Marmara University

CSE4062 Spring Group Number: 3

Delivery #3 DESCRIPTIVE ANALYSIS

Instructor: Doç. Dr. MURAT CAN GANİZ

CONTRIBUTORS:

Şeyma Kaya - Electrical and Electronics Engineering
150719013 – skaya19@marun.edu.tr

Ataberk Gedik - Electrical and Electronics Engineering -
150719058 - ataberkgedik@marun.edu.tr

Muhammet Eren Atala - Computer Engineering - 150119904
- muhammeterenatala@gmail.com

Emrullah Sevmiş - Computer Engineering - 150119016 –
emrullahsevmis@marun.edu.tr

Muhammet Eren Ünlü - Computer Engineering - 150117027
- muhammet.eren@marun.edu.tr

Can Bayrakdar -Industrial Engineering - 150319569 -
canbayrakdar@gmail.com

Introduction:

This study focuses on conducting a data science and analytics project on the JIRA system. JIRA is a widely used project management and tracking tool. The objective of this study is to utilize the data available in JIRA and perform various analyses to gain valuable insights.

Data Set:

The data set we will be using contains information from different projects and tasks within the JIRA system. Some important columns in the data set include:

- **ID:** The unique identifier of the task or project.

- **PRIORITY:** The priority level of the task.

- **COMPONENT:** The component to which the task belongs.

- **URGENCY:** The urgency level of the task.

- **IMPACT:** The impact level of the task.

- **ISSUE_CATEGORY:** The category of the task.

- **WORKLOG_ASSIGNEE1-5:** The individuals associated with the task in the worklogs.

- **POZISYON:** The position of the individual.

- **IS_BILGISI:** The knowledge level of the individual.

- **CREATED:** The creation date of the task.

- **RESOLUTIONDATE:** The date when the task was resolved.

- **Completion Time:** The time taken to complete the task.

- **ASSIGNEE, ASSIGNEE1-5:** The individuals assigned to the task.

- **Total_Assignee:** The total number of assigned individuals.

- **Total_Worklog_Assginee:** The total number of individuals with worklogs.

- **Total_Log_Hours_Assignee:** The total worklog hours per individual.

- **COMMENTOR_COUNT:** The number of comments on the task.

- **COMMENT_COUNT:** The total number of comments on the task.

- **ISSUE_TYPE:** The type of the task.

Data Preprocessing

As a result of preprocessing in our dataset, the number of columns decreased to 12.

1. Import necessary libraries: The code begins by importing the required libraries, including pandas, numpy, LabelEncoder from scikit-learn, and KNNImputer from scikit-learn's preprocessing module.

2. Define the `prepare_data` function: This function takes the input data as a parameter and performs various data preprocessing operations.

3. Select columns of interest: The code defines a list of columns to use for preprocessing. These columns will be selected from the input data for further processing.

4. Copy the data: A copy of the original data is made using the `copy()` method. This ensures that modifications made to the data do not affect the original dataset.

5. Outlier analysis: The code conducts outlier analysis on specific numerical columns using the Interquartile Range (IQR) method. Data points outside the acceptable range (1.5 times the IQR) are removed from the dataset.

6. Convert categorical columns to string type: Before encoding categorical columns, the code converts them to the string data type. This step ensures consistent handling of categorical variables during the encoding process.

7. Print count of unique values: The code prints the count of unique values in each column of the preprocessed dataset. This information helps in understanding the distribution and characteristics of the data.

8. Remove rows with 'nan' values in ISSUE_CATEGORY column: Rows that contain 'nan' values in the ISSUE_CATEGORY column are removed from the dataset.

9. Replace 'nan' strings with 'unknown': The code replaces 'nan' strings with 'unknown' in the URGENCY, IMPACT, and COMPONENT columns.

10. Replace empty strings in COMPONENT column with mode: Empty strings in the COMPONENT column are replaced with the mode (most frequently occurring value) of the non-empty values in that column.

11. Aggregate categories with low counts: Categories in the COMPONENT, ISSUE_CATEGORY, and ASSIGNEE columns that have a count less than 100 are aggregated into a single category named 'OTHER'.

12. Convert URGENCY and PRIORITY columns to ordinal values: The URGENCY and PRIORITY columns are mapped to ordinal values. For example, 'Low' is mapped to 0, 'Medium' to 1, and 'High' to 2.

13. Impute missing values in URGENCY column using KNN imputer: Missing values in the URGENCY column are imputed using the KNNImputer. The KNNImputer replaces missing values with estimates based on the values of the nearest neighbors.

14. Impute missing values in PRIORITY column using mode: Missing values in the PRIORITY column are imputed with the mode (most frequently occurring value) of the non-missing values in that column.

15. Compute weighted mean for Total_Log_Hours_Assignee: The code computes the weighted mean for the Total_Log_Hours_Assignee column. Values in the column that have a frequency below 100 are replaced with the weighted mean.

16. Print count of unique values again: The code prints the count of unique values in each column of the preprocessed dataset after performing the above steps.

17. Label encoding: Categorical columns (COMPONENT, IMPACT, ISSUE_CATEGORY, ASSIGNEE, ISSUE_TYPE) are encoded using LabelEncoder from scikit-learn. Label encoding assigns a unique numeric label to each distinct category in a column.

18. Save preprocessed data to a file: The preprocessed data is saved to an XLSX file named 'preprocessed_data.xlsx' using the `to_excel` method.

Descriptive Analysis of Our JIRA Dataset Using Clustering and Association Rule Mining Algorithms (ARM)

Clustering Algorithms

Despite our best efforts to discover patterns through clustering methods, our initial attempts did not yield satisfactory results. One potential reason for this challenge was the multidimensional nature of our dataset, which consisted of twelve distinct columns. This high dimensionality can often lead to a phenomenon known as the "curse of dimensionality", where the sparsity of high-dimensional space can make it difficult for clustering algorithms to find meaningful relationships between data points.

To address this issue, we employed Principal Component Analysis (PCA), a popular technique for dimensionality reduction. PCA transforms the original set of variables into a new set of uncorrelated variables known as principal components. These principal components are ordered by the amount of original variance they capture, allowing us to focus on the most significant features of the data while discarding the less important ones. Despite these efforts, meaningful patterns remained elusive within our cluster analysis results.

Having encountered these difficulties with clustering, we turned our attention to frequent itemset mining and association rule learning methods. These techniques, including FP-Growth and Apriori algorithms, aim to uncover relationships between variables that frequently co-occur within our dataset. These methods allowed us to derive association rules that provide valuable insights into the interplay between different variables in our data.

1) K Means

Our K-means clustering algorithm is designed to group a given data set into distinct clusters. This is done based on similarities in the data, with the intent of making sure that data points in the same cluster are more similar to each other than to those in other clusters. The number of clusters is chosen to minimize within-cluster variance.

The steps involved in our algorithm are:

1. Load Data: The algorithm begins by loading the preprocessed data from an Excel file. The data must be preprocessed to ensure that it's suitable for clustering. This could involve cleaning, normalization, encoding, etc.

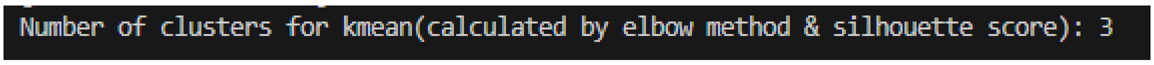
2. PCA (Principal Component Analysis): This is a technique used for dimensionality reduction in the dataset. It transforms the data into a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. In our algorithm, you've chosen two components, which means you're reducing the dimensionality of our data to two dimensions. This makes it easier to plot and visualize.

3. Elbow Method and Silhouette Score: These are used to find the optimal number of clusters for K-means. The elbow method looks at the total within-cluster sum of square (wcss) as a function of the number of clusters, and picks the 'elbow' of the curve as the number of clusters to use. The silhouette score measures the quality of the clustering by calculating how close each point in one cluster is to the points in the neighboring clusters. This is done for different numbers of clusters.

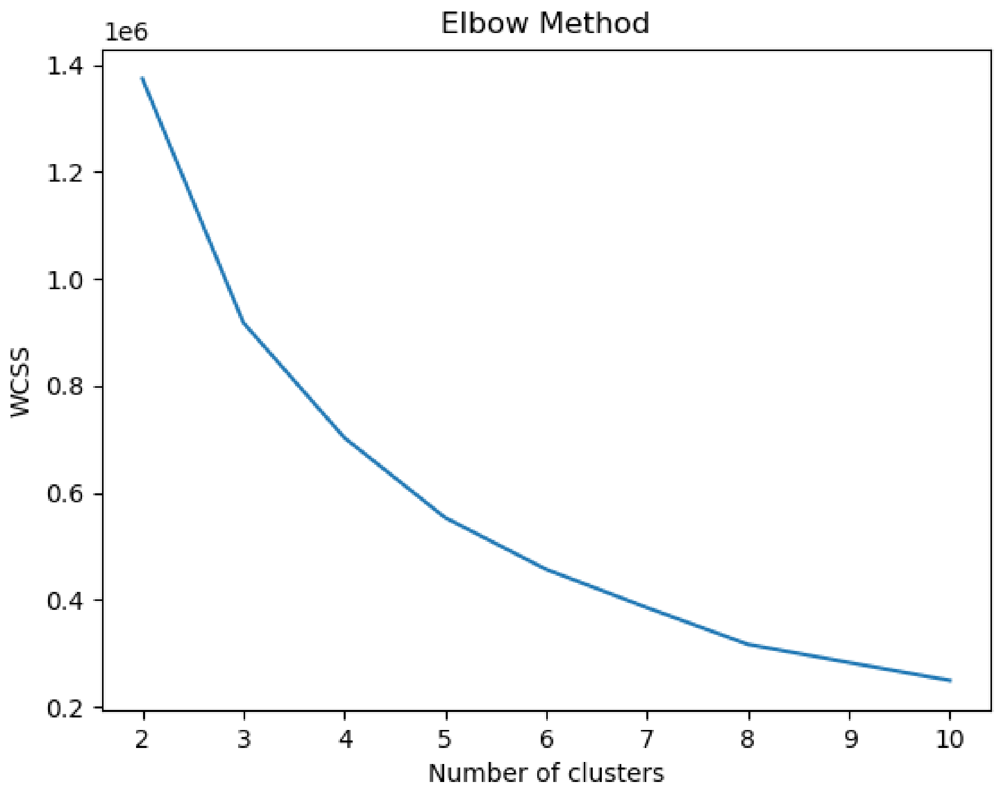
4. K-means Clustering: After determining the optimal number of clusters, you perform K-means clustering. The algorithm assigns each data point to the cluster whose centroid is nearest. The 'fit_predict' method computes cluster centers and predicts the cluster index for each sample.

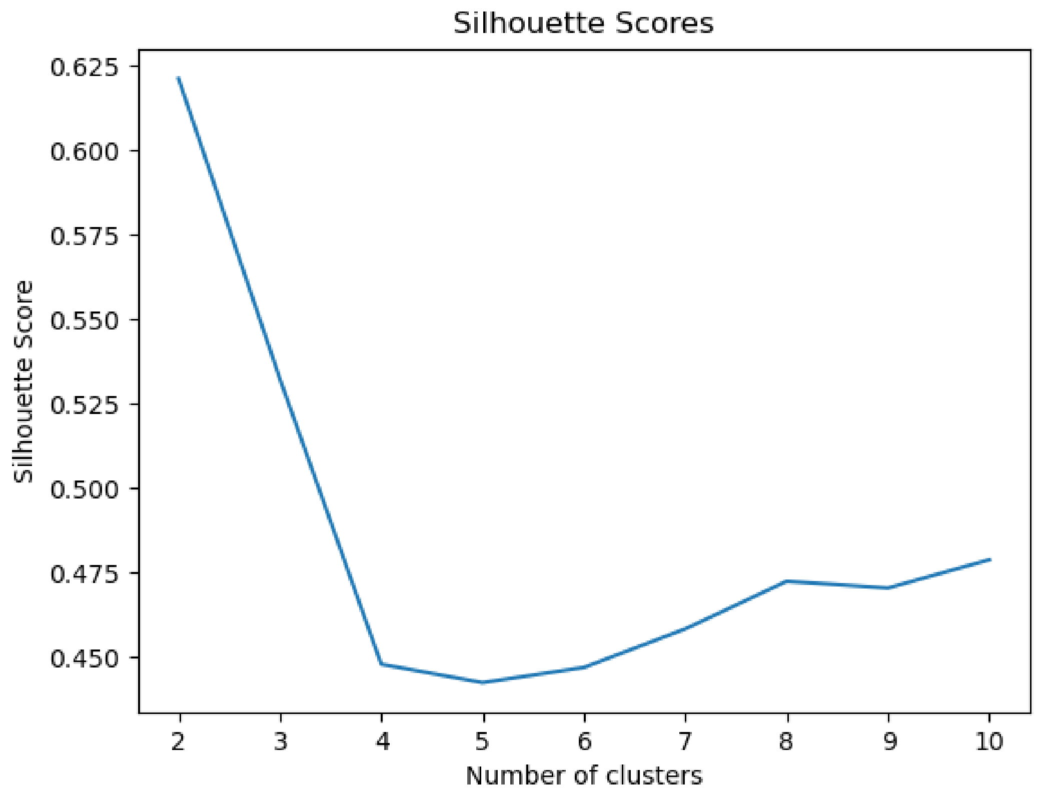
5. Plotting the Clusters: Finally, you create a scatter plot of the first two principal components, with the data points colored according to their cluster label. This helps in visualizing the distinct clusters.

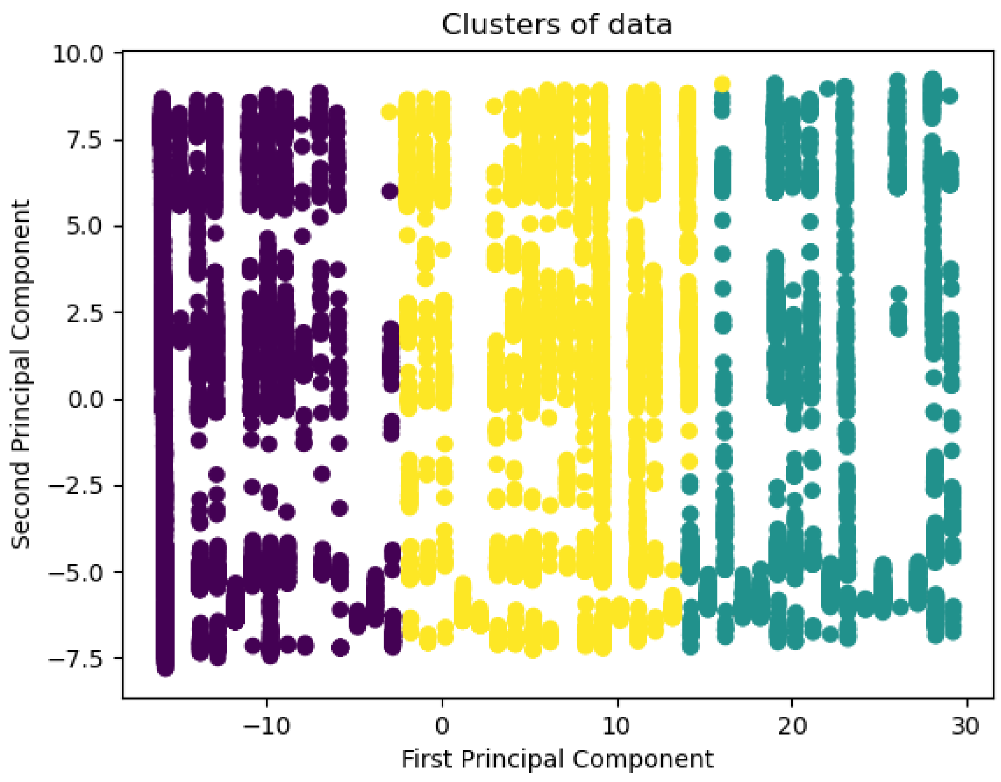
The output of this algorithm is a set of clusters, each containing data points that are similar to each other based on the given data set. It also provides visualizations of the Elbow method, Silhouette scores, and the final clusters. These visualizations are saved in a figures directory for further use or inspection.



```
Number of clusters for kmean(calculated by elbow method & silhouette score): 3
```







2) Hierarchical Clustering

Our hierarchical clustering algorithm follows these steps to group a given data set into distinct clusters:

1. Load Data: The algorithm starts by loading preprocessed data from an Excel file. Preprocessing is a crucial step that ensures the data is cleaned and formatted correctly, enabling optimal performance of the clustering algorithm.

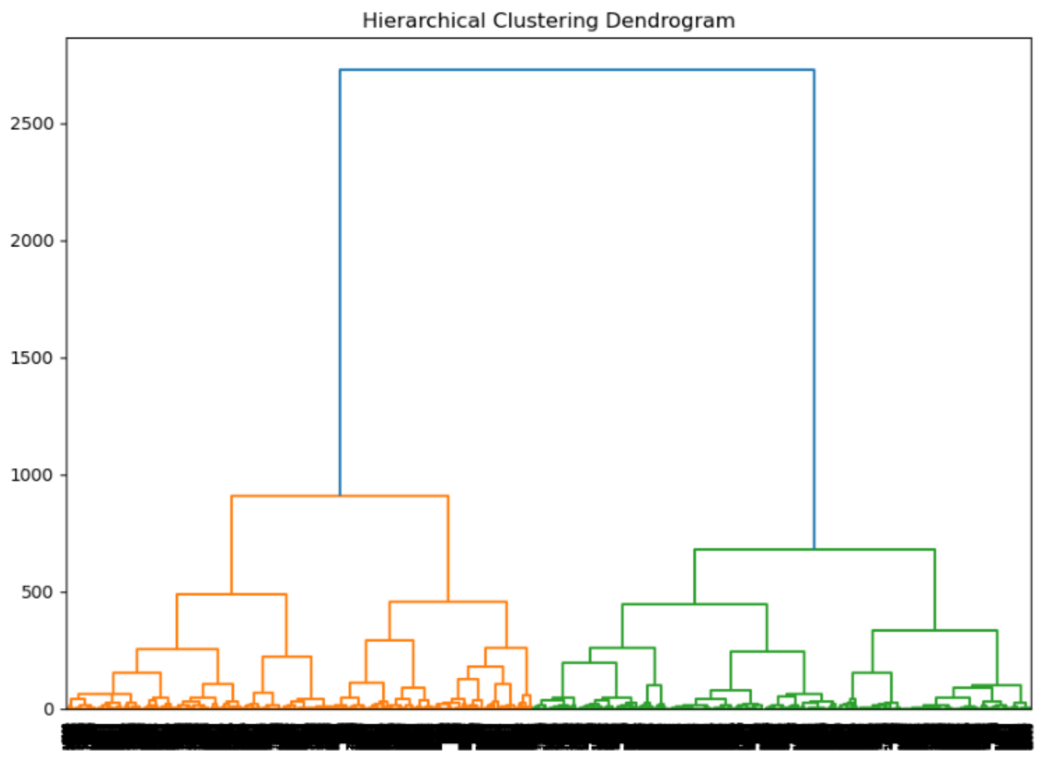
2. PCA (Principal Component Analysis): After loading the data, our algorithm uses PCA for dimensionality reduction. PCA transforms a high-dimensional data set into a lower-dimensional one, while preserving as much of the original variance as possible. In this case, you've selected two principal components, reducing our data to two dimensions. This also makes it possible to visualize the data and the results of the clustering algorithm.

3. Linkage Matrix: The linkage matrix is a key part of hierarchical clustering. It is a table that records the pairings of data points and clusters during the execution of the clustering algorithm. You've chosen to use Ward's method, which aims to minimize the variance in each of the clusters.

4. Dendrogram: A dendrogram is a tree-like diagram that displays the sequences of merges or splits. The 'leaves' of the tree are the individual data points, and each internal node represents a cluster. The height of each internal node indicates the distance (or dissimilarity) between the two clusters that were merged.

5. Plotting the Dendrogram: Finally, our algorithm plots the dendrogram. This visualization gives a big-picture view of the clustering process, allowing you to see how the clusters are formed at each step, and how similar (or dissimilar) the clusters are.

This algorithm results in a hierarchical clustering of the data, represented visually by a dendrogram. Unlike K-means clustering, hierarchical clustering does not require the number of clusters to be specified in advance, and it can suggest an appropriate number of clusters by inspecting the dendrogram. The output dendrogram is saved in a figures directory for future use or reference.



3)DBSCAN

Our Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm carries out the following steps to categorize our dataset into different clusters:

1.Load Data: The algorithm initiates by loading the preprocessed data from an Excel file. Preprocessing ensures the data is suitable for clustering - it may include cleaning, normalization, encoding, etc.

2. Nearest Neighbors Computation: In this step, you compute the distances to the nearest neighbors of each data point. You're using `NearestNeighbors` from sklearn with n_neighbors set to 2. This means for each point, you are computing the distance to its two nearest neighbors.

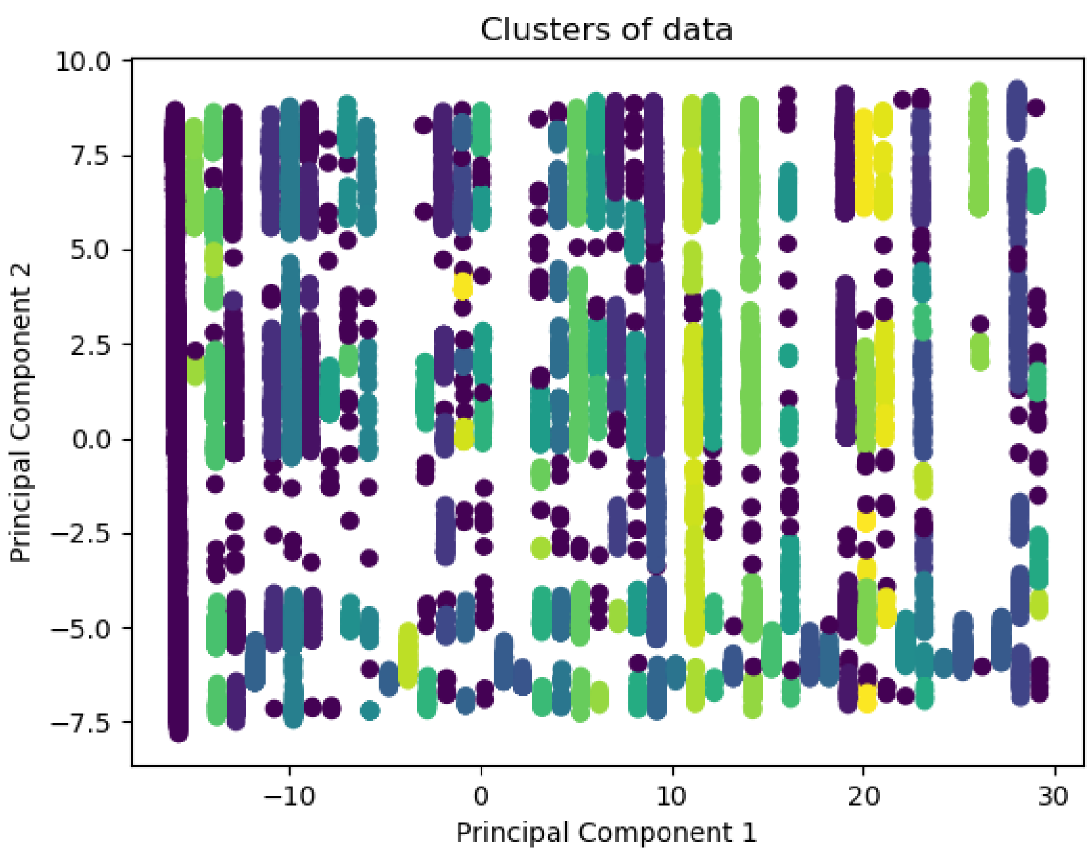
3. K-distance Graph: The distances obtained in the previous step are sorted and plotted. This graph assists in determining the optimal epsilon parameter for DBSCAN. Epsilon is a key parameter for DBSCAN that specifies the maximum distance between two samples for them to be considered as in the same neighborhood.

4. Epsilon Determination: Epsilon is determined by finding the 'knee' or 'elbow' point in the K-distance graph, which should ideally be where a sharp change occurs. In our case, epsilon is assumed to be 0.3.

5. DBSCAN Clustering: After choosing the epsilon value, DBSCAN clustering is carried out. Here, the `DBSCAN` function is initialized with the chosen epsilon value and minimum number of samples in a cluster, which is 5 in this case. The 'fit_predict' function is then used to assign each data point to a cluster.

6. Plotting the Clusters: Finally, if the data is 2-dimensional, you visualize the different clusters by creating a scatter plot. The points are colored according to their assigned clusters.

The DBSCAN clustering algorithm provides clusters based on data density. It's different from K-means and Hierarchical clustering as it doesn't require the number of clusters to be pre-defined, and it can discover clusters of arbitrary shape, as opposed to the spherical clusters that K-means usually finds.



Association Rule Mining Algorithms (ARM)

1) Apriori Algorithm

The Apriori algorithm is a fundamental method used for frequent itemset mining (FIM), which is a significant aspect of association rule mining in data science. Here is how the provided `apriori_algorithm` function works:

1. Directory Setup: Initially, the function checks if a directory named "figures" exists. If not, it creates one.

2. Data Load and Preparation: The function reads an Excel file named "preprocessed_data.xlsx" to load the preprocessed data. Each row in the data is treated as a transaction, and transactions are created as a list of column names where the cell value is True.

3. Transaction Encoding: The TransactionEncoder from the mlxtend library is used to transform the transactions into a format compatible with the Apriori algorithm. The raw dataset is converted into an array format that can be processed by the algorithm.

4. Apriori Algorithm Application: The Apriori algorithm is applied to the processed data using the `apriori` function from the mlxtend library. The minimum support threshold is set to 0.6, indicating that an itemset must occur in at least 60% of transactions to be deemed frequent.

5. Frequent Itemsets Extraction: The function extracts frequent itemsets from the data. These itemsets appear often in the dataset and are likely to occur together in transactions.

6. Result Sorting and Saving: The frequent itemsets are sorted in descending order based on their support values. This support value indicates the frequency of occurrence of an itemset in the dataset. The results are then saved to a JSON file "frequent_itemsets_apriori.json" for future reference.

7. Result Displaying: The results are displayed for immediate observation. The maximum column width is set to a large number to avoid truncation of the printed output.

Some Extracted Patterns

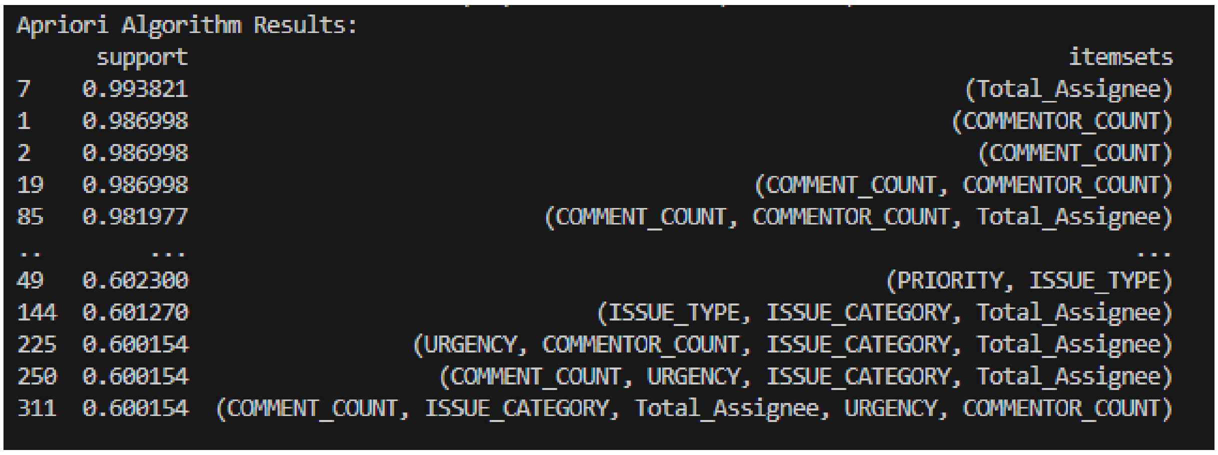
1. Highly Frequent Items: Similar to the FP-Growth results, 'Total_Assignee', 'COMMENT_COUNT', and 'COMMENTOR_COUNT' appear most frequently across transactions. This suggests that issues often have an assignee and involve comments and commentors.

2. Commonly Paired Items: 'COMMENT_COUNT' and 'COMMENTOR_COUNT' are frequently found together. This relationship suggests that the number of comments and the number of unique commentors are often directly proportional.

3. Interactions with 'Total_Assignee': There's a strong association between 'COMMENTOR_COUNT' and 'Total_Assignee', implying that the number of unique commentors is possibly related to the total number of assignees.

4. Complex Relationships: Some frequent itemsets contain more than two items, such as ('COMMENT_COUNT', 'ISSUE_CATEGORY', 'Total_Assignee', 'URGENCY', 'COMMENTOR_COUNT'). These show complex associations and can lead to interesting insights.

5. Other Patterns: Certain items appear more frequently with other specific items. For instance, 'PRIORITY' and 'ISSUE_TYPE' often appear together. Similarly, 'ISSUE_TYPE', 'ISSUE_CATEGORY', and 'Total_Assignee' are frequently grouped together. These patterns could provide valuable insights into the relationships among these variables.


```
Apriori Algorithm Results:
      support                                                            itemsets
7    0.993821                                                      (Total_Assignee)
1    0.986998                                                     (COMMENTOR_COUNT)
2    0.986998                                                       (COMMENT_COUNT)
19   0.986998                                      (COMMENT_COUNT, COMMENTOR_COUNT)
85   0.981977                      (COMMENT_COUNT, COMMENTOR_COUNT, Total_Assignee)
..        ...                                                                   ...
49   0.602300                                                (PRIORITY, ISSUE_TYPE)
144  0.601270                          (ISSUE_TYPE, ISSUE_CATEGORY, Total_Assignee)
225  0.600154           (URGENCY, COMMENTOR_COUNT, ISSUE_CATEGORY, Total_Assignee)
250  0.600154             (COMMENT_COUNT, URGENCY, ISSUE_CATEGORY, Total_Assignee)
311  0.600154  (COMMENT_COUNT, ISSUE_CATEGORY, Total_Assignee, URGENCY, COMMENTOR_COUNT)
```

2) FP Growth

The FP-Growth algorithm is a popular method for frequent itemset mining (FIM), a key part of association rule mining in data science. Here's how our algorithm works:

1. Load Data: The algorithm loads preprocessed data from an Excel file. Each row in the data is considered a transaction.

2. Transactions Creation: In this step, the data is prepared for processing. Transactions are created as a list of column names where the cell value is True.

3. Transaction Encoding: The data is then encoded into a format suitable for the FP-Growth algorithm. The `TransactionEncoder` object from the `mlxtend` library is used for this purpose. This process transforms the dataset into an array format that the FP-Growth algorithm can work with.

4. FP-Growth Application: The FP-Growth algorithm is then applied to the data. The `fpgrowth` function is called on the encoded DataFrame. Here, the `min_support` parameter is set to 0.6, meaning that an itemset must be present in at least 60% of transactions to be considered frequent.

5. Frequent Itemsets Extraction: The algorithm extracts frequent itemsets, which are itemsets that occur together frequently in the dataset.

6. Sorting Frequent Itemsets: The frequent itemsets are then sorted in descending order by their support value. The support value indicates how frequently an itemset appears in the dataset.

7. Saving and Displaying Results: The results are saved in a JSON file for future reference and printed for immediate viewing.

Some Extracted Patterns

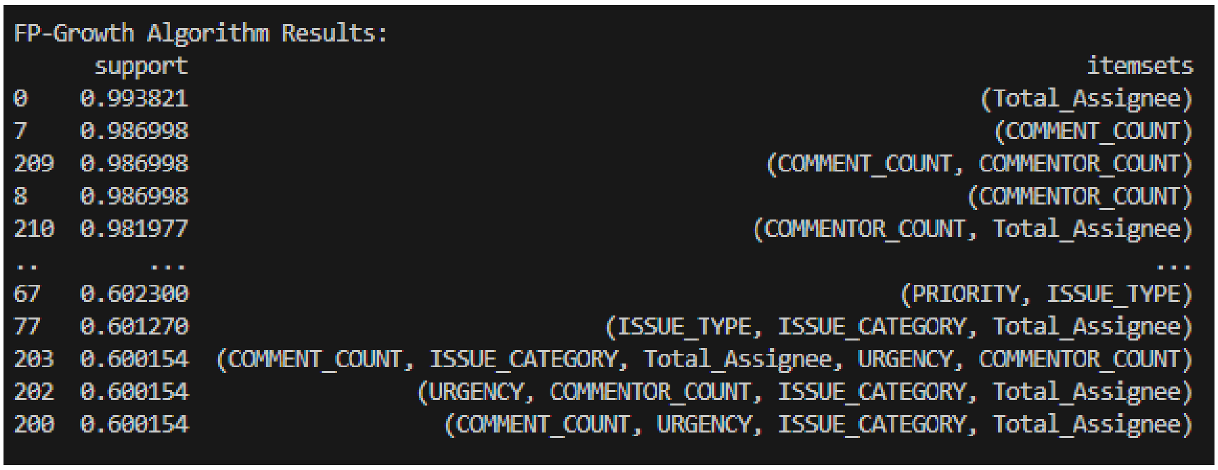
1. Highly Frequent Items: 'Total_Assignee', 'COMMENT_COUNT', and 'COMMENTOR_COUNT' are the most frequent items in the dataset, appearing in almost all transactions. This indicates that most issues have a 'Total_Assignee' and involve some comments and commentors.

2. Commonly Paired Items: The pair 'COMMENT_COUNT' and 'COMMENTOR_COUNT' frequently occur together. This implies a strong association between these two variables, indicating that when the number of comments is high, the number of unique commentors is also likely to be high, and vice versa.

3. Interactions with 'Total_Assignee': 'COMMENTOR_COUNT' and 'Total_Assignee' often appear together, suggesting a relationship between these two items. It might indicate that the total number of assignees is related to the number of unique commentors.

4. Complex Relationships: There are frequent itemsets that contain more than two items, such as ('COMMENT_COUNT', 'ISSUE_CATEGORY', 'Total_Assignee', 'URGENCY', 'COMMENTOR_COUNT'). These imply more complex relationships and can be particularly interesting for further analysis.

5. Other Patterns: Certain items are more likely to appear with specific other items, such as 'PRIORITY' with 'ISSUE_TYPE' and 'ISSUE_TYPE' with 'ISSUE_CATEGORY' and 'Total_Assignee'. These can be further investigated for more detailed insights.



```
FP-Growth Algorithm Results:
      support                                                          itemsets
0    0.993821                                                  (Total_Assignee)
7    0.986998                                                   (COMMENT_COUNT)
209  0.986998                                  (COMMENT_COUNT, COMMENTOR_COUNT)
8    0.986998                                                 (COMMENTOR_COUNT)
210  0.981977                                 (COMMENTOR_COUNT, Total_Assignee)
..        ...                                                               ...
67   0.602300                                              (PRIORITY, ISSUE_TYPE)
77   0.601270                         (ISSUE_TYPE, ISSUE_CATEGORY, Total_Assignee)
203  0.600154  (COMMENT_COUNT, ISSUE_CATEGORY, Total_Assignee, URGENCY, COMMENTOR_COUNT)
202  0.600154               (URGENCY, COMMENTOR_COUNT, ISSUE_CATEGORY, Total_Assignee)
200  0.600154                 (COMMENT_COUNT, URGENCY, ISSUE_CATEGORY, Total_Assignee)
```