Dimension Reduction and Unsupervised Learning

Data Science and Machine Learning Team

??/??/2021

Andrew Wheeler, PhD

andrew.wheeler@hms.com

Agenda

- Dimension Reduction
  - Motivations
  - Example using principle components analysis (social determinants of health)

- Unsupervised Learning
  - Distances and overview of types of clustering
  - Example of hierarchical clustering (OPTICS) using claims data

Dimension Reduction

- What is it? Reduce multiple *columns* in data to a smaller number
    - E.g. input 100 columns, reduce to 5 columns

- Why?
    - Ease of interpretation – instead of a dozen fields, can reduce to one field (e.g. social determinants of health, DRG)
    - Model building – some models it is easier to build using a smaller number of dimensions than many (e.g. reduce high dimensional text using Word2Vec)
    - Clustering – difficult to cluster with a high number of dimensions (curse of dimensionality)

Example Dimension Reduction

- Social Determinants of Health using Principle Components Analysis (PCA)

- Data Source
  - American Community Survey data (demographics from Census) 5 year estimates for 2019
  - All Census Tracts in Texas (n = 5,265)
  - Fields: Poverty, Single Parent Headed Household with Children, Limited-English, No Car. All variables as % per relevant denominator (households, pop over 5, workers)

Example Dimension Reduction

In [1]:

```
# Libraries we need
import pandas as pd
from sklearn import decomposition
from sklearn.preprocessing import scale
import os

# Changing the directory to where I have the data stored
os.chdir(r'C:\Users\e009156\Documents\GitHub\data-science-utils\education\Intro_

# Reading in the census data, social determinants of health -> sdet
sdet = pd.read_csv('SocialDet_TexCT.csv', index_col='LOGRECNO')
sdet.head(10) #these are all as proportions
```

Out[1]:

LOGRECNO	PovertyFamily	SingleHeadwithKids	LimitedEnglishPop	NoCarWorkers
4295	0.134318	0.029823	0.013348	0.015126
4296	0.000000	0.021807	0.007292	0.000000
4297	0.329545	0.141463	0.008532	0.000000
4298	0.144231	0.052799	0.095979	0.031536
4299	0.113090	0.034755	0.008384	0.160606
4300	0.181655	0.044419	0.136452	0.013725
4301	0.172199	0.063035	0.034688	0.019985
4302	0.055024	0.023200	0.016935	0.018149
4303	0.069044	0.006887	0.024245	0.046362
4304	0.095735	0.040089	0.002184	0.001268

Example Dimension Reduction

In [2]:
```
# Lets look at the correlations between each of these variables
sdet.corr()
```

Out[2]:

	PovertyFamily	SingleHeadwithKids	LimitedEnglishPop	NoCarWorkers
PovertyFamily	1.000000	0.547829	0.587112	0.449034
SingleHeadwithKids	0.547829	1.000000	0.256104	0.304791
LimitedEnglishPop	0.587112	0.256104	1.000000	0.259051
NoCarWorkers	0.449034	0.304791	0.259051	1.000000

Example Dimension Reduction

In [3]:

```
# Before you do PCA, you should scale the variables (this creates z-scores)
scale_det = pd.DataFrame(scale(sdet), columns=list(sdet))
print( scale_det.describe() )

# But does not change the correlations
scale_det.corr()
```

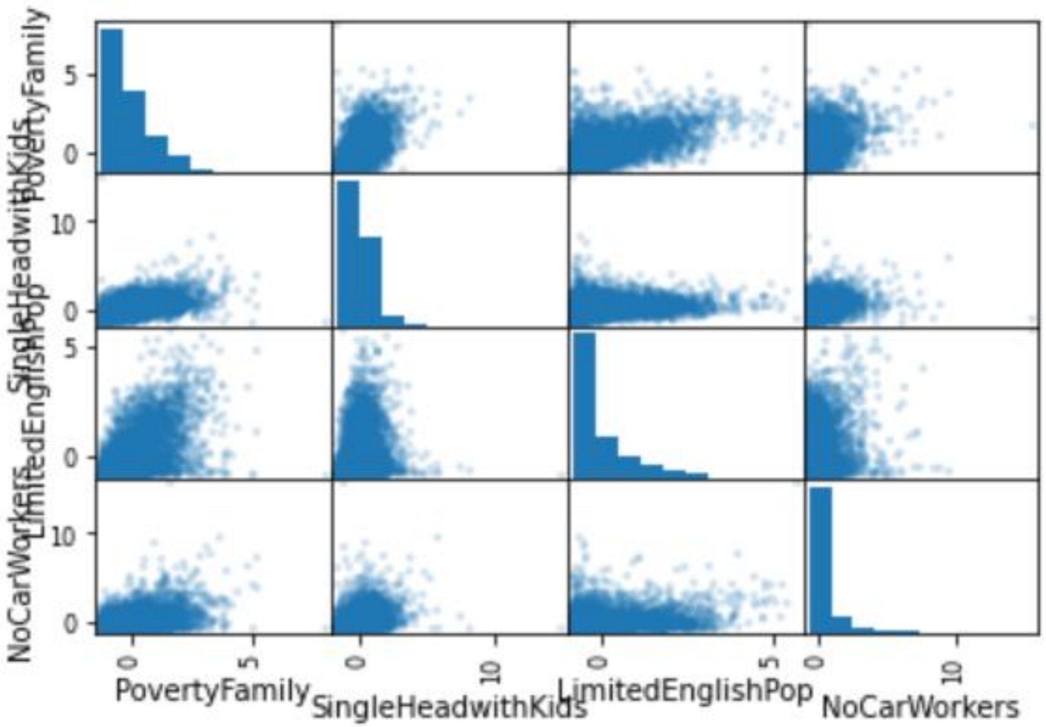
```
       PovertyFamily  SingleHeadwithKids  LimitedEnglishPop  NoCarWorkers
count   5.265000e+03        5.265000e+03       5.265000e+03  5.265000e+03
mean    1.158090e-16        1.308650e-16       3.182402e-16 -6.470818e-16
std     1.000095e+00        1.000095e+00       1.000095e+00  1.000095e+00
min    -1.187493e+00       -1.586051e+00      -8.596388e-01 -7.340924e-01
25%    -7.603680e-01       -7.156873e-01      -6.949373e-01 -6.294640e-01
50%    -2.605318e-01       -1.336897e-01      -4.079064e-01 -3.288020e-01
75%     5.002991e-01        5.466501e-01       3.402167e-01  2.521727e-01
max     8.095365e+00        1.498024e+01       5.750510e+00  1.571080e+01
```

Out[3]:

	PovertyFamily	SingleHeadwithKids	LimitedEnglishPop	NoCarWorkers
PovertyFamily	1.000000	0.547829	0.587112	0.449034
SingleHeadwithKids	0.547829	1.000000	0.256104	0.304791
LimitedEnglishPop	0.587112	0.256104	1.000000	0.259051
NoCarWorkers	0.449034	0.304791	0.259051	1.000000

Example Dimension Reduction

In [4]:
```python
# Annoying error for matplotlib
import warnings
warnings.filterwarnings("ignore")

# Scatterplot matrix
axes = pd.plotting.scatter_matrix(scale_det, alpha=0.2)
```



Example Dimension Reduction

In [5]:
```python
# Now we are going to conduct PCA

# Helper function to turn principal component scores into nice pandas datafram
def pd_comp(PCA, data):
    res = PCA.transform(data)
    cols = ['PC' + str(i+1) for i in range(data.shape[1])]
    res_dat = pd.DataFrame(res,columns=cols)
    return res_dat

# sklearn object to fit PCA
pca = decomposition.PCA()
pca.fit(scale_det)
res = pd_comp(pca, scale_det)

# We get 4 new variables!
print( res.head(10) )

# And they have zero correlation with one another
res.corr()
```

```
        PC1       PC2       PC3       PC4
0 -0.470241  0.323328 -0.231908  0.457786
1 -1.638550  0.121828 -0.184339 -0.264323
2  3.017197  2.120470 -4.545233 -0.077607
3  0.795040  0.340068 -0.535677 -0.416899
4  1.429447  2.847041  2.714926 -0.524956
5  0.790415 -0.499015 -0.589200 -0.108288
6  0.703555  0.840933 -1.260273 -0.018250
7 -1.006926  0.300257  0.128568 -0.033911
```

Out[5]:

	PC1	PC2	PC3	PC4
PC1	1.000000e+00	-2.349428e-16	5.070204e-17	6.808179e-16
PC2	-2.349428e-16	1.000000e+00	-1.767719e-16	-3.564981e-16
PC3	5.070204e-17	-1.767719e-16	1.000000e+00	3.316231e-16
PC4	6.808179e-16	-3.564981e-16	3.316231e-16	1.000000e+00

Example Dimension Reduction

```
In [6]: # The first PC component tends to describe a larger amount of variance
        print('Explained Variance per each component')
        print(pca.explained_variance_ratio_) #PC1 55% of variance, PC2 19%, PC3 18%, etc.

        # The loadings tell us how each of the original variables
        # contributes to the new PCA results

        # A helper function to get the loadings, adapted from
        # https://scentellegher.github.io/machine-learning/2020/01/27/pca-loadings-sklearn.html
        def loadings(data,pca):
            comps = pca.components_.T
            cols = ['PC' + str(i+1) for i in range(comps.shape[0])]
            load_dat = pd.DataFrame(comps,columns=cols,index=list(data))
            return load_dat

        load_dat = loadings(scale_det,pca)
        load_dat
```

```
Explained Variance per each component
[0.55732931 0.19241149 0.17539124 0.07486796]
```

Out[6]:

	PC1	PC2	PC3	PC4
PovertyFamily	0.598198	-0.122298	-0.085184	0.787366
SingleHeadwithKids	0.474038	0.400756	-0.689855	-0.372536
LimitedEnglishPop	0.476419	-0.734364	0.149112	-0.459890
NoCarWorkers	0.436430	0.533992	0.703285	-0.172546

Example Dimension Reduction

In [7]:
```python
# The loadings show how the PC variables are created back into
# the original data

print( res['PC1'].head(5).round(2) )

pc1 = scale_det['PovertyFamily']*0.598198 + scale_det['SingleHeadwithKids']*0.474038 + \
      scale_det['LimitedEnglishPop']*0.476419 + scale_det['NoCarWorkers']*0.436430
pc1.head(5).round(2)

# Or this is a more automatic way
# (scale_det*load_dat['PC1']).sum(axis=1).head(5)
```

```
0   -0.47
1   -1.64
2    3.02
3    0.80
4    1.43
Name: PC1, dtype: float64
```

Out[7]:
```
0   -0.47
1   -1.64
2    3.02
3    0.80
4    1.43
dtype: float64
```

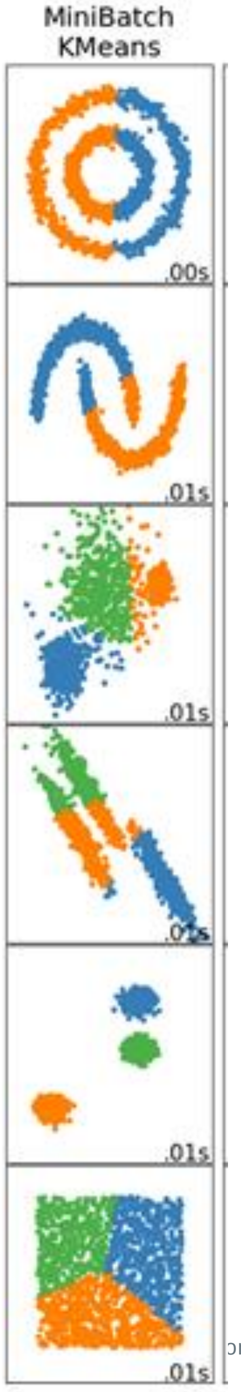
Other Techniques for Dimension Reduction

- PCA only relevant for numeric columns (e.g. dummy/one-hot/0-1 columns not applicable)

- Other dimension reduction techniques include
  - Hidden layers in Deep Learning (so have a target outcome)
  - Word2Vec embedding's for text strings
  - Simple tree based models (if-then rules, easy to translate to SQL)
  - Category reductions via Association Rules (e.g. common pairs of categories)

Unsupervised Learning

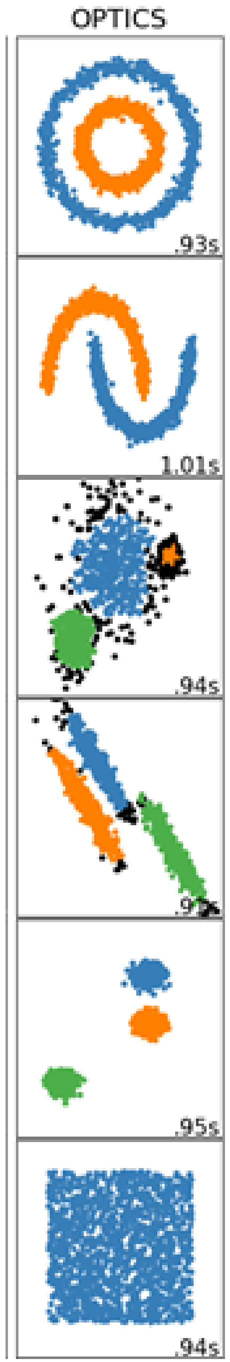
- What is it? Clustering like rows together
  - E.g. take 100,000 rows and produce 12 different groupings
- Why?
  - Market segmentation (create groups to do an intervention)
  - Exploratory data analysis (reduce complicated data into smaller groups) – DRG is an example!
  - For subsequent modelling
- The difficulties
  - Many different techniques (e.g. k-means, hierarchical clustering, graph clustering) rely on calculating *distances* between cases
  - User needs to decide many parameters, especially how to combine different fields to make a single metric distance



Unsupervised Learning

- K Means
  - Need to choose # of clusters
  - Clusters ~equal in size
  - Every point is within some cluster
  - Variance approximately equal, potentially good for circular shaped clusters of equal size

- OPTICS (hierarchical clustering)
  - Need to choose distance to not agglomerate and minimum cluster size
  - Points can be outliers (so in no cluster) or in one giant cluster
  - Clusters can grow to very weird shapes



Unsupervised Learning

- Example using de-identified surgical claims data from New York from 2009
- Clustering using 6 Fields
  - Age (in 10 year bins)
  - APR DRG numeric code (1-999)
  - Log (base 10) of total charges on bill
  - Medicare/Medicaid (0/1)
  - Length of Stay (capped at 120 days)
  - Admission Type (0 = Elective, 1 = Urgent, 2 = Emergency)

Unsupervised Learning

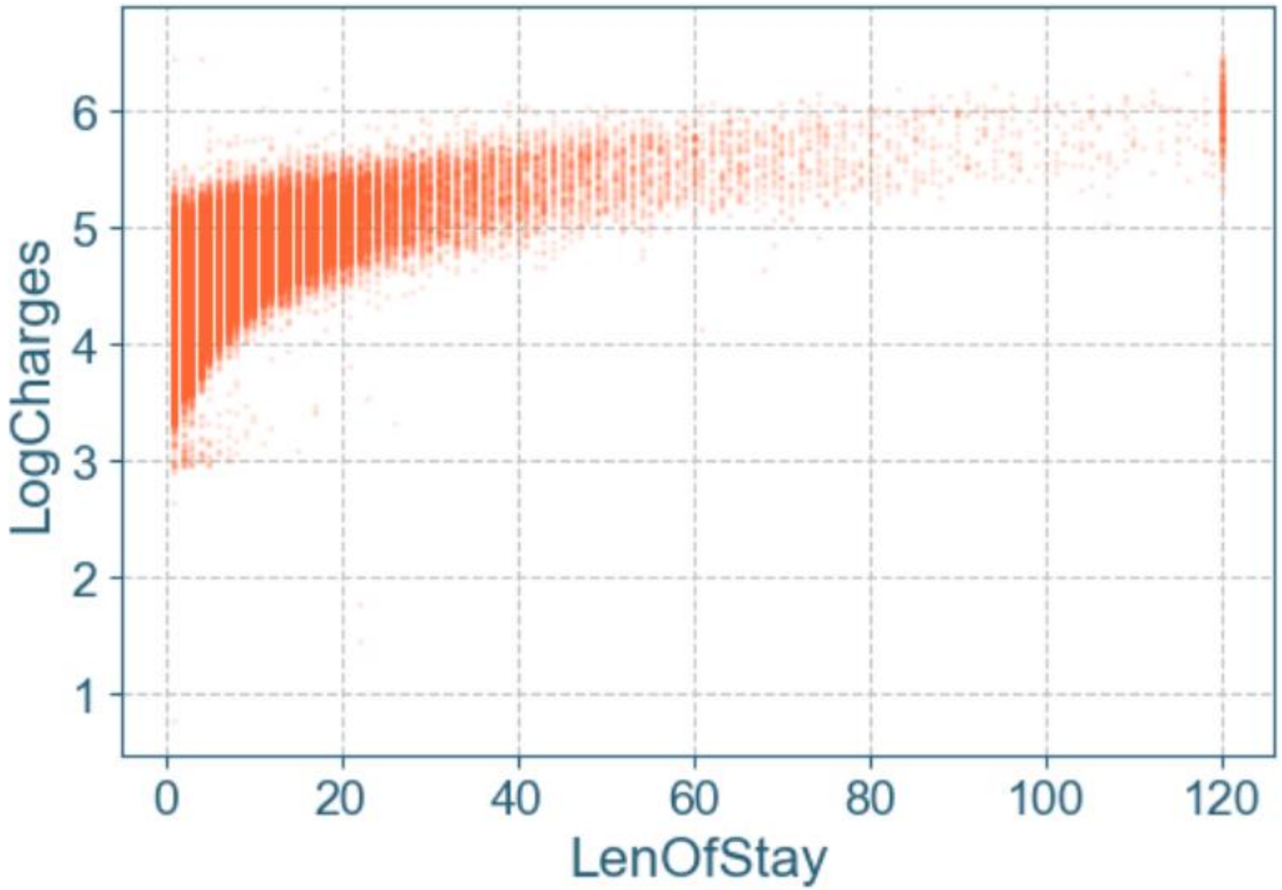
```
In [1]: import pandas as pd
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.cluster import OPTICS
        from matplotlib import pyplot as plt
        import os
        import sys

        # Adding in HMS plotstyle
        sys.path.append(r'C:\Users\e009156\Documents\GitHub\data-science-utils\plt')
        import hms_plotstyle

        # Reading in Data and Prepping it
        os.chdir(r'C:\Users\e009156\Documents\GitHub\data-science-utils\education\Intro_DataScience\Dime
        sparc = pd.read_csv('SparcSample.csv')
        # To make it simpler, lets only look at surgical procedures (still over 100k observations)
        sparc = sparc[sparc['Surgical'] == 1].copy()
        sparc.drop(columns=['Surgical','APRSevere'], inplace=True)
        sparc.describe().T
```

Out[1]:

	count	mean	std	min	25%	50%	75%	max
APRDRG	155430.0	335.181104	189.412420	1.000000	175.0000	302.000000	512.000000	952.000000
AgeBin	155430.0	54.090620	20.357301	10.000000	40.0000	60.000000	80.000000	80.000000
LenOfStay	155430.0	6.017275	9.593985	1.000000	2.0000	3.000000	6.000000	120.000000
MedicareMedicaid	155430.0	0.398051	0.489498	0.000000	0.0000	0.000000	1.000000	1.000000
LogCharges	155430.0	4.474219	0.408140	0.778151	4.1928	4.446529	4.726112	6.612332
AdmissEmergency	155430.0	0.865026	0.932242	0.000000	0.0000	0.000000	2.000000	2.000000

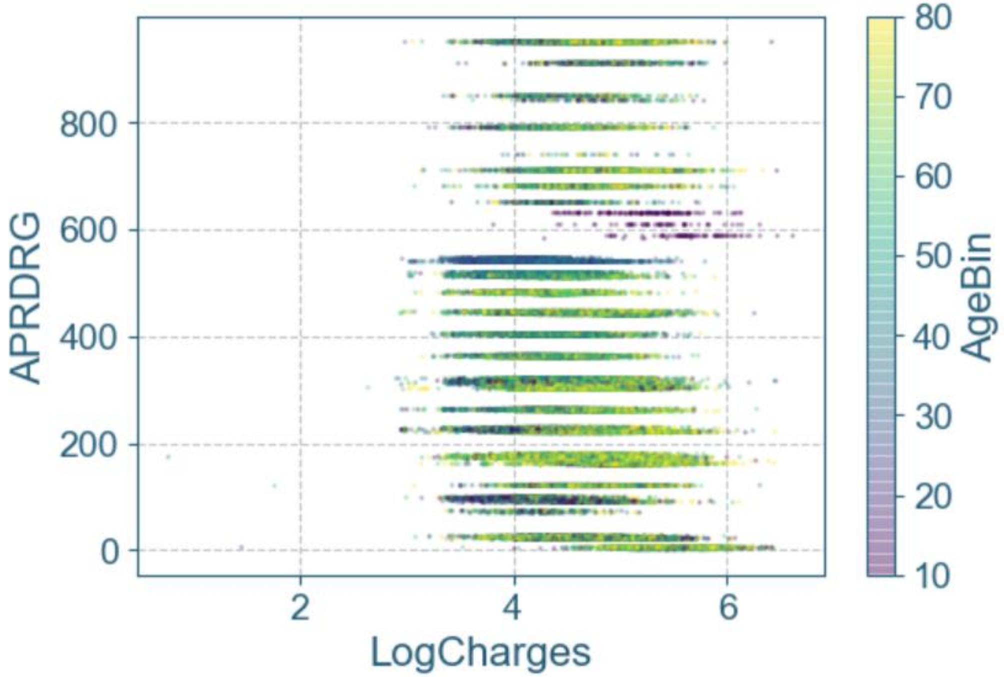
Unsupervised Learning

```
In [2]: ax = sparc.plot.scatter(x='LenOfStay', y='LogCharges', s=1, alpha=0.1)
```



Unsupervised Learning

```
In [3]: # Annoying error
        import warnings
        warnings.filterwarnings("ignore")

        # Error with cutting off X axis when using color and pandas plotting
        fig, ax = plt.subplots()
        ax = sparc.plot.scatter(x='LogCharges', y='APRDRG', c='AgeBin',
                                colormap='viridis', ax=ax, s=1, alpha=0.25)

        # 600 codes are neonate
        # See https://www.health.ny.gov/facilities/hospital/reimbursement/apr-drg/weights/2018-07-01_final_weights.htm
```



Unsupervised Learning

```
In [4]: # MinMax scaling each column to 0/1
        scaler = MinMaxScaler()
        sparcS = sparc.sample(20000, random_state=10) #10k is 13 seconds, 20k is not quite a minute
        scaler.fit(sparcS)
        sparc_scaled = pd.DataFrame(scaler.transform(sparcS),columns=list(sparcS))

        # OPTICS hierarchical clustering, takes a few minutes!
        from datetime import datetime
        print(datetime.now()) #to show how long it takes
        clustering = OPTICS(min_samples=200, max_eps=3) #min samples means a cluster has to have at least 500
                                                        #claims, max_eps is the max Euclidean distance
        clustering.fit(sparc_scaled)
        print(datetime.now())

        # Adding labels back into dataset
        sparcS['ClusterLabel'] = clustering.labels_
        sparcS['ClusterLabel'].value_counts() #-1 means it is an outlier, in no cluster

        2021-04-28 12:16:06.532727
        2021-04-28 12:16:34.444145

Out[4]: -1     9613
         5     2665
         1     1838
         4     1334
         9     1191
         2     1082
         3      501
         6      431
         0      425
         7      418
         8      269
         10     233
        Name: ClusterLabel, dtype: int64
```

Unsupervised Learning

In [5]:
```
# Now lets aggregate the characteristics for each cluster
aggG = sparcS.groupby('ClusterLabel').mean()
aggG.sort_values(by='LogCharges', ascending=False, inplace=True)
aggG
```

Out[5]:

ClusterLabel	APRDRG	AgeBin	LenOfStay	MedicareMedicaid	LogCharges	AdmissEmergency
0	204.992941	80.000000	6.432941	1.00000	4.621469	1.000000
9	229.214945	60.000000	6.042821	0.00000	4.593270	2.000000
7	211.674641	60.000000	4.626794	0.00000	4.586308	1.000000
6	269.638051	80.000000	4.211137	0.00000	4.530215	0.000000
1	269.360174	80.000000	4.300871	1.00000	4.522712	0.000000
2	278.124769	60.000000	4.265250	1.00000	4.519124	0.000000
-1	342.671487	48.371476	8.039218	0.47675	4.515160	1.379278
5	315.526079	60.000000	3.499062	0.00000	4.478188	0.000000
8	529.078067	40.000000	3.598513	0.00000	4.167527	1.000000
4	531.648426	40.000000	3.158921	0.00000	4.149872	0.000000
3	538.471058	25.000000	3.489022	0.00000	4.093349	0.000000
10	531.180258	25.000000	3.240343	0.00000	4.092125	2.000000

Unsupervised Learning

```
In [6]: # Lets check out top two groups and bottom two groups
        gp = aggG.index.to_list()
        sparcS[sparcS['ClusterLabel'] == gp[0]]
```

Out[6]:

	APRDRG	AgeBin	LenOfStay	MedicareMedicaid	LogCharges	AdmissEmergency	ClusterLabel
322830	20	80	14	1	5.003663	1	0
150586	221	80	9	1	4.846931	1	0
476449	161	80	5	1	5.077445	1	0
635426	301	80	5	1	4.374224	1	0
88548	175	80	2	1	4.462500	1	0
...	...	...	...	...	...	...	...
651790	165	80	12	1	4.882123	1	0
504540	161	80	1	1	5.161348	1	0
586223	305	80	17	1	4.611939	1	0
130964	173	80	16	1	4.796859	1	0
185797	228	80	1	1	3.924331	1	0

425 rows × 7 columns

Unsupervised Learning

```
In [8]: sparcS[sparcS['ClusterLabel'] == gp[-1]] #540 is C-section
```

Out[8]:

	APRDRG	AgeBin	LenOfStay	MedicareMedicaid	LogCharges	AdmissEmergency	ClusterLabel
639062	540	25	4	0	4.055509	2	10
295062	540	25	3	0	4.003383	2	10
257231	540	25	6	0	4.281826	2	10
248836	540	25	3	0	3.953037	2	10
441031	540	25	3	0	3.964303	2	10
...	...	...	...	...	...	...	...
372893	540	25	3	0	4.212135	2	10
573714	540	25	4	0	4.018169	2	10
433393	540	25	3	0	3.949146	2	10
348790	540	25	4	0	3.988648	2	10
7736	545	25	1	0	4.139061	2	10

233 rows × 7 columns

Future Advanced Topics

- Feature Selection (e.g. selecting 5 columns out of 100 in the modelling stage)

- Text analysis and high dimensional analysis

- Deep Learning and hidden layers

- Mixture Models for fuzzy clustering with a target outcome

Questions?

Future Topics

Have requests?

Let me know!

Introduction to Data Science Course Outline

Andrew Wheeler, PhD, andrew.wheeler@hms.com

- Lesson 01: Data Science 101
- Lesson 02: Machine Learning 101
- Lesson 03: Evaluating Predictions
- Lesson 04: Intro Data Transformation in Python
- Lesson 05: Data Visualization 101
- Lesson 06: Feature Engineering
- Lesson 07: Missing Data
- Lesson 08: Big Data and Parallel Computing Intro
- Lesson 09: Dimension Reduction and Unsupervised Learning
- Lesson 10: High Cardinality (Many Categories)
- Lesson 11: Intro to Forecasting
- Lesson 12: Conducting Experiments

Dimension Reduction and Unsupervised Learning

Data Science and Machine Learning Team

??/??/2021

Andrew Wheeler, PhD

andrew.wheeler@hms.com