DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

MARKET BASKET ANALYSIS

by

Sena YURTSEVEN

Zeynep Hilal DESTEBAŞI

Halil İbrahim ÇAĞIRKAN

Advisor

Asst. Prof. Dr. Feriştah ORUCU

December, 2021

İZMİR

CONTENTS

CHAPTER ONE

EXPLANATION ABOUT ALGORITHM

1.1. Apriori Algorithm

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent itemsets. (Apriori Algorithm In Data Mining: Implementation With Examples,2021).

Our dataset, basket.csv, is the best example of the apriori algorithm. The Apriori algorithm is the application that can find the product(s) closest to a best product.

Apriori is the first FIM algorithm. The algorithm takes a transaction database and the minimum support threshold as input. It uses a standard database representation also called a horizontal database.

Apriori first scans the database to calculate the support of each item. Then the algorithm uses this information to identify the set of all frequent items. Then, it performs a breadth-first search to find larger frequent item sets. During the search, it uses the frequent itemsets of a given length $k - 1$ to generate potentially frequent item sets of length k. This is done by combining pairs of items of length k that share all but one item.

Apriori is an important algorithm as it has inspired many other algorithms. However, it suffers from important limitations. The first one is that the algorithm generates candidates by combining item sets without looking at the database; it can generate some patterns that don't even appear in the database. Thus, it can spend a huge amount of time processing candidates that don't exist in the database. The second limitation is it has to repeatedly scan the database to count the support of candidates (very costly). The third limitation is the breadth-first search approach can be quite costly in terms of memory as it requires at any moment to keep in the worst case all k and $k - 1$ item sets in memory. Briefly, the time complexity is $O(m^2n)$ when m is the number of distinct items and n is the number of transactions.

```
1  Scan the database to calculate the support of all items in I;
2  F_1 = {i | i ∈ I ∧ sup({i}) ≥ minsup} ;          // F_1 : frequent 1-itemsets
3  k = 2;
4  while F_k ≠ ∅ do
5      C_k = CandidateGeneration (F_{k-1}) ;        // C_k : candidate k-itemsets
6      Remove each candidate X ∈ C_k that contains a (k-1)-itemset that is not in F_{k-1};
7      Scan the database to calculate the support of each candidate X ∈ C_k;
8      F_k = {X | X ∈ C_k ∧ sup(X) ≥ minsup} ;      // F_k : frequent k-itemsets
9      k = k + 1;
10 end
11 return ⋃_{k=1...k} F_k;
```

Figure 1.1.1 Pseudocode of Apriori Algorithm

1.2. FP-Growth

The Frequent Pattern Growth Algorithm created with the development of Apriori. Apriori is an algorithm that is not preferred for large data sets due to its disadvantages. The FP-Growth algorithm aims to provide a more effective working area than the apriori with the help of a tree. To talk about this tree structure used, it purpose is to find the itemsets with the best relationship, just like apriori. Each node in the tree represents an item in an itemset. The root node is shown as null *(Frequent Pattern Growth Algorithm in Data Mining,November 2021)*. To create this tree, two scans are required in the database. The first scan selects frequent items, which are then sorted in descending order, to build the F-list. The second scan creates the FP-Tree. First, processes are reordered according to list F, removing infrequent items. The rearranged transactions are then added to the FP Tree. The FP-Growth entry is the FP-Tree and the minimum number of supports. FP-Growth separates the nodes in the FP-Tree from the least common element in list F. While visiting each node, FP-Growth collects items in the path from the node to the root of the tree. These items form the conditional pattern base of that item. A conditional pattern base is a small database of patterns that occur with the item. Then the FP-Growth is created. Small FP-Tree from the base of the conditional pattern and FP-Growth on the FP-Tree are executed. The process is iteratively repeated without creating a conditional pattern base (FP-Growth Algorithm in Weka,2018). Below is the FP-tree. On the other hand, you can examine the summarized version of this structure.

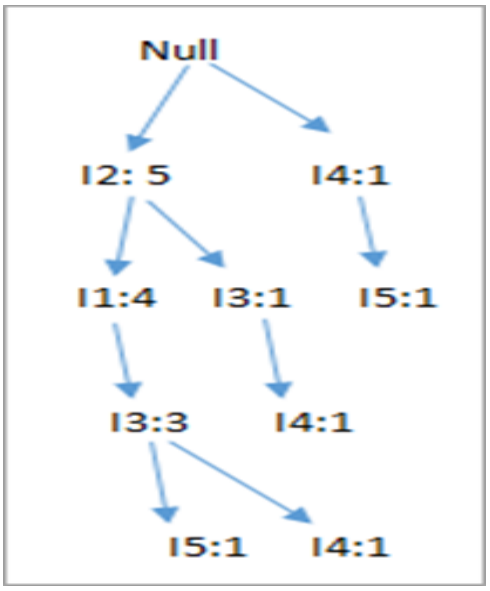


Figure 1.2.1: Example of FP-Tree

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I3,I4:2}, {I2,I3,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

Figure 1.2.2: Summarize of FP-Growth

1.3. Classification

Classification can be performed on structured or unstructured data. Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under. Classification algorithms learn which data to assign to which class from the given training set. It then tries to assign the test data to the correct classes.

The classification is not actually a linear function, but we can approach the classification problem by ignoring the fact that y is discrete and use the linear regression algorithm to try to estimate the value of x.

Our function uses the "Sigmoid Function", also called the "Logistic Function":

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

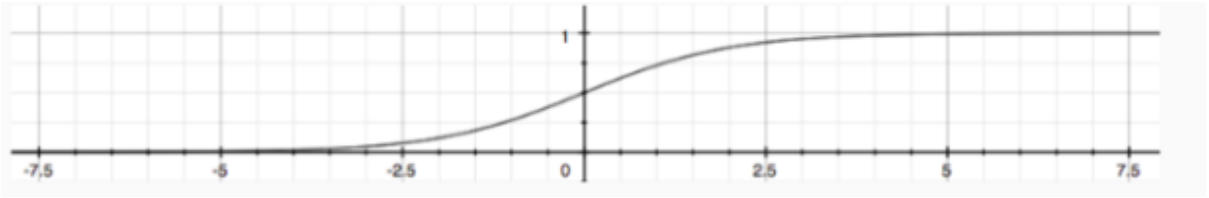


Figure 1.3.1 Sigmoid Function

The function shown here maps any real number to the range (0, 1), allowing a random-valued function to be converted to a more suitable one for classification.

In classification problems, we first separate our data set as train and test sets, then create a model on the train data set and test the predictions made on the test data set. However, there may be some problems with the train/test separation. We may not have been able to make the data set separation randomly. We may have chosen only men or women of a certain age, from a certain region, and built a model on them. This will cause an overfitting problem. We can solve this problem with Cross Validation.

In K-Folds Cross Validation, we divide our data into k different subsets. We use k-1 subsets to train our data and leave the final subset as test data. The average error value obtained as a result of k experiments indicates the validity of our model.

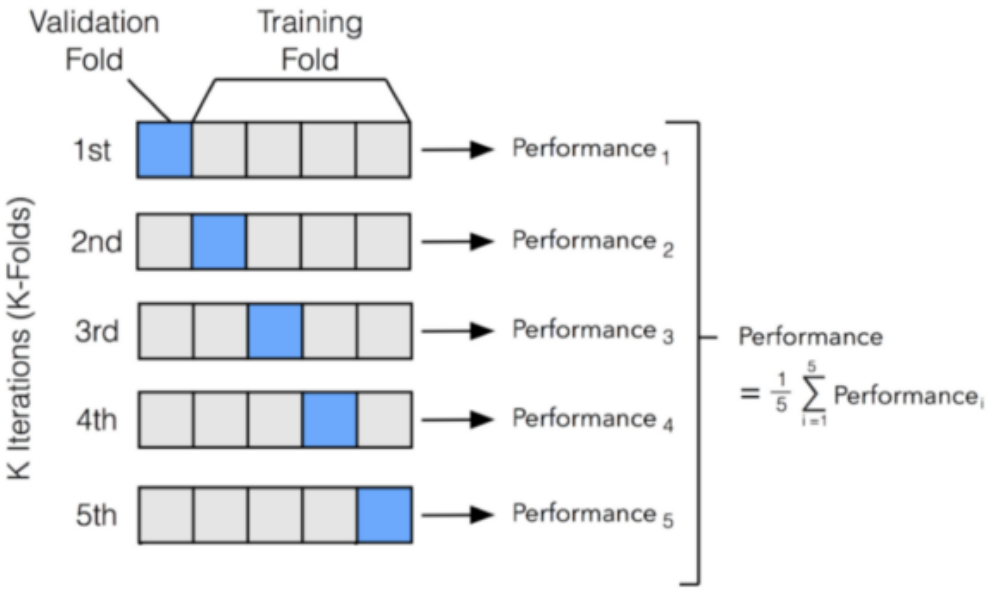


Figure 1.3.2 K-Folds Cross Validation

1.4. Clustering

Cluster Analysis means that to find out the group of objects which are similar to each other in the group but are different from the object in other groups. The quality of a clustering method depends on the similarity measurement, implementation and its ability to discover some or all of the hidden patterns.

Clustering algorithms are used to group data points based on certain similarities. There's no criterion for good clustering. Clustering determines the grouping with unlabelled data. It mainly depends on the specific user and the scenario. (Exploring Clustering Algorithms: Explanation and Use Cases, 2021)

Typical cluster models include:

- Connectivity models – like hierarchical clustering, which builds models based on distance connectivity.
- Centroid models – like K-Means clustering, which represents each cluster with a single mean vector.
- Distribution models – here, clusters are modeled using statistical distributions.

- Density models – like DBSCAN and OPTICS, which define clustering as a connected dense region in data space.
- Group models – these models don't provide refined results. They only offer grouping information.
- Graph-based models – a subset of nodes in the graph such that an edge connects every two nodes in the subset can be considered as a prototypical form of cluster.
- Neural models – self-organizing maps are one of the most commonly known Unsupervised Neural networks (NN), and they're characterized as similar to one or more models above.

Note, there are different types of clustering:

- Hard clustering – the data point either entirely belongs to the cluster, or doesn't. For example, consider customer segmentation with four groups. Each customer can belong to either one of four groups.
- Soft clustering – a probability score is assigned to data points to be in those clusters.

One of these clustering types, the K-means algorithm was used. The purpose of this algorithm is to ensure that the clusters obtained at the end of the partitioning process have maximum similarities within clusters and minimum similarities between clusters.

K-means is one of the most commonly used clustering algorithms. It is easy to apply. It can cluster large-scale data quickly and effectively. "K" refers to the fixed number of clusters needed before starting the algorithm. With its iterative partitioner structure, the K-means algorithm reduces the sum of the distances of each data to the cluster it belongs to. The K-means algorithm tries to detect K clusters that will make the squared error the smallest.

With K-means, it can be said that clustering is correct as long as the similarity within the cluster is large and the similarity between the clusters is small. Although the problem is NP-hard, the K-means algorithm usually gives a good solution with an iterative approach (K-means kümeleme, 2021).

We clustered according to the products purchased by the users with the K-means clustering method. Thanks to this, we can find the cluster to which the user belongs according

to the products purchased. After the cluster it belongs to is found, it can be predicted which products it can buy other than its own.

CHAPTER TWO

DESCRIPTION OF DATASET

2.1. Description

In the developing economy and technology, it has become very difficult for many companies and stores to survive. For this reason, companies and sales firms want to get ahead of their competitors with different techniques. Among these techniques, data mining is the most preferred one today. Data mining and machine learning is the collection of raw data of a company that has not been processed yet and turning it into useful information. Here, our aim is to increase our commercial profit from raw data and to get ahead of a competitor in the market.

The initial dataset was collected from Groceries dataset. Then data was modified and fragmented into 2 datasets for ease of MBA implementation. The first dataset shows the dates on which the products were purchased in the market. The second dataset contains information about which products these products are taken with. This dataset is taken from kaggle.

The basket.csv file from this dataset, which is divided into two different groups, shows the combination of certain products. Since these products are in groups of three and four, the columns are not completely filled. When a purchase is complete, nine ',' are inserted to indicate the end of the line. When 9 commas are encountered in the file, it is understood that the shopping is completed. There is no missing value in it. Because in this dataset, the products taken are listed in order. If the features are not filled in, it shows that no more items are taken.

The Groceries data.csv file from this data set, shows which items are sold on a specific date. There are no missing values in the dataset. Also this dataset can predict which items may be sold more than the others in a season. So we have to create a derivative feature to keep season value. Relationships between products in certain months can also be examined.

2.2. Meanings of columns

Member_n...	Date	itemDescri...	year	month	day	day_of_week
1808	2015-07-21	tropical fruit	2015	7	21	1
2552	2015-05-01	whole milk	2015	5	1	4
2300	2015-09-19	pip fruit	2015	9	19	5
1187	2015-12-12	other vegetables	2015	12	12	5
3037	2015-01-02	whole milk	2015	1	2	4
4941	2015-02-14	rolls/buns	2015	2	14	5
4501	2015-08-05	other vegetables	2015	8	5	2
3803	2015-12-23	pot plants	2015	12	23	2
2762	2015-03-20	whole milk	2015	3	20	4
4119	2015-12-02	tropical fruit	2015	12	2	2
1340	2015-02-24	citrus fruit	2015	2	24	1
2193	2015-04-14	beef	2015	4	14	1
1997	2015-07-21	frankfurter	2015	7	21	1
4546	2015-03-09	chicken	2015	3	9	0
4736	2015-07-21	butter	2015	7	21	1
1959	2015-03-30	fruit/vegetable juice	2015	3	30	0
1974	2015-03-05	packaged fruit/vegetables	2015	3	5	3
2421	2015-02-09	chocolate	2015	2	9	0
1513	2015-03-08	specialty bar	2015	3	8	6
1905	2015-07-07	other vegetables	2015	7	7	1

Figure 2.2.1: The Analytics Base Table for Groceries Data Set

The first dataset, Groceries, consists of seven columns. These columns are; Member_number, Date, itemDescription, year, month, day, day_of_week. This dataset contains 38,800 data. The first column, Member_number, is unique and gives the ID of each member. The minimum value in this column is 1000. The maximum value is 5000. The most repeated value is 4007. Numbers between 1000 and 5000 have an average of 3000 and a standard deviation of 1.150. The second column contains the date. This date is written in the form of year, month, day. These dates are written with numbers and a '-' between them. The third column writes what the products found are. There are 167 different products. The most common product is 'whole milk'. Columns 4th, 5th, and 6th represent the fragmented form of the second column. The last column shows which day of the week it represents. The days of the week are indicated by numbers from 0 to 6.

```
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
 3   year             38765 non-null  int64
 4   month            38765 non-null  int64
 5   day              38765 non-null  int64
 6   day_of_week      38765 non-null  int64
dtypes: int64(5), object(2)
```

Figure 2.2.2: Data Type of Groceries Data.csv

We can solve the gain problem we have identified above with the features we have defined here. The part where the Date is written completely in the first data set does not contain any necessary information. For this reason, it can be removed from the dataset later. Second, a derived feature 'Season' feature can be added to this dataset. With this feature seasonal sales can be predicted. Another information we can extract is that suggestions can be made in accordance with the Member Id. We can only offer products or discounts specific to that customer. In addition, for the whole data set, the relationships between the products can be examined only depending on the months. We can later think about what we can do to use the day feature.

0	1	2	3	4	5	6	7	8	9	10
whole milk	pastry	salty snack								
sausage	whole milk	semi-finished bread	yogurt							
soda	pickled vegetables									
canned beer	misc. beverages									
sausage	hygiene articles									
sausage	whole milk	rolls/buns								
whole milk	soda									
frankfurter	soda	whipped/sour cream								
frankfurter	curd									
beef	white bread									
butter	whole milk									
frozen vegetables	other vegetables									
tropical fruit	sugar									
butter milk	specialty chocolate									
frozen meals	dental care									
rolls/buns	rolls/buns									
root vegetables	detergent									
sausage	rolls/buns									
dish cleaner	cling film/bags									
canned beer	frozen fish									

Figure 2.2.3: The Analytics Base Table for the Basket Dataset

The second data set has 15,000 rows. This dataset shows the products in the other dataset to be sold together. The dataset consists of 11 columns. There are no empty elements in the first and second columns. If no other product is received after two products are purchased, the row is completed. 167 items in the other dataset are used. Each column in the same row is different from each other.

We can use our second dataset to solve our main problem again. This time unlike the other data set, all products were ordered temporally. On these features we will try to find the relationship between products depending on time. In this way we will be able to recommend another product after purchasing a product.

```
RangeIndex: 14963 entries, 0 to 14962
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       14963 non-null  object
 1   1       14963 non-null  object
 2   2       4883 non-null   object
 3   3       2185 non-null   object
 4   4       795 non-null    object
 5   5       451 non-null    object
 6   6       276 non-null    object
 7   7       196 non-null    object
 8   8       51 non-null     object
 9   9       1 non-null      object
 10  10      1 non-null      object
dtypes: object(11)
memory usage: 1.3+ MB
Dataset information None
```

Figure 2.2.4:Data Type of Basket.csv

CHAPTER THREE

APPLIED DATA PREPARATION TECHNIQUES ON THE DATASET

3.1. Getting Know the Data

3.1.1. Basic Statistical Descriptions of Data

If we want to examine some basic statistical results for our data set, we can make some inferences based on the knowledge that our data is categorical. For example, mode will be a very important inductor for us. For this reason, let's create a frequency table for our data. (The top 5 most purchased products are shown in the table.).

```
whole milk                2502
other vegetables          1898
rolls/buns                1716
soda                      1514
yogurt                    1334
```

Figure 3.1.1: Shows the most purchased variable

```
Summer    10082
Spring     9790
Autumn     9454
Winter     9439
Name: Season, dtype: int64
```

Figure 3.2.2: Shows the most purchased season

When we look at the figures we can clearly see what season most purchased and what products bought.

Season	Autumn	Spring	Summer	Winter	All
itemDescription					
Instant food products	9	14	14	23	60
UHT-milk	68	95	92	68	323
abrasive cleaner	5	7	6	4	22
artif. sweetener	4	9	8	8	29
baby cosmetics	1	0	0	2	3
...	...	...	...	...	...
white wine	30	52	49	45	176
whole milk	629	652	662	559	2502
yogurt	334	367	328	305	1334
zwieback	18	14	14	14	60
All	9454	9790	10082	9439	38765

Figure 3.2.3: Shows the which season which item bought

Table 3.2.3 shows us how many seasonally a product is bought.

3.2. Data Integration

Our data is keeped in two different tables. The data is stored as a .csv file. No merge operation is performed. The second table is formed by preparing the same data by paying attention to the time order. For this reason, no join operation will be applied on the two tables. It is known which company the data belongs to, but we know that it is a grocery store. The range of date in the dataset is between 1 January, 2014 and December 30, 2015. Although there is no known lost date, only the names of products sold from that date are included in our table. However, we do not have any demographic information of the customers, and only customers are represented as an ID.

3.3. Data Selection & Reduction

Depending on the number of our data, a mode can take quite a long time to run, so simplifying the data and working with them by taking some samples in a large data set will affect the results very little. For this reason, we have to perform some learning operations on our data. Some procedures for this will be discussed below:

- Data Aggregation: The number of features is reduced by applying some mathematical operations to the data. Some of these mathematical operations: average, summing can be given as an example. There is no data in our data set that we need for this operation.

- Dimensionality Reduction: There is a cell in the first dataset where we host a date as month, year and day. Using this column, it is clearly seen that there are separate features with month day and year. So, the column containing all the date values in the data set can be deleted and we can lighten the load on our dataset. However, from the data of which day of the month it is, the information about which day of the week it is has been deduced. For this reason, we do not plan to make any transactions regarding the day data. The day column can also be deleted. In our second data set, there is no operation suitable for this situation.

- Data Compression: It is the process of encoding the coded data in the dataset. There is no situation where we would apply this strategy on our data.

- Clustering: It is the prıcess of clustering data based on similarity. We can use this method to show which products are preferred seasonally, or we can create customer segmentation by clustering people according to what they buy.
- Concept Hierarchy Generation: It is the process of transforming lower level concepts into higher level concepts. This process was not considered necessary for our data set.
- Sampling: Sometimes, when the data sets are very large, some selections are made and a data evaluation is made on them. Care must be taken while performing the sampling process. If this operation greatly changes the results of the data set, we should avoid applying this operation. Some techniques that can be applied for sampling are:
    - Top Sampling: It is one of the methods that is not recommended to use Sampling. According to this method, a percentile at the top of the data set is selected. This data set may cause some data loss for our data set as well.
    - Random Sampling: A few data are selected randomly from the data set and a dataset is created with them. It might be a good option as it uses natural selection.
    - Stratified Sampling: It is a sampling method in which selection is made by preserving the proportions.In our case, especially in the first data set, resampling can be applied by preserving the pırchase rates of the products.
    - Under-Over Sampling: Stratified Sampling has two more specialized and known sampling methods. Under sampling groups are created and selection is made according to the smallest of these groups,while in over sampling the selection is made according to the higher number. Of course, the answer to a question that comes to mind is that the small data set is filled with random sampling with a replacement method.

3.4. Data Preprocessing

Data preprocessing involves transforming raw data to well-formed data sets so that data mining analytics can be applied. Raw data is often incomplete and has inconsistent formatting. The adequacy or inadequacy of data preparation has a direct correlation with the success of any project that involves data analytics.

Preprocessing involves both data validation and data imputation. The goal of data validation is to assess whether the data in question is both complete and accurate. The goal of data imputation is to correct errors and input missing values -- either manually or automatically through business process automation (BPA) programming.

Data preprocessing is used in both database-driven and rules-based applications. In machine learning (ML) processes, data preprocessing is critical for ensuring large datasets are formatted in such a way that the data they contain can be interpreted and parsed by learning algorithms. (Data preprocessing, 2021)

There are different data preprocessing. These transactions will not be used as there is no missing or meaningless data in our dataset. The first of these operations is to fill in the empty data. There are empty fields in the basket.csv file, but these empty fields are not meaningless. The second process is to make meaningless data meaningful. For example, the age attribute is negative. Another process is the inconsistency of the data. An example of this is to write the age attribute as the date of birth or the direct age. These two data may have the same meaning in the end, but they are different in spelling. Finally, in large datasets, the same data may be repeated and this may affect the result. When we look at close values, since it is the same data, it creates errors in operations such as averaging close data, so we delete the duplicate data.

In order to adapt the classification algorithms of the Groceries data.csv dataset, we subjected the dataset to some preprocessing. The new dataset we wanted to create was a dataset that would show whether customers are regular customers and what products they buy. We developed an algorithm for this process using Python.(Codes will be in attached as preprocessing_for_classification.py) First of all, we deleted the year, month, day and day_of_week features in the data set that we do not need to use. Then we calculated the number of times the customer came to the store using the Member_number and date data. In our opinion, more than 50% of the customers who come to this store can be counted as

regular customers. We found the rate of customers who shopped from the store at least 4 times as 55% and labeled these customers as regular customers.

Then we also deleted the date feature. Then we created a table structure so that all products in the data set are features. Features of our new data became Member_number, all products in the data set and class.When we uploaded the dataset to the WEKA, we realized that the member_number was also unnecessary and deleted it. We used this new data set for classification algorithms.

3.5. Data Transformation

Some operations for Data Transformation operation are explained below:

➔ Normalization: Normalization process is not used for this data set. The normalization technique is used to assign numerical data to a certain range and to make the relationship between the data consistent. In our data, there is no feature suitable for this.

When simulating the clustering process, the data overlaps and an intelligible image cannot be produced. That's why 0-1 data needs to be changed before some operations. We found that 2 different methods can be used when normalizing the cluster plot.

The first is to digitize a multidimensional set by making it 2D. To do this, sklearn.decomposition.PCA is used. Direct dimensionality decrease utilizing Singular Value Decomposition of the information to extend it to a lower layered space. The information is focused however not scaled for each component prior to applying the SVD (sklearn.decomposition.PCA, 2021).

The second is sklearn.preprocessing.StandardScaler. Focusing and scaling happen autonomously on each component by registering the applicable insights on the examples in the preparation set. Mean and standard deviation are then put away to be utilized on later information utilizing change.

Normalization of a dataset is a typical necessity for some, AI assessors: they may act seriously assuming the singular elements don't pretty much look like standard regularly appropriated information (for example Gaussian with 0 mean and unit fluctuation).

For example numerous components utilized in the genuine capacity of a learning calculation (like the RBF part of Support Vector Machines or the L1 and L2 regularizers of straight models) accept that all highlights are revolved around 0 and have difference in a similar request. Assuming an element has a fluctuation that is significant degrees bigger that others, it may overwhelm the true capacity and make the assessor incapable to gain from different elements accurately true to form (sklearn.preprocessing.StandardScaler, 2021).

Two normalization methods, StandardSclared, were used in this project. Because reducing the size of our data caused us to lose some data. StandardScaler was the best solution for normalization.

➔ Discretization: This is the process of dividing continuous data into certain intervals. Here, we have defined the months as seasons by dividing them into certain intervals and giving them names.

When creating the data set we used for Classification, the values of the products were 0 and 1 s. We applied discretization to these products using the WEKA application. Thus, the results we will obtain in the algorithms we will use have become more readable and stable.

3.6. Data Visualization

Here we will talk about showing our data with some features after going through the techniques.

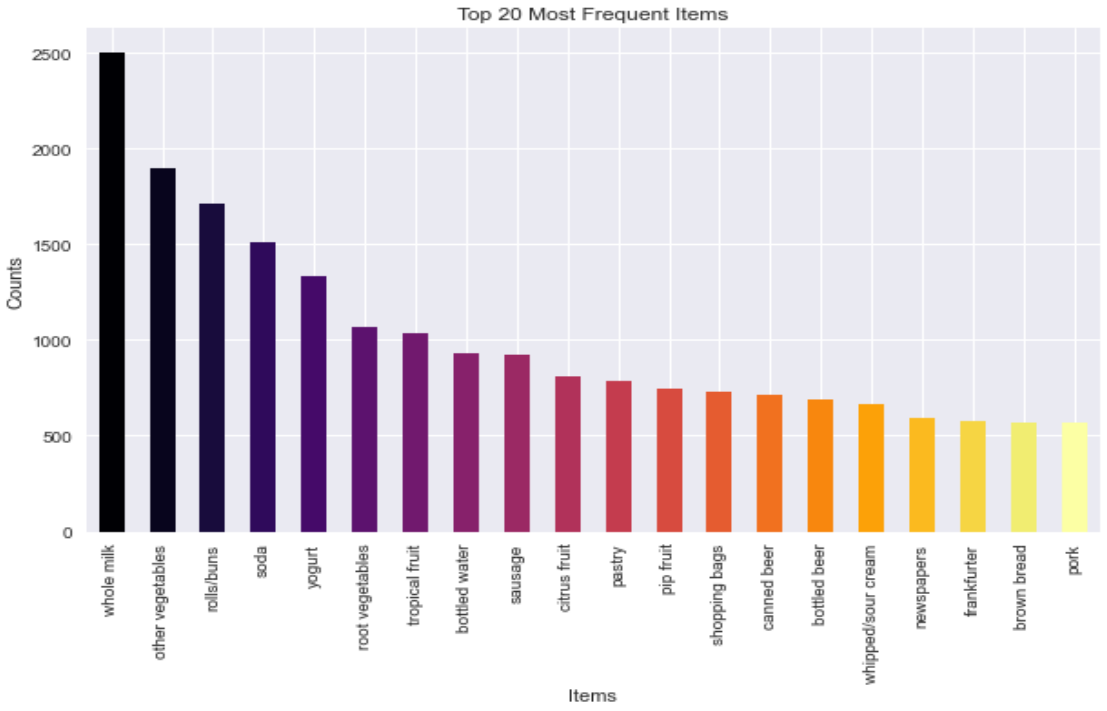


Figure 3.6.1: Top 20 Frequent Items

We can clearly see that the most bought item is whole milk. We can also say that this bar plot shows the products that have a high level of relations to be determined.

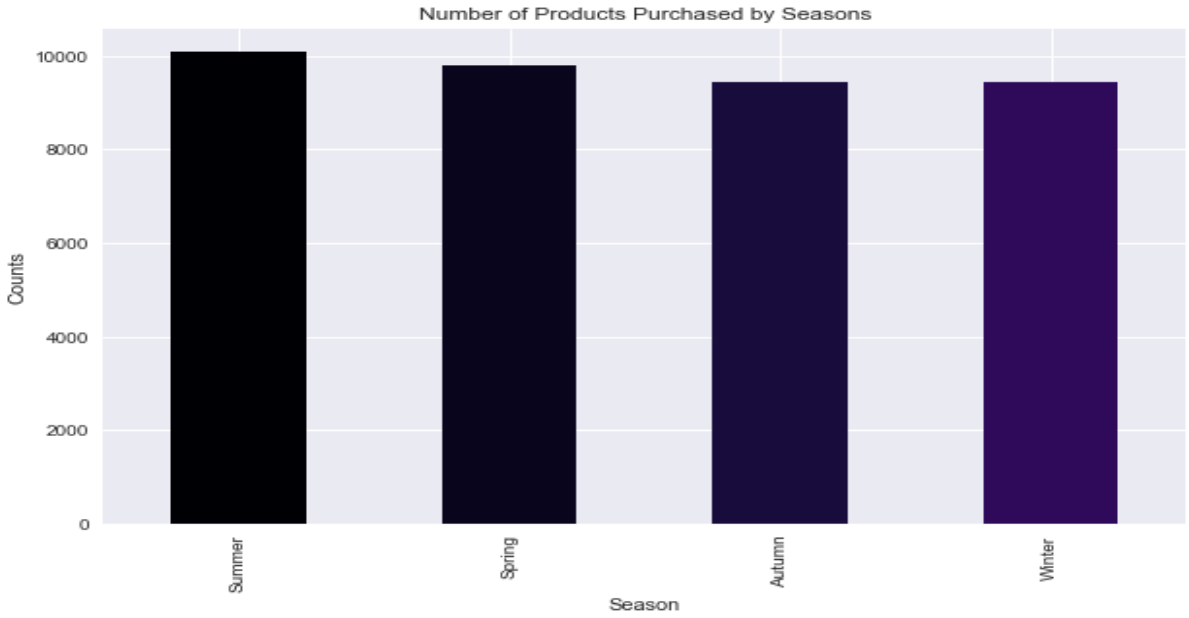


Figure 3.6.2: Number of Products Purchased by Seasons

When looking at the seasons, it is clear that sales are approximately close to each other in all seasons.

CHAPTER FOUR

TOOLS

4.1. Anaconda

Anaconda is an environment manager for data science and similar applications in this field. This open source structure currently has more than 25 million users around the world. It is prepared to run Python/R codes that are frequently used in data science and machine learning fields. It has more than 7,500 open source packages and libraries in accordance with the algorithm you will work in or the subject you need. We can use them by including them in the project with the help of download commands. Also,Anaconda supports cross-platform. In Anaconda, build machine learning models. Anaconda has a desktop GUI. In this GUI, some platforms can be shown which are Jupyter Notebook, Spyder, Pycharm,JupyterLab.In this project, Spyder is used. Spyder also supports visualization of the data set. Some popular machine learning libraries in spyder are: scipy, numpy, matplotlib, pandas, sklearn. These libraries are also preferred in Python.

4.2. Python

We thought that we could use two different languages for data mining algorithms. One of them is Python and one is R. R is chiefly utilized for measurable investigation while Python gives a more broad way to deal with information science. R and Python requires a period of speculation, and such extravagance isn't accessible for everybody. Python is a broadly useful language with a comprehensible sentence structure. R, be that as it may, is worked by analysts and envelops their particular language (R Vs Python: What’s the Difference?, 2021).

Some of the reasons for choosing the Python language;

- R is hard to learn toward the start while Python is Linear and smooth to learn.
- R is coordinated to Run locally while Python is very much incorporated with applications.
- Creating new models from scratch is easy in Python.

After these analyzes, we decided that it was python. Python is a deciphered, object-arranged, significant level programming language with dynamic semantics. Its

significant level of implicit information structures, joined with dynamic composing and dynamic restricting, make it exceptionally appealing for Rapid Application Development, just as for use as a pre-arranging or paste language to associate existing parts together. Python's straightforward, simple to learn grammar stresses comprehensibility and accordingly lessens the expense of program upkeep. Python upholds modules and bundles, which energizes program particularity and code reuse. The Python mediator and the broad standard library are accessible in source or paired structure without charge for every single significant stage, and can be openly appropriated (What is Python? Executive Summary, 2021).

Libraries used in python language;

➔ *import pandas as pd*

pandas is a product library composed for the Python programming language for information control and examination. Specifically, it offers information constructions and activities for controlling mathematical tables and time series. It is free programming delivered under the three-proviso BSD license. The name is gotten from the expression "panel data", an econometrics term for informational indexes that incorporate perceptions throughout numerous time spans for similar people (pandas(software), 2021). It is used in this project to convert csv file to dataFrame type.

➔ *import matplotlib.pyplot as plt*

matplotlib.pyplot is a state-based point of interaction to matplotlib. It gives an implied, MATLAB-like, method of plotting. It additionally opens figures on your screen, and goes about as the figure GUI supervisor. The express (object-arranged) API is suggested for complex plots, however pyplot is still typically used to make the figure and regularly the tomahawks in the figure (matplotlib.pyplot, 2021). It is used in this project to simulate data.

➔ *from sklearn.cluster import KMeans*

Clustering of unlabeled information can be performed with the module sklearn.cluster. Each clustering calculation comes in two variations: a class, that carries out the fit technique to become familiar with the clustering on train information, and a capacity, that, given train information, returns a variety of whole number names compared to

the various clustering. For the class, the labels over the preparation information can be found in the labels_ trait. (Clustering, 2021). Imported by sklearn.cluster to use KMeans.

➔ *from sklearn.preprocessing import StandardScaler*

Normalize highlights by eliminating the mean and scaling to unit difference.

The standard score of an example **x** is determined as:

z = (x - u)/s

where **u** is the mean of the preparation tests or zero if *with_mean=False*, and **s** is the standard deviation of the preparation tests or one if *with_std=False*.

Focusing and scaling happen autonomously on each element by figuring the significant measurements on the examples in the preparation set. Mean and standard deviation are then put away to be utilized on later information utilizing change.

Normalization of a dataset is a typical necessity for some, AI assessors: they may act gravely in the event that the singular highlights don't pretty much look like standard ordinarily circulated information (for example Gaussian with 0 mean and unit change) (sklearn.preprocessing.StandardScaler, 2021).

4.3. WEKA

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. We chose WEKA because of its ease of use and the adequacy of the results it gives.

We used WEKA for classification algorithms. We run our dataset on all tree structures that are already defined in Weka. The algorithms that gave the most correct results were Logistic Model Tree and Random Forest algorithms. We show the results of these two algorithms below. At WEKA, we also performed data preprocessing and data transformation. We used the discritize filtering available in weka for classification. We used the WEKA to delete the Member_number feature in the dataset that we will not use.

CHAPTER FIVE

RESULTS

5.1. Association Rule Mining Results

Association Mining Algorithms were run with Spyder Anaconda. The results are described in this section. Association rule mining has been chosen because it is a method that includes the most appropriate algorithm for the data set we have and can provide us with important information about the relationships.Our aim is to examine the relations of products with each other. Many algorithms have been developed for this. Below, we will examine the implementation of only a few of these algorithms on our dataset with different parameters.When Association Mining will be applied, there is no train set, test set and validation set separation. Because the purpose of these algorithms is to discover the relationships between the data. Therefore, there will be no performance measure after the model implementation.

5.1.1. Apriori

- Method 1: Apyori Module:

The dataset we always use when working with Association Rule Mining will be basket.csv. The main reason for this is that it is a data set that gives the information we need in a simple way. Data sets are examined in detail in the Description of Dataset title.

This module works with a dataframe. For this reason, we have converted the data in our csv file into a dataframe with its values that appear as null. In Figure 5.1.1, it is seen that our data has been transformed into a dataframe and 0 is assigned to its empty places.

index	0	1	2	3	4	5	6	7	8	9	10
0	whole milk	pastry	salty snack	0	0	0	0	0	0	0	0
1	sausage	whole milk	semi-finished bread	yogurt	0	0	0	0	0	0	0
2	soda	pickled vegetables	0	0	0	0	0	0	0	0	0
3	canned beer	misc. beverages	0	0	0	0	0	0	0	0	0
4	sausage	hygiene articles	0	0	0	0	0	0	0	0	0
5	sausage	whole milk	rolls/buns	0	0	0	0	0	0	0	0
6	whole milk	soda	0	0	0	0	0	0	0	0	0
7	frankfurter	soda	whipped/sour cream	0	0	0	0	0	0	0	0
8	frankfurter	curd	0	0	0	0	0	0	0	0	0
9	beef	white bread	0	0	0	0	0	0	0	0	0
10	butter	whole milk	0	0	0	0	0	0	0	0	0

Figure 5.1.1.1: Basket.csv convert the DataFrame

This module does not work directly with the dataframe we created. It has its own format. According to this format, our data should be stored as a list. In Figure 5.1.2, it is arranged in accordance with this format.

0	list	3	['whole milk', 'pastry', 'salty snack']
1	list	4	['sausage', 'whole milk', 'semi-finished bread', 'yogurt']
2	list	2	['soda', 'pickled vegetables']
3	list	2	['canned beer', 'misc. beverages']
4	list	2	['sausage', 'hygiene articles']
5	list	3	['sausage', 'whole milk', 'rolls/buns']
6	list	2	['whole milk', 'soda']
7	list	3	['frankfurter', 'soda', 'whipped/sour cream']
8	list	2	['frankfurter', 'curd']
9	list	2	['beef', 'white bread']
10	list	2	['butter', 'whole milk']

Figure 5.1.1.2: List Format for the Module

Now we are ready to apply the Apriori algorithm on the data we have. If we look at some preliminary data analysis before carrying out this application, there are 167 different products in the data we have. It is very difficult to determine the relationships between these products. The main reason for this is that relationships increase exponentially depending on the number of items we have. We can know the number of rules created by the products we have with the help of the permutation function in the itertools module in python. For example, there are 27722 rules created by 167 products in pairs with each other, and there are 4574130 rules in triples.Considering these numbers, the working speed of the apriori algorithm will be greatly affected. However, these rules may not be necessary for us. These numbers should decrease with certain performance metrics and pruning operations. Otherwise, the working speed of the apriori algorithm will be greatly reduced. For our first method, we can determine the

minimum support value, minimum confidence value and minimum lift value to ensure that the most accurate and necessary information is found for us. The larger the support and confidence from these values, the stronger the association rule. One of the most challenging steps for Apriori is determining these values. Because if these values are too large, some important relationships will be missed, and if they are too small, we will get misleading results with some unnecessary relationship rules.The explanations of the confidence metrics we prefer for this algorithm are given below.

- ☐ Support:Indicates the rate at which a relationship is repeated in all purchases.

$$\frac{number\ of\ transactions\ with\ items(s)}{number\ of\ transactions}$$

- ☐ Confidence: It indicates the probability that the customer who buys product X will buy product Y.

$$\frac{Support\ X\&Y}{SupportX}$$

- ☐ Lift:It is another metric that calculates relationships between X and Y. If the lift value is greater than 1, these two products appear together more often than expected and have a positive effect on sales, if it is less than 1, they appear together less frequently and have a negative effect, if values close to 1, they appear together as expected and have no effect. indicates not.

$$\frac{Support(X\&Y)}{Support(X)Support(Y)}$$

The default values for this library are 0.1, 0.0, 0.0, None for support, confidince, lift and maximum length parameters, respectively. Below are some results according to these values.

Item	Support	Confidence	Lift	Maximum Length
Other Vegetable	0.122	0.122	1.0	None
Rolls/Buns	0.11	0.11	1.0	None
Whole milk	0.15	0.15	1.0	None

Table 5.1.1.1: The results of Apyroi Module with default values.

As seen in Table 5.1.1, when the first method runs in Spyder, the values of confidence and support look the same. When I researched this issue, I saw that it was prepared for the apriori python version 2.7. For this reason, Python 3 and higher versions cannot give correct results, as seen above.

- Method 2: mlxtend Module

27722 rules created by the 167 products we calculated above in pairs with each other, and the rule 4574130 in triples are valid in this module and all the pruning metrics we mentioned above are also used in this module. But in addition to these, there are some more pruning metrics. I will talk about these metrics that we can use in this module. The first of these metrics is Leverage.

☐ Leverage measures what to expect when product X and product Y are statistically dependent, and the difference between X and Y.

$$Leverage(X \rightarrow Y) = Support(X\&Y) - Support(X)Support(Y)$$

☐ Conviction, on the other hand, compares the probability that product X will appear without product Y if the probability of X appearing without product Y depends on the actual frequency of appearance without Y.

$$Conviction(X \rightarrow Y) = Support(X)Support(Y)/Support(X\&Y)$$

For us, these metrics were less important. The main reason for this is that we do not take into account the independence or dependence between products. In fact, we will go through the metrics we defined in the previous module.

As we have defined above, the data set is defined on python, but this module cannot be used directly with the format we defined. This module expects those products to be defined as an attribute. According to whether it is in the market basket or not, it defines it as binary (True or False) and asks it to be converted into a table. The table where these operations are applied is also called one-hot encoding. Figure 5.1.3 shows one hot encoding application.

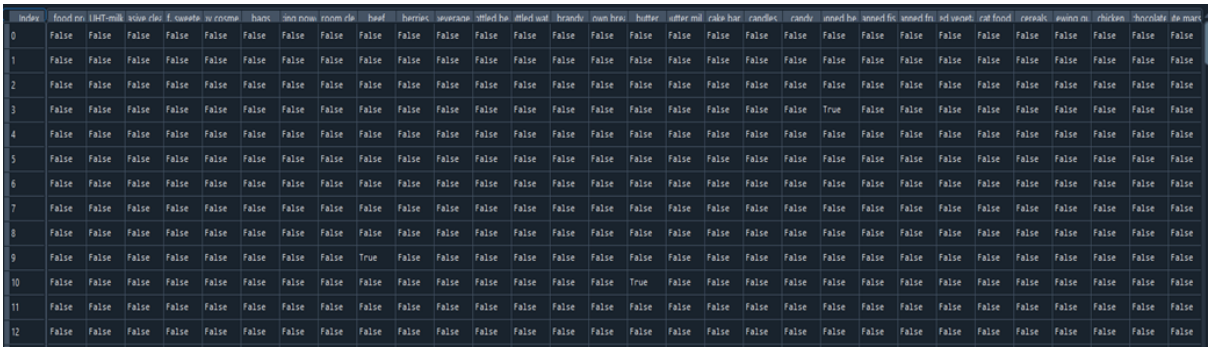
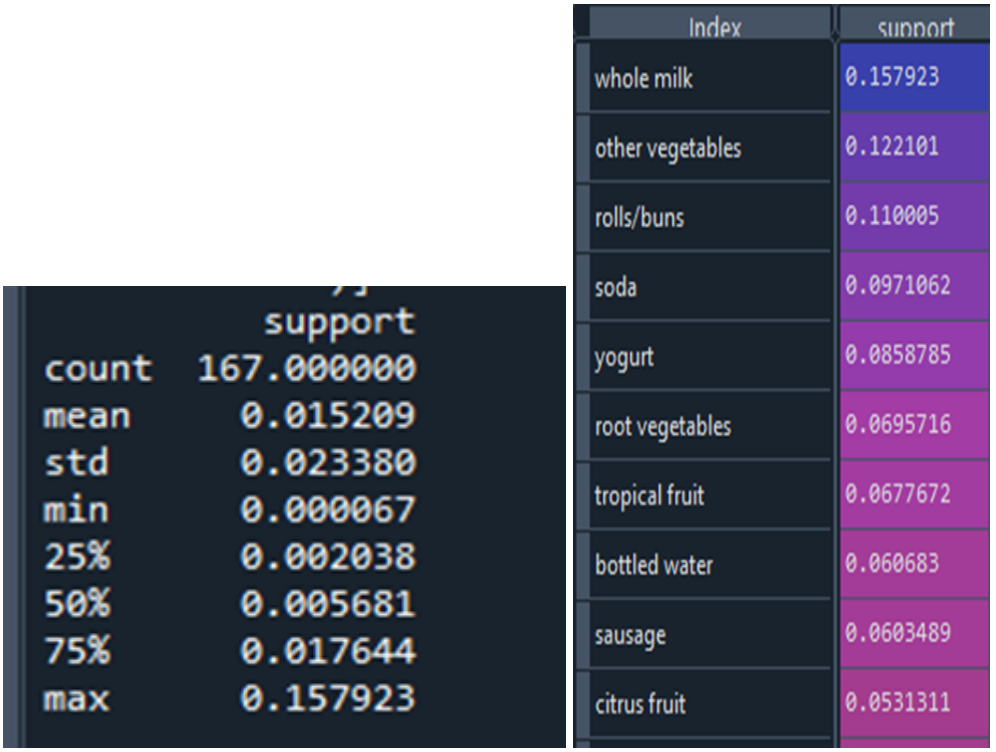


Figure 5.1.1.3: One-hot Encoding Table

We mentioned that it is quite challenging to decide on the values in the upper method. We found the support values of each product before deciding on the minimum support value. Figure 5.1.4 shows these values in the table in order from most to least.



Index	support
whole milk	0.157923
other vegetables	0.122101
rolls/buns	0.110005
soda	0.0971062
yogurt	0.0858785
root vegetables	0.0695716
tropical fruit	0.0677672
bottled water	0.060683
sausage	0.0603489
citrus fruit	0.0531311

	support
count	167.000000
mean	0.015209
std	0.023380
min	0.000067
25%	0.002038
50%	0.005681
75%	0.017644
max	0.157923

Figure 5.1.1.5:Statistical Analysis of Support for each Product Figure 5.1.1.4: Support Values for Each Product

We can start by setting the initial support values to a minimum of 0.017644. When no confidence value or lift was specified, 42 rules were selected from 27722 rules. However, within these rules, there are also values that provide support value alone. If we want to look at the bilateral relations, we will see that there is no bilateral relationship, so let's update the support value to 0.008. This time we can see that 82 rules have been selected. Among these rules are binary rules. As seen in Figure 5.1.6, there are 9 rules.

0.0105594	frozenset({'other vegetables', 'rolls/buns'})	2
0.00969057	frozenset({'other vegetables', 'soda'})	2
0.0148366	frozenset({'other vegetables', 'whole milk'})	2
0.00808661	frozenset({'other vegetables', 'yogurt'})	2
0.00808661	frozenset({'soda', 'rolls/buns'})	2
0.0139678	frozenset({'whole milk', 'rolls/buns'})	2
0.00895542	frozenset({'whole milk', 'sausage'})	2
0.0116287	frozenset({'whole milk', 'soda'})	2
0.00822028	frozenset({'tropical fruit', 'whole milk'})	2
0.0111609	frozenset({'yogurt', 'whole milk'})	2

Figure 5.1.1.6: Support Value is 0.008 and shown 2 length rules.

So let's support these values a little more with confidence and lift operations. Since these transactions represent relations between products, bilateral relations will now be examined. In the figure 5.1.7, the procedures for these 9 rules are given.

index	antecedents	consequents	cedent sup	equent sup	support	confidence	lift	leverage	conviction
0	frozenset({'other vegetables'})	frozenset({'rolls/buns'})	0.122101	0.110005	0.0105594	0.0864806	0.786154	-0.00287232	0.974249
1	frozenset({'rolls/buns'})	frozenset({'other vegetables'})	0.110005	0.122101	0.0105594	0.0959903	0.786154	-0.00287232	0.971117
2	frozenset({'soda'})	frozenset({'other vegetables'})	0.0971062	0.122101	0.00969057	0.0997935	0.817302	-0.00216621	0.975219
3	frozenset({'other vegetables'})	frozenset({'soda'})	0.122101	0.0971062	0.00969057	0.0793651	0.817302	-0.00216621	0.980729
4	frozenset({'other vegetables'})	frozenset({'whole milk'})	0.122101	0.157923	0.0148366	0.121511	0.76943	-0.00444597	0.958551
5	frozenset({'whole milk'})	frozenset({'other vegetables'})	0.157923	0.122101	0.0148366	0.0939484	0.76943	-0.00444597	0.968928
6	frozenset({'yogurt'})	frozenset({'other vegetables'})	0.0858785	0.122101	0.00808661	0.0941634	0.771192	-0.00239925	0.969158
7	frozenset({'other vegetables'})	frozenset({'yogurt'})	0.122101	0.0858785	0.00808661	0.0662288	0.771192	-0.00239925	0.978957
8	frozenset({'soda'})	frozenset({'rolls/buns'})	0.0971062	0.110005	0.00808661	0.083276	0.757022	-0.00259552	0.970843
9	frozenset({'rolls/buns'})	frozenset({'soda'})	0.110005	0.0971062	0.00808661	0.0735115	0.757022	-0.00259552	0.974533
10	frozenset({'rolls/buns'})	frozenset({'whole milk'})	0.110005	0.157923	0.0139678	0.126974	0.804028	-0.00340447	0.96455
11	frozenset({'whole milk'})	frozenset({'rolls/buns'})	0.157923	0.110005	0.0139678	0.0884469	0.804028	-0.00340447	0.97635
12	frozenset({'sausage'})	frozenset({'whole milk'})	0.0603489	0.157923	0.00895542	0.148394	0.939663	-0.000575042	0.988811
13	frozenset({'whole milk'})	frozenset({'sausage'})	0.157923	0.0603489	0.00895542	0.0567076	0.939663	-0.000575042	0.99614
14	frozenset({'soda'})	frozenset({'whole milk'})	0.0971062	0.157923	0.0116287	0.119752	0.758296	-0.00370661	0.956636
15	frozenset({'whole milk'})	frozenset({'soda'})	0.157923	0.0971062	0.0116287	0.0736352	0.758296	-0.00370661	0.974663
16	frozenset({'tropical fruit'})	frozenset({'whole milk'})	0.0677672	0.157923	0.00822028	0.121302	0.768108	-0.00248171	0.958323
17	frozenset({'whole milk'})	frozenset({'tropical fruit'})	0.157923	0.0677672	0.00822028	0.0520525	0.768108	-0.00248171	0.983422
18	frozenset({'yogurt'})	frozenset({'whole milk'})	0.0858785	0.157923	0.0111609	0.129961	0.82294	-0.00240132	0.967861
19	frozenset({'whole milk'})	frozenset({'yogurt'})	0.157923	0.0858785	0.0111609	0.0706729	0.82294	-0.00240132	0.983638

Figure 5.1.1.7: The association rules between two items.

Figure 5.1.7 when examined carefully, the first thing to notice is that the lift value is not seen as 1 or -1. So, by changing with the parameters a little more, it is important for us to find the most accurate relationships by taking into account the lift confidence and support values, by obtaining a table similar to the one above, so let's compare the results with the changing parameters in the table below.First of all, the data set we have is the annual data set. According to this data set, if a product was purchased 3 times in a month, then it was taken 3x30=90 times monthly. If we divide this data by the total number of datasets, 14963, let's determine our initial support value as 0.0060.

Support	Confidence	Lift	İkili İlişki Sayısı
0.0060	0.04	1	2
0.0050	0.04	1	7
0.0040	0.07	1	5

Table 5.1.1.2: The result for the different support and confidence value

Situation 1:

antecedents	consequents	cedent sup	equent sup	support	confidence	lift	leverage	conviction
frozenset({'bottled beer'})	frozenset({'whole milk'})	0.0453118	0.157923	0.00715097	0.157817	0.99933	-4.7925e-06	0.999874
frozenset({'whole milk'})	frozenset({'bottled beer'})	0.157923	0.0453118	0.00715097	0.0452814	0.99933	-4.7925e-06	0.999968

Figure 5.1.1.8: In Table 5.1.2, the first instance

Situation 2:

antecedents	consequents	cedent sup	equent sup	support	confidence	lift	leverage	conviction
frozenset({'bottled beer'})	frozenset({'whole milk'})	0.0453118	0.157923	0.00715097	0.157817	0.99933	-4.7925e-06	0.999874
frozenset({'whole milk'})	frozenset({'bottled beer'})	0.157923	0.0453118	0.00715097	0.0452814	0.99933	-4.7925e-06	0.999968
frozenset({'other vegetables'})	frozenset({'frankfurter'})	0.122101	0.0377598	0.00514603	0.0421456	1.11615	0.00053551	1.00458
frozenset({'frankfurter'})	frozenset({'other vegetables'})	0.0377598	0.122101	0.00514603	0.136283	1.11615	0.00053551	1.01642
frozenset({'soda'})	frozenset({'sausage'})	0.0971062	0.0603489	0.00594801	0.0612526	1.01497	8.77568e-05	1.00096
frozenset({'sausage'})	frozenset({'soda'})	0.0603489	0.0971062	0.00594801	0.0985604	1.01497	8.77568e-05	1.00161
frozenset({'yogurt'})	frozenset({'sausage'})	0.0858785	0.0603489	0.00574751	0.0669261	1.10899	0.000564841	1.00705
frozenset({'sausage'})	frozenset({'yogurt'})	0.0603489	0.0858785	0.00574751	0.0952381	1.10899	0.000564841	1.01034

Figure 5.1.1.9: In Table 5.1.2, the second instance

Situation 3:

frozenset({'bottled beer'})	frozenset({'whole milk'})	0.0453118	0.157923	0.00715097	0.157817	0.99933	-4.7925e-06	0.999874
frozenset({'frankfurter'})	frozenset({'other vegetables'})	0.0377598	0.122101	0.00514603	0.136283	1.11615	0.00053551	1.01642
frozenset({'sausage'})	frozenset({'soda'})	0.0603489	0.0971062	0.00594801	0.0985604	1.01497	8.77568e-05	1.00161
frozenset({'sausage'})	frozenset({'yogurt'})	0.0603489	0.0858785	0.00574751	0.0952381	1.10899	0.000564841	1.01034
frozenset({'citrus fruit'})	frozenset({'yogurt'})	0.0531311	0.0858785	0.00461137	0.0867925	1.01064	4.85593e-05	1.001

Figure 5.1.1.8: In Table 5.1.3, the third instance

When we examine the relations of the above products with each other, we can make some comments for 3 cases. Since the support value shows how much these two products are seen together, we can see that they are seen together with 0.05 close to the previous values. Unlike the other two cases, we have examined the cases where the probability of someone who buys the product x to buy the product y is high, and the relationships that provide this are seen above. Lift, one of our last and most valuable data, indicates the effect on the sales of product y when product x is purchased. We have mentioned that being 1 has a positive effect, so we can say that the relationships seen in the above table are the products that have the most meaningful relationship with each other.We have shown our data as scatter plot and heat map below.

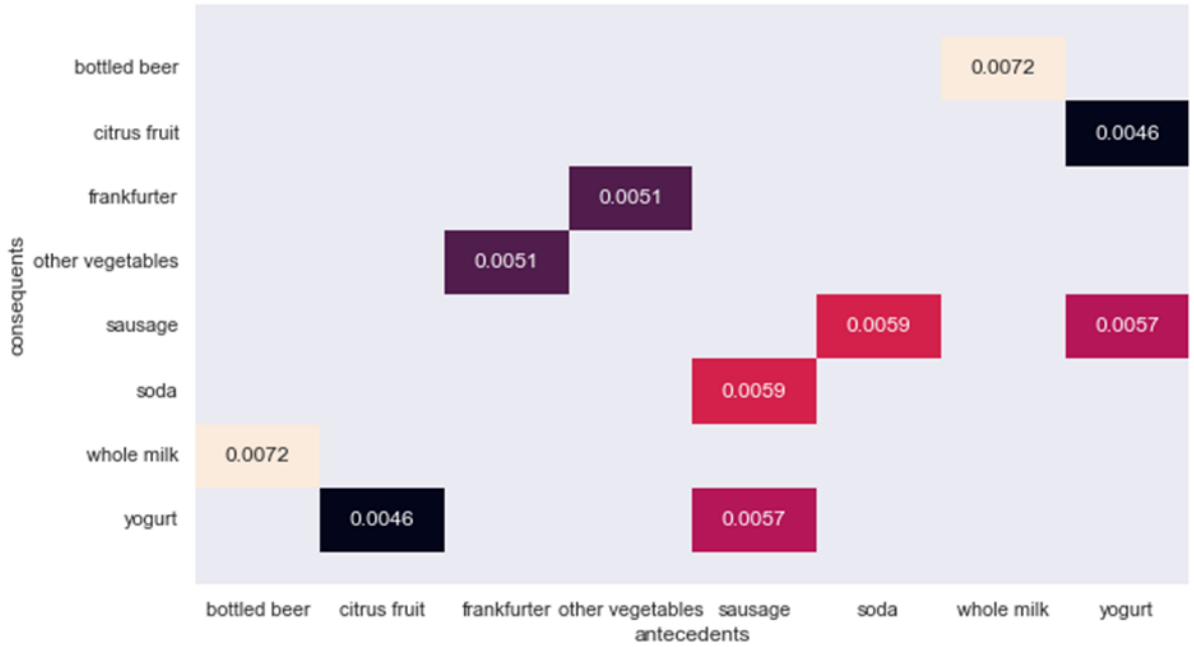


Figure 5.1.1.9: This heatmap is made according to the support values

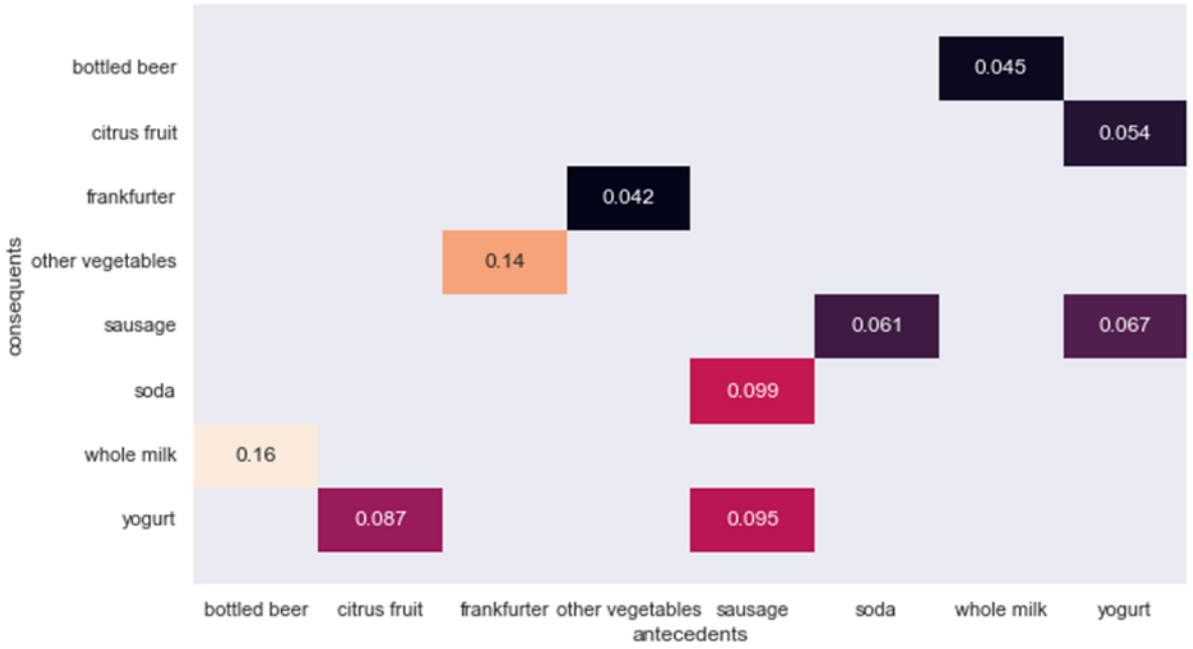


Figure 5.1.1.10: The heatmap above is based on confidence values.

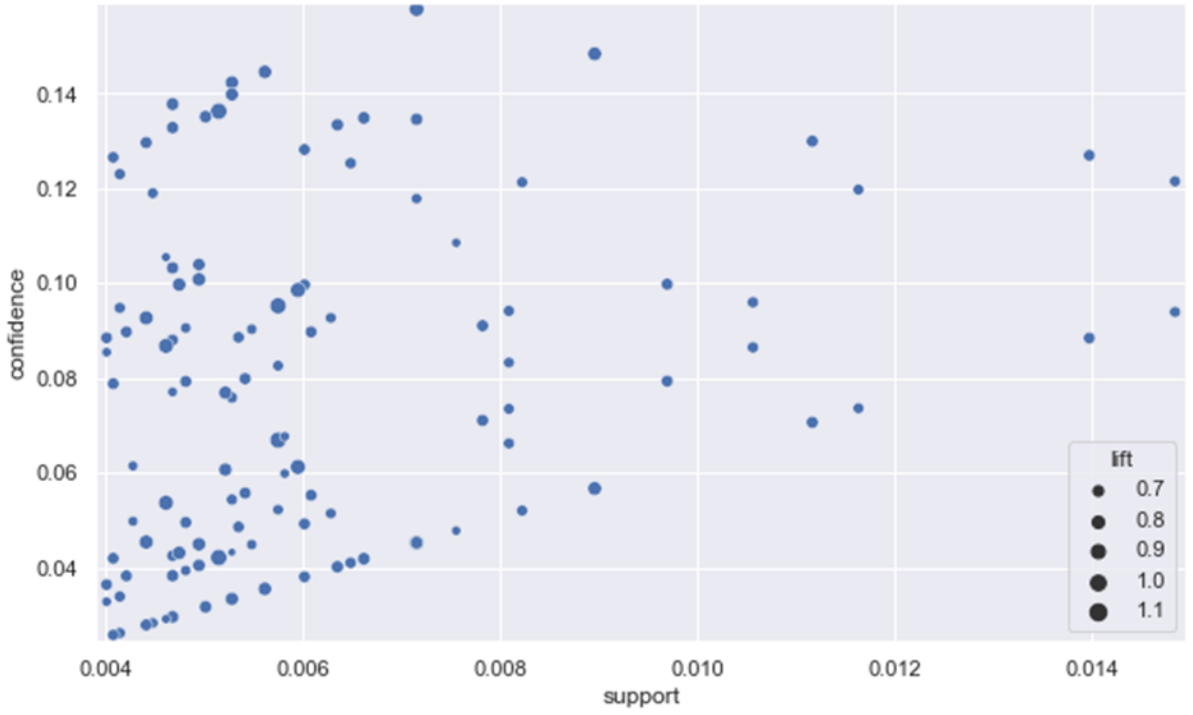


Figure 5.1.1.11: It has been seen that our values of support and confidence are in the form of a scatter plot above.

According to a study, the accuracy values for the apriori algorithm coincide with the measured confidence values. In our study, we evaluated the 3rd case as the most optimal case. For this reason, the accuracy value of the rules matches the confidence value. However,

according to the information in the articles, it is not seen as the best algorithm in terms of working speed in the apriori association mining rule. There are various different applications of the Apriori algorithm.

5.1.2. FP-Growth

FP-Growth, which stands out with its fast operation as opposed to the slow operation of the Apriori algorithm, has the same preprocessing stage as the Apriori application as stated above. The results of this algorithm are shared below:

When we apply the value of 0.017644, which is the support value that we run for the first time for the Apriori algorithm, we see that 42 values are selected again. Again, no bilateral relationship was found among these 42 values. When we run it with 0.008, we see that 9 results are given again.

antecedents	consequents	cedent sup	equent sup	support	confidence	lift	leverage	conviction
frozenset({'yogurt'})	frozenset({'whole milk'})	0.0858785	0.157923	0.0111609	0.129961	0.82294	-0.00240132	0.967861
frozenset({'whole milk'})	frozenset({'yogurt'})	0.157923	0.0858785	0.0111609	0.0706729	0.82294	-0.00240132	0.983638
frozenset({'yogurt'})	frozenset({'other vegetables'})	0.0858785	0.122101	0.00808661	0.0941634	0.771192	-0.00239925	0.969158
frozenset({'other vegetables'})	frozenset({'yogurt'})	0.122101	0.0858785	0.00808661	0.0662288	0.771192	-0.00239925	0.978957
frozenset({'sausage'})	frozenset({'whole milk'})	0.0603489	0.157923	0.00895542	0.148394	0.939663	-0.000575042	0.988811
frozenset({'whole milk'})	frozenset({'sausage'})	0.157923	0.0603489	0.00895542	0.0567076	0.939663	-0.000575042	0.99614
frozenset({'soda'})	frozenset({'whole milk'})	0.0971062	0.157923	0.0116287	0.119752	0.758296	-0.00370661	0.956636
frozenset({'whole milk'})	frozenset({'soda'})	0.157923	0.0971062	0.0116287	0.0736352	0.758296	-0.00370661	0.974663
frozenset({'other vegetables'})	frozenset({'soda'})	0.122101	0.0971062	0.00969057	0.0793651	0.817302	-0.00216621	0.980729
frozenset({'soda'})	frozenset({'other vegetables'})	0.0971062	0.122101	0.00969057	0.0997935	0.817302	-0.00216621	0.975219
frozenset({'rolls/buns'})	frozenset({'soda'})	0.110005	0.0971062	0.00808661	0.0735115	0.757022	-0.00259552	0.974533
frozenset({'soda'})	frozenset({'rolls/buns'})	0.0971062	0.110005	0.00808661	0.083276	0.757022	-0.00259552	0.970843
frozenset({'rolls/buns'})	frozenset({'whole milk'})	0.110005	0.157923	0.0139678	0.126974	0.804028	-0.00340447	0.96455
frozenset({'whole milk'})	frozenset({'rolls/buns'})	0.157923	0.110005	0.0139678	0.0884469	0.804028	-0.00340447	0.97635
frozenset({'rolls/buns'})	frozenset({'other vegetables'})	0.110005	0.122101	0.0105594	0.0959903	0.786154	-0.00287232	0.971117
frozenset({'other vegetables'})	frozenset({'rolls/buns'})	0.122101	0.110005	0.0105594	0.0864806	0.786154	-0.00287232	0.974249
frozenset({'other vegetables'})	frozenset({'whole milk'})	0.122101	0.157923	0.0148366	0.121511	0.76943	-0.00444597	0.958551
frozenset({'whole milk'})	frozenset({'other vegetables'})	0.157923	0.122101	0.0148366	0.0939484	0.76943	-0.00444597	0.968928
frozenset({'tropical fruit'})	frozenset({'whole milk'})	0.0677672	0.157923	0.00822028	0.121302	0.768108	-0.00248171	0.958323
frozenset({'whole milk'})	frozenset({'tropical fruit'})	0.157923	0.0677672	0.00822028	0.0520525	0.768108	-0.00248171	0.983422

Figure 5.1.2.1: Support value is 0.008 and see two associations.

When the results are compared with Apriori, although the relationships are the same, as can be seen, the pruning metrics have small differences.

Support	Confidence	Lift	İkili İlişki Sayısı
0.0060	0.04	1	2
0.0050	0.04	1	7
0.0040	0.07	1	5

Table 5.1.2.1: The result for the different support and confidence value

When the support value was run as 0.04 and no lift and confidence values were examined, 163 rules were found. Only 60 of these rules have a binary relationship. For the situation 3 comparing the results for the case:

antecedents	consequents	cedent sup	equent sup	support	confidenc	lift	leverage	conviction
frozenset({'bottled beer'})	frozenset({'whole milk'})	0.0453118	0.157923	0.00715097	0.157817	0.99933	-4.7925e-06	0.999874
frozenset({'frankfurter'})	frozenset({'other vegetables'})	0.0377598	0.122101	0.00514603	0.136283	1.11615	0.00053551	1.01642
frozenset({'sausage'})	frozenset({'soda'})	0.0603489	0.0971062	0.00594801	0.0985604	1.01497	8.77568e-05	1.00161
frozenset({'sausage'})	frozenset({'yogurt'})	0.0603489	0.0858785	0.00574751	0.0952381	1.10899	0.000564841	1.01034
frozenset({'citrus fruit'})	frozenset({'yogurt'})	0.0531311	0.0858785	0.00461137	0.0867925	1.01064	4.85593e-05	1.001

Figure 5.1.2.2: Result for the situation 3 in table 5.1.2.1

For fp-growth, just as in apriori, the precision of the rules is directly related to the confidence value. In line with the studies carried out, the main points to decide on the performance of both algorithms were especially space and speed. As a result of studies on this, traditional apriori has a very slow performance, while fp-growth has a very good performance.

As a result, even though the operating speeds and performances of the algorithms are different from each other, apriori and fp growth gave the same results. However, there are minor variations between the two algorithms. While calculating these results, the lift parameter especially stood out for us. Calculating the positive contribution of two products to each other's sales is our main priority.

5.2. Classification Results

We tried multiple algorithms for classification. We will share the two of them (Random Tree and Logistic Model Tree) with the most accurate results and the results we got here.

5.2.1. Random Forest Algorithm

The Random Forest method consists of assemblies of the Classification Tree or the Regression Tree in accordance with the purpose, as many as the number of trees to be created. Therefore, one of the most widely used algorithms among the ensemble methods is the Random Forest. The basic idea underlying the method is to generate ensembles with the help of a randomly selected subset from a large number of predictive trees (*Breiman, 2001*).

Random Forest Method is used in both categorical and continuous data sets and both; at the same time, it can be easily used in large or small sized data sets. The disadvantage of the method is that unlike the Classification Tree Method, it does not output a tree (*Akman et al., 2011*).

The advantage of choosing a random estimator in this way is that less correlation is obtained between the trees in the ensemble, so the accuracy of the model is higher (*Suchetana et al., 2017*).

We tried different parameters(Table 5.2.1.1) and the best results we got were as follows(Figure 5.2.1.1).

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation:     with_classes-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.supervised.attribute.Discretize-Rfirst-last-precision6
Instances:    3898
Attributes:   168
              [list of attributes omitted]
Test mode:    split 90.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 3.2 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.07 seconds

=== Summary ===

Correctly Classified Instances         326               83.5897 %
Kappa statistic                          0.6725
Mean absolute error                      0.3197
Root mean squared error                  0.3689
Relative absolute error                 64.1071 %
Root relative squared error             73.8867 %
Total Number of Instances              390

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,800    0,124    0,877      0,800    0,837      0,675    0,918     0,924     regular
                 0,876    0,200    0,798      0,876    0,835      0,675    0,918     0,916     non-regular
Weighted Avg.    0,836    0,160    0,840      0,836    0,836      0,675    0,918     0,920

=== Confusion Matrix ===

   a   b   <-- classified as
 164  41 |   a = regular
  23 162 |   b = non-regular
```

Figure 5.2.1.1: Results Of Random Forest Algorithm with 10-fold cross validation

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation:     with_classes-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.supervised.attribute.Discretize-Rfirst-last-precision6
Instances:    3898
Attributes:   168
              [list of attributes omitted]
Test mode:    split 90.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 3.2 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.07 seconds

=== Summary ===

Correctly Classified Instances         326               83.5897 %
Kappa statistic                          0.6725
Mean absolute error                      0.3197
Root mean squared error                  0.3689
Relative absolute error                 64.1071 %
Root relative squared error             73.8867 %
Total Number of Instances              390

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,800    0,124    0,877      0,800    0,837      0,675    0,918     0,924     regular
                 0,876    0,200    0,798      0,876    0,835      0,675    0,918     0,916     non-regular
Weighted Avg.    0,836    0,160    0,840      0,836    0,836      0,675    0,918     0,920

=== Confusion Matrix ===

   a   b   <-- classified as
 164  41 |   a = regular
  23 162 |   b = non-regular
```

Figure 5.2.1.2: Results Of Random Forest Algorithm with split %90 train

	Number Of Trees	Max Depth	10-Fold Validation	Split %90 Training	Time Taken	Accuracy
1	100	0	✓	X	3.71 seconds	83.76 %
2	100	0	X	✓	0.07 seconds	83.58 %
3	100	7	✓	X	1.21 seconds	84.40 %
4	100	7	X	✓	0.02 seconds	82.30 %
5	500	7	✓	X	6.45 seconds	84.83 %
6	500	7	X	✓	0.1 seconds	82.30 %
7	1000	7	✓	X	11.4 seconds	84.78 %
8	1000	7	X	✓	1.28 seconds	82.30 %

Table 5.2.1.1: Results Of Random Forest Algorithm with split %90 train

5.2.2. Logistic Model Tree Algorithm

Logistic model tree (LMT) is a classification model with an associated supervised training algorithm that combines logistic regression (LR) and decision tree learning.

Logistic model trees are based on the earlier idea of a model tree: a decision tree that has linear regression models at its leaves to provide a piecewise linear regression model (where ordinary decision trees with constants at their leaves would produce a piecewise constant model).(*Niels Landwehr, 2003*)In the logistic variant, the LogitBoost algorithm is used to produce an LR model at every node in the tree; the node is then split using the C4.5 criterion. Each LogitBoost invocation is warm-started from its results in the parent node. Finally, the tree is pruned.

```
Time taken to build model: 35.19 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3369               86.4289 %
Kappa statistic                          0.728
Mean absolute error                      0.2031
Root mean squared error                  0.3129
Relative absolute error                 40.7185 %
Root relative squared error             62.6546 %
Total Number of Instances             3898

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,865    0,137    0,875      0,865    0,870      0,728    0,940     0,943     regular
                 0,863    0,135    0,853      0,863    0,858      0,728    0,940     0,941     non-regular
Weighted Avg.    0,864    0,136    0,864      0,864    0,864      0,728    0,940     0,942

=== Confusion Matrix ===

    a    b   <-- classified as
 1773  276 |    a = regular
  253 1596 |    b = non-regular
```

Figure 5.2.2.1: Results Of LMT Algorithm with 10-fold cross validation

```
Time taken to build model: 34.25 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Correctly Classified Instances         335               85.8974 %
Kappa statistic                          0.718
Mean absolute error                      0.2072
Root mean squared error                  0.325
Relative absolute error                 41.5402 %
Root relative squared error             65.0772 %
Total Number of Instances              390

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,839    0,119    0,887      0,839    0,862      0,719    0,933     0,939     regular
                 0,881    0,161    0,832      0,881    0,856      0,719    0,933     0,932     non-regular
Weighted Avg.    0,859    0,139    0,861      0,859    0,859      0,719    0,933     0,935

=== Confusion Matrix ===

   a   b   <-- classified as
 172  33 |   a = regular
  22 163 |   b = non-regular
```

Figure 5.2.2.2: Results Of LMT Algorithm with split %90 train

5.3. Clustering Results

In this project, clustering was done according to the products purchased by a customer. Different scenarios were tried to achieve the best results.

Encountered problems and solutions;

- When we simulate the data set we have, the points appear only in 4 different sections (Figure 5.3.1: Multidimensional plot without normalization). Since the data is in the form of 0-1, different numbers cannot be obtained. As a result, it does not make sense to create multiple clusters, since clusters are formed in 4 different regions.

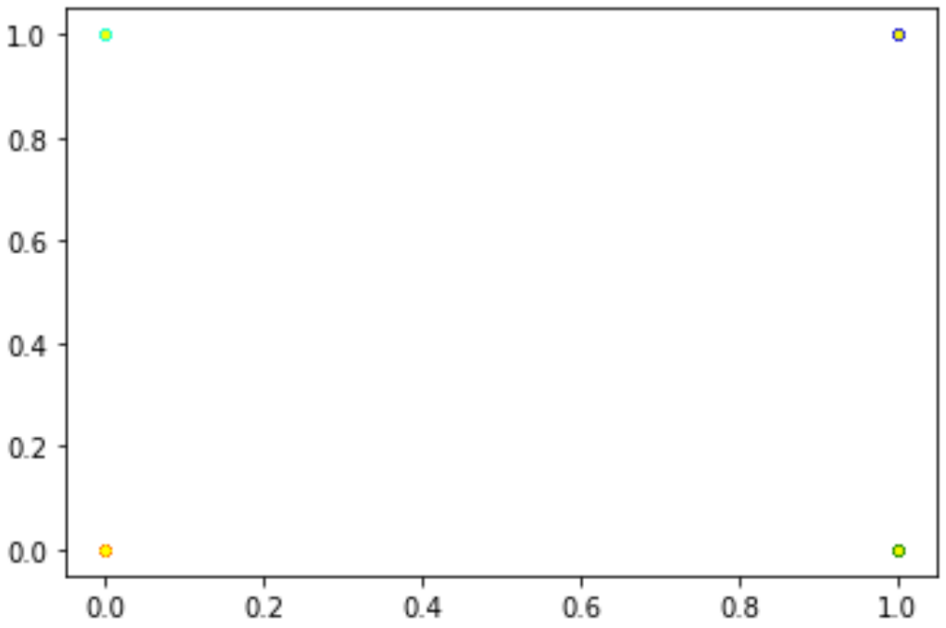


Figure 5.3.1: Multidimensional plot without normalization

- Normalization was done by reducing the size. Reduced from 167 columns to 2 columns (Figure 5.3.2: One dimensional plot without normalization). However, this was not enough because while clustering, the same clusters overlapped, although in different regions.

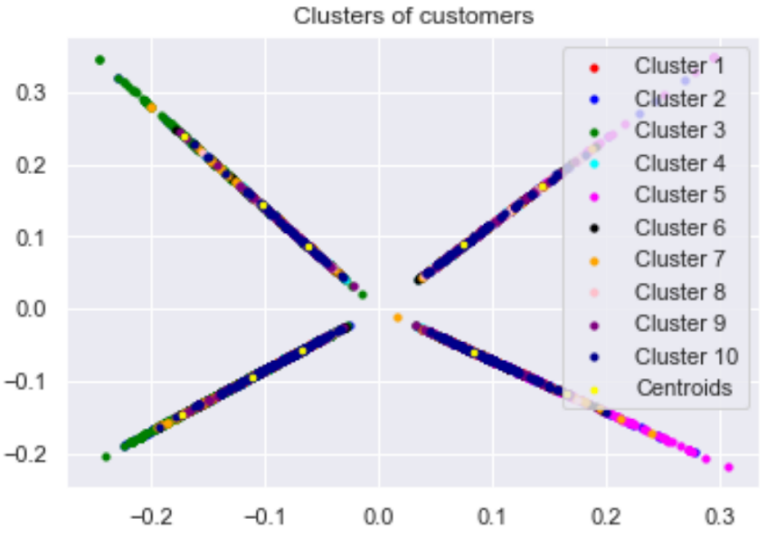


Figure 5.3.2: One-dimensional plot without normalization

- In order to solve these problems, we first converted each column of the data set consisting of 167 columns into unique numbers different from 0-1 by applying sclared. We then converted this multidimensional dataset into 2 dimensions.

- For clustering, we first tried the number of clusters as 5. But 5 clusters give us a very wide solution and this is not enough for us (Figure 5.3.3: n cluster = 5).

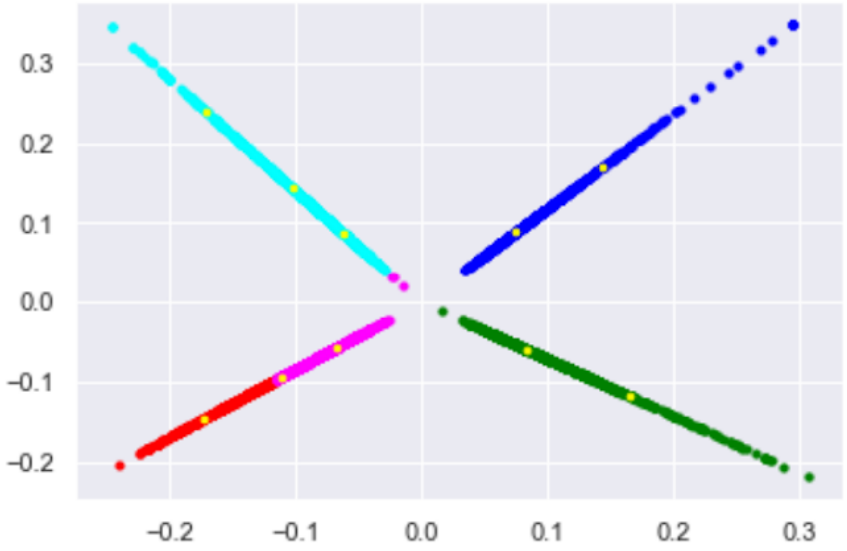


Figure 5.3.3: n cluster = 5

- After the cluster number is set to 10, it becomes a more understandable solution for us. As seen in Figure 5.3.4, 10 different clusters were created. The yellow dots show the centers. With the solutions mentioned above, more understandable clusters can appear in the drawing. As the number of clusters is increased, a more understandable drawing emerges.

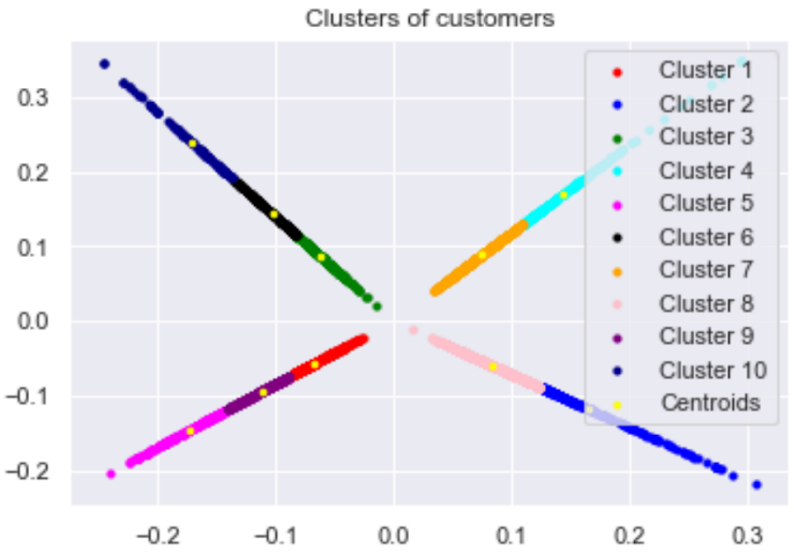


Figure 5.3.4: n cluster = 10

After different problems and solutions, testing and training phase was started. While 10% of the data set was used for testing, 90% was used for train. We can see in Figure 5.3.5 and Figure 5.3.6 that cluster centers are always shifting because the data numbers change. We can see that the center of Cluster 9 is in the regions of 0.1 / 0.2 in Figure 5.3.5, while it is in the regions of -0.1 / 0.0 in Figure 5.3.6. In this way, we can understand that cluster centers change with different numbers of data sets.

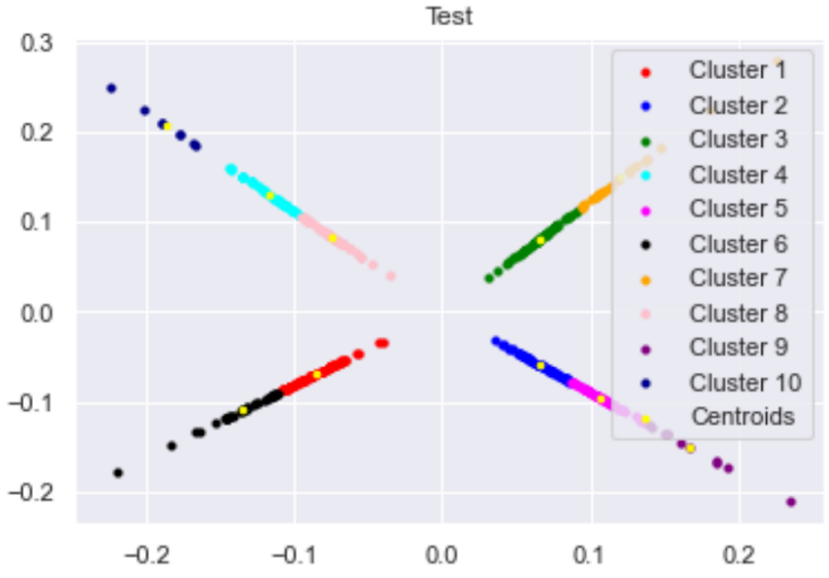


Figure 5.3.5: Test plot

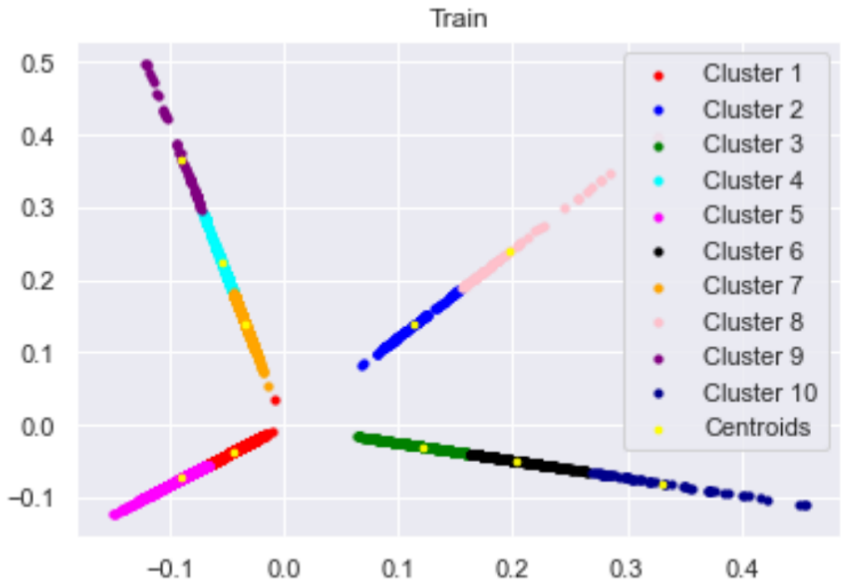


Figure 5.3.6: Train plot

By fitting the model with a range of values for K in the K-means algorithm, the elbow approach is one of the most popular methods for selecting the best number of clusters. The elbow approach is generating a line plot of SSE (Sum of Squared Errors) vs. the number of clusters and locating the "elbow point" (the point after which the SSE or inertia starts decreasing in a linear fashion). Here's an example of an elbow point. It is demonstrated in the later sections how to build the line plot and determine the elbow point.

The elbow point in the SSE / Inertia diagram depicts the point at which SSE or inertia begins to decrease linearly. In Figure 5.3.7, you can see that when the number of clusters reaches 10, the SSE begins to decrease linearly.

Here's a quick recap of what you learned in this post about identifying elbow point using the elbow method, which involves generating the SSE / Inertia plot:

- In the K-means clustering algorithm, the elbow technique is used to find the most ideal value of K, which represents the number of clusters.
- Drawing a line plot between SSE (Within-clusters Sum of Squared errors) and the number of clusters is required for the elbow approach.
- The elbow point is the point on the line plot where the SSE or inertia values begin to decrease linearly.

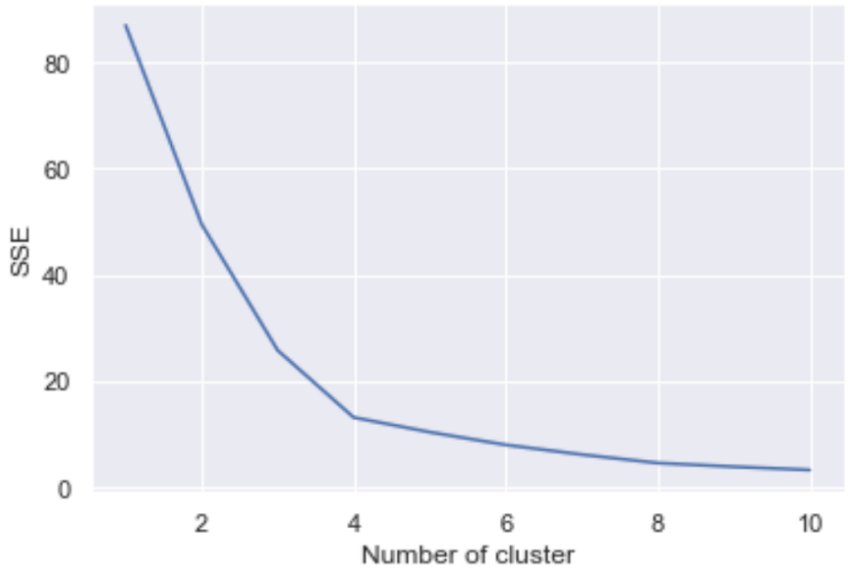


Figure 5.3.7: SSE plot

Figure 5.3.8. Figure 5.3.9 is the data set we obtained through classification. the data set we normalized.

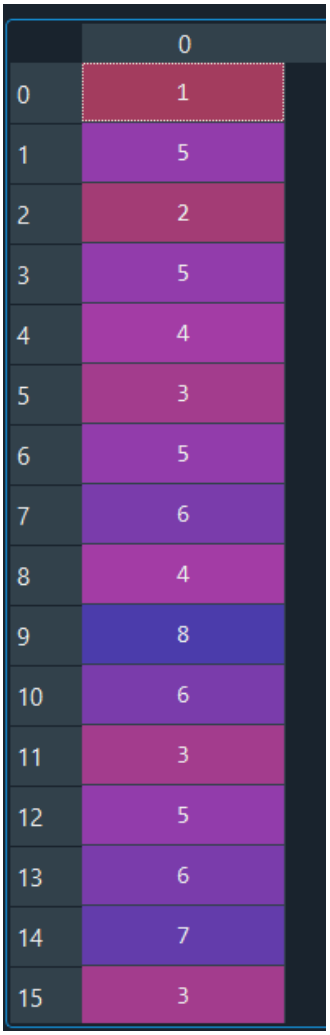
	tropical fruit	whole milk	...	toilet cleaner	preservation products
0	1	1	...	0	0
1	1	1	...	0	0
2	0	0	...	0	0
3	0	0	...	0	0
4	0	1	...	0	0
...	...	...	...	...	...
3893	0	0	...	0	0
3894	0	0	...	0	0
3895	0	0	...	0	0
3896	0	0	...	0	0
3897	0	0	...	0	0

Figure 5.3.8: Dataset obtained from classification

	0	1	2	...	165	166	167
0	0.171277	0.102860	-0.042899	...	-0.003390	-0.001515	-0.117275
1	0.130736	0.078513	-0.032745	...	-0.002587	-0.001157	0.020985
2	-0.050046	-0.083333	0.199809	...	-0.003248	-0.001452	-0.077681
3	-0.080979	-0.134841	-0.066502	...	-0.005255	-0.002349	0.042618
4	-0.059444	0.117050	-0.048817	...	-0.003858	-0.001724	-0.009900
...	...	...	...	...	...	...	...
3893	-0.097390	-0.162169	-0.079980	...	-0.006320	-0.002825	-0.151170
3894	-0.134778	-0.224425	-0.110684	...	-0.008746	-0.003909	-0.209204
3895	-0.088917	-0.148060	-0.073022	...	-0.005770	-0.002579	-0.138018
3896	-0.070968	-0.118173	-0.058282	...	-0.004605	-0.002059	-0.110158
3897	-0.072731	-0.121107	-0.059729	...	-0.004720	-0.002110	-0.112893

Figure 5.3.9: Normalized dataset

As a result of these processes, each of our members belongs to a cluster. According to the distances of the members from each other, 10 different clusters were formed and a table was created as in Figure 5.3.10.



	0
0	1
1	5
2	2
3	5
4	4
5	3
6	5
7	6
8	4
9	8
10	6
11	3
12	5
13	6
14	7
15	3

Figure 5.3.10: Conclusion dataset

REFERENCES

Akman, M., Genç, Y., Ankaralı, H., 2011. Random forests yöntemi ve sağlık alanında bir uygulama, Türkiye Klinikleri Journal of Biostatistics, 3 (1): 36-48.(2021, December, 31). ESOGÜ Akademik Açık Erişim Sistemi http://openaccess.ogu.edu.tr:8080/xmlui/handle/11684/249

Angeline, D. M. D. (2013). Association rule generation for student performance analysis using apriori algorithm. The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), 1(1), 12-16

Apriori Algorithm In Data Mining: Implementation With Examples. (2021, November, 19). Software Testing Help. https://www.softwaretestinghelp.com/apriori-algorithm/

Apriori: Association Rule Mining In-depth Explanation and Python Implementation. (2021, December, 31). towards data science https://towardsdatascience.com/apriori-association-rule-mining-explanation-and-python-implementation-290b42afdfc6

Breiman, L., 2001. Random Forests, Machine Learning, 45 (1): 5-32. (2021, December, 31). https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

Clustering. (2021, December, 31). scikit learn. https://scikit-learn.org/stable/modules/clustering.html

Data preprocessing. (2021, July, 11). techopedia https://www.techopedia.com/definition/14650/data-preprocessing

Exploring Clustering Algorithms: Explanation and Use Cases. (2021, October, 13). neptuneblog. https://neptune.ai/blog/clustering-algorithms

FP-Growth Algoritması-Weka Uygulaması (2021, November, 31)..EJONS International Journal on Mathematic, Engineering and Natural Sciences https://www.ejons.co.uk/Makaleler/619573122_%c4%b0lk%20sayfa%206-5.pdf

Frequent Pattern Growth Algorithm in Data Mining. (2021, November 29) https://www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/

K-means Kümeleme. (2021, December 31). Vikipedi. https://tr.wikipedia.org/wiki/K-means_k%C3%BCmeleme

Lift in an association rule (IBM Documentation). (2021, December, 31). IBM https://www.ibm.com/docs/en/db2/9.7?topic=associations-lift-in-association-rule

Market Basket Analysis in Python Example. (31, December, 2021). https://goldinlocks.github.io/Market-Basket-Analysis-in-Python/#Leverage-and-Conviction

matplotlib.pyplot. (2021, December, 31). matplotlib. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

Model Seçimi-K Fold Cross Validation. (2021, December, 31). https://medium.com/@gulcanogundur/model-se%C3%A7imi-k-fold-cross-validation-4635b61f143c

Niels Landwehr, Mark Hall, and Eibe Frank (2003). (2021, December, 31). http://www.cs.waikato.ac.nz/~ml/publications/2003/landwehr-etal.pdf

pandas(software). (2021, December, 31). Wikipedia. https://en.wikipedia.org/wiki/Pandas_(software)

R Vs Python: What's the Difference?. (2021, December 31). Guru99. https://www.guru99.com/r-vs-python.html

Sequential Pattern Mining. (2021, December, 4). https://www.cc.gatech.edu/~hic/CS7616/pdf/lecture13.pdf

sklearn.decomposition.PCA. (2021, December, 31). scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

sklearn.preprocessing.StandardScaler. (2021, December, 31). scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Suchetana, B., Rajagopalan, B., & Silverstein, J. (2017). Assessment of wastewater treatment facility compliance with decreasing ammonia discharge limits using a regression tree model. Science of the total environment, 598, 249-257.

Tomovic, S., & Stanisic, P. (2011). Cross Validation Method in Frequent Itemset Mining. In Central European Conference on Information and Intelligent Systems (p. 297). Faculty of Organization and Informatics Varazdin

What is Python? Executive Summary. (2021, December, 31). python. https://www.python.org/doc/essays/blurb/