NJIT New Jersey Institute of Technology

# CS 634-Data mining

# Midterm Project Implementation

# Topic: Apriori Algorithm

Name: Veena Chaudhari
Email: vac38@njit.edu
Github: https://github.com/vac38/DataMining.git

# Table of contents

# Introduction

The Apriori algorithm is a data mining algorithm that is used in mining frequent product sets and relevant association rules from a transactional database. With the help of these association rules, one can determine how strongly or weakly two objects are connected. For example: In case of shopping for groceries , if we find that milk and bread are strongly connected we can safely say that a customer buying milk is likely to buy bread as well and vice versa.

The overall steps followed to generate association rules are

1) Determine the support of itemsets in the transactional database

   $support\ (X \rightarrow Y)\ = number\ of\ times\ X\ and\ Y\ appear\ together\ /Total\ number\ of\ transactions$

2) Select all itemsets with support greater than specified minimum support.
3) Find all the rules for these subsets and determining confidence of each rule

   $confidence(X \rightarrow Y)\ =\ support(X \rightarrow Y)\ /\ support(X)$

4) Select rules that have higher confidence value than the minimum confidence.

The advantage of apriori algorithm is that it is a simple algorithm which is easy to understand. The algorithm is exhaustive i.e. it finds all possible rules for a given support and confidence value. The cons of the algorithm is that it scans the database multiple times which makes it slow and therefore reduces the overall performance. The algorithm also requires large memory space due to the generation of a large number of itemsets.

# Implementation

The implementation is built from scratch and uses Python 3.8 as the programming language. Each dataset has two parameters that could be specified by user: minimum_support and minimum_confidence.

Input format:
>> main(dataset, minimim_support, minimum_confidence)

The main function calls all other functions to process the data→ calculates frequent item sets-->calculates confidence and then creates association rules.

To determine association rules, 1-itemset, 2-itemset, 3-itemset and 4-itemset are combined to create a final itemset dictionary with frequency for each item. Then Association rules are created by calculating confidence for each itemset using the following formula in point 3 of section 1 ( introduction)

The Association rules with confidence greater than the user specified minimum confidence determined as the final association rules

## Implementation Architecture:

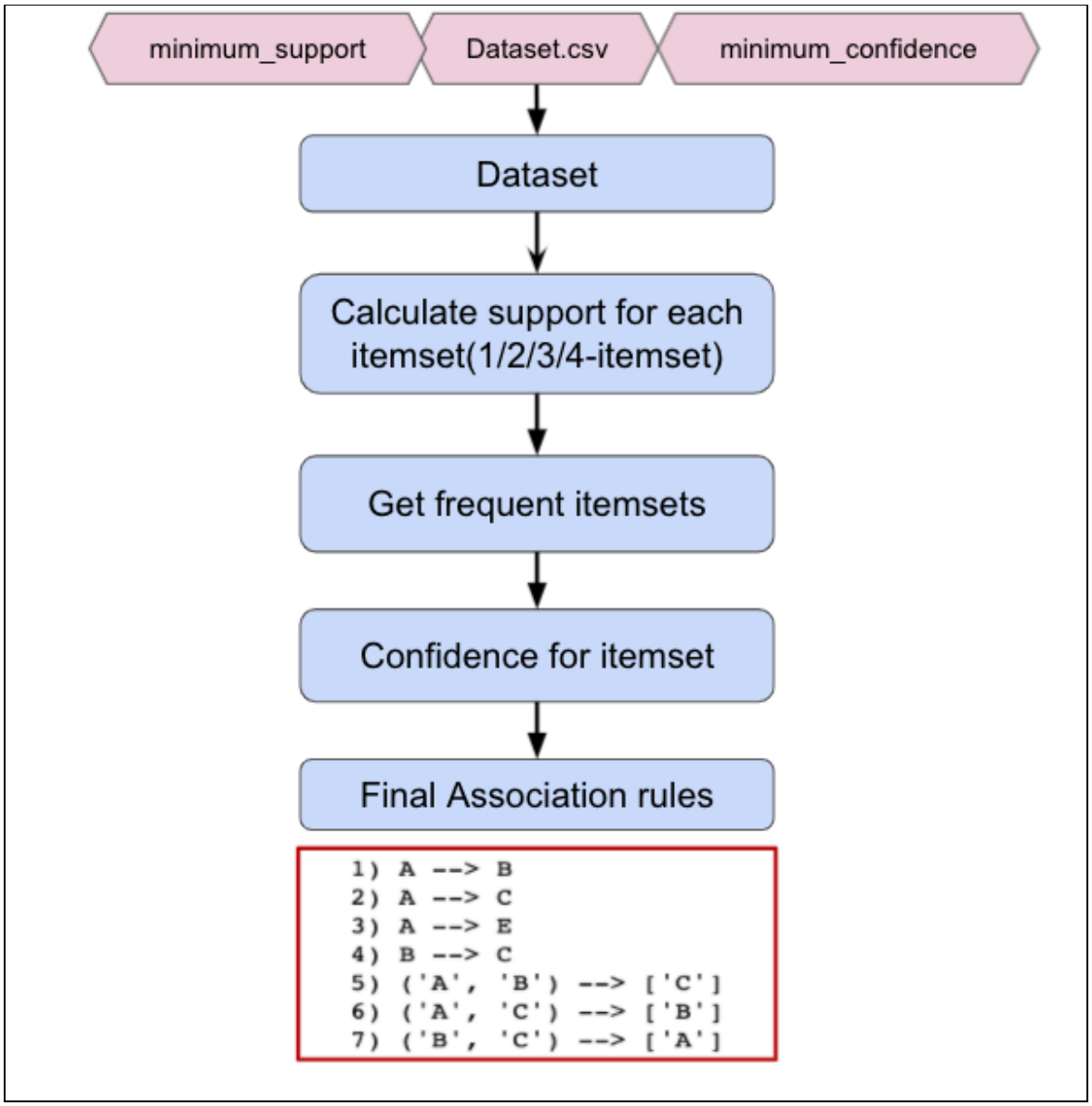


Figure 1: Implementation architecture for Apriori algorithm

# Implementation List of Source Code functions

1) **Main function**:
   -Syntax: *main(file , minimum_support, minimum_confidence)*
   -Input:
      file : link to csv file,
      minimum_support: support value specified by user (integer value)
      minimum_confidence: confidence value specified by user (integer value)
   -Returns: Rules with confidence and final association rules

To create association rules the following functions are called in the main function

2) **data_prep**: Load CSV into a dictionary and remove 'nan' values from the dataset.
   -Syntax: *data_prep(file)*
   -Input:
      File: link to csv file
   -Returns: dataset in dictionary format and unique list of items in the dataset.

3) **create_itemset** : Create an itemset of 1 or 2 or 3 or 3 items using specified n value. The combinations functions from itertools module is used for creating the itemset.
   -Syntax: *create_itemset(k,itemset)*
   -Input:
      k: number of items in the itemset (k can take values from 1 to 4)
      itemset: unique list of itemsets
   -Returns: list of combination of items

4) **frequency:** To calculate the support of each all itemsets by checking the occurrences of the itemsets in the transaction dataset.
   -Syntax: *frequency (item_list, data)*
   -Input:
      item_list: unique item list of items
      data: all transactions from data set in form dictionary
   -Returns: itemset with support for each item

5) **min_support:** Items set with frequency less than the user specified minimum support are removed and lists of items which satisfy the minimum support are returned.

-Syntax: *min_support(item_freq, data, min_sup, n, previous_removed)*

-Input:

item_freq: itemset with support for each item

data: all transactions from data set in form dictionary

min_sup: minimum support

n : is number of item n itemset

Previous_removed: value previously removed itemset

-Returns: lists of items which satisfy the minimum support, list of items removed from itemsets.

6) **Check_if_superset** function is used inside min_support to check if any of the 3-itemset are superset of previously removed 2-itemset.

-Syntax: *check_if_superset(itemset, previ, n)*

-Input:

itemset: unique item from list of items

previ: previously removed 2-itemset

n = is number of item n itemset

-Returns: True if not superset, False if superset

# Requirements

## Software:

python 3.8 with Python libraries installed:  Pandas, numpy, csv and itertools.

## Hardware:

MacBook Air 13-inch, Dual-Core Intel Core i5 with 8 GB RAM

# List of Data Sets

Table 1: Datasets used in project

| | |
|---|---|
| 1 | K-mart |
| 2 | Best Buy |
| 3 | costco |
| 4 | Nike |
| 5 | Generic |

# Testing Implementation and Data Set

## 1) Kmart

```
#File for dataset
kmart = '/Users/veena/Desktop/Dmining/Projects/Datasets/K-mart.csv'
#Specify minimum support
minimum_support = 40
#Specify minimum confidence
minimum_confidence = 50

#To determine association rules, the abover parameters(minimum support and confidence) are pass
for_rules,rules_kmart = main(kmart,minimum_support,minimum_confidence)

#Printing all Rules which satify mininmum confidence criterion. Rules: Tansactiona --> Transact
print('All association rules with confidence \n')
print(rules_kmart,'\n')
print('-------------------------------------------------------------------------\n')
#Printing Final association rules in format A,B --> C
print("Final Association Rules for Kmart with support = {} and Confidence = {} \n" .format(mini
for i in range(len(rules_kmart['Rule_Confidence'])):
    print ("{}) {} --> {} ".format(i+1, rules_kmart['Antecedent'][i],rules_kmart['Consequent'][
```

Figure 2: Python code for Association rules for K-mart

a) Running Kmart dataset with support 40 and confidence 50

```
Final Association Rules for Kmart with support = 40 and Confidence = 50

1) Bed Skirts --> Kids Bedding
2) Kids Bedding --> Bed Skirts
3) Bed Skirts --> Shams
4) Shams --> Bed Skirts
5) Bed Skirts --> Sheets
6) Sheets --> Bed Skirts
7) Kids Bedding --> Shams
8) Shams --> Kids Bedding
9) Kids Bedding --> Sheets
10) Sheets --> Kids Bedding
11) ('Bed Skirts', 'Kids Bedding') --> ['Shams']
12) ('Bed Skirts', 'Shams') --> ['Kids Bedding']
13) ('Kids Bedding', 'Shams') --> ['Bed Skirts']
14) ('Bed Skirts', 'Kids Bedding') --> ['Sheets']
15) ('Bed Skirts', 'Sheets') --> ['Kids Bedding']
16) ('Kids Bedding', 'Sheets') --> ['Bed Skirts']
```

Figure 3: Final association rules for Kmart (support 40 , confidence 50)

b) Running Kmart dataset with support 50 and confidence 75

```
Final Association Rules for Kmart with support = 50 and Confidence = 75

1) Bed Skirts --> Kids Bedding
2) Kids Bedding --> Bed Skirts
3) Kids Bedding --> Sheets
4) Sheets --> Kids Bedding
```

Figure 4: Final association rules for Kmart (support 50 , confidence 75)

c) Running Kmart dataset with support 30 and confidence 90

```
Final Association Rules for Kmart with support = 30 and Confidence = 90

1) Bedspreads --> Bed Skirts
2) Bed Skirts --> Kids Bedding
3) Bedspreads --> Kids Bedding
4) Bedspreads --> Sheets
5) Sheets --> Kids Bedding
6) ('Bed Skirts', 'Bedspreads') --> ['Kids Bedding']
7) ('Bedspreads', 'Kids Bedding') --> ['Bed Skirts']
8) ('Bed Skirts', 'Bedspreads') --> ['Sheets']
9) ('Bedspreads', 'Sheets') --> ['Bed Skirts']
10) ('Bed Skirts', 'Sheets') --> ['Kids Bedding']
11) ('Shams', 'Sheets') --> ['Bed Skirts']
12) ('Bedspreads', 'Kids Bedding') --> ['Sheets']
13) ('Bedspreads', 'Sheets') --> ['Kids Bedding']
14) ('Shams', 'Sheets') --> ['Kids Bedding']
15) ('Bed Skirts', 'Bedspreads', 'Kids Bedding') --> ['Sheets']
16) ('Bed Skirts', 'Bedspreads', 'Sheets') --> ['Kids Bedding']
17) ('Bedspreads', 'Kids Bedding', 'Sheets') --> ['Bed Skirts']
18) ('Bed Skirts', 'Shams', 'Sheets') --> ['Kids Bedding']
19) ('Kids Bedding', 'Shams', 'Sheets') --> ['Bed Skirts']
```

Figure 5: Final association rules for Kmart (support 30 , confidence 90)

d) Running Kmart dataset with support 35 and confidence 60

```
Final Association Rules for Kmart with support = 35 and Confidence = 60

1) Bed Skirts --> Bedspreads
2) Bedspreads --> Bed Skirts
3) Bed Skirts --> Kids Bedding
4) Kids Bedding --> Bed Skirts
5) Bed Skirts --> Shams
6) Shams --> Bed Skirts
7) Bed Skirts --> Sheets
8) Sheets --> Bed Skirts
9) Bedspreads --> Kids Bedding
10) Bedspreads --> Sheets
11) Sheets --> Bedspreads
12) Kids Bedding --> Shams
13) Shams --> Kids Bedding
14) Kids Bedding --> Sheets
15) Sheets --> Kids Bedding
16) Shams --> Sheets
17) Sheets --> Shams
18) ('Bed Skirts', 'Bedspreads') --> ['Kids Bedding']
19) ('Bed Skirts', 'Kids Bedding') --> ['Bedspreads']
20) ('Bedspreads', 'Kids Bedding') --> ['Bed Skirts']
21) ('Bed Skirts', 'Bedspreads') --> ['Sheets']
22) ('Bed Skirts', 'Sheets') --> ['Bedspreads']
23) ('Bedspreads', 'Sheets') --> ['Bed Skirts']
24) ('Bed Skirts', 'Kids Bedding') --> ['Shams']
25) ('Bed Skirts', 'Shams') --> ['Kids Bedding']
26) ('Kids Bedding', 'Shams') --> ['Bed Skirts']
27) ('Bed Skirts', 'Kids Bedding') --> ['Sheets']
28) ('Bed Skirts', 'Sheets') --> ['Kids Bedding']
29) ('Kids Bedding', 'Sheets') --> ['Bed Skirts']
30) ('Bed Skirts', 'Shams') --> ['Sheets']
31) ('Bed Skirts', 'Sheets') --> ['Shams']
32) ('Shams', 'Sheets') --> ['Bed Skirts']
33) ('Bedspreads', 'Kids Bedding') --> ['Sheets']
34) ('Bedspreads', 'Sheets') --> ['Kids Bedding']
35) ('Kids Bedding', 'Sheets') --> ['Bedspreads']
36) ('Kids Bedding', 'Shams') --> ['Sheets']
37) ('Kids Bedding', 'Sheets') --> ['Shams']
38) ('Shams', 'Sheets') --> ['Kids Bedding']
39) ('Bed Skirts', 'Bedspreads', 'Kids Bedding') --> ['Sheets']
40) ('Bed Skirts', 'Bedspreads', 'Sheets') --> ['Kids Bedding']
41) ('Bed Skirts', 'Kids Bedding', 'Sheets') --> ['Bedspreads']
42) ('Bedspreads', 'Kids Bedding', 'Sheets') --> ['Bed Skirts']
43) ('Bed Skirts', 'Kids Bedding', 'Shams') --> ['Sheets']
44) ('Bed Skirts', 'Kids Bedding', 'Sheets') --> ['Shams']
45) ('Bed Skirts', 'Shams', 'Sheets') --> ['Kids Bedding']
46) ('Kids Bedding', 'Shams', 'Sheets') --> ['Bed Skirts']
```

Figure 6: Final association rules for Kmart (support 35 , confidence 60)

# 2) Best Buy

```
bb = '/Users/veena/Desktop/Dmining/Projects/Datasets/Best_buy.csv'
#Specify minimum support
minimum_support = 40
#Specify minimum confidence
minimum_confidence = 90

#To determine association rules, the abover parameters(minimum support and confidence) are passed to main function alog
for_rules,rules_bb = main(bb,minimum_support,minimum_confidence)

#Printing all Rules which satify mininmum confidence criterion. Rules: Tansactiona --> Transaction
print('All association rules with confidence \n')
print(rules_bb,'\n')
print('------------------------------------------------------------------------')
#Printing Final association rules in format A,B --> C
print("Association Rules for Best buy  with support = {} and Confidence = {} \n" .format(minimum_support, minimum_confi
for i in range(len(rules_bb['Rule_Confidence'])):
    print ("{}) {} --> {}  ".format(i+1, rules_bb['Antecedent'][i],rules_bb['Consequent'][i]))
```

Figure 7: Python code for association rules for Best Buy

a) Running Best buy dataset with support 40 and confidence 90

```
Association Rules for Best buy  with support = 40 and Confidence = 90

1) External Hard-Drive --> Anti-Virus
2) Microsoft Office --> Flash Drive
3) Printer --> Flash Drive
4) ('External Hard-Drive', 'Lab Top Case') --> ['Anti-Virus']
5) ('Flash Drive', 'Lab Top Case') --> ['Anti-Virus']
6) ('Anti-Virus', 'Microsoft Office') --> ['Flash Drive']
7) ('Microsoft Office', 'Printer') --> ['Flash Drive']
```

Figure 8: Final association rules for Best buy (support 40 , confidence 90)

b) Running Best buy dataset with support 55 and confidence 80

```
Association Rules for Best buy  with support = 55 and Confidence = 80

1) Anti-Virus --> Lab Top Case
2) Lab Top Case --> Anti-Virus
3) Flash Drive --> Microsoft Office
4) Microsoft Office --> Flash Drive
```

Figure 8: Final association rules for Best buy (support 55 , confidence 80)

c) Running Best buy dataset with support 45 and confidence 95

```
Association Rules for Best buy  with support = 45 and Confidence = 95

1) External Hard-Drive --> Anti-Virus
2) Microsoft Office --> Flash Drive
3) Printer --> Flash Drive
4) ('Flash Drive', 'Lab Top Case') --> ['Anti-Virus']
5) ('Microsoft Office', 'Printer') --> ['Flash Drive']
```

Figure 9: Final association rules for Best buy (support 45 , confidence 95)

d) Running Best buy dataset with support 35 and confidence 90

```
Association Rules for Best buy  with support = 35 and Confidence = 90

1) External Hard-Drive --> Anti-Virus
2) Microsoft Office --> Flash Drive
3) Printer --> Flash Drive
4) ('External Hard-Drive', 'Lab Top Case') --> ['Anti-Virus']
5) ('Flash Drive', 'Lab Top Case') --> ['Anti-Virus']
6) ('Anti-Virus', 'Microsoft Office') --> ['Flash Drive']
7) ('Anti-Virus', 'Printer') --> ['Flash Drive']
8) ('Lab Top Case', 'Microsoft Office') --> ['Anti-Virus']
9) ('Lab Top Case', 'Microsoft Office') --> ['Flash Drive']
10) ('Microsoft Office', 'Printer') --> ['Flash Drive']
11) ('Anti-Virus', 'Lab Top Case', 'Microsoft Office') --> ['Flash Drive']
12) ('Flash Drive', 'Lab Top Case', 'Microsoft Office') --> ['Anti-Virus']
```

Figure 10: Final association rules for Best buy (support 35 , confidence 90)

# 3)Costco

```
costco = '/Users/veena/Desktop/Dmining/Projects/Datasets/costco.csv'
#Specify minimum support
minimum_support = 40
#Specify minimum confidence
minimum_confidence = 30

#To determine association rules, the abover parameters(minimum support and confidence) are passed to main function
for_rules,rules_costco = main(costco,minimum_support,minimum_confidence)

#Printing all Rules which satify mininmum confidence criterion. Rules: Tansactiona --> Transaction
print('All association rules with confidence \n')
print(rules_costco,'\n')
print('-------------------------------------------------------------------------')
#Printing Final association rules in format A,B --> C
print("Association Rules for costco with support = {} and Confidence = {} \n" .format(minimum_support, minimum_conf
for i in range(len(rules_costco['Rule_Confidence'])):
    print ("{}) {} --> {}  ".format(i+1, rules_costco['Antecedent'][i],rules_costco['Consequent'][i]))
```

Figure 11: Python code for Association rules for Costco

a) Running Costco dataset with support 40 and confidence 30

```
Association Rules for costco with support = 40 and Confidence = 30

1) Peanut Butter --> Roasted seaweed
2) Roasted seaweed --> Peanut Butter
3) Peanut Butter --> chocolate almonds
4) chocolate almonds --> Peanut Butter
5) Roasted seaweed --> chocolate almonds
6) chocolate almonds --> Roasted seaweed
7) Roasted seaweed --> coffee
8) coffee --> Roasted seaweed
9) ('Peanut Butter', 'Roasted seaweed') --> ['chocolate almonds']
10) ('Peanut Butter', 'chocolate almonds') --> ['Roasted seaweed']
11) ('Roasted seaweed', 'chocolate almonds') --> ['Peanut Butter']
```

Figure 12: Final association rules for Costco (support 40 , confidence 30)

b) Running Costco dataset with support 45 and confidence 65

```
Association Rules for costco with support = 45 and Confidence = 65

1) Peanut Butter --> Roasted seaweed
2) Roasted seaweed --> Peanut Butter
3) Peanut Butter --> chocolate almonds
4) chocolate almonds --> Peanut Butter
5) Roasted seaweed --> chocolate almonds
6) chocolate almonds --> Roasted seaweed
7) Roasted seaweed --> coffee
8) coffee --> Roasted seaweed
9) ('Peanut Butter', 'Roasted seaweed') --> ['chocolate almonds']
10) ('Peanut Butter', 'chocolate almonds') --> ['Roasted seaweed']
11) ('Roasted seaweed', 'chocolate almonds') --> ['Peanut Butter']
```

Figure 13: Final association rules for Costco (support 45 , confidence 65)

c) Running Costco dataset with support 30 and confidence 75

```
Association Rules for costco with support = 30 and Confidence = 75

1) Peanut Butter --> Roasted seaweed
2) Roasted seaweed --> Peanut Butter
3) Peanut Butter --> chocolate almonds
4) chocolate almonds --> Peanut Butter
5) chocolate almonds --> Roasted seaweed
6) ('Peanut Butter', 'Roasted seaweed') --> ['chocolate almonds']
7) ('Peanut Butter', 'chocolate almonds') --> ['Roasted seaweed']
8) ('Roasted seaweed', 'chocolate almonds') --> ['Peanut Butter']
9) ('Peanut Butter', 'coffee') --> ['Roasted seaweed']
```

Figure 14: Final association rules for Costco (support 30 , confidence 75)

d) Running Costco dataset with support 40 and confidence 40

```
Association Rules for costco with support = 40 and Confidence = 40

1) Peanut Butter --> Roasted seaweed
2) Roasted seaweed --> Peanut Butter
3) Peanut Butter --> chocolate almonds
4) chocolate almonds --> Peanut Butter
5) Roasted seaweed --> chocolate almonds
6) chocolate almonds --> Roasted seaweed
7) Roasted seaweed --> coffee
8) coffee --> Roasted seaweed
9) ('Peanut Butter', 'Roasted seaweed') --> ['chocolate almonds']
10) ('Peanut Butter', 'chocolate almonds') --> ['Roasted seaweed']
11) ('Roasted seaweed', 'chocolate almonds') --> ['Peanut Butter']
```

Figure 15: Final association rules for Costco (support 40 , confidence 40)

# 4) Nike

```
nike = '/Users/veena/Desktop/Dmining/Projects/Datasets/Nike.csv'
#Specify minimum support
minimum_support = 60
#Specify minimum confidence
minimum_confidence = 60
#To determine association rules, the abover parameters(minimum support and confidence) are passed to main function alog
for_rules,rules_nike = main(nike,minimum_support,minimum_confidence)

#Printing all Rules which satify mininmum confidence criterion. Rules: Tansactiona --> Transaction
print('All association rules with confidence \n')
print(rules_nike,'\n')
print('------------------------------------------------------------------------')
#Printing Final association rules in format A,B --> C
print("Association Rules for Nike with support = {} and Confidence = {} \n" .format(minimum_support, minimum_confidence
for i in range(len(rules_nike['Rule_Confidence'])):
    print ("{}) {} --> {}  ".format(i+1, rules_nike['Antecedent'][i],rules_nike['Consequent'][i]))
```

Figure 15: Python code for association rules for Nike

a) Running Nike dataset with support 60 and confidence 60

```
Association Rules for Nike with support = 60 and Confidence = 60

1) Socks --> Sweatshirts
2) Sweatshirts --> Socks
```

Figure 16: Final association rules for Nike (support 60 , confidence 60)

b) Running Nike dataset with support 60 and confidence 90

```
Association Rules for Nike with support = 60 and Confidence = 90

1) Socks --> Sweatshirts
2) Sweatshirts --> Socks
```

Figure 17: Final association rules for Nike (support 60 , confidence 90)

c) Running Nike dataset with support 55 and confidence 70

```
Association Rules for Nike with support = 55 and Confidence = 70

1) Running Shoe --> Socks
2) Socks --> Running Shoe
3) Running Shoe --> Sweatshirts
4) Sweatshirts --> Running Shoe
5) Socks --> Sweatshirts
6) Sweatshirts --> Socks
```

Figure 18: Final association rules for Nike (support 55 , confidence 70)

d) Running Nike dataset with support 55 and confidence 55

```
Association Rules for Nike with support = 55 and Confidence = 55

1) Running Shoe --> Socks
2) Socks --> Running Shoe
3) Running Shoe --> Sweatshirts
4) Sweatshirts --> Running Shoe
5) Socks --> Sweatshirts
6) Sweatshirts --> Socks
```

Figure 19: Final association rules for Nike (support 55 , confidence 55)

# 5)Generic

```
generic = '/Users/veena/Desktop/Dmining/Projects/Datasets/Generic.csv'
#Specify minimum support
minimum_support = 20
#Specify minimum confidence
minimum_confidence = 90

#To determine association rules, the abover parameters(minimum support and confidence) are passed to main
for_rules,rules_generic = main(generic,minimum_support,minimum_confidence)

#Printing all Rules which satify mininmum confidence criterion. Rules: Tansactiona --> Transaction
print('All association rules with confidence \n')
print(rules_generic,'\n')
print('-------------------------------------------------------------------------\n')
#Printing Final association rules in format A,B --> C
print("Association Rules Generic with support = {} and Confidence = {} \n" .format(minimum_support, minimu
for i in range(len(rules_generic['Rule_Confidence'])):
    print ("{}) {} --> {}  ".format(i+1, rules_generic['Antecedent'][i],rules_generic['Consequent'][i]))
```

Figure 20: Python code for association rules for Generic

a) Running Generic dataset with support 20 and confidence 90

```
Association Rules Generic with support = 20 and Confidence = 90

1) B --> A
2) C --> A
3) D --> A
4) E --> A
5) ('B', 'C') --> ['A']
6) ('B', 'D') --> ['A']
7) ('B', 'E') --> ['A']
8) ('C', 'D') --> ['A']
9) ('C', 'E') --> ['A']
10) ('D', 'E') --> ['A']
11) ('B', 'C', 'D') --> ['A']
```

Figure 21: Final association rules for Generic (support 20 , confidence 90)

b) Running Generic dataset with support 35 and confidence 95

```
Association Rules Generic with support = 35 and Confidence = 95

1) B --> A
2) C --> A
3) D --> A
4) E --> A
5) ('B', 'C') --> ['A']
```

Figure 22: Final association rules for Generic (support 35 , confidence 95)

c) Running Generic dataset with support 40 and confidence 42

```
Association Rules Generic with support = 40 and Confidence = 42

1) A --> B
2) B --> A
3) A --> C
4) C --> A
5) D --> A
6) A --> E
7) E --> A
8) B --> C
9) C --> B
10) ('A', 'B') --> ['C']
11) ('A', 'C') --> ['B']
12) ('B', 'C') --> ['A']
```

Figure 23: Final association rules for Generic (support 40 , confidence 42)

d) Running Generic dataset with support 10 and confidence 70

```
Association Rules Generic with support = 10 and Confidence = 70

1) B --> A
2) C --> A
3) D --> A
4) E --> A
5) D --> B
6) D --> C
7) ('B', 'C') --> ['A']
8) ('A', 'D') --> ['B']
9) ('B', 'D') --> ['A']
10) ('B', 'E') --> ['A']
11) ('A', 'D') --> ['C']
12) ('C', 'D') --> ['A']
13) ('C', 'E') --> ['A']
14) ('D', 'E') --> ['A']
15) ('B', 'D') --> ['C']
16) ('C', 'D') --> ['B']
17) ('D', 'E') --> ['B']
18) ('A', 'B', 'D') --> ['C']
19) ('A', 'C', 'D') --> ['B']
20) ('B', 'C', 'D') --> ['A']
21) ('B', 'C', 'E') --> ['A']
22) ('A', 'D', 'E') --> ['B']
23) ('B', 'D', 'E') --> ['A']
24) ('C', 'D', 'E') --> ['A']
25) ('C', 'D', 'E') --> ['B']
```

Figure 24: Final association rules for Generic (support 10 , confidence 70)

# Conclusion

Apriori algorithm was implemented from scratch in python and tested on 5 different datasets (Kmart, Best Buy, Costco, nike and Generic) by specifying different combinations of support and confidence values.

---