Bài tập thực hành-Khai thác dữ liệu-tuần 5

Phan Hồng Trâm - 21110414

May 2024

Mục lục

1 trình bày tóm tắt lại phần code

- Import thư viện cần thiết và đọc file data:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori

dataset = pd.read_csv('C:\\Users\\PC\\Desktop\\CODE\\PY\\Nm 3 HKII\\KPDL\\lab05\\data.csv', encoding
    ='utf-8', header=None)
print(dataset)
```

- Tiền xử lý dữ liệu, đưa dữ liệu về dạng mảng và loại bỏ các giá trị null:

```
# Data Preprocessing
transactions = []
for i in range(dataset.shape[0]):
    row = dataset.iloc[i].dropna().tolist()
    transactions.append(row)
print(transactions)
```

- Mô hình hóa dữ liệu:

```
association_rules = apriori(transactions, min_support = 0.5, min_length = 2)
results = list(association_rules)
results_filter = filter(lambda x: len(x.items) > 1, results) # filter transactions containing at
    least 2 items
# print(results_filter)
```

  - Đầu tiên ta sử dụng hàm `apriori` từ thư viện `apyori` để tạo ra một đối tượng chứa thông tin về các tập itemset thường xuyên (frequent itemsets), bao gồm support (frequency), confidence, lift, etc.
  - Ta truyền tham số vào hàm `apriori` với:
    * `transactions`: Danh sách các giao dịch ta đã tiền xử lý ở trên.
    * `min_support=0.5`: Ngưỡng hỗ trợ tối thiểu.
    * `min_length=2`: Tham số này chỉ định độ dài tối thiểu của các itemset cần xem xét.
  - `results = list(association_rules)`: chuyển đổi đối tượng generator trả về bởi hàm `apriori` thành một danh sách. (không bắt buộc nhưng có thể hữu ích cho việc xử lý hoặc kiểm tra thêm).
  - `results_filter`: tạo ra một danh sách lọc các kết quả mà ở đó độ dài của itemset trong thuộc tính `items` của mỗi đối tượng quy tắc `x` lớn hơn 1 (bằng cách sử dụng hàm `lambda`). Việc lọc này đảm bảo rằng chỉ giữ lại các quy tắc có ít nhất hai mặt hàng trong kết quả.

- `def inspect(results)`: xử lý và sắp xếp các quy tắc liên kết đã lọc được lấy từ danh sách `results_filter`.

```
def inspect(results):
    item_sets = []
    supports = []
    for result in results:
        item_sets.append(tuple(result[0]))
        supports.append(result[1])
    return list(zip(item_sets, supports))
results_df = pd.DataFrame(inspect(results_filter), columns = ['Item Set', 'Support'])
```

- sort thứ tự theo `Support` giảm dần và in ra kết quả:

```
pd.set_option('display.max_rows', dataset.shape[0])
results_df.sort_values('Support', ascending=False)
print(results_df)
```

  - `pd.set_option('display.max_rows', dataset.shape[0])`: sử dụng hàm `pd.set_option` từ thư viện Pandas để có thể điều chỉnh cách hiển thị DataFrame, trong đó `'display.max_rows'`: Cài đặt tùy chọn liên quan đến số lượng hàng tối đa được hiển thị trong đầu ra DataFrame, `dataset.shape[0]`: Truy xuất số lượng hàng trong DataFrame `dataset`.

2 So sánh kết quả

```
Combination 2 by 2
15it [00:00, 154.61it/s]
Combination 3 by 3
20it [00:00, 240.91it/s]
Combination 4 by 4
15it [00:00, 230.73it/s]
Combination 5 by 5
6it [00:00, 239.93it/s]
Combination 6 by 6
1it [00:00, 166.69it/s]
                    Item   Support
0          Apple & Bread  0.545455
1         Apple & Butter  0.500000
2           Apple & Wine  0.500000
3           Apple & Milk  0.500000
4         Bread & Butter  0.590909
5           Bread & Wine  0.590909
6           Bread & Milk  0.590909
7         Butter & Wine   0.500000
8         Butter & Milk   0.590909
9           Wine & Milk   0.636364
10  Bread & Butter & Milk  0.500000
11    Bread & Wine & Milk  0.500000
```

	Item Set	Support
9	(Wine, Milk)	0.636364
4	(Butter, Bread)	0.590909
5	(Bread, Milk)	0.590909
6	(Bread, Wine)	0.590909
7	(Butter, Milk)	0.590909
0	(Bread, Apple)	0.545455
1	(Butter, Apple)	0.500000
2	(Apple, Milk)	0.500000
3	(Wine, Apple)	0.500000
8	(Butter, Wine)	0.500000
10	(Butter, Bread, Milk)	0.500000
11	(Bread, Wine, Milk)	0.500000

Hình 1: Kết quả code của cô và tự code

Nhận xét: Kết quả bài code của cô và kết quả tự code giống nhau.