**Business Objective:** The business problem, following the revenue metrics, is to know which strategies can potentially lead to an increase in sales. This problem is further breakdown into the following specific objectives: Keystone Products Product placement strategy 3. Strategies to sell high margin products 4. Sale on products that can help to sell other products. Market basket analysis is done applying Apriori algorithm which is based on Association learning. This helped to determine the answers and strategies for the above stated objectives, and are: 1. Keystone products: 'mineral water', 'eggs', 'spaghetti', 'french fries', 'chocolate', 'green tea', 'milk', 'ground beef', 'frozen vegetables', 'pancakes', 'burgers', 'cake', 'cookies', 'escalope', 'low fat yogurt', 'shrimp', 'tomatoes', 'olive oil', 'frozen smoothie', 'turkey'. Product placement strategy: 'spaghetti' is sold more in a combined cart containing 'ground beef', 'tomatoes' or 'tomato sauce', 'olive oil' or 'cooking oil'. Selling 'milk' is more favored by the presence of 'bread' and 'frozen vegetables'. 1. Strategies to sell high margin products: Neighboring placement of 'burgers' and 'milk' alongside 'cake' can help increase sales of 'cake'. Similarly, placement of 'light cream' and 'olive oil' alongside 'Chicken' can increase sales of 'Chicken', and 'spaghetti' and 'milk' can increase sales of 'frozen vegetables'. 1. Sale on products that can help to sell other products: Discounts on 'fromage blanc' will boost sales of 'honey'. Sales on 'Ground beef' and 'spaghetti' will increase selling of 'tomato sauce' and 'olive oil'. **Table of Contents** 1. Exploring Dataset Exploring Itemset 3. Apriori Algorithm theory 4. Preparation 5. Rules resulted from the Apriori algorithm 6. Finding answers to the business objectives 7. Conclusion 1. Exploring Dataset In [1]: import numpy as np, pandas as pd import matplotlib.pyplot as plt, seaborn as sns from wordcloud import WordCloud import squarify from mlxtend.frequent patterns import apriori from mlxtend.frequent patterns import association rules In [2]: data = pd.read csv('dataset.csv', header = None) print(data.shape) data.head() (7501, 20)Out[2]: 15 whole low vegetables cottage tomato green energy antic green mineral shrimp salmon almonds avocado weat yams fat honey salad mix grapes cheese drink juice tea water flour yogurt burgers meatballs NaN NaN NaN NaN eggs NaN chutney NaN NaN NaN NaN NaN NaN NaN NaN turkey avocado NaN mineral energy whole green milk NaN bar wheat rice tea The dataset is a sample that contains list of products purchased by 7500 customers. Some customers have only purchased one product whereas maximum 20 products have been purchased by one customer. 2. Exploring Itemset In [3]: itemset = [] for i in range(data.shape[0]): itemset.append([str(data.values[i,j]) for j in range(data.shape[1])]) itemset contains the list of products separately, but, 'nan' have been inserted wherever the purchase had less than 20 products. In [4]: for product in itemset[4]: print(product, end=" | ") mineral water | milk | energy bar | whole wheat rice | green tea | nan | Let's create a cleaned itemset from itemset having only valid entries to explore itemset through 'Word cloud', 'Frequency plot', and 'Tree Map'. In [5]: cleaned itemset = [] for each in itemset: for item in each: if str(item).lower() != 'nan': cleaned itemset.append(item) print("There are {} unique items.".format(len(set(cleaned\_itemset)))) There are 120 unique items. In [6]: plt.rcParams['figure.figsize'] = (15, 10) wordcloud = WordCloud(background color = 'white', width = 1200, height = 600)\ .generate(str(cleaned itemset)) plt.imshow(wordcloud) plt.axis('off') plt.title('Most Popular Items', fontsize = 25) plt.show() Most Popular Items green grapes' fries' cookies pancakes' cake beef' herb pepper champagne pancakes' eggs wheat pasta parmesan cheese cake ted light mayo escalope water uice eggs'turkey' burgers fresh tuna' protein bar candy bars brownies oil french beef mineral milk' olive wine honey ' eggs' cream sauce chocolate milk mi In [7]: # looking at the frequency of Top 20% popular items ten = int(0.15\*len(set(cleaned itemset))) plt.rcParams['figure.figsize'] = (15, 10) color = plt.cm.Purples(np.linspace(0.2, 1, ten)) pd.DataFrame(cleaned itemset)[0].value counts().head(ten).sort values().plot.barh(color = color) plt.title('Frequency of Top 10% popular items', fontsize = 20) plt.xticks(rotation = 90 ) plt.show() Frequency of Top 10% popular items mineral water eggs spaghetti french fries chocolate green tea ground beef frozen vegetables pancakes burgers cake cookies escalope low fat yogurt shrimp tomatoes olive oil 250 20 In [8]: tree = pd.DataFrame(cleaned itemset)[0].value counts().head(ten).to frame() plt.rcParams['figure.figsize'] = (15, 10) color = plt.cm.Purples(np.linspace(0.2, 1, ten)) squarify.plot(sizes = tree.values, label = tree.index, alpha=.8, color = color) plt.title('Tree Map for Top 10% Popular Items', fontsize = 20) plt.axis() plt.show() Tree Map for Top 10% Popular Items 100 shrimp olive oil frozen vegetables spaghetti 80 low fat yogurt tomatoes ground beef cookies 60 eggs milk pancakes burgers mineral water 20 french fries chocolate green tea 100 3. Apriori algorithm theory Association Rule finds an association between different products in a set and finds frequent patterns in a transaction database such as purchasing behavior analysis. The applications of Association Rule are in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering, and classification. The association rule learning has three popular algorithms – Apriori, Eclat, and FP-Growth. This repository is dedicated to the Apriori algorithm for Market Basket Analysis. The approach is based on the theory that customers who buy a specific product (or group of products) are more likely to buy another particular product (or group of products). Market Basket Analysis aims to find relationships and establish patterns across purchases. The relationship is modeled in the form of a conditional algorithm: IF { itemset 'A' } THEN { itemset 'C' } itemset': A collection of products/items purchased by a customer antecedent: The set of products on the Left-hand side of the rule consequent: The set of products on the Right-hand side of the rule. 4. Preparation **Removing NaNs** In [9]: new = { 'items': []} for i in range(data.shape[0]): product list = [] for j in range(data.shape[1]): product = data.iloc[i,j] if str(product).lower() != 'nan': product list.append(product) new['items'].append(product list) new df = pd.DataFrame(new) new df.head(3) Out[9]: items [shrimp, almonds, avocado, vegetables mix, gre... [burgers, meatballs, eggs] [chutney] One-hot encoding In [10]: def do one hot(dataframe): df new = pd.DataFrame() #For every row in the dataframe, iterate through the list of items and place a 1 into the correspon for index, row in dataframe.iterrows(): for each in row['items']: df new.at[index, each] = 1#Filling in the NaN values with 0 to show that a transaction doesn't have that column's item df new = df new.fillna(0) dataframe = dataframe.drop(['items'], axis=1) dataframe = pd.concat([dataframe, df new], axis=1) return dataframe new df = do one hot(new df)new df.head(3) Out[10]: whole vegetables green cottage energy tomato shrimp almonds avocado ... melons cauliflower ketchup brar weat yams cheese drink juice beans mix grapes flour 0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3 rows × 120 columns 5. Rules resulted from the Apriori algorithm Suppose we want to find the association of products which are sold at least 3 times a day. So, the minimum support here will be (3 times per day) (7 days a week) / (the total number of transactions). That means (37)/7501 = 0.00279. So the equivalent 0.003 is taken here as support. In [11]: itemsets = apriori(new df, min support=0.003, use colnames=True) rules = association rules(itemsets, metric="lift", min threshold=1.5) rules.shape Out[11]: (3542, 9) In [12]: rules.iloc[27] Out[12]: antecedents (shrimp) consequents (pasta) 0.0714571 antecedent support 0.0157312 consequent support 0.00506599 support confidence 0.0708955 lift 4.50667 leverage 0.00394188 conviction 1.05937 Name: 27, dtype: object Explanation for the above resulted rule: For instance from the first item, we can see that 'pasta' and 'shrimp' are commonly bought together. This makes sense because "the shrimp The antecedent support is 0.0157312 means that 'pasta' is purchased in 118 transactions out of 7501. The **consequent support** is 0.071457 means that 'shrimp' is purchased in 536 transactions out of all. The **support** is 0.005066 means that 'pasta' and 'shrimp' are both purchased together in 38 transactions out of all. The **confidence level** for the rule is 0.32 which shows that out of all the transactions that contain 'pasta', 32% of the transactions also contain 'shrimp'. The confidence is 1 (maximal) for a rule A->C if the consequent and antecedent always occur together. • The lift of 4.5 tells us that 'shrimp' is 4.5 times more likely to be bought by the customers who buy 'pasta' compared to the default likelihood of the sale of 'shrimp'. If Antecedent and Consequent are independent, the Lift score will be exactly 1. • The leverage of 0.003942 indicates the difference of 30 transactions between the observed frequency of 'Pasta and Shrimp' appearing together and the frequency that would be expected if they were independent. An leverage value of 0 indicates independence. Finally, the **conviction** value 1.369601 is the dependency measure of 'Shrimp' on 'Pasta'. A high conviction value means that the consequent is highly depending on the antecedent. For instance, in the case of a perfect confidence score, the denominator becomes 0 (due to 1 - 1) for which the conviction score is defined as 'inf'. Similar to lift, if items are independent, the conviction is 1. 6. Finding answers to the business objectives 1. Keystone Products: Keystone products are those that differentiate themselves in the market and could potentially hurt business if they were unavailable or more expensive. Unavailability is the question of Stock-outs. Expensiveness of the product is subjected to Cost/Pricing analysis. Keystone products are generally the antecedent products with the highest support values. Support is the probability that the customer will buy the product. In [13]: key\_list = rules.sort\_values(by="antecedent support", ascending=False)[:1500]['antecedents'].to\_list() keystone = [] for each in key list: for product in each: keystone.append(product) if product not in keystone else None print(len(keystone)) for product in keystone: print(product, end=" | ") 20 mineral water | eggs | spaghetti | french fries | chocolate | green tea | milk | ground beef | frozen vegetables | pancakes | burgers | cake | cookies | escalope | low fat yogurt | shrimp | tomatoes | ol ive oil | frozen smoothie | turkey | Here, we are taking unique products from top 20% Antecedents in terms of support values. Keystone products are: 'mineral water', 'eggs', 'spaghetti', 'french fries', 'chocolate', 'green tea', 'milk', 'ground beef', 'frozen vegetables', 'pancakes', 'burgers', 'cake', 'cookies', 'escalope', 'low fat yogurt', 'shrimp', 'tomatoes', 'olive oil', 'frozen smoothie', 'turkey'. 2. Product placement strategy: Confidence can be used for product placement strategy and increasing sales. The probability that a customer will purchase a Consequent on the condition of purchasing an antecedent is referred to as the confidence of the rule. Product pairs with highest confidence should be placed together always. In [14]: rules[(rules.consequents!={'mineral water'}) & (rules.confidence >= 0.5)]\ .sort\_values(by='confidence', ascending=False).head(10) Out[14]: antecedent consequent antecedents consequents support confidence lift leverage conviction support support 3002 (cereals, ground beef) (spaghetti) 0.004533 0.174110 0.003066 0.676471 3.885303 0.002277 2.552751 2086 (tomatoes, olive oil) (spaghetti) 0.007199 0.174110 0.004399 0.611111 3.509912 0.003146 2.123717 (soup, frozen vegetables, 3265 0.005066 0.129583 0.003066 0.605263 4.670863 0.002410 (milk) 2.205057 mineral water) 2992 0.005333 0.174110 0.003066 3.302508 0.002138 1.943270 (ground beef, tomato sauce) (spaghetti) 2874 0.008399 0.174110 0.004799 0.571429 3.281995 0.003337 1.927076 (ground beef, cooking oil) (spaghetti) (mineral water, tomatoes, 3461 0.005466 0.174110 0.003066 0.560976 3.221959 0.002115 1.881194 (spaghetti) ground beef) (chocolate, frozen vegetables, 3531 0.005733 0.174110 0.003066 0.534884 3.072100 0.002068 1.775663 (spaghetti) ground beef) (frozen vegetables, ground 3517 0.005733 0.174110 0.003066 0.534884 3.072100 0.002068 1.775663 (spaghetti) beef, milk) 2404 0.007066 0.174110 0.003733 0.528302 3.034297 0.002503 (red wine, eggs) (spaghetti) 1.750887 822 (shrimp, ground beef) (spaghetti) 0.011465 0.174110 0.005999 0.523256 3.005315 0.004003 1.732354 3. Strategy to sell high margin products: Placing high margin items near associated high confidence (driver) items (if one of the Keystone products than better) can increase the overall margin on purchases. Let's say that 'spaghetti' and 'ground beef' are the high margin products. In [15]: bottom = rules.sort\_values(by="antecedent support")[:75]['antecedents'].to\_list() least sold = [] for each in bottom: for product in each: if product not in least\_sold: least sold.append(product) if product not in keystone else None print(len(least sold)) for product in least sold: print(product, end=" | ") 15 soup | cereals | tomato sauce | light cream | chicken | herb & pepper | whole wheat rice | salmon | g rated cheese | pepper | fresh bread | cottage cheese | red wine | cooking oil | avocado | In [16]: rules[rules['consequents'] == {'cake'}].sort values(by='confidence', ascending=False).head(2) Out[16]: antecedent consequent support confidence antecedents consequents lift leverage conviction support support 2204 (burgers, milk) 0.017864 0.081056 0.003733 0.208955 2.577916 0.002285 1.161684 (cake) (burgers, mineral 1326 (cake) 0.024397 0.081056 0.004799 0.196721 2.426984 0.002822 1.143992 water) rules[rules['consequents'] == {'chicken'}].sort values(by='confidence', ascending=False).head(2) In [17]: Out[17]: antecedents consequents antecedent support consequent support support confidence lift leverage conviction 468 (light cream) 0.015598 0.059992 0.004533 0.290598 4.843951 0.003597 1.325072 (chicken) 0.017064 2030 (olive oil, milk) (chicken) 0.059992 0.003600 0.210938 3.516094 0.002576 1.191297 In [18]: rules[rules['consequents'] == {'frozen smoothie'}].sort values(by='confidence', ascending=False).head(2 Out[18]: antecedent consequent antecedents consequents support confidence lift leverage conviction support support (spaghetti, milk, mineral (frozen 0.203390 3.211847 3076 0.015731 0.063325 0.003200 0.002203 1.175826 smoothie) (frozen 1953 0.035462 0.063325 0.005599 0.157895 2.493407 0.003354 1.112302 (spaghetti, milk) smoothie) 4. Sale on products that can help to sell other products: Providing discounts or sale on the products which provide greater lift to the consequents can help improve the overall sales metrics. In [22]: rules[rules.lift>=5].sort values(by=["lift", "antecedent support"], ascending=False) Out[22]: antecedent consequent support confidence antecedents consequents lift leverage conviction support support (soup, frozen 3273 0.047994 0.007999 0.003066 0.063889 7.987176 0.002682 1.059704 (milk, mineral water) vegetables) (soup, frozen 0.383333 7.987176 0.002682 3268 0.007999 0.003066 (milk, mineral water) 0.047994 1.543794 vegetables) (olive oil, frozen 0.003333 0.294118 1.348676 3092 0.011332 0.047994 6.128268 0.002789 (milk, mineral water) vegetables) (olive oil, frozen (milk, mineral water) 3097 0.047994 0.011332 0.003333 6.128268 0.002789 0.069444 1.062449 vegetables) (whole wheat pasta 6.115863 0.003234 1224 (olive oil) 0.003866 0.402778 0.009599 0.065858 1.564145 mineral water) (whole wheat pasta, 0.003866 1225 0.065858 0.009599 0.058704 6.115863 0.003234 1.052168 (olive oil) mineral water) (chocolate, frozen 3050 0.022930 0.023597 0.003200 0.139535 5.913283 0.002658 (shrimp, mineral water) 1.134739 vegetables) (chocolate, frozen 0.003200 5.913283 0.002658 3055 0.023597 0.022930 0.135593 1.130336 (shrimp, mineral water) vegetables) (frozen vegetables, 0.015198 3270 0.035729 0.003066 5.646864 0.002523 (soup, milk) 0.085821 1.077253 mineral water) (frozen vegetables, 1.207988 3271 0.035729 0.003066 0.201754 5.646864 0.002523 (soup, milk) 0.015198 mineral water) 0.023064 5.634140 3272 (soup, mineral water) (frozen vegetables, milk) 0.023597 0.003066 0.132948 0.002522 1.126118 0.002522 3269 (frozen vegetables, milk) 0.023597 0.023064 0.003066 0.129944 5.634140 1.122842 (soup, mineral water) 0.014131 5.535971 2990 (spaghetti, ground beef) (tomato sauce) 0.039195 0.003066 0.078231 0.002512 1.069540 2995 (spaghetti, ground beef) 0.003066 0.002512 1.227052 0.014131 0.039195 0.216981 5.535971 (tomato sauce) (frozen vegetables, 3275 0.050527 0.003066 0.060686 (soup) 0.011065 5.484407 0.002507 1.052827 mineral water, milk) (frozen vegetables, 3266 0.050527 0.003066 0.277108 5.484407 0.002507 (soup) 0.011065 1.313438 mineral water, milk) (frozen vegetables, 3096 0.017064 0.003333 0.093284 5.466564 0.002723 1.084061 (olive oil, milk) 0.035729 mineral water) (frozen vegetables, 3093 0.035729 0.003333 0.195313 5.466564 0.002723 1.198318 (olive oil, milk) 0.017064 mineral water) 0.013598 0.002688 171 (fromage blanc) 0.047460 0.003333 0.070225 5.164271 1.060903 (honey) 170 (fromage blanc) 0.003333 5.164271 1.261806 (honey) 0.013598 0.047460 0.245098 0.002688 3094 (olive oil, mineral water) 0.027596 0.023597 0.003333 0.120773 5.118180 0.002682 1.110524 (frozen vegetables, milk) 3095 0.027596 0.003333 0.141243 5.118180 0.002682 1.132338 (frozen vegetables, milk) 0.023597 (olive oil, mineral water) (frozen vegetables, 0.002466 3519 (spaghetti, milk) 0.035462 0.016931 0.003066 0.086466 5.106950 1.076117 ground beef) (frozen vegetables, 0.181102 5.106950 3522 (spaghetti, milk) 0.016931 0.035462 0.003066 0.002466 1.177849 ground beef) (spaghetti, frozen 3036 (shrimp, mineral water) 0.027863 0.023597 0.003333 0.119617 5.069202 0.002675 1.109067 vegetables) (spaghetti, frozen 0.023597 3041 (shrimp, mineral water) 0.027863 0.003333 0.141243 5.069202 0.002675 1.132028 vegetables) (spaghetti, frozen 5.002842 3518 (milk, ground beef) 0.027863 0.021997 0.003066 0.110048 0.002453 1.098939 vegetables) (spaghetti, frozen 3523 (milk, ground beef) 0.021997 0.027863 0.003066 0.139394 5.002842 0.002453 1.129596 vegetables) 7. Conclusion Market Basket Analysis or the rules resulted applying The Apriori algorithm which is another style of the association learning can help the business find interesting insights to find the Keystone products, their placement with poorly performing products to increase sales and strategies to improve revenue by enabling sell of high margin products.