Machine learning Nano degree Capstone Project

Report on Project for Clustering of Medicare 2014 Part D Opioid claims to identify over prescribing providers

Definition

Project Overview

Detection and Investigation of Fraud Waste and Abuse in Healthcare using Machine learning. During Fiscal Year (FY) 2016, the Federal Government won or negotiated over $2.5 billion in health care fraud judgments and settlements. Up until recent past almost all healthcare payers have been practicing "Pay" and "Chase" model for Fraud Waste and Abuse. This can be changed using predictive modeling; health Insurance payers have large amounts of claims Data available, upon which we can analyze to build various ML models to identify fraud in near real time when claims are adjudicated.

This FWA does not end with Provider; it is even committed by beneficiaries or members. Medicaid paid for over 34 million opioid claims in 2012, with 15 percent of Medicaid enrollees having at least one opioid prescription. About 5% received prescriptions from five or more prescribers and about 2% filled them at five or more pharmacies

I am utilizing publicly available "Medicare Provider Utilization and Payment Data: 2014 Part D Prescriber" CMS data, following is the link for the data.

> https://data.cms.gov/Medicare-Part-D/Medicare-Provider-Utilization-and-Payment-Data-201/465c-49pb

The file has 24121660 (24.1 million) records. I tried to load the csv file using "pd.read_csv", it took forever on my local machine and as well as AWS. I had to Unix commands to scan through the file, apply some filters (grep) and identify a state where the records are less that 1 million. I have tried using many state provider files such as NY, CA, PA the laptop that I have could not manage and the AWS account that I have also could not manage. I have narrowed down to Alabama. So from the above data set I shall be using Alabama providers/claims data for this project. To perform provider benchmarking and identify the providers who do not fit the normal patter of opioid prescription.

Following is the pdf detailing about the columns in the dataset

> https://data.cms.gov/api/views/465c-49pb/files/0931bfc7-1069-4437-961b-e3f43e26ac33?download=true&filename=Part_D_Prescriber_PUF_Methodology_2017-05-25.pdf

Problem Statement

Opioid crisis is another issue that has been troubling USA. In this Capstone project I shall perform provider benchmarking on a subset of "Pain Management" Providers who prescribe opioids for Pain relief. I will be performing clustering to identify the providers who are prescribing opioids above the normal levels. This can be achieved by clustering the providers based on the claims submitted; smaller clusters can be used to investigate for over prescription/fraud. The goal is to get the list of providers who should be investigated for opioid prescription. Whether a provider should be prosecuted for opioid over prescription shall be decided by reviewing the list produced by clustering.

Metrics

Silhouette Coefficient: If the ground truth labels are not known, evaluation must be performed using the model itself. The Silhouette Coefficient (sklearn.metrics.silhouette_score) is an example of such an evaluation, where a higher Silhouette Coefficient score relates to a model with better-defined clusters. The Silhouette Coefficient is defined for each sample and is composed of two scores:

a: The mean distance between a sample and all other points in the same class.
b: The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample is then given as:
$s = (b - a)/\max(a, b)$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

http://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation

Analysis

Data Exploration

We do not need all the Medicare Part D claim data; this file consists of prescription claims that are not for opioids also. So let us filter for Opioid claims, and then we will have to do some data clean such as removing comma from columns where the values are integers and floats. Convert the columns in proper data types. Determine the columns where there are only flags and remove them. Also please note that Medicare is mostly for 65 and above population, so we do not need the data that is specific to age 65 and above.

For a list of drugs that include opioids, visit

> https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/OpioidDrugList.zip

Also please note that some of the columns have blank (NaN) values. Fill NaN values with 5, rather using sklearn.preprocessing "Imputer", taken from the pdf detailing about the columns in the dataset. Following is the copy past of the column descriptions when read are self-explanatory as to why I choose 5. This is mid value between 1 and 10(or 11, using 5.5 does not make sense so took the lower side). Please read through patiently.

- bene_count – The total number of unique Medicare Part D beneficiaries with at least one claim for the drug. Counts fewer than 11 are suppressed and are indicated by a blank.
- total_claim_count – The number of Medicare Part D claims. This includes original prescriptions and refills. Aggregated records based on total_claim_count fewer than 11 are not included in the data file.
- total_30_day_fill_count – The aggregate number of Medicare Part D standardized 30-day fills. The standardized 30-day fill is derived from the number of days supplied on each Part D claim divided by 30. Standardized 30-day fill values less than 1.0 were bottom-coded with a value of 1.0 and standardized 30- day fill values greater than 12.0 were top-coded with a value of 12.0.
- bene_count_ge65_suppress_flag – A flag indicating the reason the bene_count_ge65 variable is suppressed.
    - "*" = Primary suppressed due to bene_count_ge65 between 1 and 10.
    - "#" = Counter suppressed because the "less than 65 year old" group (not explicitly displayed) contains a beneficiary count between 1 and 10, which can be mathematically determined from bene_count_ge65 and bene_count.
- ge65_suppress_flag – A flag that indicates the reason the total_claim_count_ge65, total_30_day_fill_count ge65, total_day_supply_ge65, and total_drug_cost_ge65 variables are suppressed.
    - "*" = Primary suppressed due to total_claim_count_ge65 between 1 and 10.
    - "#" = Counter suppressed because the "less than 65 year old" group (not explicitly displayed) contains a small claim count between 1 and 10, which can be mathematically determined from the total_claim_count_ge65 and total_claim_count.
- total_30_day_fill_count_ge65 – The number of Medicare Part D standardized 30-day fills for beneficiaries age 65 and older. The standardized 30-day fill is derived from the number of days supplied on each Part D claim divided by 30. Standardized 30-day fill values less than 1.0 were bottom-coded with a value of 1.0 and standardized 30-day fill values greater than 12.0 were top-coded with a value of 12.0. If

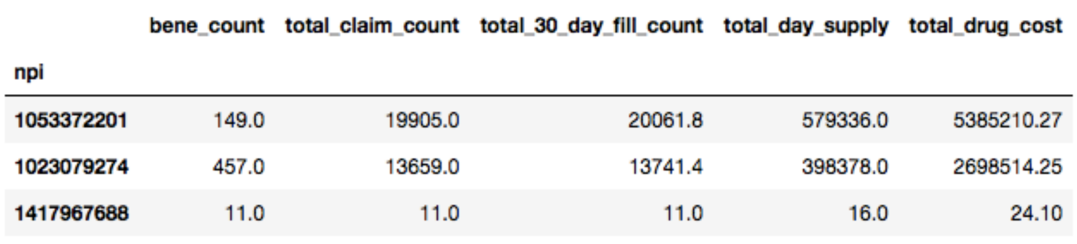
total_claim_count_ge65 is suppressed, this variable is suppressed. A blank indicates the value is suppressed. See ge65_suppress_flag regarding suppression of data.

We will do some data exploration to find out whether we would need preprocessing of data. Mostly we will end up doing some kind of pre-processing of data.

One key aspect to take care here is that; a single NPI (provider) can prescribe multiple generic Opioids, so we group the claims by NPI. Please note that NPI stands for National Provider Identification. It is an intelligence free unique number assigned to each provider in the nation. Grouping the data by NPI helps us aggregate the data. We will end with aggregate values for all columns when we do sum on the group by NPI. Another important point for me to do this is to index the data by NPI and most importantly my laptop can handle the record count.

Let us also take some samples here, let us have some fraud providers in the sample. Just doing a google of providers charged with opioid prescription, we will get a list of providers
https://www.justice.gov/usao-sdal/pr/dr-couch-and-dr-ruan-sentenced-240-and-252-months-federal-prison-running-massive-pill

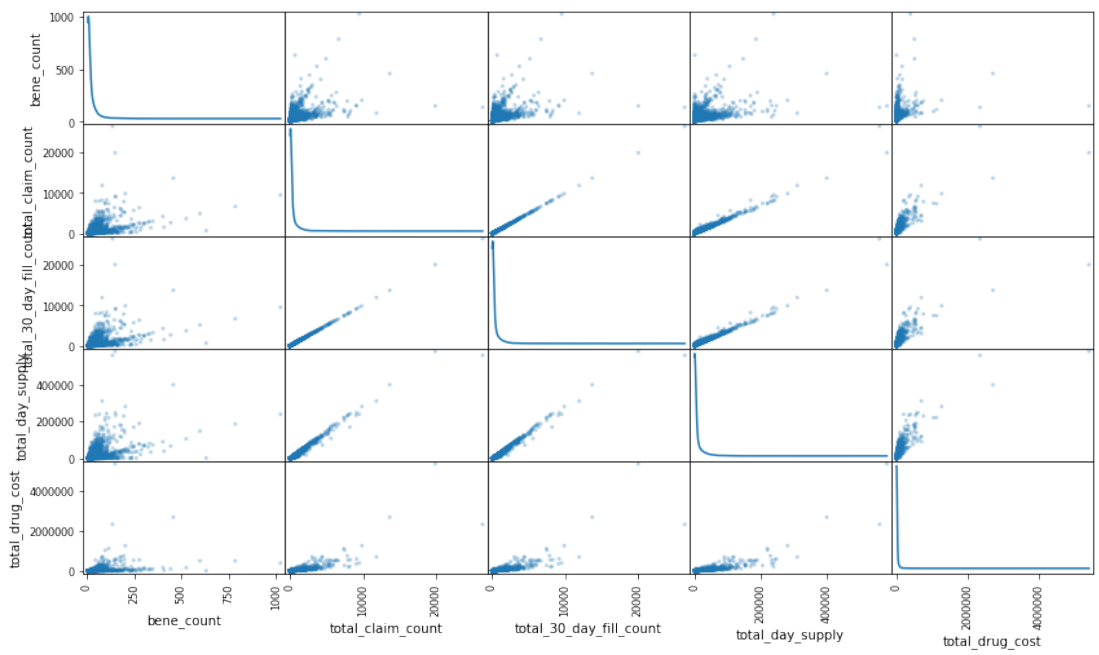
The NPI numbers of the two providers are 1053372201 and 1023079274.



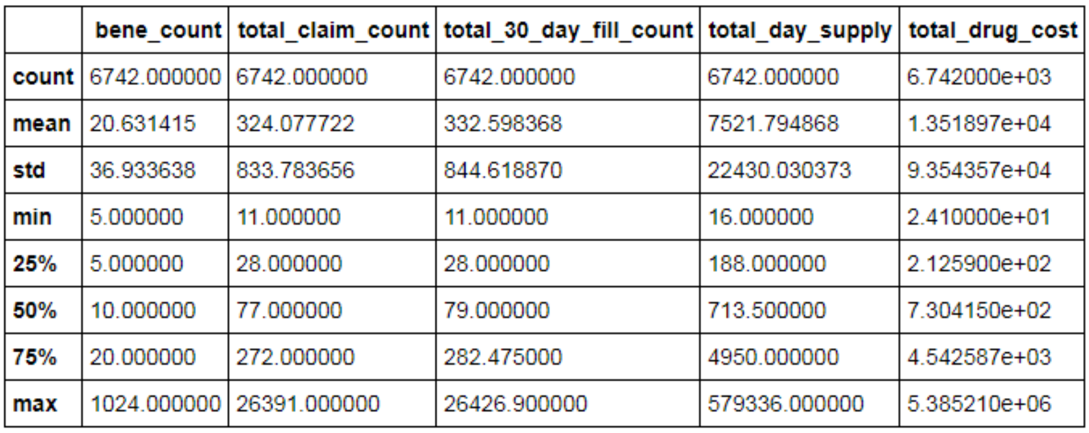
	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
npi					
1053372201	149.0	19905.0	20061.8	579336.0	5385210.27
1023079274	457.0	13659.0	13741.4	398378.0	2698514.25
1417967688	11.0	11.0	11.0	16.0	24.10

Exploratory Visualization

Visualizing the data help me understand the data structure. Here is what the scatter plot of the aggregated data by NPI looks like.

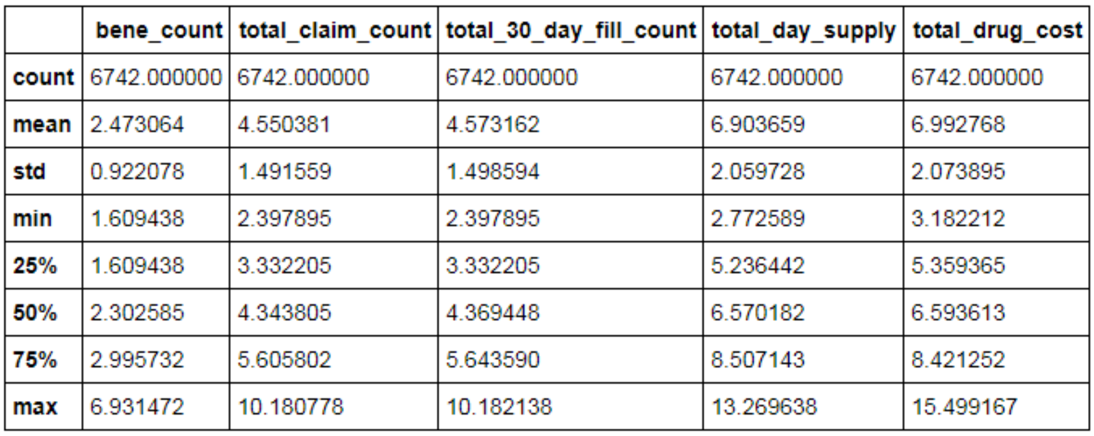


We can notice that there is an overall trend as the feature values increase. Even the total_drug_cost shows a minor trend. We have not done preprocessing of features so we will not be able to see some kind of distribution of data.



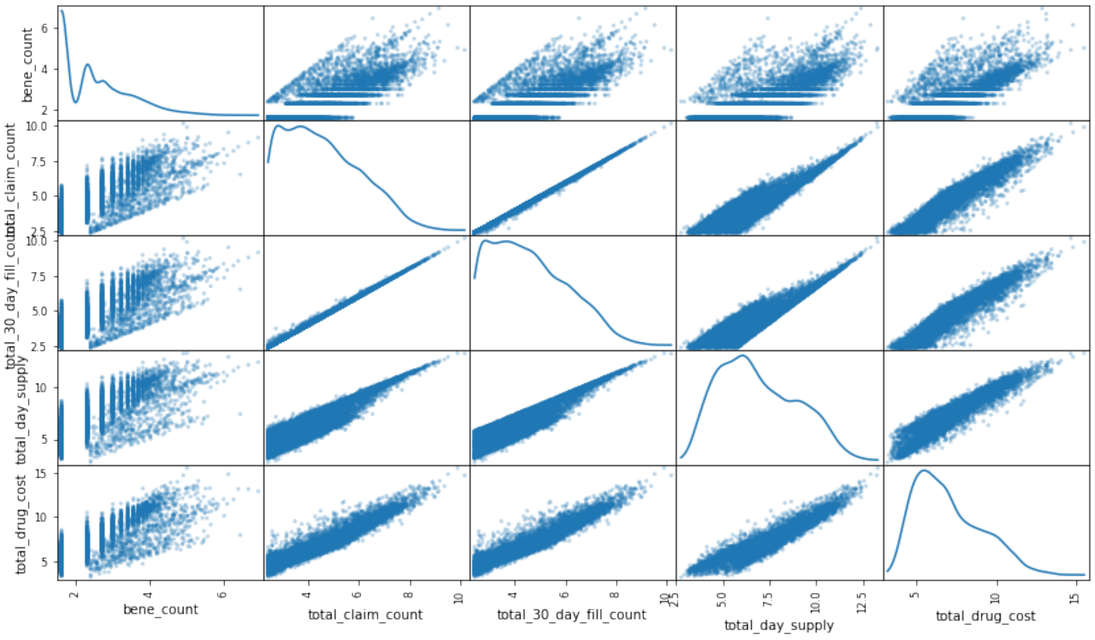
	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
count	6742.000000	6742.000000	6742.000000	6742.000000	6.742000e+03
mean	20.631415	324.077722	332.598368	7521.794868	1.351897e+04
std	36.933638	833.783656	844.618870	22430.030373	9.354357e+04
min	5.000000	11.000000	11.000000	16.000000	2.410000e+01
25%	5.000000	28.000000	28.000000	188.000000	2.125900e+02
50%	10.000000	77.000000	79.000000	713.500000	7.304150e+02
75%	20.000000	272.000000	282.475000	4950.000000	4.542587e+03
max	1024.000000	26391.000000	26426.900000	579336.000000	5.385210e+06

The variance, std, min and max shows that data spread from the mean is huge on both sides of mean, this has to be reduce by performing data preprocessing, specifically feature scaling. Apply feature scaling by applying natural log to the values we get the following description of the data



	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
count	6742.000000	6742.000000	6742.000000	6742.000000	6742.000000
mean	2.473064	4.550381	4.573162	6.903659	6.992768
std	0.922078	1.491559	1.498594	2.059728	2.073895
min	1.609438	2.397895	2.397895	2.772589	3.182212
25%	1.609438	3.332205	3.332205	5.236442	5.359365
50%	2.302585	4.343805	4.369448	6.570182	6.593613
75%	2.995732	5.605802	5.643590	8.507143	8.421252
max	6.931472	10.180778	10.182138	13.269638	15.499167

Data spread is reduced to a large extent. Following is what the scatter plot will look like once data has been pre-processed



An important thing to note we should not be performing outlier detection on this data, because there is a high probability that providers submitting high volumes of claims would be on the upper outliers and we would end loosing key data.

Algorithms and Techniques

Before we proceed we will have to ask this question, do we need to do PCA here? If we look at the data, we will notice that almost all the features are related with each other. The total_claim_count is proportional to the bene_count, the rest of the features total_30_day_fill_count, total_day_supply and total_drug_cost are all proportional to total_claim_count. The more the total_claim_count the more the rest of the columns. In general PCA is use to reduce the number of features but here even though we do not have many features we shall use it to combine the features which would represent the complete data. This way we will end up with features that are independent of each other. Now I could perform combination of featured manually by dividing total_30_day_fill_count, total_day_supply and total_drug_cost features by total_claim_count feature. I will still prefer pca as it would derive the inter dependent feature mathematically rather than my assumptions

Finally perform the clustering on the cleaned up and preprocessed data. I will be selecting GaussianMixture, KMeans, and MiniBatchKMeans from sklearn to see which one would be the best. Since we do not know the underlying data structure we shall use silhouette coefficient to find out the best one which will tell us how many clusters are present in the data.

Why did I select Kmeans/ MiniBatchKMeans?

- I made an assumption about the data, in specific about the behavior of the providers. I assumed that providers who are not fraudulent vs. fraudulent would have their respective opioid prescribing patterns. Now this would create clusters of data having similar variance. Assuming that there would be three patterns of behavior such as providers prescribing very low opioids, normal level of opioids prescriptions and then very high levels, now all of these might have same variance for the samples in the each of these clusters. I have selected Kmeans algorithm because of its simplicity, it would calculate the mean of the samples in the cluster and adjust the centroids accordingly. So this would create clusters of providers who have same/similar opioid prescription patterns.
- Now MiniBatchKMeans in not much different from kmeans except that it does kmeans clustering in batches. I selected this just as a back up in case my system slows down performing kmeans clustering.

Why did I select GaussianMixture?

- Looking at the scatter plot after we have done preprocessing of data, I notice normal distribution of data w.r.t to 2 features (total_day_supply and total_drug_cost). This GaussianMixture algorithm is specifically used for fitting data where there are multiple normally distributed data of features. Now this being a probabilistic model we will end up having soft clustering, which I would not prefer because we are looking for a definitive list, having a provider on both lists is not advisable for the type of classification we are trying to do here. I will still use the GaussianMixture to see what the metrics look like.

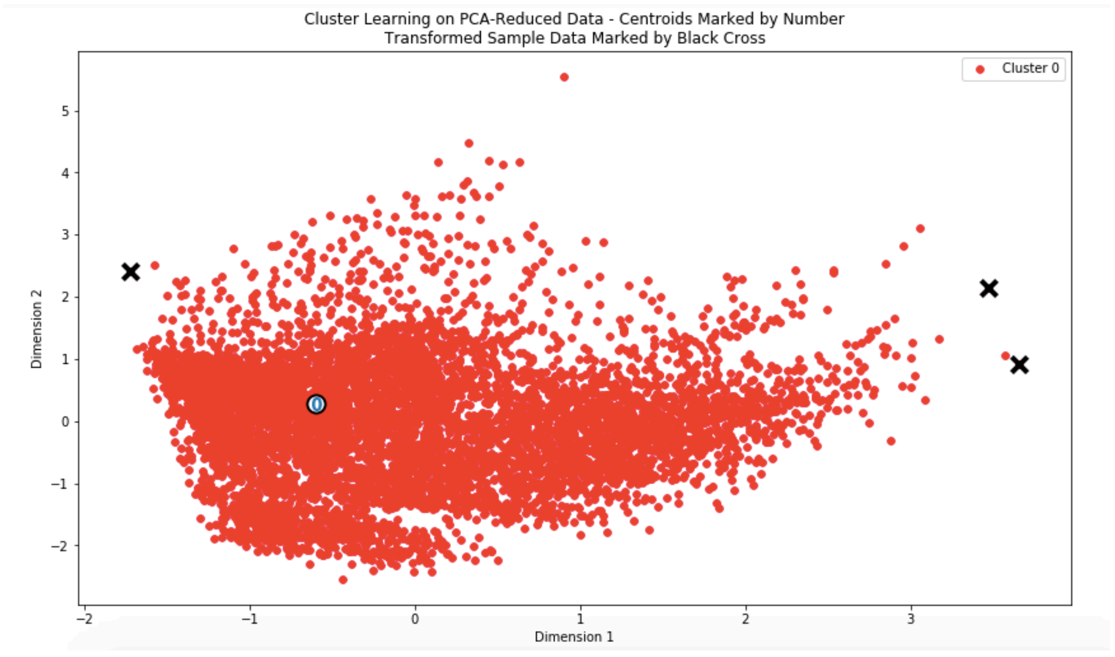
Benchmark

There might be several of models that have been developed since I am using 2014 CMS data and since CMS itself came up with models in 2010 to identify the fraudulent providers. The office of inspector general of department of Health and Human Services has come up with list of excluded providers, this is called exclusion list. Now please keep in mind that this exclusion list is for all types of frauds/misdiagnosis/malpractices not just Opioid over prescription. Goal is identify the list of providers who shall be further investigated.

OIG exclusion list, list of providers identified as fraudulent

https://oig.hhs.gov/exclusions/exclusions_list.asp https://oig.hhs.gov/exclusions/downloadables/UPDATED.csv

Let us create a very basic clustering. I have used use MeanShift algorithm, with all default parameters. The clustering took a while and gave a flat clustering. The algorithm clustered the whole data into one blob. Following is the cluster and its centroid.



Please note for us to generate silhouette_score for the benchmark model it is not possible because the Number of labels is 1. Valid values are 2 to n_samples - 1 (inclusive)

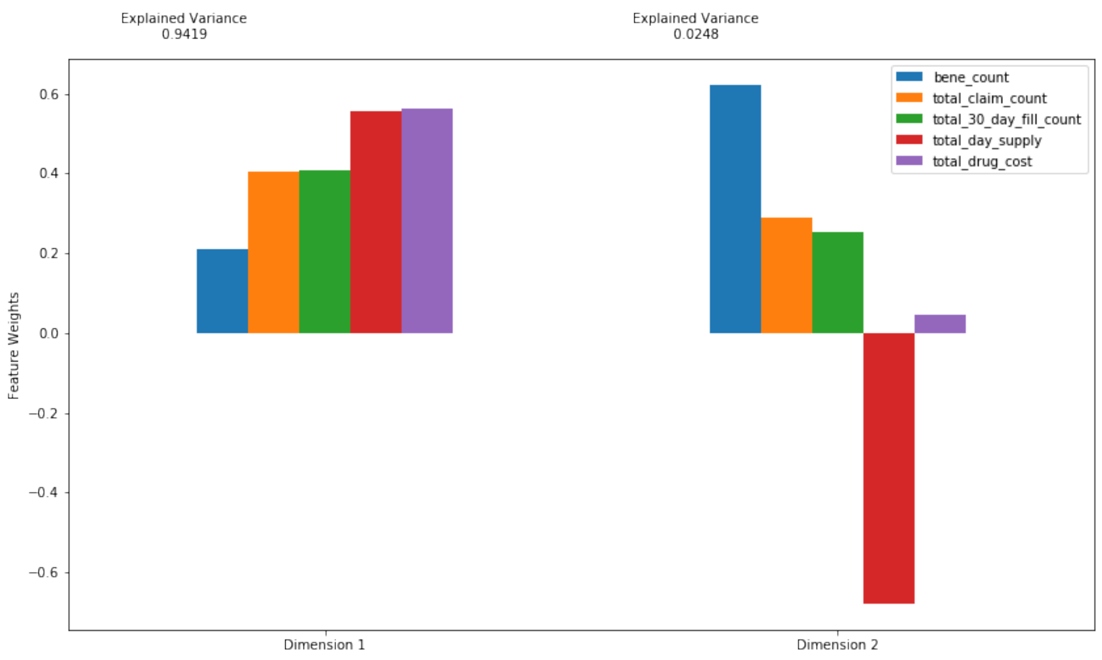
Methodology

Data Preprocessing

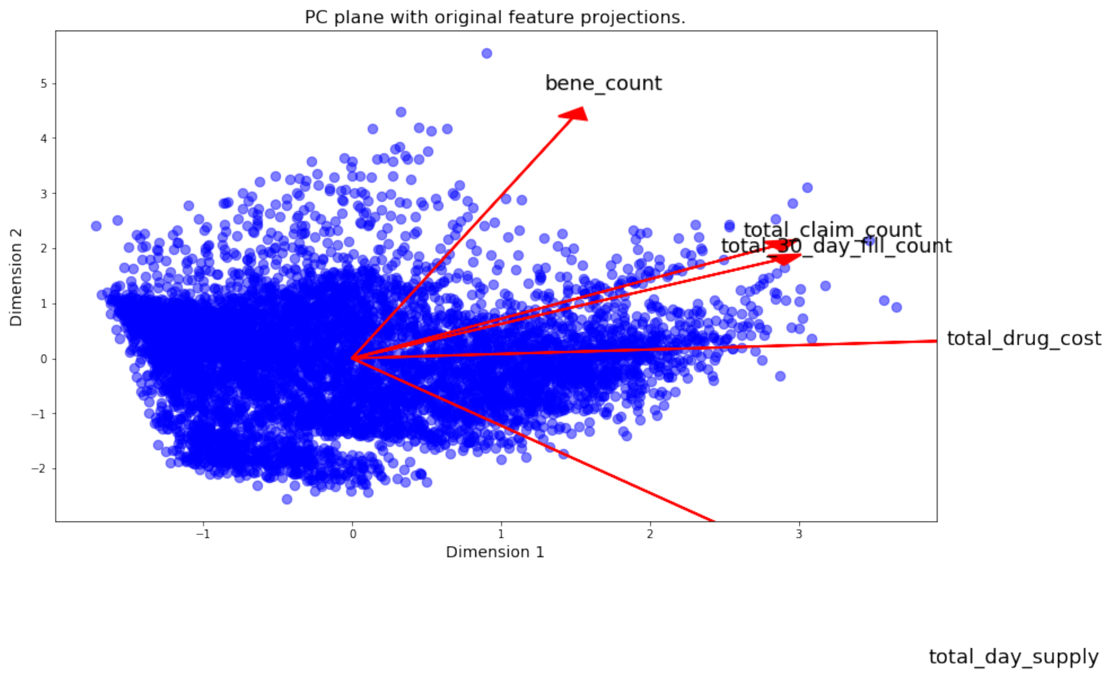
Feature Transformation

We will use principal component analysis (PCA) to see what combination of features/columns best represents the underlying data structure. We will start with 2 dimensions and later see what the "explained variance" for each dimension is. Let us examine the total variance to see whether the pca components were able to completely represent the data. Later on we will proceed to what is the best value for pca component by combining this with silhouette_score.

Looks like the first dimension that is Dimension 1 itself represents the whole of the data. The total variance of the two dimensions is close to 1 which represents the complete data well.



Let us create a biplot and see what feature are closely related, we will notice that total_claim_count and total_30_day_fill_count are closely related.



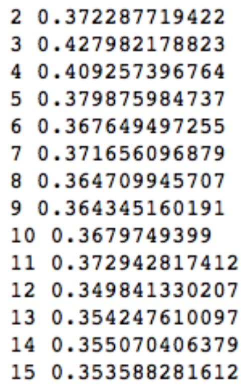
Implementation

I shall use 2 algorithms initially to build the clusters, later I shall check fine tune it to find out the best parameter by combining the PCA components and cluster size.

I will initially select Kmeans, following are the reasons for doing so.

- It is simple in nature, easy to build and execute the algorithm.
- Because of simple in nature it is one of the faster clustering algorithms.

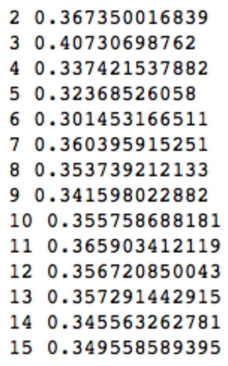
There is one major disadvantage for k means - Clusters are highly dependent on the initialization of the centroids. Local minimum. Following is the score that I have got with 2 pca_components and clusters in the range of (2,16)



```
2 0.372287719422
3 0.427982178823
4 0.409257396764
5 0.379875984737
6 0.367649497255
7 0.371656096879
8 0.364709945707
9 0.364345160191
10 0.3679749399
11 0.372942817412
12 0.349841330207
13 0.354247610097
14 0.355070406379
15 0.353588281612
```

If you look at it the cluster size 3 has the highest score.

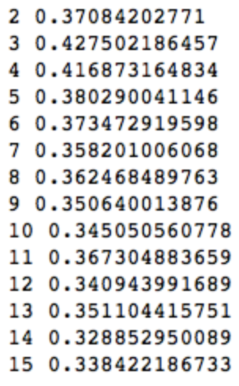
I will also go with Gaussian Mixture Model clustering. GMM clustering is more flexible because you can view it as soft clustering method. Soft clustering methods assign a score to a data point for each cluster. The value of the score indicates the association strength of the data point to the cluster. As opposed to hard clustering methods, soft clustering methods are flexible in that they can assign a data point to more than one cluster. Following is the score that I have got with 2 pca_components and clusters in the range of (2,16)



```
2 0.367350016839
3 0.40730698762
4 0.337421537882
5 0.32368526058
6 0.301453166511
7 0.360395915251
8 0.353739212133
9 0.341598022882
10 0.355758688181
11 0.365903412119
12 0.356720850043
13 0.357291442915
14 0.345563262781
15 0.349558589395
```

Even in here also when you look at the scores you will notice the cluster size 3 has the highest score.

Following are the scores for MiniBatchKMeans, they are almost same as kmeans.



```
2 0.37084202771
3 0.427502186457
4 0.416873164834
5 0.380290041146
6 0.373472919598
7 0.358201006068
8 0.362468489763
9 0.350640013876
10 0.345050560778
11 0.367304883659
12 0.340943991689
13 0.351104415751
14 0.328852950089
15 0.338422186733
```

Even in here also when you look at the scores you will notice the cluster size 3 has the highest score. Now whether this is the best score or not we will get to know when we search through the combination of parameters and other algorithms.

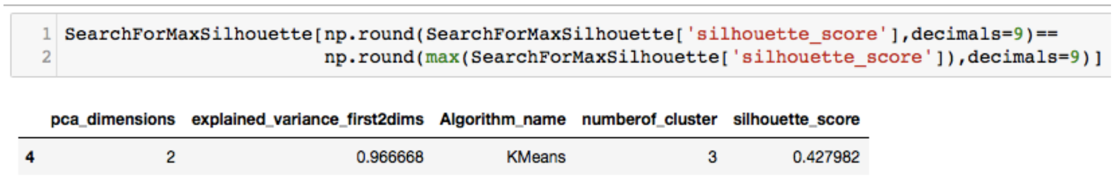
To come to this current state I had to redo the project twice before my initial submission, reasons being

1. Understanding Data: Without reading enough about the dataset I proceeded to do the clustering, later I realized that the original dataset consists of all claims not just opioids. Wasted time here.
2. Then while performing group by I complicated the dataset by grouping by NPI and generic drug name. To perform clustering and then recover the data has become hard.
3. In the first run I completely ignored PCA and did my own feature transformation by diving the features total_day_supply and total_drug_cost each by total_claim_count, results looked good but still I wanted to use PCA to see the close correlation between columns to my surprise I was wrong and noticed that total_claim_count and total_30_day_fill_count are closely correlated.
4. Data clean up took huge time for me, my lack of proficiency in python was clearly a pain point.

Refinement

Clustering, selecting the algorithm, pca components and clusters

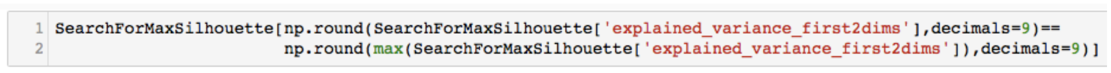
Here we will select from 3 different algorithms. I would prefer hard clustering than soft clustering the reason being we do not want to have a provider labeled for investigation and not for investigation. I am going to play around with multiple algorithms and then multiple clusters to see which combination has highest silhouette_score. Rather than doing this combination manually, I shall put this in a data frame and pick the record with max "total explained variance" for first two dimensions and max silhouette_score. If I were to select only the max silhouette_score from the dataframe then this what I would get



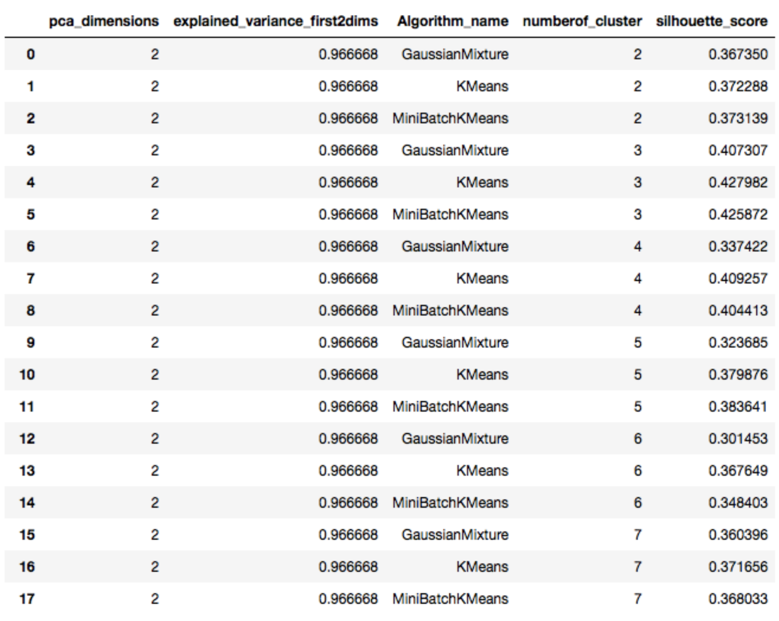
```
SearchForMaxSilhouette[np.round(SearchForMaxSilhouette['silhouette_score'],decimals=9)==
                       np.round(max(SearchForMaxSilhouette['silhouette_score']),decimals=9)]
```

	pca_dimensions	explained_variance_first2dims	Algorithm_name	numberof_cluster	silhouette_score
4	2	0.966668	KMeans	3	0.427982

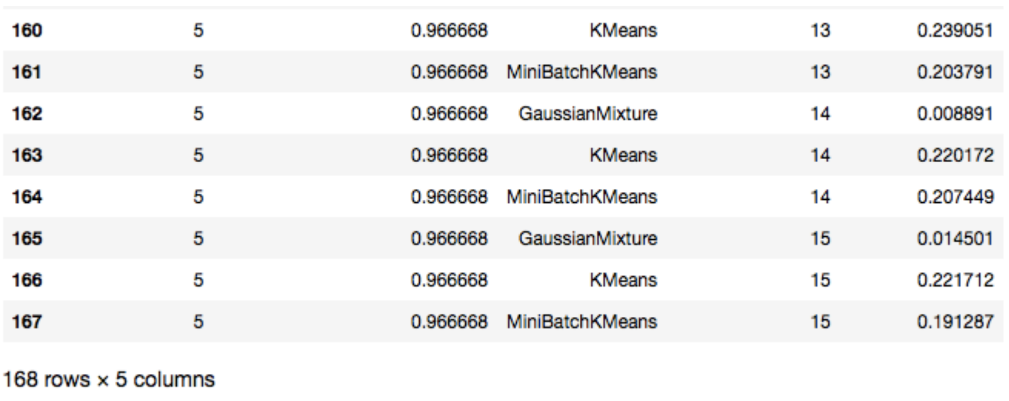
If I were to select the rows with max total variance for the first two columns, this is what we get. We get around 168 rows.



```
SearchForMaxSilhouette[np.round(SearchForMaxSilhouette['explained_variance_first2dims'],decimals=9)==
                          np.round(max(SearchForMaxSilhouette['explained_variance_first2dims']),decimals=9)]
```



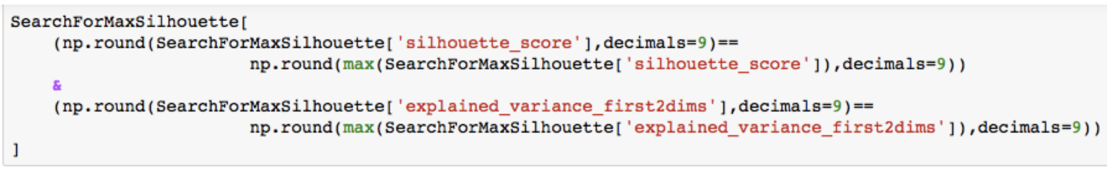
	pca_dimensions	explained_variance_first2dims	Algorithm_name	numberof_cluster	silhouette_score
0	2	0.966668	GaussianMixture	2	0.367350
1	2	0.966668	KMeans	2	0.372288
2	2	0.966668	MiniBatchKMeans	2	0.373139
3	2	0.966668	GaussianMixture	3	0.407307
4	2	0.966668	KMeans	3	0.427982
5	2	0.966668	MiniBatchKMeans	3	0.425872
6	2	0.966668	GaussianMixture	4	0.337422
7	2	0.966668	KMeans	4	0.409257
8	2	0.966668	MiniBatchKMeans	4	0.404413
9	2	0.966668	GaussianMixture	5	0.323685
10	2	0.966668	KMeans	5	0.379876
11	2	0.966668	MiniBatchKMeans	5	0.383641
12	2	0.966668	GaussianMixture	6	0.301453
13	2	0.966668	KMeans	6	0.367649
14	2	0.966668	MiniBatchKMeans	6	0.348403
15	2	0.966668	GaussianMixture	7	0.360396
16	2	0.966668	KMeans	7	0.371656
17	2	0.966668	MiniBatchKMeans	7	0.368033



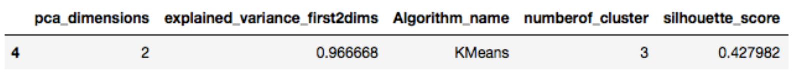
160	5	0.966668	KMeans	13	0.239051
161	5	0.966668	MiniBatchKMeans	13	0.203791
162	5	0.966668	GaussianMixture	14	0.008891
163	5	0.966668	KMeans	14	0.220172
164	5	0.966668	MiniBatchKMeans	14	0.207449
165	5	0.966668	GaussianMixture	15	0.014501
166	5	0.966668	KMeans	15	0.221712
167	5	0.966668	MiniBatchKMeans	15	0.191287

168 rows × 5 columns

If I were to take a record with both max values then this is what I would get



```
SearchForMaxSilhouette[
    (np.round(SearchForMaxSilhouette['silhouette_score'],decimals=9)==
                      np.round(max(SearchForMaxSilhouette['silhouette_score']),decimals=9))
    &
    (np.round(SearchForMaxSilhouette['explained_variance_first2dims'],decimals=9)==
                      np.round(max(SearchForMaxSilhouette['explained_variance_first2dims']),decimals=9))
]
```



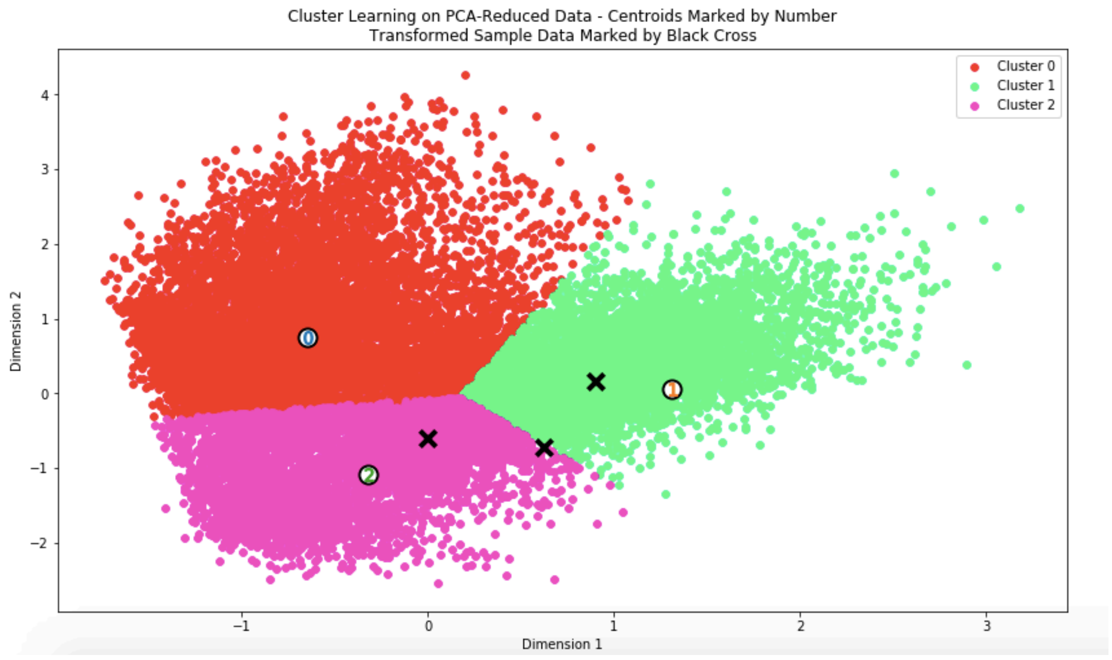
	pca_dimensions	explained_variance_first2dims	Algorithm_name	numberof_cluster	silhouette_score
4	2	0.966668	KMeans	3	0.427982

The optimal number of PCA dimensions are 2 and 3 cluster for Kmeans algorithm has maximum silhouette_score.

Results

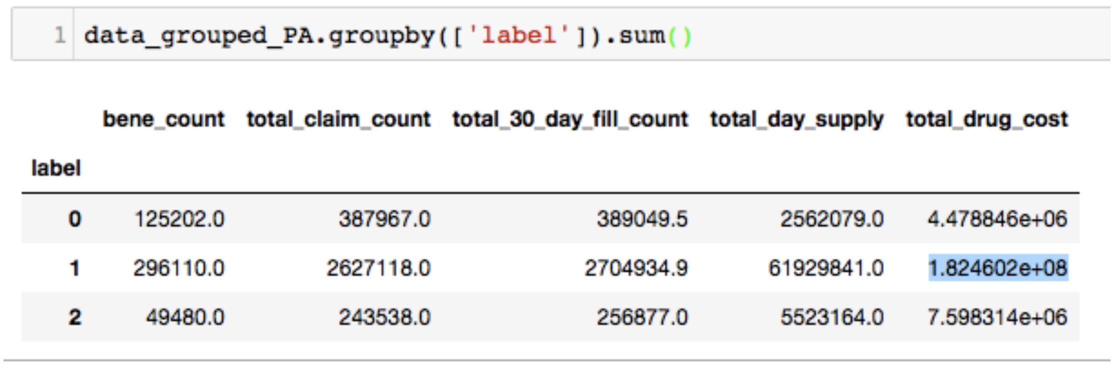
Model Evaluation and Validation

To evaluate whether this model is robust enough let us try to predict the labels for another state, let us PA. Let us use the same model that we built and use it on the PA data to get the labels. Now in this case also I would expect 3 clusters where in 1 cluster should stand out with respect to aggregated total_drug_cost. First let us see how the cluster look like, please note that I have taken 3 samples two of which have been "Charged"(not sure whether they have been prosecuted). One has not been charged. So with the samples and the clustered data here is how the clusters look



Please note that the cluster centers are from the model where we fit AL data, but still the centers are pretty close except for cluster 1.

Now let us see which cluster has the highest aggregated total_drug_cost, I would expect it to be cluster 1, if you notice cluster 1 has the highest amount



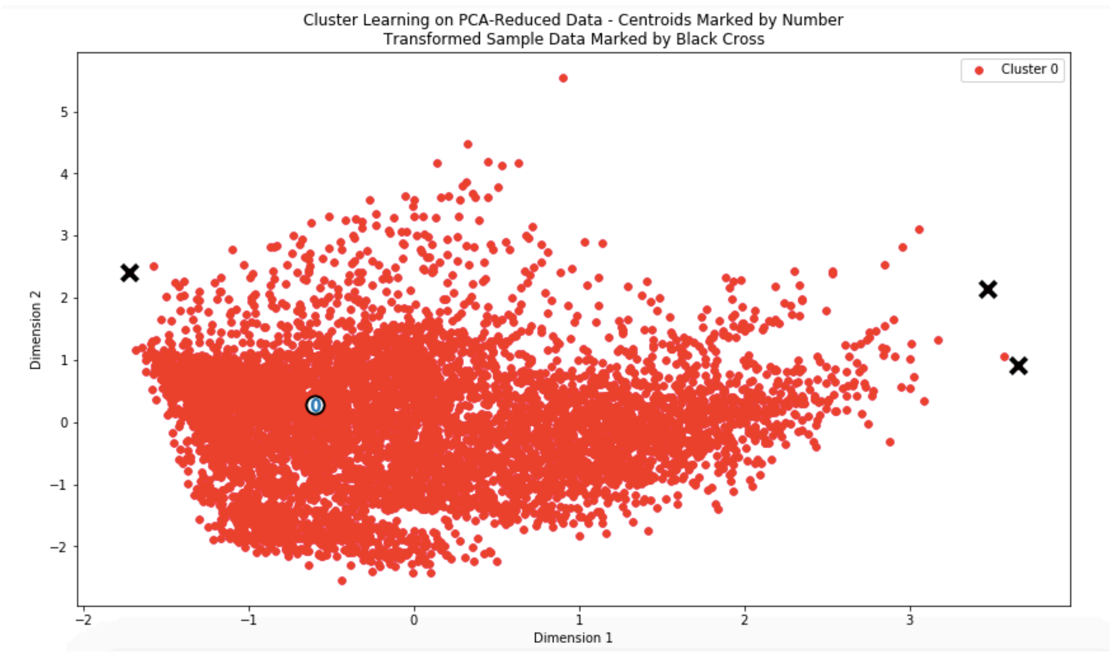
```
data_grouped_PA.groupby(['label']).sum()
```

label	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
0	125202.0	387967.0	389049.5	2562079.0	4.478846e+06
1	296110.0	2627118.0	2704934.9	61929841.0	1.824602e+08
2	49480.0	243538.0	256877.0	5523164.0	7.598314e+06

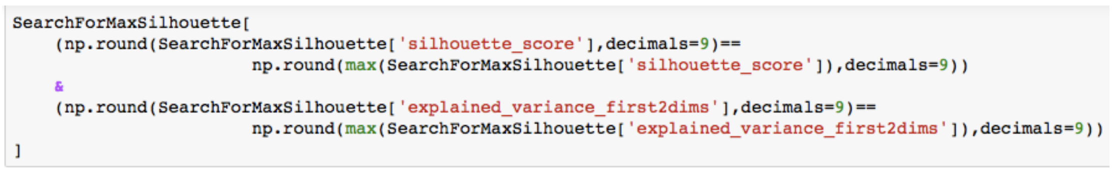
So I would conclude that the model is robust enough for us to consider the labels and prepare a classification model

Justification

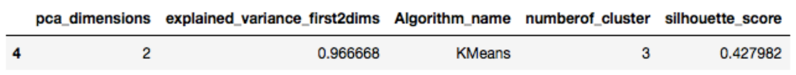
If we recollect we have a benchmark model. Now if we look at the benchmark model (Mean Shift) all of the samples have been clustered into one blob.



Now let us see how the clustering is for the kmeans model with the following parameters

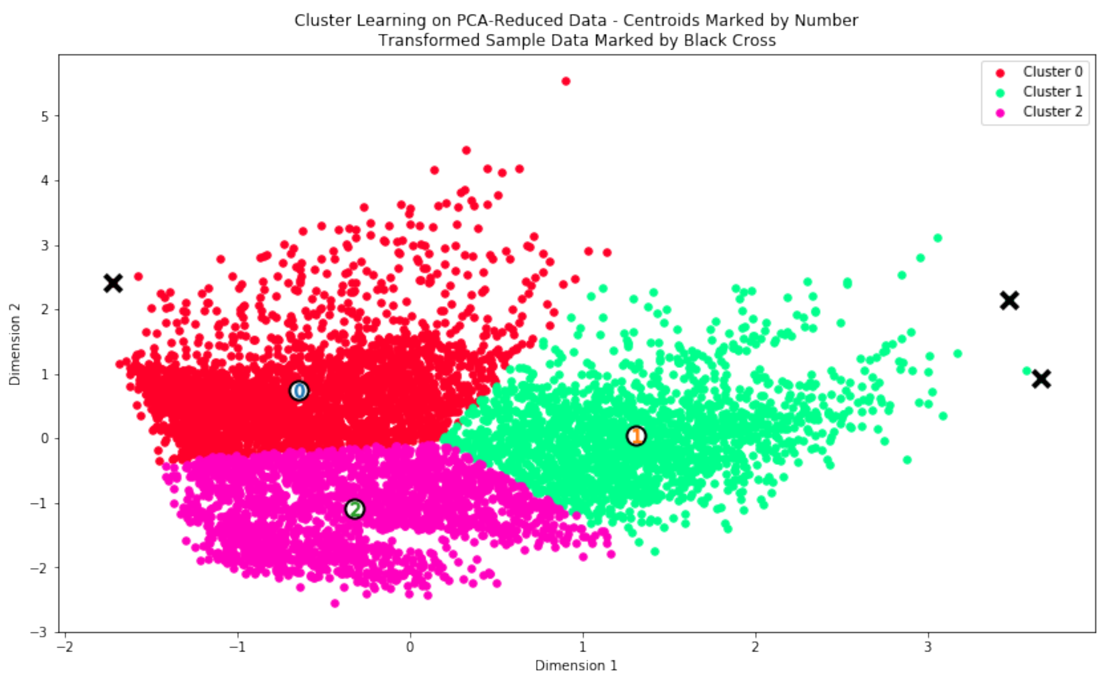


```
SearchForMaxSilhouette[
    (np.round(SearchForMaxSilhouette['silhouette_score'],decimals=9)==
                      np.round(max(SearchForMaxSilhouette['silhouette_score']),decimals=9))
    &
    (np.round(SearchForMaxSilhouette['explained_variance_first2dims'],decimals=9)==
                      np.round(max(SearchForMaxSilhouette['explained_variance_first2dims']),decimals=9))
]
```

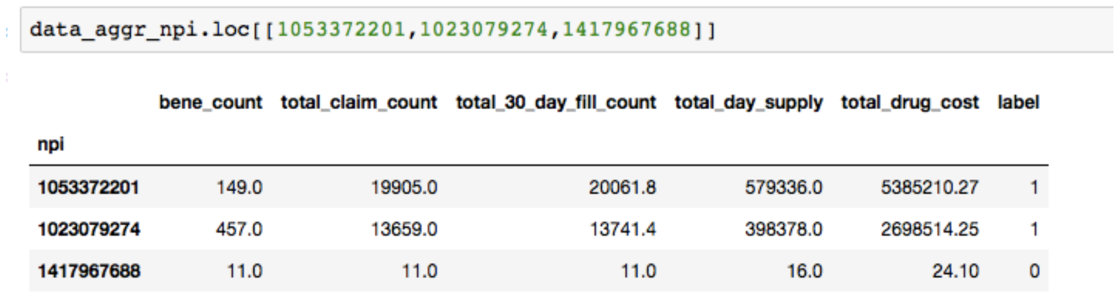


	pca_dimensions	explained_variance_first2dims	Algorithm_name	numberof_cluster	silhouette_score
4	2	0.966668	KMeans	3	0.427982

So here is how the clustering looks like



We could not calculate the score for benchmark model because we have only 1 label, but we were able to get a score which is far better than nothing we compared to the base model. Also please do recollect I was mentioning of three patterns of provider behavior, low, mid and high level of prescription of opioids. Now the above clustering fits my thought process. Also if we recollect I have taken 2 fraudulent providers as, recollect their NPI are 1053372201,1023079274 and in addition to that we have taken an NPI who has the least total drug cost that is 1417967688. Those 2 (1053372201,1023079274) fraudulent fall under cluster 1 and the last npi-1417967688 falls under cluster 0.



```
data_aggr_npi.loc[[1053372201,1023079274,1417967688]]
```

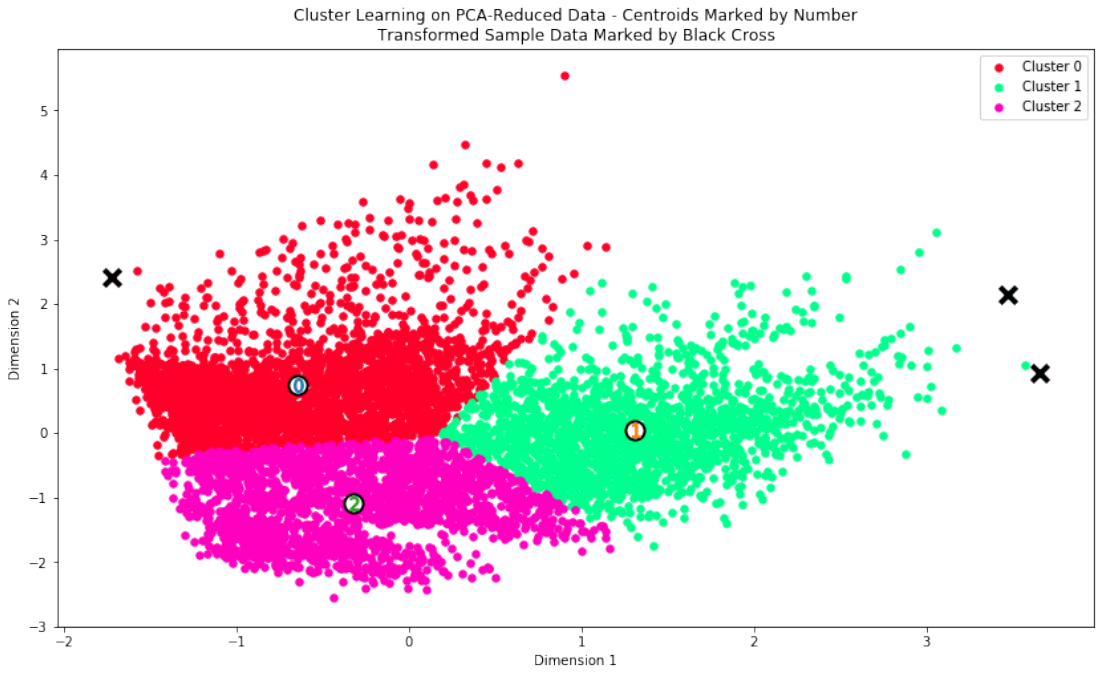
npi	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost	label
1053372201	149.0	19905.0	20061.8	579336.0	5385210.27	1
1023079274	457.0	13659.0	13741.4	398378.0	2698514.25	1
1417967688	11.0	11.0	11.0	16.0	24.10	0

I will confidently tell that we have better model, which is robust and far better than the benchmark model.

Conclusion

Free-Form Visualization

When clustering performed using the above parameters, here is what we get for AL data



```
print len(preds[preds==0])
print len(preds[preds==1])
print len(preds[preds==2])
```

```
2812
1892
2038
```

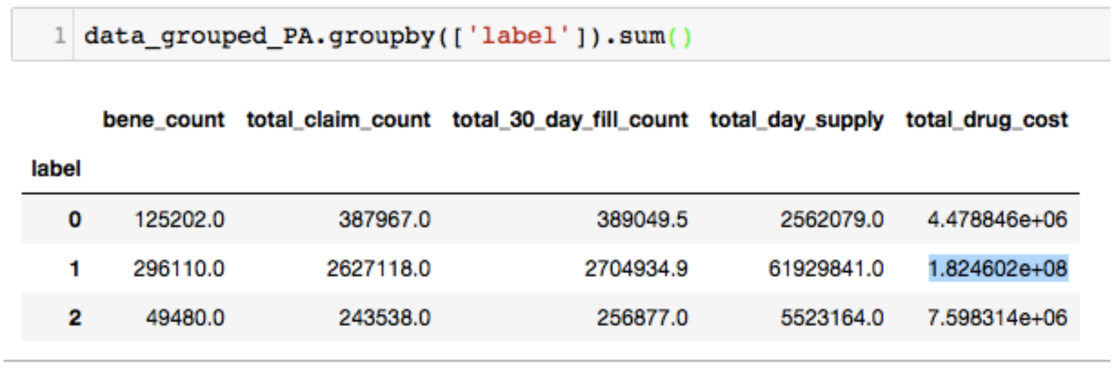
Now if we notice we will see that cluster number 1 is of smaller size, so I would pick that one as my investigation list. The other way of doing this is to get the aggregate total_drug_cost and aggregate other columns and see which one is having the highest per NPI in the cluster and take that cluster for further investigation.

We shall perform pca inverse transform and then apply np.exp to get the values, now if we notice these values are not same as the original values but closer to them original values. This difference is what the pca could not explain in its dimensions. In either case if we were to take the sum of total_drug_cost we will notice that cluster 1 has higher total_drug_cost.

	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
label					
0	37074.0	190779.0	191032.7	982166.0	1480624.12
1	85962.0	1846135.0	1894610.3	46419323.0	87488960.36
2	16061.0	148018.0	156735.2	3310452.0	2175316.59

Looking at the data one can notice that total_drug_cost for cluster 1 is huge when compared to other clusters. I will pick cluster 1 for further investigation.

When we go back and look at the PA data we will notice that 1 cluster has it aggregate total_drug_cost is way huge than the other two cluster.

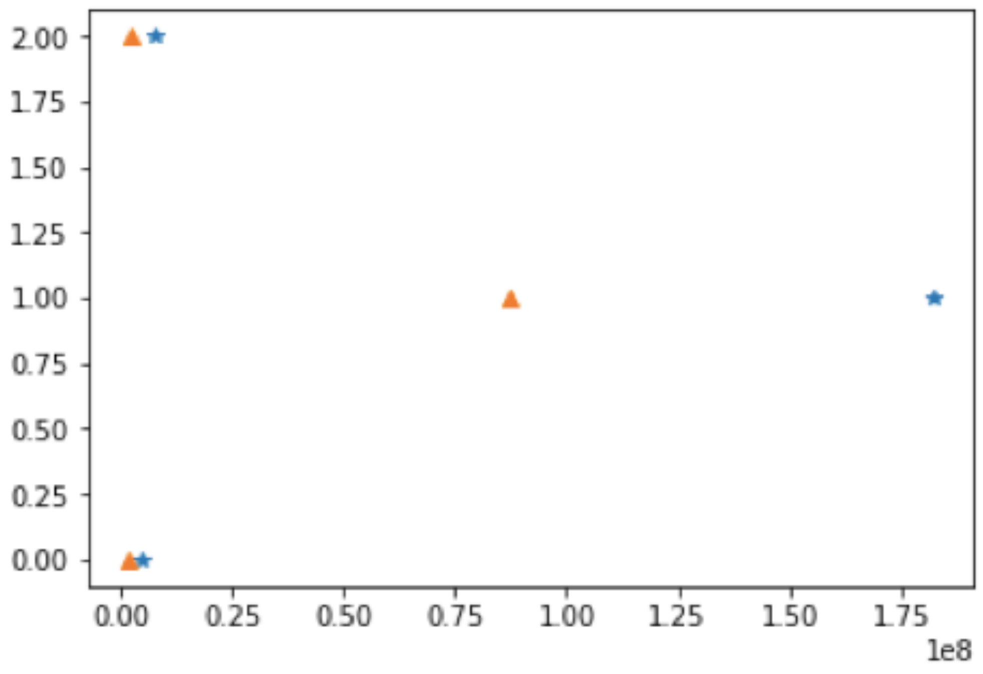


```
data_grouped_PA.groupby(['label']).sum()
```

label	bene_count	total_claim_count	total_30_day_fill_count	total_day_supply	total_drug_cost
0	125202.0	387967.0	389049.5	2562079.0	4.478846e+06
1	296110.0	2627118.0	2704934.9	61929841.0	1.824602e+08
2	49480.0	243538.0	256877.0	5523164.0	7.598314e+06

Now providers do not prescribe all the opioids at once, they bill it at different times and different drugs. Grouping them by NPI and taking the sum provides us a clear picture of the providers behavior and then when grouping by labels and taking the sum of total_drug_cost provides us an idea of which cluster has to be selected for investigation. If we look at the following plot you will notice the same

Aggregated total_drug_cost for AL and PA Clusters
X-axis is the Aggregated total_drug_cost and Y-axis is the cluster numbers
Orange is AL data and Blue is PA data. You will notices that for cluster 1 both AL and PA have high total_drug_cost.



Reflection

When I reflect back the most pain staking activity as part of this project was documentation. I will have to say that out just so that I am relived of my frustration ☺.

From this project one important thing that I learnt is you need to define what problem you want solve, you need to know the domain and most importantly understand your data. I started with a grand idea of identifying the exact provider who is fraudulent then narrowed down to opioid claims then again since hardware could not handle I narrowed down further to individual states. Once I looked at the data noticed that it is possible to find out the exact provider list, so changed the problem statement to find the list of probable fraudulent providers.

Then comes the next difficult part was data clean up, it was pain staking activity and my lack of proficiency in python was clearly shown. The interesting part was the thought "oh it is very easy to build a benchmark model"; it is not from my experience. Your benchmark model should not be stupid one or an intelligent one, there are lots of help to build a good model, but when it comes to benchmark model you have little help.

Overall the model turned out to be as per my expectations. My expectations such as having 3 clusters and then finding at least few of the providers provider in the fraudulent cluster are also present in the OIG list, which I have detailed in Improvement section.

Improvement

We are not done yet. We have the exclusion list that we got from OIG exclusion list, list of providers identified as fraudulent. Now please keep in mind that this exclusion list is for all types of frauds/misdiagnosis/malpractices not just Opioid over prescription. Goal is identify the list of providers who shall be further investigated. The list of providers should consist all of the providers from the list for which ever state we are doing clustering.

https://oig.hhs.gov/exclusions/exclusions_list.asp
https://oig.hhs.gov/exclusions/downloadables/UPDATED.csv

When we bump the data against the exclusion list we will notice that there are total 18 providers from the exclusion list present in our data. Out of 18, 12 have been labeled in cluster 1 and 6 in other clusters. So for the cluster 1 we have total 1892 providers for investigation, out of which in real world if we were to investigate we will end up finding 12 providers as fraudulent. 12 out of 1892 is kind of small number to investigate 1892 providers and find these 12 is tedious. Please do note that rather than investigation ~10000 providers (https://www.aamc.org/download/447144/data/alabamaprofile.pdf
) in the state of Alabama it is better that we narrow down the scope and look into 1892 providers. If we have more features we can use those features to further

narrow down the list, the NPI exclusion list is not a good source of data, a quick look at the data shows that lot of values are blank.

We can use this newly labeled data an input to classification problem. Use the Classification model to predict the labels for rest of the opioid claims data. We can definitely add more features such as income level to further narrow down the investigation list. We will have to search for such data and combine it with current CMS data so that we can even get a better clustering, and subsequent classification model.