In the name of God

استاد : دکتر هراتی

دانشجو : توحید حقیقی سیس

شماره دانشجویی : 830598021

موضوع : تمرین پنجم

تمرین اول :

۱-(طراحی و پیاده سازی الگوریتم)

داده مورد استفاده در این تمرین شامل سبدهای خرید یک خرده فروشی است. هر سطر از این دیتاست نشانگر یک سبد خرید و در هر سطر تعدادی شناسه محصول وجود دارد که نشانگر محصولات خریداری شده است. در این تمرین شما فقط با product_id و order_id کار خواهید داشت.

الف) با استفاده از الگوریتم Apriori و support=0.03 کالاهای پر تکرار در سبدهای خرید را پیدا کنید.

ب) با استفاده از الگوریتم PCY و support=0.04 کالاهای پر تکرار در سبدهای خرید را پیدا کنید.

برای حل مسائل این قسمت چون هدف ما پیدا کردن اقلام پر تکرار است باید ابتدا تمام داده های درون دیتا ست و یا در روش هایی قسمتی از ان را به داخل رم اورده و تعداد هر قلم را بشماریم و بعد ترکیب دو دویی و بعد سه تایی و ... را به دست اوریم .

در این مثال چون دیتاست کوچک هست ما میتوانیم ان را خط به خط خوانده و تعداد اقلام ان را شمارش کنیم ولی در سوالاتی که داده بزرگ هست میتوانیم از الگوریتم Son و از ایده مپ ردیوس استفاده کنیم تا بتوانیم ان را به صورت موازی اجرا کنیم و هم چنین با روش Tivonen نیز میتوانیم ان را حل کنیم چون این روش فقط قسمتی از داده ها را به داخل رم می آورد و از روی ان تصمیم میگیرد و با مشکل حافظه بر نمیخوریم .

ولی در این سوال Apriory و PCY خواسته شده با داده های کم که یک حالت ایده ال از این روش است .

در زیر مراحل انجام الگوریتم را توضیح میدهم :

- **خواندن داده ها در حافظه**
- **شمردن داده های تکی**
- **همزمان با خواندن داده های تکی تشکیل داده های دوتایی و هش ان ها در جدول**

- انتخاب یک th از داده ها برای مرحله بعد از تکی ها
- تکرار این روش تا n تایی شدن و تشکیل همه حالت ها

پیاده سازی :

در مرحله اول داده ها را از فایل خوانده و ایتم های ان را شمارش میکنیم هم زمان ان را در داخل ارایه نیز میریزم تا در مراحل بعدی از این ارایه استفاده کنیم .

```
In [33]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt

In [71]: from collections import Counter

In [72]: a = Counter()

In [81]: main_array=[]
         basket_count = Counter()
         with open("retail.txt", "r") as f:
             lines = f.readlines()
         for line in lines[1:]:
             items = line.split(" ")
             main_array.append(items)
             for item in items:
                 if(item!='\n'):
                     basket_count[item] += 1
```

من در این بخش که در عکس بالا میبینید از یک کتابخانه برای شمارش استفاده کردم که کار را راحت میکند به نام Collections import Counter که یک متغیر از نوع Counter میشازیم و کلید را متن و مقدار ان را تعداد تکرار ان محصول میگذاریم .

و خروجی بخش بالا به صورت زیر است :

که مقدار ان شامل یک کلید و یک تعداد میشود که به ان دیکشنری نیز میکویند .

```
In [82]: basket_count
```

```
Out[82]: Counter({'30': 540,
                  '31': 920,
                  '32': 15167,
                  '33': 37,
                  '34': 14,
                  '35': 78,
                  '36': 2936,
                  '37': 1074,
                  '38': 15596,
                  '39': 50675,
                  '40': 211,
                  '41': 14945,
                  '42': 101,
                  '43': 219,
                  '44': 196,
                  '45': 911,
                  '46': 75,
                  '47': 444,
                  '48': 42135,
```

مرحله بعدی محاسبه ساپورت هر محصول است که در زیر ان را میبینید هر تعداد را بر تعداد کل تقسیم میکنیم .

```
all_item_count=16471
onepair_support_list=[]
for item in basket_count:
    if((basket_count[item]/all_item_count)>0.03):
        onepair_support_list.append(item)

onepair_support_list
```

```
['30',
 '31',
 '32',
 '36',
 '37',
 '38',
 '39',
 '41',
 '45',
 '48',
 '49',
 '52',
 '53',
 '55',
 '56',
 '60',
 '62',
 '65',
 '76',
```

خروجی ان نیز به صورت بالا میباشد به این صورت که انهایی که ساپورت بالای 0.03 دارند را بر میگرداند .

دیتا ستی که به ارایه تبدیل کردیم نیز به صورت زیر است از ان در قسمت های بعد برای یافتن تعداد جفت ها استفاده میکنیم .

```
In [83]: main_array
```

```
Out[83]: [['30', '31', '32', '\n'],
          ['33', '34', '35', '\n'],
          ['36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '\n'],
          ['38', '39', '47', '48', '\n'],
          ['38',
           '39',
           '48',
           '49',
           '50',
           '51',
           '52',
           '53',
           '54',
           '55',
           '56',
           '57',
           '58',
           '\n'],
          ['32', '41', '59', '60', '61', '62', '\n'],
```

در مرحله بعد از روی اعضای onepair_compare اعضای 2 تایی را میسازیم این کار را با کتابخانه زیر انجام میدهیم .

```
In [116]: from itertools import combinations
```

```
In [117]: twopair_count = Counter()
          compine2array=combinations(onepair_support_list, 2)
          for i, j in compine2array:
              for item in main_array:
                  if(i in item):
                      if(j in item):
                          twopair_count[i+'|'+j]+=1
```

```
In [124]: count_combine=len(list(combinations(onepair_support_list, 2)))
          print(count_combine)
          twopair_support_list=[]
          for item in twopair_count:
              if((twopair_count[item]/count_combine)>0.03):
                  twopair_support_list.append(item)
```

از کتابخانه itertools و از تابع combinations ان استفاده میکنیم و بر روی تمام سبد کالا ها حرکت کرده و چک میکنیم هر دو در چند تا سبد خرید وجود دارد برای این کار من کلید را i+’|’+j میگزارم تا کلید ان هایی که یکی است به مقدارشان یکی اضافه شود .

بعد از بین همه ان ها ان هایی که th ان ها از 0.03 بیشتر بود را نمایش میدهیم . خروجی این قسمت به صورت زیر است و تمام 2 تایی های مشابه رو نشان میدهد .

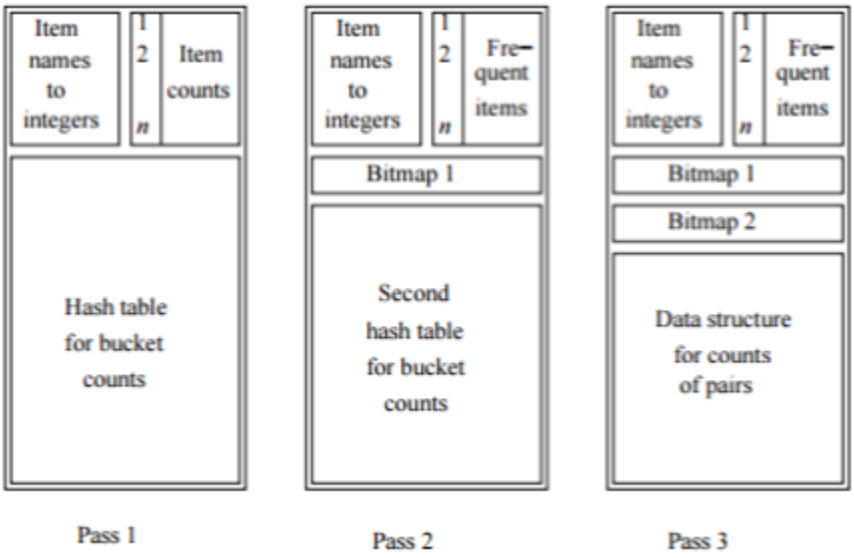
```
In [125]: twopair_support_list
```

```
Out[125]: ['31|39',
           '32|38',
           '32|39',
           '32|41',
           '32|48',
           '32|65',
           '32|89',
           '32|170',
           '32|225',
           '32|237',
           '32|310',
           '36|38',
           '36|39',
           '36|41',
           '36|48',
           '37|38',
           '37|39',
           '37|48',
           '38|39',
```

این روش را تا جایی ادامه میدهیم که دیگر نتوانیم دسته بندی کنیم .

قسمت 2 :

در این قسمت مشابه قسمت دوم عمل میکنیم با تفاوت هایی در این روش در هنگام شمارش هر سبد خرید 2 تایی های ا را نیز تولید و به خانه ای از ارایه هش میکنیم تا در مراحل بعدی به ما برای کاهش تعداد مقایسات کمک کند و سرعت و حافظه ما را بهینه تر کند .



بعد از هش کردن مقادیر دودویی در یک جدول 2 در 2 ان جدول را به bit array تبدیل میکنیم هر کدام از اعضای این ارایه از th بیشتر بود 1 وبقیه مقادیر 0 خواهد بود .

و مانند pass 2 جدول pass 1 در مرحله دوم تبدیل به جدول بیتی میشود و این باعث میشود که ما دیگر نخواهیم تمام حالت های 2 تایی رو بررسی کنیم فقط ان دوتایی هایی را بررسی میکنیم که در جدول هش به مقدار 1 هش شوند .

مرحله اول سوال :

خواندن هر basket و شمارش ان ها بعد از خواندن هر سبد و هش کردن دو دویی هر کدام از انها :

```
hash_table=np.zeros((500, 500))
main_array=[]
basket_count = Counter()
with open("retail.txt", "r") as f:
    lines = f.readlines()
for line in lines[1:]:
    items = line.split(" ")
    items.remove("\n")
    main_array.append(items)
    for item in items:
        if(item!='\n'):
            basket_count[item] += 1
```

بعد از خواندن هر سبد خرید کارهای زیر را در کد بالا انجام میدهم :

1. **Split کردن داده ها برای دسترسی به هر کالا**
2. **در هنگام split یک کارکتر "\n" وجود دارد ان را حذف میکنم**
3. **ان را به ارایه اضافه میکنم تا در مراحل بعد راحت تر استفاده کنم**
4. **تمام ایتم های هر سبد را شمرده و به شمارنده ان اضافه میکنم**

```
#----------------- two pair --------------

compine2array=combinations(items, 2)
for i,j in compine2array:
    m=hash_functions[0](int(i))
    n=hash_functions[0](int(j))
    if(i in items):
        if(j in items):
            hash_table[m,n]+=1
```

در مرحله بعد از تابع هش زیر برای هش کردن i , j هر جفت استفاده میکنم و به شمارنده ارایه 2 بعدی اضافه میکنم این ارایه به ان معنی است که درایه هایی که مقدار ان ها از صفر بیشتر بوده 2 تایی های موجود در سبد است .

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from collections import Counter
from itertools import combinations
import random
```

```

```

```
def get_hash_function(hash_range):
    r = random.randint(1, 100000)
    def hash(x):
        nonlocal r
        return (r * x + (r + 1)) % hash_range
    return hash
```

```
TABLE_SIZE=500
hash_functions = [get_hash_function(TABLE_SIZE) for _ in range(5)]
```

```
print(hash_functions[0](100))
```

از کتابخانه های بالا برای حل مسایل استفاده کردم و تابع هش را نیز به نام get_hash_function نوشتم که خروجی این تابع یک تابع هش خواهد بود

در این روش چون از یک تابع هش استفاده میکنیم من در کل سوال فقط از تابع hash_function[0] استفاده کردم .

```
#second step

all_item_count=16471
onepair_support_list=[]
for item in basket_count:
    if((basket_count[item]/all_item_count)>0.03):
        onepair_support_list.append(item)

onepair_support_list
```

```
['30',
 '31',
 '32',
 '36',
 '37',
 '38',
 '39',
 '41',
 '45',
 '48',
 '49',
 '52',
 '53',
 '55',
 '56',
 '60',
 '62'.
```

اگر مقدار ان از Th بیشتر بود ان را به عنوان frequent item انتخاب کن .

```
In [81]: hash_table=hash_table>30
         hash_table

Out[81]: array([[ True, False,  True, ..., False, False,  True],
                [False, False, False, ..., False, False, False],
                [False, False,  True, ..., False, False, False],
                ...,
                [False, False, False, ..., False, False, False],
                [False, False,  True, ..., False, False, False],
                [False, False,  True, ..., False, False, False]])
```

و ارایه هش را در حد بالا به ارایه بیتی تبدیل میکنیم .

```
In [82]: twopair_count = Counter()
compine2array = combinations(onepair_support_list, 2)
for i, j in compine2array:
    m = hash_functions[0](int(i))
    n = hash_functions[0](int(j))
    if (hash_table[m,n]==True):
        for item in main_array:
            if(i in item):
                if(j in item):
                    twopair_count[i + '|' + j] += 1
```

```
In [83]: twopair_count
```

```
                '39|379': 429,
                '39|389': 544,
                '39|396': 436,
                '39|398': 320,
                '39|405': 500,
                '39|407': 418,
                '39|413': 1130,
                '39|420': 340,
                '39|425': 396,
                '39|426': 334,
                '39|438': 1260,
                '39|23': 512,
                '39|441': 393,
                '39|449': 372,
                '39|464': 412,
                '39|475': 1500,
                '39|479': 580,
                '39|488': 389,
                ...})
```

در مرحله بعد دودویی های تولید شده را تک تک هش کرده و بررسی میکنیم که در ان سطر و ستون مقدار 1 وجود دارد . و بعد از شمارش مقادیر خروجی تمام دودویی های ان را شمارش میکنیم .

این مراحل را میتوانیم تا جایی ادامه دهیم که دیگر زوجی وجود نداشته باشد .

تمرین 2 :

(پژوهش)

۲-(پژوهش)

الف) یکی از بهبودهایی که روی تعریف کلاسیک مجموعه اقلام پرتکرار برای بعضی کاربردها ضروی است، اضافه شدن وزن به اقلام درون مجموعه‌ها است، به عبارت دیگر مجموعه اقلام پرتکرار وزن‌دار[۲] .

چرا وزن؟ بدیهی است که در بسیاری از کاربردهای دنیای واقعی، اقلام درون مجموعه‌ها اهمیت و ارزش یکسانی ندارند. بعنوان مثال، سهام شرکت‌های مختلف درون یه یک سبد سهام را در نظر بگیرید که هر کدام نرخ سود متفاوتی در یک روز معاملاتی دارند. با بررسی یک تحقیق کاربردی معتبر، الگوریتم شناسایی مجموعه اقلام پرتکرار وزن‌دار و همچنین کاربردی که از آن استفاده کرده را با جزئیات کافی به ما نیز معرفی کنید.

ب) شاید برایتان جالب باشد که یک زمینه کاربردی و تحقیقاتی به نام **کاوش مجموعه اقلام کم تکرار**[۳]!!!! نیز وجود دارد. کمی در این مورد تحقیق کرده و الگوریتم‌ها و کاربردهای آن را گزارش کنید.

افزوده شدن وزن به کالا ها باعث میشود درجه اهمیت هر کالا مشخص شود ممکن است کالا هایی باشد با این که در تمام خرید ها وجود دارد اما از دید فروشگاه درجه اهمیت بالایی دارد برای مثال ادامس و یا بیسکوییت گلرنگ که به عنوان پول خرد نیز از ان ها استفاده میشود . در این مورد مقالاتی وجود دارد که به عنوان اقلام پرتکرار وزن دار میتوان از ان ها نام برد .

A Weighted Frequent Itemset MiningAlgorithm for Intelligent Decisionin Smart Systems

مقاله بالا یکی از مقالات این زمینه میباشد .

تصمیمات هوشمند یکی از فواید پیشرفت تکنولوژی است یکی از این روش ها برای تصمیم گیری هوشمند یافتن اقلام پرتکرار است و همچنین در نظر گرفتن وزن برای اقلام نیز یکی از بهبود ها در این زمینه بود که برای تصمیم گیری خیلی کاربرد مفیدی داشت .

در الگوریتم حالت بدون وزن دار یک مشکل داشتیم ان هم بسته بودن با گذشت زمان بود این به ان معنی است که بعد از مدتی که تمام کالاها در سبد های زیادی تکرار شد از ان به بعد فقط نتیجه یکسانی از یافتن اقلام پر تکرار گرفته میشود برای حل این مشکل افزودن وزن به کالاها نتیجه خوبی داد .

در این مقاله در مورد روش های MBP و بهبود یافته ان IMBP اورده شده که این ها برای روش های بدون وزن هستند ولی بحث جالب تر این مقاله در مورد الگوریتم های وزن دار هستند مانند :

- WAR (Weighted Associate Rule Algorithm)
- WARM (Weighted Associate Rule Minnig Algorithm
- WFIM (Weighted Frequent Item Mining Algorithm)
- WMFP_SW (Weighted Maximal Frequent Pattern data Stream Base)
- MWS
- WEP

و تعدادی الگوریتم دیگر نیز در این مقاله مقایسه شده است .

بخش دوم سوال مربوط به مجموعه اقلام کم تکرار است .

برای این روش نیز مقالاتی وجود دارد که یکی از ان ها را در کاربرد های این روش توضیح میدهم .

اگرچه ، سودمندی بالا مهم است ، اما تنها تصمیم برای تصمیم گیری در مورد استراتژی های کارآمد تجاری مانند پیشنهادات تخفیف نیست. در نظر گرفتن الگوی اجناس بر اساس فرکانس و همچنین ابزار برای پیش بینی مجموعه های سودآور بیشتر بسیار مهم است. به عنوان مثال ، در یک سوپرمارکت یا رستوران ، نوشیدنی هایی مانند شامپاین یا شراب ممکن است سود (سود) بالایی داشته باشد ، اما در مقایسه با سایر نوشیدنی ها مانند نوشابه و آبجو نیز فروش کمتری دارد. در مطالعات قبلی مشاهده شده است که افرادی که از سوپرمارکت شیر ، نان یا پوشک می خرند ، تمایل به خرید آبجو یا نوشابه نیز دارند. اما اقلامی مانند شیر ، پوشک ، آبجو یا سودا در مقایسه با نوشیدنی هایی مانند شامپاین یا شراب ، سود کمتری (ارزش سود) تولید می کنند. اگر مواردی مانند شامپاین یا شراب را که دارای کاربرد بالا اما فرکانس کمتری هستند ، با کالاهای اغلب فروخته شده مانند شیر ، پوشک یا آبجو ترکیب کنیم ، می توانیم با ارائه پیشنهادهای تخفیف شامپاین یا شراب ، سود معامله را افزایش دهیم. در این مقاله ، ما در حال ادغام مجموعه های مورد با فرکانس پایین با مجموعه های با فرکانس بالا هستیم ، هر دو دارای کاربرد کم یا زیاد هستند ، و قوانین ارتباط متفاوتی را برای این ترکیب از مجموعه ها ارائه می

دهیم. به این ترتیب ، ما می توانیم اندازه گیری دقیق تری در مورد استخراج الگو برای استراتژی های مختلف تجاری ایجاد کنیم .

از این روش برای کالاهایی که سود بالایی دارند ولی فروش کم تری دارند برای پیشنهاد های ویژه و اعمال تخفیف برای خرید بیشتر این کالا توسط افراد می باشد .

معمولا بهترین روش استفاده این روش استفاده هم زمان این الگوریتم ها با هم است که در مقاله زیر به ترکیب این دو پرداخته است .

Mining Association rules for Low-Frequency itemsets

- Jimmy Ming-Tai Wu ,
- Justin Zhan ,
- Sanket Chobe

LUIM: New Low-Utility ItemsetMining Framework

- NAJI ALHUSAINI1,
- SALEEM KARMOSHI2,
- AMMAR HAWBANI1,
- LI JING1,
- ABDULLAH ALHUSAINI3,
- AND YASER AL-SHARABI