|  |
| --- |
| **IBM - Artificial Intelligence**  PRATHISRAJ  **422221104302**  **Team 10** |

**Market Basket Insights**

**What is Market Basket Analysis?**

Market basket analysis in data mining is a kind of data analytics that pinpoints the goods or things that people usually buy together. In order to comprehend consumer behavior and pinpoint product combinations that are well-liked by clients, this study is typically carried out in the retail sector. The outcomes of a market basket study may be utilized for a variety of tasks, including developing specialized marketing campaigns, enhancing product placement in stores, and enhancing inventory control.

Market basket analysis is a procedure that includes gathering information about client transactions and then applying association rule learning algorithms to find patterns in the data. The findings of these algorithms, which search for pairings of goods that are commonly bought together, are presented as “association rules.”

**Examples of Market Basket Analysis**

A grocery store evaluating customer purchase data to discover which goods are usually purchased together is a real-world example of market basket analysis. Customers who buy bread may also buy peanut butter, jelly, and bananas, according to the study. With this knowledge, the retailer may make modifications to improve sales of these products, such as positioning them near each other on the shelf or providing discounts when consumers purchase all four items together.

Another example might be an online store examining customer purchase data to see which goods are often purchased together. The study may indicate that customers who buy laptops also buy mouse pads, extra hard drives, and extended warranties. With this information, the online merchant might build targeted product bundles or upsell opportunities, such as giving a package deal for a laptop, mouse pad, external hard drive, and extended warranty.

A healthcare organization uses market basket analysis to determine that patients who are diagnosed with diabetes frequently also have high blood pressure and high cholesterol. Based on this information, the organization creates a care plan that addresses all three conditions, which leads to improved patient outcomes and reduced healthcare costs.

**Predictive Market Basket Analysis**

Predictive market basket analysis is a type of data mining technique that uses historical data on customer purchases to make predictions about future customer behavior. The goal of predictive market basket analysis is to identify items that are likely to be purchased together and use this information to inform business decisions such as product placement, marketing strategies, and inventory management.

This type of analysis often involves using statistical and machine learning models to analyze the relationships between items, such as association rules and sequence analysis. The model is trained on historical data and can be used to make predictions about future purchases, such as suggesting items that a customer is likely to buy in the future or identifying products that are likely to be out of stock.

Predictive market basket analysis is a valuable tool for retailers and other businesses that want to gain a deeper understanding of their customers and improve their operations.

Applications of Market Basket Analysis

Market basket analysis has several uses in various sectors, the most popular of which are:

**The Retail Industry**

Market basket research can assist retailers to find goods that are commonly purchased together, which can help them make product placement, marketing, and price decisions. This can result in greater revenue and better client satisfaction.

**E-Commerce**

Market basket analysis can be used by online merchants to evaluate client purchase data and discover which goods are often purchased together. This data may be utilized to develop targeted product bundles and upsell chances.

**Healthcare**

Market basket analysis can be used by healthcare organizations to evaluate patient data and find co-occurring illnesses or treatments. This data may be utilized to enhance patient outcomes while also lowering healthcare expenses.

**Financial Services and Banking**

Market basket analysis can be used by banks and financial organizations to evaluate client data and uncover trends in their purchasing habits. This data may be utilized to create customized marketing initiatives and boost consumer loyalty.

**Telecommunications**

Telecommunications firms can use market basket analysis to study consumer data and detect trends in their service consumption. This data may be utilized to enhance the customer experience and boost revenue.

**Apriori** **Groceries** **data**

**Python · Groceries dataset for Market Basket Analysis(MBA)**

**# This Python 3 environment comes with many helpful analytics libraries installed**

**# It is defined by the kaggle/python Docker image: [https://github.com/kaggle/docker-python](https://github.com/kaggle/docker-python" \t "https://word-edit.officeapps.live.com/we/_blank)**

**# For example, here’s several helpful packages to load**

**Import numpy as np # linear algebra**

**Import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)**

**# Input data files are available in the read-only “../input/” directory**

**# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory**

**Import os**

**For dirname, \_, filenames in os.walk(‘/kaggle/input’):**

**For filename in filenames:**

**Print(os.path.join(dirname, filename))**

**# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using “Save & Run All”**

**# You can also write temporary files to /kaggle/temp/, but they won’t be saved outside of the current session**

**/kaggle/input/groceries-dataset-for-market-basket-analysismba/Groceries data.csv**

**/kaggle/input/groceries-dataset-for-market-basket-analysismba/basket.csv**

**Import pandas as pd**

**Import numpy as np**

**From mlxtend.preprocessing import TransactionEncoder**

**From mlxtend.frequent\_patterns import apriori**

**From mlxtend.frequent\_patterns import association\_rules**

**/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5**

**Warnings.warn(f”A NumPy version >={np\_minversion} and <{np\_maxversion}”**

**Dataset = pd.read\_csv(r’/kaggle/input/groceries-dataset-for-market-basket-analysismba/basket.csv’)**

**Dataset**

**0** **1** **2** **3** **4** **5** **6** **7** **8** **9** **10**

**0** **whole milk** **pastry** **salty snack** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**1** **sausage** **whole milk** **semi-finished bread** **yogurt** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**2** **soda** **pickled vegetables** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**3** **canned beer** **misc. beverages** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**4** **sausage** **hygiene articles** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…**

**14958** **butter milk** **whipped/sour cream** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**14959** **bottled water** **herbs** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**14960** **fruit/vegetable juice** **onions** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**14961** **bottled beer** **other vegetables** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**14962** **soda** **root vegetables** **semi-finished bread** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN** **NaN**

**14963 rows × 11 columns**

**Dataset.shape**

**(14963, 11)**

**Dataset.fillna(‘1’, inplace = True)**

**Transactions=[]**

**For I in range (14963):**

**Transaction = []**

**For j in range(11):**

**If dataset.iloc[I,j] != ‘1’:**

**Transaction.append(dataset.iloc[I,j])**

**Transactions.append(transaction)**

**Transactions[2]**

**[‘soda’, ‘pickled vegetables’]**

**Te = TransactionEncoder()**

**Te\_bin = te.fit\_transform(Transactions)**

**Transactions = pd.DataFrame(te\_bin, columns = te.columns\_)**

**Transactions**

**Instant food products** **UHT-milk** **abrasive cleaner** **artif. Sweetener** **baby cosmetics** **bags** **baking powder** **bathroom cleaner** **beef** **berries** **…** **turkey** **vinegar** **waffles** **whipped/sour cream** **whisky** **white bread** **white wine** **whole milk** **yogurt** **zwieback**

**0** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **True** **False** **False**

**1** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **True** **True** **False**

**2** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**3** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**4** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…**

**14958** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **True** **False** **False** **False** **False** **False** **False**

**14959** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**14960** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**14961** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**14962** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False** **…** **False** **False** **False** **False** **False** **False** **False** **False** **False** **False**

**14963 rows × 167 columns**

**Def encode(x):**

**If x <= 0:**

**Return 0**

**If x >= 1:**

**Return 1**

**Transactions = Transactions.applymap(encode)**

**Transactions**

**Instant food products** **UHT-milk** **abrasive cleaner** **artif. Sweetener** **baby cosmetics** **bags** **baking powder** **bathroom cleaner** **beef** **berries** **…** **turkey** **vinegar** **waffles** **whipped/sour cream** **whisky** **white bread** **white wine** **whole milk** **yogurt** **zwieback**

**0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **1** **0** **0**

**1** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **1** **1** **0**

**2** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**3** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**4** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…** **…**

**14958** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **1** **0** **0** **0** **0** **0** **0**

**14959** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**14960** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**14961** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**14962** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0** **…** **0** **0** **0** **0** **0** **0** **0** **0** **0** **0**

**14963 rows × 167 columns**

**Frequent\_items = apriori(Transactions, min\_support = 0.002,use\_colnames = True)**

**Frequent\_items.head()**

**/opt/conda/lib/python3.10/site-packages/mlxtend/frequent\_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type**

**Warnings.warn(**

**Support** **itemsets**

**0** **0.004010** **(Instant food products)**

**1** **0.021386** **(UHT-milk)**

**2** **0.008087** **(baking powder)**

**3** **0.033950** **(beef)**

**4** **0.021787** **(berries)**

**Rules = association\_rules(frequent\_items, metric=’lift’,min\_threshold =1)**

**Rules.head()**

**Antecedents** **consequents** **antecedent support** **consequent support** **support** **confidence** **lift** **leverage** **conviction** **zhangs\_metric**

**0** **(berries)** **(other vegetables)** **0.021787** **0.122101** **0.002673** **0.122699** **1.004899** **0.000013** **1.000682** **0.004984**

**1** **(other vegetables)** **(berries)** **0.122101** **0.021787** **0.002673** **0.021894** **1.004899** **0.000013** **1.000109** **0.005553**

**2** **(sausage)** **(bottled beer)** **0.060349** **0.045312** **0.003342** **0.055371** **1.222000** **0.000607** **1.010649** **0.193337**

**3** **(bottled beer)** **(sausage)** **0.045312** **0.060349** **0.003342** **0.073746** **1.222000** **0.000607** **1.014464** **0.190292**

**4** **(brown bread)** **(canned beer)** **0.037626** **0.046916** **0.002406** **0.063943** **1.362937** **0.000641** **1.018191** **0.276701**

**Rules = rules.sort\_values(by=’lift’, ascending = False)**

**Rules**

**Antecedents** **consequents** **antecedent support** **consequent support** **support** **confidence** **lift** **leverage** **conviction** **zhangs\_metric**

**13** **(curd)** **(sausage)** **0.033683** **0.060349** **0.002941** **0.087302** **1.446615** **9.078510e-04** **1.029531** **0.319493**

**12** **(sausage)** **(curd)** **0.060349** **0.033683** **0.002941** **0.048726** **1.446615** **9.078510e-04** **1.015814** **0.328559**

**4** **(brown bread)** **(canned beer)** **0.037626** **0.046916** **0.002406** **0.063943** **1.362937** **6.406768e-04** **1.018191** **0.276701**

**5** **(canned beer)** **(brown bread)** **0.046916** **0.037626** **0.002406** **0.051282** **1.362937** **6.406768e-04** **1.014394** **0.279398**

**21** **(sausage)** **(frozen vegetables)** **0.060349** **0.028002** **0.002072** **0.034330** **1.225966** **3.818638e-04** **1.006553** **0.196155**

**20** **(frozen vegetables)** **(sausage)** **0.028002** **0.060349** **0.002072** **0.073986** **1.225966** **3.818638e-04** **1.014726** **0.189627**

**2** **(sausage)** **(bottled beer)** **0.060349** **0.045312** **0.003342** **0.055371** **1.222000** **6.070623e-04** **1.010649** **0.193337**

**3** **(bottled beer)** **(sausage)** **0.045312** **0.060349** **0.003342** **0.073746** **1.222000** **6.070623e-04** **1.014464** **0.190292**

**15** **(frankfurter)** **(other vegetables)** **0.037760** **0.122101** **0.005146** **0.136283** **1.116150** **5.355097e-04** **1.016420** **0.108146**

**14** **(other vegetables)** **(frankfurter)** **0.122101** **0.037760** **0.005146** **0.042146** **1.116150** **5.355097e-04** **1.004579** **0.118536**

**35** **(sausage)** **(yogurt)** **0.060349** **0.085879** **0.005748** **0.095238** **1.108986** **5.648409e-04** **1.010345** **0.104587**

**34** **(yogurt)** **(sausage)** **0.085879** **0.060349** **0.005748** **0.066926** **1.108986** **5.648409e-04** **1.007049** **0.107508**

**19** **(root vegetables)** **(frozen vegetables)** **0.069572** **0.028002** **0.002139** **0.030740** **1.097751** **1.904361e-04** **1.002824** **0.095705**

**18** **(frozen vegetables)** **(root vegetables)** **0.028002** **0.069572** **0.002139** **0.076372** **1.097751** **1.904361e-04** **1.007363** **0.091612**

**8** **(rolls/buns)** **(chocolate)** **0.110005** **0.023592** **0.002807** **0.025516** **1.081592** **2.117455e-04** **1.001975** **0.084761**

**9** **(chocolate)** **(rolls/buns)** **0.023592** **0.110005** **0.002807** **0.118980** **1.081592** **2.117455e-04** **1.010188** **0.077260**

**16** **(frozen meals)** **(other vegetables)** **0.016775** **0.122101** **0.002139** **0.127490** **1.044134** **9.039652e-05** **1.006176** **0.042990**

**17** **(other vegetables)** **(frozen meals)** **0.122101** **0.016775** **0.002139** **0.017515** **1.044134** **9.039652e-05** **1.000754** **0.048148**

**26** **(meat)** **(other vegetables)** **0.016842** **0.122101** **0.002139** **0.126984** **1.039991** **8.223631e-05** **1.005593** **0.039112**

**27** **(other vegetables)** **(meat)** **0.122101** **0.016842** **0.002139** **0.017515** **1.039991** **8.223631e-05** **1.000686** **0.043801**

**6** **(pastry)** **(brown bread)** **0.051728** **0.037626** **0.002005** **0.038760** **1.030127** **5.863558e-05** **1.001179** **0.030841**

**7** **(brown bread)** **(pastry)** **0.037626** **0.051728** **0.002005** **0.053286** **1.030127** **5.863558e-05** **1.001646** **0.030389**

**28** **(pastry)** **(sausage)** **0.051728** **0.060349** **0.003208** **0.062016** **1.027617** **8.621145e-05** **1.001777** **0.028341**

**29** **(sausage)** **(pastry)** **0.060349** **0.051728** **0.003208** **0.053156** **1.027617** **8.621145e-05** **1.001509** **0.028601**

**33** **(sausage)** **(soda)** **0.060349** **0.097106** **0.005948** **0.098560** **1.014975** **8.775684e-05** **1.001613** **0.015702**

**32** **(soda)** **(sausage)** **0.097106** **0.060349** **0.005948** **0.061253** **1.014975** **8.775684e-05** **1.000963** **0.016341**

**25** **(ham)** **(whole milk)** **0.017109** **0.157923** **0.002740** **0.160156** **1.014142** **3.821049e-05** **1.002659** **0.014188**

**24** **(whole milk)** **(ham)** **0.157923** **0.017109** **0.002740** **0.017351** **1.014142** **3.821049e-05** **1.000246** **0.016560**

**11** **(citrus fruit)** **(yogurt)** **0.053131** **0.085879** **0.004611** **0.086792** **1.010642** **4.855926e-05** **1.001001** **0.011121**

**10** **(yogurt)** **(citrus fruit)** **0.085879** **0.053131** **0.004611** **0.053696** **1.010642** **4.855926e-05** **1.000598** **0.011520**

**31** **(root vegetables)** **(shopping bags)** **0.069572** **0.047584** **0.003342** **0.048031** **1.009388** **3.107757e-05** **1.000469** **0.009996**

**30** **(shopping bags)** **(root vegetables)** **0.047584** **0.069572** **0.003342** **0.070225** **1.009388** **3.107757e-05** **1.000702** **0.009765**

**1** **(other vegetables)** **(berries)** **0.122101** **0.021787** **0.002673** **0.021894** **1.004899** **1.303311e-05** **1.000109** **0.005553**

**0** **(berries)** **(other vegetables)** **0.021787** **0.122101** **0.002673** **0.122699** **1.004899** **1.303311e-05** **1.000682** **0.004984**

**23** **(fruit/vegetable juice)** **(rolls/buns)** **0.034017** **0.110005** **0.003743** **0.110020** **1.000136** **5.091755e-07** **1.000017** **0.000141**

**22** **(rolls/buns)** **(fruit/vegetable juice)** **0.110005** **0.034017** **0.003743** **0.034022** **1.000136** **5.091755e-07** **1.000005** **0.000153**