گزارش تمرین سوم مبانی داده‌کاوی و کاربردهای آن (۲۱۰۱۹)

استاد: دکتر مانا مس‌کار
دستیار آموزشی: علیرضا دهقان
اعضای گروه: صبا عبدی (۴۰۱۱۰۴۲۷۶)، آوا صدیقی (۴۰۱۱۱۰۱۵۹)
دانشکده مهندسی صنایع

● سوال 1 (20 نمره)

دیتاست این سوال شامل تراکنش‌های یک فروشگاه آنلاین بریتانیایی است.

با استفاده از پارتیشن بندی داده‌ها به چهار قسمت، به سوالات زیر پاسخ دهید:

➢ با min_support=0.05، محصولاتی که اغلب با هم خریداری می‌شوند را با Apriori پیدا کنید.

پیش از شروع پاسخگویی به سوالات، باید داده‌ها را بررسی کنیم. پیش از هر نوع اقدام، ابتدا نام ستون‌ها را استاندارد می‌کنیم تا در استفاده از آن‌ها به مشکل برنخوریم. سپس با استفاده از دستور info تلاش می‌کنیم تا دید مناسبی از داده‌ها به دست بیاوریم. متوجه می‌شویم که در ستون‌های description و customer_id مقادیر Nan داریم. به دلیل اینکه صرفا محصولات موجود در سبد خرید برای مهم است و هر سبد خرید با تاریخ و زمان خرید مشخص شده است، به ستون‌های customer_id و country نیازی نداریم. سپس حروف الفبای اضافی موجود در stock_code و invoice_No را حذف می‌کنیم و فرمت تاریخ و زمان را به datetime تبدیل می‌کنیم.

```
online_retail_copy['stock_code'] = online_retail_copy['stock_code'].astype(str).str.replace(r'[a-zA-Z]', '', regex=True)
online_retail_copy['invoice_No'] = online_retail_copy['invoice_No'].astype(str).str.replace(r'^[a-zA-Z]', '', regex=True)
online_retail_copy['invoice_date'] = pd.to_datetime(online_retail_copy['invoice_date'])
```

پس از این، تمام محصولات خرید را lowercase کرده و عمل strip را برای حذف فاصله‌های اضافی آن‌ها انجام می‌دهیم. سپس موارد غیرصحیح مثل مرجوعی را از داده خود حذف می‌کنیم.

```
online_retail_copy['description'] = online_retail_copy['description'].str.lower()
online_retail_copy['description'] = online_retail_copy['description'].str.strip()
invalid_desc = ['postage', 'adjustment', 'delivery', 'discount', 'manual']
```

در داده‌های باقی مانده، مقدار description بعضی رکوردها خالی است. برای پر کردن این رکوردها می‌توان از stock_No استفاده کرد. به این صورت که رکوردهایی که کد کالا دارند را متناسب با کد کالا و کالای مربوط به آن کد، که آن کالا در دیگر سطرها مشخص است، پر می‌کنیم. در نهایت نیز سفارش‌هایی که مقدار quantity مثبت دارند را فیلتر کرده و در نظر می‌گیریم. لازم به ذکر است که برای پاسخ به تمامی بخش‌های این سوال از داده‌های تمیز شده ذکر شده استفاده می‌کنیم.

برای پاسخ به خواسته سوال یک، ابتدا متناسب با invoice_No و invoice_date، که برای هر سفارش یکتا هستند، هر سفارش را شناسایی کرده و سبد خرید آن را تشکیل می‌دهیم. سپس داده‌ها را به ۴ پارتیشن تقسیم می‌کنیم و سپس الگوریتم apriori را بر روی هر پارتیشن اجرا می‌کنیم:

```
partitions = np.array_split(basket, 4)

# 2. تابع کمکی برای one-hot encoding و Apriori در هر پارتیشن
def get_local_freq(df_part, min_support=0.5):
    # تبدیل لیست کالاها به ماتریس یکگره ای
    ohe = df_part['products'].str.join('|').str.get_dummies('|')
    # اجرای Apriori
    return apriori(ohe, min_support=min_support, use_colnames=True)

# استخراج آیتم‌ست‌های مکرر محلی
local_freq_list = [get_local_freq(p, min_support=0.0005) for p in partitions]
local_freq_list
```

در مرحله بعد، تمامی آیتم‌های پیدا شده در هر پارتیشن را در نظر گرفته و آن‌ها را به صورت global می‌سنجیم:

```
# 3. ادغام همه‌ی آیتم‌ست‌های مکرر محلی برای ساخت نامزدهای کلی
candidates = pd.concat(local_freq_list)['itemsets'].drop_duplicates().tolist()

# 4. اسکن نهایی: شمارش نامزدها روی کل داده
ohe_full = basket['products'].str.join('|').str.get_dummies('|')
# محاسبه‌ی فرکانس واقعی نامزدها
global_counts = []
n_transactions = len(basket)
for itemset in candidates:
    mask = ohe_full[list(itemset)].all(axis=1)
    support = mask.sum() / n_transactions
    if support >= 0.0005:
        global_counts.append((frozenset(itemset), support))
```

نکته قابل توجه در این قسمت این است که با مقدار ساپورت ۰.۰۰۰۵ گوگل کولب به ارور long time برخورده و کل نوت‌بوک را ری‌استارت می‌کند. به دلیل نداشتن پاسخ این قسمت، در به دست آوردن خروجی قسمت‌های بعد نیز دچار مشکل خواهیم شد. زیرا که جواب بخش‌ها به یکدیگر مرتبط است.

- قوانین انجمنی با حداقل confidence=0.4 را استخراج کنید.

همانگونه که ذکر شد، در کد این قسمت باید از جواب بخش بالا استفاده کنیم که به دلیل ذکر شده، آن را نداریم. در این بخش تنها لازم است که از دستور association rule استفاده کنیم و frequent_itemsetها را به عنوان ورودی در نظر می‌گیریم:

```
# تغییر نام ستون itemset به itemsets
freq_itemsets = freq_itemsets.rename(columns={'itemset': 'itemsets'})

# حالا می‌توانیم قواعد انجمنی را استخراج کنیم
rules = association_rules(freq_itemsets, metric="confidence", min_threshold=0.0005)

# مرتب‌سازی و نمایش
rules = rules.sort_values(['confidence', 'lift'], ascending=[False, False]).reset_index(drop=True)
print(rules[['antecedents','consequents','support','confidence','lift']])
```

نکته قابل توجه در این قسمت این است که با مقدار ساپورت ۰.۰۰۰۵ گوگل کولب به ارور long time برخورده و کل نوت‌بوک را ری‌استارت می‌کند. لذا output به ما نمی‌دهد. همچنین جواب‌های این سه بخش به ترتیب به یکدیگر وابسته هستند.

- اگر بخواهید تخفیف گروهی ارائه دهید، کدام ترکیب محصولات را پیشنهاد می‌کنید؟

برای پاسخ به این قسمت ابتدا آیتم‌ست‌های پرتکرار، که باید از بخش اول به دست آیند را در نظر گرفته و مواردی از آن‌ها که طولی بزرگتر مساوی دو دارند را در نظر می‌گیریم.

```
# 1) برترین آیتم‌ست‌ها بر اساس support (اندازه ≥ 2)
top_by_support = (freq_itemsets[
                    freq_itemsets['itemsets'].apply(lambda s: len(s) >= 2)]
                  .sort_values('support', ascending=False)
                  .head(5))
print("Top 5 itemsets by support:")
```

سپس با استفاده از معیار lift بهترین این ترکیبات پرتکرار را پیدا می‌کنیم.

```
# 2) برترین قوانین بر اساس lift
top_by_lift = (rules[
                 rules['antecedents'].apply(lambda s: len(s) >= 1) &
                 rules['consequents'].apply(lambda s: len(s) >= 1)]
               .sort_values('lift', ascending=False)
               .head(5)[['antecedents','consequents','support','confidence','lift']])
print("\nTop 5 association rules by lift:")
```

سپس تلاش می‌کنیم تا با استفاده از این ترکیبات، باندل‌های متفاوتی را تشکیل می‌دهیم و سپس با استفاده از معیارهایی مثل lift و confidence بهترین این باندل‌ها را انتخاب می‌کنیم.

```
bundle_suggestions = []
for _, row in top_by_lift.iterrows():
    combo = set(row['antecedents']) | set(row['consequents'])
    bundle_suggestions.append((frozenset(combo), row['lift'], row['confidence']))

print("\nBundle suggestions (from top rules):")
for combo, lift, conf in bundle_suggestions:
    print(f"{set(combo)}  →  lift={lift:.2f}, confidence={conf:.2f}")
```

نکته قابل توجه در این قسمت این است که با مقدار ساپورت ۰.۰۰۰۵ گوگل کولب به ارور long time برخورده و کل نوت‌بوک را ری‌استارت می‌کند. لذا output به ما نمی‌دهد. همچنین جواب‌های این سه بخش به ترتیب به یکدیگر وابسته هستند.

- آیا محصولاتی هستند که فقط در آخر هفته (شنبه و یکشنبه) باهم خریداری شوند؟

برای این کار، ابتدا به دیتاست basket ستونی تحت عنوان weekday اضافه می‌کنیم تا بتوان آخر هفته را شناسایی کرد. سپس سبدهای خرید آخر هفته را برای ادامه کار انتخاب می‌کنیم:

```
basket['weekday'] = pd.to_datetime(basket['invoice_date']).dt.weekday
ohe_full= basket['products'].str.join('|').str.get_dummies('|')
weekend_mask = basket['weekday'].isin([5,6])
weekday_mask = ~weekend_mask
```

سپس مشابه حالات قبل و با استفاده از پارتیشن‌بندی و apriori، محصولات موردنظر را پیدا می‌کنیم:

```
# 1) تقسیم به 4 پارتیشن
b_parts = np.array_split(basket, 4)
o_parts = np.array_split(ohe_full, 4)

# 2) استخراج جفت‌های محلی (آخر هفته و روز کاری) در هر پارتیشن
local_wknd = []
local_wkdy = []

for b, o in zip(b_parts, o_parts):
    m_wknd = b['weekday'].isin([5,6])
    # برای سرعت آپریوری با max_len=2
    wknd_sets = apriori(o[m_wknd], min_support=0.0005, use_colnames=True, max_len=2)['itemsets']
    wkdy_sets = apriori(o[~m_wknd], min_support=0.0005, use_colnames=True, max_len=2)['itemsets']
    # نگه‌داشتن فقط جفت‌ها (len==2)
    local_wknd.append({frozenset(s) for s in wknd_sets if len(s)==2})
    local_wkdy.append({frozenset(s) for s in wkdy_sets if len(s)==2})
```

نکته قابل توجه در این قسمت این است که با مقدار ساپورت ۰.۰۰۰۵ گوگل کولب به ارور long time برخورده و کل نوت‌بوک را ری‌استارت می‌کند. لذا output به ما نمی‌دهد.

- سوال 2 (20 نمره)

بر اساس دیتاست موجود در فایل زیپ و از طریق روش FPGrowth به سوالات زیر پاسخ دهید:

➢ دمای بالا ($\geq 30°C$) و رطوبت بالا ($\geq 80\%$) را به عنوان آیتم در نظر بگیرید.

پیش از هر کاری ابتدا داده‌ها را خوانده و با استفاده از دستور info آن‌ها را بررسی می‌کنیم:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
```

متناسب با این خروجی، داده NaN نداریم و می‌توانیم کار با داده‌ها را آغاز کنیم. ابتدا دمای بالای ۳۰ درجه و رطوبت بالای ۸۰ درصد را در دیتاست خود مشخص می‌کنیم. سپس رکوردهای موجود را متناسب با این دو threshold بررسی می‌کنیم تا تراکنش‌های صحیح را بیابیم.

```
transactions = weather_history_q1.apply(
    lambda row: ['high_temp'] if row['high_temp'] else []
                + ['high_hum']  if row['high_hum']  else [],
    axis=1
).tolist()
transactions
```

سپس تراکنش‌های موردنظر را encode کرده و با استفاده از الگوریتم fpgrowth آیتم‌های پرتکرار را پیدا می‌کنیم. در نهایت به نتیجه زیر می‌رسیم:

	support	itemsets
0	0.477279	(high_hum)
1	0.028688	(high_temp)

بنابراین هرکدام از موارد خواسته شده به تنهایی پرتکرار هستند ولی مجموعه آن‌ها پرتکرار نیست.

- با min_support=0.2، الگوهای مکرر بین شرایط آب‌وهوایی را پیدا کنید.

برای پاسخ به این به سوال، ابتدا باید تراکنش‌های موردنظر این سوال را حساب کرده و در لیستی قرار دهیم:

```
def make_transaction(row):
    items = []
    items.append(str(row['Summary']))
    if pd.notna(row['Precip Type']):
        items.append(str(row['Precip Type']))
    return items

transactions_q2 = weather_history_q2.apply(make_transaction, axis=1).tolist()
```

قسمتی از خروجی:

```
['Breezy and Mostly Cloudy', 'rain'],
['Overcast', 'rain'],
['Mostly Cloudy', 'rain'],
```

سپس، مانند بخش پیشین، تراکنش‌های موردنظر را encode کرده و با استفاده از الگوریتم fpgrowth الگوهای مکرر را پیدا می‌کنیم. قسمتی از خروجی به شرح زیر است:

	support	itemsets
0	0.883581	(rain)
1	0.329000	(Partly Cloudy)
2	0.291271	(Mostly Cloudy)
3	0.172073	(Overcast)
4	0.074109	(Foggy)
5	0.005350	(Breezy and Mostly Cloudy)
6	0.112905	(Clear)
7	0.004002	(Breezy and Partly Cloudy)
8	0.005474	(Breezy and Overcast)

- سوال 3 (20 نمره)

بیمارستان فیروزآبادی قصد دارد ریسک بستری مجدد (Readmission) بیمارانش را بر اساس درمان های انجام شده تجزیه و تحلیل کند. مجموعه داده حاضر شامل 10000 پرونده بیمار (TID) و 15 درمان رایج (آیتم‌ها) است. درمان‌ها به $(T_1, T_2, T_3, \dots, T_{15})$ برچسب‌گذاری شده‌اند. هر درمان یک هزینه‌ای دارد و بیمارستان می‌خواهد ارتباط بین درمان‌هایی را که بستری مجدد را پیش‌بینی می‌کنند و در عین حال به محدودیت‌های بودجه پایبند هستند، شناسایی کند.

- از TID_set برای T_1 و داده‌های پذیرش مجدد برای انجام این موارد استفاده کنید: یک جدول دو در دو بسازید و برای T_1 ⇒ Readmission با هفتاد درصد پذیرش مجدد، لیفت را محاسبه کنید. آماره مربع کای را محاسبه کنید.

گزارش تحلیل آماری برای درمان T1 و پذیرش مجدد بیماران در این تحلیل از داده‌های موجود در فایل Q3_Data.xlsx (Sheet2) و مجموعه بیماران مربوط به درمانT1 استفاده شده است تا تأثیر درمان T1بر احتمال پذیرش مجدد (Readmission) بررسی شود. هدف، ساخت یک جدول دو در دو (2×2 contingency table)، محاسبه Lift و انجام آزمون آماری Chi-Square برای ارزیابی وابستگی بین درمان و نتیجه است.

برای بررسی قدرت رابطه‌ی بین درمان T1 و پذیرش مجدد، از معیار Lift استفاده شد:

$$\frac{P(T1 \mid \text{Readmission})}{P(T1)} = \text{Lift}$$

اگر مقدار لیفت بزرگ‌تر از 1باشد، نشان‌دهنده‌ی این است که دریافت درمان T1 با پذیرش مجدد **همبستگی مثبت** دارد. اگر کوچک‌تر از ۱ باشد، ممکن است درمان T1 با کاهش پذیرش مجدد همراه باشد.

حال آزمون را تشکیل می‌دهیم:

فرض صفر: درمان T1 و پذیرش مجدد مستقل از یکدیگرند.

فرض مقابل: بین این دو وابستگی وجود دارد.

باید توجه شود که با توجه به آزمون فرض، می‌بایست در دستور زیر correction=False نوشته شود.

```
chi2, p_val, dof, expected = chi2_contingency(contingency_table, correction=False)
```

در نهایت خروجی کد به شرح زیر خواهد بود:

```
Contingency Table:
Readmission     0     1
T1
0            7997  1993
1               3     7

Lift for T1: 3.5000
Chi-square Statistic: 15.6406
P-value: 0.0001
```

که چون مقدار **p-value < 0.05** است، فرض صفر رد شده و نتیجه می‌گیریم که رابطه‌ی معناداری بین دریافت درمان T1 و پذیرش مجدد وجود دارد.

- با در نظر گرفتن $support(X) \geq 5$ پس از اعمال قید/محدودیت تمام آیتم‌ست‌های مکرر تکی را لیست کنید. آیتم‌ست‌های معتبر دوتایی و سه‌تایی را با استفاده از اشتراک TID-list تولید کنید.

با الگوریتم A-priori لیست‌های کاندید را تشکیل می‌دهیم و اگر شرایط مینیمم ساپورت را دارا باشند در لیست الگوهای مکرر قرار می‌گیرند.

```python
[6]  frequent_1_itemsets = {item for item, tids in item_tid.items() if len(tids) >= 5}
     print("Frequent 1-itemsets (support >= 5):")
     print(frequent_1_itemsets)

     # 2-item
     from itertools import combinations

     frequent_2_itemsets = {}

     for item1, item2 in combinations(frequent_1_itemsets, 2):
         tids1 = item_tid[item1]
         tids2 = item_tid[item2]
         intersection = tids1 & tids2
         if len(intersection) >= 5:
             frequent_2_itemsets[(item1, item2)] = intersection

     print("\nFrequent 2-itemsets:")
     for pair, tids in frequent_2_itemsets.items():
         print(f"{pair} => support: {len(tids)}")


     # 3-item
     frequent_3_itemsets = {}

     for item1, item2, item3 in combinations(frequent_1_itemsets, 3):
         tids1 = item_tid[item1]
         tids2 = item_tid[item2]
         tids3 = item_tid[item3]
         intersection = tids1 & tids2 & tids3
         if len(intersection) >= 4:
             frequent_3_itemsets[(item1, item2, item3)] = intersection
```

و حاصل چنین می‌شود:

```
Frequent 1-itemsets (support >= 5):
{'T3', 'T4', 'T7', 'T2', 'T1', 'T6'}

Frequent 2-itemsets:
('T3', 'T1') => support: 5
('T4', 'T1') => support: 5
('T7', 'T1') => support: 5
('T2', 'T1') => support: 6
('T1', 'T6') => support: 5

Frequent 3-itemsets:
('T3', 'T7', 'T1') => support: 4
```

هیچ الگوی مکرر ۳ آیتمی با ۵ ساپورت وجود ندارد و بیشترین ساپورت یک الگوی مکرر ۳ آیتمی ۴ است.

- تراکنش‌هایی که مجموع هزینه درمان‌ها بیش از ۳۰۰ دلار است را حذف کنید جدول جدید برای $T_1 \Rightarrow$ Readmission را بسازید. مقدار آماره مربع کای را مجدداً محاسبه کنید.

```
Patient 407: Total cost = $620
Patient 612: Total cost = $1175
Patient 101: Total cost = $1160
Patient 714: Total cost = $1245
Patient 203: Total cost = $680
Patient 816: Total cost = $1315
Patient 305: Total cost = $1470
Patient 919: Total cost = $1540
Patient 1020: Total cost = $845
Patient 509: Total cost = $1170
```

همهٔ تراکنش‌های هزینه‌ای بیش‌از ۳۰۰ دلار دارند و در این صورت هیچ تراکنشی با هزینهٔ کمتر از ۳۰۰ باقی نمی‌ماند.

اگر هم تراکنشی باقی می‌ماند آن‌ها را جدا می‌کردیم و جدول را برای آن‌ها تشکیل می‌دادیم. بر اساس نتیجهٔ جدول جدید می‌توانستیم تفسیری دربارهٔ اثر T1 در تراکنش‌های ارزان‌تر و ارتباط هزینهٔ درمان و اثربخشی درمان T1 با بازگشت بیمار ارائه دهیم.

- سوال 4 (15 نمره)

پایگاه داده بیمارستان میلاد علائم و تشخیص‌های بیمار را ثبت می‌کند. هر تراکنش نشان دهنده علائم بیمار (به عنوان مثال، {تب، سرفه، خستگی}) و تشخیص آنها است. فرض کنید مجموعه داده به صورت زیر باشد:

T1:{Fever, Cough, Headache, COVID}

T2:{Fever, Fatigue, COVID}

T3:{Cough, Headache, Influenza}

T4:{Fever, Cough, Headache, Fatigue, COVID}

T5:{Fever, Cough, Influenza}

با فرض (min_sup=2)، تمام الگوهای پرتکرار بسته (closed frequent patterns) و الگوهای حداکثری (max-patterns) را شناسایی کنید.

ابتدا ساپورت را برای آیتم‌ها بررسی می‌کنیم تا آیتم‌های پرتکرار را بیابیم:

آیتم‌های تکی:

۱. Fever: ۴
۲. Cough: ۴
۳. Headache: ۳
۴. Fatigue: ۲
۵. COVID: ۳
۶. Influenza: ۲

همهٔ آیتم‌های تکی از مینیمم ساپورت برخوردار هستند.

آیتم‌های دوتایی (C2):

۱. {Fever, Cough}: ۳
۲. {Fever, Headache}: ۲
۳. {Fever, Fatigue}: ۲
۴. {Fever, COVID}: ۳
۵. {Fever, Influenza}: ۱
۶. {Cough, Headache}: ۳
۷. {Cough, Fatigue}: ۱

۸. {Cough, COVID}: ۲
۹. {Cough, Influenza}: ۲
۱۰. {Headache, Fatigue}: ۱
۱۱. {Headache, COVID}: ۲
۱۲. {Headache, Influenza}: ۱
۱۳. {Fatigue, COVID}: ۲
۱۴. {Fatigue, Influenza}: ۰
۱۵. {COVID, Influenza}: ۰

با آیتم‌های باقی مانده آیتم‌های ۳ تایی را تشکیل می‌دهیم:

۱. {Fever, Cough, Headache}: ۲
۲. {Fever, Cough, COVID}: ۲
۳. {Fever, Headache, COVID}: ۲
۴. {Cough, Headache, COVID}: ۲
۵. {Fever, COVID, Fatigue}: ۲

آیتم‌های ۴ تایی:

۱. {Fever, Cough, Headache, COVID}: ۲

حال با استفاده از الگوهای تکرار شونده، الگوهای بسته را تشکیل می‌دهیم:

۱. {Fever, Cough, Headache, COVID}: ۲
۲. {Fever, COVID, Fatigue}: ۲
۳. {Fever, Cough}: ۳
۴. {Cough, Headache}: ۳
۵. {Cough, Influenza}: ۲
۶. {Fever, COVID}: ۳
۷. {Fever}: ۴
۸. {Cough}: ۴

حال الگوهای حداکثری را تشکیل می‌دهیم:

۱. {Fever, Cough, Headache, COVID}: ۲
۲. {Fever, COVID, Fatigue}: ۲
۳. {Cough, Influenza}: ۲

- توضیح دهید چرا الگوی {تب، سرفه، سردرد} یک الگوی بسته نیست.

الگوی {Fever, Cough, Headache} علیرغم آنکه یک الگوی تکرارشونده است یک الگوی بسته نیست چراکه مجموعه‌ای بزرگتر از آن وجود دارد که تکرارش برابر با تکرار مجموعهٔ {Fever, Cough, Headache} و برابر با ۲ است. به عبارتی این الگو در واقع یک زیرمجموعه از مجموعهٔ {Fever, Cough, Headache, COVID} با ساپورت ۲ است.

- *امتیازی* پیامدهای عملی استفاده از الگوهای بسته در مقابل الگوهای حداکثری را در تشخیص پزشکی بررسی کنید (مثلاً کاهش قوانین تکراری در مقابل شناسایی جامع علائم ترکیبی).

استفاده از الگوهای حداکثری می‌تواند به تصمیم گیری سریعتری بیانجامد به نحوی که اگر بطور تصادفی بیماری دقیقا همان علائم موجود در مجموعهٔ الگوهای حداکثری را داشته باشد می‌توان به سرعت و با احتمالی بالا دربارهٔ او تشخیص داد اما از طرفی احتمال آنکه علائم بیماری با الگوهای حداکثری تطابق داشته باشد احتمالی کمتر از آنکه با الگوهای بسته تطابق داشته باشد خواهد داشت. از آنجایی که الگوهای بسته بیشتراند در موارد بیشتری قابل استفاده هستند اما دقت کمتری خواهند داشت و ممکن است نیاز به پردازش یا انجام فرایندهای آماری باشد. در مجموع بهتر است ابتدا الگوهای حداکثری بررسی شود و در صورت نیاز به الگوهای بسته مراجعه کرد.

- سوال 5 (25 نمره)

یک پلتفرم استریمینگ قصد دارد سیستم پیشنهاد محتوای خود را با شناسایی الگوهای پرتکرار تماشای همزمان فیلم‌ها/سریال‌ها بهینه‌سازی کند. این الگوها باید بر اساس جلسات/زمان‌های تماشا در روزهای هفته (شنبه تا پنجشنبه) و پایان هفته (جمعه و شنبه) تفکیک شوند. همچنین، الگوها باید محدودیت‌های ژانری را رعایت کنند تا با توصیه‌های موضوعی همسو باشند. برای این منظور از این مجموعه داده و به طور خاص فایل‌های زیر استفاده کنید.

- `ratings.csv`: شامل امتیازات کاربران با زمانبندی (timestamp).

- `movies.csv`: نگاشت شناسه فیلم‌ها به ژانرها (مثلاً کمدی، مستند).

- باید در نظر داشته باشید مراحل پیش‌پردازش زیر برای انجام تسک‌ها مورد نیاز است.

۱. تبدیل timestamp های در `ratings.csv` به تاریخ-زمان. پیشنهاد می‌شود از کتابخانه‌ی `datetime` برای تبدیل استفاده کنید.

۲. طبقه‌بندی هر تراکنش (زمان/جلسه تماشای کاربر) به روزهای هفته (شنبه–پنجشنبه) یا پایان هفته (جمعه و شنبه).

۳. گروه‌بندی فیلم‌های تماشاشده توسط یک کاربر به عنوان یک تراکنش.

در ابتدا دو مجموعه داده‌ی اصلی پروژه بارگذاری می‌شوند:

۱. اطلاعات فیلم‌ها شامل شناسه، عنوان و ژانر فیلم‌ها (movies.csv)

۲. امتیازدهی کاربران شامل شناسه‌ی کاربر، شناسه‌ی فیلم، نمره‌ی داده شده و زمان ثبت امتیاز (ratings.csv)

در کد مربوط به این بخش (اوایل فایل)، ابتدا ساختار داده‌ها بررسی شده و سپس برای سهولت در تحلیل، نام ستون‌ها به صورت استاندارد بازنویسی شده‌اند. این اقدام باعث افزایش خوانایی و نظم در ادامه‌ی پروژه شده است.

در ادامه، زمان ثبت هر امتیاز (که در قالب timestamp ذخیره شده) به فرمت تاریخ و ساعت تبدیل شده است. از این تاریخ، دو ویژگی مهم استخراج می‌شود:

- تاریخ (date): برای شناسایی فعالیت کاربران در هر روز خاص
- نام روز هفته (day_name): جهت تشخیص نوع روز (مثل دوشنبه، جمعه، و غیره)

سپس بر اساس نام روز، روزها به دو دسته کلی تقسیم شده‌اند:

- Weekend: شامل روزهای جمعه و شنبه
- Weekdayشامل سایر روزها

➢ استخراج الگوهای پرتکرار با محدودیت‌های ژانری (پیشنهاد می‌شود از کتابخانه‌ی `mlxtend.frequent_patterns` استفاده کنید):

ابتدا باید تفکیک زمانی صورت بگیرد به صورتی که تراکنش‌ها به دو زیرمجموعه‌ی روزهای هفته و پایان هفته تقسیم شوند. سپس محدودیت‌های ژانری را باید لحاظ کنید به این صورت که در تراکنش‌های پایان هفته الگوها باید حداقل یک فیلم با ژانر "خانوادگی" یا "کودک" داشته باشند و در تراکنش‌های روزهای هفته الگوها باید حداقل یک فیلم با ژانر "مستند" یا "آموزشی" داشته باشند. همچنین، نیازمندی‌های استخراج از این قرار هستند: از الگوریتم FP-Growth برای یافتن الگوهای پرتکرار در هر زیرمجموعه استفاده کنید. از محدودیت‌های مختصر (succinct constraints) برای پیش‌فیلتر کردن ژانرها استفاده کنید (مثلاً فقط تراکنش‌های دارای ژانرهای خانوادگی/کودک برای پایان هفته). حداقل پشتیبانی برای هر زیرمجموعه ۵٪ است.

در این مرحله، تحلیل به صورت تراکنش‌محور انجام شده است. منظور از تراکنش، مجموعه‌ای از ژانرهای فیلم‌هایی است که یک کاربر خاص در یک روز مشخص تماشا کرده است. برای این منظور، داده‌ها بر اساس شناسه‌ی کاربر، تاریخ، و نوع روز گروه‌بندی شده‌اند. خروجی این بخش، لیستی از ژانرهای دیده شده در هر تراکنش است.

برای تمرکز دقیق‌تر روی نوع خاصی از محتوا، در این مرحله تنها برخی از تراکنش‌ها انتخاب شده‌اند:

- در آخر هفته، تنها تراکنش‌هایی انتخاب شده‌اند که شامل فیلم‌هایی با ژانرFamily یا Children باشند.
- در روزهای کاری، تنها تراکنش‌هایی انتخاب شده‌اند که حداقل شامل ژانرDocumentary یا Educational باشند.

هدف از این فیلترگذاری، بررسی عمیق رفتار کاربران در شرایط خاص است، نه صرفاً همه‌ی تراکنش‌ها.

این بخش در کد، با توابعی انجام شده که در آن‌ها بررسی می‌شود آیا لیست ژانرها شامل کلیدواژه‌های مورد نظر هست یا خیر.

برای جلوگیری از پراکندگی زیاد ژانرها و ساده‌سازی تحلیل، تنها تعدادی از ژانرهای مهم انتخاب شده‌اند تا در الگوریتم مورد بررسی قرار گیرند. این ژانرها عبارت‌اند از:

- Family
- Children
- Documentary
- Educational

- Drama
- Comedy
- Adventure

این محدودسازی باعث می‌شود الگوریتم FP-Growth سریع‌تر و دقیق‌تر عمل کرده و خروجی قابل تفسیرتری ارائه دهد.

در این مرحله، الگوریتم FP-Growth بر روی تراکنش‌های مربوط به دو بازه زمانی weekend و weekday اعمال شده است. FP-Growth یکی از معروف‌ترین الگوریتم‌ها برای استخراج الگوهای پرتکرار (frequent itemsets) از داده‌های تراکنشی است.

برای اجرای این الگوریتم:

۱. ابتدا تراکنش‌ها با استفاده از یک تکنیک به نام Transaction Encodingبه ماتریسی دودویی تبدیل می‌شوند که در آن، هر سطر نشان‌دهنده یک تراکنش و هر ستون نماینده‌ی یک ژانر خاص است (مقدار ۱ یعنی ژانر در تراکنش وجود دارد).

۲. سپس FP-Growth اجرا می‌شود تا مجموعه‌هایی از ژانرهایی که با حداقل فراوانی مشخص (مثلاً ۵٪ کل تراکنش‌ها) با هم ظاهر شده‌اند، شناسایی شود.

این الگوریتم برای هر دو دسته‌ی روزهای کاری و آخر هفته به صورت جداگانه اجرا شده است (بخش‌های پایانی کد).

Weekend:

در نتایج مربوط به آخر هفته، مشاهده شده که ژانرهایی مانند Family، Children، Comedyو Adventureبا همدیگر به‌طور مکرر ظاهر می‌شوند. این الگو نشان‌دهنده‌ی تمایل کاربران به تماشای فیلم‌های خانوادگی و سرگرم‌کننده در روزهای تعطیل است.

Weekday:

در روزهای کاری، الگوهایی با محتوای آموزنده‌تر دیده می‌شوند. ترکیب‌هایی مانند Documentary + Dramaو Educational Documentary +بیشتر تکرار شده‌اند. این رفتار می‌تواند نشانه‌ای از تمایل کاربران به تماشای محتوای آموزشی و جدی در روزهای عادی باشد.

- به شکل زیر Association Rules ها را به دست بیاورید. محدودیت‌های Min confidence: 65% و Min lift: 1.3 را لحاظ کنید.

Weekend: {Family_Movie, ...} → {Adventure_Movie}

Weekday: {Documentary, ...} → {Drama}

بر اساس لیفت برای هر دو زیرمجموعه ۵ قانون برتر را مقایسه کنید و تفاوت‌های الگوها بین روزهای هفته و پایان هفته را مشخص کنید.

کتابخانهٔ ذکر شده را فراخوانی می‌کنیم.

```python
from mlxtend.frequent_patterns import association_rules

# Weekend rules
weekend_rules = association_rules(frequent_patterns_weekend, metric="confidence", min_threshold=0.65)
filtered_weekend_rules = weekend_rules[weekend_rules.lift >= 1.3]
top5_weekend = filtered_weekend_rules.nlargest(5, 'lift')

print("Top 5 Association Rules - Weekend:")
print(top5_weekend[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Weekday rules
weekday_rules = association_rules(frequent_patterns_weekday, metric="confidence", min_threshold=0.65)
filtered_weekday_rules = weekday_rules[weekday_rules.lift >= 1.3]
top5_weekday = filtered_weekday_rules.nlargest(5, 'lift')

print("Top 5 Association Rules - Weekday:")
print(top5_weekday[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

در این کد، هدف استخراج ۵ قانون برتر انجمنی برای روزهای هفته (Weekday) و آخر هفته (Weekend) است. برای این کار از الگوریتم Association Rules با معیار confidence استفاده شده است.

۱. **تولید قوانین انجمنی**

با استفاده از تابع association_rules از کتابخانهٔ mlxtend قوانین انجمنی از frequent_patterns_weekend و frequent_patterns_weekday استخراج شده‌اند، به‌طوری که مقدار confidence حداقل ۰.۶۵ باشد.

۲. **فیلتر بر اساس معیار lift**

از میان تمام قوانین، تنها قوانینی نگه داشته می‌شوند که lift آن‌ها **بزرگ‌تر یا مساوی ۱.۳** باشد. این کار برای اطمینان از معنادار بودن رابطه بین اقلام انجام می‌شود.

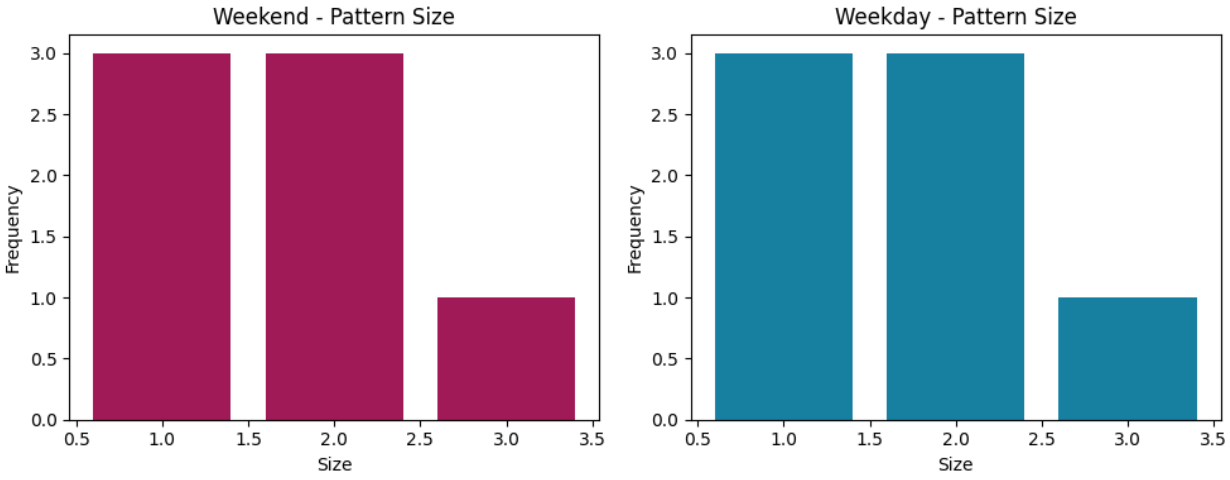
۳. **انتخاب ۵ قانون برتر**

سپس ۵ قانون با **بالاترین مقدار lift** انتخاب می‌شود که معرف قوی‌ترین روابط بین اقلام هستند.

در نهایت خروجی کد به شرح زیر است:

```
Top 5 Association Rules - Weekend:
             antecedents      consequents   support  confidence      lift
6          (Documentary)  (Drama, Comedy)  0.135043    0.774510  1.696960
5  (Comedy, Documentary)          (Drama)  0.135043    0.940476  1.549799
2          (Documentary)          (Drama)  0.158974    0.911765  1.502486
4   (Drama, Documentary)         (Comedy)  0.135043    0.849462  1.461575
3          (Documentary)         (Comedy)  0.143590    0.823529  1.416955
Top 5 Association Rules - Weekday:
             antecedents           consequents   support  confidence      lift
7  (Comedy, Documentary)               (Drama)  0.411546    0.902041  1.368350
2                (Drama)              (Comedy)  0.527002    0.799435  1.349989
3               (Comedy)               (Drama)  0.527002    0.889937  1.349989
6   (Drama, Documentary)              (Comedy)  0.411546    0.797834  1.347286
8                (Comedy)  (Drama, Documentary)  0.411546    0.694969  1.347286
```

- در این بخش مطلوب است که اعتبارسنجی تحلیلی، محاسبه و مصورسازی را به عمل بیاورید، برای این امر توزیع طول الگوهای پرتکرار برای هر دو زیرمجموعه را رسم کنید. نقشه حرارتی (heatmap) هم‌زمانی ژانرها برای تراکنش‌های پایان هفته و روزهای هفته را ایجاد کنید. تأثیر محدودیت‌های زمانی (روزهای هفته/پایان هفته) بر تنوع و اطمینان قوانین را تحلیل کنید.



در این تحلیل، توزیع اندازه‌ی الگوهای پرتکرار استخراج‌شده از داده‌های آخر هفته و روزهای کاری با هم مقایسه شده‌اند.

- در **آخر هفته**، اغلب الگوها شامل **۲ یا ۳ آیتم** هستند. این موضوع نشان می‌دهد که کاربران در تعطیلات آخر هفته تمایل دارند محتواهایی با **ترکیب‌های محدودتر و مشخص‌تر از ژانرها** (مثلاً درام + کمدی)تماشا کنند.

- در **روزهای کاری**، اگرچه الگوهای کوچک همچنان غالب هستند، اما توزیع اندازه‌ها **پراکندگی بیشتری** دارد و الگوهای بزرگ‌تر (تا اندازه ۴ یا ۵) نیز به‌صورت قابل توجهی ظاهر می‌شوند .این می‌تواند نشان‌دهنده‌ی **تنوع بیشتر در رفتار تماشای کاربران در طول هفته** باشد.
- **ماتریس همزمانی** ژانرها نیز تایید کرد که در روزهای هفته میزان همزمانی ژانرها بالاتر و رفتار کاربران متنوع‌تر است، در حالی که در آخر هفته تمرکز بر ژانرهای خاص‌تر و کمتر بودن هم‌زمانی‌ها مشهود بود.
- به طور کلی، با افزایش اندازه‌ی الگو، **فراوانی آن کاهش می‌یابد**، که در راستای ماهیت الگوریتم‌های کشف الگوی پرتکرار است.

در مجموع، تفاوت در توزیع نشان می‌دهد که **ترکیب ژانرهای منتخب کاربران در آخر هفته ساده‌تر، متمرکزتر و قابل پیش‌بینی‌تر** بوده و قوانین انجمنی قوی‌تر اما محدودتری دارند، در حالی که **در روزهای هفته تنوع و پراکندگی بیشتری در ترجیحات کاربران** مشاهده می‌شود.