AYDIN ADNAN MENDERES UNIVERSITY

ENGINEERING FACULTY
COMPUTER SCIENCE ENGINEERING DEPARTMENT

Association Rule Mining

CSE419 – Artificial Intelligance Association Rule Mining 2023/2024

Burak TÜZEL

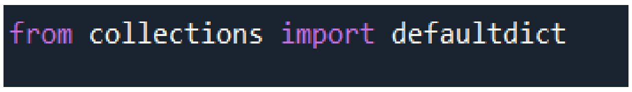
Lecturer:

Asst. Prof. Dr. Fatih SOYGAZİ

Association Rule Mining

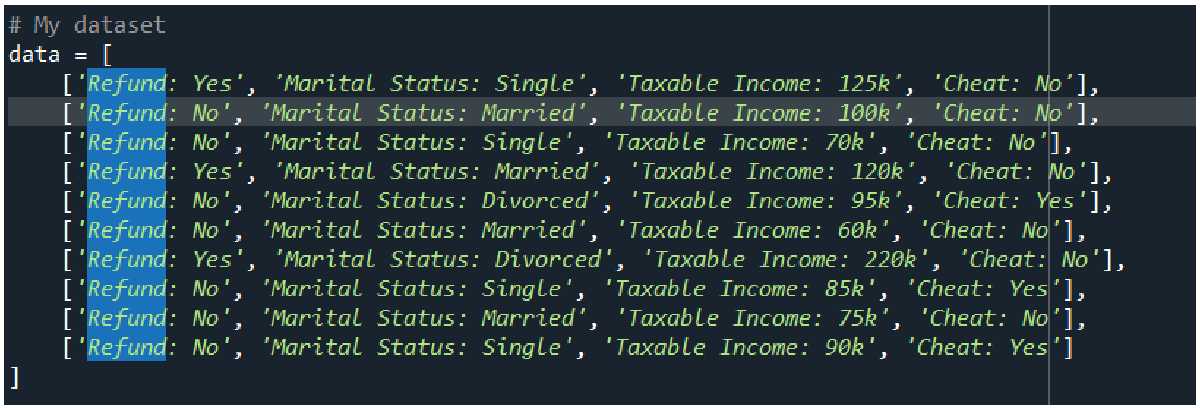
Installing and importing necessary libraries:

1- Import required libraries



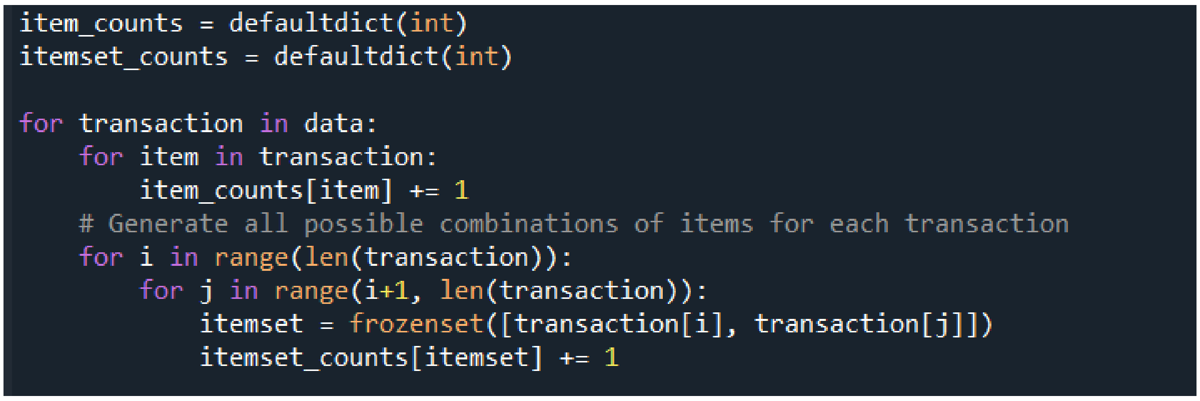
```
from collections import defaultdict
```

Defining related dataset:



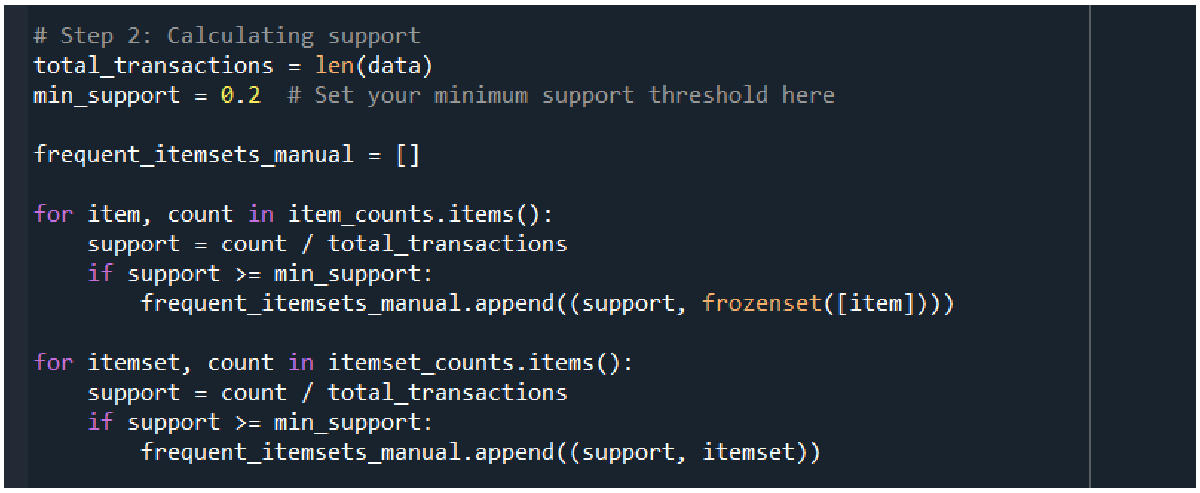
```
# My dataset
data = [
    ['Refund: Yes', 'Marital Status: Single', 'Taxable Income: 125k', 'Cheat: No'],
    ['Refund: No', 'Marital Status: Married', 'Taxable Income: 100k', 'Cheat: No'],
    ['Refund: No', 'Marital Status: Single', 'Taxable Income: 70k', 'Cheat: No'],
    ['Refund: Yes', 'Marital Status: Married', 'Taxable Income: 120k', 'Cheat: No'],
    ['Refund: No', 'Marital Status: Divorced', 'Taxable Income: 95k', 'Cheat: Yes'],
    ['Refund: No', 'Marital Status: Married', 'Taxable Income: 60k', 'Cheat: No'],
    ['Refund: Yes', 'Marital Status: Divorced', 'Taxable Income: 220k', 'Cheat: No'],
    ['Refund: No', 'Marital Status: Single', 'Taxable Income: 85k', 'Cheat: Yes'],
    ['Refund: No', 'Marital Status: Married', 'Taxable Income: 75k', 'Cheat: No'],
    ['Refund: No', 'Marital Status: Single', 'Taxable Income: 90k', 'Cheat: Yes']
]
```

Generating all possible Combinations of items for each row:



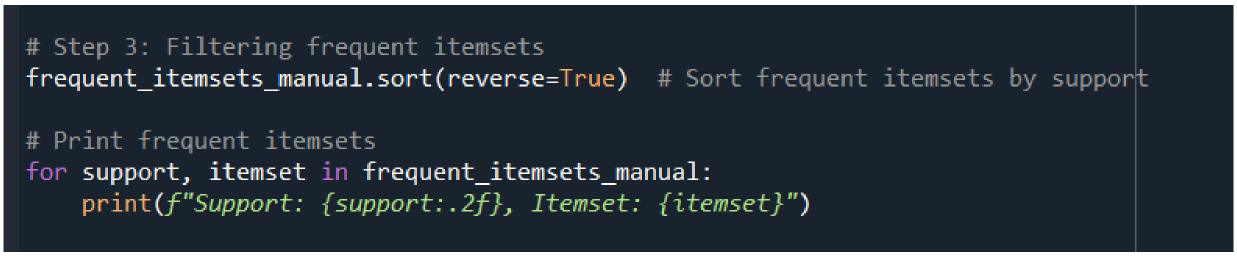
```
item_counts = defaultdict(int)
itemset_counts = defaultdict(int)

for transaction in data:
    for item in transaction:
        item_counts[item] += 1
    # Generate all possible combinations of items for each transaction
    for i in range(len(transaction)):
        for j in range(i+1, len(transaction)):
            itemset = frozenset([transaction[i], transaction[j]])
            itemset_counts[itemset] += 1
```

Calculating Support for each generated item:



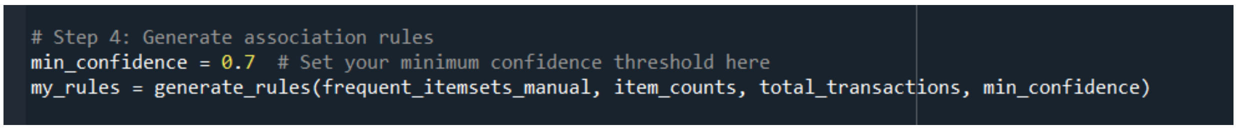
```
# Step 2: Calculating support
total_transactions = len(data)
min_support = 0.2  # Set your minimum support threshold here

frequent_itemsets_manual = []

for item, count in item_counts.items():
    support = count / total_transactions
    if support >= min_support:
        frequent_itemsets_manual.append((support, frozenset([item])))

for itemset, count in itemset_counts.items():
    support = count / total_transactions
    if support >= min_support:
        frequent_itemsets_manual.append((support, itemset))
```

Sorting Frequent itemsets by their support values:



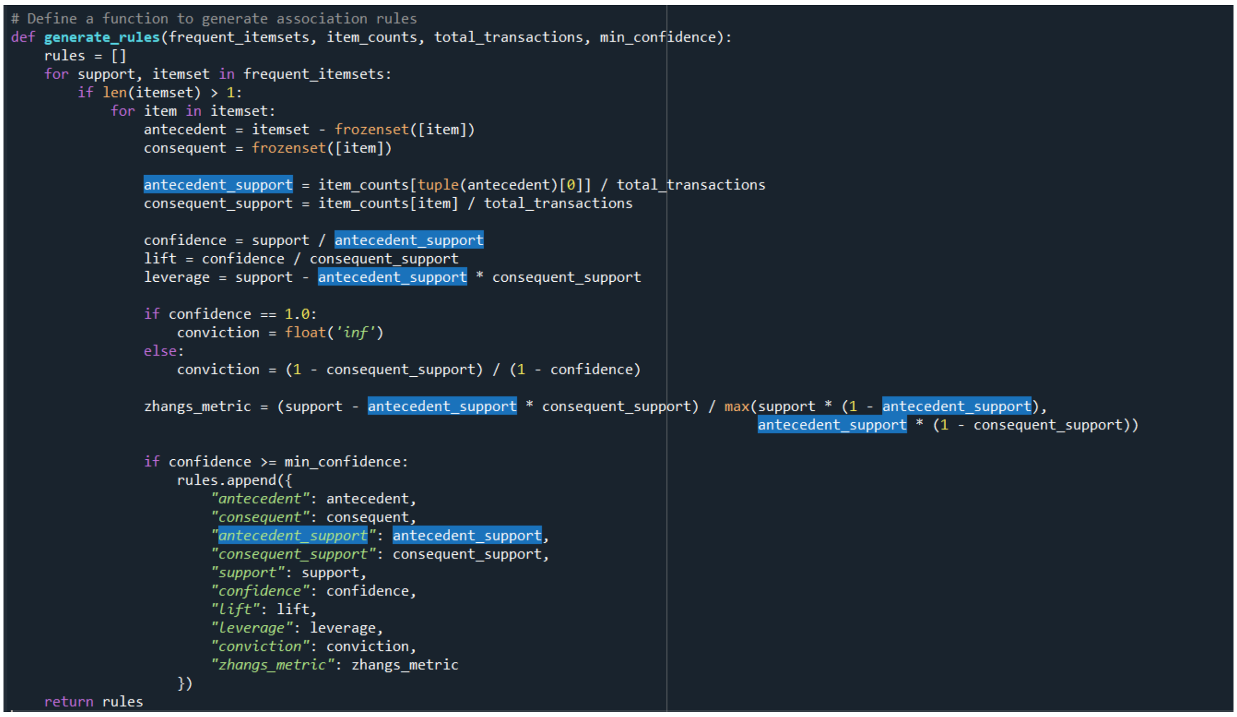
```python
# Step 3: Filtering frequent itemsets
frequent_itemsets_manual.sort(reverse=True)  # Sort frequent itemsets by support

# Print frequent itemsets
for support, itemset in frequent_itemsets_manual:
    print(f"Support: {support:.2f}, Itemset: {itemset}")
```

Generating Association Rules in generate_rules function:



```python
# Step 4: Generate association rules
min_confidence = 0.7  # Set your minimum confidence threshold here
my_rules = generate_rules(frequent_itemsets_manual, item_counts, total_transactions, min_confidence)
```

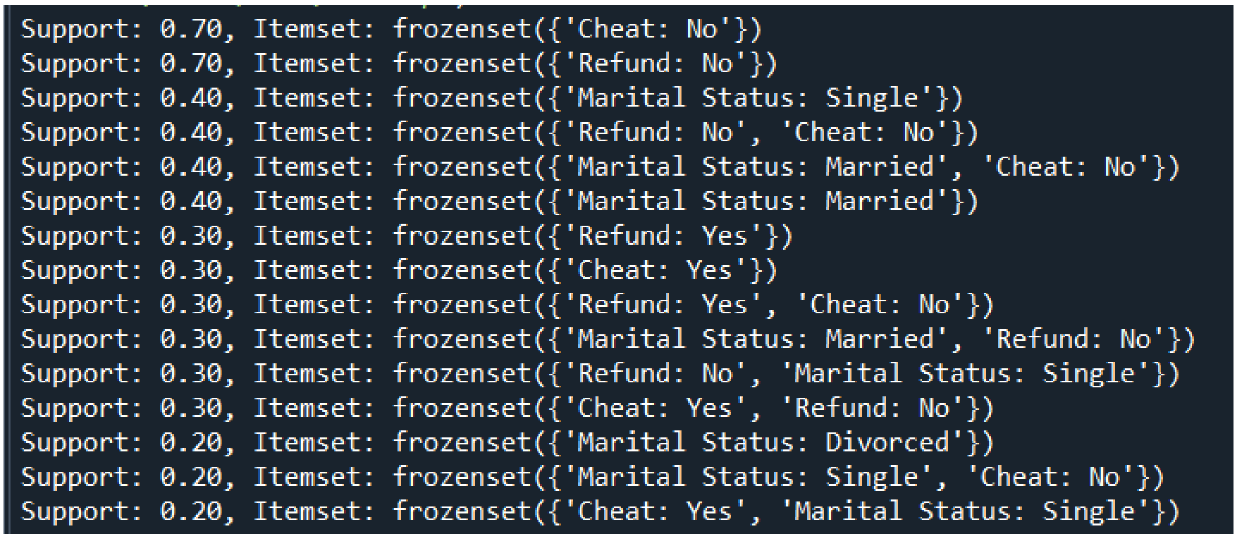
Generate_rules function:



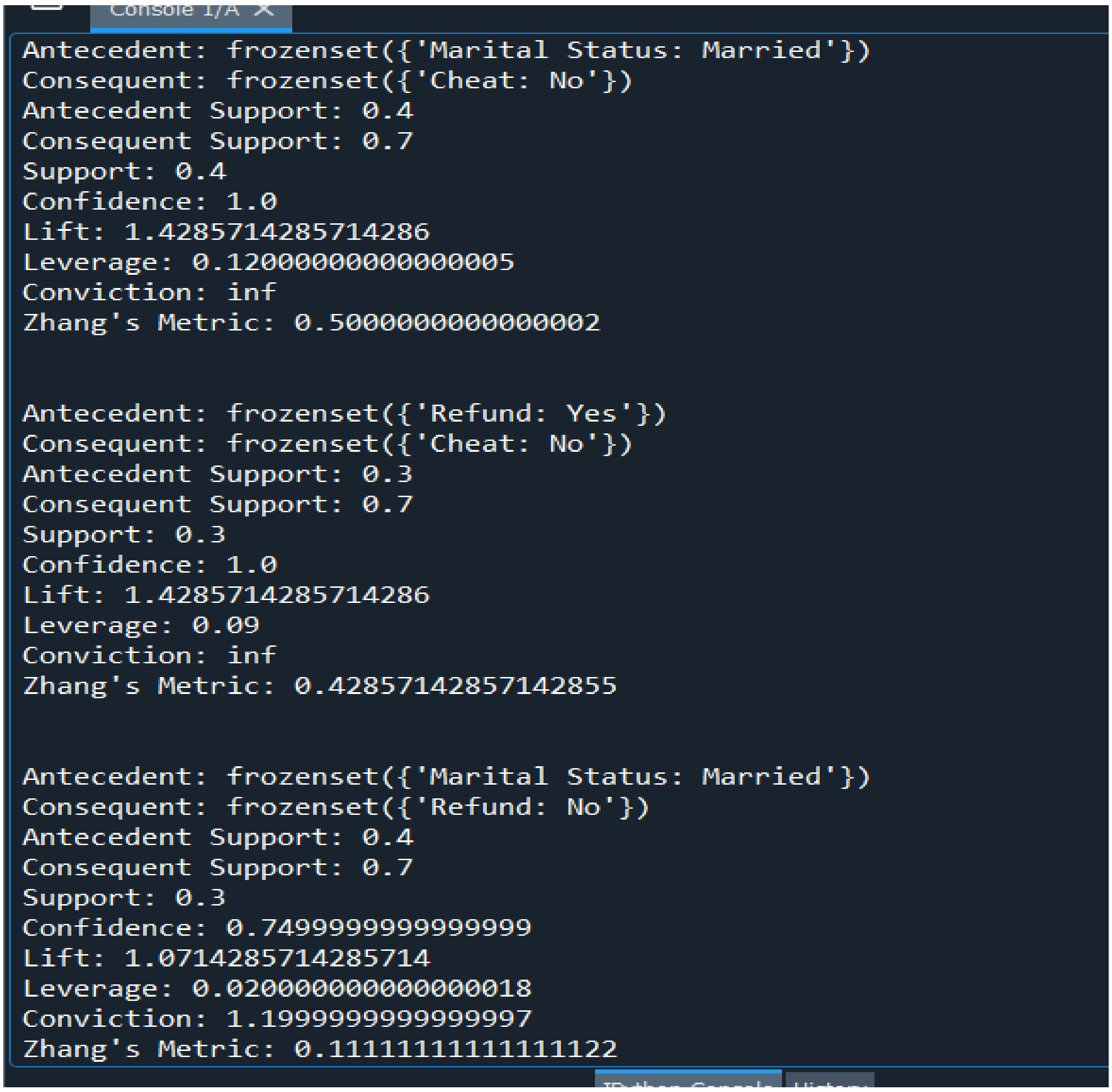
```python
# Define a function to generate association rules
def generate_rules(frequent_itemsets, item_counts, total_transactions, min_confidence):
    rules = []
    for support, itemset in frequent_itemsets:
        if len(itemset) > 1:
            for item in itemset:
                antecedent = itemset - frozenset([item])
                consequent = frozenset([item])

                antecedent_support = item_counts[tuple(antecedent)[0]] / total_transactions
                consequent_support = item_counts[item] / total_transactions

                confidence = support / antecedent_support
                lift = confidence / consequent_support
                leverage = support - antecedent_support * consequent_support

                if confidence == 1.0:
                    conviction = float('inf')
                else:
                    conviction = (1 - consequent_support) / (1 - confidence)

                zhangs_metric = (support - antecedent_support * consequent_support) / max(support * (1 - antecedent_support),
                                                                                          antecedent_support * (1 - consequent_support))

                if confidence >= min_confidence:
                    rules.append({
                        "antecedent": antecedent,
                        "consequent": consequent,
                        "antecedent_support": antecedent_support,
                        "consequent_support": consequent_support,
                        "support": support,
                        "confidence": confidence,
                        "lift": lift,
                        "leverage": leverage,
                        "conviction": conviction,
                        "zhangs_metric": zhangs_metric
                    })
    return rules
```

Generate_rules function take frequent_itemsets, item_counts, total_transactions, and min_confidence to calculate rules for each frequent itemset. It’s done by finding ancedent_support, consequent_support, confidence, lift, leverage, conviction, and zhangs_metric. These metrics defines the frequent itemset’s rule parameters.

The Results:



```
Support: 0.70, Itemset: frozenset({'Cheat: No'})
Support: 0.70, Itemset: frozenset({'Refund: No'})
Support: 0.40, Itemset: frozenset({'Marital Status: Single'})
Support: 0.40, Itemset: frozenset({'Refund: No', 'Cheat: No'})
Support: 0.40, Itemset: frozenset({'Marital Status: Married', 'Cheat: No'})
Support: 0.40, Itemset: frozenset({'Marital Status: Married'})
Support: 0.30, Itemset: frozenset({'Refund: Yes'})
Support: 0.30, Itemset: frozenset({'Cheat: Yes'})
Support: 0.30, Itemset: frozenset({'Refund: Yes', 'Cheat: No'})
Support: 0.30, Itemset: frozenset({'Marital Status: Married', 'Refund: No'})
Support: 0.30, Itemset: frozenset({'Refund: No', 'Marital Status: Single'})
Support: 0.30, Itemset: frozenset({'Cheat: Yes', 'Refund: No'})
Support: 0.20, Itemset: frozenset({'Marital Status: Divorced'})
Support: 0.20, Itemset: frozenset({'Marital Status: Single', 'Cheat: No'})
Support: 0.20, Itemset: frozenset({'Cheat: Yes', 'Marital Status: Single'})
```



```
Antecedent: frozenset({'Marital Status: Married'})
Consequent: frozenset({'Cheat: No'})
Antecedent Support: 0.4
Consequent Support: 0.7
Support: 0.4
Confidence: 1.0
Lift: 1.4285714285714286
Leverage: 0.12000000000000005
Conviction: inf
Zhang's Metric: 0.5000000000000002


Antecedent: frozenset({'Refund: Yes'})
Consequent: frozenset({'Cheat: No'})
Antecedent Support: 0.3
Consequent Support: 0.7
Support: 0.3
Confidence: 1.0
Lift: 1.4285714285714286
Leverage: 0.09
Conviction: inf
Zhang's Metric: 0.42857142857142855


Antecedent: frozenset({'Marital Status: Married'})
Consequent: frozenset({'Refund: No'})
Antecedent Support: 0.4
Consequent Support: 0.7
Support: 0.3
Confidence: 0.7499999999999999
Lift: 1.0714285714285714
Leverage: 0.020000000000000018
Conviction: 1.1999999999999997
Zhang's Metric: 0.11111111111111122
```