NAME: Yuli Kim

NJIT UCID: yk348

Email Address: yk348@njit.edu

Date: March 10, 2024

Professor: Yasser Abduallah

CS 634 104 Data Mining

Midterm Project Report

Abstract

Throughout this project, I explained the foundational concepts, principles, and data mining methods. Also, I evaluated the effectiveness and efficiency of these data mining tools. This involved designing and executing data mining algorithms, heuristics, methods, and techniques within the context of specific datasets.

Introduction

Data mining is discovering patterns and relationships in large datasets through various algorithms and techniques. Its goal is to extract valuable insights and knowledge from data to support decision-making and predictive analysis. Among the several algorithms used in data mining, this project mainly explored the concept of apriori algorithm. The Apriori algorithm is a popular algorithm used in data mining to discover frequent item sets in a dataset. It works by generating candidate itemsets based on the frequency of occurrence of items in the dataset and then pruning those that do not meet a specified support threshold. This process is repeated iteratively to find all frequent item sets, which can be used for association rule mining, often applied in market basket analysis and recommendation systems.

In this project, the apriori algorithm was used for transaction data in retail stores. With the algorithm, frequent items and their associations were calculated. Here are the steps that were followed:

1. Setting up dictionaries for candidates and frequent itemsets
2. Making datasets in CSV files
3. Preparing the dataset to ensure the order and uniqueness of items.
4. Gathering user input regarding the minimum support and confidence thresholds.
5. Iteratively creating candidate itemsets and updating frequent itemsets using the Apriori algorithm, which employs a systematic approach by examining all potential combinations of items.

The Apriori Algorithm is central to finding frequent itemsets, which are sets of items that often appear together in transactions. These itemsets provide valuable insights into how customers make purchases and their preferences.

Support and confidence are two essential metrics in data mining. Support measures the frequency of occurrence of an item or itemset, while confidence indicates the likelihood of items being bought together. These metrics play a crucial role in guiding our analysis.

Association Rules: By finding strong association rules, I can discover which items tend to be bought together frequently. These rules play a crucial role in improving sales strategies, like making better recommendations.

Project Workflow: This project adheres to a systematic workflow that includes multiple stages and the implementation of the Apriori Algorithm.

Data Loading and Preprocessing:
The process is started by importing transaction data from a dataset of a retail store. Each transaction contains a customer's list of purchased items. To maintain data accuracy, we prepare the dataset by removing duplicate items and arranging them according to a specified sequence.

Determination of Minimum Support and Confidence:
In data mining, user input is essential. We ask users for their preferred minimum support and confidence levels to filter out less important patterns.

Iteration Through Candidate Itemsets:
Using the Apriori Algorithm, we go through an iterative process. We begin with single items (itemset size K = 1) and move on to K = 2, K = 3, and so forth. This method involves systematically generating all possible combinations of itemsets.

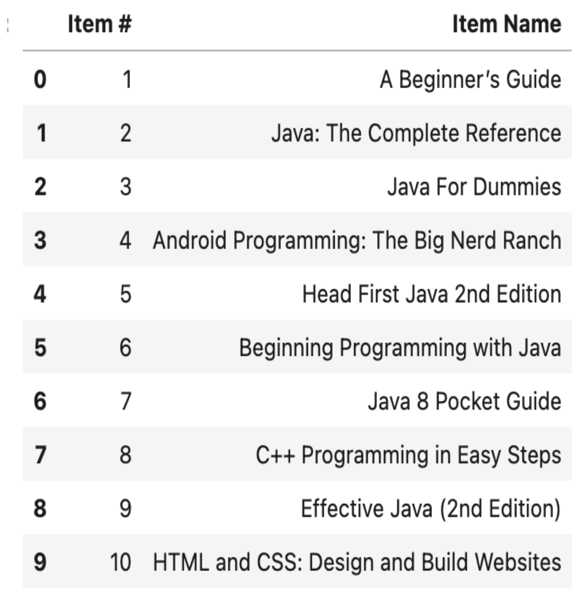
Support Count Calculation:
For every potential itemset, we determine its support by counting how many transactions include that itemset. If an itemset meets the minimum support threshold, we keep it; otherwise, we discard it.

Confidence Calculation:
We measure the confidence of association rules to gauge the strength of relationships between items. This step involves carefully comparing the support values for individual items and itemsets.

Association Rule Generation:
We extract association rules that fulfill both the minimum support and confidence criteria. These rules provide valuable insights into which items tend to be bought together frequently.

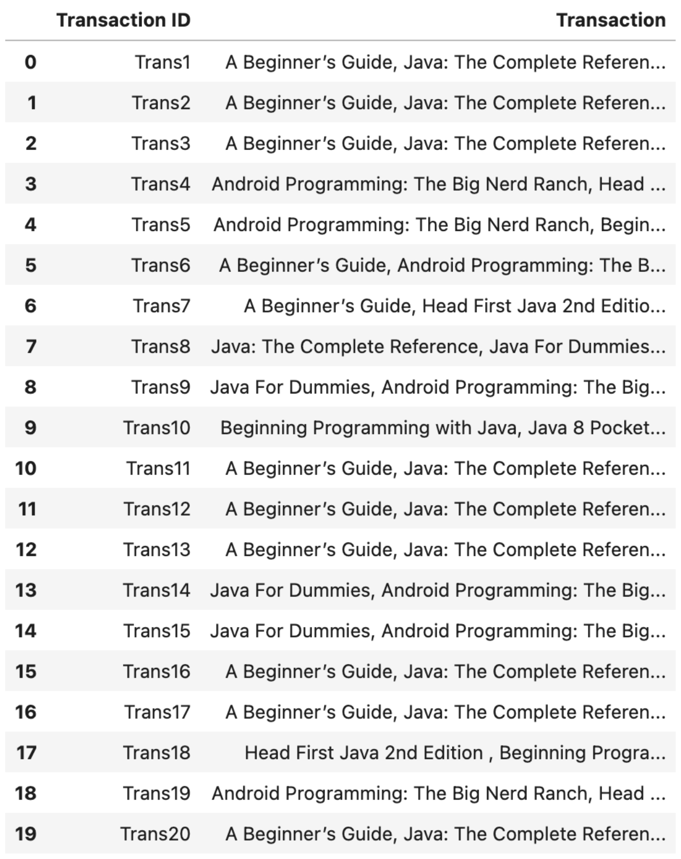
Evaluating Results:
To evaluate the project's success, we analyze its efficiency and effectiveness using metrics like support, confidence, and the association rules obtained. Additionally, we compare our custom Apriori Algorithm implementation with the Apriori library to ensure its dependability
Conclusion:
In conclusion, this project showcases how data mining principles and methods can be applied. I effectively used the Apriori Algorithm to uncover meaningful association rules from retail transaction data. The iterative approach, along with our tailored algorithm and consideration of user inputs, highlights the value of data mining in uncovering insights for decision-making in retail.

CSV Files



	Item #	Item Name
0	1	A Beginner’s Guide
1	2	Java: The Complete Reference
2	3	Java For Dummies
3	4	Android Programming: The Big Nerd Ranch
4	5	Head First Java 2nd Edition
5	6	Beginning Programming with Java
6	7	Java 8 Pocket Guide
7	8	C++ Programming in Easy Steps
8	9	Effective Java (2nd Edition)
9	10	HTML and CSS: Design and Build Websites

Figure 1: amazon item names csv

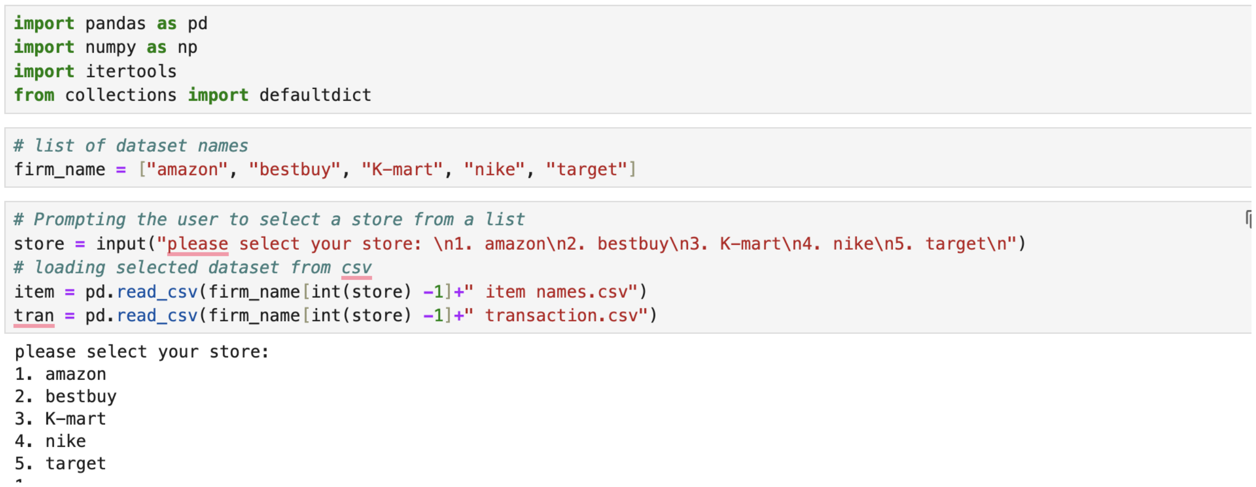


	Transaction ID	Transaction
0	Trans1	A Beginner’s Guide, Java: The Complete Referen...
1	Trans2	A Beginner’s Guide, Java: The Complete Referen...
2	Trans3	A Beginner’s Guide, Java: The Complete Referen...
3	Trans4	Android Programming: The Big Nerd Ranch, Head ...
4	Trans5	Android Programming: The Big Nerd Ranch, Begin...
5	Trans6	A Beginner’s Guide, Android Programming: The B...
6	Trans7	A Beginner’s Guide, Head First Java 2nd Editio...
7	Trans8	Java: The Complete Reference, Java For Dummies...
8	Trans9	Java For Dummies, Android Programming: The Big...
9	Trans10	Beginning Programming with Java, Java 8 Pocket...
10	Trans11	A Beginner’s Guide, Java: The Complete Referen...
11	Trans12	A Beginner’s Guide, Java: The Complete Referen...
12	Trans13	A Beginner’s Guide, Java: The Complete Referen...
13	Trans14	Java For Dummies, Android Programming: The Big...
14	Trans15	Java For Dummies, Android Programming: The Big...
15	Trans16	A Beginner’s Guide, Java: The Complete Referen...
16	Trans17	A Beginner’s Guide, Java: The Complete Referen...
17	Trans18	Head First Java 2nd Edition , Beginning Progra...
18	Trans19	Android Programming: The Big Nerd Ranch, Head ...
19	Trans20	A Beginner’s Guide, Java: The Complete Referen...

Figure 2: amazon transaction csv

Screenshots for code:

Reading csv files and choosing which store to use.



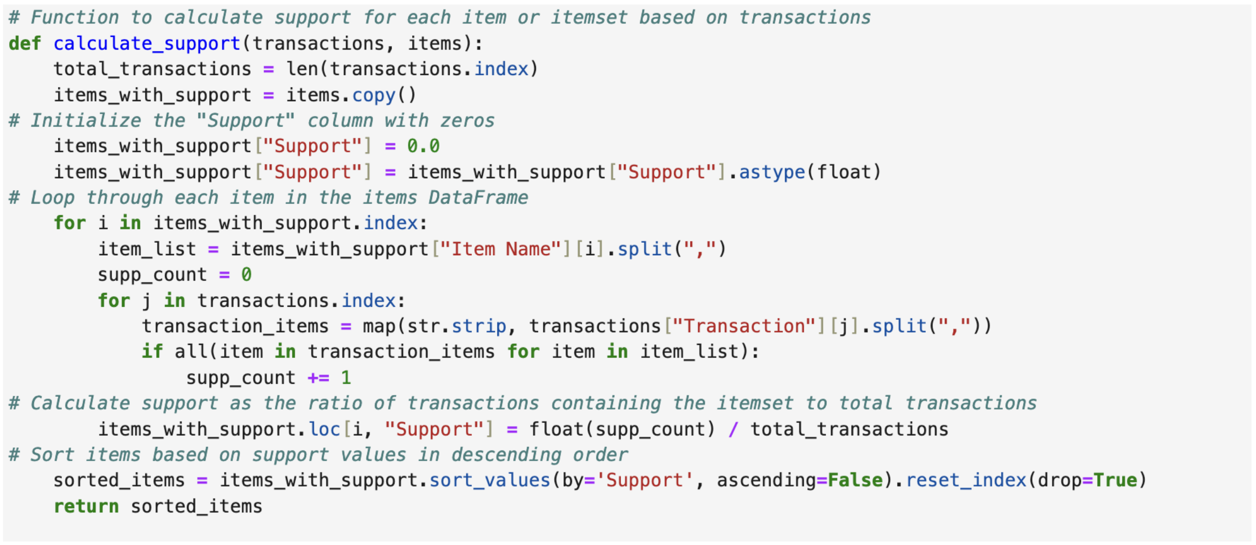
```
import pandas as pd
import numpy as np
import itertools
from collections import defaultdict
```

```
# list of dataset names
firm_name = ["amazon", "bestbuy", "K-mart", "nike", "target"]
```

```
# Prompting the user to select a store from a list
store = input("please select your store: \n1. amazon\n2. bestbuy\n3. K-mart\n4. nike\n5. target\n")
# loading selected dataset from csv
item = pd.read_csv(firm_name[int(store) -1]+" item names.csv")
tran = pd.read_csv(firm_name[int(store) -1]+" transaction.csv")
```

```
please select your store:
1. amazon
2. bestbuy
3. K-mart
4. nike
5. target
```

Calculating support



```
# Function to calculate support for each item or itemset based on transactions
def calculate_support(transactions, items):
    total_transactions = len(transactions.index)
    items_with_support = items.copy()
# Initialize the "Support" column with zeros
    items_with_support["Support"] = 0.0
    items_with_support["Support"] = items_with_support["Support"].astype(float)
# Loop through each item in the items DataFrame
    for i in items_with_support.index:
        item_list = items_with_support["Item Name"][i].split(",")
        supp_count = 0
        for j in transactions.index:
            transaction_items = map(str.strip, transactions["Transaction"][j].split(","))
            if all(item in transaction_items for item in item_list):
                supp_count += 1
# Calculate support as the ratio of transactions containing the itemset to total transactions
        items_with_support.loc[i, "Support"] = float(supp_count) / total_transactions
# Sort items based on support values in descending order
    sorted_items = items_with_support.sort_values(by='Support', ascending=False).reset_index(drop=True)
    return sorted_items
```

Calculating brute force

```
# Function to perform brute force item support calculation
def brute_force_item_support(tran, item, min_support):
    k = 1
    dist_items = len(list(set(list(item["Item Name"]))))
    item_support = pd.DataFrame(columns=["Item Name", "Support"])
    all_supports = []  # List to store all new support DataFrames
    while k < dist_items:
        sup_item = pd.DataFrame(columns=["Item Name", "Support"])
        sup_item["Item Name"] = [",".join(i) for i in itertools.combinations(list(item["Item Name"]), k)]
        new_support = calculate_support(tran, sup_item)
        if not new_support.empty:  # Check if new_support is not empty
            all_supports.append(new_support)
        k += 1
    # Concatenate all new support DataFrames
    if all_supports:
        item_support = pd.concat(all_supports)
    # Filter item_support DataFrame to keep itemsets with support greater than min_support
    filtered_item_support = item_support[item_support["Support"] > min_support]
    return filtered_item_support
```

Calculating support

```
# Function to calculate confidence for association rule A -> AB
def calculate_confidence(A,AB, item_support_df):
    A_support = item_support_df[item_support_df["Item Name"]==A]["Support"][0]
    AB_support = item_support_df[item_support_df["Item Name"]==AB]["Support"][0]
# Calculate and return the confidence
    return A_support/AB_support
```

```
                                   Item Name  Support
0                               Lab Top Case     0.70
1                                 Anti-Virus     0.70
2                                Flash Drive     0.65
3                                    Lab Top     0.60
4                           Microsoft Office     0.55
5                                   Speakers     0.55
6                                    Printer     0.50
7                             Digital Camera     0.45
8                        External Hard-Drive     0.45
0               Flash Drive,Microsoft Office     0.55
1                     Flash Drive,Anti-Virus     0.50
2                        Printer,Flash Drive     0.50
3                         Lab Top,Anti-Virus     0.50
4                       Lab Top,Lab Top Case     0.50
5                   Flash Drive,Lab Top Case     0.45
6                   Printer,Microsoft Office     0.45
7                Microsoft Office,Anti-Virus     0.40
8                    Digital Camera,Speakers     0.35
9                         Printer,Anti-Virus     0.35
10          Lab Top Case,External Hard-Drive     0.35
11             Microsoft Office,Lab Top Case     0.35
12                   Lab Top Case,Anti-Virus     0.35
13               Digital Camera,Lab Top Case     0.35
14                       Lab Top,Flash Drive     0.35
0       Printer,Flash Drive,Microsoft Office     0.45
1    Flash Drive,Microsoft Office,Anti-Virus     0.40
2             Printer,Flash Drive,Anti-Virus     0.35
3  Flash Drive,Microsoft Office,Lab Top Case     0.35
```

```
# Function to calculate confidence of an association rule
min_support = input("please writre minimum support i.e. 0.35: ")
min_support = float(min_support)
```

Generating association rules

```python
# Generating association rules
def generate_association_rules(pruned_items, min_confidence):
    rules_df = pd.DataFrame(columns=["Association Rules", "Support(%)", "Confidence(%)"])

# Extract item lists from pruned items
    pruned_items["item_list"] = pruned_items["ItemName"].apply(lambda x: x.split(","))

# Filter items with more than one element
    eligible_items = pruned_items[pruned_items["item_list"].apply(len) > 1]

    for idx, row in eligible_items.iterrows():
        item_list = row["item_list"]
# Generate all permutations of item lists
        for perm in itertools.permutations(item_list):
            for n in range(1, len(perm)):
                rule_left, rule_right, support, confidence = calculate_confidence(list(perm[:-n]), perm[-n:], pruned_items)
                association_rule = f"{rule_left} --> {rule_right}"
                rules_df = rules_df.append({"Association Rules": association_rule,
                                            "Support(%)": support * 100,
                                            "Confidence(%)": confidence * 100},
                                           ignore_index=True)

# Remove duplicate rules, sort by confidence, and filter by minimum confidence
    rules_df.drop_duplicates(inplace=True)
    rules_df = rules_df.sort_values(by="Confidence(%)", ascending=False).reset_index(drop=True)
    rules_df = rules_df[rules_df["Confidence(%)"] >= min_confidence * 100]

# Save results to CSV file
    rules_df.to_csv("output/Association_Rules.csv", index=False)

    return rules_df
```

Calculating time

```python
# for running time
from mlxtend.frequent_patterns import apriori, fpgrowth, association_rules
import time

def run_apriori_algorithm(items_df, transactions_df, min_sup, min_conf):
    # Apriori
    start_time_apriori = time.time()
    frequent_itemsets_apriori = apriori(items_df, min_support=min_sup, use_colnames=True)
    association_rules_apriori = association_rules(frequent_itemsets_apriori, metric="confidence", min_threshold=min_conf)
    end_time_apriori = time.time()
    apriori_time = end_time_apriori - start_time_apriori

    # FP-Growth
    start_time_fp = time.time()
    frequent_itemsets_fp = fpgrowth(items_df, min_support=min_sup, use_colnames=True)
    association_rules_fp = association_rules(frequent_itemsets_fp, metric="confidence", min_threshold=min_conf)
    end_time_fp = time.time()
    fp_time = end_time_fp - start_time_fp

    return association_rules_apriori, association_rules_fp, apriori_time, fp_time

# Function to compare results
def compare_results(brute_force_df, apriori_df, fp_df):
    print("Brute Force Results:")
    print(brute_force_df.head())
    print("\nApriori Results:")
    print(apriori_df.head())
    print("\nFP-Growth Results:")
    print(fp_df.head())
```

Github link:
https://github.com/yk348/Kim_Yuli_midtermproj