Homework 3

October 3, 2024

0.0.1 Homework 3 - Emma Brown - 9/27/24

Christian helped me on question 2. He showed me how to apply the lambda function to the code to help pair down the output.

0.0.2 Homework Questions

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

import dhs_util
from dhs_util import *

os.chdir('/Users/emmabrown/Downloads/DS-Unit 1/')
df = pd.read_csv('dhs_service_records_synthesized_final copy.csv')

df = dhs_preprocessing(df)
df, service_map = add_service_label(df)
df = add_age_bin(df)

recipient = get_recipient_attribute(df)
```

[4]:
```
import mlxtend
```

[5]:
```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.preprocessing import *
from mlxtend.frequent_patterns import association_rules
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import fpmax
from mlxtend.frequent_patterns import hmine
```

[6]:
```
serv_list = []
for groups in df.groupby('id').groups.values():
    serv_list.append(df.loc[groups]['serv'].tolist())
```

```python
# following the tutorial example
def oneHotCoding(serv_list):
    te = TransactionEncoder()
    te_ary = te.fit(serv_list).transform(serv_list)
    te_df = pd.DataFrame(te_ary, columns=te.columns_)
    return te_df

serv_oneHot = oneHotCoding(serv_list)
```

[7]:
```python
def get_id_service_matrix(df):
    df_temp = df.groupby(["id","serv"]).agg(
        num_serv = ('service', 'nunique')
    ).reset_index()
    df_serv = df_temp.pivot_table(
        values='num_serv', index=["id"],
        columns="serv", aggfunc=np.sum
    ).reset_index()
    return df_serv
```

[8]:
```python
df_id_serv = get_id_service_matrix(df)
df_id_serv.iloc[:,1:23] = df_id_serv.iloc[:,1:23] > 0
```

Number 1 - Frequent itemsets Extract service itemsets that have support greater than 0.01 using apriori.

Extract service itemsets that have support greater than 0.0001 using fpgrowth.

Extract service itemsets that have support greater than 0.0001 using hmine.

[10]:
```python
apriori(df_id_serv.iloc[:,1:23], use_colnames=True, min_support=0.01)\
    .sort_values(by="support", ascending=False)
```

[10]:

	support	itemsets
4	0.941422	(S12)
2	0.153844	(S09)
14	0.139131	(S12, S09)
15	0.103528	(S14, S09)
6	0.103528	(S14)
25	0.094436	(S12, S14, S09)
20	0.094436	(S12, S14)
3	0.040882	(S11)
18	0.032106	(S12, S11)
10	0.031396	(S19)
7	0.024002	(S15)
21	0.022561	(S12, S15)
16	0.019431	(S15, S09)
13	0.018307	(S09, S11)

```
26  0.018280       (S12, S15, S09)
24  0.016508       (S12, S09, S11)
0   0.013687                 (S03)
9   0.013573                 (S18)
17  0.013468            (S09, S18)
23  0.013460            (S14, S18)
28  0.013460       (S14, S09, S18)
11  0.013260                 (S21)
8   0.012967                 (S17)
1   0.012649                 (S05)
12  0.012284            (S12, S03)
5   0.011915                 (S13)
19  0.011448            (S12, S13)
22  0.010384            (S12, S18)
27  0.010285       (S12, S09, S18)
29  0.010277       (S12, S14, S18)
30  0.010277  (S12, S14, S09, S18)
```

[11]:
```python
fpgrowth(df_id_serv.iloc[:,1:23], use_colnames=True, min_support=0.0001)\
    .sort_values(by="support", ascending=False)
```

[11]:
```
      support                       itemsets
0    0.941422                          (S12)
1    0.153844                          (S09)
22   0.139131                     (S12, S09)
3    0.103528                          (S14)
30   0.103528                     (S14, S09)
..        …                              …
218  0.000101       (S12, S09, S21, S16, S18)
505  0.000101  (S15, S06, S12, S14, S09, S11)
504  0.000101  (S15, S06, S12, S14, S21, S11)
219  0.000101       (S12, S14, S21, S16, S18)
502  0.000101       (S15, S06, S12, S14, S11)

[935 rows x 2 columns]
```

[12]:
```python
hmine(df_id_serv.iloc[:,1:23], use_colnames=True, min_support=0.0001)\
    .sort_values(by="support", ascending=False)
```

[12]:
```
      support             itemsets
784  0.941422                (S12)
458  0.153844                (S09)
548  0.139131           (S12, S09)
633  0.103528           (S14, S09)
872  0.103528                (S14)
..        …                    …
479  0.000101      (S09, S17, S10)
```

```
379  0.000101                  (S07, S10)
380  0.000101             (S07, S11, S10)
715  0.000101                  (S19, S10)
815  0.000101  (S12, S14, S21, S16, S18)

[935 rows x 2 columns]
```

Make observations of the difference between them.

The three of these all give you the same support numbers and itemsets in order, but their indices are different. Apriori also only gives you 31 values whereas fpgrowth and hmine give you 935.

[14]:
```
fpmax(df_id_serv.iloc[:,1:23], use_colnames=True, min_support=0.0001)\
    .sort_values(by="support", ascending=False)
```

[14]:
```
     support                             itemsets
56  0.000326       (S15, S12, S14, S09, S17, S18)
1   0.000305                           (S12, S01)
34  0.000253  (S07, S12, S14, S09, S16, S11, S18)
63  0.000247            (S12, S14, S09, S19, S11)
11  0.000223       (S02, S12, S14, S09, S03, S18)
..       …                                    …
16  0.000101                      (S07, S11, S10)
15  0.000101                           (S19, S10)
29  0.000101  (S15, S06, S12, S14, S09, S21, S11)
17  0.000101                      (S09, S17, S10)
40  0.000101       (S12, S14, S09, S21, S16, S18)

[64 rows x 2 columns]
```

Make observations between fpmax and others.

fpmax gives you completely different support and itemsets. Their supports start a lot lower than apriori, fpgrowth and hmine. fpmax contains more rows than apriori, but significantly less than fpgrowth and hmine.

Number 2 - Frequent itemsets with length (size of the itemsets) Extract length-1 service itemsets that have support greater than 0.01 using apriori.

Extract length-2 service itemsets that have support greater than 0.005 using hmine.

Extract length-5 service itemsets that have support greater than 0.0001 using fpmax.

I used the 'iloc' command to "grab" the first 12 rows of data because the function listed the results in increasing lengths of itemsets. I printed out all of ext_apr_1 and used this observation to grab the rows I needed.

[18]:
```
ext_apr_1 = apriori(serv_oneHot,min_support=0.01,use_colnames=True)
ext_apr_1.iloc[0:12, :]
```

[18]:

	support	itemsets
0	0.013687	(S03)
1	0.012649	(S05)
2	0.153844	(S09)
3	0.040882	(S11)
4	0.941422	(S12)
5	0.011915	(S13)
6	0.103528	(S14)
7	0.024002	(S15)
8	0.012967	(S17)
9	0.013573	(S18)
10	0.031396	(S19)
11	0.013260	(S21)

[19]:

```python
ext_hmine = hmine(serv_oneHot, min_support=0.005,use_colnames=True)
ext_hmine_2 = ext_hmine[ext_hmine['itemsets'].apply(lambda x: len(x)==2)]
ext_hmine_2
```

[19]:

	support	itemsets
1	0.005274	(S03, S09)
3	0.012284	(S12, S03)
5	0.008057	(S12, S05)
8	0.006398	(S07, S11)
10	0.005105	(S12, S07)
12	0.018307	(S09, S11)
16	0.139131	(S12, S09)
27	0.008419	(S09, S13)
28	0.103528	(S14, S09)
32	0.019431	(S15, S09)
33	0.009372	(S16, S09)
34	0.006731	(S09, S17)
35	0.013468	(S09, S18)
36	0.005268	(S19, S09)
37	0.007949	(S21, S09)
39	0.005238	(S12, S10)
41	0.032106	(S12, S11)
43	0.009374	(S14, S11)
45	0.011448	(S12, S13)
46	0.094436	(S12, S14)
50	0.022561	(S12, S15)
51	0.00683	(S12, S16)
52	0.008674	(S12, S17)
53	0.010384	(S12, S18)
54	0.009949	(S12, S19)
55	0.008848	(S12, S21)
58	0.009888	(S14, S15)
59	0.009371	(S14, S16)

```
60   0.01346  (S14, S18)
```

[20]:
```
ext_fpmax = fpmax(serv_oneHot, min_support=0.0001,use_colnames=True)
ext_fpmax_5 = ext_fpmax[ext_fpmax['itemsets'].apply(lambda x: len(x)==5)]
ext_fpmax_5
```

[20]:
```
     support                   itemsets
4   0.000131  (S12, S14, S09, S04, S03)
18  0.000163  (S12, S14, S09, S10, S18)
19  0.000157  (S06, S12, S09, S21, S10)
21  0.000124  (S15, S12, S09, S21, S10)
22  0.000208  (S15, S12, S14, S09, S10)
23  0.000139  (S12, S14, S09, S21, S10)
24  0.000197  (S12, S14, S09, S10, S11)
25  0.000103  (S06, S14, S09, S21, S16)
31  0.000103  (S07, S12, S09, S19, S11)
33  0.000120  (S07, S12, S09, S03, S11)
37  0.000131  (S14, S09, S19, S16, S18)
38  0.000142  (S12, S14, S09, S19, S16)
45  0.000120  (S12, S14, S09, S03, S13)
46  0.000111  (S12, S14, S09, S17, S13)
47  0.000150  (S15, S12, S14, S09, S13)
49  0.000150  (S12, S14, S09, S13, S11)
50  0.000111  (S12, S14, S09, S05, S18)
53  0.000109  (S12, S09, S21, S17, S11)
61  0.000204  (S12, S14, S09, S19, S18)
62  0.000169  (S15, S12, S14, S09, S19)
63  0.000247  (S12, S14, S09, S19, S11)
```

Number 3 - Association Rules Extract association rules that have lift greater than 0.5, using itemsets from fpgrowth with support greater than 0.05.

Extract association rules that have support greater than 0.0003, using itemsets from fpmax with support greater than 0.0001.

[22]:
```
def serv_rules(freq_itemsets,metrics,threshold):
    asso_rules = association_rules(freq_itemsets, metric=metrics,␣
 ↪min_threshold=threshold)
    return asso_rules.sort_values(by='lift', ascending=False)[['antecedents',␣
 ↪'consequents', 'support', 'confidence', 'lift']]
```

[23]:
```
min_freq = 1000 # if we want to set threshold by frequency of the itemsets
min_support = min_freq/serv_oneHot.shape[0]
min_confidence = 0.6
min_rule_support = 0.2
min_lift = 0.15
```

[24]:
```
freq_itemset_fpgrowth =␣
↪fpgrowth(serv_oneHot,min_support=min_support,use_colnames=True)
```

[25]:
```
rule_fpgrowth = serv_rules(freq_itemset_fpgrowth,"support",0.05)
```

[26]:
```
rule_fpgrowth[rule_fpgrowth['lift'].apply(lambda x: x>0.5)]
```

[26]:

	antecedents	consequents	support	confidence	lift
7	(S12, S09)	(S14)	0.094436	0.678758	6.556292
10	(S14)	(S12, S09)	0.094436	0.912184	6.556292
2	(S14)	(S09)	0.103528	1.000000	6.500073
3	(S09)	(S14)	0.103528	0.672938	6.500073
6	(S12, S14)	(S09)	0.094436	1.000000	6.500073
11	(S09)	(S12, S14)	0.094436	0.613843	6.500073
4	(S12)	(S14)	0.094436	0.100312	0.968942
5	(S14)	(S12)	0.094436	0.912184	0.968942
8	(S14, S09)	(S12)	0.094436	0.912184	0.968942
9	(S12)	(S14, S09)	0.094436	0.100312	0.968942
0	(S12)	(S09)	0.139131	0.147788	0.960634
1	(S09)	(S12)	0.139131	0.904362	0.960634

[27]:
```
freq_itemset_fpmax = fpmax(serv_oneHot,min_support=0.0001, use_colnames=True)
```

[28]:
```
asso_rules = association_rules(freq_itemset_fpmax, metric='lift',␣
↪min_threshold=0.0003, support_only=True)
asso_rules.sort_values(by='lift', ascending=False)[['antecedents',␣
↪'consequents', 'support', 'confidence', 'lift']]
```

[28]:

	antecedents	consequents	support \
0	(S12)	(S01)	0.000305
1	(S01)	(S12)	0.000305
2	(S15, S12, S14, S09, S17)	(S18)	0.000326
3	(S15, S12, S14, S09, S18)	(S17)	0.000326
4	(S15, S12, S14, S17, S18)	(S09)	0.000326
..	…	…	…
59	(S12)	(S15, S14, S09, S17, S18)	0.000326
60	(S14)	(S15, S12, S09, S17, S18)	0.000326
61	(S09)	(S15, S12, S14, S17, S18)	0.000326
62	(S17)	(S15, S12, S14, S09, S18)	0.000326
63	(S18)	(S15, S12, S14, S09, S17)	0.000326

	confidence	lift
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

```
..              …    …
59            NaN   NaN
60            NaN   NaN
61            NaN   NaN
62            NaN   NaN
63            NaN   NaN

[64 rows x 5 columns]
```

Number 4 - Make Predictions (draw conclusion) If you were to make suggestions for DHS to consider increase the offering of one service, say, "Families_Receiving_Child_Welfare_Services" (S06), what other services would you suggest to offer together with?

Run predict({"S06"}, rules, consequents_only=False), with rules generated from different thresholds and algorithms.

[30]:
```
asso_rules = association_rules(freq_itemset_fpmax, metric='lift',␣
 ↪min_threshold=0.000001, support_only=True)
asso_rules.sort_values(by='lift', ascending=False)[['antecedents',␣
 ↪'consequents', 'support', 'confidence', 'lift']]
```

[30]:
```
     antecedents           consequents   support  confidence  lift
0          (S12)                 (S08)  0.000199         NaN   NaN
1          (S08)                 (S12)  0.000199         NaN   NaN
2          (S12)                 (S01)  0.000305         NaN   NaN
3          (S01)                 (S12)  0.000305         NaN   NaN
4          (S12)                 (S22)  0.000124         NaN   NaN
…            …                   …        …           …    …
2455       (S12)  (S14, S19, S09, S11)  0.000247         NaN   NaN
2456       (S14)  (S12, S19, S09, S11)  0.000247         NaN   NaN
2457       (S09)  (S12, S14, S19, S11)  0.000247         NaN   NaN
2458       (S19)  (S12, S14, S09, S11)  0.000247         NaN   NaN
2459       (S11)  (S12, S14, S19, S09)  0.000247         NaN   NaN

[2460 rows x 5 columns]
```

[31]:
```
def predict(antecedent, rules, consequents_only = False):
    # get the rules for this antecedent
    preds = rules[rules['antecedents'] == antecedent]
    if consequents_only:
        # a way to convert a frozen set with one element to string
        preds = preds['consequents'].apply(iter).apply(next)
    return preds
```

[32]:
```
predict({'S06'}, asso_rules, consequents_only=False)
```

[32]:

```
     antecedents                           consequents  antecedent support  \
399        (S06)              (S12, S21, S09, S10)                 NaN
563        (S06)              (S14, S21, S16, S09)                 NaN
624        (S06)         (S12, S14, S09, S21, S18)                 NaN
686        (S06)         (S12, S14, S09, S21, S17)                 NaN
749        (S06)         (S15, S12, S09, S21, S17)                 NaN
874        (S06)    (S15, S12, S14, S09, S21, S11)                 NaN

     consequent support   support  confidence  lift  leverage  conviction  \
399                 NaN  0.000157         NaN   NaN       NaN         NaN
563                 NaN  0.000103         NaN   NaN       NaN         NaN
624                 NaN  0.000146         NaN   NaN       NaN         NaN
686                 NaN  0.000105         NaN   NaN       NaN         NaN
749                 NaN  0.000114         NaN   NaN       NaN         NaN
874                 NaN  0.000101         NaN   NaN       NaN         NaN

     zhangs_metric
399            NaN
563            NaN
624            NaN
686            NaN
749            NaN
874            NaN
```

Based on the above table, if DHS was considering increasing the offering of "Families_Receiving_Child_Welfare_Services" I would suggest they also offer support in the areas of "Income_Support" (S12) and "Individuals_Receiving_Mental_Health_Services"(S14) (also S09 and S21, but these are already accounted for in S06).

[34]:

```
asso_rules = association_rules(freq_itemset_fpmax, metric='lift',␣
↪min_threshold=0.0002, support_only=True)
asso_rules.sort_values(by='lift', ascending=False)[['antecedents',␣
↪'consequents', 'support', 'confidence', 'lift']]
```

[34]:

```
                     antecedents           consequents   support  confidence  \
0                          (S12)                 (S01)  0.000305         NaN
1                          (S01)                 (S12)  0.000305         NaN
2    (S02, S12, S14, S09, S03)                 (S18)  0.000223         NaN
3    (S02, S12, S14, S09, S18)                 (S03)  0.000223         NaN
4    (S02, S12, S14, S03, S18)                 (S09)  0.000223         NaN
..                         …                     …         …           …
587                        (S12)  (S14, S19, S09, S11)  0.000247         NaN
588                        (S14)  (S12, S19, S09, S11)  0.000247         NaN
589                        (S09)  (S12, S14, S19, S11)  0.000247         NaN
590                        (S19)  (S12, S14, S09, S11)  0.000247         NaN
591                        (S11)  (S12, S14, S19, S09)  0.000247         NaN

     lift
```

```
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
..     …
587    NaN
588    NaN
589    NaN
590    NaN
591    NaN

[592 rows x 5 columns]
```

[35]:

```python
predict({'S14'}, asso_rules, consequents_only=False)
```

[35]:

```
    antecedents                    consequents  antecedent support  \
60        (S14)       (S02, S12, S09, S03, S18)                 NaN
91        (S14)            (S12, S15, S09, S10)                 NaN
215       (S14)  (S07, S12, S09, S16, S11, S18)                 NaN
342       (S14)  (S15, S07, S12, S09, S11, S18)                 NaN
404       (S14)       (S15, S12, S09, S21, S17)                 NaN
466       (S14)       (S15, S12, S09, S17, S18)                 NaN
528       (S14)       (S15, S12, S09, S21, S18)                 NaN
558       (S14)            (S12, S19, S09, S18)                 NaN
588       (S14)            (S12, S19, S09, S11)                 NaN

     consequent support   support  confidence  lift  leverage  conviction  \
60                  NaN  0.000223         NaN   NaN       NaN         NaN
91                  NaN  0.000208         NaN   NaN       NaN         NaN
215                 NaN  0.000253         NaN   NaN       NaN         NaN
342                 NaN  0.000223         NaN   NaN       NaN         NaN
404                 NaN  0.000210         NaN   NaN       NaN         NaN
466                 NaN  0.000326         NaN   NaN       NaN         NaN
528                 NaN  0.000210         NaN   NaN       NaN         NaN
558                 NaN  0.000204         NaN   NaN       NaN         NaN
588                 NaN  0.000247         NaN   NaN       NaN         NaN

     zhangs_metric
60             NaN
91             NaN
215            NaN
342            NaN
404            NaN
466            NaN
528            NaN
558            NaN
```

```
588            NaN
```

Based on the above table, if DHS was considering increasing the offering of "Individuals_Receiving_Mental_Health_Services" I would suggest they also offer support in the areas of "Income_Support" (S12), "Mental_Health_Crisis" (S18), and "Individuals_Receiving_Substance_Use_Disorder_Services" (S15).