MARKET BASKET INSIGHTS

PHASE 2- INNOVATIOIN

Introduction

- Nowadays Machine Learning is helping the Retail Industry in many different ways.
- You can imagine that from forecasting the performance of sales to identify the buyers, there are many applications of machine learning(ML) in the retail industry.
- “Market Basket Analysis” is one of the best applications of machine learning in the retail industry.
- By analyzing the past buying behavior of customers, we can find out which are the products that are bought frequently together by the customers.

- In this article, we will cover a hands-on guide on Market Basket Analysis, its components comprehensively and then deep dive into Market Basket Analysis including how to perform it in Python on a real-world dataset

Market Basket Analysis

- Frequent itemset mining leads to the discovery of associations and correlations between items in huge transactional or relational datasets.

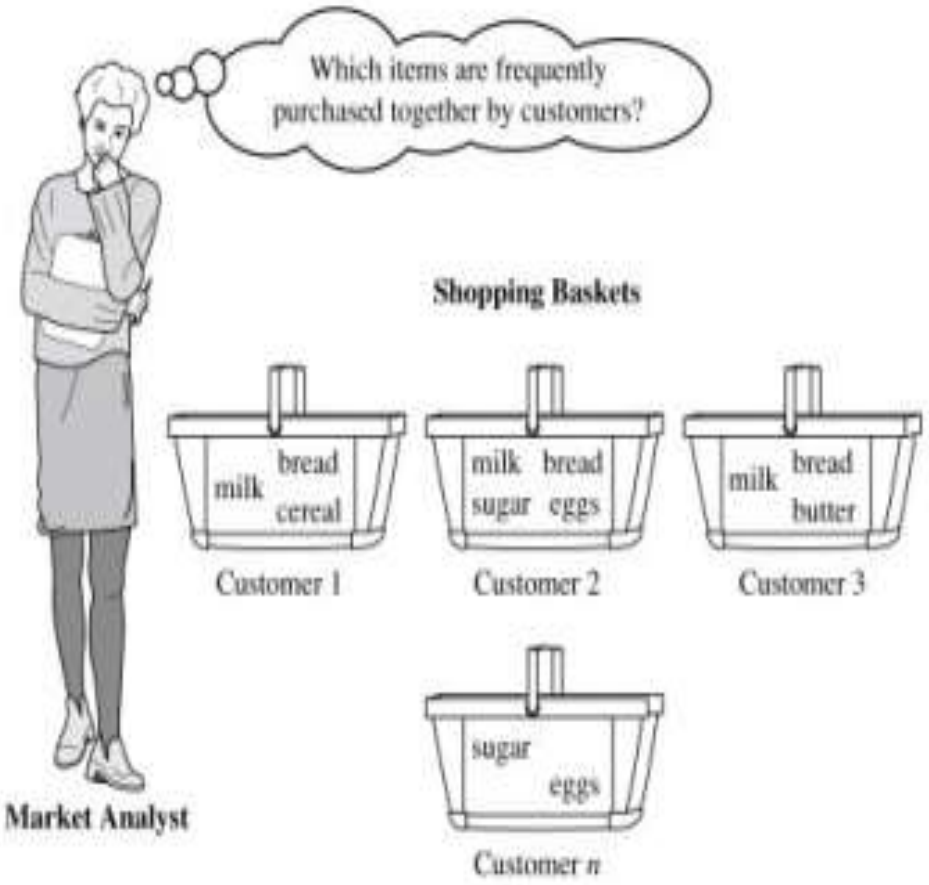
- With vast amounts of data continuously being collected and stored, many industries are becoming interested in mining such kinds of patterns from their databases.

- The disclosure of “Correlation Relationships” among huge amounts of transaction records can help in many decision-making processes such as the design of catalogs, cross-marketing, and behavior customer shopping Analysis.

- A popular example of frequent itemset mining is Market Basket Analysis.

- This process identifies customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.

- The discovery of this kind of association will be helpful for retailers or marketers to develop marketing strategies by gaining insight into which items are frequently bought together by customers.



- Suppose just think of the universe as the set of items available at the store, then each item has a Boolean variable that represents the presence or absence of that item.

- Now each basket can then be represented by a Boolean vector of values that are assigned to these variables.

- The Boolean vectors can be analyzed of buying patterns that reflect items that are frequently associated or bought together.

- Such patterns will be represented in the form of association rules.

Association Rule for Market basket Analysis

- Let I = {I1, I2,…, I m} be an item set. Let D, the data, be a set of database transactions where each transaction T is a nonempty item set such that $T \subseteq I$.

- Each transaction is associated with an identifier, called a TID. Let A be a set of items(item set). T is the Transaction which is said to contain A if $A \subseteq T$. An Association Rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \phi$.

- The rule $A \Rightarrow B$ holds in the data set(transactions) D with supports, where 's' is the percentage of transactions in D that contain $A \cup B$ (that is the union of set A and set B, or, both A and B). This is taken as the probability, $P(A \cup B)$.

- Rule $A \Rightarrow B$ has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contains B. This is taken to be the conditional probability, like $P(B|A)$. That is,

  support(A⇒ B) =P(A ∪ B)

.

  confidence(A⇒ B) =P(B|A)

- Rules that satisfy both a minimum support threshold (called min sup) and a minimum confidence threshold (called min conf) are called "**Strong**".

  Confidence(A⇒ B) = P(B|A) =
  support(A ∪ B) /support(A) =
  support count(A ∪ B) / support count(A)

The Apriori algorithm is created and used for market basket analysis, which is a technique in data mining and business intelligence.

Discovering Association Rules:

The primary purpose of the Apriori algorithm is to discover association rules between items in transactional data. These rules help us understand patterns and relationships between items that frequently co-occur in customer purchases

Market Basket Insights:

By analyzing association rules, businesses can gain valuable market basket insights. These insights provide an understanding of customer buying behavior, product associations, and cross-selling opportunities. They can be used to optimize merchandising strategies, improve product recommendations, and enhance customer experience

Business Decision-Making:

Apriori algorithm enables businesses to make informed decisions by i identifying strong relationships between items. For example, if customers who purchase item A also tend to buy item B, this information can be used to plan promotional campaigns, design product bundles, or optimize inventory management

Retail Optimization:

The algorithm helps retailers optimize store layout, product placement, and stock management. By understanding which items are frequently purchased together, retailers can strategically position related products to encourage upselling and increase sales.

the step-by-step implementation of the Apriori algorithm. Below is an example in Python:

Step 1: Initialize the dataset and minimum support threshold python

```
        dataset = [['Milk', 'Bread', 'Butter'],
                   ['Milk', 'Bread', 'Cheese'],
                   ['Milk', 'Eggs'],
                   ['Bread', 'Butter', 'Cheese'],
                   ['Bread', 'Eggs'],
                   ['Cookies', 'Butter', 'Cheese'],
                   ['Cookies', 'Eggs']]
Min_support = 0.3  # Minimum support threshold (adjust as needed)
```

Step 2: Calculate the support for each item (1-itemset)

```
python
  from collections import defaultdict
def calculate_support(data):
      support_count = defaultdict(int)
      for transaction in data:
            for item in transaction:
                  support_count[item] += 1
       num_records = len(data)
       support = {item: count / num_records
for item, count in
support_count.items()}
    return support

support = calculate_support(dataset)
```

Step 3: Generate frequent 1-itemsets

```
frequent_1_itemsets = {item for item, frequency in support.items()
                                    if frequency >= min_support}
```

Step 4:Generate candidate itemsets iteratively

```
def get_candidate_k_itemsets(frequent_itemsets, k):
    candidates = set()
    for itemset1 in frequent_itemsets:
        for itemset2 in frequent_itemsets:
            if itemset1 != itemset2 and len(itemset1.union(itemset2))
== k:
                candidates.add(itemset1.union(itemset2))
    return candidates

k = 2
frequent_k_itemsets = set()
while True:
    candidates = get_candidate_k_itemsets(frequent_1_itemsets, k)
    frequent_k_itemsets.clear()
    for candidate in candidates:
        count = 0
        for transaction in dataset:
            if candidate.issubset(set(transaction)):
                count += 1
        support = count / len(dataset)
        if support >= min_support:
            frequent_k_itemsets.add(candidate)
    if len(frequent_k_itemsets) == 0:
        break
    k += 1
```

Step 5: Generate association rules from frequent itemsets

```
def generate_association_rules(frequent_itemsets):
    rules = []
    for itemset in frequent_itemsets:
        subsets = powerset(itemset)
        for antecedent in subsets:
            consequent = itemset.difference(antecedent)
            if antecedent and consequent:
                support_itemset = support_count[itemset]
                support_antecedent = support_count[frozenset(antecedent)]
                confidence = support_itemset / support_antecedent
                rules.append((antecedent, consequent, support_itemset,
confidence))
    return rules

association_rules = generate_association_rules(frequent_k_itemsets)
```

Source Programs(Sample):

```
import pandas as pd

df = pd.read_csv('Groceries_dataset.csv')
df.head()
df[‘single_transaction’] = df[‘Member_number’].astype(str)+’_’+df[‘Date’].astype(str)

df.head()
df2 = pd.crosstab(df['single_transaction'], df['itemDescription'])
df2.head()
def encode(item_freq):
   res = 0
   if item_freq > 0:
      res = 1
   return res

basket_input = df2.applymap(encode)
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

frequent_itemsets = apriori(basket_input, min_support=0.001, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift")

rules.head()
rules.sort_values(["support", "confidence","lift"],axis = 0, ascending =False).head(8)
```