PROBLEM DEFINICION AND DESING THINKING

Phase1

Reg no :813921121022

Name :D.Rishika

PROJECT NAME:MARKET BASKET INSIGHTS

ABSTRACT

Market basket analysis finds out customers' purchasing patterns by discovering important associations among the products which they place in their shopping baskets. It not only assists in decision making process but also increases sales in many business organizations. Apriori and FP Growth are the most common

algorithms for mining frequent itemsets. For both algorithms predefined minimum support is needed to satisfy for identifying the frequent itemsets. But when the minimum support is low, a huge number of candidate sets will be generated which requires large computation. In this paper, an approach has been proposed to avoid this large computation by reducing the items of dataset with top selling products. Various percentages of top selling products like 30%, 40%, 50%, 55% have been taken and for both algorithms frequent itemsets and association rules are generated. The results show that if top selling items are used, it is possible to get almost same frequent itemsets and association rules within a short time comparing with that outputs which are derived by computing all the items. From time comparison it is also found that FP Growth algorithm takes smaller time than Apriori algorithm.

INTRODUCTION

Market basket analysis is a data mining technique used by retailers to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together.

The adoption of market basket analysis was aided by the advent of electronic point-of-sale (POS) systems. Compared to handwritten records kept by store owners, the digital records generated by POS systems made it easier for applications to process and analyze large volumes of purchase data.

Implementation of market basket analysis requires a background in statistics and data science, as well as some algorithmic computer programming skills. For those without the needed technical skills, commercial, off-the-shelf tools exist

Examples of market basket analysis

Amazon's website uses a well-known example of market basket analysis. On a product page, Amazon presents users with related products, under the headings of "Frequently bought together" and "Customers who bought this item also bought."

Market basket analysis also applies to bricks-and-mortar stores. If analysis showed that magazine purchases often include the purchase of a bookmark, which could be considered an unexpected combination as the consumer did not purchase a book, then the bookstore might place a selection of bookmarks near the magazine rack.

Benefits of market basket analysis

Market basket analysis can increase sales and customer satisfaction. Using data to determine that products are often purchased together, retailers can optimize product placement, offer special deals and create new product bundles to encourage further sales of these combinations.

DATA SOURCE

The different metrics for Market Basket Analysis in Data source are discussed here. To your association rules, you can apply a variety of interesting measures. These include:

- Support
- Confidence
- Lift

Think about this scenario: A popular e-commerce website processed 4000 transactions. They want to figure out how much support, confidence, and lift there is for the two items, a phone, and a phone case, out of 5000 transactions. There are 500 transactions for the phone, 800 transactions for the phone case, and 1000 transactions for both.

Support

- Support will be able to tell us how frequently this particular combination of items was purchased. The percentage of your transactions that contain an association rule is the easiest metric to calculate. It also refers to the percentage of transactions that includes both A and B. Support informs us how frequently an item or a group of items is purchased.

- To calculate, divide the number of transactions that include A by the total number of transactions. *Support = freq (A, B)/N*

  support(phone) = transactions related to phone/total transactions

i.e. support -> 500/4000=12.5%

Do the same calculation for B. This allows us to recognize the frequency of item sets and filter out the less common ones.

Confidence

- With confidence, you can be more specific in your evaluation of this association rule. Confidence provides us with the likelihood of the "consequents" following the **"antecedents."** It also tells us how often A and B are purchased together for the number of times A is purchased.
- Therefore, confidence is calculated with combined/individual transactions.

Confidence = freq (A, B)/ freq(A)

Confidence = combined transactions/individual transactions

i.e. confidence -> 1000/500=20%

As you can see, this metric provides us with a unique and potentially more valuable insight into the nature of client behavior; we get not only frequency but also a measure of likelihood.

Lift

- It represents the power of an association rule over the occurrence of A and B at random. The lift is calculated to determine the sales ratio.
- *Lift = support (A, B) percent/ support (A) percent x support (B) percent*

Lift -> (500) / 12.5 x 20 = 2

When the Lift value is less than 1, buyers are less likely to purchase the combination

DATA PREPROCESSING

These improvements can generate additional sales for the retailer, while making the shopping experience more productive and valuable for customers. By using market basket analysis, customers may feel a stronger sentiment or brand loyalty toward the company.

```
retail <- read_excel('Online_retail.xlsx') retail <-
retail[complete.cases(retail), ] retail <- retail
%>% mutate(Description = as.factor(Description))
retail <- retail %>% mutate(Country =
as.factor(Country))
retail$Date <- as.Date(retail$InvoiceDate)
retail$Time <-
format(retail$InvoiceDate,"%H:%M:%S")
retail$InvoiceNo <-
as.numeric(as.character(retail$InvoiceNo))glimpse(
retail)
```

```
Observations: 406,829
Variables: 10
$ InvoiceNo   <dbl> 536365, 536365, 536365, 536365, 536365, 536365, 536365, 53.
$ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "22752", .
$ Description <fctr> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LANTERN, .
$ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, 3, .
$ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01 08:2.
$ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1.69.
$ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17.
$ Country     <fctr> United Kingdom, United Kingdom, United Kingdom, United Ki.
$ Date        <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-0.
$ Time        <chr> "08:26:00", "08:26:00", "08:26:00", "08:26:00", "08:26:00".
```

After preprocessing, the dataset includes 406,829 records and 10 fields: InvoiceNo, StockCode, Description,

Quantity, InvoiceDate, UnitPrice, CustomerID, Country, Date, Time.

1. Import libraries

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori,
association_rulesdf =
pd.read_csv("GroceryStoreDataSet.csv", names =
['transaction'], sep = ',')
```

- pandas for dataframe management
- mlxtend (machine learning extensions) is a Python library of useful tools for day-to-day data science tasks.
- mlxtend.preprocessing for data preprocessing before performing the task.
- mlxtend.frequent_patterns for association rule task.
- load data as a variable df

	transaction
0	MILK,BREAD,BISCUIT
1	BREAD,MILK,BISCUIT,CORNFLAKES
2	BREAD,TEA,BOURNVITA
3	JAM,MAGGI,BREAD,MILK
4	MAGGI,TEA,BISCUIT

2. One-hot encoding

We have to change the data to the appropriate format before inputting it into the Apriori algorithm. The goal is a **one-hot DataFrame**.

- Change the DataFrame to a list of lists.

```
df = list(df["transaction"].apply(lambda x:x.split(",")))
```

```
[['MILK', 'BREAD', 'BISCUIT'],
 ['BREAD', 'MILK', 'BISCUIT', 'CORNFLAKES'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['JAM', 'MAGGI', 'BREAD', 'MILK'],
 ['MAGGI', 'TEA', 'BISCUIT'],
 ['BREAD', 'TEA', 'BOURNVITA'],
 ['MAGGI', 'TEA', 'CORNFLAKES'],
 ['MAGGI', 'BREAD', 'TEA', 'BISCUIT'],
 ['JAM', 'MAGGI', 'BREAD', 'TEA'],
 ['BREAD', 'MILK'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'COCK', 'BISCUIT', 'CORNFLAKES'],
 ['COFFEE', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'COCK'],
 ['BREAD', 'SUGER', 'BISCUIT'],
 ['COFFEE', 'SUGER', 'CORNFLAKES'],
 ['BREAD', 'SUGER', 'BOURNVITA'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['BREAD', 'COFFEE', 'SUGER'],
 ['TEA', 'MILK', 'COFFEE', 'CORNFLAKES']]
```

list of list

- Use TransactionEncoder from the mlxtend library to change the list of transactions to a one-hot array.

```
one_hot_transformer = TransactionEncoder()df_transform =
one_hot_transformer.fit_transform(df)
```

```
array([[ True, False,  True, False, False, False, False, False,  True,
        False, False],
       [ True, False,  True, False, False,  True, False, False,  True,
        False, False],
       [False,  True,  True, False, False, False, False, False, False,
        False,  True],
       [False, False,  True, False, False, False,  True,  True,  True,
        False, False],
       [ True, False, False, False, False, False, False,  True, False,
        False,  True],
       [False,  True,  True, False, False, False, False, False, False,
        False,  True],
       [False, False, False, False, False,  True, False,  True, False,
        False,  True],
```

one-hot array

It is still hard to read, so we will change it to a dataframe with column names.

- Change the one-hot array to a DataFrame.

```
df =
pd.DataFrame(df_transform,columns=one_hot_transformer.colum
ns_ )
```

	BISCUIT	BOURNVITA	BREAD	COCK	COFFEE	CORNFLAKES	JAM	MAGGI	MILK	SUGER	TEA
0	True	False	True	False	False	False	False	False	True	False	False
1	True	False	True	False	False	True	False	False	True	False	False
2	False	True	True	False	False	False	False	False	False	False	True
3	False	False	True	False	False	False	True	True	True	False	False
4	True	False	False	False	False	False	False	True	False	False	True
5	False	True	True	False	False	False	False	False	False	False	True
6	False	False	False	False	False	True	False	True	False	False	True
7	True	False	True	False	False	False	False	True	False	False	True
8	False	False	True	False	False	False	True	True	False	False	True
9	False	False	True	False	False	False	False	False	True	False	False
10	True	False	False	True	True	True	False	False	False	False	False
11	True	False	False	True	True	True	False	False	False	False	False
12	False	True	False	False	True	False	False	False	False	True	False
13	False	False	True	True	True	False	False	False	False	False	False
14	True	False	True	False	False	False	False	False	False	True	False
15	False	False	False	False	True	True	False	False	False	True	False
16	False	True	True	False	False	False	False	False	False	True	False
17	False	False	True	False	True	False	False	False	False	True	False
18	False	False	True	False	True	False	False	False	False	True	False
19	False	False	False	False	True	True	False	False	True	False	True

one-hot dataframe

This is our desired format, one-hot. The columns are items in the store and each row represents a transaction. If the value is True, that item is sold in that transaction. Now, the data is ready to be fed to the algorithm.

3. Find the frequent itemsets using Apriori

We will use Apriori to find the frequent itemsets from the one-hot transaction DataFrame. This step's objective is to decrease the computational workload in the association rule.

Frequent itemsets' supports are higher than minimum support.
You can adjust min_support to suit your dataset.

```
df = apriori(df, min_support = 0.2, use_colnames = True)
df.sort_values(['support'],ascending=False, inplace =
True)
```

4. Association rule

It is time for the main step!!

The association_rules function will automatically calculate key metrics of our transaction data including support, confidence, lift, leverage, and conviction.

```
df_ar = association_rules(df, metric="lift", min_threshold=1)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(MILK)	(BREAD)	0.25	0.65	0.2	0.800000	1.230769	0.0375	1.750000
1	(BREAD)	(MILK)	0.65	0.25	0.2	0.307692	1.230769	0.0375	1.083333
2	(SUGER)	(BREAD)	0.30	0.65	0.2	0.666667	1.025641	0.0050	1.050000
3	(BREAD)	(SUGER)	0.65	0.30	0.2	0.307692	1.025641	0.0050	1.011111
4	(COFFEE)	(CORNFLAKES)	0.40	0.30	0.2	0.500000	1.666667	0.0800	1.400000
5	(CORNFLAKES)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.800000
6	(COFFEE)	(SUGER)	0.40	0.30	0.2	0.500000	1.666667	0.0800	1.400000
7	(SUGER)	(COFFEE)	0.30	0.40	0.2	0.666667	1.666667	0.0800	1.800000
8	(TEA)	(MAGGI)	0.35	0.25	0.2	0.571429	2.285714	0.1125	1.750000
9	(MAGGI)	(TEA)	0.25	0.35	0.2	0.800000	2.285714	0.1125	3.250000

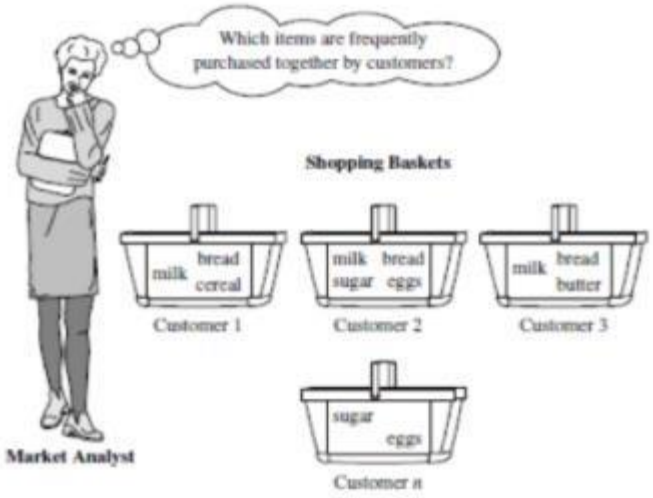
The result is shown below.

The result is required interpretation for further business actions.

ASSOCIATION ANALYSIS

Association rule mining is based on the concept of support, confidence, and lift. Support is the proportion of transactions that contain a certain item or a set of items. Confidence is the probability that a transaction that contains one item or a set of items also contains another item or a set of items. Lift is the ratio of the observed confidence to the expected confidence if the items were independent. An association rule is a statement that expresses the relationship between two or more items, such as bread -> butter, and has a certain support, confidence, and lift.

To apply association rule mining to market basket analysis, you need to have a dataset that contains the transactions or purchases of your customers. You can use any source of data, such as online orders, receipts, loyalty cards, or surveys. You then need to transform your data into a binary matrix, where each row is a transaction and each column is an item. You can use the pandas get_dummies function to do this easily. You then need to use the MLxtend library to generate frequent itemsets and association rules from your data. You can specify the minimum support, confidence, and lift that you want to consider for your rules. You can then explore and analyze the rules to gain insights into your customers' behavior and preferences.



An Example Of Association Rules: -

- Assume There Are 1000 Customers
- 100 Of Them Bought Shampoo, 80 Bought Conditioner And 60 Bought Both Of Them.
- Bought Shampoo => Bought Conditioner
- Support = P(Shampoo & Conditioner) = 60/1000 = 0.06
- Confidence = Support/P(Conditioner) = 0.06/0.08 = 0.75
- Lift = Confidence/P(Shampoo) = 0.75/0.10 = 7.5

This Example Is Extremely Small Representation. In Reality, A Rule Needs The Support Of Several Hundred Transactions, Before It Can Be Consider Statistically Significant, And Datasets Often Contain Thousands Or Millions Of Transactions.

For Instance, Market Basket Analysis May Help You Design Different Store Layouts. In One Strategy, Items That Are Frequently Purchase Together Can Be Place In Proximity To Further Encourage The Combined Sale Of Such Items. If Customers Who Purchase Computers Also Tend To Buy Antivirus Software At The Same Time.

Then Placing The Hardware Display Close To The Software Display May Help Increase The Sales Of Both Items. The Discovery Of Interesting Correlation Relationships Among Huge Amounts Of Business Transaction Records Can Help In Many Business Decision-Making Processes Such As Catalog Design, Cross-Marketing, And Customer Shopping Behavior Analysis.

INSIGHT GENERATIONS

Customer Insights: At its core, MBA enables businesses to gain a deeper understanding of their customers' preferences and behaviors. By uncovering associations between products that

customers frequently buy together, companies can tailor their offerings to meet these preferences. For example, if MBA reveals that customers often purchase toothpaste and toothbrushes together, a store can create bundled promotions to increase sales.

Store Layout Optimization: The layout of a physical store can significantly impact customer behavior. By understanding which products tend to be bought together, businesses can strategically place these items near each other in the store. This can lead to increased crossselling opportunities as customers find related products conveniently.

R Code

```
library("arules") library("arulesViz")
```

```
#Load data set:
data("Groceries") summary(Groceries)
#Look at data:
inspect(Groceries[1]) LIST(Groceries)[1]
```

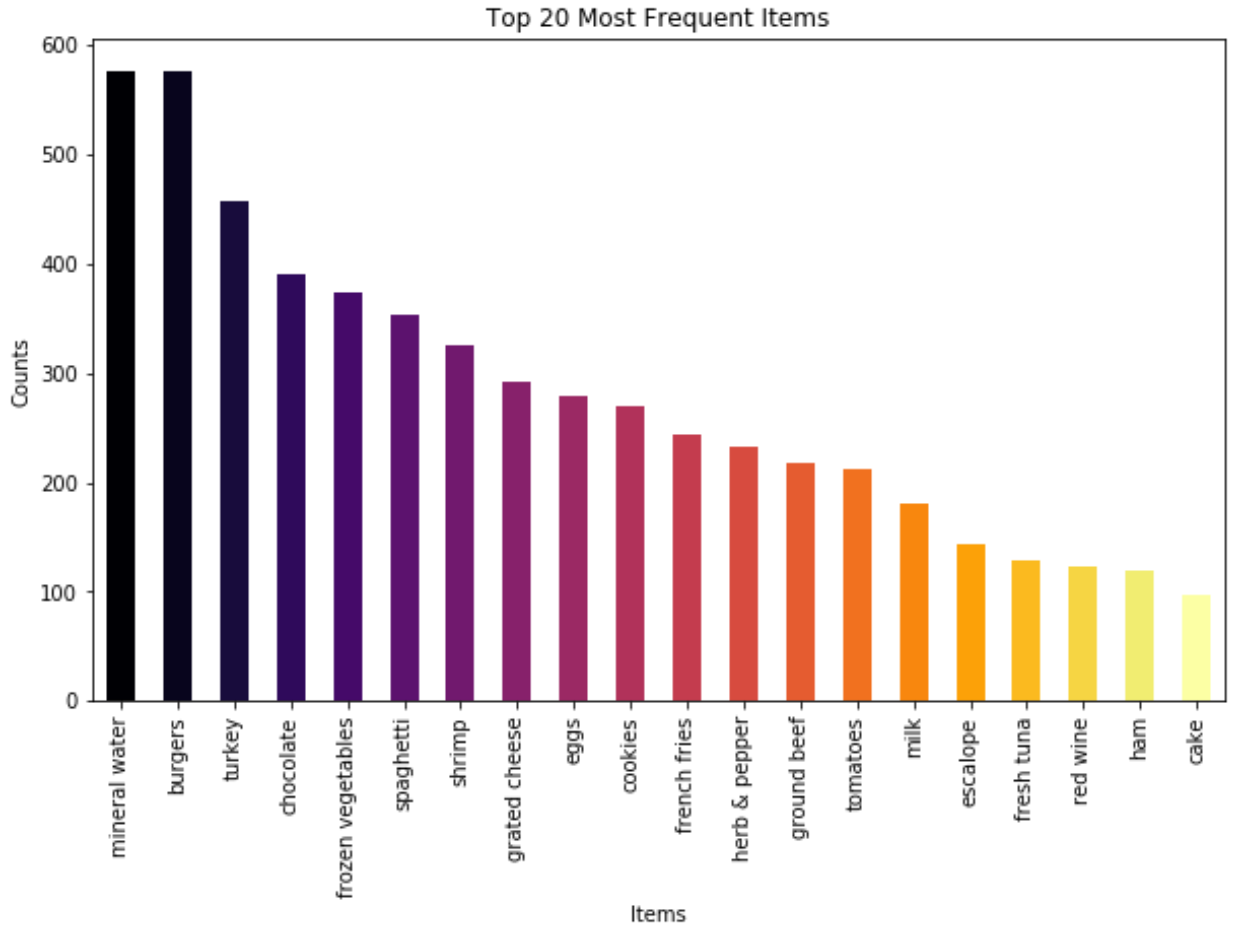
```
#Calculate rules using apriori algorithm and specifying support and confidence thresholds:
rules = apriori(Groceries, parameter=list(support=0.001, confidence=0.5))
#Inspect the top 5 rules in terms of lift:
inspect(head(sort(rules, by ="lift"),5))
#Plot a frequency plot:
itemFrequencyPlot(Groceries, topN = 25)
```

```
#Scatter plot of rules:
library("RColorBrewer") plot(rules,control=list(col=brewer.pal(11,"Spectral")),main="")
#Rules with high lift typically have low support.
#The most interesting rules reside on the support/confidence border which can be clearly seen in this plot.

#Plot graph-based visualisation:
subrules2 <- head(sort(rules, by="lift"), 10)
plot(subrules2, method="graph",control=list(type="items"
```

VISUALIZATION

```
# Most Frequent Items Bar plot
```

```
plt.rcParams['figure.figsize'] = (10,6) color
= plt.cm.inferno(np.linspace(0,1,20))
data[0].value_counts().head(20).plot.bar(color = color) plt.title('Top
20 Most Frequent Items')
plt.ylabel('Counts')
plt.xlabel('Items') plt.show()
```



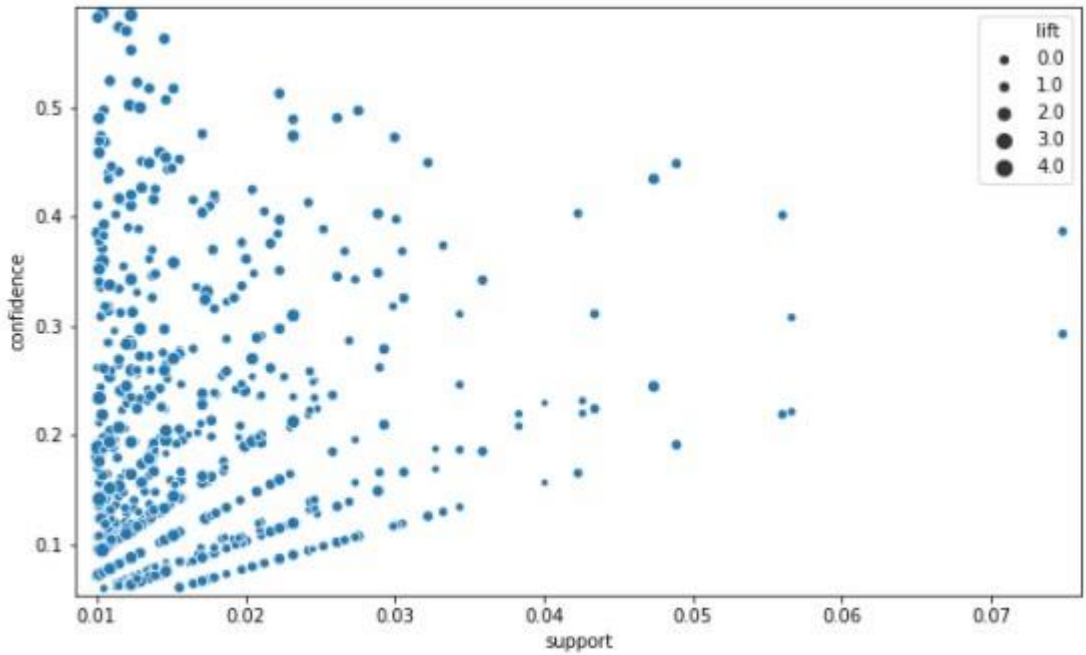

```
import squarify
plt.rcParams['figure.figsize']=(10,10)
Items = data[0].value_counts().head(20).to_frame()
size = Items[0].values lab = Items.index
color = plt.cm.copper(np.linspace(0,1,20))
squarify.plot(sizes=size, label=lab, alpha = 0.7,
color=color) plt.title('Tree map of Most Frequent Items')
plt.axis('off') plt.show()
```

```
data['Items'] = 'items'
df = data.truncate(before=-1,after=15)

import networkx as nx

Items = nx.from_pandas_edgelist(df, source = 'Items', target = 0,
edge_att r = True)

plt.rcParams['figure.figsize'] = (20,20)
nx.draw_networkx_nodes(G=Items,pos=nx.spring_layout(Items), node_size=1500
0,node_color='green')
nx.draw_networkx_edges(G=Items,pos=nx.spring_layout(Items), alpha=0.6, wid
th=3 ,edge_color='black')
nx.draw_networkx_labels(G=Items,pos=nx.spring_layout(Items),font_size=20,
font_family = 'sans-serif') plt.axis('off') plt.grid()
plt.title('Top 15 First Choices', fontsize = 20)
plt.show()
```

BUSINESS RECOMMENDATION

Import seaborn as sns

```
Generate
scatterplot
using support
and
confidence
 plt.figure(figsize=(10,6))
 sns.scatterplot(x = "support", y = "confidence",
                 size = "lift", data = rules)
 plt.margins(0.01,0.01)

              plt.show()
```



Conclusion

In this post, we have had a glimpse into what Affinity Analysis is and how to implement it in python.

Affinity Analysis or Market Basket Analysis is used to extract valuable insights from transaction data. It can be used to determine what products to discount. Also, it can increase sales and customer satisfaction. It is important to realize that there are many other areas in which it can be