Machine Data and Learning
Assignment-3
Apriori, K-Means, and DBSCAN

Maximum Marks: 100
Deadline: 12 April 2025, 11:59pm

1 Apriori Algorithm Implementation [30 Marks]

The Apriori Algorithm is a fundamental data mining technique used to identify frequent itemsets and generate association rules from large datasets. It follows a level-wise approach, where frequent itemsets are expanded iteratively based on the Apriori property: If an itemset is frequent, its subsets must also be frequent. Widely applied in market basket analysis, healthcare, and recommendation systems, the algorithm helps uncover hidden patterns and relationships in transactional data.

Implement the Apriori Algorithm in Python from scratch without using built-in libraries like mlxtend. Your implementation should efficiently generate frequent itemsets and derive association rules based on a given minimum support and confidence threshold. Your implementation will be automatically tested, so it is crucial to follow the exact input and output format specified.

Implement the below function -

```
def Apriori(input_list, min_support, min_confidence):
```

Input:

- `input_list`: A list of lists, where each sublist represents a record containing different items.
- `min_support`: A float representing the minimum support threshold.
- `min_confidence`: A float representing the minimum confidence threshold.

Sample Input

```
input_list = [
    ['A', 'B', 'C'],
    ['D', 'E'],
    ['A', 'D', 'F'],
    ['E', 'F'],
    ['B', 'D', 'E'],
    ['A', 'D'],
    ['A', 'B', 'E', 'D'],
    ['C', 'F'],
    ['A', 'B', 'D'],
    ['D', 'E']
]

min_support = 0.2

min_confidence = 0.6
```

Expected Output

```
A -> D
B -> A
A -> B
E -> D
B -> D
B, D -> A
B, A -> D
B, E -> D
B, D -> E
```

The output should be in the following format:
`word1, word2, ... -> word1, word2, ...`
The order of the rules does not matter.

Explanation:

For the rule `A` $\rightarrow$ `D`, the calculations are as follows:

1. **Total Records:** There are 10 Records in the dataset.
2. **Support Calculation:**
    - **Count of records containing both `A` and `D`:**

- Record 3: `[A, D, F]`
- Record 6: `[A, D]`
- Record 7: `[A, B, E, D]`
- Record 9: `[A, B, D]`
- **Total count = 4**

- **Support** $= \frac{4}{10} = 0.4$

3. **Confidence Calculation:**

   - **Count of records containing `A`:**
     - Record 1: `[A, B, C]`
     - Record 3: `[A, D, F]`
     - Record 6: `[A, D]`
     - Record 7: `[A, B, E, D]`
     - Record 9: `[A, B, D]`
     - **Total count = 5**
   - **Confidence** $= \frac{4}{5} = 0.8$

Thus, the rule `A` $\rightarrow$ `D` has a support of 0.4 and a confidence of 0.8. Similarly, the same calculations can be applied to compute the support and confidence for the remaining rules.

2 Analyzing Clustering Techniques[70 Marks]

Clustering is a fundamental technique in machine learning and data analysis, used to group data points based on similarity. Two widely used clustering algorithms are K-Means and DBSCAN, each with unique strengths:

- K-Means Clustering: A centroid-based method that partitions data into a predefined number of clusters (K). It iteratively adjusts cluster centers to minimize intra-cluster variance.
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): A density-based algorithm that groups points that are closely packed together while treating outliers as noise. It relies on parameters ε (neighborhood radius) and `min_samples` (minimum points to form a cluster).

This assignment explores the effectiveness of these clustering techniques in identifying letter shapes formed on a Cartesian plane. You need to manually apply both algorithms and show detailed calculations for each step.

2.1 Task [35 Marks]

1. **Letter Representation in a Cartesian Plane:**
   - Consider a 2D Cartesian coordinate system.
   - Select the first three letters of your name.
   - For each letter, select 7 points that, when connected, outline the letter shape.
   - Maintain uniform dimensions:
     - **Letter width:** 10 units
     - **Letter height:** 15 units
     - **Distance between letters:** 10 units
2. **Plotting Points:**
   - Arrange the selected points on the Cartesian plane according to the fixed dimensions.
   - Label each point with its corresponding letter for visualization.
3. **Manual Clustering Analysis**
   - Manually apply K-Means clustering with an appropriate number of clusters
     - Choose an initial set of cluster centroids.
     - Assign each point to the nearest centroid.
     - Recalculate cluster centroids based on the assigned points.
     - Iterate until centroids stabilize.
     - Show all calculations, including distance computations and centroid updates.
   - Manually apply DBSCAN clustering, tuning ε and `min_samples` appropriately:
     - Compute the neighborhood of each point based on ε.
     - Identify core points, border points, and noise points.
     - Expand clusters from core points.
     - Show step-by-step calculations and justifications for parameter selection.
   - Analyze the clustering results:
     - Determine whether points belonging to the same letter form distinct clusters.
     - Compare and contrast the results obtained from K-Means and DBSCAN.

2.2 Conclusion [35 Marks]

After performing the clustering analysis, discuss the effectiveness of each algorithm. Consider the following aspects:

- How well each method separates letter clusters.
- Strengths and limitations of K-Means and DBSCAN in this context.
- Potential improvements for better clustering accuracy.