Bài tập Thực hành môn Khai phá Dữ liệu

Họ và tên: Huỳnh Nguyễn Thế Dân

MSSV: 21110256

Lớp: 21TTH1

1. Cài đặt lại hàm TransactionEncoder

Định nghĩa lớp MyTransactionEncoder

```
class MyTransactionEncoder:
    def __init__(self):
        self.unique_items = set()  # Khởi tạo một set để lưu trữ các mục độc nhất gặp trong các giao dịch
        self.column_names = []  # Khởi tạo danh sách rỗng để lưu trữ tên cột

    def fit(self, transactions):
        for transaction in transactions:
            for item in transaction:
                self.unique_items.add(item)  # Thêm mỗi mục trong giao dịch vào set các mục độc nhất
        self.unique_items = sorted(self.unique_items)  # Sắp xếp các mục độc nhất
        self.column_names = self.unique_items  # Lưu tên cột dưới dạng các mặt hàng được sắp xếp
        return self

    def transform(self, transactions):
        item_index = {item: idx for idx, item in enumerate(self.unique_items)}  # Tạo bảng tra cứu để nhanh chóng tìm chỉ số của mỗi mục
        transaction_matrix = []  # Khởi tạo danh sách rỗng để lưu trữ ma trận của các giao dịch
        for transaction in transactions:
            row = [False] * len(self.unique_items)  # Khởi tạo một hàng với giá trị mặc định là False
            for item in transaction:
                if item in item_index:  # Kiểm tra nếu mục có trong giao dịch
                    row[item_index[item]] = True  # Đánh dấu chỉ số tương ứng là True
            transaction_matrix.append(row)  # Thêm hàng vào ma trận
        return transaction_matrix

    def fit_transform(self, transactions):
        self.fit(transactions)  # Áp dụng phương thức fit để xác định
```

```
các mục độc nhất
        return self.transform(transactions)  # Biến đổi danh sách các
giao dịch thành ma trận

    def get_feature_names(self):
        """ Trả về danh sách các tên cột, tương ứng với các mặt hàng
duy nhất """
        return self.column_names
```

Tóm tắt thuật toán:

Lớp **MyTransactionEncoder** được thiết kế để biến đổi các giao dịch, mỗi giao dịch là một danh sách các mục, thành một ma trận nhị phân. Thuật toán này bao gồm:

1. **Khởi tạo**: Khởi tạo một set và một danh sách để lưu các mục duy nhất và tên cột.
2. **Phương thức fit**: Duyệt qua mỗi giao dịch để xác định tất cả các mục duy nhất có trong các giao dịch, sau đó sắp xếp chúng để lưu vào danh sách tên cột.
3. **Phương thức transform**: Dựa trên các mục được xác định từ fit, tạo một ma trận nhị phân, mỗi hàng tương ứng với một giao dịch và mỗi cột tương ứng với sự hiện diện của một mục trong giao dịch đó.
4. **Phương thức fit_transform**: Kết hợp cả hai bước fit và transform để trực tiếp chuyển đổi các giao dịch nhập vào thành ma trận nhị phân.
5. **Phương thức get_feature_names**: Trả về danh sách tên các cột, giúp hiểu rõ mỗi cột trong ma trận biểu diễn cho mục nào.

---

2. Cài đặt lại thuật toán Apriori

Định nghĩa hàm MyApriori

```
def MyApriori(transactions, min_support):
    from itertools import combinations

    # Hàm lấy tất cả các mục duy nhất từ các giao dịch
    def items_in_transactions(transactions):
        items = set()
        for transaction in transactions:
            items.update(transaction)
        return items

    # Hàm tạo ra các tập ứng viên k-item từ các tập thường xuyên (k-1)-
item
    def generate_candidate_sets(frequent_sets, k):
        return set([
            frozenset(set1 | set2)
            for set1 in frequent_sets
            for set2 in frequent_sets
            if len(set1 | set2) == k
        ])
```

```
    # Hàm tính tỷ lệ hỗ trợ cho một tập ứng viên cụ thể
    def calculate_support(candidate_set, transactions):
        return sum(1 for transaction in transactions if
candidate_set.issubset(transaction)) / len(transactions)

    # Lấy tất cả các mục từ các giao dịch
    items = items_in_transactions(transactions)
    # Khởi tạo các tập ứng viên 1-item
    candidate_1_item_sets = [{item} for item in items]
    frequent_sets = []
    support_sets = []

    # Xác định các tập thường xuyên 1-item
    for candidate in candidate_1_item_sets:
        support = calculate_support(candidate, transactions)
        if support >= min_support:
            frequent_sets.append(frozenset(candidate))
            support_sets.append(support)

    # Khởi tạo k = 2
    k = 2
    while True:
        # Tạo các tập ứng viên k-item
        candidate_k_item_sets = generate_candidate_sets(frequent_sets,
k)
        current_frequent_sets = []
        current_support_sets = []
        for candidate_set in candidate_k_item_sets:
            support = calculate_support(candidate_set, transactions)
            if support >= min_support:
                current_frequent_sets.append(candidate_set)
                current_support_sets.append(support)

        # Dừng nếu không có thêm tập ứng viên thường xuyên
        if not current_frequent_sets:
            break

        # Cập nhật tập thường xuyên và tăng k lên 1
        frequent_sets.extend(current_frequent_sets)
        support_sets.extend(current_support_sets)
        k += 1

    # Trả về tập thường xuyên và tỷ lệ hỗ trợ
    return frequent_sets, support_sets
```

Tóm tắt thuật toán Apriori trong hàm MyApriori:

1. **Thu thập các mục độc nhất**: Hàm đầu tiên (items_in_transactions) đi qua mọi giao dịch để lấy ra tất cả các mục độc nhất.
2. **Khởi tạo các tập ứng viên**: Bắt đầu từ các tập 1-item, xác định những tập nào có tỷ lệ hỗ trợ đủ lớn để trở thành các tập thường xuyên.
3. **Lặp qua các kích thước tập hợp**: Tăng dần kích thước của tập ứng viên và lặp lại việc tạo tập ứng viên mới và tính toán hỗ trợ cho chúng.

4. **Dừng lại**: Quá trình dừng lại khi không còn tạo được tập ứng viên mới nào vượt qua ngưỡng hỗ trợ.
5. **Kết quả**: Trả về các tập thường xuyên và tỷ lệ hỗ trợ tương ứng của chúng, cho phép khám phá các mẫu và quy tắc liên kết trong dữ liệu.

End.