# CS 634

# Midterm Project - Apriori Algorithm

***Submitted by:*** Preeti Boddupally

***UCID:*** pb384@njit.edu

# Apriori Algorithm

## Definition

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules.

## Principle

- Any subset of a frequent itemset must be frequent.
- Any superset of a non-frequent itemset must be non must be non-frequent frequent.

## Algorithm

- Find the sets of items that have minimum support (frequent itemsets) starting from 1-itemsets and expanding to k-itemsets; if a j-itemset is already not frequent, then do not consider any supererset of it
- Use the frequent itemsets to generate all association rules

## Key Concepts

- Every association rule has a support and a confidence
- **Support**: The support is the percentage of transactions that demonstrate the rule.

  An itemset is called frequent if its support is greater than or equal to agreed upon minimal support.
  - An association rule is of the form: X => Z i.e., if someone buys X, he also buys Z.
  - To calculate the support of itemset X => Z:
    - ✓ Calculate the support of itemset {X, Z}, i.e. the number of transactions buying X, Z divided by the total number of transactions.
- **Confidence**: The confidence is the **conditional probability** that, given **X** present in a transaction, **Z** will also be present when the association rule is of the form: X => Z
  - To calculate the confidence of X => Z
    - ✓ Calculate the support of itemset set {X}
    - ✓ (Support of {X, Z})/(Support of {X})

## Programming Language used: JAVA

## Source code:

1. ItemSet.java

```
public class ItemSet {
      private String item ;
      private Integer freq;
      private Boolean active = true;
```

```
	public ItemSet(String item, Integer freq) {
		super();
		this.item = item;
		this.freq = freq;
	}
	public String getItem() {
		return item;
	}
	public void setItem(String item) {
		this.item = item;
	}
	public Integer getFreq() {
		return freq;
	}
	public void setFreq(Integer freq) {
		this.freq = freq;
	}
	public Boolean isActive() {
		return active;
	}
	public void setActive(Boolean active) {
		this.active = active;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
	result = prime * result + ((active == null) ? 0 : active.hashCode());
	result = prime * result + ((item == null) ? 0 : item.hashCode());
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		ItemSet other = (ItemSet) obj;
		if (active == null) {
			if (other.active != null)
				return false;
		}
		else if (!active.equals(other.active))
			return false;
		if (item == null) {
			if (other.item != null)
				return false;
		}
		else if (!item.equals(other.item))
			return false;
		return true;
	}
	@Override
	public String toString() {
		return item + "   " + freq;
	}
}
```

## 2. AssociationRule.java

```
public class AssociationRule {

	private static final String PERCENT = "%";
	private static final String FORMAT_DOUBLE = "%.2f";
	private static final String CONFIDENCE = "Confidence";
	private String left;
	private String right;
	private double confidence;
	private String rule;
	public AssociationRule(String left, String right,String rule) {
		super();
		this.left = left;
		this.right = right;
		this.rule = rule;
	}
	public String getleft() {
		return left;
	}
	public void setleft(String left) {
		this.left = left;
	}
	public String getright() {
		return right;
	}
	public void setright(String right) {
		this.right = right;
	}
	public double getConfidence() {
		return confidence;
	}
	public void setConfidence(double confidence) {
		this.confidence = confidence;
	}
	public String getRule() {
		return rule;
	}
	public void setRule(String rule) {
		this.rule = rule;
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
	result = prime * result + ((left == null) ? 0 : left.hashCode());  result =
prime * result + ((right == null) ? 0 : right.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		AssociationRule other = (AssociationRule) obj;
		if (left == null) {
			if (other.left != null)
```

```
                        return false;
                } else if (!left.equals(other.left))
                        return false;
                if (right == null) {
                        if (other.right != null)
                                return false;
                } else if (!right.equals(other.right))
                        return false;
                return true;
        }

        @Override
        public String toString() {
                return left + " -> " + right + "\t[" + CONFIDENCE + " : "
+String.format(FORMAT_DOUBLE, confidence) + PERCENT + "]";
        }

}
```

## 3. AprioriAlgorithm.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;

public class AprioriAlgorithm {
        private static AprioriAlgorithm apriori;
        private static final String COMMA = ",";
        private static final String SPACE_REGEX = "\\s+";
        private List<List<ItemSet>> allItemSet;
        private List<ItemSet> prevItemList;
        private List<ItemSet> currItemList;
        private List<String> transactionList;
        private List<String> itemList;
        private int minimumSupport;
        private int minimumConfidence;

        private List<AssociationRule> associationRules;
        /** Private constructor of aprioriAlgorithm to implement a pattern. **/
        private AprioriAlgorithm() {
                itemList = new ArrayList<String>();
                currItemList = new ArrayList<ItemSet>();
                prevItemList = new ArrayList<ItemSet>();
                transactionList = new ArrayList<String>();
                allItemSet = new ArrayList<List<ItemSet>>();
                associationRules = new ArrayList<AssociationRule>();
        }
        public static AprioriAlgorithm getInstance() {
                if (apriori == null) {
                        apriori = new AprioriAlgorithm();
                }
                return apriori;
        }
```

```
	public void toGenerateRules(int minimumSupport, int minimumConfidence, String
fileName) {
		toReadTransactions(fileName);
		toCalculateSupportConfidence(minimumSupport, minimumConfidence);
		System.out.println();
		System.out.println("Minimum Support Count: " + this.minimumSupport);
		System.out.println("Minimum Confidence : " + this.minimumConfidence);
		System.out.println();
		System.out.print("Items all together present in 20 transactions are :
");
		String itemString = "";
		for (String item : itemList) {
			itemString += item + COMMA;
		}
		itemString = itemString.substring(0, itemString.length() - 1);
		System.out.println(itemString);
		System.out.println();
		Collections.sort(currItemList, new ItemListComparator());
		System.out.println("C1\n");
		toPrintCurrItemsList();
		System.out.println();
		toRemoveNotFreqItems();
		System.out.println("L1\n");
		toPrintCurrItemsList();
		int stage = 2;
		while (true) {
			prevItemList.clear();
			prevItemList.addAll(currItemList);
			currItemList.clear();

			if (stage != 2) {
				List<ItemSet> ItemSets = new ArrayList<ItemSet>(
								prevItemList);
				allItemSet.add(ItemSets);
			}
			if (!toMergeItems(stage)) {
				break;
			}
			toFindFreq();
			System.out.println();
			Collections.sort(currItemList, new ItemListComparator());
			System.out.println("C" + stage + "\n");
			toPrintCurrItemsList();
			System.out.println();
			toRemoveNotFreqItems();
			System.out.println("L" + stage + "\n");
			toPrintCurrItemsList();
			stage++;
		}
		toGenerateAssociationRules();

	}

	public void toGenerateAssociationRules() {
		toRemoveInactiveRules();
		for (ItemSet ItemSet : prevItemList) {
			toGenerateRule(ItemSet);
		}
		System.out.println();
		System.out.println("Association Rules generated from frequent itemsets :
```

```
");
            System.out.println();
            for (AssociationRule rule : associationRules) {
                System.out.println(rule);
            }

            System.out.println();
            System.out.println("Association Rules generated from frequent itemsets
that meet minimum Confidence : ");
            System.out.println();
            for (AssociationRule rule : associationRules) {
                if (rule.getConfidence() >= minimumConfidence) {
                    System.out.println(rule);
                }
            }

        }

        public void toGenerateRule(ItemSet ItemSet) {
            String[] items = ItemSet.getItem().split(COMMA);
            List<String> subsets = toGenerateSubset(items);
            for (int i = 0; i < subsets.size(); i++) {
                String leftSide = subsets.get(i);
                for (int j = 0; j < subsets.size(); j++) {
                    String rightSide = subsets.get(j);
                    if (ToCheckLeftRightSideItems(leftSide, rightSide,
ItemSet.getItem().split(COMMA).length)) {
                        AssociationRule associationRule = new
AssociationRule(leftSide, rightSide, ItemSet.getItem());
                        toCalConfidence(associationRule);
                        associationRules.add(associationRule);
                    }
                }
            }
        }

        public boolean ToCheckLeftRightSideItems(String leftSide,
                String rightSide, int length) {
            boolean flag = true;
            String[] leftSideItems = leftSide.split(COMMA);
            String[] rightSideItems = rightSide.split(COMMA);
            if (leftSideItems.length + rightSideItems.length != length) {
                return false;
            }
            outer: for (String leftItem : leftSideItems) {
                for (String rightItem : rightSideItems) {
                    if (leftItem.equals(rightItem)) {
                        flag = false;
                        break outer;
                    }
                }
            }
            return flag;
        }

        public List<String> toGenerateSubset(String[] items) {
            List<String> subsets = new ArrayList<String>();
            int value = (int) Math.pow(2, items.length) - 1;
            for (int i = 1; i < value; i++) {
                String subset = "";
                String binaryValue = Integer.toBinaryString(i);
```

```
                while (binaryValue.length() != items.length) {
                        binaryValue = "0" + binaryValue;
                }
                for (int j = 0; j < binaryValue.length(); j++) {
                        switch (binaryValue.charAt(j)) {
                        case '1':
                                subset += items[j] + COMMA;
                                break;
                        }
                }
        subset = subset.substring(0, subset.length() - 1);
                subsets.add(subset);
        }
        return subsets;
    }

    public void toCalConfidence(AssociationRule rule) {
        double ruleConfidence = toCalculateFreq(rule.getRule());
        double leftSideConfidence = toCalculateFreq(rule.getleft());
    rule.setConfidence((ruleConfidence / leftSideConfidence) * 100.0);
    }

    public void toRemoveInactiveRules() {
        for (Iterator<ItemSet> iterator = prevItemList.iterator();
iterator.hasNext();) {
                if (!iterator.next().isActive()) {
                        iterator.remove();
                }
        }
    }

    public void toPrintCurrItemsList() {
        for (ItemSet ItemSet : currItemList) {
                if (ItemSet.isActive()) {
                        System.out.println(ItemSet);
                }
        }
    }

    public void toFindFreq() {
        for (ItemSet ItemSet : currItemList) {
                int frequency = toCalculateFreq(ItemSet.getItem());
                ItemSet.setFreq(frequency);
        }

    }

    public boolean toMergeItems(int stage) {
        boolean mergeFlag = false;
        for (int i = 0; i < prevItemList.size(); i++) {
                ItemSet ItemSetOne = prevItemList.get(i);
                if (!ItemSetOne.isActive()) {
                        continue;
                }
                for (int j = i + 1; j < prevItemList.size(); j++) {
                        ItemSet ItemSetTwo = prevItemList.get(j);
                        String mergedString = merge(ItemSetOne.getItem(),
                                        ItemSetTwo.getItem(), stage);
                        if (!mergedString.isEmpty()
                                        && toCheckActiveItems(mergedString)) {
                                mergeFlag = true;
```

```
                        ItemSet ItemSet = new ItemSet(mergedString, 1);
                                if (!currItemList.contains(ItemSet)) {
                                        currItemList.add(ItemSet);
                                }
                        }
                }
        }
        return mergeFlag;
}

public boolean toCheckActiveItems(String item) {
        boolean flag = true;
        outer: for (List<ItemSet> ItemSets : allItemSet) {
                for (ItemSet ItemSet : ItemSets) {
                        if (!ItemSet.isActive()) {
                                String items[] = item.split(COMMA);
                                String value = ItemSet.getItem();
                                String values[] = value.split(COMMA);
                                int count = 0;
                                for (String ItemItem : items) {
                                        if (value.contains(ItemItem)) {
                                                count++;
                                        }
                                }
                                if (count == values.length) {
                                        flag = false;
                                        break outer;
                                }
                        }
                }
        }

        return flag;
}

public String merge(String itemOne, String itemTwo, int stage) {
        String mergedItem = "";
        String[] itemOneArray = itemOne.split(COMMA);
        String[] itemTwoArray = itemTwo.split(COMMA);
        List<String> items = new ArrayList<String>(Arrays.asList(itemOneArray));
        for (String item : itemTwoArray) {
                if (!items.contains(item)) {
                        items.add(item);
                }
        }
        if (items.size() == stage) {
                Collections.sort(items);
                for (String item : items) {
                        mergedItem += item + COMMA;
                }
        mergedItem = mergedItem.substring(0, mergedItem.length() - 1);
        }
        return mergedItem;
}

public int toCalculateFreq(String item) {
        int count = 0;
        for (String transaction : transactionList) {
                String[] items = item.split(COMMA);
                boolean flag = true;
                for (String singleItem : items) {
```

```
                        if (!transaction.contains(singleItem)) {
                                flag = false;
                                break;
                        }
                }
                if (flag) {
                        count++;
                }
        }
        return count;
}

public void toRemoveNotFreqItems() {
        for (ItemSet ItemSet : currItemList) {
                if (ItemSet.isActive()
                                && ItemSet.getFreq() < this.minimumSupport) {
                        ItemSet.setActive(false);
                }
        }

}

public void toCalculateSupportConfidence(int minimumSupport,
                int minimumConfidence) {
Double support = (minimumSupport / 100.0) * transactionList.size();
        this.minimumSupport = support.intValue();
        this.minimumConfidence = minimumConfidence;
}

public void toReadTransactions(String fileName) {
        File file = new File(fileName);
        try {
BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
                String myline = null;
                while ((myline = bufferedReader.readLine()) != null) {
                        String[] transcation = myline.split(SPACE_REGEX);
                        transactionList.add(transcation[1]);
                        String[] items = transcation[1].split(COMMA);
                        for (String item : items) {
                                ItemSet ItemSet = new ItemSet(item, 1);
                                if (!itemList.contains(item)) {
                                        itemList.add(item);
                                        currItemList.add(ItemSet);
                                } else {
                                int index = currItemList.indexOf(ItemSet);
                                        ItemSet = currItemList.get(index);
                                        ItemSet.setFreq(ItemSet.getFreq() + 1);
                                }
                        }
                }
                bufferedReader.close();
        } catch (FileNotFoundException e) {
                System.out.println("File not found");
        } catch (IOException e) {
                System.out.println("Error while reading file");
        }
}

private class ItemListComparator implements Comparator<ItemSet> {
        public int compare(ItemSet ItemSetOne, ItemSet ItemSetTwo) {
                return ItemSetOne.getFreq().compareTo(ItemSetTwo.getFreq());
```

```
		}
	}
}
```

## 4. AprioriOutput.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class AprioriOutput {
	public static void main(String[] args) {
		if (args == null || args.length == 0 || args[0] == null ||
args[0].isEmpty()) {
			System.out.println("Please pass file name");
		}
		else {
			AprioriAlgorithm apriori = AprioriAlgorithm.getInstance();
			BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
			try {
			System.out.print("Enter the percentage of Minimum Support : ");
			int minimumSupport = Integer.parseInt(bufferedReader.readLine());
			System.out.print("Enter the percentage of Minimum Confidence: ");
		int minimumConfidence = Integer.parseInt(bufferedReader.readLine());
		apriori.toGenerateRules(minimumSupport, minimumConfidence, args[0]);
				bufferedReader.close();
			} catch (IOException e) {
				System.out.println("Please enter valid input");
			}
		}
	}
}
```

- **Files containing 20 Transactions:**

1. amazon.txt

amazon.txt — Edited

```
Shampoo,Perfume
Table,Cooker,HairSpray
WallPoster,Shampoo
WallPoster,Shampoo,Perfume,HairSpray
Toaster,Table,Cooker
FloorLamp,Modem,Toaster,Cooker
Shampoo,Perfume,WallPoster
HairSpray,Toaster,HairSpray,Modem
Table,Perfume,Toaster,FloorLamp
Shampoo,HairSpray,Perfume,Knifeset
WallPoster,Toaster,FloorLamp
Knifeset,Shampoo,HairSpray
Toaster,FloorLamp
WallPoster,FloorLamp
Table,Toaster,WallPoster
Shampoo,Perfume,HairSpray,Perfume,Modem
Toaster,FloorLamp,Shampoo
WallPoster,Perfume
Knifeset,FloorLamp,Toaster
Shampoo,HairSpray,FloorLamp
```

2. walmart.txt

walmart

```
Diaper,Bounty,Napkins,Books
Diaper,Bounty,FabricSoftner,Books
Diaper,FabricSoftner,Clothes
Diaper,Bounty,Books
Diaper,TidePod,Clothes
Bulb,Bounty,TidePod
Bulb,Napkins,Candles
Bulb,Bounty,Napkins,Candles
Bulb,Bounty,Napkins,Clothes
Diaper,TidePod,FabricSoftner,Clothes
Diaper,TidePod,FabricSoftner
Bounty,Napkins,Books
Bounty,Napkins,TidePod
Bounty,Napkins,FabricSoftner
Diaper,Candles,Books,Books
Diaper,TidePod,FabricSoftner,Books
Bounty,Napkins,TidePod,FabricSoftner
Bounty,Napkins,Candles,TidePod
Bounty,Napkins,TidePod,Books,Clothes
Bulb,Bounty,Candles,Books,Clothes
```

## 3. shoprite.txt

shoprite

```
Eggs,Tomato,Spinach
Milk,Bread,Gummy
Milk,Eggs
Eggs,Tomato
Tomato,Bread,Nuttela
Nuttela,Bread,Eggs
Bread,Spinach
Spinach,Tomato
Tomato,Spinach,Milk
Milk,Gummy,Butter
Gummy,Spinach,Butter,Bread,Eggs
Eggs,Biscuit,Nuttela,Butter,Milk
Milk,Spinach
Bread,Tomato,Butter,Biscuit
Nuttela,Milk
Tomato,Gummy,Milk
Spinach,Nuttela,Biscuit
Bread,Spinach,Eggs
Butter,Eggs
Gummy,Milk,Tomato
```

## 4. staples.txt

staples

```
Folder,Marker,StickyNote
Folder,Stapler,StickyNote
Pen,Marker,Eraser
Pen,Stapler,StickyNote
Folder,Puncher,Eraser,Board
Folder,Puncher,Eraser
Pen,Stapler,Pushpin,Board
Pen,Stapler,Pushpin
Marker,Puncher
Marker,Puncher
Marker,StickyNote
Pen,Puncher,Eraser
Pen,Puncher
Folder,Board
Folder,StickyNote
Marker,Puncher,Pushpin,Board
Marker,Puncher,Pushpin
Stapler,Pushpin,StickyNote
Folder,Pushpin,Eraser
Pen,Puncher,StickyNote
```

5. jcpenny.txt

jcpenny

```
dress,blazer,top,boots,skirt
jeans,top,jacket,boots
suit,blazer,shoe,boots
dress,boots,skirt
jeans,blazer,shoe
dress,top,jewellery
suit,shoe,jacket,boots
dress,blazer,skirt
suit,shoe,jacket
jeans,jewellery,skirt
suit,jewellery
dress,suit top,boots,jewellery
blazer,shoe,jewellery
jeans,jacket,skirt
top,boots,jewellery
dress,jeans,top,jewellery
top,shoe,skirt,blazer
blazer,jewellery
jeans,suit,top
dress,suit,shoe,blazer,jewellery
```

- **Screenshots of output:**

1. Output with amazon transactions:

```
Association Rules generated from frequent itemsets that meet minimum Confidence :

shoe -> blazer  [Confidence : 71.43%]
blazer -> shoe  [Confidence : 62.50%]
Deexits-MacBook-Air:DataMining pb384$ javac ItemSet.java
Deexits-MacBook-Air:DataMining pb384$ javac AssociationRule.java
Deexits-MacBook-Air:DataMining pb384$ javac AprioriAlgorithm.java
Deexits-MacBook-Air:DataMining pb384$ javac AprioriOutput.java
Deexits-MacBook-Air:DataMining pb384$ java AprioriOutput amazon.txt
Enter the percentage of Minimum Support : 20
Enter the percentage of Minimum Confidence : 40

Minimum Support Count: 4
Minimum Confidence : 40

Items all together present in 20 transactions are :   Shampoo,Perfume,Table,Cooker,HairSpray,WallPoster,Toaster,FloorLamp,Modem,Knifeset

C1

Cooker   3
Modem   3
Knifeset   3
Table   4
WallPoster   7
Perfume   8
HairSpray   8
FloorLamp   8
Shampoo   9
Toaster   9

L1

Table   4
WallPoster   7
Perfume   8
HairSpray   8
FloorLamp   8
Shampoo   9
Toaster   9

C2

Shampoo,Table   0
Table,WallPoster   1
Perfume,Table   1
HairSpray,Table   1
FloorLamp,Table   1
HairSpray,WallPoster   1
FloorLamp,Perfume   1
Perfume,Toaster   1
FloorLamp,HairSpray   1
HairSpray,Toaster   1
Shampoo,Toaster   1
FloorLamp,WallPoster   2
```

```
Shampoo   9
Toaster   9

C2

Shampoo,Table   0
Table,WallPoster   1
Perfume,Table   1
HairSpray,Table   1
FloorLamp,Table   1
HairSpray,WallPoster   1
FloorLamp,Perfume   1
Perfume,Toaster   1
FloorLamp,HairSpray   1
HairSpray,Toaster   1
Shampoo,Toaster   1
FloorLamp,WallPoster   2
Toaster,WallPoster   2
FloorLamp,Shampoo   2
Table,Toaster   3
Perfume,WallPoster   3
Shampoo,WallPoster   3
HairSpray,Perfume   3
Perfume,Shampoo   5
HairSpray,Shampoo   5
FloorLamp,Toaster   6

L2

Perfume,Shampoo   5
HairSpray,Shampoo   5
FloorLamp,Toaster   6

Association Rules generated from frequent itemsets :

Shampoo -> Perfume      [Confidence : 55.56%]
Perfume -> Shampoo      [Confidence : 71.43%]
Shampoo -> HairSpray    [Confidence : 55.56%]
HairSpray -> Shampoo    [Confidence : 71.43%]
Toaster -> FloorLamp    [Confidence : 66.67%]
FloorLamp -> Toaster    [Confidence : 75.00%]

Association Rules generated from frequent itemsets that meet minimum Confidence :

Shampoo -> Perfume      [Confidence : 55.56%]
Perfume -> Shampoo      [Confidence : 71.43%]
Shampoo -> HairSpray    [Confidence : 55.56%]
HairSpray -> Shampoo    [Confidence : 71.43%]
Toaster -> FloorLamp    [Confidence : 66.67%]
FloorLamp -> Toaster    [Confidence : 75.00%]
Deexits-MacBook-Air:DataMining pb384$
```

## 2. Output with walmart transactions:

```
Toaster -> FloorLamp    [Confidence : 66.67%]
FloorLamp -> Toaster    [Confidence : 75.00%]
Deexits-MacBook-Air:DataMining pb384$ java AprioriOutput walmart.txt
Enter the percentage of Minimum Support : 25
Enter the percentage of Minimum Confidence : 35

Minimum Support Count: 5
Minimum Confidence : 35

Items all together present in 20 transactions are :   Diaper,Bounty,Napkins,Books,FabricSoftner,Clothes,TidePod,Bulb,Candles

C1

Bulb   5
Candles   5
Clothes   6
FabricSoftner   7
Diaper   9
Books   9
TidePod   9
Napkins   10
Bounty   13

L1

Bulb   5
Candles   5
Clothes   6
FabricSoftner   7
Diaper   9
Books   9
TidePod   9
Napkins   10
Bounty   13

C2

Bulb,FabricSoftner   0
Bulb,Diaper   0
Candles,FabricSoftner   0
Books,Bulb   1
Bulb,TidePod   1
Candles,Clothes   1
Candles,Diaper   1
Candles,TidePod   1
Diaper,Napkins   1
Bulb,Clothes   2
Books,Candles   2
Clothes,FabricSoftner   2
Books,Clothes   2
Clothes,Napkins   2
```

```
Terminal  Shell  Edit  View  Window  Help                27%  Sat 12:26 PM  pb384
DataMining — -bash — 193×54
Bounty,Candles   3
Clothes,Diaper   3
Clothes,TidePod   3
Bounty,Clothes   3
Bounty,FabricSoftner   3
Bounty,Diaper   3
Books,Napkins   3
Bounty,Bulb   4
FabricSoftner,TidePod   4
Diaper,TidePod   4
Napkins,TidePod   4
Diaper,FabricSoftner   5
Books,Diaper   5
Bounty,TidePod   5
Books,Bounty   6
Bounty,Napkins   9

L2

Diaper,FabricSoftner   5
Books,Diaper   5
Bounty,TidePod   5
Books,Bounty   6
Bounty,Napkins   9

Association Rules generated from frequent itemsets :

FabricSoftner -> Diaper [Confidence : 71.43%]
Diaper -> FabricSoftner [Confidence : 55.56%]
Diaper -> Books [Confidence : 55.56%]
Books -> Diaper [Confidence : 62.50%]
TidePod -> Bounty       [Confidence : 55.56%]
Bounty -> TidePod       [Confidence : 38.46%]
Bounty -> Books [Confidence : 46.15%]
Books -> Bounty [Confidence : 75.00%]
Napkins -> Bounty       [Confidence : 90.00%]
Bounty -> Napkins       [Confidence : 69.23%]

Association Rules generated from frequent itemsets that meet minimum Confidence :

FabricSoftner -> Diaper [Confidence : 71.43%]
Diaper -> FabricSoftner [Confidence : 55.56%]
Diaper -> Books [Confidence : 55.56%]
Books -> Diaper [Confidence : 62.50%]
TidePod -> Bounty       [Confidence : 55.56%]
Bounty -> TidePod       [Confidence : 38.46%]
Bounty -> Books [Confidence : 46.15%]
Books -> Bounty [Confidence : 75.00%]
Napkins -> Bounty       [Confidence : 90.00%]
Bounty -> Napkins       [Confidence : 69.23%]
Deexits-MacBook-Air:DataMining pb384$
```

## 3. Output with shoprite transactions:

```
Terminal  Shell  Edit  View  Window  Help                27%  Sat 12:27 PM  pb384
DataMining — -bash — 193×54
Bounty -> Books [Confidence : 46.15%]
Books -> Bounty [Confidence : 75.00%]
Napkins -> Bounty       [Confidence : 90.00%]
Bounty -> Napkins       [Confidence : 69.23%]
Deexits-MacBook-Air:DataMining pb384$ java AprioriOutput shoprite.txt
Enter the percentage of Minimum Support : 10
Enter the percentage of Minimum Confidence : 60

Minimum Support Count: 2
Minimum Confidence : 60

Items all together present in 20 transactions are :   Eggs,Tomato,Spinach,Milk,Bread,Gummy,Nuttela,Butter,Biscuit

C1

Biscuit   3
Gummy   5
Nuttela   5
Butter   5
Bread   7
Eggs   8
Tomato   8
Spinach   8
Milk   9

L1

Biscuit   3
Gummy   5
Nuttela   5
Butter   5
Bread   7
Eggs   8
Tomato   8
Spinach   8
Milk   9

C2

Biscuit,Gummy   0
Gummy,Nuttela   0
Biscuit,Bread   1
Biscuit,Eggs   1
Biscuit,Tomato   1
Biscuit,Spinach   1
Biscuit,Milk   1
Eggs,Gummy   1
Gummy,Spinach   1
Butter,Nuttela   1
Nuttela,Tomato   1
Nuttela,Spinach   1
```

```
Terminal  Shell  Edit  View  Window  Help                27%  Sat 12:27 PM  pb384
DataMining — -bash — 193×54
Eggs,Spinach   3
Spinach,Tomato   3
Milk,Tomato   3
Gummy,Milk   4

C3

Bread,Gummy,Tomato   0
Bread,Eggs,Tomato   0
Bread,Spinach,Tomato   0
Eggs,Milk,Tomato   0
Eggs,Milk,Spinach   0
Bread,Butter,Gummy   1
Butter,Gummy,Milk   1
Bread,Eggs,Nuttela   1
Eggs,Milk,Nuttela   1
Bread,Butter,Eggs   1
Butter,Eggs,Milk   1
Eggs,Spinach,Tomato   1
Milk,Spinach,Tomato   1
Gummy,Milk,Tomato   2
Bread,Eggs,Spinach   2

L3

Gummy,Milk,Tomato   2
Bread,Eggs,Spinach   2

Association Rules generated from frequent itemsets :

Tomato -> Gummy,Milk    [Confidence : 25.00%]
Milk -> Gummy,Tomato    [Confidence : 22.22%]
Milk,Tomato -> Gummy    [Confidence : 66.67%]
Gummy -> Milk,Tomato    [Confidence : 40.00%]
Gummy,Tomato -> Milk    [Confidence : 100.00%]
Gummy,Milk -> Tomato    [Confidence : 50.00%]
Spinach -> Bread,Eggs   [Confidence : 25.00%]
Eggs -> Bread,Spinach   [Confidence : 25.00%]
Eggs,Spinach -> Bread   [Confidence : 66.67%]
Bread -> Eggs,Spinach   [Confidence : 28.57%]
Bread,Spinach -> Eggs   [Confidence : 66.67%]
Bread,Eggs -> Spinach   [Confidence : 66.67%]

Association Rules generated from frequent itemsets that meet minimum Confidence :

Milk,Tomato -> Gummy    [Confidence : 66.67%]
Gummy,Tomato -> Milk    [Confidence : 100.00%]
Eggs,Spinach -> Bread   [Confidence : 66.67%]
Bread,Spinach -> Eggs   [Confidence : 66.67%]
Bread,Eggs -> Spinach   [Confidence : 66.67%]
Deexits-MacBook-Air:DataMining pb384$
```

# 4. Output with staples transactions:

```
Terminal  Shell  Edit  View  Window  Help                27%  Sat 12:27 PM  pb384
DataMining — -bash — 193×54
Milk,Tomato -> Gummy    [Confidence : 66.67%]
Gummy,Tomato -> Milk    [Confidence : 100.00%]
Eggs,Spinach -> Bread   [Confidence : 66.67%]
Bread,Spinach -> Eggs   [Confidence : 66.67%]
Bread,Eggs -> Spinach   [Confidence : 66.67%]
Deexits-MacBook-Air:DataMining pb384$ java AprioriOutput staples.txt
Enter the percentage of Minimum Support : 15
Enter the percentage of Minimum Confidence : 35

Minimum Support Count: 3
Minimum Confidence : 35

Items all together present in 20 transactions are :   Folder,Marker,StickyNote,Stapler,Pen,Eraser,Puncher,Board,Pushpin

C1

Board   4
Stapler   5
Eraser   5
Pushpin   6
Folder   7
Marker   7
StickyNote   7
Pen   7
Puncher   9

L1

Board   4
Stapler   5
Eraser   5
Pushpin   6
Folder   7
Marker   7
StickyNote   7
Pen   7
Puncher   9

C2

Board,StickyNote   0
Eraser,Stapler   0
Marker,Stapler   0
Puncher,Stapler   0
Eraser,StickyNote   0
Folder,Pen   0
Board,Stapler   1
Board,Eraser   1
Board,Marker   1
Board,Pen   1
Folder,Stapler   1
```

```
Folder,StickyNote   3
Pen,Puncher   3
Marker,Puncher   4

L2

Pushpin,Stapler   3
Stapler,StickyNote   3
Pen,Stapler   3
Eraser,Folder   3
Eraser,Puncher   3
Folder,StickyNote   3
Pen,Puncher   3
Marker,Puncher   4

Association Rules generated from frequent itemsets :

Stapler -> Pushpin      [Confidence : 60.00%]
Pushpin -> Stapler      [Confidence : 50.00%]
StickyNote -> Stapler   [Confidence : 42.86%]
Stapler -> StickyNote   [Confidence : 60.00%]
Stapler -> Pen  [Confidence : 60.00%]
Pen -> Stapler  [Confidence : 42.86%]
Folder -> Eraser        [Confidence : 42.86%]
Eraser -> Folder        [Confidence : 60.00%]
Puncher -> Eraser       [Confidence : 33.33%]
Eraser -> Puncher       [Confidence : 60.00%]
StickyNote -> Folder    [Confidence : 42.86%]
Folder -> StickyNote    [Confidence : 42.86%]
Puncher -> Pen  [Confidence : 33.33%]
Pen -> Puncher  [Confidence : 42.86%]
Puncher -> Marker       [Confidence : 44.44%]
Marker -> Puncher       [Confidence : 57.14%]

Association Rules generated from frequent itemsets that meet minimum Confidence :

Stapler -> Pushpin      [Confidence : 60.00%]
Pushpin -> Stapler      [Confidence : 50.00%]
StickyNote -> Stapler   [Confidence : 42.86%]
Stapler -> StickyNote   [Confidence : 60.00%]
Stapler -> Pen  [Confidence : 60.00%]
Pen -> Stapler  [Confidence : 42.86%]
Folder -> Eraser        [Confidence : 42.86%]
Eraser -> Folder        [Confidence : 60.00%]
Eraser -> Puncher       [Confidence : 60.00%]
StickyNote -> Folder    [Confidence : 42.86%]
Folder -> StickyNote    [Confidence : 42.86%]
Pen -> Puncher  [Confidence : 42.86%]
Puncher -> Marker       [Confidence : 44.44%]
Marker -> Puncher       [Confidence : 57.14%]
Deexits-MacBook-Air:DataMining pb384$
```

## 5. Output with jcpenny transactions:

```
Puncher -> Marker       [Confidence : 44.44%]
Marker -> Puncher       [Confidence : 57.14%]
Deexits-MacBook-Air:DataMining pb384$ java AprioriOutput jcpenny.txt
Enter the percentage of Minimum Support : 12
Enter the percentage of Minimum Confidence : 38

Minimum Support Count: 2
Minimum Confidence : 38

Items all together present in 20 transactions are :   dress,blazer,top,boots,skirt,jeans,jacket,suit,shoe,jewellery

C1

jacket   4
boots   6
skirt   6
jeans   6
dress   7
top   7
suit   7
shoe   7
blazer   8
jewellery   8

L1

jacket   4
boots   6
skirt   6
jeans   6
dress   7
top   7
suit   7
shoe   7
blazer   8
jewellery   8

C2

dress,jacket   0
blazer,jacket   0
jacket,jewellery   0
skirt,suit   0
jacket,skirt   1
jacket,top   1
boots,jeans   1
boots,jewellery   1
shoe,skirt   1
jewellery,skirt   1
dress,jeans   1
jeans,suit   1
```

Terminal Shell Edit View Window Help 26% Sat 12:29 PM pb384

DataMining — -bash — 193×54

```
dress   7
top   7
suit   7
shoe   7
blazer   8
jewellery   8

C2

dress,jacket   0
blazer,jacket   0
jacket,jewellery   0
skirt,suit   0
jacket,skirt   1
jacket,top   1
boots,jeans   1
boots,jewellery   1
shoe,skirt   1
jewellery,skirt   1
dress,jeans   1
jeans,suit   1
jeans,shoe   1
blazer,jeans   1
dress,shoe   1
suit,top   1
shoe,top   1
boots,jacket   2
jacket,jeans   2
jacket,suit   2
jacket,shoe   2
boots,skirt   2
boots,dress   2
boots,suit   2
boots,shoe   2
blazer,boots   2
jeans,skirt   2
skirt,top   2
jeans,jewellery   2
dress,suit   2
blazer,top   2
blazer,suit   2
jewellery,suit   2
jewellery,shoe   2
boots,top   3
dress,skirt   3
blazer,skirt   3
jeans,top   3
dress,top   3
blazer,dress   3
dress,jewellery   3
jewellery,top   3
```

Terminal Shell Edit View Window Help 26% Sat 12:30 PM pb384

DataMining — -bash — 193×54

```
shoe,suit   4
blazer,shoe   5

L2

boots,jacket   2
jacket,jeans   2
jacket,suit   2
jacket,shoe   2
boots,skirt   2
boots,dress   2
boots,suit   2
boots,shoe   2
blazer,boots   2
jeans,skirt   2
skirt,top   2
jeans,jewellery   2
dress,suit   2
blazer,top   2
blazer,suit   2
jewellery,suit   2
jewellery,shoe   2
boots,top   3
dress,skirt   3
blazer,skirt   3
jeans,top   3
dress,top   3
blazer,dress   3
dress,jewellery   3
jewellery,top   3
blazer,jewellery   3
shoe,suit   4
blazer,shoe   5

C3

boots,dress,suit   0
jeans,skirt,top   0
blazer,jewellery,top   0
boots,jacket,suit   1
boots,jacket,shoe   1
blazer,boots,skirt   1
boots,skirt,top   1
blazer,boots,dress   1
boots,dress,top   1
blazer,boots,suit   1
blazer,boots,shoe   1
blazer,boots,top   1
dress,skirt,top   1
jeans,jewellery,top   1
blazer,dress,suit   1
```

```
Terminal  Shell  Edit  View  Window  Help                26%  Sat 12:30 PM  pb384
DataMining — -bash — 193×54

C3

boots,dress,suit   0
jeans,skirt,top   0
blazer,jewellery,top   0
boots,jacket,suit   1
boots,jacket,shoe   1
blazer,boots,skirt   1
boots,skirt,top   1
blazer,boots,dress   1
boots,dress,top   1
blazer,boots,suit   1
blazer,boots,shoe   1
blazer,boots,top   1
dress,skirt,top   1
jeans,jewellery,top   1
blazer,dress,suit   1
dress,jewellery,suit   1
blazer,dress,top   1
blazer,jewellery,suit   1
jewellery,shoe,suit   1
blazer,dress,jewellery   1
jacket,shoe,suit   2
boots,dress,skirt   2
boots,shoe,suit   2
blazer,skirt,top   2
blazer,shoe,suit   2
blazer,jewellery,shoe   2
blazer,dress,skirt   2
dress,jewellery,top   2

L3

jacket,shoe,suit   2
boots,dress,skirt   2
boots,shoe,suit   2
blazer,skirt,top   2
blazer,shoe,suit   2
blazer,jewellery,shoe   2
blazer,dress,skirt   2
dress,jewellery,top   2

Association Rules generated from frequent itemsets :

suit -> jacket,shoe     [Confidence : 28.57%]
shoe -> jacket,suit     [Confidence : 28.57%]
shoe,suit -> jacket     [Confidence : 50.00%]
jacket -> shoe,suit     [Confidence : 50.00%]
jacket,suit -> shoe     [Confidence : 100.00%]
jacket,shoe -> suit     [Confidence : 100.00%]
```

```
Terminal  Shell  Edit  View  Window  Help                26%  Sat 12:31 PM  pb384
DataMining — -bash — 193×54
L3

jacket,shoe,suit   2
boots,dress,skirt   2
boots,shoe,suit   2
blazer,skirt,top   2
blazer,shoe,suit   2
blazer,jewellery,shoe   2
blazer,dress,skirt   2
dress,jewellery,top   2

Association Rules generated from frequent itemsets :

suit -> jacket,shoe     [Confidence : 28.57%]
shoe -> jacket,suit     [Confidence : 28.57%]
shoe,suit -> jacket     [Confidence : 50.00%]
jacket -> shoe,suit     [Confidence : 50.00%]
jacket,suit -> shoe     [Confidence : 100.00%]
jacket,shoe -> suit     [Confidence : 100.00%]
skirt -> boots,dress    [Confidence : 33.33%]
dress -> boots,skirt    [Confidence : 28.57%]
dress,skirt -> boots    [Confidence : 66.67%]
boots -> dress,skirt    [Confidence : 33.33%]
boots,skirt -> dress    [Confidence : 100.00%]
boots,dress -> skirt    [Confidence : 100.00%]
suit -> boots,shoe      [Confidence : 28.57%]
shoe -> boots,suit      [Confidence : 28.57%]
shoe,suit -> boots      [Confidence : 50.00%]
boots -> shoe,suit      [Confidence : 33.33%]
boots,suit -> shoe      [Confidence : 100.00%]
boots,shoe -> suit      [Confidence : 100.00%]
top -> blazer,skirt     [Confidence : 28.57%]
skirt -> blazer,top     [Confidence : 33.33%]
skirt,top -> blazer     [Confidence : 100.00%]
blazer -> skirt,top     [Confidence : 25.00%]
blazer,top -> skirt     [Confidence : 100.00%]
blazer,skirt -> top     [Confidence : 66.67%]
suit -> blazer,shoe     [Confidence : 28.57%]
shoe -> blazer,suit     [Confidence : 28.57%]
shoe,suit -> blazer     [Confidence : 50.00%]
blazer -> shoe,suit     [Confidence : 25.00%]
blazer,suit -> shoe     [Confidence : 100.00%]
blazer,shoe -> suit     [Confidence : 40.00%]
shoe -> blazer,jewellery        [Confidence : 28.57%]
jewellery -> blazer,shoe        [Confidence : 25.00%]
jewellery,shoe -> blazer        [Confidence : 100.00%]
blazer -> jewellery,shoe        [Confidence : 25.00%]
blazer,shoe -> jewellery        [Confidence : 40.00%]
blazer,jewellery -> shoe        [Confidence : 66.67%]
skirt -> blazer,dress   [Confidence : 33.33%]
dress -> blazer,skirt   [Confidence : 28.57%]
```

```
Terminal  Shell  Edit  View  Window  Help                    26%  Sat 12:31 PM  pb384
DataMining — -bash — 193×54
shoe,suit -> blazer     [Confidence : 50.00%]
blazer -> shoe,suit     [Confidence : 25.00%]
blazer,suit -> shoe     [Confidence : 100.00%]
blazer,shoe -> suit     [Confidence : 40.00%]
shoe -> blazer,jewellery        [Confidence : 28.57%]
jewellery -> blazer,shoe        [Confidence : 25.00%]
jewellery,shoe -> blazer        [Confidence : 100.00%]
blazer -> jewellery,shoe        [Confidence : 25.00%]
blazer,shoe -> jewellery        [Confidence : 40.00%]
blazer,jewellery -> shoe        [Confidence : 66.67%]
skirt -> blazer,dress   [Confidence : 33.33%]
dress -> blazer,skirt   [Confidence : 28.57%]
dress,skirt -> blazer   [Confidence : 66.67%]
blazer -> dress,skirt   [Confidence : 25.00%]
blazer,skirt -> dress   [Confidence : 66.67%]
blazer,dress -> skirt   [Confidence : 66.67%]
top -> dress,jewellery  [Confidence : 28.57%]
jewellery -> dress,top  [Confidence : 25.00%]
jewellery,top -> dress  [Confidence : 66.67%]
dress -> jewellery,top  [Confidence : 28.57%]
dress,top -> jewellery  [Confidence : 66.67%]
dress,jewellery -> top  [Confidence : 66.67%]

Association Rules generated from frequent itemsets that meet minimum Confidence :

shoe,suit -> jacket     [Confidence : 50.00%]
jacket -> shoe,suit     [Confidence : 50.00%]
jacket,suit -> shoe     [Confidence : 100.00%]
jacket,shoe -> suit     [Confidence : 100.00%]
dress,skirt -> boots    [Confidence : 66.67%]
boots,skirt -> dress    [Confidence : 100.00%]
boots,dress -> skirt    [Confidence : 100.00%]
shoe,suit -> boots      [Confidence : 50.00%]
boots,suit -> shoe      [Confidence : 100.00%]
boots,shoe -> suit      [Confidence : 100.00%]
skirt,top -> blazer     [Confidence : 100.00%]
blazer,top -> skirt     [Confidence : 100.00%]
blazer,skirt -> top     [Confidence : 66.67%]
shoe,suit -> blazer     [Confidence : 50.00%]
blazer,suit -> shoe     [Confidence : 100.00%]
blazer,shoe -> suit     [Confidence : 40.00%]
jewellery,shoe -> blazer        [Confidence : 100.00%]
blazer,shoe -> jewellery        [Confidence : 40.00%]
blazer,jewellery -> shoe        [Confidence : 66.67%]
dress,skirt -> blazer   [Confidence : 66.67%]
blazer,skirt -> dress   [Confidence : 66.67%]
blazer,dress -> skirt   [Confidence : 66.67%]
jewellery,top -> dress  [Confidence : 66.67%]
dress,top -> jewellery  [Confidence : 66.67%]
dress,jewellery -> top  [Confidence : 66.67%]
Deexits-MacBook-Air:DataMining pb384$
```