Apriori Algorithm Implementation

Python Code: Apriori Algorithm

```
import csv
import os
from itertools import chain, combinations

"""
-------------------------------- Apriori Algorithm ---------------------------------------
Description:
- Input:
    +) dataset: is entire available transactions
    +) sample: we randomly select several items available in the given dataset.
               The algorithm will rely on the domain of the sample to give corresponding
               association rules
    +) min_support
    +) min_confidence
- Output:
    +) It returns association rules with corresponding level of confidences
------------------------------------------------------------------------------------------
"""
def Apriori(dataset, sample, min_support, min_confidence):
    temp_antecedents = []
    temp_consequents = []
    antecedents = []
    consequents = []
    confidences = []

    for k in range(1, len(sample)):
        subsets = find_all_combinations(sample, k)
        for sub in subsets:
            antecedent = sub
            consequent = list(set(sample) - set(sub))

            antecedent_occurrences = count_occurrence_itemset(antecedent, dataset)
            consequent_occurrences = count_occurrence_itemset(consequent, dataset)

            if antecedent_occurrences >= min_support and consequent_occurrences >= min_support:
                temp_antecedents.append(antecedent)
                temp_consequents.append(consequent)

    sample_occurrences = count_occurrence_itemset(sample, dataset)
    for i in range(len(temp_antecedents)):
        confidence = sample_occurrences/count_occurrence_itemset(temp_antecedents[i], dataset)
        if confidence >= min_confidence:
            antecedents.append(temp_antecedents[i])
            consequents.append(temp_consequents[i])
            confidences.append(confidence)

    return antecedents, consequents, confidences



"""
------------------------------ count_occurrence_itemset ----------------------------------
Description:
- Input:
```

```
  +) itemset: a set of items
  +) dataset: is entire available transactions
- Output:
  +) It returns the number of occurrences of the item set (or the number of transactions
     that contain this item set)
-------------------------------------------------------------------------------------------
"""
def count_occurrence_itemset(itemset, dataset):
    counter = 0
    for row in dataset:
        if set(itemset).issubset(set(row)):
            counter = counter + 1
    return counter



"""
-------------------------------- find_all_combinations -----------------------------------
Description:
- This function returns a list of all possible k-combinations from a superset

-------------------------------------------------------------------------------------------
"""
def find_all_combinations(superset, k):
    return [list(comb) for comb in combinations(superset, k)]


"""
------------------------------- load_and_clean_data ---------------------------------------
Description:
- The input of function is the path of the csv file. Please ensure that the path of input
file must be RELATIVE path. Also, you must ensure that the csv file must be located in the
same folder of this code.
- This function reads the content in csv file row by row. For each row, it will remove
duplicate and empty elements.
- The output of function is the 2D-array.
-------------------------------------------------------------------------------------------
"""
def load_and_clean_data(relative_file_path):
    # Convert relative file path to absolute
    absolute_file_path = os.path.abspath(relative_file_path)

    # Now, we process the data row by row
    cleanned_data = []
    try:
        with open(file=absolute_file_path, mode='r', newline='') as file:
            raw_data = csv.reader(file)
            for row in raw_data:
                # Collect distinct elements
                row = list(dict.fromkeys(row))

                # Remove empty elements
                for element in row:
                    if element == '':
                        row.remove(element)

                # Convert string to numeric
                row = [int(item) for item in row]

                # Collect non-empty rows
                if len(row) > 0:
                    cleanned_data.append(row)
    except FileNotFoundError:
        print(f"Error: The file at '{absolute_file_path}' was not found.")
    except IOError:
        print(f"Error: An IO error occurred while trying to open '{absolute_file_path}'.")
    return cleanned_data



"""
```

```
126 ------------------------------------ export_to_CSV ----------------------------------------
127 Description:
128 - This function is used to write the data to csv file
129 - Please ensure that the path for destination file must be RELATIVE
130 -------------------------------------------------------------------------------------------
131 """
132 def export_to_CSV(data, destination_file_name):
133     # Determine the current directory of the script
134     current_directory = os.path.dirname(os.path.abspath(__file__))
135     file_path = os.path.join(current_directory, destination_file_name + '.csv')
136 
137     # Exporting the matrix to a CSV file
138     with open(file_path, mode='w', newline='') as file:
139         writer = csv.writer(file)
140         # Write each row of the matrix
141         writer.writerows(data)
142     print(f"Matrix has been exported to {file_path}")
```