2 - PreProcess & Label

September 14, 2022

1 DI Chronic Rule Search Pre-Processing Routines

This code pre-processes DI client data for use when identifying potentially chronic shelter users during monthly housing case worker meetings.

[1]:
```
%load_ext autoreload
%autoreload 1
```

[2]:
```
import numpy as np
import pandas as pd
import datetime, copy, imp
import time
import os
import re

from tqdm.auto import tqdm, trange
from tqdm.notebook import tqdm
tqdm.pandas()

from pandas.tseries.offsets import DateOffset

import sys
sys.path.insert(0, '../util/')
```

```
%aimport di_data
%aimport data_cache

from di_data import *
from data_cache import CacheResult
```

1.0.1 Data Load and Merge

- Load the raw DI data.
- Merge keyword categories that are obviously connected.

[6]:
```
#dataDirStr = '/hd2/data/di/plwh/'
#cacheDirStr = '/hd2/data/di/plwh/cache/'

dataDirStr = '~/data/plwh/'
cacheDirStr = '~/plwh/cache/'
```

[7]:
```
# Pre-merge some keyword categories with obvious similarities.
ftrMerges = {
    'Addiction': [ 'Addiction' ],
    'Bar': [ 'Bar' ],
    'PhysicalHealth': [ 'PhysicalHealth', 'Medication', 'Health', 'Death' ],
    'MentalHealth': [ 'MentalHealth' ],
    'Violence': [ 'Brawl', 'Gun', 'Weapon', 'Spray', 'PhysicalViolence',␣
 ↪'SexualViolence', 'Knife' ],
    'PoliceJustice': [ 'Justice', 'CPS', 'PoliceLogFlag' ],
    'Conflict': [ 'Conflict' ],
    'EMS': [ 'EMS', 'EmsLogFlag' ],
    'Supports': [ 'Supports', 'Housing', 'Financial', 'Employment', 'Education',
                  'FriendsFamily', 'Property', 'EmployeeIsCounsellor' ],
    'Overdose': [ 'Overdose' ]
}

ftrKeep = [ 'ClientId', 'Date', 'Age' ]
```

[8]:
```
tblAll = pd.read_hdf(dataDirStr + 'UniversityExportAnonymized.hd5')
```

[9]:
```
tblAll.columns
```

[9]:
```
Index(['ClientId', 'Date', 'EmployeeId', 'EmployeeIsCounsellor', 'EmsLogFlag',
       'PoliceLogFlag', 'BarDuration', 'Location', 'EntryType', 'ClientState',
       'Age', 'Addiction', 'Bar', 'Biometrics', 'Brawl', 'CPS', 'Conflict',
       'Death', 'EMS', 'Education', 'Employment', 'Financial', 'FriendsFamily',
       'Gun', 'Health', 'Housing', 'ID', 'Indigenous', 'Justice', 'Knife',
       'Medication', 'MentalHealth', 'NegativeWord', 'Overdose',
       'PhysicalHealth', 'PhysicalViolence', 'PositiveWord', 'Property',
       'Seniors', 'SexualViolence', 'Spray', 'Supports', 'Weapon'],
```

```
      dtype='object')
```

[5]:
```
@CacheResult
def load_raw_data():
    tblAll = pd.read_hdf(dataDirStr + 'UniversityExportAnonymized.hd5')
    tblAll = tblAll[tblAll.Date >= pd.to_datetime('2008-07-01')]

    print('Total Entries: {}'.format(len(tblAll.index)))
    print('Dates: ',min(tblAll.Date),' to ',max(tblAll.Date))

    tbl = copy.deepcopy(tblAll[ftrKeep])

    # 1 hot encode entry type
    tbl['EntrySleep'] = (tblAll.EntryType == 'Sleep').astype(int)
    tbl['EntryConsl'] = ((tblAll.EntryType == 'CounsellorsNotes')
                         | (tblAll.EntryType == 'ProgressDetails')).astype(int)
    tbl['EntryLog'] = (tblAll.EntryType == 'Log').astype(int)
    tbl['EntryBar'] = (tblAll.EntryType == 'Bar').astype(int)

    # Merge pre-identified features.
    for ftr in ftrMerges.keys():
        tbl[ftr] = tblAll[ftrMerges[ftr]].sum(axis=1).clip(upper=1).astype(int)

    return tbl
```

[6]:
```
tbl = load_raw_data(path=cacheDirStr,filename='DiRules-LoadRawData.hd5')
```

[7]:
```
tbl
```

[7]:

	ClientId	Date	Age	EntrySleep	EntryConsl	\
4497	35951	2008-11-09 00:00:00.000	39.0	0	0	
4801	34334	2008-11-09 00:00:00.000	42.0	0	0	
6121	26787	2008-11-09 00:00:00.000	64.0	0	0	
14019	12902	2009-10-06 07:28:54.640	39.0	0	0	
16463	31200	2012-11-20 00:00:00.000	51.0	0	0	
…	…	…	…	…	…	
5576428	2528038	2019-11-17 00:00:00.000	73.0	1	0	
5576429	2532949	2019-11-17 00:00:00.000	53.0	1	0	
5576430	2495818	2019-11-17 00:00:00.000	54.0	1	0	
5576431	27906	2019-11-17 00:00:00.000	58.0	1	0	
5576432	18559	2019-11-17 00:00:00.000	78.0	1	0	

	EntryLog	EntryBar	Addiction	Bar	PhysicalHealth	MentalHealth	\
4497	0	1	0	0	0	0	
4801	0	1	0	0	0	0	
6121	0	1	0	0	0	0	
14019	0	1	0	1	0	0	

```
16463           0         1           0    0                0             0
…             …       …         …   …                  …             …
5576428         0         0           0    0                0             0
5576429         0         0           0    0                0             0
5576430         0         0           0    0                0             0
5576431         0         0           0    0                0             0
5576432         0         0           0    0                0             0

         Violence  PoliceJustice  Conflict  EMS  Supports  Overdose
4497            0              0         1    0         0         0
4801            0              0         0    0         0         0
6121            0              0         1    0         0         0
14019           0              0         1    0         1         0
16463           0              0         0    0         0         0
…             …             …       …   …       …       …
5576428         0              0         0    0         0         0
5576429         0              0         0    0         0         0
5576430         0              0         0    0         0         0
5576431         0              0         0    0         0         0
5576432         0              0         0    0         0         0

[5060302 rows x 17 columns]
```

[ ]:

[ ]:

1.0.2 Label Data

- Use the Canadian federal definition of chronic.

[8]:
```
@CacheResult
def generate_stay_timelines():
    return tbl.loc[tbl.EntrySleep==1].groupby('ClientId').
↪progress_apply(calculate_stay_sequence)
```

[9]:
```
tlSty =␣
↪generate_stay_timelines(path=cacheDirStr,filename='DiRules-GenerateStayTimelines.
↪hd5')
```

[10]:
```
tlSty
```

[10]:
```
                      Date  Ind
ClientId
9544     1248572 2008-07-01    1
         1249020 2008-07-02    2
         1249571 2008-07-03    3
         1250067 2008-07-04    4
```

```
          1250644 2008-07-05    5
…                         …   …
2534333   4298131 2019-11-30    1
2534334   4297950 2019-11-29    1
2534335   4297784 2019-11-30    1
2534339   4300886 2019-11-30    1
2534340   4300960 2019-11-30    1

[3580906 rows x 2 columns]
```

[11]:
```python
# Applies a time windowed threshold test to a count of stays.
def cdn_fed_chronic(tbl):

    # First Test: 180 days in past 1 year
    winSz = 365
    thresh = 180

    win = tbl.rolling('%dd' % winSz,on='Date').count().Ind
    registrationDate = tbl.Date.min()
    idDate1 = tbl[win >= thresh].Date.min()  # Will be equal to NaN if the␣
↪threshold isn't met.

    # Second Test: 546 days in past 3 years
    winSz = 365*3
    thresh = 546

    win = tbl.rolling('%dd' % winSz,on='Date').count().Ind
    registrationDate = tbl.Date.min()
    idDate2 = tbl[win >= thresh].Date.min()  # Will be equal to NaN if the␣
↪threshold isn't met.


    idDate = min([ idDate1, idDate2 ])

    if idDate == idDate:   # Satisfied if idDate is not NaN.
        return pd.Series({
            'Flag': 'chr',  # Flag indicating test was satisfied.
            'Date': idDate,   # Date client was identified.
            'Time': (idDate - registrationDate).days + 1 # Number of days it␣
↪took to identify client.
        })
    else:
        return pd.Series({   # Returned if the test is not satisfied.
            'Flag': 'tmp',
            'Date': tbl.Date.max(),
            'Time': (tbl.Date.max()-tbl.Date.min()).days + 1
        })
```

[12]:
```
@CacheResult
def label_cdn_fed_chronic():
    return tlSty.groupby('ClientId').progress_apply(cdn_fed_chronic)
```

[13]:
```
tteChr =␣
↪label_cdn_fed_chronic(path=cacheDirStr,filename='DiRules-CdnFedChronicTte.
↪hd5')
```

[14]:
```
nChron = sum(tteChr.Flag == 'chr')
nClients = len(tteChr.Flag)
print('Chronic clients: {}/{} ({:.1f}%)'.format(nChron,nClients,100.0*nChron/
↪nClients))
```

```
Chronic clients: 3191/32346 (9.9%)
```

[15]:
```
# Only consider clients that have a label.
tbl = tbl.loc[tbl.ClientId.isin(tteChr.index)]
```

[ ]:
```

```

1.1 Generate Shelter Access Statistics

[16]:
```
@CacheResult
def calculate_access_statistics():
    return tbl.groupby('ClientId').
↪progress_apply(shelter_client_access_statistics)
```

[17]:
```
accs =␣
↪calculate_access_statistics(path=cacheDirStr,filename='DiRules-AccessStatistics.
↪hd5')
```

[18]:
```
accs
```

[18]:

ClientId	Tenure	UsagePct	AvgGapLen	TotalStays	TotalEpisodes
9544	1544.0	95.531088	1.028494	1475.0	1.0
9548	2688.0	4.910714	20.404580	132.0	6.0
9558	3959.0	104.420308	0.876361	4134.0	2.0
9561	3123.0	0.576369	183.588235	18.0	13.0
9566	1.0	100.000000	NaN	1.0	1.0
…	…	…	…	…	…
2534333	1.0	100.000000	NaN	1.0	1.0
2534334	1.0	100.000000	NaN	1.0	1.0
2534335	1.0	200.000000	0.000000	2.0	1.0
2534339	1.0	100.000000	NaN	1.0	1.0
2534340	1.0	100.000000	NaN	1.0	1.0

```
[32346 rows x 5 columns]
```

[19]:
```
def summarize_shelter_access(acs):

    longFields = { 'TotalStays': 'Total Stays', 'TotalEpisodes': 'Total␣
↪Episodes', 'Tenure': 'Tenure (days)',
                   'UsagePct': 'Usage Percentage', 'AvgGapLen': 'Average Gap␣
↪Length (days)' }

    fields = [ 'TotalStays', 'TotalEpisodes', 'Tenure', 'UsagePct', 'AvgGapLen'␣
↪]
    for field in fields:
        print('%s:' % (field))
        nEntry = sum(~np.isnan(acs[field]))
        print(' Avg: {:.1f}, Med: {:.1f}, 10thPct: {:.1f}, 90thPct: {:.1f}'
              .format(acs[field].mean(),acs[field].median(),
                    acs[field].sort_values().iloc[int(nEntry*0.1)],
                    acs[field].sort_values().iloc[int(nEntry*0.9)]))


    print()
    for field in fields:
        nEntry = sum(~np.isnan(acs[field]))
        print('{} & {:.1f} & {:.1f} & {:.1f} & {:.1f}\\\\'
              .format(longFields[field],acs[field].mean(),acs[field].median(),
                    acs[field].sort_values().iloc[int(nEntry*0.1)],
                    acs[field].sort_values().iloc[int(nEntry*0.9)]))
```

[20]:
```
chrClientIds = list( tteChr.loc[tteChr.Flag == 'chr'].index )
nonChrClientIds = list( tteChr.loc[~(tteChr.Flag == 'chr')].index )
```

[21]:
```
summarize_shelter_access(accs.loc[chrClientIds])
```

```
TotalStays:
 Avg: 963.6, Med: 671.0, 10thPct: 277.0, 90thPct: 2140.0
TotalEpisodes:
 Avg: 6.9, Med: 5.0, 10thPct: 1.0, 90thPct: 15.0
Tenure:
 Avg: 2254.4, Med: 2237.0, 10thPct: 612.0, 90thPct: 3960.0
UsagePct:
 Avg: 48.1, Med: 40.9, 10thPct: 15.3, 90thPct: 93.4
AvgGapLen:
 Avg: 3.2, Med: 2.4, 10thPct: 1.0, 90thPct: 6.5

Total Stays & 963.6 & 671.0 & 277.0 & 2140.0\\
Total Episodes & 6.9 & 5.0 & 1.0 & 15.0\\
Tenure (days) & 2254.4 & 2237.0 & 612.0 & 3960.0\\
```

```
Usage Percentage & 48.1 & 40.9 & 15.3 & 93.4\\
Average Gap Length (days) & 3.2 & 2.4 & 1.0 & 6.5\\
```

[22]:
```
summarize_shelter_access(accs.loc[nonChrClientIds])
```

```
TotalStays:
 Avg: 34.6, Med: 5.0, 10thPct: 1.0, 90thPct: 108.0
TotalEpisodes:
 Avg: 3.3, Med: 2.0, 10thPct: 1.0, 90thPct: 8.0
Tenure:
 Avg: 766.1, Med: 162.0, 10thPct: 1.0, 90thPct: 2590.0
UsagePct:
 Avg: 46.6, Med: 13.7, 10thPct: 0.4, 90thPct: 100.0
AvgGapLen:
 Avg: 164.9, Med: 20.0, 10thPct: 1.0, 90thPct: 430.3

Total Stays & 34.6 & 5.0 & 1.0 & 108.0\\
Total Episodes & 3.3 & 2.0 & 1.0 & 8.0\\
Tenure (days) & 766.1 & 162.0 & 1.0 & 2590.0\\
Usage Percentage & 46.6 & 13.7 & 0.4 & 100.0\\
Average Gap Length (days) & 164.9 & 20.0 & 1.0 & 430.3\\
```

[ ]:
```

```

1.2 Generate Start of Stay Windowed Data

[78]:
```
winColumns = [ 'EntrySleep', 'EntryConsl', 'EntryLog', 'EntryBar',
               'Addiction', 'Bar', 'PhysicalHealth', 'MentalHealth', 'Violence',
               'PoliceJustice', 'Conflict', 'EMS', 'Supports',
               'Overdose' ]
fixedColumns = [ 'Age' ]
absentThrsh = 30 # days absent to be no longer active


def review_client_window(tbl,winSzDays,tte):

    clientId = tbl.iloc[0].ClientId

    startDate = tbl.Date.min()

    # The client is evaluated on the first meeting date where they have been in␣
↪shelter at
    # least winSzDays.
    evalDate = pd.to_datetime('{}{:02d}01'.format(startDate.year,startDate.
↪month)) + pd.DateOffset(months=1)
    while (evalDate-startDate).days < winSzDays:
        evalDate += pd.DateOffset(months=1)
```

```
    # Resample so that muliple data entries in each category that occur on a␣
↪particular day are counted only once.
    smry = tbl.loc[tbl.Date <= evalDate].resample('D',on='Date').max()

    # Add up all the events that have occurred in our window.
    smry = smry[ winColumns ].sum()

    # Number of days before the client is evaluated.
    smry['EvalTime'] = (evalDate-startDate).days

    # Determine if the client is ever classified as chronic.
    smry['Label'] = ( tte.loc[clientId].Flag == 'chr' )

    # Determine whether the client is active (has been seen <= absentThrsh days␣
↪ago).
    smry['Active'] = ( evalDate - tbl.loc[tbl.Date <= evalDate].Date.max() ).
↪days <= absentThrsh

    # Add any static data that should not be summed up over the window (ie.␣
↪age).
    for fixedCol in fixedColumns:
        smry[fixedCol] = tbl[fixedCol].iloc[0]

    return smry
```

[84]:
```
#review_client_window(tbl.loc[tbl.ClientId == 9544,:],30,tteChr)
#review_client_window(tbl.loc[tbl.ClientId == 20132,:],120,tteChr)
#SummarizeClient(tbl.loc[tbl.ClientId == 20132,:],30,tteChr)
#SummarizeClient(tbl.loc[tbl.ClientId == 9548,:],60,tteChr)
```

[29]:
```
@CacheResult
def review_clients_window(tbl,winSzDays,tte):
    return tbl.groupby('ClientId').
↪progress_apply(review_client_window,winSzDays=winSzDays,tte=tte)
```

[ ]:
```
```

[30]:
```
smry30 =␣
↪review_clients_window(tbl,30,tteChr,path=cacheDirStr,filename='DiRules-ClientSummaryWindow.
↪hd5')
smry60 =␣
↪review_clients_window(tbl,60,tteChr,path=cacheDirStr,filename='DiRules-ClientSummaryWindow.
↪hd5')
smry90 =␣
↪review_clients_window(tbl,90,tteChr,path=cacheDirStr,filename='DiRules-ClientSummaryWindow.
↪hd5')
```

```
smry120 =␣
↪review_clients_window(tbl,120,tteChr,path=cacheDirStr,filename='DiRules-ClientSummaryWindow
↪hd5')
```

```
  0%|          | 0/32346 [00:00<?, ?it/s]

  0%|          | 0/32346 [00:00<?, ?it/s]

  0%|          | 0/32346 [00:00<?, ?it/s]

  0%|          | 0/32346 [00:00<?, ?it/s]
```

[ ]:

[31]:
```
hdfName = 'DiRules-EvaluationData.hd5'
smry30.to_hdf(cacheDirStr+'/'+hdfName, key = 'Win30')
smry60.to_hdf(cacheDirStr+'/'+hdfName, key = 'Win60')
smry90.to_hdf(cacheDirStr+'/'+hdfName, key = 'Win90')
smry120.to_hdf(cacheDirStr+'/'+hdfName, key = 'Win120')
```

[ ]:
```
smry30
```

[ ]: