Introduction Exercise 1 - Frequent Pattern Mining

Stein Vandenbroeke s0205627

March 21, 2025

1 Python notebook

The exact process of the data inspection, preparation, and mining association rules can be found step by step in the uploaded python notebook.

2 Data Inspection and Preparation

2.1 Understanding and Pruning the Data and handling Missing Values

2.1.1 Pruning the Data and handling Missing Values

First, we analyze the dataset coming from an online e-commerce store. To start, we just print the count of our raw data. This will give a general idea and some basic information. When we look at these numbers, it immediately stands out that there are less descriptions and customer ID's than the amount of rows in the CSV. This is probably caused by wrong data in the dataset. To solve this, I created a simple function that filters out lines based on the following rules:

- Remove all lines where any column (Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer ID or Country) is empty.
- Remove lines where purchases are made in the future.
- Remove all lines where prices are not doubles.
- Remove all lines if there is an inconsistency between StockCode and Description (same StockCode but different Description).

Because our dataset is large enough, we can assume that the removal of these lines will not impact our results. When we now print out the counts, we get the same amount for all columns 713915.

2.2 Data Categorization

The items InvoiceDate, Price and Quantity contain continuous or very precise data to enhance the frequent pattern mining process. We will split them up in meaningful categories:

2.2.1 InvoiceDate

We split the invoice into the following time slots: [(00:00-5:59), (06:00-11:59), (12:00-15:59), (16:00-18:59), (19:00-24:00)]. When we look at this data, we can see that there are no purchases after 19h on Sunday and almost no purchases on Saturdays. And that most purchases are done in the afternoon.

2.2.2 Price

We can take different approaches to categorize the prices. We can categorize based on every sold item or based on every product available in the store. How we do this depends mostly on what we want to achieve. I will assume the store wants to continue their current practices and thus sell lots of cheap items and not small amounts of expensive items. We will split up the price into 4 categories: very cheap (0-1.25), cheap (1.25-1.95), normal (1.95-3.75), and expensive (3.75-inf). Because cheap products are sold the most (this can be seen in plot 1 of figure 1).

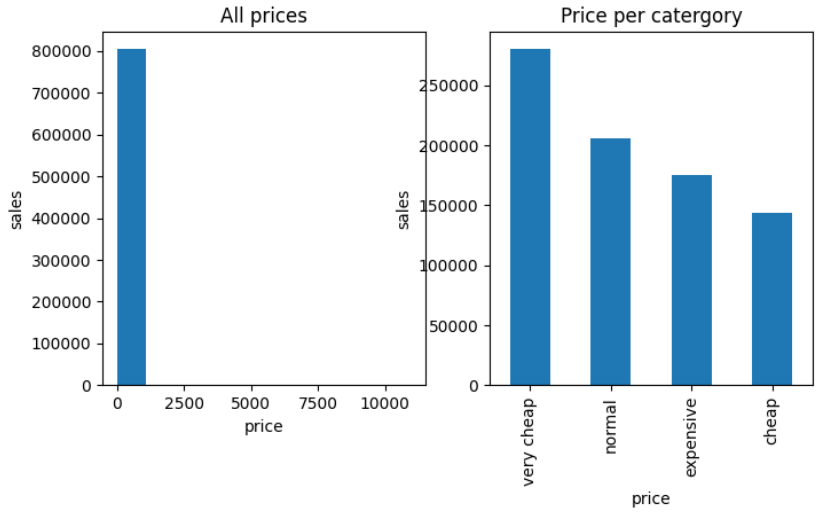

Figure 1: First plot: continuous price, Second plot: over categories

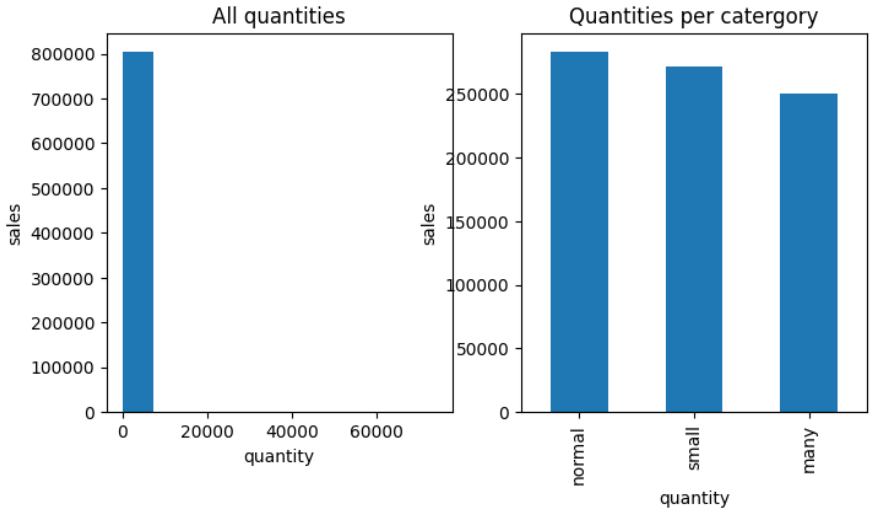

Figure 2: First plot: continuous quantities, Second plot: over categories

2.2.3 Quantity

For Quantity we will use the same approach to split up in categories. So we will split up into: small (1 item), normal (2 items - 10 items), and many (10 items - inf). In figure 2 we can see how we divided our quantities over the number of product purchases.

3 Mining Association Rules

3.1 Exploring the Dataset

3.1.1 Features

There are different features we can combine for our association rules, because running apriori for all features at the same time is computationaly intsensive and useless because for example the description and Stock code should always be linked. I decided to create the following baskets;

- All stock codes with the same invoice (this allows us to predict what a user also wants when he adds something to his shopping card).

- Baskets on the categorized quantity, price and user ID (this allows us to find useful insides about how certain users shop).
- Baskets on descriptions and categorized quantity (To find rules about how many of a certain product users buy).
- Baskets on descriptions and categorized time (To find rules about when certain products are bought).

Many more features can be explored. For example, based on the month of sale or what kind of Countries certain products are bought in. But the most obvious one is the one where we predict what a user wants to buy based on what he already has in his shopping card.

3.1.2 Minimum support

Minimum support represents how frequently an item should appear relative to the dataset. And thus the lower we make this value the more matches we will find. And thus a higher value will result in items that occur often in the dataset. In our code example (100.000 items) if we use a min_support of 0.025 (2,5%) than that item should be at least 2500 times occur.

3.1.3 Minimum confidence

Minimum confidence is the likelihood that when X is bought Y is also bought. So when we make this value higher this will result in fewer association rules.

3.1.4 Lift

The lift is automatically calculated by the apyori library. this value refers how much more likely it is that the items appear tougher compared to appear individually. So we want a value higher than 1 because otherwise it is just conisidence.

3.2 Identifying Market Insights

3.2.1 Rules based between StockCode/Descriptions

We will run the apriori algorithm on baskets filled with the Descriptions of every invoice. This is useful because when a person is shopping we can use these rules to recommend new products based on what he already has placed in the basket. To find the rules, we will set the min_confidence to 0.7 and the support to 0.025, meaning it must appear at least 104 times in a transaction and sort it on the confidence values. Note that this is still a high value and can be lowered for more rules, but this will take longer to compute. When we take the top 3 confidence values this results in the following 3 top rules:

Rule 1. **Items**: 'wooden star christmas scandinavian', 'wooden tree christmas scandinavian', 'wooden heart christmas scandinavian'
Confidence: 90%
Support: 3% ($\pm$100 occurrences over the transactions)
Rule: 'wooden tree christmas scandinavian', 'wooden heart christmas scandinavian' $\rightarrow$ 'wooden star christmas scandinavian'
Explanation: This rule has a high confidence of 90% this indicates that if the 2 products are bought in an invoice the wooden star will also be bought. The support of 3% indicates that this happened in around 125 transactions (0.03 * 4143). When analyzing the product descriptions this is an expected rule.

Rule 2. **Items**: 'wooden star christmas scandinavian', 'wooden tree christmas scandinavian'
Confidence: 85%
Support: 3% ($\pm$100 occurrences over the transactions)
Rule: 'wooden star christmas scandinavian' $\rightarrow$ 'wooden tree christmas scandinavian'

Rule 3. **Items**: 'set of 3 wooden stocking decoration', 'set of 3 wooden tree decorations'
Confidence: 78%
Support: 3% (±100 occurrences over the transactions)
Rule: 'set of 3 wooden tree decorations' → 'set of 3 wooden stocking decoration'

3.2.2 Purchase Timing

Now we will combine the Descriptions with the time category to find rules about when certain products are bought. This will be done using tuples containing the product description and the time category.

Rule 1. **Items**: 'hand warmer babushka design', 'time:12:00-15:59'
Confidence: 80%
Support: 0,073% (±50 occurrences over the transactions)
Rule: 'hand warmer babushka design' → 'time:12:00-15:59'

Rule 2. **Items**: 'christmas toilet roll', 'time:12:00-15:59'
Confidence: 79%
Support: 0,073% (±100 occurrences over the transactions)
Rule: 'christmas toilet roll' → 'time:12:00-15:59'

Rule 3. **Items**: 'glitter heart decoration', 'time:12:00-15:59'
Confidence: 75%
Support: 0,073% (±100 occurrences over the transactions)
Rule: 'glitter heart decoration' → 'time:12:00-15:59'

If we compare these rules to the rules based on StockCode/Descriptions we see a lower confidence this can be explained by the fact that in an online store the sales of items are only limited related to the product. When we look at the descriptions there is also not a clear reason why this products would be bought between 12:00 and 15:59.