به نام خدا

دانشگاه صنعتی امیرکبیر

Amirkabir University
of Technology

تکلیف اول - تحلیل کلان داده ها

استادمربوطه: دکتر حقیر چهرقانی

نام: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

بهار ۱۴۰۲

فهرست:

بخش اول MapReduce

سوال اول

الف)

ضرب ماتریس ها استفاده از رویکرد MapReduce در دو مرحله به صورت زیر میباشد:

مرحله اول:

- فاز Map: برای ماتریس M1 با سطر i و ستون j جفت کلید و مقدار به صورت $(j,(M1,i,M1_{ij}))$ میباشد و برای ماتریس M2 با سطر j و ستون k جفت کلید و مقدار به صورت $(j,(M2,k,M2_{jk}))$ میباشد.
- فاز Reduce: در ورودی این مرحله به ازای هر کلید j، دو مقدار وجود دارد، یکی برای ماتریس اول که به صورت $(M1,i,M1_{ij})$ و برای ماتریس دوم که به صورت $(M2,k,M2_{jk})$ میباشد. که در نتیجه آنها در خروجی به کلید (i, k) و مقدار برابر $M1_{ij}*M2_{jk}$ تبدیل میشوند.

مرحله دوم (این مرحله بر روی خروجی حاصل از مرحله قبل اعمال میشود):

- فاز Map: برای هر عنصر ورودی با کلید (i، k) و مقدار v، این جفت کلید-مقدار را تولید کنید.
- فاز Reduce: در ورودی این فاز به ازای کلید (i, k) چند مقدار وجود دارد، خروجی حاصل جمع مقادیر برای کلید یکسان میباشد. که نشان دهنده مقدار سطر i , ستون k در ماتریس نتیجه میباشد

ب)

$M_1: \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$ $M_2: \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix}$

۱- هفت جفت خروجی تولید شده توسط mapper ها برای ماتریس M_1 :

$(1, (M_1, 1, 2))$, $(2, (M_1, 1, 3))$, $(1, (M_1, 2, 1))$, $(2, (M_1, 2, 2))$

" " " " " " M_2 :

$(1, (M_2, 1, 1))$, $(1, (M_2, 2, 4))$, $(2, (M_2, 1, 2))$, $(2, (M_2, 2, 3))$

ورودی mapper اول ۴ جفت خروجی و mapper دوم نیز ۴ جفت خروجی دارد.

ج)

۲) جفت خروجی و مقدار متناظر از دید reducer دوم:

$((1,1), 2)$	$((2,1), 1)$
$((1,1), 6)$	$((2,1), 4)$
$((1,2), 8)$	$((2,2), 4)$
$((1,2), 9)$	$((2,2), 6)$

۸ جفت خروجی با مقدار متناظر ورودی reducer در مرحله دوم می‌باشند.

سوال دوم

برای پیاده سازی این مسئله با استفاده از mapReduce از دو مرحله استفاده شده:

در مرحله اول از متد map برای نگاشت هر خط به لیستی از شناسه های کانال، سپس از متد flatMap برای نگاشت هر شناسه کانال به یک جفت کلید-مقدار استفاده شده، که در آن کلید یک جفت کانال است (این کار به صورتی انجام میشود که در هر سطر کانال اول با همه کانال هایی که آن را تبلیغ کرده اند به صورت جفت کلید ایجاد میشود) و مقدار آن 1 است. در قسمت reduce از reduceByKey برای جمع بندی تعداد هر جفت کانال استفاده شده است (برای اینکه هر تبادل دو بار محاسبه شده حاصل جمع بر دو تقسیم میشود).

در مرحله دوم، از متد flatMap برای نگاشت هر کانال در هر جفت به یک جفت کلید-مقدار استفاده شده که در آن کلید کانال و مقدار تعداد آن کانال است. سپس در قسمت reduce از reducByKey برای جمع‌بندی تعداد هر کانال استفاده می‌کنیم.

الف)

```
Channel 859 has been advertised 1933.0 times.
Channel 5306 has been advertised 1741.0 times.
Channel 2664 has been advertised 1528.0 times.
Channel 5716 has been advertised 1426.0 times.
Channel 6306 has been advertised 1394.0 times.
```

ب)

Filter for the key-value pair where the key is 1748

```
[8]  filtered_counts = total_counts.filter(lambda pair: "1748" in pair[0])

     # Sum up the counts for the filtered pair
     count_1748 = filtered_counts.map(lambda pair: pair[1]).sum()
     count_1748

     130.0
```

Filter for the key-value pair where the key is 5633

```
[9]  filtered_counts = total_counts.filter(lambda pair: "5633" in pair[0])

     # Sum up the counts for the filtered pair
     count_5633 = filtered_counts.map(lambda pair: pair[1]).sum()
     count_5633
```

```
30.0
```

Filter for the key-value pair where the key is 3469

```
filtered_counts = total_counts.filter(lambda pair: "3469" in pair[0])

# Sum up the counts for the filtered pair
count_3469 = filtered_counts.map(lambda pair: pair[1]).sum()
count_3469
```

```
119.0
```

بخش دوم Rule Association

سوال اول

برای یافتن آیتم های پرتکرار در ابتدا باید support هر آیتم را محاسبه کنیم. support برای یک آیتم درصد احتمال حضور آن آیتم درون سبد ها میباشد، که به صورت زیر میباشد:

support(i) = probability that item i is in a basket = 1/i

با توجه به اینکه حد آستانه یک درصد در نظر گرفته شده، مجموعه ای شامل یک آیتم پرتکرار است و مجموعه هایی با تعداد بالاتر را باید بررسی کنیم.
برای مجموعه دوتایی داریم:

Pair (1, 2): support(1, 2) = support(2) = 0.5
Pair (1, 3): support(1, 3) = support(3) ≈ 0.33
Pair (1, 4): support(1, 4) = support(4) = 0.25
Pair (1, 5): support(1, 5) = support(5) = 0.2
Pair (1, 6): support(1, 6) = support(6) ≈ 0.17
Pair (1, 7): support(1, 7) = support(7) ≈ 0.14
Pair (1, 8): support(1, 8) = support(8) ≈ 0.125
Pair (1, 9): support(1, 9) = support(9) ≈ 0.11
Pair (1, 10): support(1, 10) = support(10) = 0.1
Pair (2, 3): support(2, 3) ≈ 0.166
Pair (2, 4): support(2, 4) ≈ 0.125
Pair (2, 5): support(2, 5) = 0.1
Pair (2, 6): support(2, 6) ≈ 0.083
Pair (2, 7): support(2, 7) ≈ 0.071
Pair (2, 8): support(2, 8) ≈ 0.063
Pair (2, 9): support(2, 9) ≈ 0.056

Pair (2, 10): support(2, 10) = 0.05

Pair (3, 4): support(3, 4) ≈ 0.083

Pair (3, 5): support(3, 5) ≈ 0.066

Pair (3, 6): support(3, 6) ≈ 0.055

Pair (3, 7): support(3, 7) ≈ 0.048

Pair (3, 8): support(3, 8) ≈ 0.041

Pair (3, 9): support(3, 9) ≈ 0.037

Pair (3, 10): support(3, 10) ≈ 0.033

Pair (4, 5): support(4, 5) =0.05

Pair (4, 6): support(4, 6) ≈ 0.041

Pair (4, 7): support(4, 7) ≈ 0.035

Pair (4, 8): support(4, 8) ≈ 0.031

Pair (4, 9): support(4, 9) ≈ 0.027

Pair (4, 10): support(4, 10) ≈ 0.025

.....

Pair(10,9): support(10, 9) ≈ 0.01

با توجه به اینکه حد آستانه یک درصد در نظر گرفته شده، مجموعه ای شامل دو آیتم پرتکرار هستند، زیرا کمترین مقدار support برای جفت (10,9) میباشد که بزرگتر مساوی ۱ درصد است.
استفاده از این حد آستانه برای support نمیتوند برای سبد های شامل ۱ الی ۲ آیتم برای ما فیلتری ایجاد کند و باید حد آستانه بزرگتر در نظر گرفته میشد.
برای مجموعه سه تایی با حد آستانه کوچکتر از ۱ درصد، داریم:

(1,2,3)-(1,2,4)-(1,2,5)-(1,2,6)-(1,2,7)-(1,2,8)-(1,2,9)-(1,2,10)-(1,3,4)-(1,3,5)
(1,3,6)-(1,3,7)-(1,3,8)-(1,3,9)-(1,3,10)-(1,4,5)-(1,4,6)-(1,4,7)-(1,4,8)-(1,4,9)
(1,4,10)-(1,5,6)-(1,5,7)-(1,5,8)-(1,5,9)-(1,5,10)-(1,6,7)-(1,6,8)-(1,6,9)-(1,6,10)
(1,7,8)-(1,7,9)-(1,7,10)-(1,8,9)-(1,8,10)-(1,9,10)-(2,3,4)-(2,3,5)-(2,3,6)-(2,3,7)
(2,3,8)-(2,3,9)-(2,3,10)-(2,4,5)-(2,4,6)-(2,4,7)-(2,4,8)-(2,4,9)-(2,4,10)-(2,5,6)

(2,5,7)-(2,5,8)-(2,5,9)-(2,5,10)-(2,6,7)-(2,6,8)-(3,4,5)-(3,4,6)-(3,4,7)-(3,4,8)
(3,5,6)

در مجموعه بالا تمام آیتم های سه تایی شامل عنصر ۱ پرتکرار هستند.
برای مجموعه چهار تایی با حد آستانه کوچکتر از ۱ درصد، داریم:

(1,2,4,5)-(1,2,3,10)-(1,2,3,9)-(1,2,3,8)-(1,2,3,7)-(1,2,3,6)-(1,2,3,5)-(1,2,3,4)
(1,2,5,8)-(1,2,5,7)-(1,2,5,6)-(1,2,4,10)-(1,2,4,9)-(1,2,4,8)-(1,2,4,7)-(1,2,4,6)
(1,3,4,8)-(1,3,4,7)-(1,3,4,6)-(1,3,4,5)-(1,2,6,8)-(1,2,6,7)-(1,2,5,10)-(1,2,5,9)
(1,3,5,6)

همه سبد ها شامل آیتم ۱ می باشد.
مجموعه پنج تایی و با اندازه بزرگتر با حد آستانه کوچکتر از ۱ درصد، وجود ندارد.

می‌توان دریافت که برخی از مجموعه آیتم‌ها بیشتر از بقیه ظاهر می‌شوند، جزو آیتم‌های پرتکرار هستند. پرتکرارترین کالا، آیتم ۱ است که در تعداد زیادی از سبد های سه تایی و همه سبد های چهارتایی وجود دارد.بعد از آیتم ۱، آیتم ۲ بیشترین تعداد تکرار را در سبد ها دارد، به طور کلی، این مجموعه اقلام مکرر بینش هایی را در مورد رایج ترین ترکیبات اقلام در سبد ها ارائه می دهد که می تواند برای کاربردهای مختلف مانند تجزیه و تحلیل سبد بازار و سیستم های توصیه مفید باشد.
حد آستانه در نظر گرفته شده بهتر از در مراحل مختلف متفاوت باشد زیرا برای سبد های شامل ۱ و ۲ آیتم فیلتری روی داده ها ایجاد نکرد و در سبد های ۵ تایی به بالاتر با این میزان حد آستانه آیتم های پرتکراری وجود نداشت.

سوال دوم

نتیجه گسسته سازی و پیش پردازش داده ها:

برای انجام این کار ستون Time حذف شده و هر ستون با استفاده از quantile به ۲۰ قسمت تقسیم شده که هر قسمت در ستون مجزا قرار دارد و مقدار آن می تواند ۰ و یا ۱ باشد.

جدول حاصل شامل ۵۸۱ ستون می‌باشد.

```
data.head()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_discretization.py:216: FutureWarning: In version 1.3 onwards, subsample=2e5 will be used by default. Set subsample explicitly to
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_discretization.py:216: FutureWarning: In version 1.3 onwards, subsample=2e5 will be used by default. Set subsample explicitly to
  warnings.warn(
```

	Class	Amount_1	Amount_2	Amount_3	Amount_4	Amount_5	Amount_6	Amount_7	Amount_8	Amount_9	...	V28_11	V28_12	V28_13	V28_14	V28_15	V28_16	V28_17	V28_18	V28_19	V28_20
0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
4	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

5 rows × 581 columns

قوانین دو تایی به دلیل استفاده از quantile و اینکه توزیع داده ها میان ستون ها یکسان میباشد استخراج نشده، که مشکلی در حل مسئله ایجاد نمیکند.

جدول df برای قوانین سه تایی کاربرد دارد:

- مقدار count نشان دهنده تعداد دفعاتی است که ستون val1, val2 با یکدیگر ۱ هستند
- مقدار support_fraud تعداد دفعات است که ۱ بودن ستون val1,val2 باعث ایجاد داده fraud شده است
- مقدار support_non_fraud تعداد دفعات است که ۱ بودن ستون val1,val2 باعث ایجاد داده non_fraud شده است
- مقدار confidence_non_fraud حاصل تقسیم support_non_fraud بر count میباشد.

	val1	val2	count	support_fraud	support_non_fraud	confidence_non_fraud
0	Amount_1	V1_1	1074.0	18.0	1056.0	0.983240
1	Amount_1	V1_2	807.0	6.0	801.0	0.992565
2	Amount_1	V1_3	651.0	2.0	649.0	0.996928
3	Amount_1	V1_4	627.0	4.0	623.0	0.993620
4	Amount_1	V1_5	600.0	6.0	594.0	0.990000

میزان support با مشاهده مقدار describe از داده ها در هر مرحله و با تغییر دادن مقدار آنها به صورت تجربی است.

Fraud

فیلتر به کار گرفته شده برای استخراج داده fraud به صورت زیر میباشد:

برای اینکه تعداد داده های fraud نسبت به کل داده ها کم است باید برای جلوگیری از کوچک‌تر شدن confidence مقدار count را محدود کرد.

```
df_fraud = df[(df['count'] >=500.0) & (df['count'] <=1000.0)& (df['support_fraud']>30.0) ]
df_fraud
```

	Unnamed: 0	val1	val2	count	support_fraud	support_non_fraud	confidence_non_fraud
120	120	Amount_1	V7_1	691.0	45.0	646.0	0.934877
180	180	Amount_1	V10_1	888.0	55.0	833.0	0.938063
220	220	Amount_1	V12_1	710.0	52.0	658.0	0.926761
419	419	Amount_1	V21_20	507.0	33.0	474.0	0.934911
1199	1199	Amount_3	V4_20	777.0	100.0	677.0	0.871300
...	...	...	...	...	...	...	...
150809	150809	V20_19	V21_20	915.0	41.0	874.0	0.955191
153821	153821	V21_20	V24_12	783.0	34.0	749.0	0.956577
158529	158529	V24_3	V27_20	660.0	31.0	629.0	0.953030
158549	158549	V24_3	V28_20	646.0	32.0	614.0	0.950464
161809	161809	V26_18	V27_20	707.0	37.0	670.0	0.947666

268 rows × 7 columns

قوانین سه تایی استخراج شده برای داده fraud:

	Unnamed: 0	val1	val2	count	support_fraud	confidence_fraud	interest_fraud
120	120	Amount_1	V7_1	691.0	45.0	0.065123	0.063396
180	180	Amount_1	V10_1	888.0	55.0	0.061937	0.060209
220	220	Amount_1	V12_1	710.0	52.0	0.073239	0.071512
419	419	Amount_1	V21_20	507.0	33.0	0.065089	0.063361
1199	1199	Amount_3	V4_20	777.0	100.0	0.128700	0.126973
...	...	...	...	...	...	...	...
150809	150809	V20_19	V21_20	915.0	41.0	0.044809	0.043081
153821	153821	V21_20	V24_12	783.0	34.0	0.043423	0.041695
158529	158529	V24_3	V27_20	660.0	31.0	0.046970	0.045242
158549	158549	V24_3	V28_20	646.0	32.0	0.049536	0.047808
161809	161809	V26_18	V27_20	707.0	37.0	0.052334	0.050606

268 rows × 7 columns

پنج قوانین برتر سه تایی، با معیار confidence , interest برای داده fraud:

```
df_fraud.sort_values( by=["confidence_fraud","interest_fraud"],ascending=False)[:5]
```

	Unnamed: 0	val1	val2	count	support_fraud	confidence_fraud	interest_fraud
107990	107990	V12_1	V17_1	515.0	370.0	0.718447	0.716719
131110	131110	V16_1	V17_1	614.0	356.0	0.579805	0.578077
107930	107930	V12_1	V14_1	715.0	393.0	0.549650	0.547923
51530	51530	V4_20	V10_1	584.0	318.0	0.544521	0.542793
70090	70090	V7_1	V17_1	608.0	330.0	0.542763	0.541036

شرط روی قوانین ۴ تایی:

ایجاد فیلتر روی قوانین ۴ تایی

```
df_fraud4_filter = df_fraud4[(df_fraud4['count'] >=110) & (df_fraud4['support_fraud']>40.0) ]
df_fraud4_filter
```

	val1	val2	val3	count	support_fraud	confidence_fraud	interest_fraud
0	Amount_1	V7_1	V10_1	180.0	42.0	0.233333	0.231606
20	Amount_1	V10_1	V9_1	117.0	48.0	0.410256	0.408529
81	Amount_3	V4_20	V12_1	158.0	98.0	0.620253	0.618526
87	Amount_3	V4_20	V3_1	139.0	96.0	0.690647	0.688920
89	Amount_3	V4_20	V6_1	116.0	69.0	0.594828	0.593100
...	...	...	...	...	...	...	...
3284	V18_1	V19_20	V28_20	138.0	45.0	0.326087	0.324359
3304	V18_1	V20_20	V27_20	156.0	66.0	0.423077	0.421349
3306	V18_1	V20_20	V28_20	202.0	52.0	0.257426	0.255698
3322	V18_1	V27_1	V28_1	357.0	42.0	0.117647	0.115920
3325	V18_1	V27_20	V28_20	328.0	84.0	0.256098	0.254370

502 rows × 7 columns

استخراج قوانین برتر روی داده ۴ تایی:

```
[83] df_fraud4_filter.sort_values( by=["confidence_fraud","interest_fraud"],ascending=False)[:5]
```

	val1	val2	val3	count	support_fraud	confidence_fraud	interest_fraud
1672	V7_1	V12_1	V14_1	394.0	333.0	0.845178	0.843450
1264	V4_20	V10_1	V17_1	342.0	288.0	0.842105	0.840378
2202	V10_1	V17_1	V11_20	406.0	341.0	0.839901	0.838174
2204	V10_1	V17_1	V16_1	415.0	348.0	0.838554	0.836827
2203	V10_1	V17_1	V12_1	431.0	361.0	0.837587	0.835860

```
df_fraud4_filter.sort_values( by="support_fraud",ascending=False)[:5]
```

	val1	val2	val3	count	support_fraud	confidence_fraud	interest_fraud
2579	V12_1	V14_1	V17_1	458.0	365.0	0.796943	0.795216
2203	V10_1	V17_1	V12_1	431.0	361.0	0.837587	0.835860
2631	V12_1	V16_1	V17_1	440.0	355.0	0.806818	0.805091
2578	V12_1	V14_1	V16_1	450.0	352.0	0.782222	0.780495
2204	V10_1	V17_1	V16_1	415.0	348.0	0.838554	0.836827

Non_fraud

استخراج قوانین ۳ تایی و ایجاد فیلتر روی قوانین سه تایی:

```
df_non_fraud = df[(df['support_non_fraud'] >=3000.0) & (df['confidence_non_fraud']>0.9)]
df_non_fraud['interest_non_frad']= df_non_fraud['confidence_non_fraud']-pr_non_fraud
df_non_fraud['interest_non_frad'] = df_non_fraud['interest_non_frad'].abs()
df_non_fraud= df_non_fraud.drop(columns=['support_fraud'])
df_non_fraud
```

```
<ipython-input-48-1e161ea46127>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_non_fraud['interest_non_frad']= df_non_fraud['confidence_non_fraud']-pr_non_fraud
<ipython-input-48-1e161ea46127>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_non_fraud['interest_non_frad'] = df_non_fraud['interest_non_frad'].abs()
```

	Unnamed: 0	val1	val2	count	support_non_fraud	confidence_non_fraud	interest_non_frad
10091	10091	Amount_19	V1_12	3367.0	3367.0	1.000000	0.001727
10100	10100	Amount_19	V2_2	3173.0	3172.0	0.999685	0.001412
10477	10477	Amount_19	V20_19	3303.0	3298.0	0.998486	0.000214
10656	10656	Amount_20	V2_1	8137.0	8129.0	0.999017	0.000744
10716	10716	Amount_20	V5_1	5411.0	5382.0	0.994641	0.003632
...	...	...	...	...	...	...	...
159969	159969	V25_1	V28_20	3119.0	3104.0	0.995191	0.003082
161910	161910	V27_1	V28_1	5766.0	5722.0	0.992369	0.005903
161931	161931	V27_2	V28_2	3474.0	3471.0	0.999136	0.000864
162265	162265	V27_18	V28_16	4700.0	4700.0	1.000000	0.001727
162309	162309	V27_20	V28_20	5879.0	5745.0	0.977207	0.021066

146 rows × 7 columns

استخراج ۵ قانون برتر روی قوانین سه تایی:

```
df_non_fraud.sort_values( by=["confidence_non_fraud","interest_non_frad"],ascending=False)[:5]
```

	Unnamed: 0	val1	val2	count	support_non_fraud	confidence_non_fraud	interest_non_frad
10091	10091	Amount_19	V1_12	3367.0	3367.0	1.0	0.001727
17113	17113	V1_11	V27_18	3461.0	3461.0	1.0	0.001727
17669	17669	V1_12	V28_14	3786.0	3786.0	1.0	0.001727
19284	19284	V1_16	V2_3	3043.0	3043.0	1.0	0.001727
20337	20337	V1_17	V28_7	3089.0	3089.0	1.0	0.001727

```
df_non_fraud.sort_values( by=["support_non_fraud","confidence_non_fraud","interest_non_frad"],ascending=False)[:5]
```

	Unnamed: 0	val1	val2	count	support_non_fraud	confidence_non_fraud	interest_non_frad
10656	10656	Amount_20	V2_1	8137.0	8129.0	0.999017	0.000744
11035	11035	Amount_20	V20_20	7368.0	7346.0	0.997014	0.001258
69829	69829	V6_20	V24_20	7029.0	7029.0	1.000000	0.001727
22269	22269	V2_1	V20_20	7011.0	7001.0	0.998574	0.000301
107949	107949	V12_1	V14_20	6631.0	6629.0	0.999698	0.001426

ایجاد فیلتر روی قوانین ۴ تایی:

```
df_non_fraud4_filter = df_non_fraud4[(df_non_fraud4['support_non_fraud'] >=2000.0) & (df_non_fraud4['confidence_non_fraud']>0.9)]
df_non_fraud4_filter
```

	val1	val2	val3	count	support_non_fraud	confidence_non_fraud	interest_non_fraud
5	Amount_20	V2_1	V5_1	3349.0	3345.0	0.998806	0.000533
6	Amount_20	V2_1	V7_20	2468.0	2466.0	0.999190	0.000917
7	Amount_20	V2_1	V20_20	5690.0	5684.0	0.998946	0.000673
8	Amount_20	V2_1	V21_20	3216.0	3209.0	0.997823	0.000449
9	Amount_20	V2_1	V23_1	4258.0	4251.0	0.998356	0.000084
...	...	...	...	...	...	...	...
389	V8_1	V20_1	V21_20	2765.0	2748.0	0.993852	0.004421
395	V8_1	V21_1	V20_20	2145.0	2120.0	0.988345	0.009928
407	V9_20	V10_20	V20_20	2380.0	2380.0	1.000000	0.001727
413	V10_20	V20_20	V27_20	2106.0	2106.0	1.000000	0.001727
438	V20_20	V21_20	V23_1	2192.0	2160.0	0.985401	0.012871

66 rows × 7 columns

استخراج قوانین برتر روی داده ۴ تایی:

پنج قانون برتر چهار تایی برای داده non_fraud با معیار confidence و interest

```
df_non_fraud4_filter.sort_values( by=["confidence_non_fraud","interest_non_fraud"],ascending=False)[:5]
```

	val1	val2	val3	count	support_non_fraud	confidence_non_fraud	interest_non_fraud
66	V1_1	V2_20	V10_20	2839.0	2839.0	1.0	0.001727
155	V1_1	V9_20	V10_20	2667.0	2667.0	1.0	0.001727
170	V1_1	V10_20	V27_20	2046.0	2046.0	1.0	0.001727
229	V1_19	V23_18	V26_15	2352.0	2352.0	1.0	0.001727
233	V1_19	V25_6	V26_15	2319.0	2319.0	1.0	0.001727

```
df_non_fraud4_filter.sort_values( by=["support_non_fraud"],ascending=False)[:5]
```

	val1	val2	val3	count	support_non_fraud	confidence_non_fraud	interest_non_fraud
7	Amount_20	V2_1	V20_20	5690.0	5684.0	0.998946	0.000673
9	Amount_20	V2_1	V23_1	4258.0	4251.0	0.998356	0.000084
27	Amount_20	V20_20	V23_1	3966.0	3959.0	0.998235	0.000038
249	V2_1	V20_20	V21_20	3715.0	3705.0	0.997308	0.000964
250	V2_1	V20_20	V23_1	3506.0	3501.0	0.998574	0.000301

با مشاهده قوانین ۴ تایی برای داده fraud و دقت خوب آن میتوان داده هایی که در این مجموعه قرار میگیرند را fraud و در غیر اینصورت non_fraud در نظر گرفت.

صحت روی قوانین دو تایی روی داده fraud فقط دارای ۵ مقدار با confidence بالای ۰.۵ میباشدکه معیار خوبی نیست برای تشخیص داده fraud ولی با قوانین ۴ تایی با توجه به اینکه مقدار confidence ها حدود ۰.۸ نیز وجود دارد، می تواند معیار خوبی باشد.

بخش سوم Hashing Sensitive Locality

سوال اول

الگوریتم LSH تکنیکی برای جستجوی تقریبی نزدیکترین همسایه در فضاهای با ابعاد بالا است که میتواند محدودیت های زیر را داشته باشد:

High False Positive Rate: این الگوریتم می تواند نرخ FP بالایی داشته باشد، به این معنی که ممکن است نقاطی را که در واقع نزدیکترین همسایگان نیستند برگرداند، انجام این کار زمانی که دقت آن بسیار مهم است مشکل ساز باشد.

Parameter Tuning: برای دستیابی به نتایج خوب نیاز به تنظیم دقیق پارامتر دارد، این کار به زمان زیادی نیاز دارد.

High Dimensionality: با افزایش ابعاد داده ها، LSH می تواند کمتر موثر باشد. این به این دلیل است که تعداد جداول هش و توابع هش مورد نیاز برای دستیابی به عملکرد خوب به طور تصاعدی با ابعاد داده ها افزایش می یابد.

Sensitivity to Data Distribution: به توزیع داده ها حساس است. اگر داده ها به طور یکنواخت توزیع نشده باشند، LSH ممکن است نتواند توابع هش خوبی را پیدا کند.

Difficulty with Sparse Data: می تواند با داده های پراکنده، که در آن بیشتر مقادیر صفر هستند، جلوگیری کند. این به این دلیل که توابع هش احتمال کمتری برای ایجاد برخورد برای داده های پراکنده دارند.

در این تکنیک موارد مشابه را با احتمال زیاد به باکت یکسانی نگاشت میکند و امکان جستجوی شباهت کارآمد را فراهم می کند. با این حال، از احتمال نگاشت دو آیتم متفاوت به یک باکت نمی توان به طور کامل اجتناب کرد و در نتیجه منجر به FP میشود. برای کاهش FP، تعداد توابع هش و اندازه جداول هش باید افزایش یابد که پیچیدگی زمانی و مکانی الگوریتم را افزایش می دهد. از طرف دیگر، کاهش تعداد توابع هش و جداول هش، تعداد TP را نیز کاهش می دهد. بنابراین، یافتن

تعادل مناسب بین تعداد توابع هش، جداول هش و تعداد مواردFP ، یک محدودیت کلیدی در LSH است.

سوال دوم

MinHashing به دلایل زیر قابلیت preservingSimilarity را دارد:

1. MinHashing مجموعه های مشابه را با احتمال زیاد به یک باکت نگاشت می کند. در نتیجه، اگر دو مجموعه شباهت jaccard بالایی داشته باشند، احتمالاً سیگنیچر MinHash آنها دارای تعداد زیادی از حداقل مقادیر هش منطبق است.

2. احتمال اینکه حداقل مقدار هش یک عنصر در مجموعه A برابر با حداقل مقدار هش یک عنصر در مجموعه B باشد،برابر است با شباهت jaccard بین مجموعه ها. که به عنوان ویژگی MinHash شناخته می شود.

3. تخمین شباهت jaccard با استفاده از MinHashing با افزایش تعداد توابع هش مورد استفاده برای تولید سیگنیچر، دقیق‌تر می‌شود. زیرا با افزایش تعداد توابع هش، احتمال دو مجموعه دارای سیگنیچر یکسان کاهش می یابد. بنابراین، استفاده از MinHashing برای تولید سیگنیچر برای مجموعه‌ها و مقایسه تعداد حداقل مقادیر هش منطبق، می‌توان شباهت Jaccard بین مجموعه‌ها را تخمین زد و شباهت را حفظ کرد.

سوال سوم

In [5]:
```
documents_shingles = create_shingle(documents, k=K)
print(documents_shingles[2])
```

```
{'rKabi', 'AmirK', 'niver', 'Kabir', 'of Am', 'versi', 'y of ', ' of A', 'irKab', 'sity ', ' Amir', 'ersit', 'iver
s', 'unive', 'mirKa', 'ty of', 'rsity', 'f Ami', 'ity o'}
```

dont change this part just run it

In [6]:
```
res1 = {'of Am', ' of A', 'ty of', 'f Ami', 'irKab', ' Amir', 'unive', 'mirKa', 'y of ', 'ivers', 'rsity', 'sity ',
        'versi', 'ersit', 'ity o', 'rKabi', 'niver', 'Kabir', 'AmirK'}
assert res1 == documents_shingles[0], "Test 1 Failed!"
print("Test 1 Successful!")
```

```
Test 1 Successful!
```

```
t1 = {'a', 'b', 'c'}
t2 = {'a', 'b', 'd'}
assert jaccard(t1, t2) == 0.5, "Task5 faild"
print("task5 successful")
```

```
task5 successful
```

```
print('Similar sentences:')
print(get_topk_similar(target_id, candidates, k=5))
```

```
Similar sentences:
[['The lazy dog has a quick and brown fox that jumps over it.']
 ['The lazy dog has a fox that is quick and brown jumping over it.']
 ['The quick brown fox jumps over the lazy dog.']
 ['The dog that is lazy has a quick and brown fox jumping over it.']
 ['A lazy dog has a quick brown fox jumping over it.']]
```