A Grocery Analysis

by

Rudri Oza

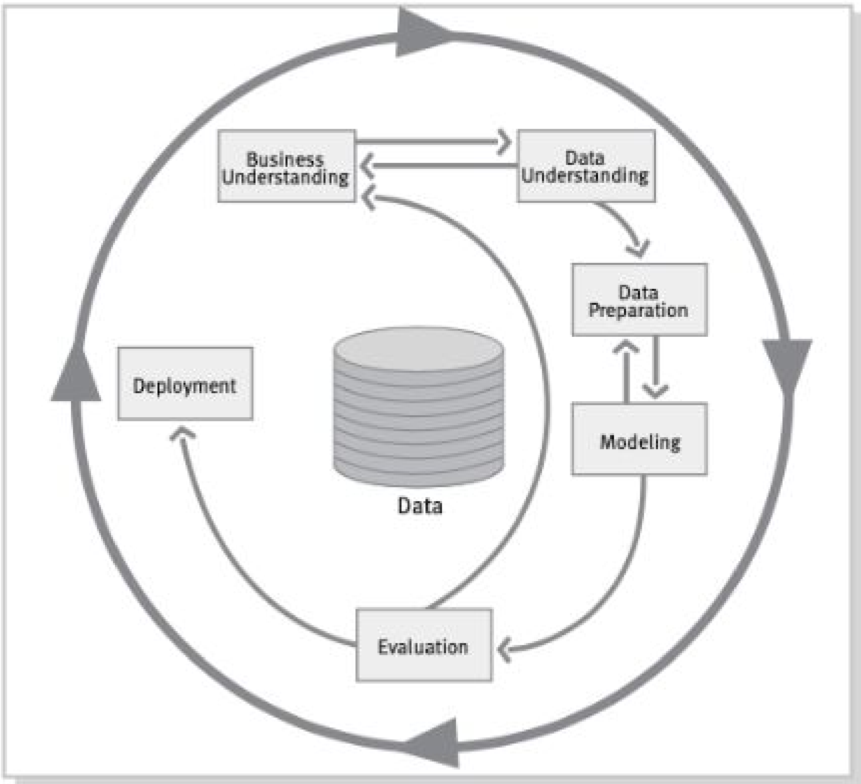
Comp 541: Data Mining
Fall 2019

Department of Computer Science
California State University, Northridge
18111 Nordhoff St., Northridge,CA- 91324

Table of Contents

1. Introduction

For this project we have followed the Crisp-DM reference model. The reference model is as follows:



1. Business Understanding:
   This initial phase focuses on understanding the objectives and requirements of the project from business objectives. Next, this knowledge is converted into a definition of a data mining problem. Moreover, create an initial plan to achieve the objectives.

2. Data Understanding:
   The data understanding phase begins with initial data aggregation and follows an exploration of the data. Exploring the data allows the discovery of data quality problems and insights that will advance the formation of hypotheses about the dataset.

3. Data Preparation:
   The data preparation phase involves preprocessing the data into a format that is the bare necessities for data modeling. Preparation may include, normalizing the data and dropping unnecessary attributes. This process may take several iterations.

4. Modeling:
   The modeling phase is where the chosen data mining or machine learning techniques are chosen and applied. Designing variable attributes of the model are done here.

5. Evaluation:
   The Evaluation phase analyzes the results from the modeling phase. The business object success is determined here.

6. Deployment:
   Deployment phase is accomplished by both data analysts and business experts. This phase determines how the model constructed will be used throughout the business.

2. Business Understanding

2.1 Determine Business Objectives:

A business objective is essentially a company's goal. A business object also includes the strategies that employees will use to get there, a time limit and collection of available resources.

The business objective we have chosen is:

Some foods are often purchased together, and stores can benefit by setting the foods together in an area or placing them in a single package. Pairing foods can play an important role when people take longer to buy items, and the possibility of people taking more items. If people are quick to find their items, they should be provoked to come back to the store next time.

In general we can say that it is a market basket analysis. Market basket analysis is a modeling technique based on the principle that if you buy a certain set of items, you are more (or less) likely to buy another set of items. For example, if you are in an English pub and you buy a pint of beer and do not buy a meal once, you are likely to buy crisps (US chips) at the same time as those who do not buy beer.

The set of items customers purchase is referred to as the itemset, and market basket analysis attempts to find a relationship between purchases.

2.2 Situation Assessment:

Market basket analysis is really important in today's technological times. Market Basket Analysis works by finding which objects are purchased together over a history of transactions. In other words, Market Basket Analysis is the finding of frequently purchased item collections. The importance of these frequent item sets, is so retailers may place these items together. Market Basket Analysis determines these correlations with various algorithms but for the purpose of this project will be looking at the Apriori Algorithm. This algorithm provides an automated means of conducting this analysis, and there is no risk of human error when properly tuned. The Apriori algorithm is a pattern mining algorithm that is used to determine the most frequently occurring patterns in a dataset. This algorithm uses a bottom-up approach to find patterns in a dataset. Patterns are generated according to the algorithm's minimum support and confidence parameters. These parameters describe the frequency of the pattern and the probability of the pattern occurring.

2.3 Data Mining Goals:

- To find correlations between items bought at a grocery store or marketplace in order to make inferences about which items are frequently bought together.
- Possibly find unknown correlations between items that are not directly related to each other.
- Examine how well a Random Forest is able to predict the purchase of a particular item.

2.4 Project plan:

Language: Python (Pandas)

Implementation Algorithms: Association Rule Mining

Steps:

1) Data Exploration
2) Data Cleaning
3) Data Visualization
4) Data Pre-processing
5) Implement Algorithms
6) Conclusion

3. Data Understanding

3.1 Initial Data Collection:

Market Basket Analysis is a very popular topic among data scientists and analysts and as a result there is a lot of data set available out there for the same. Determining the correct dataset to select was an important task for our project.

So finally, we decided to go with the Instacart market basket analysis data available on paper. This is actually a prediction contest on Kaggle. It says: Whether you shop through carefully planned grocery lists or allow your grazing to guide us, our unique food rituals define who we are. Instacart, a grocery ordering and delivery app, aims to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them. After selecting products through the Instacart app, individual shoppers review your order and do in-store shopping and delivery for you.

Instacart's data science team plays a big role in providing this delightful shopping experience. They currently use transactional data to develop models that predict which products a user will buy again, try for the first time, or add to their cart during a session.

Recently, Instacart has opened this data - 3 million Instacart orders, see their blog post on Open Sourd.

In this competition, Instacart is challenging the Kagel community to use this anonymous data on customer orders over time, which predicts that previously purchased products will be next in order to the user. They are not only looking for the best model, Instacart is also looking for machine learning engineers to develop their team.

3.2 Explore Data:

The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. Each entity (customer, product, order, aisle, etc.) has an associated unique id. Most of the files and variable names should be self-explanatory.

aisles.csv

```
aisle_id,aisle
1,prepared soups salads
2,specialty cheeses
3,energy granola bars
...
```

departments.csv

```
department_id,department
1,frozen
2,other
3,bakery
...
```

order_products__*.csv

These files specify which products were purchased in each order. order_products__prior.csv contains previous order contents for all customers. 'reordered' indicates that the customer has a previous order that contains the product. Note that some orders will have no reordered items. You may predict an explicit 'None' value for orders with no reordered items. See the evaluation page for full details.

```
order_id,product_id,add_to_cart_order,reordered
1,49302,1,1
1,11109,2,1
1,10246,3,0
...
```

orders.csv

This file tells to which set (prior, train, test) an order belongs. You are predicting reordered items only for the test set orders. 'order_dow' is the day of week.

```
order_id,user_id,eval_set,order_number,order_dow,order_hour_of_day,days_si
nce_prior_order
 2539329,1,prior,1,2,08,
 2398795,1,prior,2,3,07,15.0
 473747,1,prior,3,3,12,21.0
 ...
```

products.csv

```
 product_id,product_name,aisle_id,department_id
 1,Chocolate Sandwich Cookies,61,19
 2,All-Seasons Salt,104,13
 3,Robust Golden Unsweetened Oolong Tea,94,7
 ...
```

Though, as this dataset had many unnecessary data and has been widely used, we decided to go with the other dataset that has not been as popular. The other dataset is available from kaggle for grocery analysis. It is called groceries.csv which is as below:

groceries - groceries.csv (784.05 KB) 20 of 33 columns Views

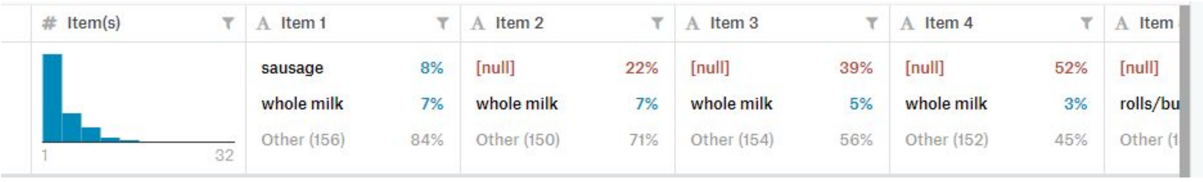
	Item(s)	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7
	1 – 32	sausage 8% whole milk 7% Other (156) 84%	[null] 22% whole milk 7% Other (150) 71%	[null] 39% whole milk 5% Other (154) 56%	[null] 52% whole milk 3% Other (152) 45%	[null] 62% rolls/buns 2% Other (149) 36%	[null] 71% soda 2% Other (136) 28%	[null] 77% soda 1% Other (137) 21%
1	4	citrus fruit	semi-finished bread	margarine	ready soups			
2	3	tropical fruit	yogurt	coffee				
3	1	whole milk						
4	4	pip fruit	yogurt	cream cheese	meat spreads			
5	4	other vegetables	whole milk	condensed milk	long life bakery product			
6	5	whole milk	butter	yogurt	rice	abrasive cleaner		
7	1	rolls/buns						
8	5	other vegetables	UHT-milk	rolls/buns	bottled beer	liquor (appetizer)		
9	1	potted plants						
10	2	whole milk	cereals					
11	5	tropical fruit	other vegetables	white bread	bottled water	chocolate		
12	9	citrus fruit	tropical fruit	whole milk	butter	curd	yogurt	flour
13	1	beef						
14	3	frankfurter	rolls/buns	soda				
15	2	chicken	tropical fruit					
16	4	butter	sugar	fruit/vegetable juice	newspapers			
17	1	fruit/vegetable juice						
18	1	packaged fruit/vegetables						
19	1	chocolate						
20	1	specialty bar						
21	1	other vegetables						

We have a transactional database dataset. A transactional database is a collection transactions, providing a historical record. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database.

The dataset was given in CSV format and contained 9835 transactions. Furthermore the dataset had 33 total columns, a column denoting the amount of items within the transactions and columns “Item 1 - Item 32” columns denoting the ith item bought. The dataset described a list of items bought per transactions.

3.3 Data Quality:

During the initial examination we discovered that the dataset we have is very relevant to our goal but it also has some flaws like every other dataset such as missing values (some of the columns in the tables are missing some values). Overall the quality of the data is good and some flaws can be removed during the data cleaning and preprocessing state.



# Item(s)	A Item 1		A Item 2		A Item 3		A Item 4		A Item
1 32	sausage	8%	[null]	22%	[null]	39%	[null]	52%	[null]
	whole milk	7%	whole milk	7%	whole milk	5%	whole milk	3%	rolls/bu
	Other (156)	84%	Other (150)	71%	Other (154)	56%	Other (152)	45%	Other (1

4. Data Preparation

4.1 Introduction:

Under the part of data preparation we also did some more data exploration. The data in the dataset may be inconsistent, missing. To get a better understanding of what we are dealing with. To plan our next steps according to the availability. To improve the efficiency of the end result. That is why data exploration is necessary.

4.2 The method:

We can get some statistical analysis of what items are purchased and by how many people. What is the average number of items a customer purchases at a time and represent them in the form of graphs.

Firstly, we are looking into what is the average number of items a person purchases at a time.In the dataset there are columns for up to 32 items a customer purchases. After loading the dataset we got a statistical analysis of how many times the column is used in the whole dataset. In this way we can find out what is the average number of items a customer purchases at a time.

```python
def get_stats(col):
    '''
    numeric
    '''
    print("===Stats===")
    print('mean:',col.mean())
    print('median:',col.median() )
    print('mode:',col.mode()[0] )
    print('range:',col.min(), "to", col.max() )
    print('std div:', col.std() )
    print('variance:', col.var())
    print('Coeff of variation:', col.std()/col.mean())
    print('skew:', col.skew())
```

The code and result

```
27        1
Name: Items, dtype: int64
===Stats===
mean: 4.409456024402644
median: 3.0
mode: 1
range: 1 to 32
std div: 3.5893845107948454
variance: 12.88368116633395
Coeff of variation: 0.8140197999323749
skew: 1.6357236148700154
quartiles 0          (3.0, 6.0]
1          (2.0, 3.0]
2        (0.999, 2.0]
3          (3.0, 6.0]
4          (3.0, 6.0]
              ...
9830      (6.0, 32.0]
9831     (0.999, 2.0]
9832      (6.0, 32.0]
9833       (3.0, 6.0]
9834       (3.0, 6.0]
Name: Items, Length: 9835, dtype: category
Categories (4, interval[float64]): [(0.999, 2.0] < (2.0, 3.0] < (3.0, 6.0] < (6.0, 32.0]]
```

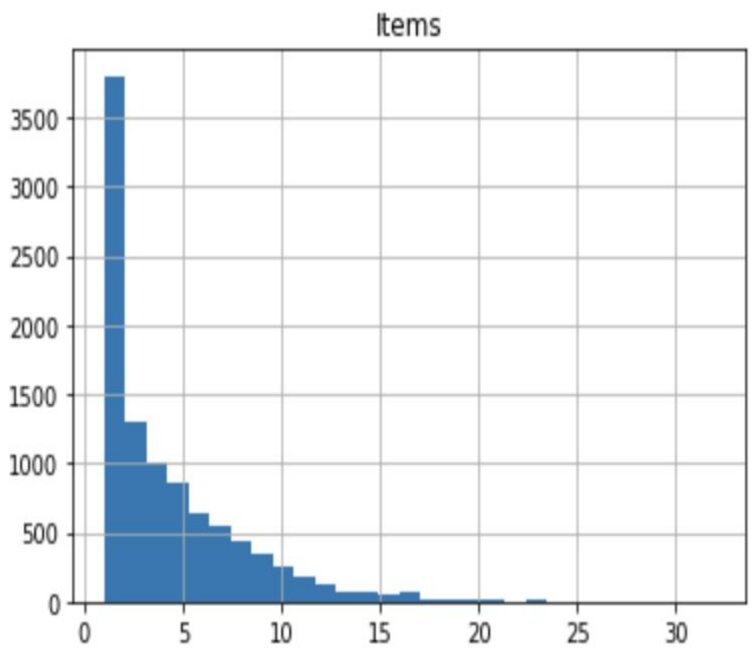
4.3 Data Visualization:

We are using bar graphs and pie charts to visualize our data. Along with these few items we have also used histogram to represent them. The graphs represent the number of times a specific item is purchased.

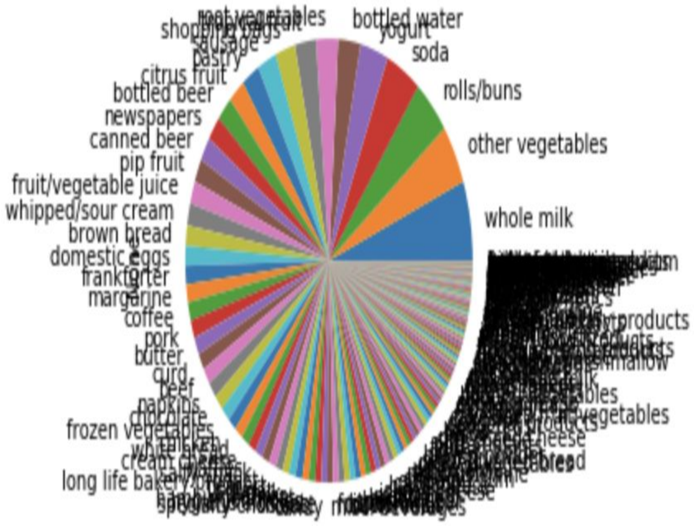
```python
def series_stats(ser):
    '''
    categorical
    '''
    print("===Stats===")
    v_counts = ser.value_counts()
    print('Unique Items:', len(v_counts))
    v_counts.plot.pie() #not a good chart
    plt.show()
    v_counts[::9].plot.bar()
    plt.show()


#lets start by getting our data data
df = pd.read_csv("groceries - groceries.csv")

#lets do an analysis of the number of items people bought
target_col = df['Items']
print(target_col.value_counts())
get_stats(target_col)

df.hist(column= "Items", bins= 29)
plt.show()

#now lets take a look at total counts for items

#here, we're going to flatten the dataset to a series
flat_list = []
for col in df.columns[1::]:
    flat_list = flat_list + df[col].tolist()
flat_list = [item for item in flat_list if type(item) is not float] #get rid of nans
df_series = pd.Series(flat_list)

#now that we have our series, lets do some analysis
print(df_series.value_counts())
series_stats(df_series)

#lets take a look at the one items
df_one = df.loc[df['Items']==1]
print(df_one['Item 1'].value_counts())
series_stats(df_one['Item 1'])
```

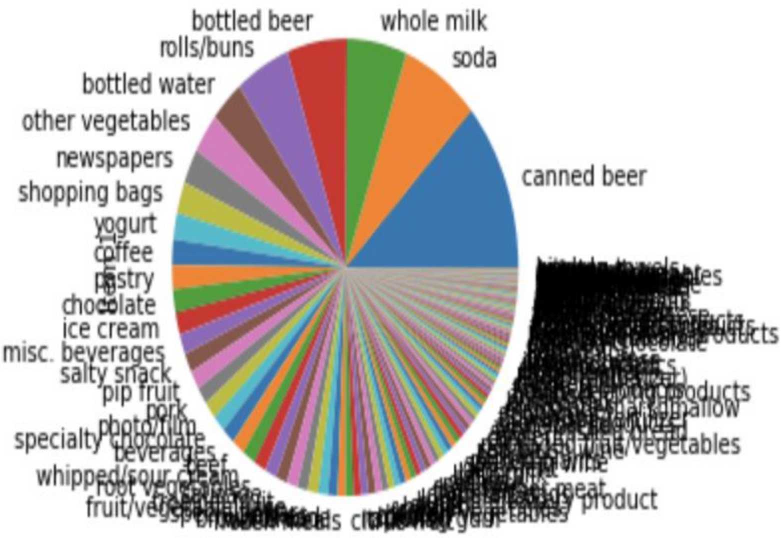
The Code and histogram

```
1     2159
2     1643
3     1299
4     1005
5      855
6      645
7      545
8      438
9      350
10     246
11     182
12     117
13      78
14      77
15      55
16      46
17      29
19      14
18      14
21      11
20       9
23       6
22       4
29       3
26       1
28       1
32       1
24       1
27       1
Name: Items, dtype: int64
```



Pie Chart: The first pie is for the items that are bought with something else and the second one is for the single items people bought in their grocery run.

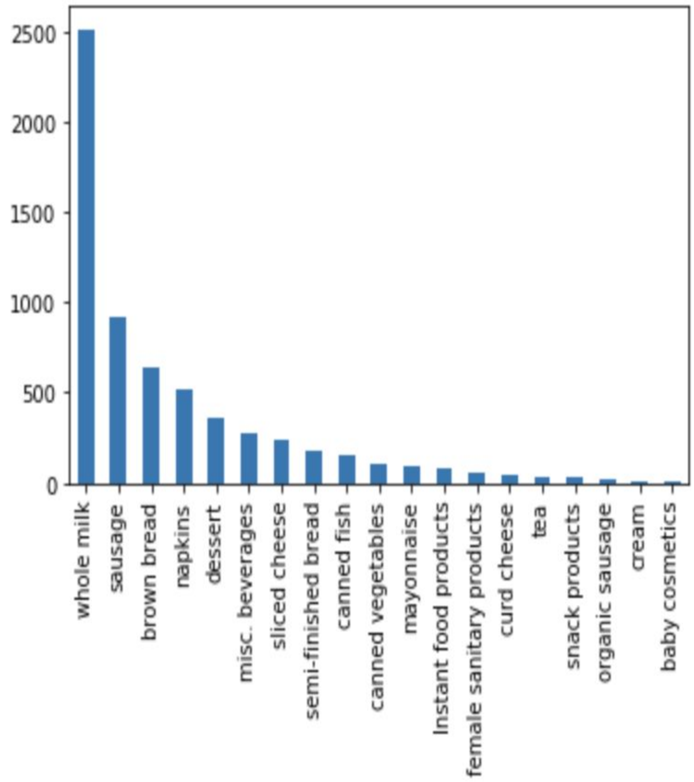


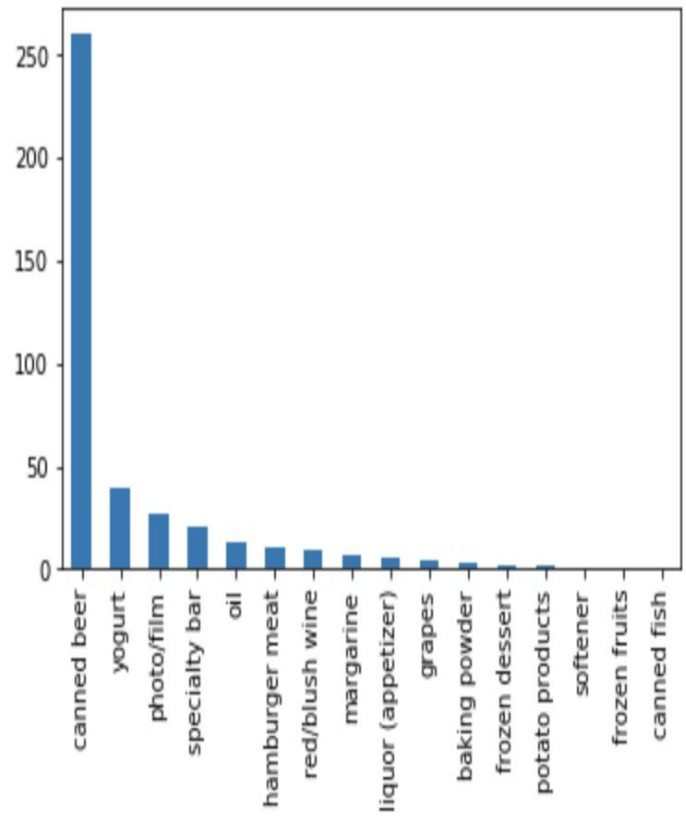


whole milk	2513
other vegetables	1903
rolls/buns	1809
soda	1715
yogurt	1372
	...
kitchen utensil	4
bags	4
preservation products	2
sound storage medium	1
baby food	1

canned beer	260
soda	156
whole milk	121
bottled beer	120
rolls/buns	109
	...
spices	1
nut snack	1
ketchup	1
cereals	1
kitchen towels	1

Bar Graphs

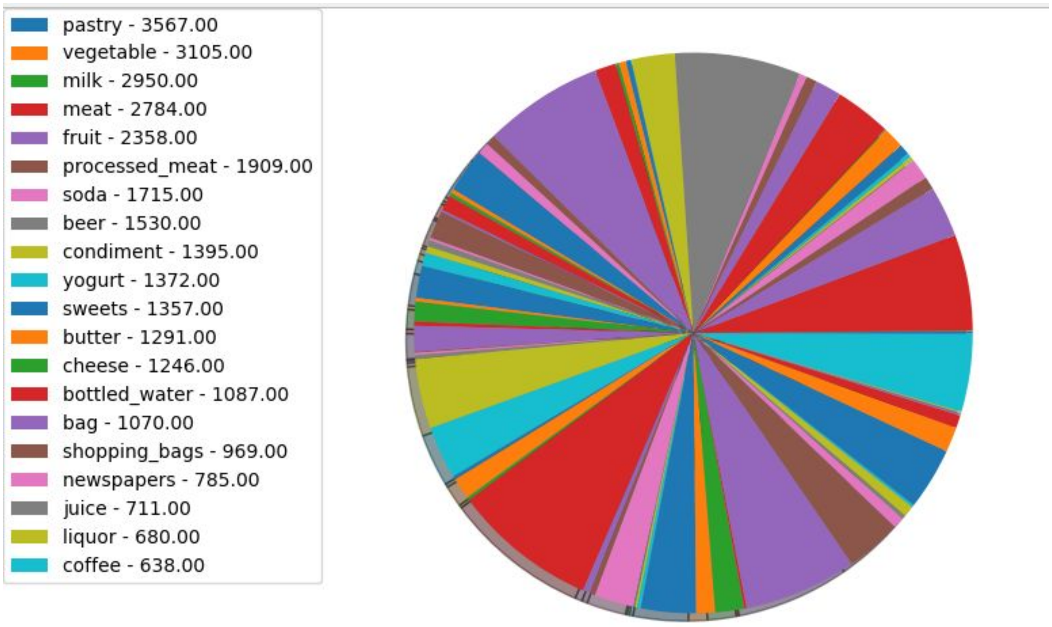

2500
2000
1500
1000
500
0
whole milk
sausage
brown bread
napkins
dessert
misc. beverages
sliced cheese
semi-finished bread
canned fish
canned vegetables
mayonnaise
Instant food products
female sanitary products
curd cheese
tea
snack products
organic sausage
cream
baby cosmetics


250
200
150
100
50
0
canned beer
yogurt
photo/film
specialty bar
oil
hamburger meat
red/blush wine
margarine
liquor (appetizer)
grapes
baking powder
frozen dessert
potato products
softener
frozen fruits
canned fish

4.4 Data Cleaning:

Looking through the dataset, we did not encounter anything that stood out in terms of cleanliness.

Data was mostly between the second and third quartiles, and null values don't correspond to lost data. The names of some items are very specific, so finding duplicates will take more effort. For example, there are "hard cheese" and there is also "soft cheese" in the Groceries dataset. These both correspond to a "Cheese" item and doesn't give not much more information. Categorization counts can be seen in this figure.



There are no outliers in the data set that appear as a result of gathering descriptive statistics. Duplicates reinforce product pairing. We do not have any redundant attributes.

5. Feature Extraction

5.1 Introduction:

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and/or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

5.2 The method:

In our project for feature extraction initially what we did was to convert our dataset into a proper list with column entry within the dataset. The code and the result for that are as below:

```
In [ ]:

In [1]: import pandas as pd
        import numpy as np
        import json

        from mlxtend.frequent_patterns import apriori
        from mlxtend.frequent_patterns import association_rules
        import mlxtend as ml

In [2]: csvFile = "groceries - groceries.csv"

In [3]: #lets start by getting our data data
        df = pd.read_csv(csvFile)

In [4]: df.columns

Out[4]: Index(['Items', 'Item 1', 'Item 2', 'Item 3', 'Item 4', 'Item 5', 'Item
        6',
               'Item 7', 'Item 8', 'Item 9', 'Item 10', 'Item 11', 'Item 12',
               'Item 13', 'Item 14', 'Item 15', 'Item 16', 'Item 17', 'Item 1
        8',
               'Item 19', 'Item 20', 'Item 21', 'Item 22', 'Item 23', 'Item 2
        4',
               'Item 25', 'Item 26', 'Item 27', 'Item 28', 'Item 29', 'Item 3
        0',
               'Item 31', 'Item 32'],
              dtype='object')

In [5]: df['concat_list_of_items'] = ''
        df['id'] = 0

In [6]: def convertToString(row):
            item_list=[]
            max_index = row['Items']
            i = 1
            for i in range(i,max_index+1):
                itemKey = 'Item '
                itemKey += str(i)
                item = row[itemKey].replace(" ", "_")
                item_list.append(item)
        #     item_list = ','.join(item_list)
            return item_list

In [7]: for i, row in df.iterrows():
            item_list = convertToString(row)
            df.at[i,'concat_list_of_items'] = item_list
            df.at[i,'id'] = i+1
        #     print(row['keywords'])
        #     print('\n')
```

```
In [8]: df.columns

Out[8]: Index(['Items', 'Item 1', 'Item 2', 'Item 3', 'Item 4', 'Item 5', 'Item
        6',
               'Item 7', 'Item 8', 'Item 9', 'Item 10', 'Item 11', 'Item 12',
               'Item 13', 'Item 14', 'Item 15', 'Item 16', 'Item 17', 'Item 1
        8',
               'Item 19', 'Item 20', 'Item 21', 'Item 22', 'Item 23', 'Item 2
        4',
               'Item 25', 'Item 26', 'Item 27', 'Item 28', 'Item 29', 'Item 3
        0',
               'Item 31', 'Item 32', 'concat_list_of_items', 'id'],
              dtype='object')

In [9]: df

Out[9]:
```

	Items	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8
0	4	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN
1	3	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN
2	1	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	4	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN
4	4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
9830	17	sausage	chicken	beef	hamburger meat	citrus fruit	grapes	root vegetables	whole milk
9831	1	cooking chocolate	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9832	10	chicken	citrus fruit	other vegetables	butter	yogurt	frozen dessert	domestic eggs	rolls/buns
9833	4	semi-finished bread	bottled water	soda	bottled beer	NaN	NaN	NaN	NaN
9834	5	chicken	tropical fruit	other vegetables	vinegar	shopping bags	NaN	NaN	NaN

9835 rows × 35 columns

```
In [10]: list_of_items = df[['concat_list_of_items']]
```

The Apriori algorithm provided by mlxtend python library only accepts a dataframe that has each feature as a column and a boolean denoting that entry is apart of the transaction. The list allowed us to easily create this type of dataframe. We iterated through the list to form a set of unique item entries to form a collection of columns for our new dataframe and initialized each item entry as 0. Once that was created the new dataframe, we iterated through the previous list per transaction and if the item existed we would change the 0 to 1 denoting the entry was a part of the transaction. After the data preprocessing was complete we found the Apriori results.

```
In [11]: cols_set = set()
         for i, row in list_of_items.iterrows():
             for j, item in enumerate(row['concat_list_of_items']):
                 cols_set.add(item)
```

```
In [12]: pandasDf2 = pd.DataFrame(columns = cols_set, index=range(9835))
         for col in pandasDf2.columns:
             pandasDf2[[col]]=0

         for i, row in df.iterrows():
             for c in row['concat_list_of_items']:
                 pandasDf2.at[i, c] = 1
```

```
In [13]: pandasDf2.head()
```

Out[13]:

	bottled_water	ready_soups	sliced_cheese	frozen_chicken	rolls/buns	specialty_fat	popcorn
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

5 rows × 169 columns

```
In [14]: frequent_itemsets = apriori(pandasDf2, min_support=0.01, use_colnames=True)
```

```
In [15]: rules = association_rules(frequent_itemsets, metric="lift")
```

```
In [16]: rules.sort_values('lift', ascending = False, inplace = True)
```

```
In [17]: rules.head()
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
518	(curd)	(whole_milk, yogurt)	0.053279	0.056024	0.010066	0.188931	3.372
515	(whole_milk, yogurt)	(curd)	0.056024	0.053279	0.010066	0.179673	3.372
574	(other_vegetables, citrus_fruit)	(root_vegetables)	0.028876	0.108998	0.010371	0.359155	3.295
579	(root_vegetables)	(other_vegetables, citrus_fruit)	0.108998	0.028876	0.010371	0.095149	3.295
473	(other_vegetables, yogurt)	(whipped/sour_cream)	0.043416	0.071683	0.010168	0.234192	3.267

```
In [18]: con = rules['confidence'] >= .7
```

```
In [19]: rules[con]
```

Out[19]:

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	convict

We configured the apriori algorithm to have a minimum support of 0.01 and a minimum confidence of 0.70, as a result we were not able to find any frequent items sets.

In order to simplify our data and have our results above a much higher minimum confidence, we categorized our data into the classes identified in Appendix A, and we were able to gain a different result. The data would be slightly more vague (i.e. instead of hard_cheese and processed_cheese, these items are listed as cheese), the data would still give us results that are overall acceptable for our objective.

6. Data Pre-Processing and Results

The original unique item count was 169, due to categorization the item count dropped to 77 unique attributes. This caused some items to be duplicated in the same transaction and therefore this needed to be cleaned as well during preprocessing, so we dropped the duplicate items. After removing duplicates we put the resulting set of items in the Apriori algorithm from the python library mlxtend.

In [91]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from scipy.stats import binned_statistic

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA, TruncatedSVD
import matplotlib.patches as mpatches
import time

# Classifier Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustSc
aler
from sklearn import svm

import collections
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB


# Other Libraries
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score, f1_score, roc
_auc_score, accuracy_score, classification_report
from collections import Counter
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.metrics import roc_curve
from sklearn.model_selection import cross_val_predict


import warnings


from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import mlxtend as ml

from sklearn.utils.multiclass import unique_labels


import copy
warnings.filterwarnings("ignore")
```

In [92]:
```
csvFile = "./OLD_output.csv"
```

In [93]:
```
pandasDf = pd.read_csv(csvFile)
```

In [94]:
```
pandasDf.columns
```

Out[94]:
```
Index(['Unnamed: 0', 'cleaning_product;', 'artif._sweetener', 'skin_car
e;',
       'baby_food;', 'bag;', 'baking_powder;', 'cleaning_product;.1',
'meat;',
       'berries;',
       ...
       'milk;', 'vinegar;', 'waffles;', 'condiment;.10', 'liquor;.8',
       'pastry;.7', 'liquor;.9', 'milk;.1', 'yogurt;', 'snack;.7'],
      dtype='object', length=170)
```

In [95]:
```
pandasDf.drop(inplace=True, axis=1, columns=['Unnamed: 0'])
```

In [96]:
```
cols = pandasDf.columns
```

In [97]:
```
cols_set = set()
for i, val in enumerate(cols):
    cols_set.add(val.split(';')[0])
```

In [98]:
```
len(cols_set)
```

Out[98]:
```
77
```

In [ ]:

In [99]:
```
pandasDf['items'] = ''
```

In [ ]:

In [100]:
```
for i, row in pandasDf.iterrows():
    item_list = []
    for col in pandasDf.columns:
        if(row[col] == True):
            for c in cols_set:
                if (col.find(c) != -1):
                    item_list.append(c)
    pandasDf.at[i, 'items'] = item_list
```

In [101]:
```
pandasDf['items'].unique
```

Out[101]:
```
<bound method Series.unique of 0                                  [fruit, butt
er, soup, pastry]
1                                   [coffee, fruit, yogurt]
2                                                    [milk]
3             [cheese, processed_meat, meat, fruit, yogurt]
4                          [sweets, pastry, vegetable, milk]
                              ...
9830    [meat, butter, poultry, sweets, fruit, coffee,...
9831                                               [pastry]
9832    [butter, poultry, fruit, bag, domestic_eggs, d...
9833                    [beer, bottled_water, pastry, soda]
9834    [poultry, vegetable, shopping_bags, bag, fruit...
Name: items, Length: 9835, dtype: object>
```

In [102]:
```
for col in pandasDf.columns:
    if col != 'items':
        pandasDf[[col]]=pandasDf[[col]].astype('int')
```

In [103]:
```
for i, row in pandasDf.iterrows():
    a = []
    if not row['items']:
        print('list is empty')
```

In [ ]:

In [ ]:

In [104]:
```
pandasDf2 = pd.DataFrame(columns = cols_set, index=range(9835))
for col in pandasDf2.columns:
    pandasDf2[[col]]=0

for i, row in pandasDf.iterrows():
    item_list = []
    for c in row['items']:
        pandasDf2.at[i, c] = 1
```

In [105]:
```
pandasDf2.head()
```

Out[105]:

	milk	packaged_fruit/vegetables	newspapers	cocoa_drinks	vinegar	fruit	pet_food	dental_ca
0	0	0	0	0	0	1	0	
1	0	0	0	0	0	1	0	
2	1	0	0	0	0	0	0	
3	0	0	0	0	0	1	0	
4	1	0	0	0	0	0	0	

5 rows × 77 columns

```
In [ ]:
```

```
In [ ]:
```

```
In [113]: frequent_itemsets = apriori(pandasDf2, min_support=0.01, use_colnames=True)

In [114]: rules = association_rules(frequent_itemsets, metric="lift")

In [115]: con = rules['confidence'] >= .7

In [116]: rules = rules[con]

In [117]: rules.sort_values('lift', ascending = False, inplace = True)

In [118]: len(rules[con])

Out[118]: 585

In [119]: rules.head()

Out[119]:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
9221	(meat, pastry, shopping_bags)	(bag, processed_meat)	0.022064	0.031317	0.018607	0.843318	26.928676
9215	(pastry, bag, meat)	(shopping_bags, processed_meat)	0.023894	0.029181	0.018607	0.778723	26.685522
7988	(milk, meat, shopping_bags)	(bag, processed_meat)	0.013421	0.031317	0.010880	0.810606	25.884125
7985	(milk, meat, bag)	(shopping_bags, processed_meat)	0.014642	0.029181	0.010880	0.743056	25.463245
9213	(pastry, bag, processed_meat)	(meat, shopping_bags)	0.019624	0.037926	0.018607	0.948187	25.001111

```
In [ ]:
```

We were able to find more results with the same minimum support of 0.01 and minimum confidence of .70. Furthermore, we were able to find frequent items sets based on the consequent and the antecedent. This concludes our process successfully, and we were able to generate strong rules that are acceptable for our objective. These rules are listed in Appendix B.

7. Machine Learning Model

In order to meet project requirements we were supposed to use a Machine Learning model, on the dataset. Based off the consequent results we formed a set of unique items.

```
In [121]: unqiue_items = rules['consequents'].unique()
```

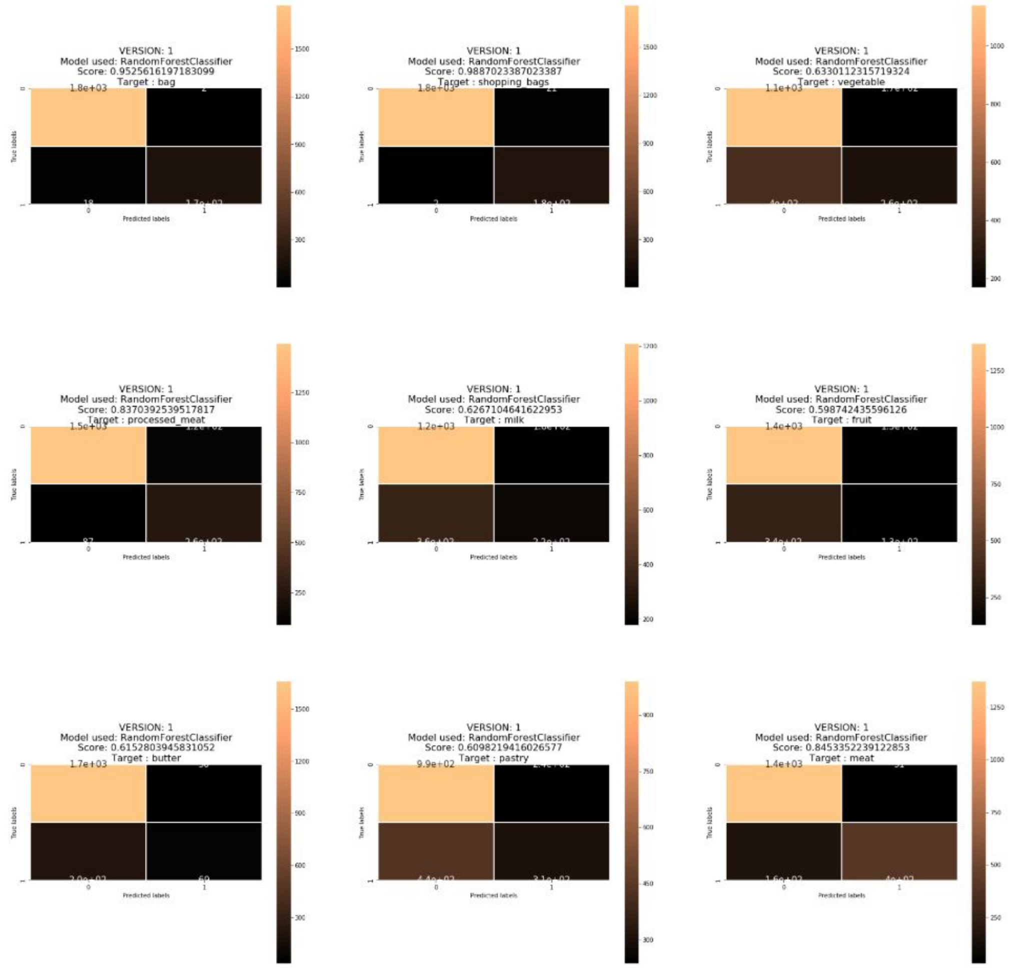
```
In [122]: target_set = set()
          for i, val in enumerate(unqiue_items):
              for j, val2 in enumerate(val):
                  target_set.add(val2)
```

```
In [123]: target_set
```

```
Out[123]: {'bag',
           'butter',
           'fruit',
           'meat',
           'milk',
           'pastry',
           'processed_meat',
           'shopping_bags',
           'vegetable'}
```

We wanted to see how well a random forest will perform accessing these target feature attributes. The Random Forest Algorithm is an ensemble algorithm. This algorithm is comprised of decision trees. Decision tree is an algorithm that makes rules based on the data to perform analysis. These trees prone to overfitting. To correct for this overfitting the average classification result of the decision trees within the forest is taken to reduce the overfitting. The output classification is the result of taking the average of the prespecified number of trees. We took a set from the consequent results and would iterate through using each item as of the set as the target feature and using the rest of the 76 unique items as our selected features. We were able to generate confusion matrices and ROC curves.

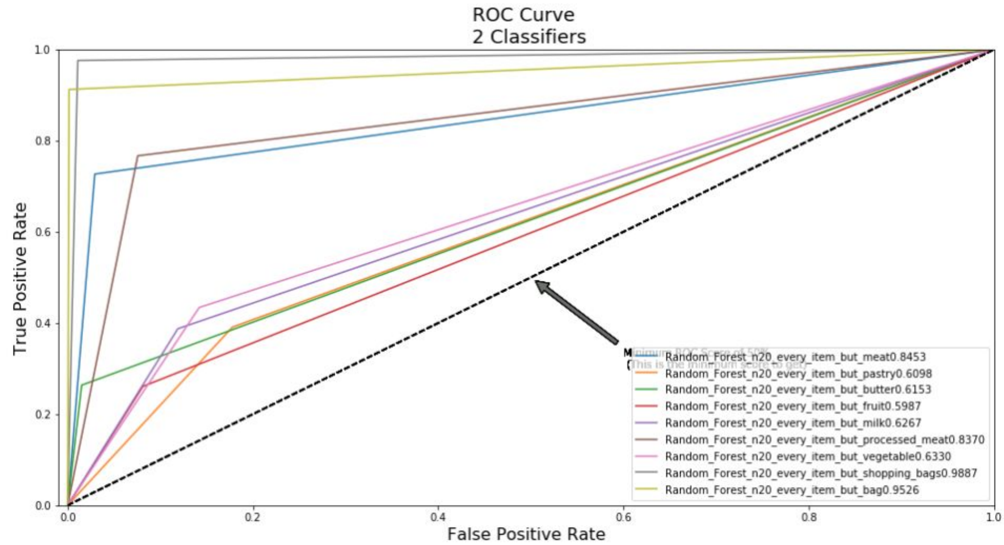
```
In [54]: show_all_confusion_matrix(results)
```



	meat	pastry	butter	fruit	milk	processed_meat	vegetable	shopping_bags	bag
TN	402	306	69	133	221	262	261	183	174
FN	31	237	36	127	177	124	171	21	2
TP	1372	989	1657	1369	1210	1494	1138	1761	1773
FP	162	435	205	338	359	87	397	2	18
TPR	0.98	0.81	0.98	0.92	0.87	0.92	0.87	0.99	1
TNR	0.71	0.41	0.25	0.28	0.38	0.75	0.4	0.99	0.91
PPV	0.89	0.69	0.89	0.8	0.77	0.94	0.74	1	0.99
NPV	0.93	0.56	0.66	0.51	0.56	0.68	0.6	0.9	0.99
FPR	0.29	0.59	0.75	0.72	0.62	0.25	0.6	0.01	0.09
FNR	0.02	0.19	0.02	0.08	0.13	0.08	0.13	0.01	0
FDR	0.11	0.31	0.11	0.2	0.23	0.06	0.26	0	0.01
Recall Score	0.71	0.41	0.25	0.28	0.38	0.75	0.4	0.99	0.91
Precision Score	0.93	0.56	0.66	0.51	0.56	0.68	0.6	0.9	0.99
F1 Score	0.81	0.48	0.36	0.36	0.45	0.71	0.48	0.94	0.95
Accuracy Score	0.9	0.66	0.88	0.76	0.73	0.89	0.71	0.99	0.99

Based on the confusion matrix, and score results not all target features performed equally as well. Some target features outperform others, this can be clearly seen with the ROC curve. The ROC curves illustrates the features that a Random Forest is able to predict are Shopping Bags and Bags. The performance for the rest of feature set items had too low of a performance for Random Forest to accurately predict if an item will be bought.

```
In [52]: create_roc_cruve(results)
```



8. Summary

In the beginning, the data we used was very straight forward and didn't require much preprocessing as we only had to clean some nullity data and later on, some duplicates when we categorized and simplified the data. After pre-processing the dataset, we planned on using both the Apriori Algorithm and the FP Growth algorithms for producing some strong rules from the dataset. These became rules for our Random Forest algorithm that was used for predicting the occurrence of items in transactions. Over all we learned the importance of data preprocessing, the power of searching for already available tools, and the whole data mining process. Reading about the importance of Market basket analysis was very informative and the skills learned from this was very applicable to real world applications. Getting exposed to Python and its libraries will be beneficial to our careers.

Appendix A

Word	Category	Word	Category	Word	Category	Word	Category	Word	Category	Word	Category
bottled_beer	beer	fruit/vegetable_juice	juice	cat_food	pet_food	canned_vegetables	vegetable	rolls/buns	pastry	liqueur	liquor
pastry	pastry	salt	salt	soda	soda	curd_cheese	cheese	sound_storage_medium	sound_storage_medium	dog_food	pet_food
canned_beer	beer	onions	vegetable	female_sanitary_products	sanitation_item	cream_cheese	cheese	syrup	sweets	dish_cleaner	cleaning_product
cling_film/bags	bag	uht-milk	milk	light_bulbs	light_bulbs	tropical_fruit	fruit	frozen_meals	frozen_food	toilet_cleaner	cleaning_product
chicken	poultry	vinegar	vinegar	white_wine	liquor	potted_plants	plants	whipped/sour_cream	condiment	oil	oil
sweet_spreads	jelly	rubbing_alcohol	rubbing_alcohol	salty_snack	snack	spices	condiment	roll_products	roll_products	rice	rice
pasta	pasta	canned_fruit	fruit	baby_food	baby_food	dental_care	dental_care	specialty_bar	specialty_bar	pudding_powder	sweets
curd	curd	cooking_chocolate	pastry	sausage	processed_meat	tidbits	snack	meat	meat	instant_coffee	coffee
cake_bar	pastry	whisky	liquor	yogurt	yogurt	skin_care	skin_care	kitchen_towels	paper_towels	make_up_remover	skin_care
citrus_fruit	fruit	herbs	condiment	mayonnaise	mayonnaise	specialty_fat	condiment	mustard	condiment	fish	fish
abrasive_cleaner	cleaning_product	butter	butter	root_vegetables	vegetable	specialty_chocolate	sweets	hair_spray	hair_spray	organic_products	organic_products
nut_snack	snack	dessert	dessert	liquor	liquor	frozen_chicken	poultry	honey	honey	candy	sweets
soap	cleaning_product	hamburger_meat	processed_meat	nuts/prunes	snack	frozen_fruits	fruit	coffee	coffee	misc._beverages	misc_beverages
brandy	liquor	meat_spreads	processed_meat	long_life_bakery_product	pastry	spread_cheese	cheese	preservation_products	preservation_products	ham	processed_meat
semi-finished_bread	pastry	frozen_potato_products	vegetable	red/blush_wine	liquor	potato_products	snack	condensed_milk	sweets	instant_food_products	instant_food_products
ready_soups	soup	processed_cheese	cheese	chocolate	sweets	flower_(seeds)	flowers	pet_care	pet_care	beverages	misc_beverages
white_bread	pastry	flour	flour	decalcifier	cleaning_product	cocoa_drinks	cocoa_drinks	chocolate_marshmallow	sweets	baking_powder	baking_powder
packaged_fruit/vegetables	packaged_fruit/vegetables	bags	bag	bathroom_cleaner	cleaning_product	bottled_water	bottled_water	margarine	butter	other_vegetables	vegetable
turkey	poultry	frozen_fish	fish	flower_soil/fertilizer	gardening_supplies	baby_cosmetics	skin_care	frankfurter	processed_meat	hygiene_articles	hygiene_articles
house_keeping_products	cleaning_product	artif_sweetener	condiment	rum	liquor	ice_cream	ice_cream	sparkling_wine	liquor	newspapers	newspapers
specialty_vegetables	vegetable	chewing_gum	sweets	candles	candles	shopping_bags	shopping_bags	jam	condiment	soft_cheese	cheese
dishes	dishes	cereals	cereals	detergent	cleaning_product	domestic_eggs	domestic_eggs	kitchen_utensil	cookware	liquor_(appetizer)	liquor
zwieback	snack	sauces	condiment	napkins	kitchen_utensil	butter_milk	butter_milk	softener	softener	cookware	cookware
canned_fish	fish	liver_loaf	processed_meat	finished_products	finished_products	berries	berries	waffles	waffles	popcorn	snack
organic_sausage	processed_meat	seasonal_products	seasonal_products	snack_products	snack	brown_bread	pastry	prosecco	liquor	beef	meat
pickled_vegetables	vegetable	ketchup	condiment	pork	meat	sugar	condiment	male_cosmetics	skin_care	cream	condiment
salad_dressing	condiment	whole_milk	milk	pip_fruit	fruit	frozen_vegetables	vegetable	grapes	fruit	sliced_cheese	cheese
frozen_dessert	dessert	specialty_cheese	cheese	cleaner	cleaning_product	hard_cheese	cheese	photo/film	photo_or_film	tea	tea
soups	soup										

Appendix B

['bag', 'beer']	['butter', 'butter_milk', 'fruit']	['cheese', 'condiment', 'meat']	['cheese', 'yogurt', 'vegetable', 'processed_meat']	['fruit', 'shopping_bags', 'meat']	['milk', 'butter', 'condiment', 'fruit']
['bag', 'butter']	['butter', 'butter_milk', 'meat']	['cheese', 'condiment', 'processed_meat']	['cleaning_product', 'meat']	['fruit', 'shopping_bags', 'vegetable']	['milk', 'butter', 'condiment', 'meat']
['bag', 'condiment']	['butter', 'butter_milk', 'pastry']	['cheese', 'condiment']	['cleaning_product', 'processed_meat']	['fruit', 'shopping_bags']	['milk', 'butter', 'condiment']
['bag', 'fruit', 'meat']	['butter', 'butter_milk', 'vegetable']	['cheese', 'fruit', 'condiment']	['cleaning_product', 'vegetable', 'meat']	['fruit', 'vegetable', 'meat', 'condiment']	['milk', 'butter', 'meat', 'fruit']
['bag', 'fruit', 'pastry']	['butter', 'butter_milk']	['cheese', 'fruit', 'meat', 'processed_meat']	['coffee', 'bag']	['fruit', 'vegetable', 'meat']	['milk', 'butter', 'meat']
['bag', 'fruit', 'vegetable']	['butter', 'condiment', 'fruit']	['cheese', 'fruit', 'meat']	['coffee', 'meat']	['fruit', 'yogurt', 'condiment']	['milk', 'butter', 'pastry', 'condiment']
['bag', 'fruit']	['butter', 'condiment']	['cheese', 'fruit', 'pastry', 'meat']	['coffee', 'processed_meat']	['fruit', 'yogurt', 'meat']	['milk', 'butter', 'pastry', 'meat']
['bag', 'juice']	['butter', 'curd']	['cheese', 'fruit', 'pastry', 'processed_meat']	['coffee', 'shopping_bags']	['fruit', 'yogurt', 'pastry', 'meat']	['milk', 'butter', 'pastry', 'processed_meat']
['bag', 'liquor']	['butter', 'meat', 'condiment']	['cheese', 'fruit', 'processed_meat']	['condiment', 'pastry', 'processed_meat', 'meat']	['fruit', 'yogurt', 'processed_meat', 'meat']	['milk', 'butter', 'processed_meat', 'fruit']
['bag', 'meat', 'soda']	['butter', 'meat', 'fruit']	['cheese', 'fruit', 'vegetable', 'meat']	['condiment', 'pastry', 'processed_meat']	['fruit', 'yogurt', 'processed_meat']	['milk', 'butter', 'processed_meat']
['bag', 'meat']	['butter', 'pastry', 'condiment']	['cheese', 'fruit', 'vegetable', 'processed_meat']	['condiment', 'poultry']	['fruit', 'yogurt', 'vegetable', 'meat']	['milk', 'butter', 'vegetable', 'processed_meat']
['bag', 'pastry', 'meat']	['butter', 'pastry', 'meat', 'fruit']	['cheese', 'meat', 'soda']	['condiment', 'processed_meat']	['juice', 'condiment']	['milk', 'cheese', 'vegetable', 'meat']
['bag', 'pastry', 'processed_meat', 'meat']	['butter', 'pastry', 'meat']	['cheese', 'meat']	['condiment', 'yogurt', 'processed_meat', 'meat']	['juice', 'meat']	['milk', 'cheese', 'vegetable', 'processed_meat']
['bag', 'pastry', 'processed_meat']	['butter', 'pastry', 'processed_meat', 'fruit']	['cheese', 'milk', 'condiment']	['condiment', 'yogurt', 'processed_meat']	['juice', 'pastry', 'meat']	['milk', 'condiment', 'meat', 'processed_meat']
['bag', 'pastry', 'soda']	['butter', 'pastry', 'processed_meat', 'meat']	['cheese', 'milk', 'fruit', 'meat']	['dessert', 'meat']	['juice', 'pastry', 'processed_meat']	['milk', 'condiment', 'meat']
['bag', 'pastry', 'vegetable', 'meat']	['butter', 'pastry', 'processed_meat']	['cheese', 'milk', 'fruit', 'processed_meat']	['dessert', 'processed_meat']	['juice', 'processed_meat', 'vegetable']	['milk', 'condiment', 'processed_meat']
['bag', 'pastry', 'vegetable']	['butter', 'pastry', 'vegetable', 'condiment']	['cheese', 'milk', 'meat', 'processed_meat']	['domestic_eggs', 'butter']	['juice', 'processed_meat']	['milk', 'fruit', 'butter_milk']
['bag', 'pastry']	['butter', 'pastry', 'vegetable', 'fruit']	['cheese', 'milk', 'meat']	['domestic_eggs', 'condiment']	['juice', 'shopping_bags']	['milk', 'fruit', 'condiment', 'processed_meat']
['bag', 'processed_meat', 'meat']	['butter', 'pastry', 'vegetable', 'meat']	['cheese', 'milk', 'pastry', 'meat']	['domestic_eggs', 'meat']	['juice', 'vegetable', 'meat']	['milk', 'fruit', 'condiment']
['bag', 'processed_meat']	['butter', 'pastry', 'vegetable']	['cheese', 'milk', 'pastry', 'processed_meat']	['domestic_eggs', 'milk', 'meat']	['kitchen_utensil', 'processed_meat']	['milk', 'fruit', 'meat', 'condiment']
['bag', 'shopping_bags', 'meat']	['butter', 'poultry']	['cheese', 'milk', 'processed_meat']	['domestic_eggs', 'milk', 'processed_meat']	['meat', 'butter', 'processed_meat', 'condiment']	['milk', 'fruit', 'meat', 'processed_meat']
['bag', 'shopping_bags', 'pastry', 'meat']	['butter', 'processed_meat', 'condiment']	['cheese', 'pastry', 'condiment', 'meat']	['domestic_eggs', 'pastry', 'meat']	['meat', 'curd']	['milk', 'fruit', 'meat']
['bag', 'shopping_bags', 'processed_meat']	['butter', 'processed_meat', 'fruit']	['cheese', 'pastry', 'condiment', 'processed_meat']	['domestic_eggs', 'pastry', 'processed_meat']	['meat', 'fruit', 'processed_meat', 'soda']	['milk', 'fruit', 'pastry', 'condiment']
['bag', 'shopping_bags', 'vegetable', 'meat']	['butter', 'processed_meat', 'meat', 'fruit']	['cheese', 'pastry', 'condiment']	['domestic_eggs', 'processed_meat']	['meat', 'milk', 'fruit', 'butter', 'processed_meat']	['milk', 'fruit', 'pastry', 'meat']
['bag', 'soda']	['butter', 'processed_meat']	['cheese', 'pastry', 'meat']	['domestic_eggs', 'vegetable', 'meat']	['meat', 'milk', 'fruit', 'condiment', 'processed_m	['milk', 'fruit', 'pastry', 'processed_meat']
['bag', 'sweets']	['butter', 'shopping_bags']	['cheese', 'pastry', 'processed_meat']	['domestic_eggs', 'vegetable', 'processed_meat']	['meat', 'milk', 'fruit', 'condiment', 'vegetable']	['milk', 'fruit', 'processed_meat']
['bag', 'vegetable', 'condiment']	['butter', 'vegetable', 'condiment']	['cheese', 'pastry', 'vegetable', 'meat']	['fruit', 'butter_milk']	['meat', 'pastry', 'vegetable', 'soda']	['milk', 'fruit', 'shopping_bags']
['bag', 'vegetable', 'meat']	['butter', 'vegetable', 'fruit']	['cheese', 'pastry', 'vegetable', 'processed_meat']	['fruit', 'condiment']	['meat', 'poultry']	['milk', 'fruit', 'vegetable', 'meat']
['bag', 'vegetable', 'processed_meat', 'meat']	['butter', 'vegetable', 'meat', 'fruit']	['cheese', 'processed_meat', 'soda']	['fruit', 'meat', 'condiment']	['meat', 'snack']	['milk', 'fruit', 'vegetable', 'processed_meat']
['bag', 'vegetable', 'processed_meat']	['butter', 'vegetable', 'meat']	['cheese', 'processed_meat']	['fruit', 'meat', 'soda']	['meat', 'soda']	['milk', 'fruit', 'yogurt', 'meat']
['bag', 'vegetable', 'shopping_bags', 'processed_meat']	['butter', 'yogurt', 'fruit']	['cheese', 'shopping_bags', 'meat']	['fruit', 'meat']	['meat', 'waffles']	['milk', 'fruit', 'yogurt', 'processed_meat']
['bag', 'vegetable']	['butter', 'yogurt', 'meat']	['cheese', 'shopping_bags', 'pastry']	['fruit', 'pastry', 'condiment', 'meat']	['milk', 'bag', 'fruit']	['milk', 'juice', 'meat']
['bag', 'yogurt']	['butter', 'yogurt', 'pastry']	['cheese', 'shopping_bags']	['fruit', 'pastry', 'condiment']	['milk', 'bag', 'meat', 'processed_meat']	['milk', 'juice', 'processed_meat']
['bag']	['butter', 'yogurt', 'processed_meat']	['cheese', 'sweets', 'meat']	['fruit', 'pastry', 'meat', 'soda']	['milk', 'bag', 'meat']	['milk', 'meat', 'beer']
['bottled_water', 'bag']	['butter', 'yogurt', 'vegetable']	['cheese', 'sweets', 'processed_meat']	['fruit', 'pastry', 'meat']	['milk', 'bag', 'pastry']	['milk', 'meat', 'poultry']
['bottled_water', 'fruit', 'processed_meat']	['butter', 'yogurt']	['cheese', 'vegetable', 'condiment', 'meat']	['fruit', 'pastry', 'processed_meat', 'condiment']	['milk', 'bag', 'processed_meat']	['milk', 'meat', 'soda']
['bottled_water', 'pastry', 'meat']	['cheese', 'bag', 'meat']	['cheese', 'vegetable', 'condiment', 'processed_meat']	['fruit', 'pastry', 'processed_meat', 'soda']	['milk', 'bag', 'shopping_bags', 'meat']	['milk', 'pastry', 'condiment', 'meat']
['bottled_water', 'pastry', 'processed_meat']	['cheese', 'bag', 'pastry']	['cheese', 'vegetable', 'meat']	['fruit', 'pastry', 'processed_meat']	['milk', 'bag', 'shopping_bags', 'processed_mea	['milk', 'pastry', 'condiment', 'processed_meat']
['bottled_water', 'processed_meat']	['cheese', 'bag']	['cheese', 'vegetable', 'processed_meat']	['fruit', 'pastry', 'shopping_bags']	['milk', 'bag', 'vegetable']	['milk', 'pastry', 'meat', 'soda']
['bottled_water', 'shopping_bags']	['cheese', 'butter', 'fruit']	['cheese', 'yogurt', 'meat']	['fruit', 'pastry', 'vegetable', 'meat']	['milk', 'bag']	['milk', 'pastry', 'meat']
['bottled_water', 'vegetable', 'processed_meat']	['cheese', 'butter', 'meat']	['cheese', 'yogurt', 'pastry', 'meat']	['fruit', 'poultry']	['milk', 'bottled_water', 'processed_meat']	['milk', 'pastry', 'poultry']
['butter_milk', 'meat']	['cheese', 'butter', 'processed_meat']	['cheese', 'yogurt', 'pastry', 'processed_meat']	['fruit', 'processed_meat', 'condiment']	['milk', 'butter_milk', 'meat']	['milk', 'pastry', 'processed_meat', 'soda']
['butter_milk', 'pastry']	['cheese', 'butter', 'vegetable']	['cheese', 'yogurt', 'processed_meat', 'meat']	['fruit', 'processed_meat', 'meat', 'condiment']	['milk', 'butter_milk', 'pastry']	['milk', 'pastry', 'processed_meat']
['butter_milk', 'vegetable']	['cheese', 'butter', 'yogurt']	['cheese', 'yogurt', 'processed_meat']	['fruit', 'processed_meat', 'soda']	['milk', 'butter_milk', 'vegetable']	['milk', 'pastry', 'vegetable', 'meat']
['butter_milk']	['cheese', 'condiment', 'meat', 'processed_meat']	['cheese', 'yogurt', 'vegetable', 'meat']	['fruit', 'processed_meat']	['milk', 'butter_milk']	['milk', 'pastry', 'vegetable', 'processed_meat']

['milk', 'poultry']	['pastry', 'poultry']	['sweets', 'fruit', 'meat']	['vegetable', 'processed_meat']
['milk', 'processed_meat', 'beer']	['pastry', 'processed_meat', 'beer']	['sweets', 'fruit', 'pastry', 'meat']	['vegetable', 'shopping_bags', 'condiment']
['milk', 'processed_meat', 'soda']	['pastry', 'processed_meat', 'poultry']	['sweets', 'fruit', 'pastry', 'processed_meat']	['vegetable', 'shopping_bags', 'processed_meat', 'meat']
['milk', 'processed_meat']	['pastry', 'processed_meat', 'soda']	['sweets', 'fruit', 'processed_meat', 'meat']	['vegetable', 'shopping_bags', 'processed_meat']
['milk', 'shopping_bags', 'meat', 'processed_mea	['pastry', 'processed_meat']	['sweets', 'fruit', 'processed_meat', 'vegetable']	['vegetable', 'yogurt', 'condiment', 'meat']
['milk', 'shopping_bags', 'meat']	['pastry', 'shopping_bags', 'processed_meat', 'meat']	['sweets', 'fruit', 'processed_meat']	['vegetable', 'yogurt', 'meat', 'fruit', 'processed_meat']
['milk', 'shopping_bags', 'pastry']	['pastry', 'shopping_bags', 'vegetable', 'meat']	['sweets', 'fruit', 'vegetable', 'meat']	['vegetable', 'yogurt', 'pastry', 'fruit', 'processed_meat']
['milk', 'shopping_bags', 'processed_meat']	['pastry', 'vegetable', 'meat']	['sweets', 'meat', 'vegetable']	['vegetable', 'yogurt', 'processed_meat']
['milk', 'shopping_bags', 'vegetable']	['pastry', 'yogurt', 'meat', 'milk', 'processed_meat']	['sweets', 'meat']	['yogurt', 'condiment', 'meat']
['milk', 'shopping_bags']	['pastry', 'yogurt', 'processed_meat']	['sweets', 'pastry', 'meat']	['yogurt', 'condiment']
['milk', 'sweets', 'meat']	['processed_meat', 'bag', 'shopping_bags', 'pastry']	['sweets', 'pastry', 'processed_meat', 'vegetable']	['yogurt', 'meat', 'milk', 'fruit', 'processed_meat']
['milk', 'sweets', 'pastry', 'meat']	['processed_meat', 'beer']	['sweets', 'pastry', 'processed_meat']	['yogurt', 'meat', 'soda']
['milk', 'sweets', 'pastry', 'processed_meat']	['processed_meat', 'curd']	['sweets', 'pastry', 'vegetable', 'meat']	['yogurt', 'meat']
['milk', 'sweets', 'processed_meat']	['processed_meat', 'liquor']	['sweets', 'processed_meat', 'vegetable']	['yogurt', 'pastry', 'condiment']
['milk', 'sweets', 'vegetable', 'meat']	['processed_meat', 'meat', 'cheese', 'fruit', 'vegetable']	['sweets', 'processed_meat']	['yogurt', 'pastry', 'meat', 'fruit', 'processed_meat']
['milk', 'sweets', 'vegetable', 'processed_meat']	['processed_meat', 'meat', 'milk', 'cheese', 'vegetable']	['sweets', 'shopping_bags']	['yogurt', 'pastry', 'meat', 'fruit', 'vegetable']
['milk', 'vegetable', 'condiment', 'processed_mea	['processed_meat', 'meat', 'poultry']	['vegetable', 'butter', 'condiment', 'fruit']	['yogurt', 'pastry', 'meat', 'milk', 'vegetable']
['milk', 'vegetable', 'processed_meat']	['processed_meat', 'milk', 'fruit', 'condiment', 'vegetable']	['vegetable', 'butter', 'condiment', 'meat']	['yogurt', 'pastry', 'meat']
['milk', 'yogurt', 'condiment']	['processed_meat', 'pastry', 'cheese', 'fruit', 'vegetable']	['vegetable', 'butter', 'pastry', 'processed_meat']	['yogurt', 'pastry', 'milk', 'fruit', 'meat']
['milk', 'yogurt', 'meat', 'processed_meat']	['processed_meat', 'pastry', 'milk', 'butter', 'vegetable']	['vegetable', 'butter', 'processed_meat', 'condiment']	['yogurt', 'pastry', 'milk', 'fruit', 'processed_meat']
['milk', 'yogurt', 'meat']	['processed_meat', 'pastry', 'milk', 'condiment', 'vegetable']	['vegetable', 'butter', 'processed_meat', 'fruit']	['yogurt', 'pastry', 'vegetable', 'meat']
['milk', 'yogurt', 'pastry', 'meat']	['processed_meat', 'pastry', 'milk', 'fruit', 'vegetable']	['vegetable', 'butter', 'processed_meat', 'meat']	['yogurt', 'processed_meat', 'soda']
['milk', 'yogurt', 'pastry', 'processed_meat']	['processed_meat', 'poultry']	['vegetable', 'butter', 'processed_meat']	['yogurt', 'processed_meat']
['milk', 'yogurt', 'processed_meat']	['processed_meat', 'shopping_bags', 'pastry']	['vegetable', 'cleaning_product', 'processed_meat']	['yogurt', 'shopping_bags']
['milk', 'yogurt', 'vegetable', 'meat']	['processed_meat', 'snack']	['vegetable', 'condiment', 'pastry', 'processed_meat']	['yogurt', 'vegetable', 'meat']
['milk', 'yogurt', 'vegetable', 'processed_meat']	['processed_meat', 'soda']	['vegetable', 'condiment', 'processed_meat']	['yogurt', 'vegetable', 'pastry', 'processed_meat']
['misc_beverages', 'meat']	['processed_meat', 'waffles']	['vegetable', 'condiment', 'yogurt', 'processed_meat']	
['misc_beverages', 'processed_meat']	['processed_meat', 'yogurt', 'meat', 'milk', 'vegetable']	['vegetable', 'fruit', 'pastry', 'processed_meat']	
['newspapers', 'processed_meat']	['processed_meat', 'yogurt', 'pastry', 'milk', 'vegetable']	['vegetable', 'fruit', 'processed_meat', 'condiment']	
['packaged_fruit/vegetables', 'fruit']	['processed_meat']	['vegetable', 'fruit', 'processed_meat']	
['packaged_fruit/vegetables', 'vegetable']	['shopping_bags', 'beer']	['vegetable', 'fruit', 'yogurt', 'processed_meat']	
['packaged_fruit/vegetables']	['shopping_bags', 'condiment']	['vegetable', 'meat', 'curd']	
['pastry', 'condiment', 'meat']	['shopping_bags', 'liquor']	['vegetable', 'meat', 'fruit', 'butter', 'processed_meat']	
['pastry', 'fruit', 'yogurt', 'processed_meat']	['shopping_bags', 'meat', 'soda']	['vegetable', 'milk', 'fruit', 'butter', 'meat']	
['pastry', 'meat', 'beer']	['shopping_bags', 'meat']	['vegetable', 'milk', 'fruit', 'butter', 'processed_meat']	
['pastry', 'meat', 'cheese', 'fruit', 'vegetable']	['shopping_bags', 'pastry', 'meat']	['vegetable', 'pastry', 'fruit', 'condiment', 'processed_meat']	
['pastry', 'meat', 'fruit', 'condiment', 'processed_	['shopping_bags', 'pastry', 'soda']	['vegetable', 'pastry', 'meat', 'butter', 'processed_meat']	
['pastry', 'meat', 'fruit', 'condiment', 'vegetable']	['shopping_bags', 'pastry', 'vegetable']	['vegetable', 'pastry', 'milk', 'cheese', 'meat']	
['pastry', 'meat', 'milk', 'butter', 'vegetable']	['shopping_bags', 'pastry']	['vegetable', 'pastry', 'milk', 'cheese', 'processed_meat']	
['pastry', 'meat', 'milk', 'condiment', 'processed_	['shopping_bags', 'processed_meat', 'meat']	['vegetable', 'pastry', 'processed_meat', 'soda']	
['pastry', 'meat', 'milk', 'fruit', 'processed_meat']	['shopping_bags', 'processed_meat']	['vegetable', 'pastry', 'processed_meat']	
['pastry', 'meat', 'milk', 'fruit', 'vegetable']	['shopping_bags', 'soda']	['vegetable', 'processed_meat', 'beer']	
['pastry', 'meat', 'poultry']	['shopping_bags', 'vegetable', 'meat']	['vegetable', 'processed_meat', 'curd']	
['pastry', 'meat', 'soda']	['shopping_bags', 'vegetable']	['vegetable', 'processed_meat', 'poultry']	
['pastry', 'meat']	['shopping_bags']	['vegetable', 'processed_meat', 'soda']	

References

1) **"Association Rules and the Apriori Algorithm: A Tutorial." *KDnuggets*, 2016, www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html**
2) **"Instacart Market Basket Analysis." *Kaggle*, www.kaggle.com/c/instacart-market-basket-analysis/data.**
3) **Ladley, John, and IDG Contributor Network. "Mastering and Managing Data Understanding." *CIO*, CIO, 28 July 2016, www.cio.com/article/3097501/mastering-and-managing-data-understanding.html.**
4) **"Market Basket Analysis." *Data Mining: Market Basket Analysis*, Albion Research Ltd., www.albionresearch.com/data_mining/market_basket.php.**
5) **"Market Basket Analysis" http://marketing409.blogspot.com/2009/04/market-basket-analysis.html**
6) **Mohit Yadav. "Apriori Algorithm." *GeeksforGeeks*, 6 Mar. 2019, www.geeksforgeeks.org/apriori-algorithm/.**
7) **Nasrullah, Irfan. "Groceries Market Basket Dataset." *Kaggle*, Kaggle, 16 July 2019, www.kaggle.com/irfanasrullah/groceries.**
8) **"Python Programming Guide." *Python Programming Guide - Spark 0.9.1 Documentation*, spark.apache.org/docs/0.9.1/python-programming-guide.html.**
9) **Rajput, Abhishek. "Python: Data Analysis Using Pandas." *GeeksforGeeks*, 20 June 2018, www.geeksforgeeks.org/python-data-analysis-using-pandas/.**
10) **"What Is a Business Objective? Definition and Meaning." *Market Business News*, Market Business News, 24 Apr. 2019, marketbusinessnews.com/financial-glossary/business-objective-definition-meaning.**