**PERFORMING ASSOCIATION ANALYSIS AND GENERATING INSIGHTS**

In the first step of preparing our data, we carefully sorted through the transactions, isolating those made in 'Germany.' This step was crucial for tailoring our analysis to a specific geographic market. Next, we organized the data systematically by grouping transactions based on their unique 'BillNo' and 'Description.' For each item, we added up the quantities purchased, creating a clear and organized basket format.

*#*Viewing transaction basket

mybasket = (dataset[dataset['Country'] == "Germany"]).groupby(['BillNo', 'Description'])['Quantity'].sum().unstack().reset\_index().fillna(0).set\_index('BillNo')

The my\_encode\_units function further transforms the data into binary format, which is necessary for the Apriori algorithm. In this binary representation, items were denoted as '1' if they were present in a transaction and '0' if they were absent. This binary representation is a prerequisite for the Apriori algorithm, allowing it to discern itemset associations and patterns with precision.

*#Converting all +ve values to 1 & everything else to 0*

def my\_encode\_units(x):

if x <=0:

return 0

if x >=1:

return 1

mybasket\_sets=mybasket.applymap(my\_encode\_units);

mybasket\_sets.drop('POSTAGE', inplace=True, axis=1)

**Apriori algorithm**

With the preprocessed basket sets in place, the Apriori algorithm is employed. A minimum support threshold of 0.07 was set, a carefully chosen value to ensure the discovery of frequent itemsets while filtering out noise and irrelevant data. This step proved indispensable as it unearthed the frequent itemsets — a collection of items frequently co-occurring in transactions. These frequent itemsets serve as the foundation for the subsequent generation of meaningful association rules. These item sets represent the core patterns in customer purchases.

*#Frequent itemsets*

my\_frequent\_items = apriori(my\_basket\_sets, min\_support=0.07, use\_colnames=True)

**Association Rules Generation:**

Association rules are pivotal in market basket analysis, revealing valuable insights into customer behavior. After discovering frequent itemsets, these rules were meticulously generated. Using the association\_rules function, each rule was systematically examined, exploring the relationships between different items within the dataset.

Evaluation Based on 'Lift' Metric:

The evaluation of these association rules hinged significantly on the 'lift' metric. 'Lift' measures how much more likely two items are to be bought together compared to if they were bought independently. Here's a breakdown of how this evaluation occurred:

1. Understanding 'Lift':

• Lift > 1: Indicates that items in the rule are more likely to be bought together. A lift score of 1 implies items are independent of each other.

• Lift < 1: Implies items are less likely to be bought together than if chosen randomly. It indicates a negative association between items.

2. Setting the Threshold:

A threshold of 1 was strategically chosen. This means that only rules demonstrating a significant increase in the likelihood of items being bought together were considered. By setting this threshold, the analysis focused on substantial and meaningful associations, filtering out weaker connections that might not offer actionable insights.

*#Frequent itemsets*

my\_frequent\_items = apriori(my\_basket\_sets, min\_support=0.07, use\_colnames=True)

*#generating rules*

from mlxtend.frequent\_patterns.association\_rules import association\_rules

my\_rules = association\_rules(my\_frequent\_items, metric='lift', min\_threshold=1)

*#viewing top 100 rules*

my\_rules.head(100)

**MACHINE LEARNING MODEL : LOGISTIC REGRESSION**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

# Splitting data into features (X) and target variable (Y)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(features, target, test\_size=0.25, random\_state=0)

# Initializing and training logistic regression model

model = LogisticRegression()

model.fit(X\_train, Y\_train)

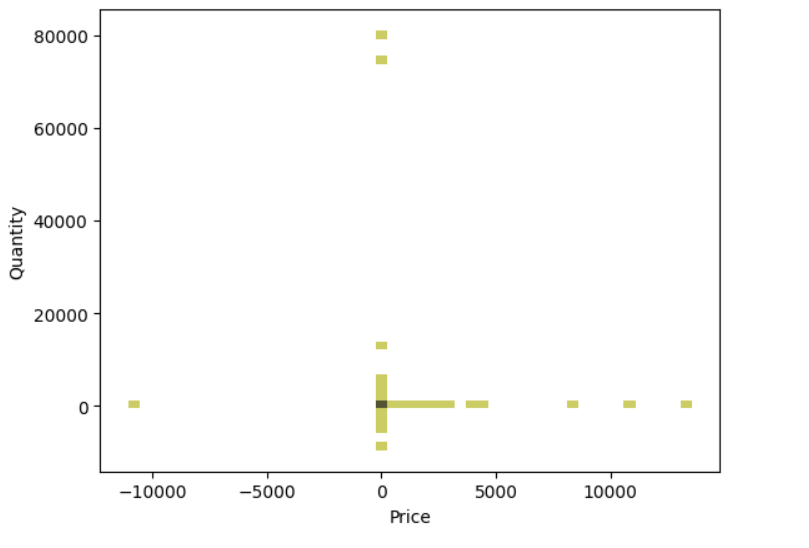
# Making predictions

predictions = model.predict(X\_test)

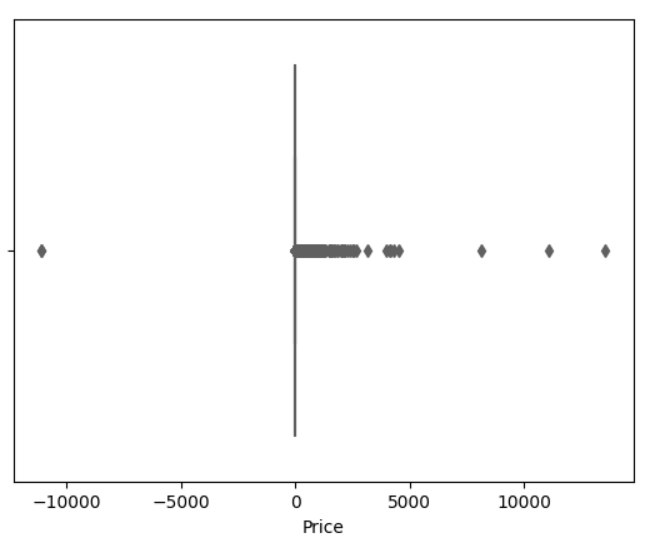
The dataset is split into features (X) and the target variable (Y) using the train\_test\_split function, allocating 75% of the data for training the model and 25% for testing. A logistic regression model is initialized and trained with the training data (X\_train and Y\_train) using the fit method. This training process involves the model learning the underlying patterns in the data. Subsequently, the trained model is used to make predictions on the testing data (X\_test), and the predicted values are stored in the predictions variable. These predictions represent the model's estimation of the target variable based on the input features.

**DATASET VISUALIZATION**

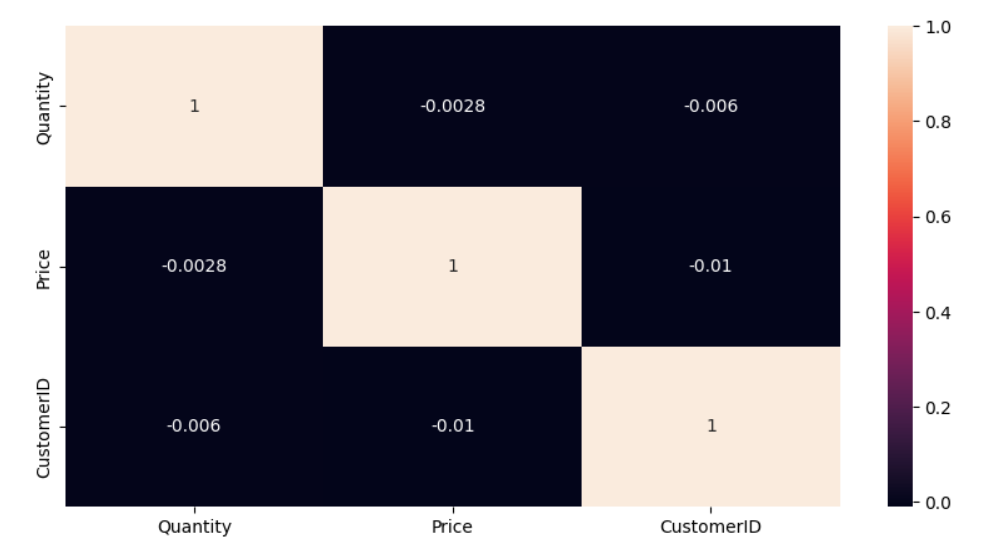
**2D Histogram:** This visualization shows the relationship between 'Price' and 'Quantity' using a 2D histogram with 50 bins. The color yellow is used to represent the data points, offering insights into the distribution and density of the data points for these two variables.



**Box Plot:** The box plot provides a summary of the distribution of the 'Price' variable. It displays key statistics such as median, quartiles, and potential outliers. This plot offers a clear understanding of the central tendency and spread of the 'Price' data, aiding in the identification of any unusual data points.



**Pair Plot and Correlation Heatmap:** The pair plot illustrates pairwise relationships between all variables in the dataset, offering a comprehensive view of how different variables correlate with one another. Additionally, a heatmap is generated to visually represent the correlation matrix of the dataset, with numerical annotations indicating the strength and direction of correlations. These visualizations collectively provide valuable insights into the relationships and correlations present within the dataset, aiding in exploratory data analysis.

****

**SCATTER PLOTS**

The array of scatter plots is also used to understand the intricate relationships within the transactional data. The 'Quantity vs. Price' scatter plot illuminates price sensitivity concerning purchase quantities, guiding pricing strategies and inventory management. Examining 'Date vs. Price' unveils temporal trends and seasonality and is important for dynamic pricing. Comparing 'Quantity vs. Country' provides geographic consumption patterns to make inventory decisions based on regional preferences. Furthermore, revisiting 'Quantity vs. Price' from different perspectives is helpful in optimizing product offerings and marketing campaigns.