Market Basket Analysis

Table of Content

Table of Figures

1. Project Overview

1.1 Introduction

Data mining techniques have led to transformations in various industries. In the retail industry these approaches can lead to significant business growth. Insight generated by various data-driven approaches can be used for smart product placement and various marketing strategies. In environments such as cafes, restaurants, and bakeries, it is very important to understand customer purchasing behavior patterns. Having a good understanding of these topics can lead to increased sales and higher customer satisfaction.

Market Basket Analysis is one of the most effective techniques to identify purchasing patterns. Market Basket Analysis is a technique in which items that are frequently bought together can be identified. Market Basket Analysis helps businesses identify hidden associations between products. It can also help businesses to make better promotion strategies and better optimize the store. (Manpreet Kaur, 2016)

To further explore Market Basket Analysis, this technique will be applied to a transactional dataset. These techniques can help reveal various meaningful relationships between items in a customer transaction. When these techniques are applied effectively, generated insights can bring various improvements such as higher sales, higher customer satisfaction, and better cross-selling opportunities for a business.

1.2 Problem Statement

In the food and beverage industry, large amounts of transactional data are generated every day. Despite such a large volume of data, most of the data remains underutilized. Many businesses lack the tools, techniques, and knowledge of extracting meaningful data from it. Without proper data mining and analysis, various critical business factors such as customer pattern recognition and customer behavior recognition can go unnoticed. (Marselina, Jaman, & Kurniawan, 2023)

Lack of these insights limits the ability of businesses to make informed and data-driven decisions regarding their products. As these businesses produce large volumes of data, manual data analysis may be impractical. Application of data mining techniques in these businesses can lead to higher customer satisfaction and help businesses come up with better promotional strategies.

This project aims to address these issues by applying Rule Mining Techniques to a real-world bakery dataset, with the goal of generating frequent item combinations and rules that can reveal insights that can be applied directly to real-world scenarios.

1.3 Rationale for Dataset Selection

For an effective Market Basket Analysis selection of an appropriate dataset is critical. For this project, a publicly available bakery dataset was chosen, as this dataset contains detailed transactional data from a real-world environment. This dataset includes which bakery items were bought together at a certain transaction.

The transactional format of this dataset makes it suitable to apply market basket analysis. Frequent item combinations and customer behavior patterns can be extracted from this dataset. Furthermore, this dataset is clean and requires minimal processing to apply rule mining algorithms. These characteristics of this dataset make it an ideal choice for this project.

1.4 Project Objectives

The goal of this project is to conduct market basket analysis on a bakery transactional dataset by applying association rule mining techniques to discover meaningful patterns, insights, and relationships. The primary objectives of this project are:

1. To perform Market Basket analysis on transactional data to understand customer purchasing behavior and to uncover various patterns.
2. To apply association rule mining algorithms to identify the most and least frequent items and to generate association rules.
3. To evaluate the quality of rules generated based on metrics such as support, confidence, and lift.
4. To compare different rule mining techniques in terms of execution time and rule quality.
5. To produce actionable insights that can inform decisions that can be made in areas such as product placement and marketing strategies.

This project aims to uncover hidden patterns from the bakery dataset, enabling bakeries to make more informed business decisions to increase sales and to provide higher customer satisfaction using market basket analysis.

2. Dataset Description

This project aims to conduct market basket analysis on a real-life transactional dataset. To fulfil this, a publicly available data set containing transactional data from a bakery was selected. This data can be used to produce real-life-like insights. This dataset includes various transactional features that can help generate insights that can be applied to real-life scenarios.

Source of this dataset: The dataset used in this project is sourced from Kaggle. This dataset is titled **Bakery Sales Dataset** and is provided by **Akashdeep Kuila**.

Kaggle link: https://www.kaggle.com/datasets/akashdeepkuila/bakery

Dataset Domain

This dataset falls under the retail and service domain as it captures sales data from a bakery.

Features of the Dataset

This dataset contains 5 features; features of this dataset are:

- **Transaction**: A unique id or code for every single transaction. This feature is numerical.
- **Items**: Items purchased in each transaction. This feature is categorical.
- **date_time**: Date and time of when a transaction was made. This feature is categorical.
- **period_day**: Which part of the day the transaction was made (Morning, Afternoon, Evening, Night). This feature is categorical.
- **weekday_weekend**: Classifies whether a transaction was made on weekday or weekend. This feature is categorical.

In this dataset, each row represents a transaction of a single item. This dataset has more than 20500 rows. The feature **transaction** has duplicated values, showcasing that in a transaction, multiple items have been bought. This dataset has more than 9500 unique transactions.

Sample of the dataset

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	2016-10-30 09:58:00	Morning	Weekend
1	2	Scandinavian	2016-10-30 10:05:00	Morning	Weekend
2	2	Scandinavian	2016-10-30 10:05:00	Morning	Weekend
3	3	Hot chocolate	2016-10-30 10:07:00	Morning	Weekend
4	3	Jam	2016-10-30 10:07:00	Morning	Weekend
5	3	Cookies	2016-10-30 10:07:00	Morning	Weekend
6	4	Muffin	2016-10-30 10:08:00	Morning	Weekend
7	5	Coffee	2016-10-30 10:13:00	Morning	Weekend
8	5	Pastry	2016-10-30 10:13:00	Morning	Weekend
9	5	Bread	2016-10-30 10:13:00	Morning	Weekend
10	6	Medialuna	2016-10-30 10:16:00	Morning	Weekend
11	6	Pastry	2016-10-30 10:16:00	Morning	Weekend
12	6	Muffin	2016-10-30 10:16:00	Morning	Weekend
13	7	Medialuna	2016-10-30 10:19:00	Morning	Weekend
14	7	Pastry	2016-10-30 10:19:00	Morning	Weekend
15	7	Coffee	2016-10-30 10:19:00	Morning	Weekend
16	7	Tea	2016-10-30 10:19:00	Morning	Weekend
17	8	Pastry	2016-10-30 10:20:00	Morning	Weekend
18	8	Bread	2016-10-30 10:20:00	Morning	Weekend
19	9	Bread	2016-10-30 10:21:00	Morning	Weekend
20	9	Muffin	2016-10-30 10:21:00	Morning	Weekend
21	10	Scandinavian	2016-10-30 10:25:00	Morning	Weekend
22	10	Medialuna	2016-10-30 10:25:00	Morning	Weekend
23	11	Bread	2016-10-30 10:27:00	Morning	Weekend
24	11	Medialuna	2016-10-30 10:27:00	Morning	Weekend

Figure 1: Dataset Sample

3. Basic Data Exploration

Before moving to the implementation of algorithms, it is necessary to analyze the data.

```
df.head()
```

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	30-10-2016 09:58	morning	weekend
1	2	Scandinavian	30-10-2016 10:05	morning	weekend
2	2	Scandinavian	30-10-2016 10:05	morning	weekend
3	3	Hot chocolate	30-10-2016 10:07	morning	weekend
4	3	Jam	30-10-2016 10:07	morning	weekend

Figure 2: Data Head

This data contains five columns transaction, item, data_time and weekday_weekend.

```
df.shape
```

```
(20507, 5)
```

```
df.dtypes
```

```
Transaction         int64
Item               object
date_time          object
period_day         object
weekday_weekend    object
dtype: object
```

Figure 3: Data Shape and Type

This dataset has 20507 entries. Four features of this dataset are categorical and one is numerical.

```
Unique values in 'Transaction':
Transaction
6716    11
6279    11
6474    11
6412    11
6045    10
        ..
4495     1
4494     1
4492     1
4491     1
9684     1
Name: count, Length: 9465, dtype: int64

Unique values in 'Item':
Item
Coffee            5471
Bread             3325
Tea               1435
Cake              1025
Pastry             856
                  ...
Bacon                1
Gift voucher         1
Olum & polenta       1
Raw bars             1
Polenta              1
Name: count, Length: 94, dtype: int64
```

```
Unique values in 'date_time':
date_time
05-02-2017 11:58    12
09-02-2017 13:44    11
11-02-2017 14:08    11
17-02-2017 14:18    11
12-02-2017 14:35    11
                    ..
11-01-2017 09:16     1
10-01-2017 16:32     1
10-01-2017 16:31     1
10-01-2017 16:29     1
09-04-2017 15:04     1
Name: count, Length: 9182, dtype: int64

Unique values in 'period_day':
period_day
afternoon    11569
morning       8404
evening        520
night           14
Name: count, dtype: int64

Unique values in 'weekday_weekend':
weekday_weekend
weekday    12807
weekend     7700
Name: count, dtype: int64
```

Figure 4: Unique Values in dataset

- This dataset includes 9465 unique transactions.
- 94 unique items have been sold in this bakery based on this dataset
- Most of the transactions have been made in the afternoon, followed by morning, evening and night. This shows that people visit the bakery more often in the daytime.
- Most of the transactions have been made on weekdays but weekends have more transactions per day, showcasing a bigger crowd at weekends.

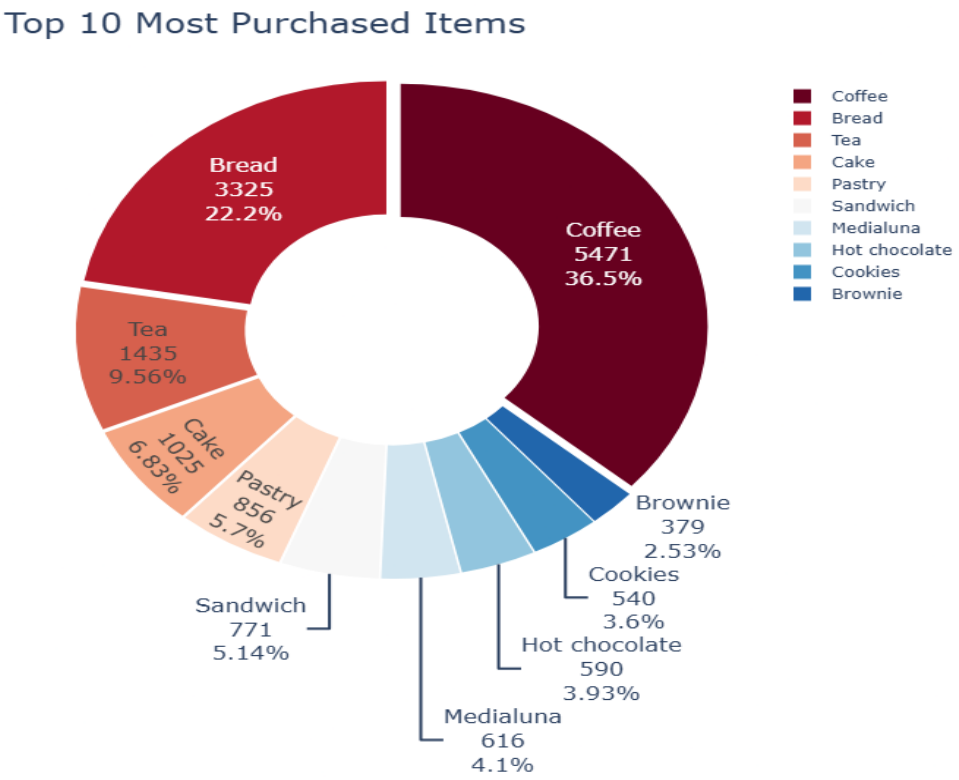

Figure 5: Top 10 Most Purchased Items

Coffee is the most purchased item in this dataset, followed by bread, tea and cake.

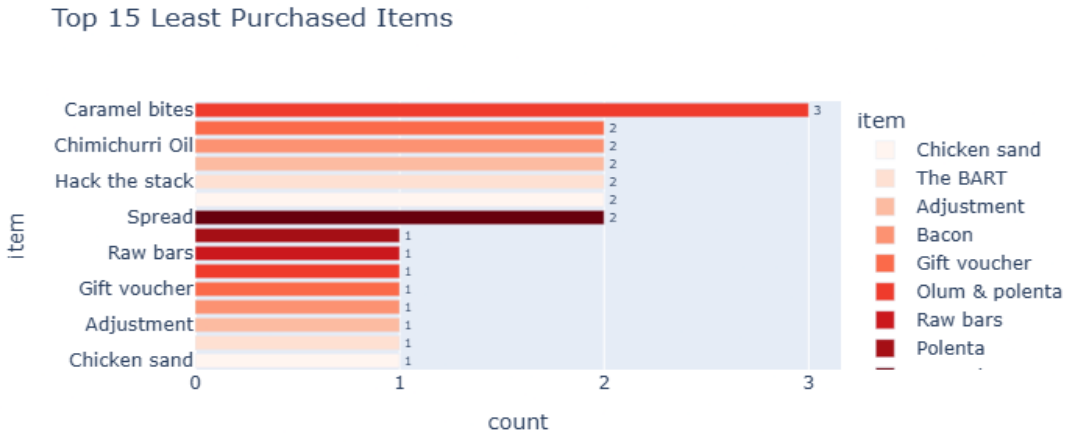

Figure 6: Top 15 least Sold

Many items do not sell well in this bakery. This also shows that most people have a common taste.

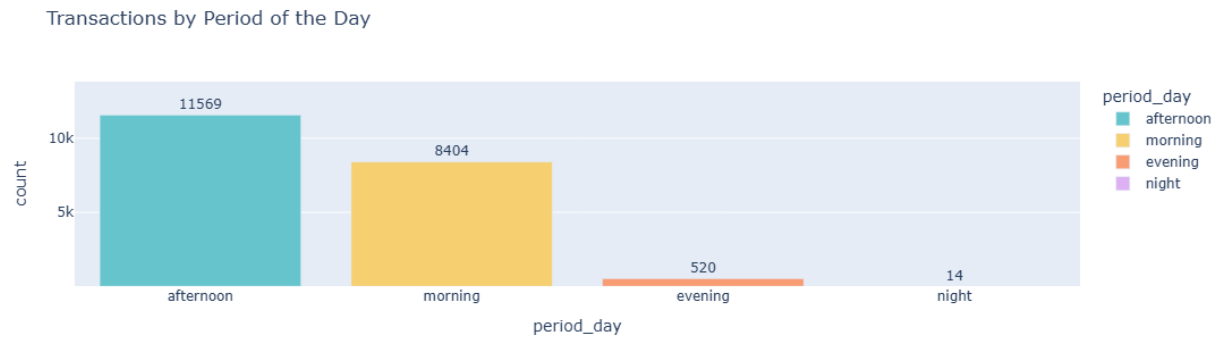

Figure 7: Transaction By period of day

Most transactions take place in the afternoon and morning, but there is very little traffic in the evening and nighttime.

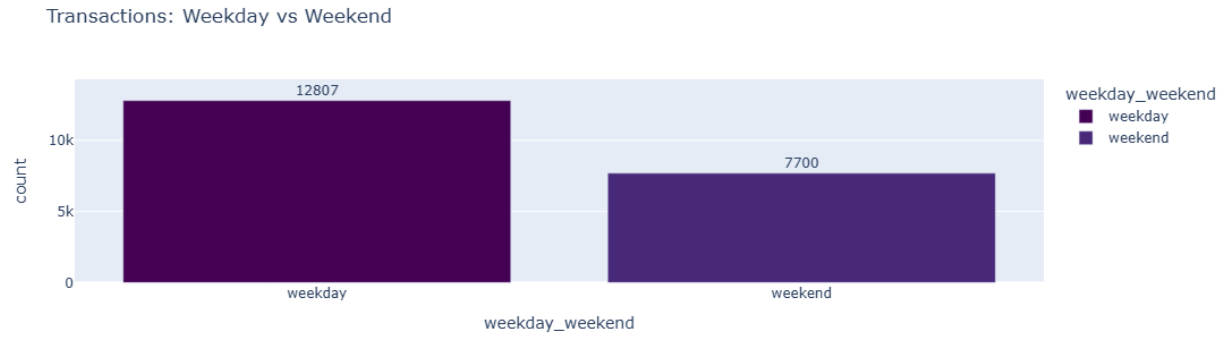

Figure 8:Weekday vs Weekend

The majority of transactions take place on weekdays but transactions per day is higher in weekends.

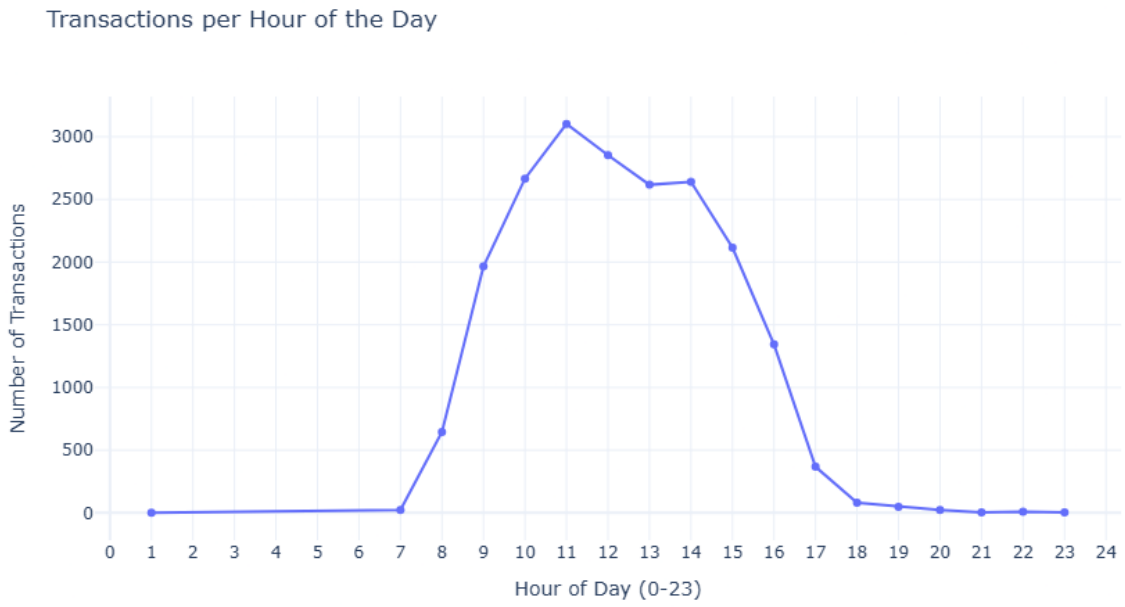

Figure 9: Transactions Per Hour

The Majority of transactions in this bakery happen between 7 am to 5 pm.

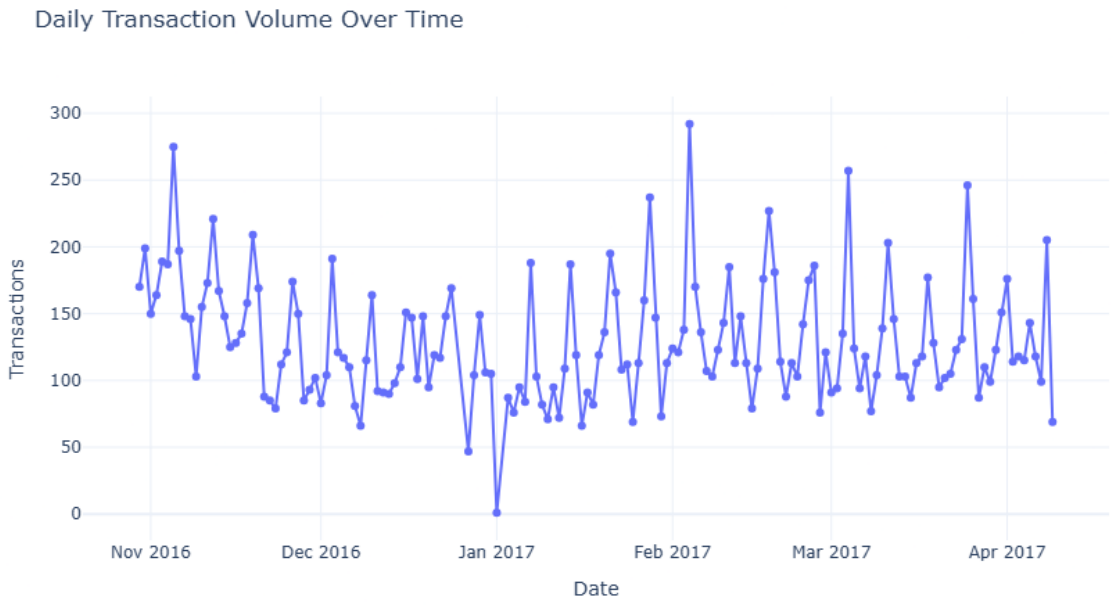

Figure 10: Daily Transaction Volume

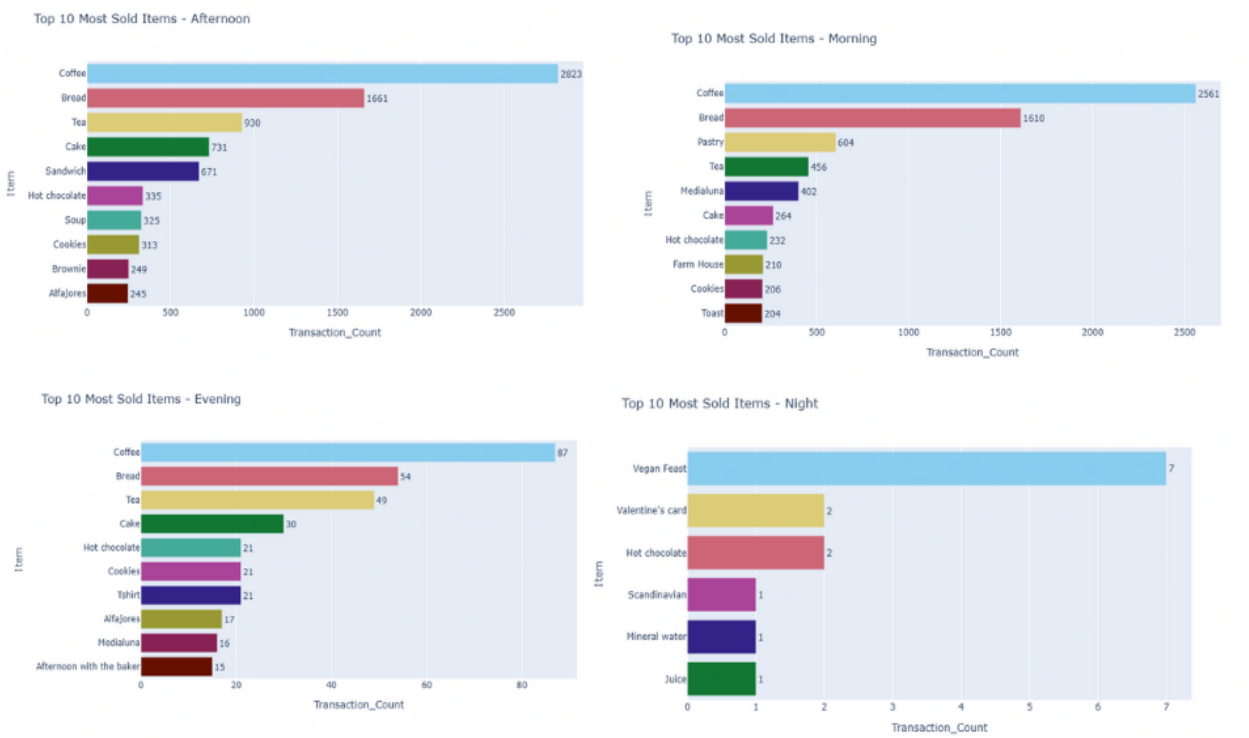

Figure 11: Most Sold items by time

- Coffee, bread and tea are popular throughout the day but at night people do not prefer them.

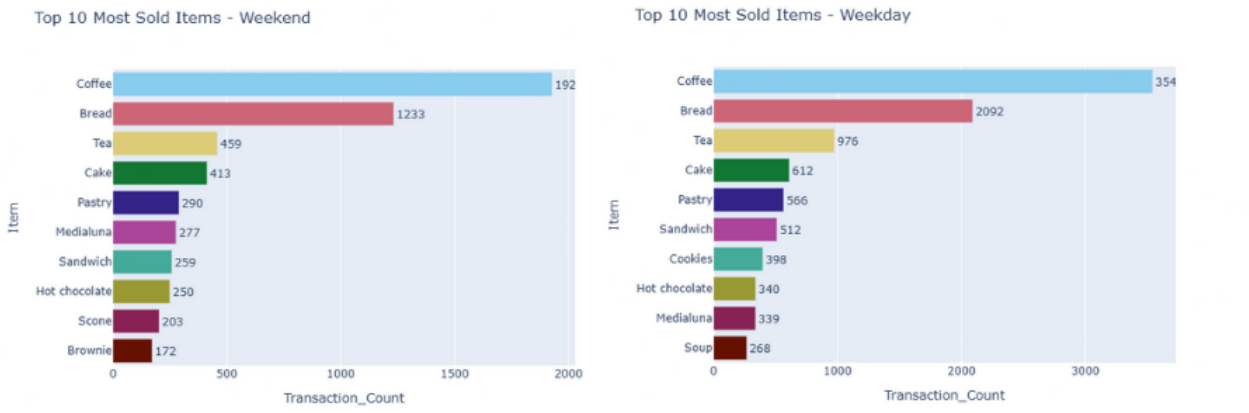

Figure 12: Most sold by type of day

- Most sold items do not change much depending on the type of day.

Figure 13: Least Sold Items by type of day

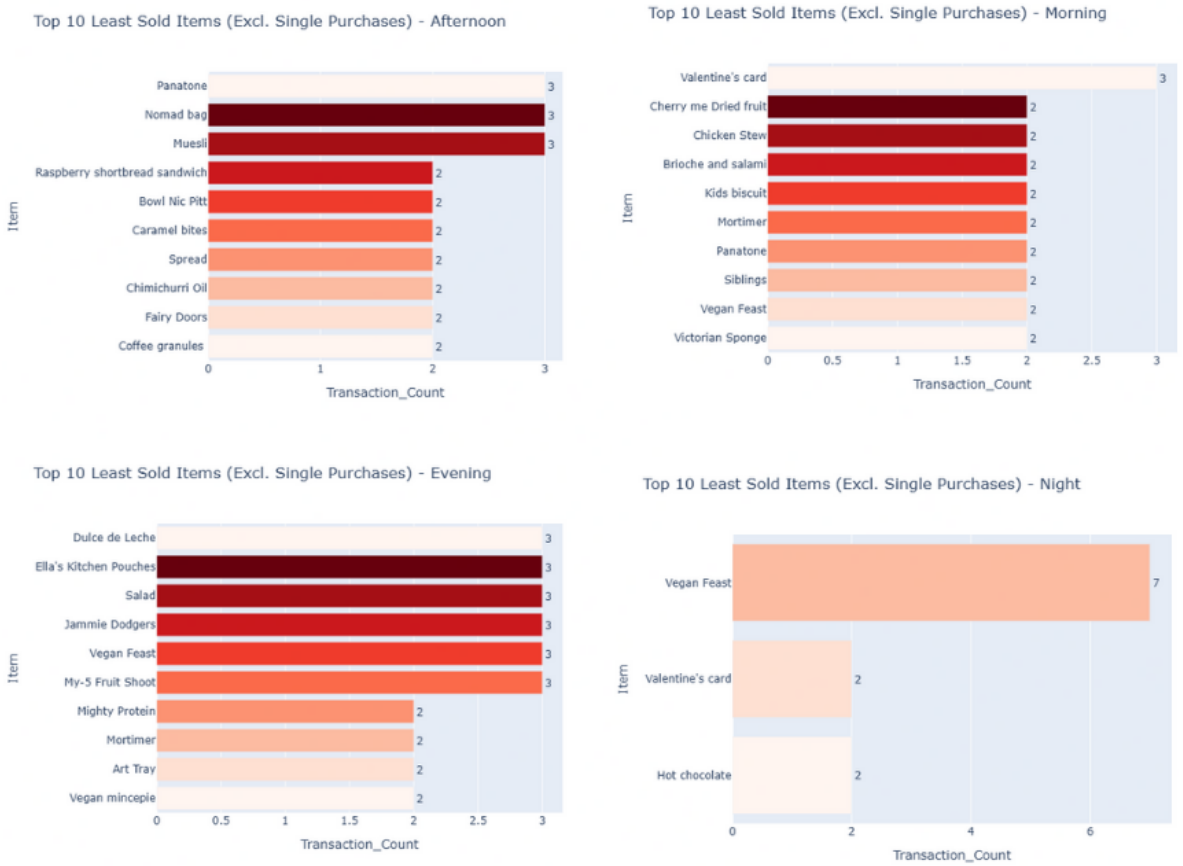

Figure 14: Least items sold by time of day

As very few items are sold at nighttime, the bar chart of the least sold items and the most sold items at night are identical.

4. Data Preprocessing

Before applying association rule mining, we need to clean and preprocess the data to a format suitable for market basket analysis. These steps ensure that the data is accurately represented into a structured format for association rule mining algorithms.

4.1 Data Cleaning

Handling Missing Values

```
df.isnull().sum()

Transaction         0
Item                0
date_time           0
period_day          0
weekday_weekend     0
dtype: int64
```

Figure 15: Checking for Null

As there are no missing values in this dataset, there is no need to handle missing values.

Handling Outliers and Noisy Data

- As that is needed for association rule mining is in categorical format, there is no need to treat outliers
- For more accurate and comprehensive rule-based learning, the noisy data (e.g. items which have only been sold once) are not going to be removed.

4.2 Grouping Items by transaction

In this dataset each row represents the sale of a single item, due to this reason the data needs to be grouped into different transactions.

```
transactions_series = df.groupby('Transaction')['Item'].apply(list)
transactions_list = transactions_series.tolist()
transactions_list
```

```
[['Bread'],
 ['Scandinavian', 'Scandinavian'],
 ['Hot chocolate', 'Jam', 'Cookies'],
 ['Muffin'],
 ['Coffee', 'Pastry', 'Bread'],
 ['Medialuna', 'Pastry', 'Muffin'],
 ['Medialuna', 'Pastry', 'Coffee', 'Tea'],
 ['Pastry', 'Bread'],
 ['Bread', 'Muffin'],
 ['Scandinavian', 'Medialuna'],
 ['Bread', 'Medialuna', 'Bread'],
 ['Jam', 'Coffee', 'Tartine', 'Pastry', 'Tea'],
 ['Basket', 'Bread', 'Coffee'],
 ['Bread', 'Medialuna', 'Pastry'],
 ['Mineral water', 'Scandinavian'],
 ['Bread', 'Medialuna', 'Coffee'],
 ['Hot chocolate'],
 ['Farm House'],
```

Figure 16: Grouping transactions based on transaction id

4.3 Data Transformation

In order to apply association rule mining techniques, such as the Apriori algorithm, to our dataset, the grouped transactional data needs to be transformed into a format suitable for analysis. In this project, to convert the data into a suitable format, a **transaction encoder** is going to be used. The **transaction encoder** takes data and transforms it into a Boolean array where each row represents a transaction, and each column represents a unique item. After using the transaction encoder on this data, the data will be suitable for association rule mining algorithms.

Data Before Transformation

```
df.head()
```

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	30-10-2016 09:58	morning	weekend
1	2	Scandinavian	30-10-2016 10:05	morning	weekend
2	2	Scandinavian	30-10-2016 10:05	morning	weekend
3	3	Hot chocolate	30-10-2016 10:07	morning	weekend
4	3	Jam	30-10-2016 10:07	morning	weekend

Figure 17: Data Before Transformation

Data Transformation Using Transaction Encoder

```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(transactions_list).transform(transactions_list)
basket = pd.DataFrame(te_ary, columns=te.columns_)
```

Figure 18: Data Transformation

Data After Transformation

```
basket.head()
```

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Truffles	Tshirt	Valentine's card	Vega Fea
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	Fal

5 rows × 94 columns

Figure 19: Data After Transformation

Now, this processed data can be used for association rule mining.

5. Implementation of Apriori Algorithm

The Apriori Algorithm is an association rule mining technique that is used to discover frequent itemsets. Once frequent itemsets are identified, the algorithm generates association rules. These rules are evaluated based on support and confidence. (Hegland, 2007)

In this project Apriori Algorithm is used using **transaction encoder** and **apriori** from **mlxtend**.

5.1 Frequent Itemset Generation

The Apriori algorithm was used on the transformed data to identify item sets that occur together. Initially, minimum support was configured to 0.01. Using these parameters, item sets that occur in at least 1% of transactions are considered.

```
frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)
```

```
frequent_itemsets.head(10)
```

	support	itemsets
0	0.036344	(Alfajores)
1	0.016059	(Baguette)
2	0.327205	(Bread)
3	0.040042	(Brownie)
4	0.103856	(Cake)
5	0.012995	(Chicken Stew)
6	0.478394	(Coffee)
7	0.019440	(Coke)
8	0.054411	(Cookies)
9	0.039197	(Farm House)

Figure 20: Frequent Itemset generation

5.2 Association Rule Extraction

From frequent item sets, various association rules were generated. These rules can help us identify relationships between products. A lift value of one was chosen to determine how strongly items are associated. Minimum lift value of 1 indicates that only positive associations are considered. Other metrics, such as support and confidence, are also considered in these rules.

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	ku
0	(Alfajores)	(Coffee)	0.036344	0.478394	0.019651	0.540698	1.130235	1.0	0.002264	1.135648	0.119574	0.039693	0.119446	0
1	(Coffee)	(Alfajores)	0.478394	0.036344	0.019651	0.041078	1.130235	1.0	0.002264	1.004936	0.220910	0.039693	0.004912	0
2	(Bread)	(Pastry)	0.327205	0.086107	0.029160	0.089119	1.034977	1.0	0.000985	1.003306	0.050231	0.075908	0.003296	0
3	(Pastry)	(Bread)	0.086107	0.327205	0.029160	0.338650	1.034977	1.0	0.000985	1.017305	0.036980	0.075908	0.017011	0
4	(Coffee)	(Brownie)	0.478394	0.040042	0.019651	0.041078	1.025860	1.0	0.000495	1.001080	0.048327	0.039398	0.001079	0
5	(Brownie)	(Coffee)	0.040042	0.478394	0.019651	0.490765	1.025860	1.0	0.000495	1.024293	0.026259	0.039398	0.023717	0
6	(Coffee)	(Cake)	0.478394	0.103856	0.054728	0.114399	1.101515	1.0	0.005044	1.011905	0.176684	0.103745	0.011765	0
7	(Cake)	(Coffee)	0.103856	0.478394	0.054728	0.526958	1.101515	1.0	0.005044	1.102664	0.102840	0.103745	0.093105	0

Figure 21: Apriori Rules

5.3 Hyperparameter Tuning and Rule Comparison

To understand how parameters like minimum confidence, support, and lift impact on production of rules and to select the most appropriate parameters, the Apriori algorithm is experimented with by passing various parameters.

```
param_grid = [
    {'min_support': 0.01, 'metric': 'lift', 'min_threshold': 1.0},
    {'min_support': 0.03, 'metric': 'lift', 'min_threshold': 1.0},
    {'min_support': 0.05, 'metric': 'lift', 'min_threshold': 1.0},
    {'min_support': 0.05, 'metric': 'confidence', 'min_threshold': 0.6},
    {'min_support': 0.05, 'metric': 'confidence', 'min_threshold': 0.8},
    {'min_support': 0.1, 'metric': 'lift', 'min_threshold': 1.2},
]

for params in param_grid:
    print(f"\nParameters: min_support={params['min_support']}, metric={params['metric']}, min_threshold={params['min_threshold']}")

    frequent_itemsets = apriori(basket, min_support=params['min_support'], use_colnames=True)

    rules = association_rules(frequent_itemsets, metric=params['metric'], min_threshold=params['min_threshold'])

    print(f"Frequent Itemsets: {len(frequent_itemsets)}")
    print(f"Rules Generated: {len(rules)}")
    print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head(5))
```

Figure 22: Hyperparameter Tuning

```
Parameters: min_support=0.01, metric=lift, min_threshold=1.0
Frequent Itemsets: 61
Rules Generated: 42
   antecedents  consequents   support  confidence      lift
0  (Alfajores)     (Coffee)  0.019651    0.540698  1.130235
1     (Coffee)  (Alfajores)  0.019651    0.041078  1.130235
2      (Bread)     (Pastry)  0.029160    0.089119  1.034977
3     (Pastry)      (Bread)  0.029160    0.338650  1.034977
4     (Coffee)    (Brownie)  0.019651    0.041078  1.025860

Parameters: min_support=0.03, metric=lift, min_threshold=1.0
Frequent Itemsets: 23
Rules Generated: 8
   antecedents  consequents   support  confidence      lift
0     (Coffee)       (Cake)  0.054728    0.114399  1.101515
1       (Cake)     (Coffee)  0.054728    0.526958  1.101515
2     (Coffee)  (Medialuna)  0.035182    0.073542  1.189878
3  (Medialuna)     (Coffee)  0.035182    0.569231  1.189878
4     (Coffee)     (Pastry)  0.047544    0.099382  1.154168
```

```
Parameters: min_support=0.05, metric=lift, min_threshold=1.0
Frequent Itemsets: 11
Rules Generated: 2
  antecedents consequents   support  confidence      lift
0    (Coffee)      (Cake)  0.054728    0.114399  1.101515
1      (Cake)    (Coffee)  0.054728    0.526958  1.101515

Parameters: min_support=0.05, metric=confidence, min_threshold=0.6
Frequent Itemsets: 11
Rules Generated: 0
Empty DataFrame
Columns: [antecedents, consequents, support, confidence, lift]
Index: []

Parameters: min_support=0.05, metric=confidence, min_threshold=0.8
Frequent Itemsets: 11
Rules Generated: 0
Empty DataFrame
Columns: [antecedents, consequents, support, confidence, lift]
Index: []

Parameters: min_support=0.1, metric=lift, min_threshold=1.2
Frequent Itemsets: 4
Rules Generated: 0
Empty DataFrame
Columns: [antecedents, consequents, support, confidence, lift]
Index: []
```

Figure 23: Hyperparameter Tuning Results

- **Parameters: minimum support = 0.01, minimum lift = 1** generated the largest number of rules and frequent item sets. These parameters captured many patterns including both strong and moderate ones.
- **Parameters: minimum support = 0.03, minimum lift = 1** generated 23 frequent item sets and only 8 rules, these parameters also allowed items which have appeared in at least 3% of transactions.
- **Parameters: minimum support = 0.05, minimum lift = 1** generated only 11 frequent item sets and 2 rules, these parameters are too strict for rich insights.
- **Parameters: minimum support = 0.05, minimum lift = 0.6/0.8** did not generate any rules as these require that at least 60/80% of time consequent must follow the antecedent.
- **Parameters: minimum support = 0.1, minimum lift = 1.2** only generated 4 item sets and no rules as lift greater than 1.2 only looks for very strong relationships.

After analyzing the results generated by hyperparameter tuning **Parameters: minimum support = 0.01, minimum lift = 1** is the most optimal for our use case as large number of rules with strong and moderate associations were generated.

5.4 Visualizations

For **Parameters: minimum support = 0.01, minimum lift = 1** these were the top 10 association rules

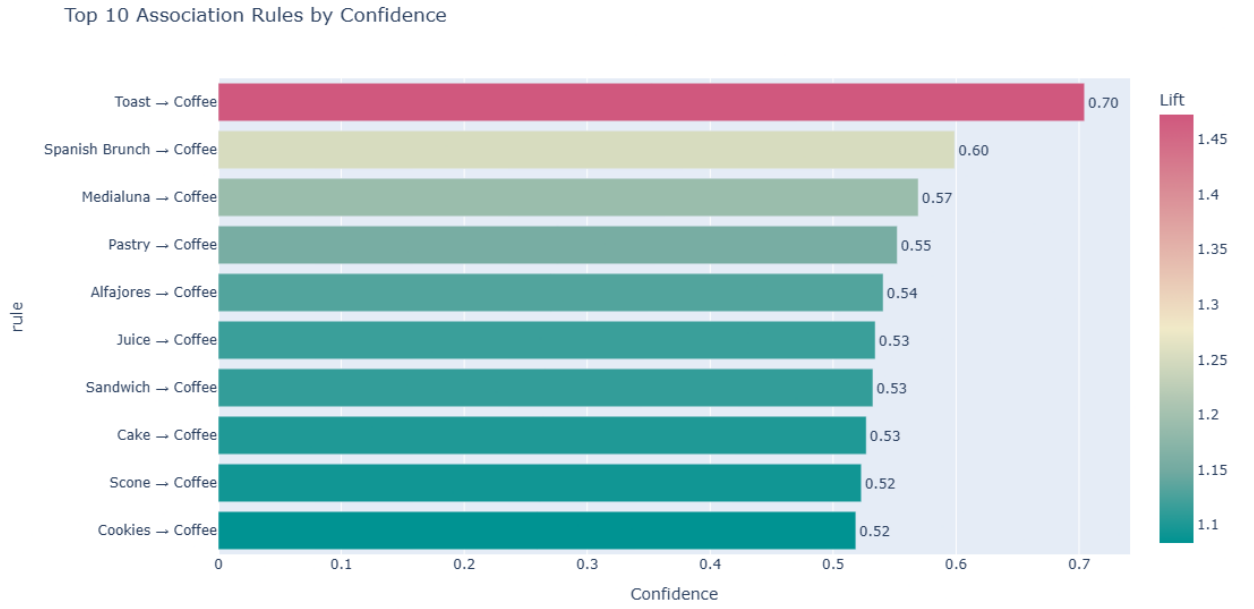

Figure 24: Top 10 Association Rules

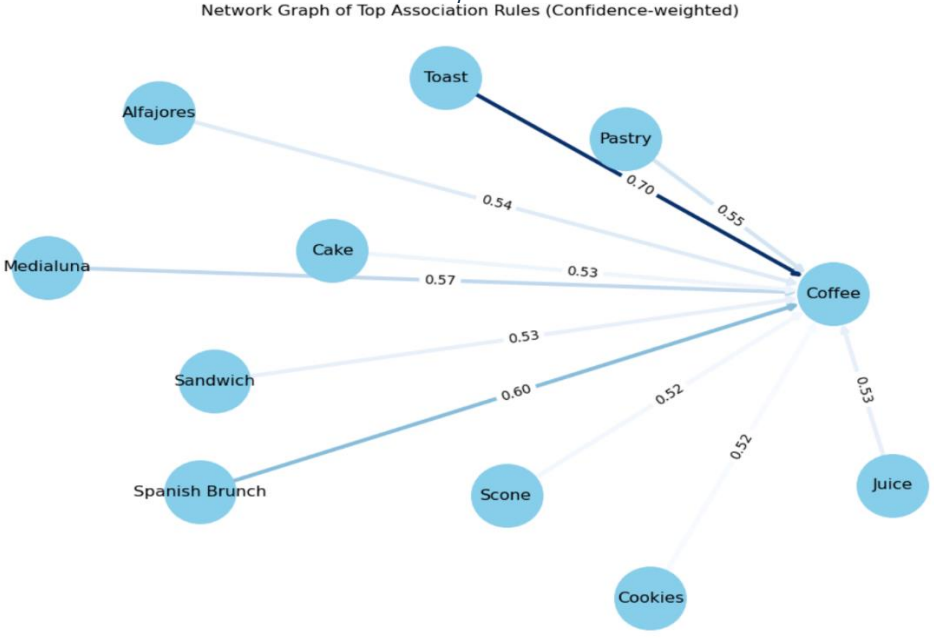

Figure 25: Network Graph of Top Association Rule

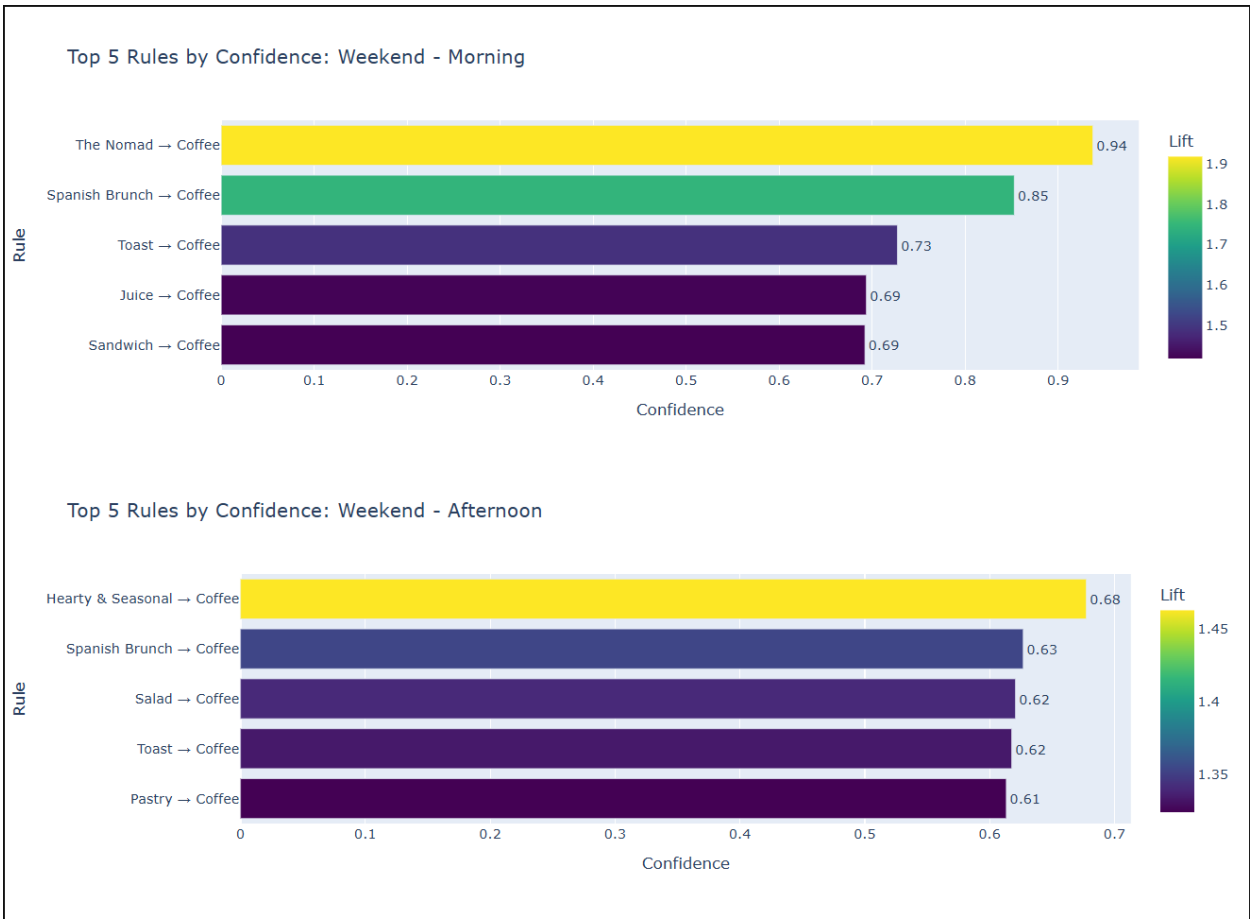

Figure 26: Top rules for Weekends and Part of the day

Figure 27: Top Rules by Weekday and parts of the day

From these graphs we can see that coffee, the most frequent itemset, is the most often ordered item. For a fairer analysis coffee is excluded from antecedents and consequents.

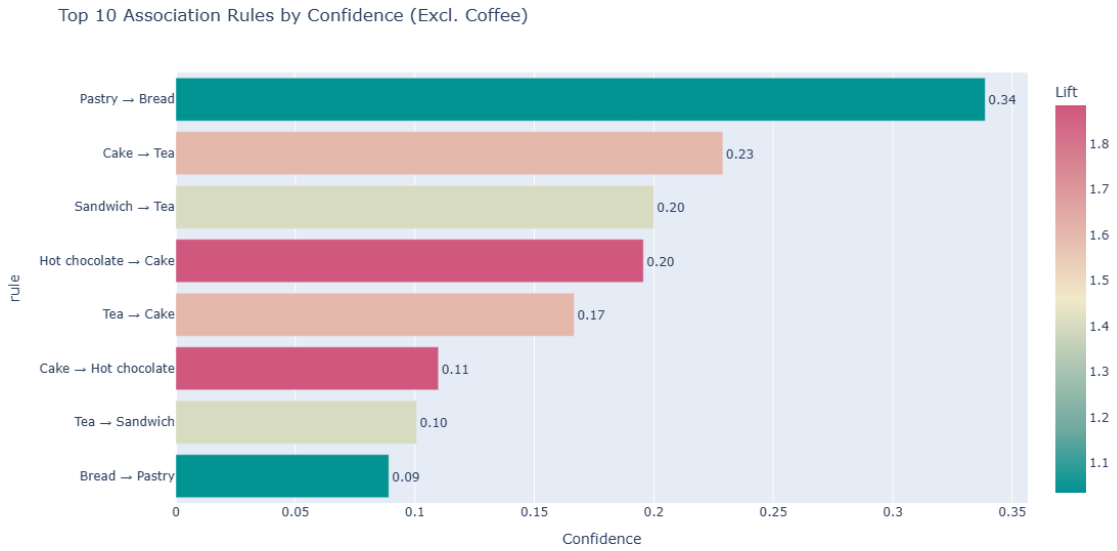

Figure 28: Top 10 Association rules excluding coffee

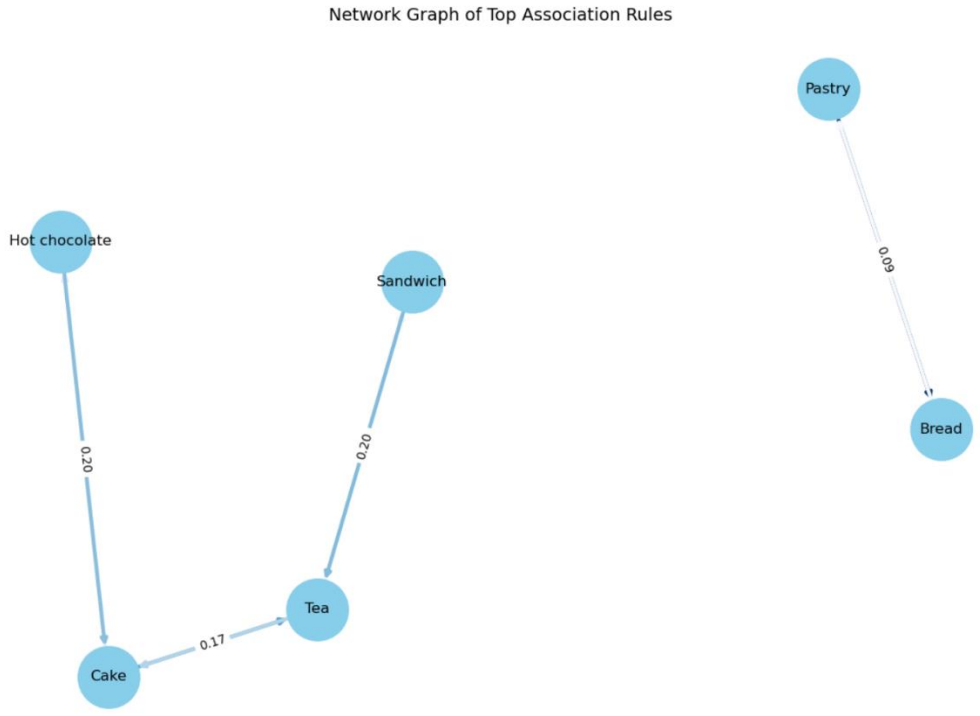

Figure 29: Network Graph of Top Association Rules excluding coffee

Figure 30: Top Rules for Weekend per part of the day excluding coffee

Figure 31: Top Rules for Weekend per part of the day excluding coffee

6. Implementation of other Algorithms

6.1 FP-Growth

The FP-Growth algorithm is one of the association rule mining techniques used in data mining. FP-Growth is efficient in discovering frequent item sets within a dataset. Unlike the Apriori algorithm, FP-Growth is less demanding. FP-Growth uses a tree structure to extract frequent itemsets, which makes it faster than Apriori. (Borgelt, 2005)

```
transactions = df.groupby('Transaction')['Item'].apply(list).tolist()

te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
```

```
from mlxtend.frequent_patterns import fpgrowth, association_rules

frequent_itemsets_fp = fpgrowth(df_encoded, min_support=0.01, use_colnames=True)
rules_fp = association_rules(frequent_itemsets_fp, metric="confidence", min_threshold=0.02)
```

Figure 32: FP-Growth Implementation

In this way FP-Growth Algorithm was implemented.

```
Top 10 FP-Growth rules by confidence:
         antecedents consequents   support  confidence      lift
69           (Toast)    (Coffee)  0.023666    0.704403  1.472431
72  (Spanish Brunch)    (Coffee)  0.010882    0.598837  1.251766
25       (Medialuna)    (Coffee)  0.035182    0.569231  1.189878
15          (Pastry)    (Coffee)  0.047544    0.552147  1.154168
60       (Alfajores)    (Coffee)  0.019651    0.540698  1.130235
32           (Juice)    (Coffee)  0.020602    0.534247  1.116750
59        (Sandwich)    (Coffee)  0.038246    0.532353  1.112792
37           (Cake)    (Coffee)  0.054728    0.526958  1.101515
70          (Scone)    (Coffee)  0.018067    0.522936  1.093107
9         (Cookies)    (Coffee)  0.028209    0.518447  1.083723
```

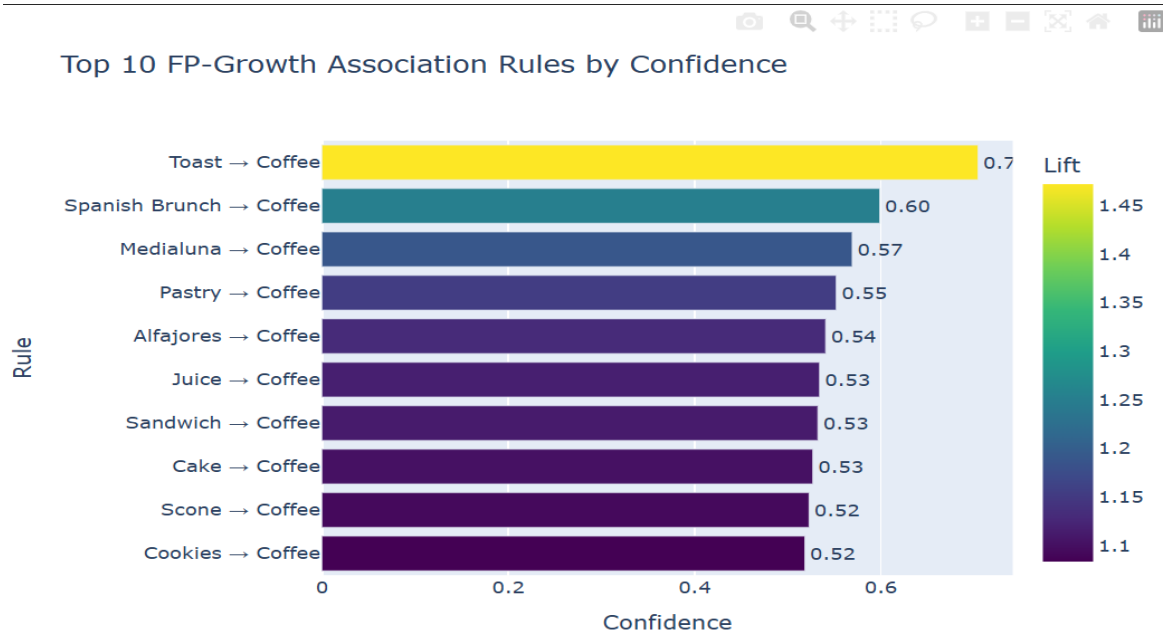

Figure 33: Top 10 FP-Growth Rules by Confidence

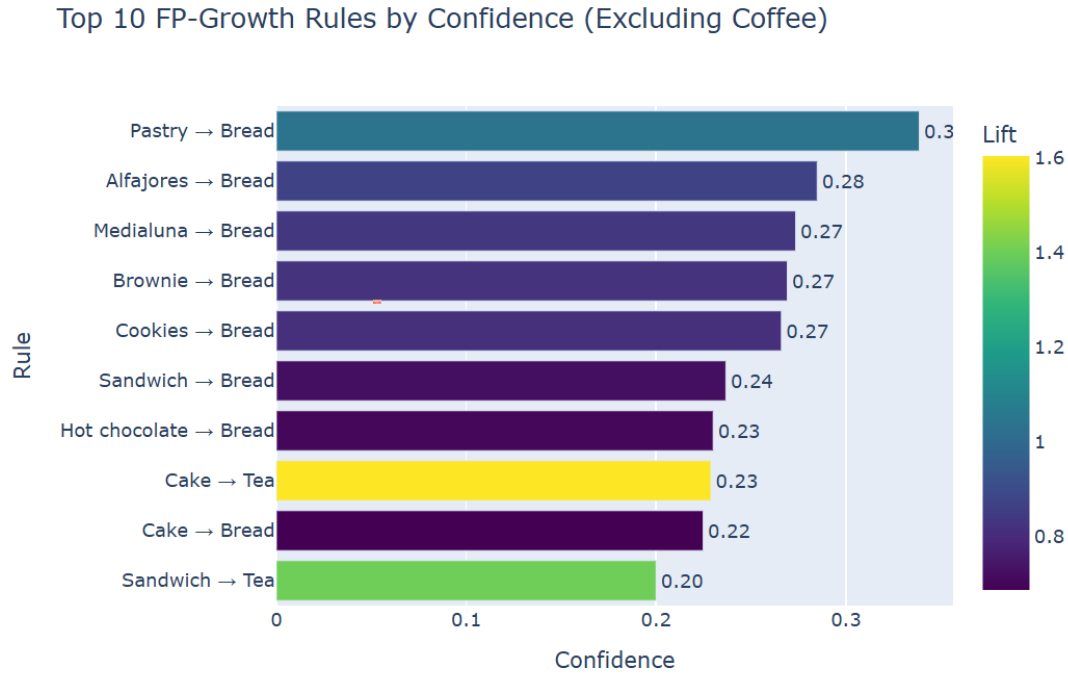

Figure 34: Top 10 FP-Growth Rules by Confidence excluding coffee

6.2 PrefixSpan

The PrefixSpan algorithm is an algorithm used for pattern recognition. This algorithm helps us discover frequently occurring patterns in a sequential dataset. PrefixSpan uses a pattern-growth approach. (Pei, et al., 2004)

```
from prefixspan import PrefixSpan

sequences = df.sort_values(['Transaction', 'date_time']).groupby('Transaction')['Item'].apply(list).tolist()

ps = PrefixSpan(sequences)

min_support = int(0.01 * len(sequences))

results = ps.frequent(min_support)
```

Figure 35: PrefixSpan Implementation

```
for seq in results[:10]:
    print(f"Support: {seq[0]}, Sequence: {seq[1]}")
```

```
Support: 3097, Sequence: ['Bread']
Support: 103, Sequence: ['Bread', 'Medialuna']
Support: 217, Sequence: ['Bread', 'Bread']
Support: 343, Sequence: ['Bread', 'Coffee']
Support: 179, Sequence: ['Bread', 'Pastry']
Support: 122, Sequence: ['Bread', 'Tea']
Support: 96, Sequence: ['Bread', 'Cookies']
Support: 134, Sequence: ['Bread', 'Cake']
Support: 275, Sequence: ['Scandinavian']
Support: 552, Sequence: ['Hot chocolate']
```

on

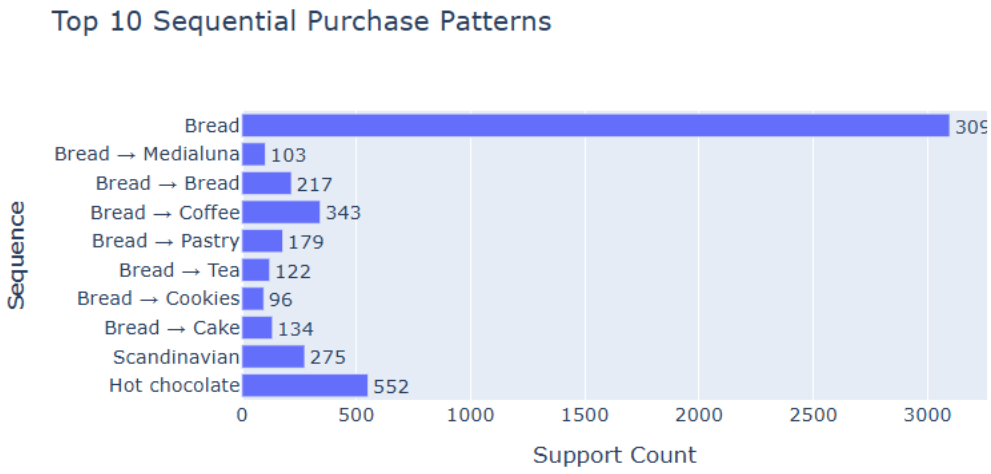

Figure 36: Top 10 Sequential Purchase Patterns

7. Evaluation And Analysis of Rules

7.1 Interpretation of Rules

Rules generated by the Apriori algorithm with parameters: 0.01 minimum support and minimum lift of 1 are:

[61]:

```
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	rep
0	(Alfajores)	(Coffee)	0.036344	0.478394	0.019651	0.540698	1.130235	
1	(Coffee)	(Alfajores)	0.478394	0.036344	0.019651	0.041078	1.130235	
2	(Bread)	(Pastry)	0.327205	0.086107	0.029160	0.089119	1.034977	
3	(Pastry)	(Bread)	0.086107	0.327205	0.029160	0.338650	1.034977	
4	(Brownie)	(Coffee)	0.040042	0.478394	0.019651	0.490765	1.025860	
5	(Coffee)	(Brownie)	0.478394	0.040042	0.019651	0.041078	1.025860	
6	(Coffee)	(Cake)	0.478394	0.103856	0.054728	0.114399	1.101515	
7	(Cake)	(Coffee)	0.103856	0.478394	0.054728	0.526958	1.101515	
8	(Cake)	(Hot chocolate)	0.103856	0.058320	0.011410	0.109868	1.883874	
9	(Hot chocolate)	(Cake)	0.058320	0.103856	0.011410	0.195652	1.883874	
10	(Tea)	(Cake)	0.142631	0.103856	0.023772	0.166667	1.604781	
11	(Cake)	(Tea)	0.103856	0.142631	0.023772	0.228891	1.604781	
12	(Coffee)	(Cookies)	0.478394	0.054411	0.028209	0.058966	1.083723	
13	(Cookies)	(Coffee)	0.054411	0.478394	0.028209	0.518447	1.083723	
14	(Coffee)	(Hot chocolate)	0.478394	0.058320	0.029583	0.061837	1.060311	
15	(Hot chocolate)	(Coffee)	0.058320	0.478394	0.029583	0.507246	1.060311	
16	(Coffee)	(Juice)	0.478394	0.038563	0.020602	0.043065	1.116750	
17	(Juice)	(Coffee)	0.038563	0.478394	0.020602	0.534247	1.116750	
18	(Medialuna)	(Coffee)	0.061807	0.478394	0.035182	0.569231	1.189878	
19	(Coffee)	(Medialuna)	0.478394	0.061807	0.035182	0.073542	1.189878	
20	(Muffin)	(Coffee)	0.038457	0.478394	0.018806	0.489011	1.022193	

21	(Coffee)	(Muffin)	0.478394	0.038457	0.018806	0.039311	1.022193
22	(Pastry)	(Coffee)	0.086107	0.478394	0.047544	0.552147	1.154168
23	(Coffee)	(Pastry)	0.478394	0.086107	0.047544	0.099382	1.154168
24	(Sandwich)	(Coffee)	0.071844	0.478394	0.038246	0.532353	1.112792
25	(Coffee)	(Sandwich)	0.478394	0.071844	0.038246	0.079947	1.112792
26	(Coffee)	(Scone)	0.478394	0.034548	0.018067	0.037765	1.093107
27	(Scone)	(Coffee)	0.034548	0.478394	0.018067	0.522936	1.093107
28	(Spanish Brunch)	(Coffee)	0.018172	0.478394	0.010882	0.598837	1.251766
29	(Coffee)	(Spanish Brunch)	0.478394	0.018172	0.010882	0.022747	1.251766
30	(Toast)	(Coffee)	0.033597	0.478394	0.023666	0.704403	1.472431
31	(Coffee)	(Toast)	0.478394	0.033597	0.023666	0.049470	1.472431
32	(Sandwich)	(Tea)	0.071844	0.142631	0.014369	0.200000	1.402222
33	(Tea)	(Sandwich)	0.142631	0.071844	0.014369	0.100741	1.402222
34	(Bread, Coffee)	(Cake)	0.090016	0.103856	0.010037	0.111502	1.073621
35	(Cake)	(Bread, Coffee)	0.103856	0.090016	0.010037	0.096643	1.073621
36	(Bread, Coffee)	(Pastry)	0.090016	0.086107	0.011199	0.124413	1.444872
37	(Pastry)	(Bread, Coffee)	0.086107	0.090016	0.011199	0.130061	1.444872
38	(Tea, Coffee)	(Cake)	0.049868	0.103856	0.010037	0.201271	1.937977
39	(Coffee, Cake)	(Tea)	0.054728	0.142631	0.010037	0.183398	1.285822
40	(Tea)	(Coffee, Cake)	0.142631	0.054728	0.010037	0.070370	1.285822
41	(Cake)	(Tea, Coffee)	0.103856	0.049868	0.010037	0.096643	1.937977

In our analysis, various findings were identified based on key metrics. The key metrics used were:

- **Support**: Support shows how frequently an itemset appear in the dataset

- **Confidence**: Confidence shows how often the consequent is bought when the antecedent is bought.

- **Lift**: Shows how likely a product is to be purchased when the antecedent is present compared to normal.

The rule with most confidence is (Toast) => (Coffee) has Support of 0.0237, Confidence of 70.44% and lift of 1.47 this means that 2.37% of the data contains both toast and coffee in the same transaction, when a customer buys toast there is 70% chance that they will also buy coffee and buying toast increases the likely hood of coffee being bought by 47%.

Rule 8: (Hot chocolate) => (Cake) has a support of 1.14%, confidence of 19.57% and lift of 1.88, this means that 1.14% of the data contains both hot chocolate and cake, when a customer buys Hot Chocolate there is a 19.57% chance that they will buy cake and the likely hood of buying a cake after buying hot chocolate is 88% higher than when being bought normally.

Rule 3: (Pastry) => (Bread) has a support of 0.029, confidence of 0.33, and lift of 1.03. This shows that 2.9% of transactions contain both pastry and bread. When a customer buys pastry, there is a 33% chance that they will buy bread. Despite having high confidence and support due to a lift of 1.03, there is only a 3% increase in likely hood of purchase of bread upon purchasing pastry.

Rule 38: (Tea, Coffee) => (Cake) has a support of 0.01, confidence of 0.20 and lift of 1.93, this shows than only 1% of transactions contain Tea, Coffee and Cake, when a customer buys both tea and coffee there is a 20% chance that they will also buy a cake and the likely hood od customers buying cake with tea and coffee is 93% higher than that of normal.

7.2 Interesting and actionable Insights

Some of the interesting and actionable insights are:

- **The Nomad => Coffee (Morning on weekends) Confidence = 0.94, lift = 1.91**

Customers tend to buy more nomads and coffee on weekends, and upon buying a nomad, customers are 91% more likely than normal to buy a coffee. These high numbers are only seen on weekends.

- **Hot chocolate => Cake; confidence = 0.20, lift = 1.88**

Customers who buy hot chocolate are also highly likely to buy a cake, as upon buying hot chocolate, customers are 88% more likely to buy a cake than usual.

- **Toast => Coffee; confidence = 0.70, lift = 1.47**

Customers who buy toast are highly likely to buy coffee, as the likelihood of these two items being bought together is 70%. Customers who eat toast are also 47% more likely to buy coffee.

- **Tea, Cake and Coffee**

Customers who buy cakes are highly likely to order coffee, tea, or both. Multiple rules with these items have been formed.

- **Spanish Brunch and Coffee**

Customers are more likely to have a Spanish brunch with Coffee on Weekends than on weekdays.

- **Bakery Timings**

The majority of transactions in this bakery dataset have happened either morning or the afternoon, which shows that customers like to visit the bakery at these times.

These insights, if applied into can significantly influence by formulation of various business, marketing, and promotional strategies.

7.3 Business Strategy Implications

Discovered associations, rules and the insights generated from them can help bakeries in the following areas:

1. **Product Bundling**

   Various combos, such as "**Buy Coffee, get 20% off on Toast**," can be introduced to increase average transactions and to have higher customer satisfaction. Another combination could be "**Buy a Hot chocolate, get 15% off on all cakes**".

2. **Weekly or Timely Offers**

   Some items sell well on weekdays, some sell well on weekends, some in the morning, some in the afternoon. Due to these reasons, the products that work together in a certain time frame can be bundled together and sold. For example, "**In the morning, get 15% off on Spanish Burch and Coffee**". Combinations popular in certain time frames can be offered at a certain discount at different time frames to boost their sales.

3. **Targeted Marketing and promotions**

   Using this data, the customers can recommend a popular item based on their purchase history. Frequent customers can also be offered personalized offers, which may boost customer satisfaction.

4. **Inventory Planning**

   With the help of these insights, the inventory can be better managed. If two items are a popular combination or if they have a high lift value, they can be stocked together in inventory. These insights also help bakeries to manage their inventory during peak demand.

With the help of these analyses, various important business strategies can be generated and applied to real-life scenarios.

8. Algorithm Comparison

Despite this project being heavily based on **the Apriori** algorithm, few other algorithms, such as FP-Growth and PrefixSpan, have been implemented to check how these association rule mining algorithms stack against each other.

Comparison based on number of Rules:

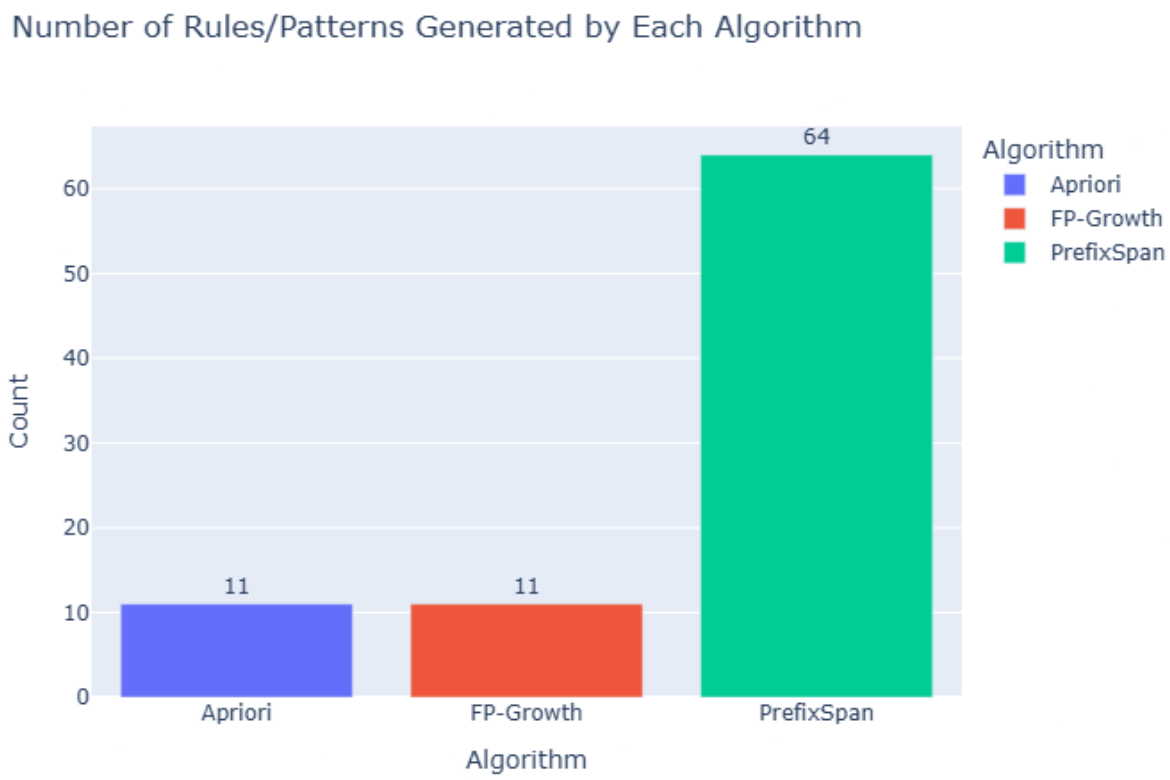

Figure 37: Number of rules generated by each algorithm

In this dataset, Apriori and FP-Growth have generated the same number of rules, but PrefixSpan has generated 64 rules/patterns. An outcome like this is expected, as prefixspan is a pattern recognition algorithm with a sole focus on pattern discovery. Apriori and FP-Growth use similar parameters, but prefix span uses a bit different parameters; this might also be the reason for a result like this.

Comparison Based on Execution Time:

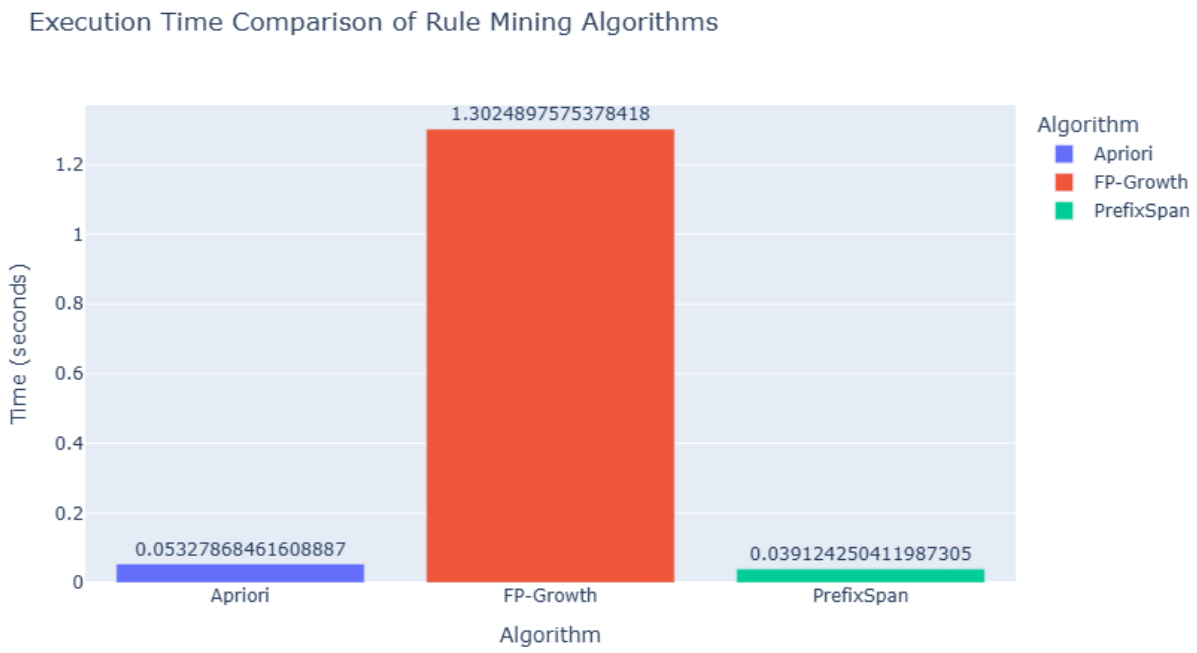

Figure 38: Comparison based on Execution Time

In this graph, we can see that Apriori and PrefixSpan executed quickly in this dataset, but FP-growth took a much longer time to execute. Apriori is known to be less efficient in large datasets, but in this case, its performance suggests that the dataset is of small to moderate size or it has a limited number of itemsets. FP-Growth is normally faster on larger datasets than Apriori, but in this particular dataset, it may be slower due to various dataset characteristics.

9. Conclusion

In this project, market basket analysis was conducted on a bakery dataset. Several Meaningful and actionable rules were discovered using Apriori, FP-Growth, and PrefixSpan algorithms. Although most of the analysis was done using Apriori, FP-growth and PrefixSpan were also used to compare these algorithms based on execution time and number of rules generated.

9.1 Key Insights generated

In this project, various meaningful and actionable insights were generated. The analysis showed that most of the transactions that take place in this bakery take place in the mornings and afternoons, and weekends have a higher rate of transactions per day. This indicates that various promotional and marketing strategies can be used to increase sales and customer satisfaction. During this analysis, multiple opportunities to bundle items together were also discovered. This analysis also shows that depending on the type of day and part of the day, the customer choices also vary. Optimal parameters for this dataset were also identified by doing hyperparameter tuning. Apriori, FP-Growth, and PrefixSpan were implemented and can be compared. In our dataset, Apriori and PrefixSpan executed quickly, but FP-Growth was a bit slower.

9.2 Challenges Faced During Implementation

Challenges faced in this project during implementation are:

- Handling Dominant Items

  In this dataset, coffee appeared in many rules and transactions, this sometimes-overshadowed association of other items. Additional filtering was required to extract insights without any bias.

- Execution Performance of FP-Growth

  Despite being widely regarded as a faster algorithm than apriori, FP-Growth took longer execution time when applied to our dataset.

9.3 Suggestions for Future Work

While various rules and insights were generated in this project using association rule mining algorithms, there remains a significant scope for expanding and applying more advanced data mining techniques. Some of the suggestions for future work are:

- Extend this analysis to better understand time-based patterns. Understanding time-based combinations can help design more precise marketing strategies and help manage inventory better.
- Combine market basket analysis and customer segmentation for a more personalized analysis. Insights generated from this can help increase the customer satisfaction rate.
- Integration of various techniques such as Neural Networks, LLMS, and other advanced data mining techniques to produce better and more detailed insights.
- Integrate cost and profit margins into the dataset, which can help increase business profitability

In summary, this project successfully demonstrated the insight generation capacity of Market Basket Analysis. By applying algorithms like Apriori, FP-Growth, and PrefixSpan, various meaningful rules, patterns, and insights were generated, which can be used for better marketing, smarter inventory management, and business strategies.

References

Borgelt, C. (2005). An implementation of the FP-growth algorithm. *OSDM '05: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, 1-5.

Hegland, M. (2007). THE APRIORI ALGORITHM – A TUTORIAL. *Mathematics and Computation in Imaging Science and Information Processing*, 209-262.

Manpreet Kaur, S. K. (2016). Market Basket Analysis: Identify the Changing Trends of Market Data Using Association Rule Mining. *Procedia Computer Science*, 78-85. doi:https://doi.org/10.1016/j.procs.2016.05.180.

Marselina, S., Jaman, J. H., & Kurniawan, D. E. (2023). Sales Analysis Using Apriori Algorithm in Data Mining Application on Food and Beverage (F&B) Transactions. *Journal of Applied Informatics and Computing*, 218-223.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., & Chen, Q. (2004). Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering* , 1424-1140.