Midterm_Project

October 13, 2024

0.1 Installing the necessary libraries

[1]:
```
# %pip install pandas
# %pip install mlxtend
```

1 Generating the Dataset

[2]:
```
import pandas as pd
import random

# Set a specific random seed to create deterministic transactions
random.seed(0)

# Function to generate and save datasets
def generate_store_dataset(store_name, items):
    # Create an empty list to store transactions
    transactions = []

    # Generate 20 transactions with random items
    for i in range(1, 21):
        num_items = random.randint(1, 10)  # Number of items in each transaction
        transaction = random.sample(items, num_items)
        transaction_str = ', '.join(transaction)
        transactions.append([f"Trans{i}", transaction_str])

    # Create DataFrames
    df_items = pd.DataFrame(items, columns=['Item Name'])
    df_transactions = pd.DataFrame(transactions, columns=['Transaction ID',␣
↪'Transaction Items'])

    # Save the DataFrames to CSV files
    df_items.to_csv(f"{store_name}_items.csv", index=False)
    df_transactions.to_csv(f"{store_name}_transactions.csv", index=False)

    print(f"CSV files have been created for {store_name}: {store_name}_items.
↪csv and {store_name}_transactions.csv")
```

```
# List of stores and their items
stores = {
    "Grocery Store": ['Milk', 'Bread', 'Eggs', 'Cheese', 'Apples', 'Bananas',␣
 ↪'Chicken', 'Rice', 'Pasta', 'Tomatoes'],
    "Electronics Store": ['Laptop', 'Smartphone', 'Headphones', 'Smartwatch',␣
 ↪'Tablet', 'Portable Charger', 'Camera', 'Laptop Bag', 'Monitor', 'Keyboard'],
    "Clothing Store": ['Jeans', 'T-Shirt', 'Dress', 'Jacket', 'Scarf', 'Hat',␣
 ↪'Sneakers', 'Socks', 'Belt', 'Sweater'],
    "Sports Equipment Store": ['Football', 'Basketball', 'Tennis Racket',␣
 ↪'Baseball Glove', 'Golf Balls', 'Yoga Mat', 'Running Shoes', 'Swim Goggles',␣
 ↪'Fitness Tracker', 'Water Bottle'],
    "Bookstore": ['Fiction Novel', 'Science Textbook', 'History Book',␣
 ↪'Biography', 'Poetry Collection', 'Mystery Novel', 'Science Fiction Novel',␣
 ↪'Cookbook', 'Art Book', 'Children\'s Book']
}

# Generate datasets for each store
for store, items in stores.items():
    generate_store_dataset(store, items)
```

```
CSV files have been created for Grocery Store: Grocery Store_items.csv and
Grocery Store_transactions.csv
CSV files have been created for Electronics Store: Electronics Store_items.csv
and Electronics Store_transactions.csv
CSV files have been created for Clothing Store: Clothing Store_items.csv and
Clothing Store_transactions.csv
CSV files have been created for Sports Equipment Store: Sports Equipment
Store_items.csv and Sports Equipment Store_transactions.csv
CSV files have been created for Bookstore: Bookstore_items.csv and
Bookstore_transactions.csv
```

2 Association Rule Mining Analysis

This script performs association rule mining using three different methods: a custom Brute Force approach, the Apriori algorithm, and the FP-Growth algorithm, comparing their performance on a selected dataset. First, we load the transaction data from a specified store.

[3]:
```
# import libraries and load data

import pandas as pd
import itertools
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, fpgrowth, association_rules
import time

def load_transactions(filename):
    """ Load and preprocess transactions from a CSV file. """
```

```
    df = pd.read_csv(filename)
    transactions = df['Transaction Items'].apply(lambda x: x.split(','))
    return transactions
```

2.1 Brute Force Method

Implementing the brute force method to generate frequent itemsets. This method evaluates every possible combination of items to determine which sets meet the minimum support threshold.

[4]:
```
# Brute Force Functino

def brute_force_frequent_itemsets(transactions, min_support):
    """ Efficiently generate frequent itemsets using the brute force method. """
    item_set = set(itertools.chain.from_iterable(transactions))
    item_list = list(item_set)
    frequent_itemsets = []
    num_transactions = len(transactions)
    transaction_sets = [set(transaction) for transaction in transactions]

    for r in range(1, len(item_list) + 1):
        for itemset in itertools.combinations(item_list, r):
            itemset_support = sum(1 for transaction in transaction_sets if␣
 ↪set(itemset).issubset(transaction)) / num_transactions
            if itemset_support >= min_support:
                frequent_itemsets.append((itemset, itemset_support))

    return pd.DataFrame(frequent_itemsets, columns=["itemsets", "support"])
```

2.2 Algorithm Execution

Here, we execute the Apriori and FP-Growth algorithms using the `mlxtend` library, and measure the execution time for each. We also include the Brute Force method executed previously.

[5]:
```
def run_mlxtend_algorithm(transactions, min_support, algorithm):
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    if algorithm == 'apriori':
        return apriori(df, min_support=min_support, use_colnames=True)
    else:
        return fpgrowth(df, min_support=min_support, use_colnames=True)

def generate_rules(frequent_itemsets, min_confidence):
    if not frequent_itemsets.empty:
        return association_rules(frequent_itemsets, metric="confidence",␣
 ↪min_threshold=min_confidence)
    return pd.DataFrame()
```

2.3 Results Display

For each algorithm, we print the rules found, sorted by the lift metric to identify the most relevant associations.

[6]:
```
def print_rules(rules):
    if not rules.empty:
        if(len(rules) > 20):
            print("Concating the rules to only 20..")
        rules_sorted = rules.sort_values(by='lift', ascending=False).head(20)
        for index, rule in rules_sorted.iterrows():
            print(f"Rule {index + 1}: {rule['antecedents']} ->␣
↪{rule['consequents']}, "
                  f"Support: {rule['support']:.4f}, Confidence:␣
↪{rule['confidence']:.4f}, Lift: {rule['lift']:.4f}")
    else:
        print("No rules generated.")

def main():
    store_options = ['Bookstore', 'Clothing Store', 'Electronics Store',␣
↪'Grocery Store', 'Sports Equipment Store']
    print("Available stores:")
    for i, option in enumerate(store_options, 1):
        print(f"{i}. {option}")

    store_index = int(input("Please select your store (1-5): ")) - 1
    transactions_file = f"{store_options[store_index]}_transactions.csv"
    transactions = load_transactions(transactions_file)

    min_support = float(input("Enter the minimum support (as a decimal, e.g., 0.
↪01 for 1%): "))
    min_confidence = float(input("Enter the minimum confidence (as a decimal, e.
↪g., 0.7 for 70%): "))

    # Running all algorithms
    # run_all_algorithms(transactions, min_support, min_confidence)
    start_time = time.time()
    bf_itemsets = brute_force_frequent_itemsets(transactions, min_support)
    bf_duration = time.time() - start_time
    print(f"Brute Force - Duration: {bf_duration:.2f}s, Itemsets Found:␣
↪{len(bf_itemsets)}")
    if not bf_itemsets.empty:
        bf_rules = generate_rules(bf_itemsets, min_confidence)
        print_rules(bf_rules)
    else:
        print("No frequent itemsets found with brute force.")
```

```
    # Apriori
    start_time = time.time()
    ap_itemsets = run_mlxtend_algorithm(transactions, min_support, 'apriori')
    ap_duration = time.time() - start_time
    ap_rules = generate_rules(ap_itemsets, min_confidence)
    print(f"Apriori - Duration: {ap_duration:.2f}s, Rules Found:␣
↪{len(ap_rules)}")
    print_rules(ap_rules)

    # FP-Growth
    start_time = time.time()
    fp_itemsets = run_mlxtend_algorithm(transactions, min_support, 'fpgrowth')
    fp_duration = time.time() - start_time
    fp_rules = generate_rules(fp_itemsets, min_confidence)
    print(f"FP-Growth - Duration: {fp_duration:.2f}s, Rules Found:␣
↪{len(fp_rules)}")
    print_rules(fp_rules)

    print("Timing Performance:")
    print(f"Brute Force - Duration: {bf_duration:.2f}s")
    print(f"Apriori - Duration: {ap_duration:.2f}s")
    print(f"FP-Growth - Duration: {fp_duration:.2f}s")

if __name__ == "__main__":
    main()
```

```
Available stores:
1. Bookstore
2. Clothing Store
3. Electronics Store
4. Grocery Store
5. Sports Equipment Store
Please select your store (1-5): 1
Enter the minimum support (as a decimal, e.g., 0.01 for 1%): 0.3
Enter the minimum confidence (as a decimal, e.g., 0.7 for 70%): 0.8
Brute Force - Duration: 3.83s, Itemsets Found: 32
Concating the rules to only 20..
Rule 39: frozenset({' Fiction Novel'}) -> frozenset({' Art Book', '
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.8571
Rule 36: frozenset({' Art Book', ' Biography'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.8571
Rule 34: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book", ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 31: frozenset({" Children's Book", ' Art Book'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 32: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({" Children's
Book"}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
```

```
Rule 33: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 13: frozenset({' Biography'}) -> frozenset({' Fiction Novel'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 12: frozenset({' Fiction Novel'}) -> frozenset({' Biography'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 8: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book"}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 7: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel'}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 37: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({'
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 38: frozenset({' Biography'}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 35: frozenset({' Fiction Novel', ' Biography'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 11: frozenset({' Fiction Novel'}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 27: frozenset({' Cookbook', ' Fiction Novel'}) -> frozenset({' Art Book'}),
Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 6: frozenset({" Children's Book"}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 16: frozenset({" Children's Book", ' Cookbook'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 19: frozenset({' Cookbook', ' Poetry Collection'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 30: frozenset({" Children's Book", ' Fiction Novel'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 23: frozenset({' Cookbook', ' History Book'}) -> frozenset({' Art Book'}),
Support: 0.3500, Confidence: 1.0000, Lift: 2.2222
Apriori - Duration: 0.01s, Rules Found: 39
Concating the rules to only 20..
Rule 20: frozenset({' Fiction Novel'}) -> frozenset({' Art Book', '
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.8571
Rule 17: frozenset({' Art Book', ' Biography'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.8571
Rule 10: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel'}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 25: frozenset({" Children's Book", ' Art Book'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 28: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book", ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 19: frozenset({' Biography'}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 18: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({'
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 27: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
```

```
Rule 11: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book"}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 26: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({" Children's
Book"}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 8: frozenset({' Biography'}) -> frozenset({' Fiction Novel'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 7: frozenset({' Fiction Novel'}) -> frozenset({' Biography'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 33: frozenset({' Cookbook', ' History Book'}) -> frozenset({' Art Book'}),
Support: 0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 29: frozenset({' Cookbook', ' Fiction Novel'}) -> frozenset({' Art Book'}),
Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 16: frozenset({' Fiction Novel', ' Biography'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 35: frozenset({' Cookbook', ' Poetry Collection'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 5: frozenset({' Fiction Novel'}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 2: frozenset({" Children's Book"}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 21: frozenset({" Children's Book", ' Cookbook'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 24: frozenset({" Children's Book", ' Fiction Novel'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
FP-Growth - Duration: 0.00s, Rules Found: 39
Concating the rules to only 20..
Rule 27: frozenset({' Fiction Novel'}) -> frozenset({' Art Book', '
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.8571
Rule 24: frozenset({' Art Book', ' Biography'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.8571
Rule 20: frozenset({' Fiction Novel'}) -> frozenset({' Biography'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 30: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book"}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 25: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({'
Biography'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 26: frozenset({' Biography'}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 21: frozenset({' Biography'}) -> frozenset({' Fiction Novel'}), Support:
0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 29: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel'}),
Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 33: frozenset({" Children's Book", ' Art Book'}) -> frozenset({' Fiction
Novel'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 34: frozenset({' Fiction Novel', ' Art Book'}) -> frozenset({" Children's
Book"}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 35: frozenset({" Children's Book"}) -> frozenset({' Fiction Novel', ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
```

```
Rule 36: frozenset({' Fiction Novel'}) -> frozenset({" Children's Book", ' Art
Book'}), Support: 0.3000, Confidence: 0.8571, Lift: 2.4490
Rule 32: frozenset({" Children's Book", ' Fiction Novel'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 12: frozenset({' Cookbook', ' Poetry Collection'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 28: frozenset({" Children's Book"}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 15: frozenset({' Fiction Novel'}) -> frozenset({' Art Book'}), Support:
0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 17: frozenset({' Cookbook', ' Fiction Novel'}) -> frozenset({' Art Book'}),
Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 4: frozenset({' Cookbook', ' History Book'}) -> frozenset({' Art Book'}),
Support: 0.3500, Confidence: 1.0000, Lift: 2.2222
Rule 37: frozenset({" Children's Book", ' Cookbook'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Rule 23: frozenset({' Fiction Novel', ' Biography'}) -> frozenset({' Art
Book'}), Support: 0.3000, Confidence: 1.0000, Lift: 2.2222
Timing Performance:
Brute Force - Duration: 3.83s
Apriori - Duration: 0.01s
FP-Growth - Duration: 0.00s
```

[ ]:

[ ]:

[ ]: