
[5] {citrus fruit,root vegetables} => {tropical fruit} 0.005693950 0.3218391  3.067139 56
```

Figure 3: Sample of some of the individual rules created

The implementation of the apriori algorithm is relatively simple and straightforward. First, we downloaded the arules package and converted our data into *transactional* format because the apriori function only accepts data of the *transactional* class. We then create our rules to groups of products bought together.

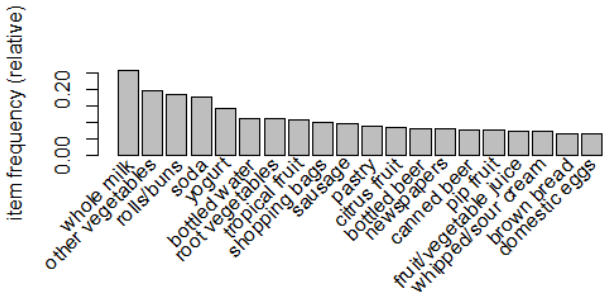


Figure 4: List of most frequently bought items

## Result

A sample of our results are shown in the figure 3. As mentioned earlier our rules are sorted by lift which is able to measure the co-occurrence of both items irrespective of the direction of the rule.We observe that all our results have a lift value greater than one which means the likelihood of associated items being together is high. The output table also contains other associations metrics, count which returns the number of transactions for each rule rule, the support measure tells us about the frequency of the items bought together and confidence measure tells us the likelihood of an item in the RHS being bought given an item in the LHS. Only ten items were bought more than 1000 times therefore we only created 10 rules for item.
We can interpret our output generated by the rules of the algorithm by comparing items in the rhs of the rule to items in the lhs of the rule and figure out which kind of recipe do they form. For example, in the first rule *milk-beef*, we

observe that customers that buy milk tend to buy products that are ingredients to pastry products, with this information we can confidently predict that this group of customers are going to baked products. On the other hand, we observe that, customers that purchase tropical fruit tend to buy other fruits and vegetables. This might imply that either they might prepare blended juice of vegetables and fruits or might plan on eating them as plain salads.

**Identifying the different types of customers**
In order to identify the different types of customers, we created a rule for each item that was purchased more than 1000 times and then compared the output sorted by lift with the clusters that we created in the previous section to predict which kind of meal or cuisine they might prepare.

The first type of customers we identified liked eating baked products. We identified these after analyzing the most bought item in the grocery which is whole milk. We observed that customers that bought whole milk also bought yoghurt, chocolate, buttermilk, sour cream, and domestic eggs either before or while buying the whole milk. All the items mentioned before are common ingredients for making pastry products such as cakes, cookies and pies so we are confident that this group of customers are likely going to prepare a meal using any the cuisines in cluster 1 also known as *pastry-based cuisines*. We could narrow it down to a Russian cuisine due to the purchase of sour cream which is a unique ingredient for Russian cuisines. If we were to define this type of customers that we would group them as people with sweet teeth who like eating sweet foods such as cakes and cookies.

The second type of customers we identified were vegans,These customers only bought other vegetables or fruits whenever they purchased items. We identified this group of customers from two rules ,that is, other vegetables and tropical tropical fruit. We observed that most customers that bought tropical fruit also bought mainly other fruits and vegetables as well or previously, the same applies for other vegetables. For example, customers that bought tropical fruit, also bought pip fruit,citrus fruit, other vegetables and root vegetables more often. From our clusters we observed that cuisines in *cluster 5 (Sea-food cuisines)* had a lot of vegetable recipes especially for making vegetable juice. With this assumption we believe most of these customers are probably going to prepare a blended juice of vegetables and fruits. Some customers might plan to eat the fruits and vegetables raw as salads which is common in European cuisines as observed in *cluster 3(omnivores)* that have recipes for both meat and plant based dishes. Further more, some customers might prefer to boil the vegetables and eat it as soup which is common practice while preparing meals that belong to cuisines in *cluster 1(vegans)*. We consider these types of customers as people that care about their well-being and focus on eating only healthy food.

Another type of customers are customers that eat both *animal and plant products*. We observed that people who bought root vegetables also might already have herbs, beef, other vegetables and onions. Similarly, customers who bought other vegetables might have bought canned fish, frozen fish, beef, root vegetables and herbs. From the items listed we can observe a good number of customers purchase fish and meat alongside these vegetables and it would be safe to assume that these customers might prepare a European cuisine which falls under *cluster 3(omnivores)*. Most especially, herbs that are well known for being used in French and Italian cuisine .
Upon further inspections by using other association rule-measuring metrics, we observed that red wine was also purchase sometimes with the above products which is a frequent ingredient in both Italian and French cuisine according to our data. On the other hand, we are confident that customers that buy fish and vegetables together will most definitely prepare a cuisine that falls under cluster 5. We believe that customers that fall under this cluster have a more complete diet which is still healthy at the same time.

The last type of customers type we observed are customers that like eating *fast foods*. We observed that most customers who bought meat might have already bought rolls/buns, root vegetables while customers that bought rolls/buns a which had previously bought items such as meat, root vegetables, margarine, sausages and frankfurters which strongly imply preparation of homemade quick and easy-to-prepare meal such as burgers, hot-dogs or sandwiches. With direct purchase of ingredients such as frankfurters, also known as Frankfurt sausage, we can easily tell what the customer is going to prepare and also which cuisine they will be preparing. The rest of the ingredients are also common ingredients that are used by cuisines in cluster 3(omnivores).

## Experimental Evaluation

To be able to validate an evaluate the algorithm, we had to carry out an implementation on a real dataset to show how the algorithm worked and how it performs in comparison to other algorithms. In this section, we will briefly describe the implementation process and the environment the implementation was done on.

### Environmental Evaluation

To implement the algorithm, we used python 3. We used python libraries like the scikit-learn cosine similarity module to help evaluate the similarities between pairs of cuisines. We primarily developed the algorithm and carried out the experiments on an Intel(R) Core(TM) i5-8250U Processor @ 1.6GHz with 8GByte main memory running a Windows 10 operating system. The experiments were carried out mainly using Jupyter note-book.

### Evaluation

We used both real data and synthetic data to perform experimental evaluation of our cosine similarity algorithm to validate how efficiently it computed the similarity between

each cuisine. We used the the *test.json* file that came with together with the *train.json* data file which was used to cluster the cuisines. We managed to only create 261 objects because we had to manually add the cuisine object for every sample data, we labelled the cuisine object in our test data as "test-data". We then generated synthetic data using an online API that generates random data in json format. We created 1000 samples of synthetic data having *id*, *cuisine* and *ingredients* objects. Id and ingredients were generated randomly while the cuisine object was labelled with "syn-data" value. Test how well the cosine similarity identified similar cuisines, we included both test samples into the train.json dataset and ran the cosine similarity again on the new dataset. As it observable in the heatmap graph below, the algorithm was able to identify the similarity in test data with actual ingredients and has similar similarity values with the actual cuisines while the black horizontal and vertical strip represents the synthetic data of our test dataset. The algorithm was able to detect the noise in the data by not finding any similarity between the random data and the ingredients in the cuisines.

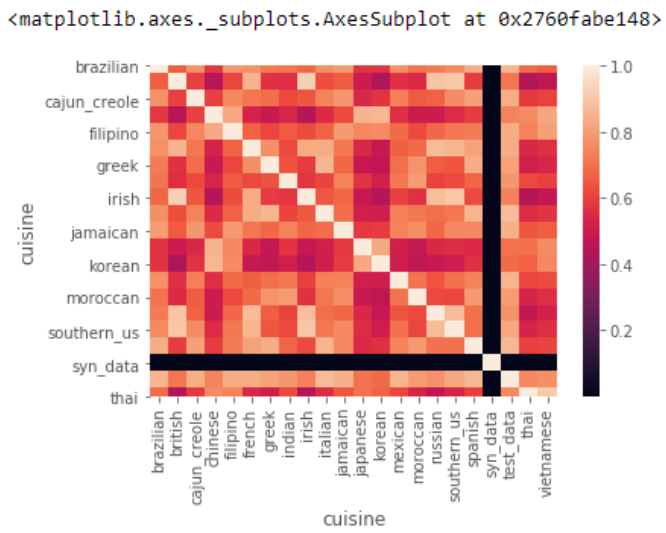


Figure 5: Cuisine similarity heat map of synthetic data

**Comparison with Kmeans Algorithm**

In this section, we discuss our output obtained from implemeting the Kmeans algorithm as an attempt to solve our problem. We also explained the shortcoming and our proposed reasoning behind it.

For our first phase, we experimented on different values of $k$ (number of clusters) to obtain the most desired distribution. We began with k=2 and iteratively incremented the value of $k$ until we obtained the number of clusters that generated a level of accurate result.

| cluster | count |
|---|---|
| 0 | 5519 |
| 1 | 5818 |
| 2 | 5893 |
| 3 | 5903 |
| 4 | 5418 |
| 5 | 5646 |
| 6 | 5577 |

Figure 6: Summary of KMeans algorithms

The table above show a summary of the clusters produced. This table shows the number of clusters created with their respective number of counts for each cluster.

| Cuisine | cluster |
|---|---|
| greek | 3 |
| southern_us | 5 |
| filipino | 1 |
| indian | 5 |
| indian | 3 |
| jamaican | 2 |
| spanish | 0 |
| italian | 2 |
| mexican | 1 |
| italian | 3 |
| italian | 2 |
| chinese | 4 |
| italian | 2 |
| mexican | 0 |
| italian | 6 |
| indian | 5 |
| british | 6 |
| italian | 2 |
| thai | 2 |
| vietnamese | 3 |

Figure 7: KMeans clusters of k=7 using Spark

This table gives an overview of each cuisine and the cluster they belong to. For example, southern-us, filipino and indian cuisines all belong to cluster 5 while only mexican cuisine belongs to cluster 6. Cuisines that belong to the same cluster can be viewed as having similar ingredients. We observed some inconsistencies with the clusters formed. For example, in our cluster model, the italian cuisine fell into different clusters. Our assumption was based on the idea that the fact that this cuisine had the highest number of ingredients in the dataset.

## Conclusion

In this work, we discussed the problem of identifying types of customers based on the food they eat. We explained the relevance of providing a solution to this problem and also reviewed some existing works and algorithms related to this topic. Based on our problem description, we managed to cluster customers according to what they eat into four categories using cosine similarity and apriori algorithm. We also tested our model on a synthetic dataset to see how well it scales. Furthermore, an implementation using KMeans clustering algorithm was used initially but it presented inconsistent results. Hence, there was need for comparison. Above all, we were able to solve the problem and produce appropriate results.

## References


Rahul Rahul V, Anand K. *Cuisine Prediction based on Ingredients using Tree Boosting Algorithms*. Indian Journal of Science and Technology, 2016.

Andreas K, Stavros A. *Large Scale Product Recommendation of Supermarket Ware Based on Customer Behaviour Analysis*, 2018.

Hendrik H, Maya N. *What Cuisine? - A Machine Learning Strategy for Multi-label Classification of Food Recipes*. University of California, 2015.

Nikolas K, Paul W., et al. *A segmentation analysis of U.S grocery Shoppers*. University of Minnesota, 2001.

Tom B., Gilbert S., et al. *Using Shopping Baskets to Cluster Supermarket Shoppers*. Department of Applied Economic Sciences Limburg University Center, Belgium.

Quiman H, Feng Z. *Research on retailer data clustering algorithm based on Spark*, 2017.

Ondřej S., Vladimír H., et al. *The Role of Shopping Mission in Retail Customer Segmentation*. University of Economics, Prague, 2019.