Cornell University

Clustering Methods and Their Comparison on Medicare Claims Data with *`Trinity`*

Brian Won (bw436), Chuyi Ma (cm966), Daisy Cho (bsc223), Ruihan Zhang (rz356)

Project Advisor: Dr. Martin T. Wells

Cornell University

Department of Computing and Information Science

May 11, 2020

Abstract

This study provides a brief analysis on healthcare insurance patients from the United States and their clinical and demographic features with obesity. With traditional cluster analysis methods (K-means and others) and non-traditional methods (Latent Class Analysis), patients are assigned into various subgroups with labels to describe them. Results are analyzed in a statistical way for each method and between methods, with business proposals made for pharmaceutical and biotech industries.

1 Introduction

1.1 Project Summary

Obesity patients with various demographic features often experience multiple chronic conditions and consider different medical treatment. To better target marketing campaigns, understanding patient populations and identifying their subgroups are important for pharmaceutical and biotech industries.

With the help of unsupervised machine learning methods, this analysis leverages healthcare claims data from obesity patients, and applies clustering methods to analyze their subgroups. More specifically, this analysis aims at using traditional cluster analysis methods including K-means, K-prototypes, Agglomerative Hierarchical Clustering, Density Based Clustering of Applications with Noise (DBSCAN), Ordering Points to Identify the Clustering Structure (OPTICS), Hierarchical Density Based Clustering of Applications with Noise (HDBSCAN) and non-traditional method including Latent Class Analysis (LCA), to assign obesity patients into subgroups, label main features within each subgroup and clarify usages, optimal situations, advantages and results accuracy for each method.

1.2 Analysis Process

Features of obesity patients are quantified into 120 dimensions, with both continuous variables and categorical variables. To build statistical models among such features, stages are considered:

Stage 1 For the first stage of analysis, patient data is processed to eliminate missing values and convert data types into suitable ones. With the help of package 'Classification And REgression Training (Caret)' from R, patient observations who have missing values in one or more features are eliminated. Besides, dummy variables for part of the features are created to convert them into factors.

Stage 2 For the second stage of analysis, important features are selected from 120 original features with the aim of clearing up variable correlations and reducing data complexity. Using Random Forest, 30 variables are considered to be sufficient for identifying patient subgroups. As a result, these features are later concerned as indicators for separating patient data into various groups.

Stage 3 For the third stage of analysis, clustering methods are applied on processed patient data from stage 1, with selected variables from stage 2. The optimal groups numbers are estimated using Elbow Method, within-group-distance, Akaike information criterion (AIC), Bayesian information criterion (BIC), entropy values or their combination. Descriptions for each class are generated from the results of each method. Comparisons between methods are made based on testing criterions and subgroups results. Analyses are conducted using Python and R.

2 Dataset Description and Pre-processing

2.1 Data Source

The patient-level data used in this project was provided by Trinity. The original dataset contains information of all patients who were enrolled in the Medicare program in 2018. Statisticians at Trinity randomly selected 5% of all entries with comprehensive data. After excluding Part C coverage, they generated a random sample of around 10,000 patients who had obesity concerns. The final version of the initial data for analysis contains 10,705 data entries and 120 variables. Among the 120 variables, there are ten categories in total, which are General Information (6), ESRD/Disability/etc. (2), Obesity Conditions (20), Comorbidities (50), Tests (4), Hospital/Physician Office Visits (5), Procedures (9), Medical Devices (10), Physician Specialty (10), and Treatments (4). Meanwhile, they have three kinds of data types, including Character (7), Number (6), and Binary 1/0 (107).

2.2 Data Summary

The 10,705 patients come from 55 different places, 50 of which are states within the United States, and the remaining 5 are other places around the world. Within the United States, around 52.57% of the patients come from the South, 23.66% from the Midwest, 18.40% from the Northeast, and 15.22% from the West. 54.89% of patients are females and 45.11% are males. The age of patients ranges from 19 to 98, with a median at 70. White (84.60%) and Black (10.30%) are the two major races that build up most of the data. 13.22% of all patients have the concern of Overweight, 79.16% have Obesity, and 28.45% have Morbid Obesity. Meanwhile, 21.42% have more than one concern. People are enrolled in the Medicare program for different reasons. There are four original reasons for Entitlement, which are Old Age and Survivors Insurance (69.01%), Disability Insurance (30.05%), ESRD (0.49%), and Both DIB and ESRD (0.46%). The categories for current reasons stay the same, but the composition has significant changes as many patients shifted the reasons to Old Age and Survivors Insurance (81.42%) from Disability Insurance (18.18%), ESRD (0.27%), and Both DIB and ESRD (0.13%).

2.3 Principal Component Analysis

Methodology

Principal Component Analysis, or PCA, is considered one of the most effective tools in exploratory data analysis. As the basic form of eigenvector-based multivariate analyses, it explains the characteristics of a set of data by exploring the variance using singular value decomposition (SVD)[**?**]. When PCA is applied, an orthogonal transformation is used to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated vectors These vectors are so-called principal components that form an orthogonal basis. The first principal component is the vector that maximizes the variance, while the succeeding component has the highest variance possible under the constraint that it is orthogonal to the preceding components. In the case of this project, PCA might be helpful in reducing the dimension of data because it can provide information on the importance and features of variables on a lower-dimensional level[**?**].

Procedure

The foremost thing to do for PCA is filtering the data. To do so, three kinds of columns should be deleted for simplicity: first, redundant columns that have no relationship with obesity (1 column); second, columns with constant values because they do not contribute to the variance of the data (21 columns); third, columns where over 10% of the values are marked NAs because the value of NA is not allowed in PCA (4 columns). After columns are cleaned, rows that still have NAs need to be cleaned (395 rows). In the last step of data preparation, all non-numeric categorical variables are converted into numeric factors. The cleaned data includes 10,310 data entries and 94 variables. Then, the R function prcomp() in the “stats” package is applied for obtaining the results[**?**].

Results

To illustrate the result of PCA, the distribution of observations on the first two principal components is visualized in a 2-dimensional space (see Figure 1). In the graph below, no clear patterns of clusters can be observed, meaning that the first two principal components alone are insufficient in capturing the features of variables.

Meanwhile, a factor map based on squared loadings (cos2) is drawn to display the importance of each variable (see Figure 2). The x-axis indicates how much a variable contributes to the first principal component while the y-axis indicates the second principal component. A darker color with a longer line means that the variable contributes more to the first two principal components. The figure indicates that the five variables, which are Age, Current Reason for Entitlement, Original Reason for Entitlement, PO, and Morbid Obesity, are relatively more important than others. Given the fact that there is no clear pattern on the 2-D space and that too few variables are selected according to the factor map, the first two principal components strongly underrate the contribution of most variables. Therefore, more principal components are needed.
Then, a scree plot indicating the proportions of variance explained by all 93 principal components is presented in a decreasing order (see Figure 3). On the graph, there seems to have a slowly decreasing pattern after the tenth column, meaning that most principal components have almost the same importance. By calculation, if all principal components are applied, 90.35112% of the variance can be explained. However, as the target is 80%, at least 78 principal components are needed, meaning that most principal components are needed. Therefore, PCA fails to capture the main characteristics of the variables, not to mention the reduction of dimensions. In conclusion, PCA is not a suitable strategy to reduce the dimension of data in the case of this project.

2.4 Pre-processing

To reduce data complexity, some features are considered to be redundant and are deleted before any manipulation. Patient ID (1) is removed; all BMI values (17) are removed, as specific indicators including overweight, morbid obesity and obesity are enough to describe physical conditions of the

patients; States (1) is removed as Region is sufficient for patients' demographics information.
For the left 101 variables, package CARET, a multi-usage package for classification and regression training is used to do data pre-processing. Continuous variables are transformed with centralization and scaling to reduce model bias; categorical variables are transformed with one-hot encoding. Variables with only one level are considered as constant and do not make any contribution to the model, as a result, these variables are removed as well.
80 variables are left after data pre-processing, which is to be selected in the next section.

2.5 Feature Selection

randomVarImpsRF Package

Random Forest is used to select the most important features, with Gini index to determine the final results. However, before feature selection, randomVarImpsRF package is used to determine the number of variables.
randomVarImpsRF is a R package to randomly permute labels into a data set and calculate the importance of each variable, with an output of an importance matrix to indicate variables importance and number of permutations. Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples.
Figure 4 shows the percentage of each variable in the decision tree, with the red dashed line representing importance percentage of 0.05 and green dashed line representing importance percentage of 0.03.
Considering 0.03 to be the threshold for feature importance percentage, variables number should be around 30, and any variable below that threshold might either contain too much noise or is highly related to others. As a result, the variable number is considered to be 30.

randomForest Package

Random Forest for unsupervised machine learning generates 'fake' data using distributions of real data. 'Real' and 'fake' data are then combined and labeled again as 'real' or 'fake', the process of which converts an unsupervised machine learning problem into a supervised machine learning problem.
To determine the selection subset from all 80 variables, mean decrease in Gini Impurity (GI) is applied to calculate the probability for each observation when it is misclassified. In this analysis, patients with obesity are considered as 'real', with non-obese patients as 'fake'. Figure 5 shows top 30 variables with their mean decrease in GI.
Figure 6 shows the selected variables and their labels (variable age is converted into a 4-level factor by 4-quantile). 30 features are labeled into 8 subgroups, deciding by their meanings. The first, also the largest subgroup is demographics, including features that can represent patients' age, living area, race and gender. The second subgroup shows obesity conditions. The third subgroup contains comorbidities, with kidney disease, hypothyroidism, fatigue, glucose abnormality and sleep apnea selected from all 50 comorbidities. The rest 5 subgroups indicate hospital visits, analysis procedure, physician specialty, physician treatment and reason for entitlement respectively.

3 Model Design and Interpretation

Picking the "optimal" number of clusters (for applicable methods)

For many clustering methods, including those that are applied for this project, the number of clusters to generate must be specified. However, there is no exact way to do so; this is one of the weakest aspects of cluster analysis (citation). There are different methods for picking the "optimal" number of clusters, most of which involve a measure of (dis)similarity (or "cost") between clusters. The most commonly used method in practice is the Elbow Method which is done as follows:

1. Run the clustering method for a range of numbers of clusters to generate (usually starting from

1); for 1, 2, 3, etc.

2. Plot cost against each number of generated clusters.

3. Identify the point(s) where the cost significantly levels off.

4. The number(s) of clusters for the point(s) is/are “optimal”.

As shown in Figure 7, the optimal numbers of clusters to generate are 3 and 5. The method gets its name because the line looks like an arm, of which the elbow is to be identified (where the “arm” bends). If multiple optimal numbers are identified, all are to be investigated.

3.1 K-Means Method

Methodology

K-Means is one of the most commonly used clustering algorithms. It revolves around distances between points and cluster centers; logically, points assigned to a cluster should be close in distance to that cluster. The K-Means algorithm can be broken down as follows:

1. Select the number of clusters to generate and randomly initialize their centers.

2. Each data point is assigned to the cluster whose center the point is closest to.

3. Based on the assigned points, recompute cluster centers by taking the *mean* of all the points in each cluster.

4. Repeat Steps 2 and 3 for a specified number of iterations or until cluster centers don’t change much.

The pros and cons of K-Means are as follows:

- Pros:
  - Fast computation

  - Robust to outliers
  - Ability to specify distance function
  - Robust to irrelevant variables

- Cons:
  - Must specify number of clusters to generate
  - Results depend on initialized centers, which are generated randomly

Procedure

The input data for K-Means consists of the 30 most important features that are extracted using Random Forest as explained in the Pre-Processing section. Euclidean distance is set as the distance metric. The following graph, as per the Elbow Method, is used to determine the number of clusters to investigate. The cost, in this case, total within-cluster sum of squares (WCSS), is plotted against number of clusters generated in Figure 7.
From Figure 8, potential optimal numbers of clusters to generate are 4 and 7, as the slope of the plot levels off at both points.

Result

Figure 9 indicates that K-Means is able to discern the severity of obesity cases; cluster labels (4,3,2,1) are assigned with respect to severity (i.e., 4 = most severe, 1 = least severe). Percentages under % of Data represent the percentage of the sample population that is assigned to each cluster. Percentages under Morbidly Obese, Kidney Disease, Hypothyroidism, Fatigue, and Sleep Apnea represent the percentage of a given cluster that is clinically diagnosed with each condition. The percentage columns for Table 3 follow the same interpretation. Cluster 3 can be interpreted as an outlying group, as it contains only 1.7% of the sample population, while having the highest diagnosis rate for all the shown medical conditions and highest average number of medical visits.

Clusters 6 and 7 seem to be subgroups of cluster 4 from Figure 10, where cluster 6 consists of cases with higher (potentially outlying) counts of medical visits and cluster 7 has the higher rate of kidney disease diagnoses. Clusters 4 and 5 seem to be subgroups of cluster 3 from Figure 10, with cluster 4 having higher rates of kidney disease and fatigue and cluster 5 having higher rates of morbid obesity and sleep apnea. Clusters 1, 2, and 3 seem to be subgroups of clusters 1 and 2 from Table 3. For the four comorbidities (kidney disease, hypothyroidism, fatigue, and sleep apnea), cluster 3 has the highest rates of the three clusters, with a hypothyroidism rate comparable to that of cluster 6. Cluster 1 represents the healthiest group of the sample population.

3.2 K-Prototypes Method

Methodology

The problem with K-Means is that it isn't ideal for categorical data. While K-Means can be applied to categorical data (and in our case, binary data), calculating Euclidean distance between points, most of whose entries are 0 or 1, is not intuitively sound. When using Euclidean distance, the binary-variable entries of the cluster centers that are calculated with K-Means will take on values between 0 and 1. The K-Modes algorithm is preferable for categorical data because it calculates the Hamming distance, or number entries that disagree, between a point and a cluster center. In addition, cluster centers are calculated by taking the *mode* of each (categorical) entry based on the points assigned to respective clusters. However, K-Modes can only be applied to categorical data. K-Prototypes is an algorithm that is a combination of K-Means and K-Modes, which enables it to be applied to mixed data. The algorithm's structure follows the same conceptual process as K-Means and K-Modes: initialize centers, then repeatedly assign points and update cluster centers until the centers don't change significantly. K-Prototypes calculates distance as follows:

$$Distance = (Euclidean distance of continuous var.) + W(Hamming distance of categorical var.)$$

Where W is a constant; it can be interpreted as how much weight to give to the categorical portion of the data. If W is set to 0, then running the algorithm would equate to using K-Means on the continuous variables. For initializing centers, two common methods are Huang and Cao. Huang focuses on the frequencies of categorical entries to generate centers, while Cao utilizes density of the data points.

Procedure

The input data for K-Prototypes consists of the 30 most important features that are extracted using Random Forest. In addition to determining the number of clusters to generate for investigation, an initialization method and a value for *W* need to be specified. While the algorithm calculates a "default" value for *W*, the provided default value is not guaranteed to be the best (Huang 1997). The Elbow Method is used to determine the three parameters. The cost, in this case, the sum of distances of all points from their respective centers, is plotted against the number of clusters generated. Figure 11 shows that there is no clear difference in minimizing cost between the two methods. The Huang Method is selected for initialization because the majority of the 30 variables used are binary. Each plot from Figure 12 is with respect to a different value for *W*. It seems that setting $W = 2$ and generating 4 clusters is optimal because that's where the clearest leveling-off of *Total Distance* takes place.

Result

From Figure 13, it seems that two main groups are apparent; those who are morbidly obese (4,3) and those who are just obese (2,1). Then those two groups can be further broken down into subgroups with respect to higher/lower rates of comorbidities. The difference between clusters 3 and 4 is more apparent than the difference between clusters 1 and 2. While clusters 3 and 4 both have higher rates of morbidly obese cases, cluster 4 clearly is the unhealthier group. One point worth noting is that like K-Means, K-Prototypes identified an outlying cluster (4), but in contrast, that cluster captured a greater portion of cases with a kidney disease diagnosis than the outlying cluster of K-Means did. Cluster 4 of K-Prototypes contains 6.0% of the data and has a kidney disease rate of 82.0% while cluster 3 of K-Means contains 1.7% of the data and has a kidney disease rate of 84.6%.

4 Agglomerative Hierarchical Clustering Method

Methodology

Agglomerative Hierarchical Clustering (AHC) operates as follows:

1. Specify metrics for calculating distance between points and linkage between clusters.
2. Treat each data point as a cluster.
3. For each iteration, combine the two closest clusters (with respect to linkage).
4. Repeat Step 3 until one cluster containing all data points is formed.
5. Use a dendrogram to determine the best number of clusters.

The pros and cons are as follows:

- Pros:
  - Clear portrayal of groupings (via dendrogram)
  - Not sensitive to choice of distance and linkage metrics
- Cons:
  - Propagation of errors $\rightarrow$ No way to undo or mitigate an error made during an iteration
  - Not robust to outliers
  - Slow computation

Procedure

For this algorithm, the input data consists of the 30 most important features extracted using Random Forest. For distance between points, Euclidean distance is used. For linkage between clusters, Ward's Method is used because it emphasizes minimizing within-cluster variance.
The method for determining the "optimal" number of clusters is as follows:

1. Find the largest vertical distance that doesn't intersect other clusters.

2. Draw a horizontal line through that vertical distance.

3. "Optimal" number of clusters is the number of vertical lines that intersect the horizontal line.

Using that method on the dendrogram shown in Figure 14 indicates that the "best" number of clusters is 2. However, generating 2 clusters yields groupings that are too broad for meaningful interpretation. The next "best" number of clusters to generate, as indicated by the solid horizontal line in Figure 10, is 4.

Result

From Figure 15, it seems that AHC identifies cluster 4 as an outlying cluster. Clusters 3 and 4 seem to consist of more severe cases compared to clusters 1 and 2. In contrast to K-Prototypes, AHC results in less clear differentiation between the two Mild clusters (1,2).

4.1 Density Based Clustering of Applications with Noise (DBSCAN) Method

Methodology

DBSCAN is based on the intuitive notion of "clusters" and "noise". In order to run DBSCAN two important parameters are required: epsilon ("eps") and minimum points ("minPts"). The parameter **eps** defines the radius of the neighborhood around a core point represented as x below and is called the epsilon-neighborhood of **x**. **minPts** is the minimum number of neighbors within the radius of the neighborhood. **y** represents the border point in the epsilon-neighborhood of **x** and **z** is noise point. There are both pros and cons to this method:

- Pros:
  - Does not require a pre-set number of clusters
  - Identifies outliers as noises
- Cons:
  - Doesn't perform as well as other methods when the clusters are of varying density $\rightarrow$ The epsilon neighborhood distance threshold and minPts vary from cluster to cluster
  - Challenging to estimate eps for high-dimensional data

Procedure

The dataset used for this analysis is the subset of the raw data using the 30 most important variables calculated. For the DBSCAN* function, the epsilon value is determined by the kNNdistplot() function in the dbscan package. First, the **minPts** value is determined by the number of variables + 1, so in this case the value for **minPts** is 31. In order to plot the k-nearest neighbor distances, the **k** value is determined by the user and corresponds to **minPts**, therefore the **k** value is 31. The optimal **eps** value is found at the "knee" of the plot, represented as the dotted line in Figure 17, which is determined to be at 4.15.

Result

The results show the DBSCAN cannot be a good representation of clusters for the current dataset. Although there are a different variety of patients, the DBSCAN analysis shows that there is one cluster which does not seem to tell any information about the data. The data that is plotted has 30 dimensions, however, the graph, produced by *fviz cluster* in the "factoextra" package, has done a Principal Components Analysis and projected the data onto the first two principal components to make a two dimensional plot. Those should be the two dimensions that show the most variation in the data. The 14.8% means that the first principal component accounts for 14.8% of the variation. The second principal component accounts for 10.2% of the variation. So together they account for 25% of the variation. The black points are the outliers. In conclusion, DBSCAN is not a good package to use to run density based clustering analysis on this data.

4.2 Ordering Points to Identify the Clustering Structure (OPTICS) Method

Methodology

Ordering Points to Identify the Clustering Structure (OPTICS), is an algorithm for finding density-based clusters in spatial data. Although the method was derived from Density-Based Clustering Analysis (DBSCAN), it can resolve the major weakness of DBSCAN, which is the problem of detecting meaningful clusters in data of varying density. Same as DBSCAN, OPTICS requires 2 parameters: one is eps that describes the maximum distance to consider within a cluster, and the other one is MinPts that sets a constraint for the minimum number of points required to form a cluster. To conduct OPTICS, the concepts of core-distance and reachability-distance are also borrowed from DBSCAN.

$$core{-}dist(p; \epsilon, minPts) = \begin{cases} UNDEFINED if |N_\epsilon(p)| < minPts, and \\ \\ \text{minPts-dist(p) otherwise.} \end{cases}$$

$$reachability{-}dist(p, q; \epsilon, minPts) = \begin{cases} UNDEFINED if |N_\epsilon(p)| < minPts, and \\ \\ \text{max(core-dist(p), d(p,q)) otherwise.} \end{cases}$$

Different from DBSCAN that searches for natural groups in the data, OPTICS algorithm is usually described as an augmented ordering method. It starts with an arbitrary point and expands its neighborhood. Afterwards, it explores other points from lowest to highest core-distance. The order in which the points are explored along with each point's core-disability and reachability-distance becomes the final result of the algorithm.

Procedure

The data used in the OPTICS is the one processed by random forest. To conduct the algorithm in R, the optics() function in the "dbscan" package is required. A value of 200 is selected for so that the core-distance is defined as the distance from the start point to the 199th nearest neighbor. The MinPts is also set to be 200 to span a neighborhood of at least 200 points. Also, the value of xi that identifies clusters is determined as 0.001. The small xi value helps magnify the difference between clusters for data.

Result

The results of OPTICS include 5 clusters and one noise point, shown in the following chart (see Figure 19). To present the results in a more visually friendly way, a dendrogram of reachability distance in an increasing order is plotted (See Figure 20). In the dendrogram, the highest column at right illustrates the only point of noise. For the remaining columns, the first cluster includes areas of red, green, dark blue, light blue, and black (except for the noise point); the second one includes areas of red, green, and dark blue; the third one includes areas of red, green, dark blue, and light blue; the fourth one includes areas of red and green; the fifth includes only the red area.
An interesting thing found in the results of OPTICS algorithm is the existence of overlapping clusters. Although it is not very common in unsupervised learning, clusters may overlap in three conditions[**?**]: there may be unavoidable noise in the data; second, the features of variables may not capture all the necessary information to clearly separate clusters; third, the overlap might be inherent

to the processes that produce and analyze the data. In the case of OPTICS, it might be a problem of algorithms because they do not assume a clear separation of the clusters. Also, the obesity data seems to have highly-correlated variables, adding difficulties for OPTICS to identify clusters. The result of overlapping clusters is not wrong at the theoretical level but may cause problems when it is conducted in the real world. Therefore, the OPTICS method is not suitable for this project.

4.3 Hierarchical Density Based Clustering of Applications with Noise (HDBSCAN) Method

Methodology

HDBSCAN is the hierarchical representation of density based clustering using DBSCAN and then uses a technique to extract flat clustering based on the cluster stability. However, unlike DBSCAN, HDBSCAN does not require calculation of epsilon-neighborhood values. Instead, it only requires the minPts value. It then uses a technique to extract a flat clustering based on the stability of the clusters. In order to do this we first cluster the data using HDBSCAN, then we find the core distance and the epsilon values and create a simplified tree of the clusters.

Procedure

The only requirement to run HDBSCAN is to specify the minPts or the minimum amount of points in each epsilon-neighborhood. As done for the DBSCAN method, 30 important variables are used as a subset of the full dataset in order to smoothly run the analysis so the minPts is specified as 31. After running the HDBSCAN algorithm, plotting the tree resulted in a messy block of data as seen in Figure 21.

In order to simplify the data and show it in clusters, the core distances are calculated using kNNdist with a k value of minPts - 1, which is 30 in this case. Since cutree doesn't see the noise points as 0, a separate function is used to cut the tree into several groups by using the number of clusters specified by HDBSCAN. Then, the epsilon values are calculated and the hierarchical tree is cut and DBSCAN is run for the number of epsilon values.

Result

The resulting tree shows that there are four ‘stable’ or flat clusters reported, the number of groups to cluster data into. Stable clusters mean that if HDBSCAN is ran again with a different random initialization, the same clusters will be resulted again.
Looking at the clusters and the corresponding number of points, HDBSCAN calculates most of the data points as noise points or outliers. So overall HDBSCAN only shows that only 7.92% of the data points, or 848 data points out of a total of 10705, can be explained by this clustering method. Therefore, it should not be considered a good clustering method for this dataset. This can be because the datasets are too variable for this specific algorithm to handle or density based clustering needs to be approached in a different manner such as using different packages or variables.

4.4 Latent Class Analysis Method

Methodology

Latent Class Analysis (LCA), first introduced by Lazarsfeld and Henry in 1968, is a way of formulating latent variables from manifest variables (also called indicator). LCA is a model-based approach, which means it assumes that a mixture of underlying probability distributions generates the data, and a statistical model is postulated for the population from which the data sample is obtained. As variables are considered to be yielded to some distributions, mass function or density function is used to express those distributions. To assign data points to different non-overlapping groups, a posterior membership probability has to be maximized.
Different from traditional clustering methods, LCA could be applied to variables of different scales; also, as posterior membership probability is calculated directly from observed variables, choice for clustering criterions is less arbitrary (compared with Euclidean distance).

Procedure

In R, the poLCA package is used to conduct lca function. Class numbers for the LCA model are assumed between 2 and 8, otherwise there might be problems for business interpretation. To decide the optimal number of classes, four model testing criterions, AIC, BIC, log likelihood and entropy are used, with AIC and BIC showing the penalty value of the model and entropy showing how much the data is separated between classes.
Figure 24 shows values of each criterion, with red line representing AIC, blue line representing BIC, yellow line representing log likelihood and green line representing entropy value. When the number of the classes increases, AIC and BIC values decrease, indicating that the model might have smaller loss of information. For entropy, however, a peak (0.85) occurs when cluster number equals 4, indicating that in the 4-cluster model, classes might be well-separated. Considering the trade-off between information integrity and clustering accuracy, 4-cluster model with the highest entropy is chosen as the final LCA model.

Result

Patients dataset is divided into four parts, each with different features. For instance, clients in each subgroup have various probability of falling into different age groups; their reasons for entitlements are different; and they also show different likelihood for having different comorbidities.

Figure 25 shows part of the result of the 4-clusters LCA model.

Class 1 A total of 21.77% of the data is expected to belong to Class 1 with a probability of 36.37% of falling between 75-year-old and 98-year-old, which might be the oldest group of all patients. This class is labeled as the 'oldest' or 'very severe' class. They show comparatively largest likelihood of being obese (86.12%); have the largest likelihood of having comorbidities of kidney disease (61.68%), fatigue (57.30%), glucose abnormality (58.84%) and recurrent respiratory infection (43.85%); have the largest number of medical visits (63.49%

for PO, 39.60% for OP, 61.34% for IP, 56.58% for Number of ER Visits). They have the largest likelihood to receive a physician specialty of GASTRO (29.09%), NEUROL (27.57%), PT (16.25%), ORTHO (38.90%), ENT (16.56%) and ENDOCRIN (15.89%), or receive treatment of glucocorticoids (46.57%). However, this class also has moderate likelihood of showing comorbidity of Hypothyroidism (35.20%), or having procedure of TSH (66.22%) and Lipid Panel (75.49%), or receiving Hormone treatment (33.73%). The Original Reason for Entitlement for Class 1 is probably OLD AGE AND SURVIVORS INSURANCE (76.22%).

Class 2 Class 2 (26.44%) is characterized as the 'severe' class as they have the second largest probability of having kidney disease (22.17%), fatigue (28.89%), Hypothyroidism (48.42%) and glucose abnormality (44.99%). They have a likelihood of 36.27% of aging between 65-year-old and 70-year-old, with the largest likelihood of receiving Hormone (53.16%) and TSH (99.02%) treatments and OPHTHAL (44.43%) physician specialty, or showing procedure of Lipid Panel (93.07%) among all 4 classes. Class 2 shows moderate likelihood of receiving physician specialty of PT (12.57%), ENT (12.55%), GASTRO (18.09%) and ENDOCRIN (12.93%). They have a small likelihood of receiving a physician specialty of NEUROL (9.80%). The Original Reason for Entitlement for Class 2 is probably OLD AGE AND SURVIVORS INSURANCE (89.29%).

Class 3 Class 3 (19.48%) is characterized as 'youngest' or 'moderate'. Different from other classes, patients in this class are all under 65-yuear-old (100%), with a comparatively larger likelihood of being African-American (21.20%), and their Original Reason for Entitlement is probably DISABILITY INSURANCE (95.82%). They show the largest likelihood of being morbid obese (43.10%) among all 4 classes, with moderate likelihood of showing comorbidities or receiving treatments. However, they also have the smallest likelihood of receiving a physician specialty of PT among classes.

Class 4 Class 4 (32.31%) are characterized as 'mild'. They have a likelihood of 40.07% between

65-year-old and 70-year-old, with the smallest likelihood of having hospital visits (2.68% for PO, 7.37% for OP and 6.17% for Number of ER Visits), or showing comorbidities (16.39% for kidney disease, 4.96% for Hypothyroidism, 8.77% for fatigue and 22.38% for glucocorticoids), or receiving those medical treatment and physician specialty. Their Original Reason for Entitlement is also probably OLD AGE AND SURVIVORS INSURANCE (89.15%).

Apart from the above differences from 4 subgroups, however, gender and region both show no obvious distinguishes, which might indicate they have a much smaller relationship with obesity severity.

5 Results Comparison

5.1 K-Means vs. K-Prototypes vs. AHC

The three clustering methods share the common trait of using distance to assign points to clusters. This seems to be the main reason that all three algorithms, with applying the Elbow Method, agree that generating 4 clusters is optimal. From Figures and , the cluster breakdowns for K-Means and AHC, respectively, the outlying groups for both algorithms seem to be most characterized by their high average counts of medical visits. For K-Means and AHC, their outlying clusters contain 1.7% and 0.9% of the sample population, respectively. The average number of medical visits are 122.8 and 146.9, respectively. In contrast, the outlying group for K-Prototypes contains 6.0% of the sample population and has an average count of medical visits of 73.9, indicating that it doesn't include as many cases with outlying counts of medical visits as the outlying clusters of the other two algorithms. In addition, there is clearer differentiation between the groups produced by K-Prototypes when looking at their diagnosis rates of comorbidities. This can be attributed to K-Prototypes using different distance metrics for categorical and continuous variables. In turn, more investigation should be done into how different values for the weight parameter, W, yield different clustering results.
There is no exact criteria for determining which algorithm is 'best'. However, it's important to

look at the clusters that are produced across the different clustering methods and take note of their interpretability. That requires asking, "How are groups differentiated?" and "Are the differentiations meaningful at all?" For the data that is used for this project, the strongest case can be made for K-Prototypes, given that more emphasis can be placed on diagnosis rates when differentiating between its clusters.

5.2 Distance-Based Clustering (K-Means) vs. LCA

K-means and LCA are considered as typical representatives for traditional and non-traditional unsupervised machine learning methods, respectively. There are similarities between two methods, for instance, patients could be assigned into 4 subgroups, each with an obesity severity label and percentage of being diagnosed or the likelihood of being diagnosed with obesity comorbidities. However, merely by looking at the results of 4-clusters models, obvious differences are shown in two aspects:

Numbers of percentage representing two different things: In K-means, percentages under % of Data represent the percentage of the sample population that is assigned to each cluster, and percentages under comorbidities represent the percentage of a given cluster that is clinically diagnosed with each condition. While in LCA, percentages represent the posterior membership probability that observations are assigned into 4 subgroups or the likelihood of carrying with those features, which in this case represent the probability of showing the obesity comorbidities or receiving physician specialties and treatments. This difference could be viewed as the 'real' population vs the 'estimated' likelihood.

Estimated populations for each subgroup having different numbers: For K-means, patients with obesity severity levels from 1 to 4 consist of 60.5%, 24.1%, 13.6% and 1.7% of whole dataset, while for LCA, the likelihood of being assigned into these subgroups are 32.31%, 19.48%, 26.44% and 21.77%. Estimated population for the mild patients show similarity: percentage for this class is the smallest in both methods. However, populations assigned for each subgroup tend to have a sharper change for K-means, while the changes in possibilities of being assigned into each subgroup are comparatively mild for LCA.

Reasons for such differences might be explained from 2 aspects:

Centroids for each subgroup in K-means are assigned with randomness. As a result, it lacks certain stability. As Krantz, et al. has mentioned, the random assignment of the cores of clusters is problematic to some, and the researchers' determination of natural clusters is problematic to others. Moreover, cluster results may not be robust. Adding cases to an existing data set or using an entirely new data set may yield a cluster solution that is quite different (2009, 297). For instance, the order in which the data is assembled is a huge impact for K-means results, as it might produce different solutions that could fit in some business expectations. Such problems could not be fixed as criteria to judge the suitability of solutions do not exist.

statistical distributions might exist under biological observations. As LCA assumes distributions generate data, it outcomes K-means in the analysis of dataset with some statistical patterns, which is highly possible in demographics data or biological data. So even as K-means and LCA both seek divisions that maximize the between-cluster differences and minimize the within-cluster differences, in K-means this decision might be more arbitrary and subjective, while in LCA, such statistical model allows the comparison to be statistically tested, so that the decision to adopt a particular model is less subjective.

6 Conclusions

In a healthcare-insurance-based sample, 4 classes adequately account for variations in comorbidities, demographics and clinical patterns of 30 medical-related variables. In the process of analyzing, K-means, K-prototypes, hierarchical, OPTICS, DBSCAN and LCA all show their preferences and drawbacks, and there exist some situations where one or more methods outcome the others. However, after deep comparisons between K-means and LCA, a preliminary conclusion could be made that for healthcare dataset, LCA method might be an optimal choice for a more stable result.

Certain suggestions for business could be made according to the analysis above:

1. Clients for industries have no need to focus on patients' gender or bureau region, as they show similar possibility of being female or male, and similar possibility of living in the area of midwest, northeast, south, west or others. Races, however, tend to show some relationship with obesity. African-American patients under 65-year-old, with an original reason for entitlement of disability insurance should be paid attention to, as this class shows a moderate tendency of having obesity comorbidities.

2. When identifying obesity patients, a look into Kidney disease, Hypothyroidism, Fatigue, Glucose abnormality and Sleep apnea might be a quick decision for obese patients and non-obese patients, or an easy indicator to determine the severity of those patients. However, further clinical diagnoses should be conducted on patients to identify more thorough features.

3. Physician specialties and medical treatments for patients from 4 classes tend to have various choices. For patients with the most severe obesity comorbidities, they tend to be more suitable to receive physician specialties of GASTRO, NEUROL, PT, ORTHO, ENT and ENDOCRIN, or receive treatment of glucocorticoids. For the less severe patients, while the physician specialties and treatments above also have positive impacts on them, Hormone, TSH and OPHTHAL might have better results.

This analysis, in a nutshell, presents a meaningful way of understanding multiple features of obesity patients, and might give some useful insights for pharmaceutical and biotech industries.

References

[1] Abdi. H. Williams, L.J. (2010). "Principal component analysis". Wiley Interdisciplinary Reviews: Computational Statistics. 2 (4): 433–459. arXiv:1108.4372. doi:10.1002/wics.101.

[2] Jolliffe, I. T. (2002). Principal Component Analysis, second edition Springer-Verlag. ISBN 978-0-387-95442-4.

[3] Everitt, B., Hothorn, T. (2006). A handbook of statistical analyses using R. .

[4] Adam A, Blockeel H. (2016). “Dealing with overlapping clustering: a constraint-based approach to algorithm selection” .

[5] Cran.r-Project, .

[6] DBSCAN: Density-Based Clustering for Discovering Clusters in Large Datasets with Noise - Unsupervised Machine Learning.” STHDA.

[7] Hahsler, Michael, et al. (2019). “Package ‘Dbscan.’” Cran.r-Project(2019,10). .

[8] Halabisky Brian. Determining The Right Number Of Clusters.

[9] “Hdbscan.” RDocumentation. .

[10] Huang Zhexue. (1997) Clustering Large Data Sets With Mixed Numeric and Categorical Values. CSIRO.

[11] McInnes, Leland, et al. (2016). How HDBSCAN works. Read the Docs. .

[12] SCHREIBER, J. B., PEKARIK, A. J. (2014). Technical Note: Using Latent Class Analysis versus K-means orHierarchical Clustering to Understand Museum Visitors. https://doi.org/10.1111/cura.12050.

[13] Cao, F., Liang, J. Bai, L. (2009). A new initialization method for categorical data clustering, Expert Systems with Applications 36(7), 10223-10228.

Appendix I: Figures

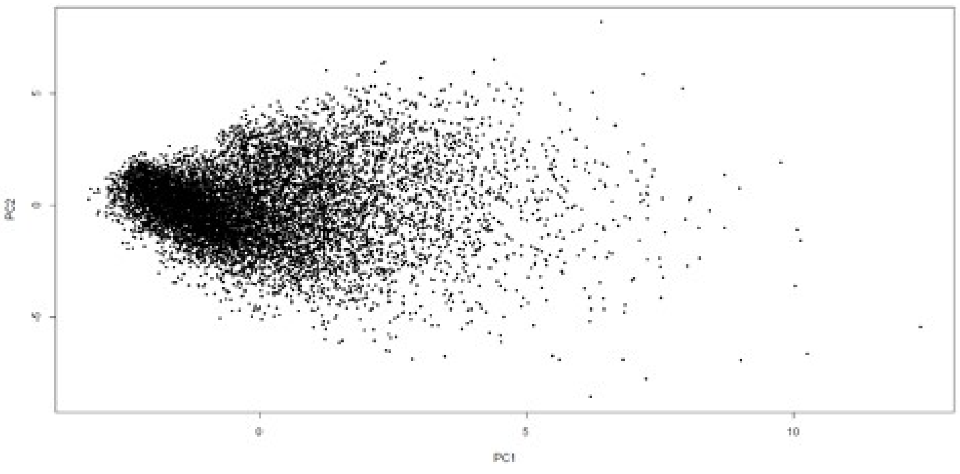


Figure 1: Graph of Data Distribution on First Two Principal Components

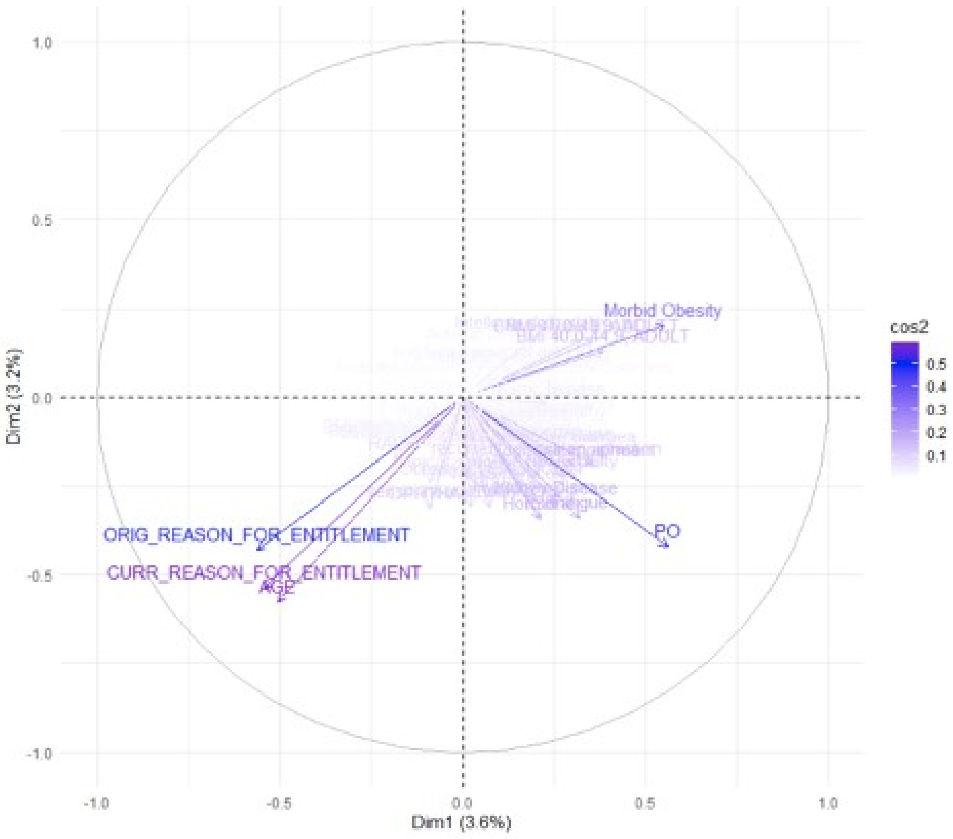


Figure 2: Factor Map of Variables on First Two Principal Components

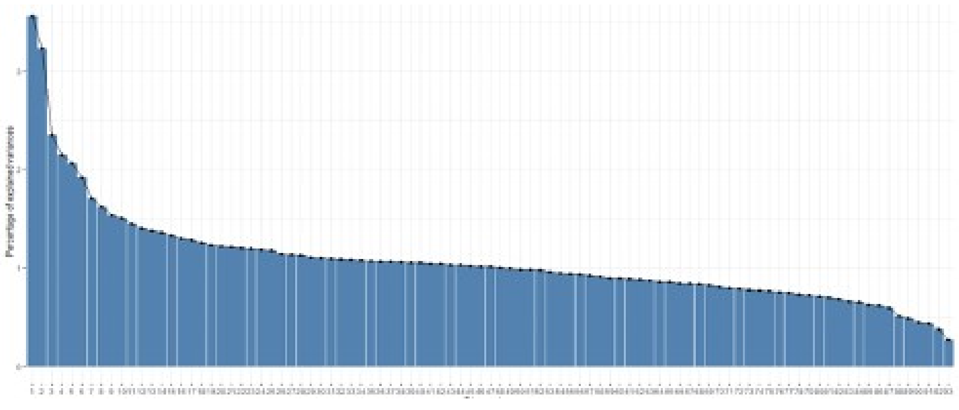


Figure 3: Scree Plot of Percentage of Explained Variance

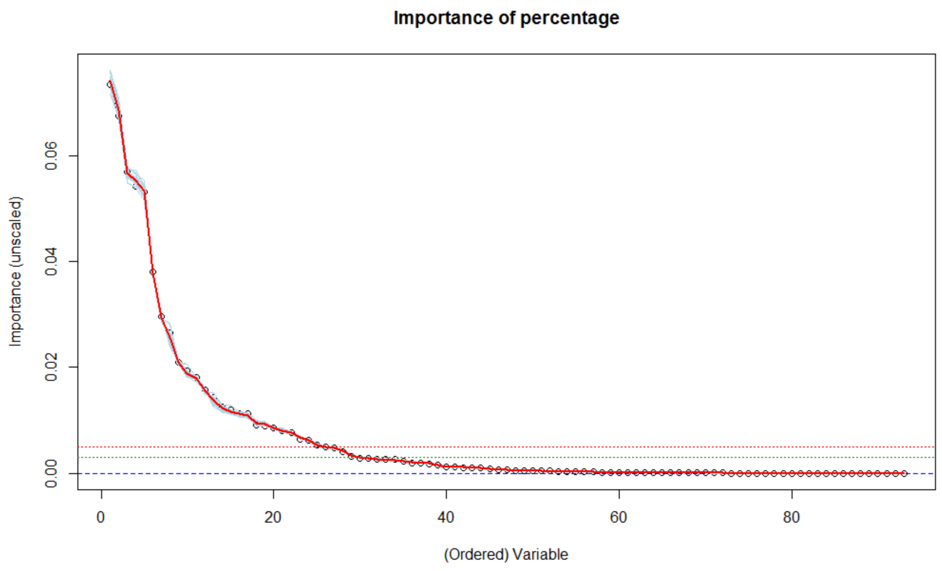


Figure 4: Importance Percentage for Each Variable

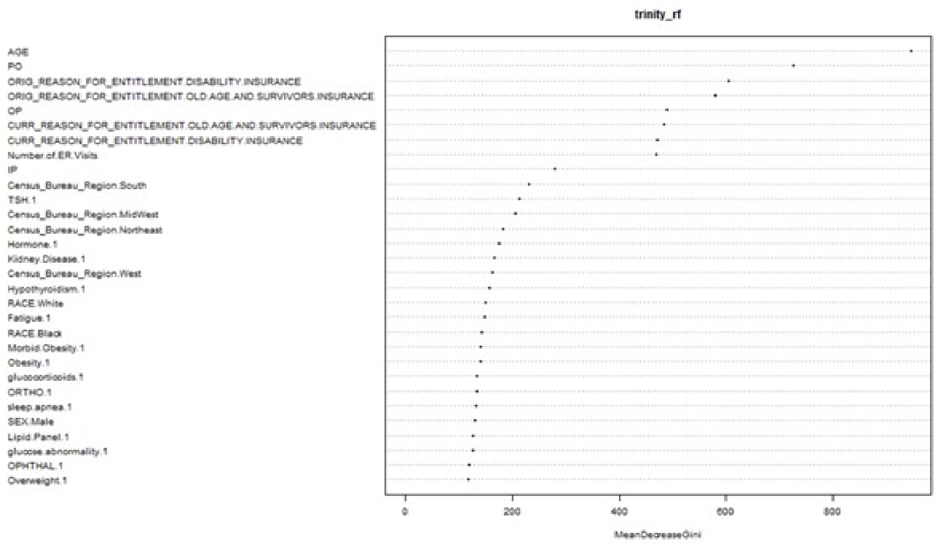
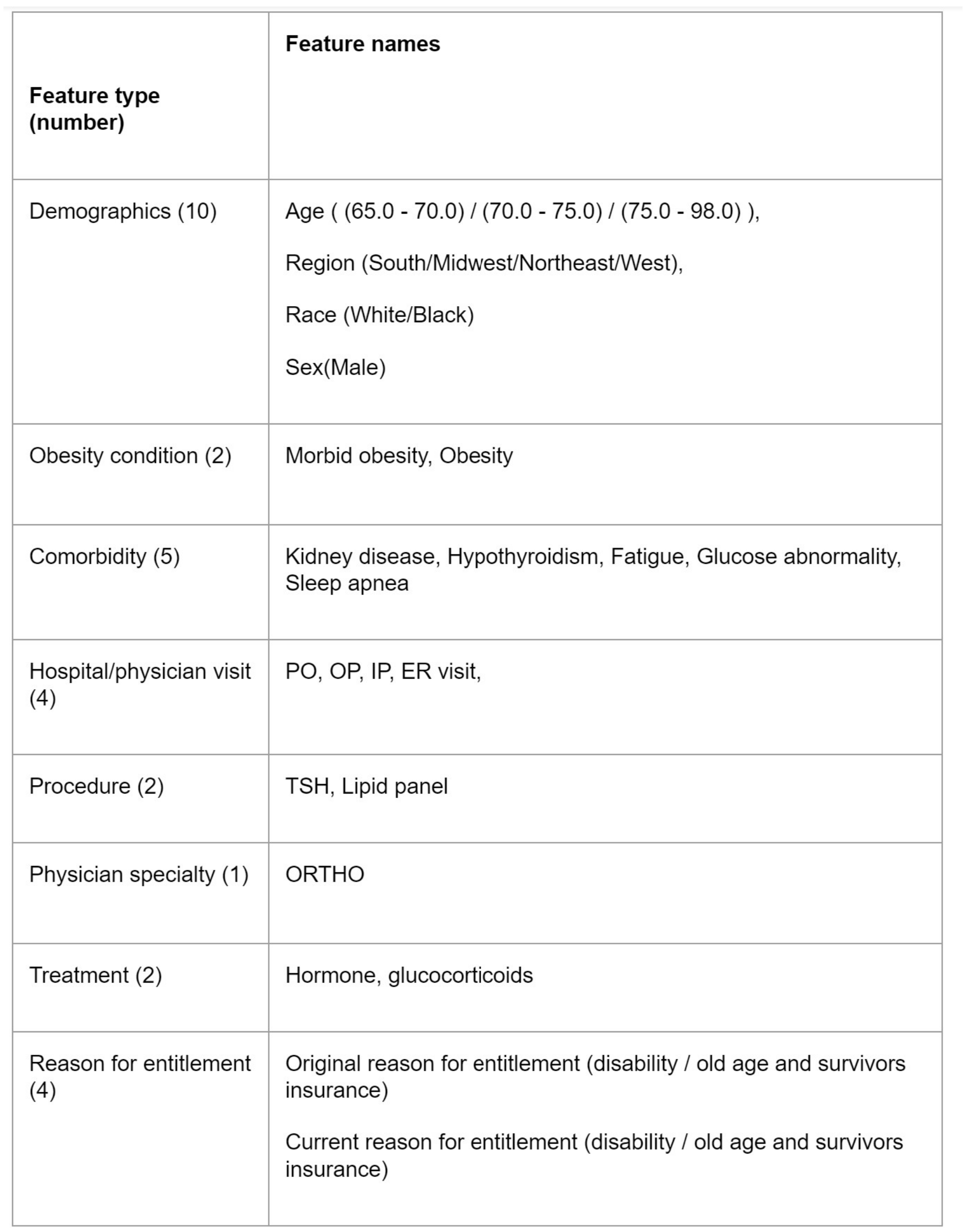


Figure 5: Mean Decrease in Gini Impurity



Feature type (number)	Feature names
Demographics (10)	Age ((65.0 - 70.0) / (70.0 - 75.0) / (75.0 - 98.0)), Region (South/Midwest/Northeast/West), Race (White/Black) Sex(Male)
Obesity condition (2)	Morbid obesity, Obesity
Comorbidity (5)	Kidney disease, Hypothyroidism, Fatigue, Glucose abnormality, Sleep apnea
Hospital/physician visit (4)	PO, OP, IP, ER visit,
Procedure (2)	TSH, Lipid panel
Physician specialty (1)	ORTHO
Treatment (2)	Hormone, glucocorticoids
Reason for entitlement (4)	Original reason for entitlement (disability / old age and survivors insurance) Current reason for entitlement (disability / old age and survivors insurance)

Figure 6: Variables Selected

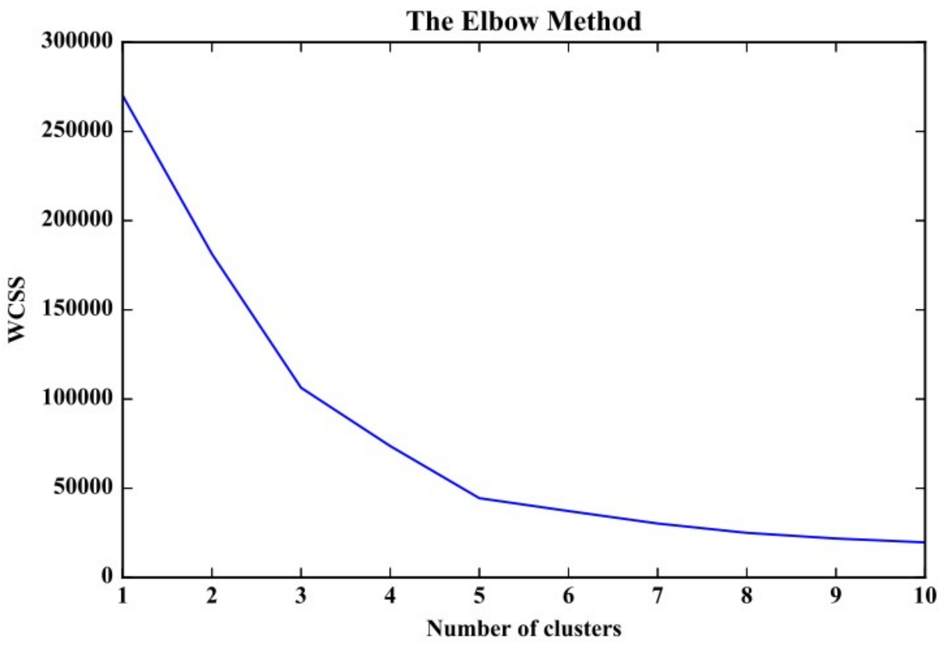


Figure 7: Example of the Elbow Method

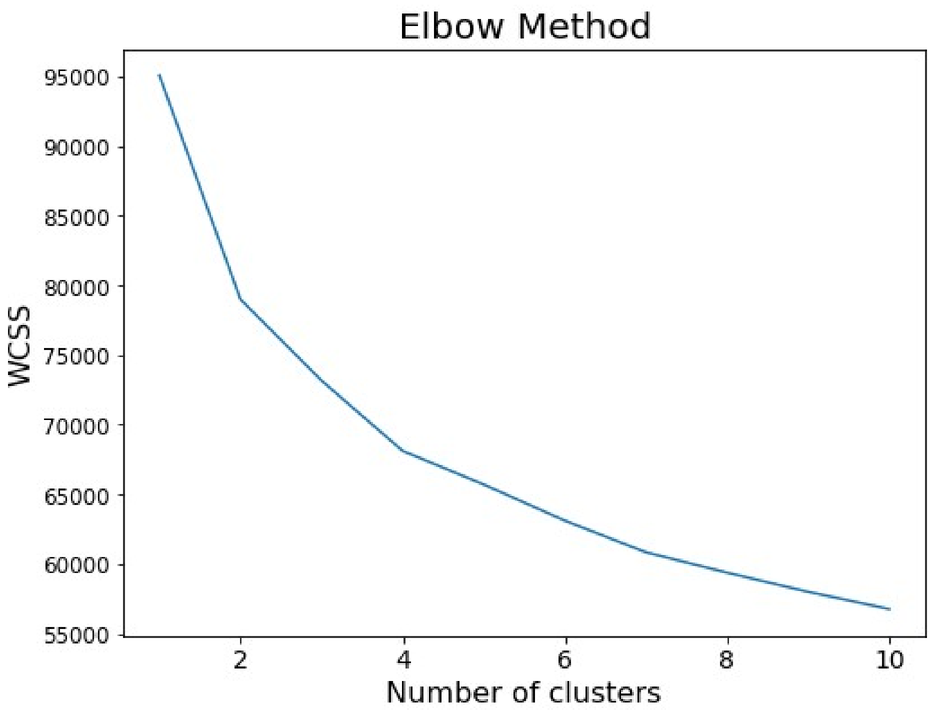


Figure 8: WCSS vs. Number of Clusters

Cluster (Severity)	% of Data	Morbidly Obese	Kidney Disease	Hypothyroidism	Fatigue	Sleep Apnea	Avg. # of Med. Visits
4	1.7%	54.9%	84.6%	45.1%	82.4%	48.9%	122.8
3	13.6%	44.0%	67.0%	38.2%	62.5%	47.3%	40.0
2	24.1%	36.1%	19.3%	23.7%	23.1%	24.4%	11.7
1	60.5%	21.1%	23.5%	24.5%	21.5%	21.6%	9.7

Figure 9: WGenerating 4 clusters using K-Means

Cluster	% of Data	Morbidly Obese	Kidney Disease	Hypothyroidism	Fatigue	Sleep Apnea	Avg. # of Med. Visits
7	0.3%	51.7%	55.2%	24.1%	72.4%	27.6%	177.4
6	1.3%	58.3%	92.8%	51.1%	87.8%	56.1%	113.4
5	5.0%	48.0%	61.3%	35.0%	49.4%	50.2%	45.3
4	8.3%	41.8%	67.8%	35.6%	67.0%	41.9%	37.1
3	29.0%	24.8%	31.0%	51.9%	36.5%	27.4%	13.6
2	22.3%	36.9%	18.4%	19.8%	22.3%	23.6%	11.2
1	33.8%	18.5%	18.6%	4.8%	10.5%	18.5%	7.2

Figure 10: Generating 7 clusters using K-Means

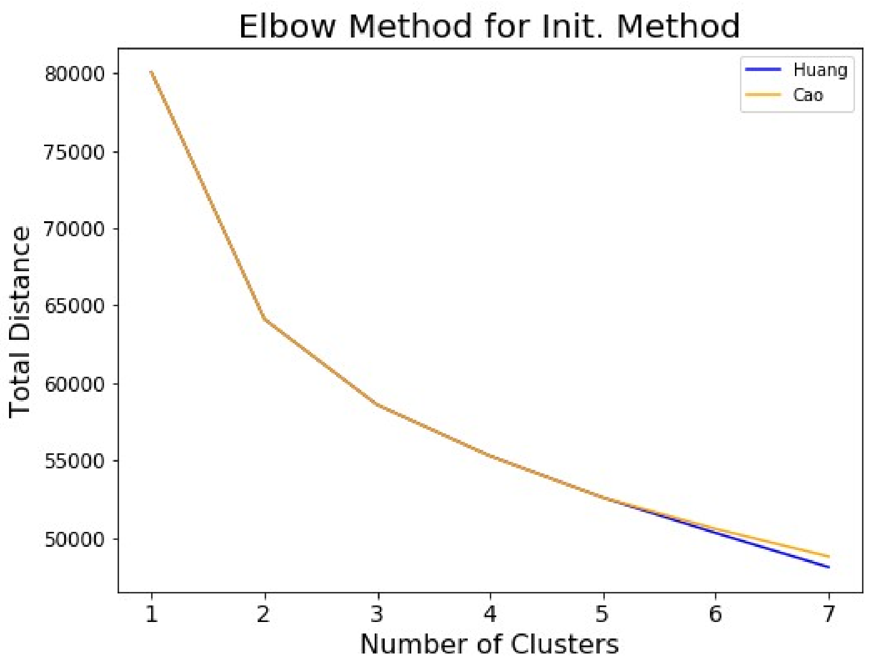


Figure 11: Total Distance with Number of Clusters

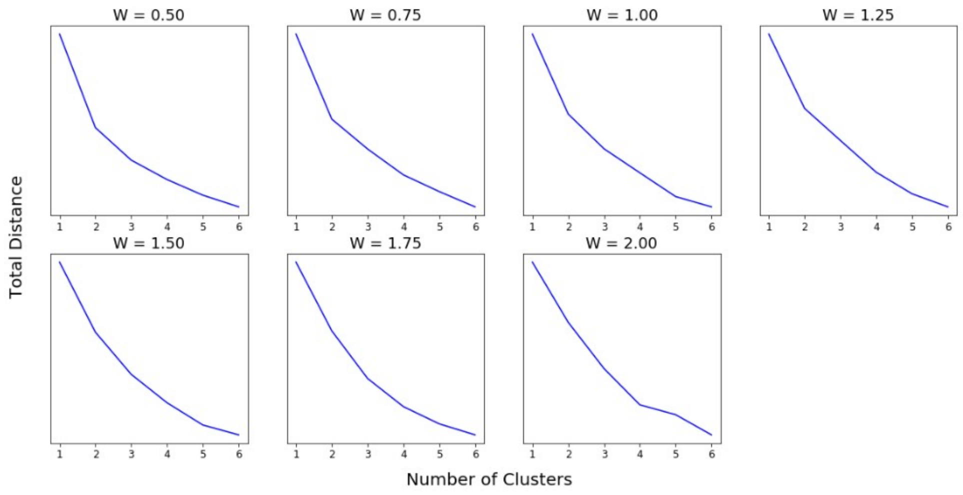


Figure 12: Total Distance with Number of Clusters, W Value differs

Cluster	% of Data	Morbidly Obese	Kidney Disease	Hypothyroidism	Fatigue	Sleep Apnea	Avg. # of Med. Visits
(m. obese) 4	6.0%	46.4%	82.0%	39.4%	81.1%	45.7%	73.9
(m. obese) 3	19.7%	41.9%	22.7%	24.3%	26.4%	27.7%	16.2
(obese) 2	30.3%	23.3%	32.8%	23.3%	26.3%	32.3%	13.7
(obese) 1	44.0%	23.5%	23.0%	28.1%	23.8%	18.8%	10.0

Figure 13: Initialization: Huang; W = 2; Number of clusters = 4

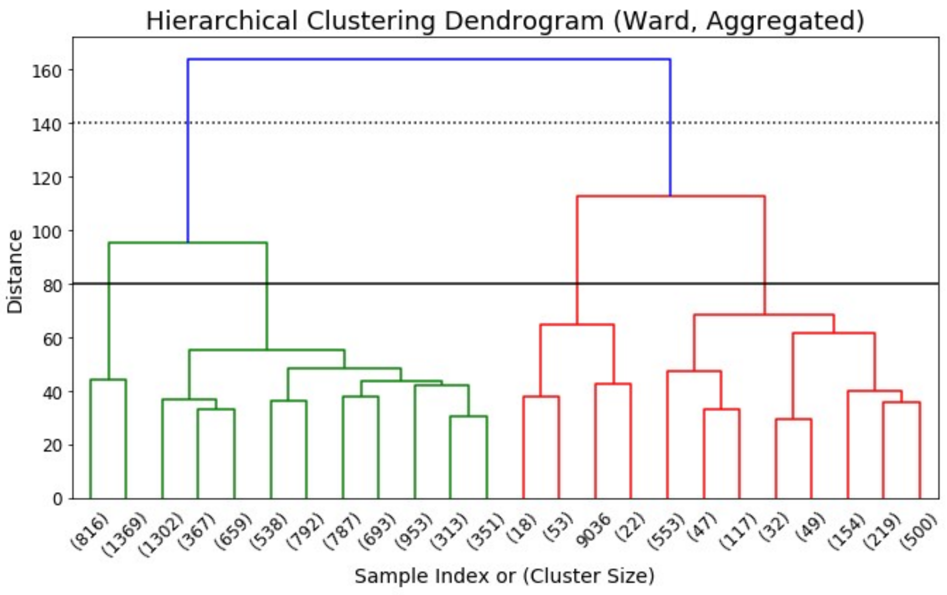


Figure 14: Dendrogram

Cluster (Severity)	% of Data	Morbidly Obese	Kidney Disease	Hypothyroidism	Fatigue	Sleep Apnea	Avg. # of Med. Visits
4	0.9%	54.2%	88.3%	44.7%	85.1%	46.8%	146.9
3	15.6%	47.0%	57.7%	35.9%	58.7%	43.5%	40.2
2	20.4%	33.3%	18.0%	23.2%	21.2%	22.0%	10.5
1	63.1%	21.9%	25.3%	25.0%	22.6%	43.5%	10.3

Figure 15: Generating 4 clusters using AHC

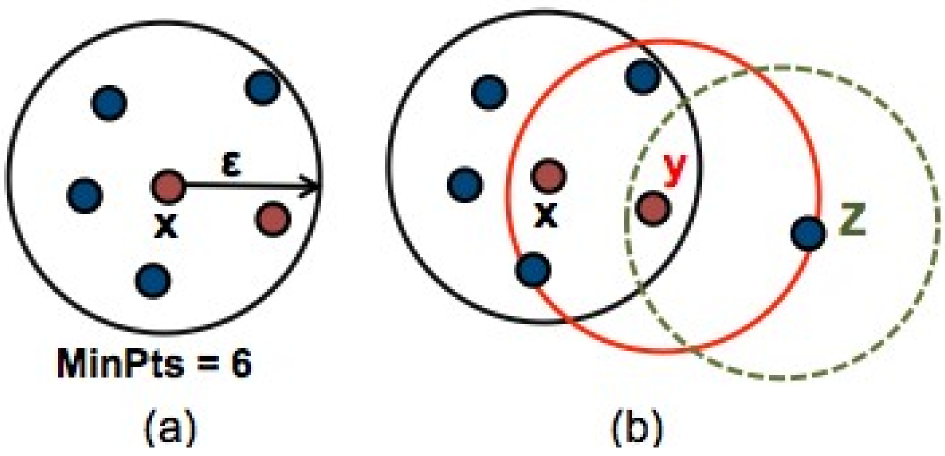


Figure 16: Representation of DBSCAN

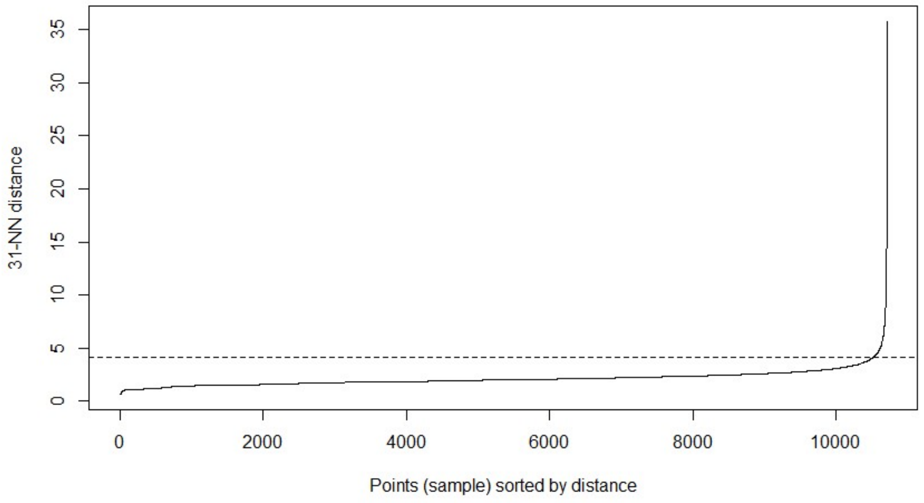


Figure 17: kNN distance plot to find optimal epsilon value

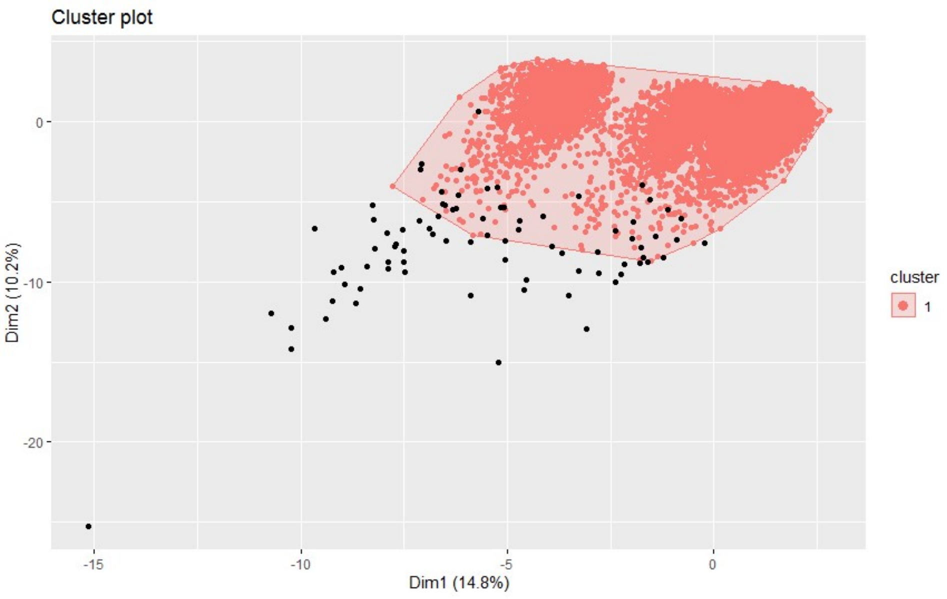
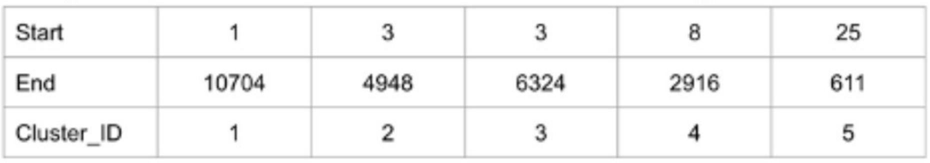


Figure 18: Cluster plot of DBSCAN



Start	1	3	3	8	25
End	10704	4948	6324	2916	611
Cluster_ID	1	2	3	4	5

Figure 19: Resulting Clusters of OPTICS

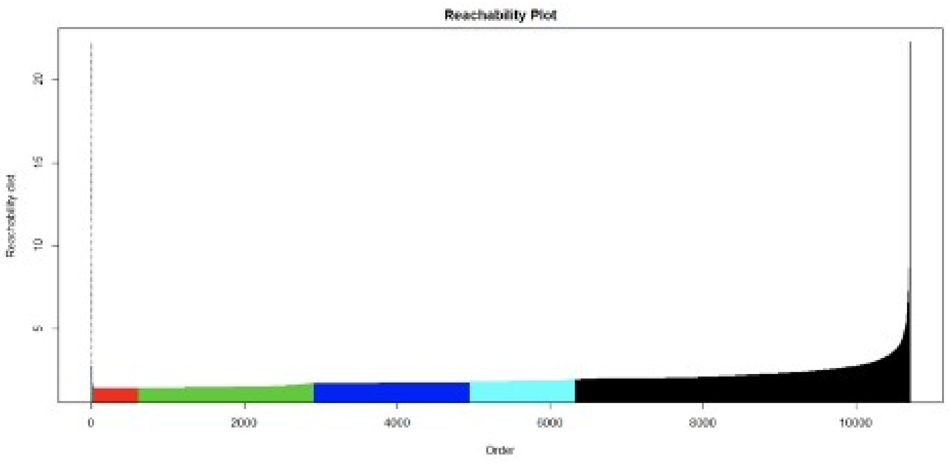


Figure 20: Dendrogram of OPTICS Results

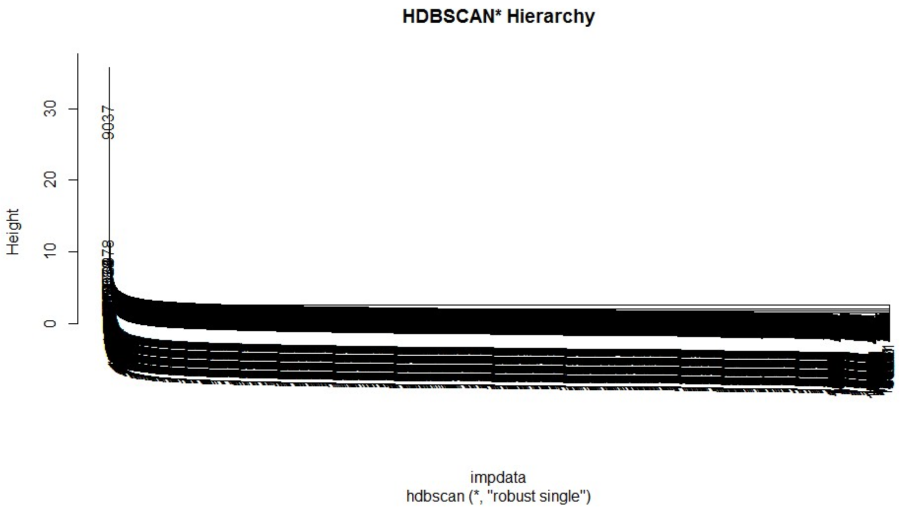


Figure 21: Raw tree plot of HDBSCAN

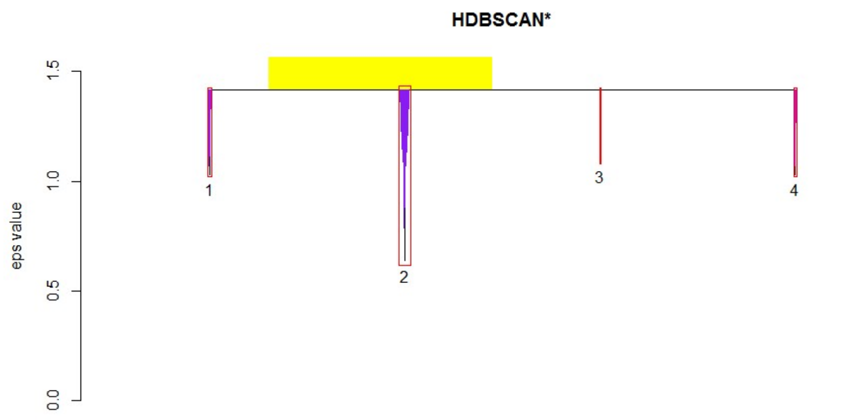


Figure 22: Simplified tree for HDBSCAN

Cluster#	0	1	2	3	4
Points	9857 (92.08%)	177 (1.65%)	481 (4.49%)	53 (0.50%)	137 (1.28%)

Figure 23: Resulting number of clusters and points in each cluster

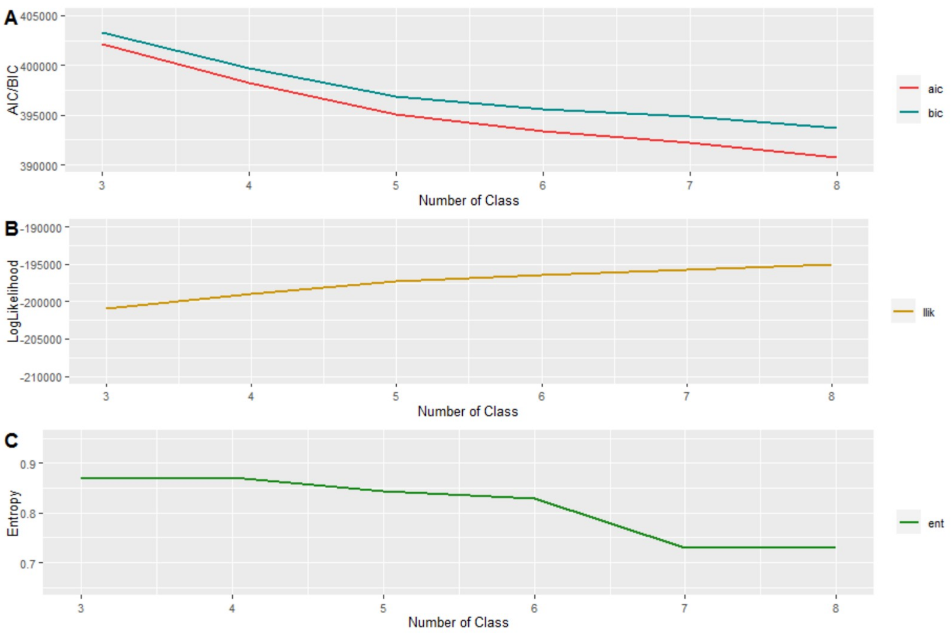
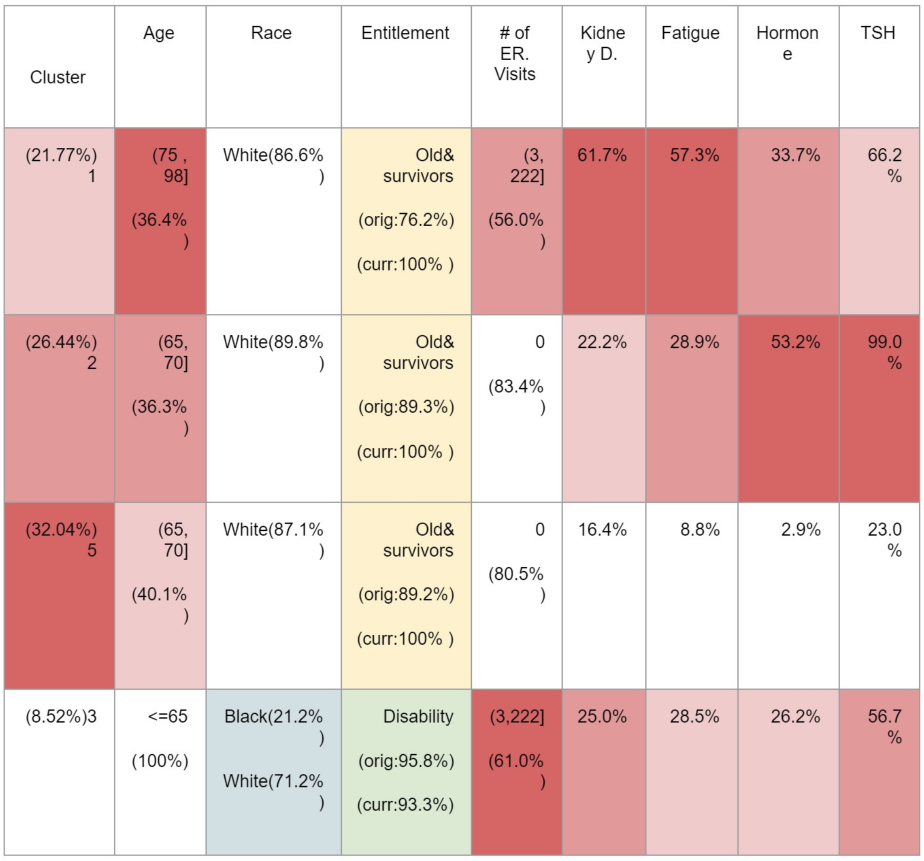


Figure 24: Resulting number of clusters and points in each cluster



Cluster	Age	Race	Entitlement	# of ER. Visits	Kidney D.	Fatigue	Hormone	TSH
(21.77%) 1	(75 , 98] (36.4%)	White(86.6%)	Old& survivors (orig:76.2%) (curr:100%)	(3, 222] (56.0%)	61.7%	57.3%	33.7%	66.2%
(26.44%) 2	(65, 70] (36.3%)	White(89.8%)	Old& survivors (orig:89.3%) (curr:100%)	0 (83.4%)	22.2%	28.9%	53.2%	99.0%
(32.04%) 5	(65, 70] (40.1%)	White(87.1%)	Old& survivors (orig:89.2%) (curr:100%)	0 (80.5%)	16.4%	8.8%	2.9%	23.0%
(8.52%)3	<=65 (100%)	Black(21.2%) White(71.2%)	Disability (orig:95.8%) (curr:93.3%)	(3,222] (61.0%)	25.0%	28.5%	26.2%	56.7%

Figure 25: Results for LCA

Appendix II: Code

```
########### Data Pre−processing and LCA ##########
trinity  <− data.frame(read.csv('Cornell_SPRING_2020_Data_030520.csv',
    header = 1))
# remove Patient_ID,17 bmi buckets, State
c <− c(2:5, 7:11, 29:120)
trinity <− trinity [,c]  #101 variables left
# convert Age into factor with 4 levels
library ( fabricatr )
trinity $AGE <− split_quantile (x = trinity $AGE, type = 4)
# convert int into factor
c <− c(2:77,81:99)
trinity [,c] <− lapply( trinity [,c], as. factor )
# 153 'NA' in variable RACE and 2 in Census_Bureau_Region
trinity $RACE<− as.character( trinity $RACE)
trinity $RACE[is.na( trinity $RACE)]<−0
trinity $RACE<− as.factor( trinity $RACE)
trinity $Census_Bureau_Region<− as.character( trinity $Census_Bureau_Region)
trinity $Census_Bureau_Region[is.na( trinity $Census_Bureau_Region)]<−0
trinity $Census_Bureau_Region<− as.factor( trinity $Census_Bureau_Region)
trinity [is.na( trinity )] <− 0
# remove variables with only 1 level
sum(lapply( trinity , nlevels )==1)   # 21
trinity <− trinity [, −which(lapply( trinity , nlevels )==1)] # 80

```

```
library ( caret )
# for numeric variables , center and scale
# trinity _preprocessed
value_pre <- preProcess( trinity , method=c('center' , ' scale ' ))
 trinity _pre <- predict (value_pre,  trinity )
# for categorical variables , one hot encoding. convert into nums
# using dummy variable x
# trinity _transformed
dmx <- dummyVars("␣~␣.", data = trinity _pre, fullRank = T)
 trinity _tf <- data.frame( predict (dmx, newdata = trinity _pre))
ncol( trinity _tf )   # 95 variables , all num
# feature selection by regularized random forest
write.csv( trinity _tf , 'C:\\Users\\Cornell3\\Documents\\TransformedData_02.
    csv',row.names = FALSE)
set.seed(100)
# try rrf in caret package. not fot unsupervised learning .
# rrf_model, rrf_importance
# rrf_mod <- train(x= trinity _tf ,y=NULL,data=trinity_rf, method="RRF")
# rrf_imp <- varImp(rrf_mod, scale=F)
library (randomForest)
library (varSelRF)
# feature selection by random forest
# trinity _randomForest, hclust_randomForest, random_Forest_cluster
 trinity _rf <- randomForest(x = trinity _tf , y = NULL, ntree = 2000,
    importance = TRUE, proximity = TRUE, oob.prox = TRUE)
# feature importance, shown by mean decrease accuracy/ gini index
# randomForest_importance
rf_imp <- importance( trinity _rf)
# mean decrease  gini  index
# randomForest_giniIndex
rf_gini  <- importance( trinity _rf ,  type=2)
varImpPlot( trinity _rf,  type  =  2,  pch  =  20,  cex  =  0.6)
# when premuted randomly, importance  of   variables
 trinity _rfv  <- randomVarImpsRF(trinity_tf,  NULL, trinity_rf ,  numrandom
     =30, usingCluster=FALSE)
summary(trinity_rfv)
randomVarImpsRFplot(trinity_rfv ,   trinity _rf ,  main = 'Importance␣of␣
     percentage')
abline (h=0.005,  col='red2' ,   lty =3)
abline (h=0.003,  col=' forestgreen ' ,   lty =3)

#  select   variables   after  random  forest
c  <- c
         (81,16,18,80,94,19,21,79,1,3,2,14,74,11,12,77,53,44,15,10,63,24,75,62)
 trinity _sel  <-  trinity _tf [, c]
write .csv(  trinity _sel ,  'C:\\Users\\ SelectedFeatures .csv' ,row.names =
     FALSE)

#   hierarchical
hclust _mod <- hclust(as . dist (1- trinity _rf$proximity) ,  method = "ward.D2")
cluster _cut  =  cutree ( hclust _mod, k=5)
plot ( hclust _mod, cex  =  0.3,  hang  = -1)
```

```
rect.hclust(hclust_mod, k = 5, border = 2:5)
library(cluster)
library(factoextra)
fviz_nbclust(trinity_sel, FUN = hcut, method = "silhouette")
gap <- clusGap(trinity_sel, FUN = hcut, nstart = 25, K.max = 7, B = 50)
fviz_gap_stat(gap)

library(dendextend)
hclust_obj <- as.dendrogram(hclust_mod)
hclust_col <- color_branches(hclust_obj, k=5)
plot(hclust_col)


# poLCA:
# convert every variables into categorical
trinity <- data.frame(read.csv('Cornell_SPRING_2020_Data_030520.csv',
    header = 1))
c <- c(2:5, 7:11, 29:120)
trinity <- trinity[,c]
trinity$IP[is.na(trinity$IP)] <- 0
trinity$OP[is.na(trinity$OP)] <- 0
trinity$PO[is.na(trinity$PO)] <- 0
trinity$Number.of.ER.Visits[is.na(trinity$Number.of.ER.Visits)] <- 0
trinity$Number.of.Urgent.Care.Visits[is.na(trinity$Number.of.Urgent.Care.
    Visits)] <- 0
trinity$AGE <- split_quantile(x = trinity$AGE, type = 4)
```

```
    trinity $IP <- quantcut(x =  trinity $IP,  q = 4)
    trinity $OP <- quantcut(x =  trinity $OP, type  = 4)
    trinity $PO <- quantcut(x =  trinity $PO, type  = 4)
    trinity $Number.of.ER.Visits  <- quantcut(x =  trinity $Number.of.ER.Visits,
        type  = 4)
    trinity $Number.of.Urgent.Care. Visits  <- quantcut(x =  trinity $Number.of.
        Urgent.Care. Visits ,  type  = 4)
    trinity  [,]  <- lapply( trinity  [,],  as. factor )
    trinity $RACE<- as.character( trinity $RACE)
    trinity $RACE[is.na( trinity $RACE)]<-0
    trinity $RACE<- as.factor( trinity $RACE)
    trinity $Census_Bureau_Region<- as.character( trinity $Census_Bureau_Region)
    trinity $Census_Bureau_Region[is.na( trinity $Census_Bureau_Region)]<-0
    trinity $Census_Bureau_Region<- as.factor( trinity $Census_Bureau_Region)
    trinity [ is .na( trinity )] <- 0
    trinity  <-  trinity [,  -which(lapply( trinity ,  nlevels )==1)]


    library ( foreign )
    library (poLCA)
   # no one_hot encoding:
   f <- cbind(
      AGE, PO, OP, Number.of.ER.Visits,  Census_Bureau_Region,
      CURR_REASON_FOR_ENTITLEMENT, ORIG_REASON_FOR_
           ENTITLEMENT,
      TSH, IP,  Hormone, Kidney.Disease,  Hypothyroidism,  RACE, Fatigue,
    Obesity, Morbid.Obesity, glucocorticoids , SEX, ORTHO, sleep.apnea,
    glucose.abnormality, Lipid.Panel, OPHTHAL, Overweight, recurrent.
        respiratory . infection ,
    GASTRO, NEUROL, PT, ENT, ENDOCRIN
  )~1
  lca3 <- poLCA(f, trinity , nclass = 3, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)
  lca4 <- poLCA(f, trinity , nclass = 4, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)
  lca5 <- poLCA(f, trinity , nclass = 5, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)
  lca6 <- poLCA(f, trinity , nclass = 6, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)
  lca7 <- poLCA(f, trinity , nclass = 7, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)
  lca8 <- poLCA(f, trinity , nclass = 8, maxiter = 500, graphs = FALSE, tol =
      1e-10,
                nrep = 3, verbose = TRUE, calc.se = TRUE)

  # entropy for each number of clusters
  # optimal # of clusters : 5
  error_prior<-entropy(lca3$P)
  error_post<-mean(apply(lca3$posterior,1, entropy),na.rm = TRUE)
 e3 <- round(((error_prior-error_post) / error_prior),3)

  error_prior<-entropy(lca4$P)
  error_post<-mean(apply(lca4$posterior,1, entropy),na.rm = TRUE)
 e4 <- round(((error_prior-error_post) / error_prior),3)

  error_prior<-entropy(lca5$P)
  error_post<-mean(apply(lca5$posterior,1, entropy),na.rm = TRUE)
 e5 <- round(((error_prior-error_post) / error_prior),3)

  error_prior<-entropy(lca6$P)
  error_post<-mean(apply(lca6$posterior,1, entropy),na.rm = TRUE)
 e6 <- round(((error_prior-error_post) / error_prior),3)

  error_prior<-entropy(lca7$P)
  error_post<-mean(apply(lca7$posterior,1, entropy),na.rm = TRUE)
 e7 <- round(((error_prior-error_post) / error_prior),3)

  error_prior<-entropy(lca8$P)
  error_post<-mean(apply(lca8$posterior,1, entropy),na.rm = TRUE)
 e8 <- round(((error_prior-error_post) / error_prior),3)

 # after one-hot encoding (just for comparison. do rnot run):
 dmx <- dummyVars(" ~ .", data = trinity , fullRank = T)
```

```
  trinity _tf <- data.frame( predict (dmx, newdata =  trinity ))
  trinity _tf [,] <- lapply( trinity _tf [,],  as. factor )
 f1 <- cbind(
   AGE.2,AGE.3,AGE.4,PO..3.7.,PO..12.273.,PO ..7.12., ORIG_REASON_FOR_
        ENTITLEMENT.OLD.AGE.AND.SURVIVORS.INSURANCE,ORIG_
        REASON_FOR_ENTITLEMENT.DISABILITY.INSURANCE,
   OP ..0.2., OP ..2.4., OP ..4.86., Number.of.ER.Visits ..3.222., Number.of.ER.
        Visits ..0.3., CURR_REASON_FOR_ENTITLEMENT.DISABILITY.
        INSURANCE,CURR_REASON_FOR_ENTITLEMENT.OLD.AGE.AND.
        SURVIVORS.INSURANCE,
   IP ..0.15., Census_Bureau_Region.Northeast,Census_Bureau_Region.West,
        Census_Bureau_Region.South,Census_Bureau_Region.MidWest,Hormone.1,
   Kidney.Disease .1, Hypothyroidism.1,RACE.White,Fatigue.1,Morbid.Obesity .1,
        Obesity .1,
    glucocorticoids .1, TSH.1, SEX.Male
 )~1
 lca3 <- poLCA(f1, trinity _tf , nclass = 3, maxiter = 100, graphs = FALSE,
     tol = 1e-10,
                              nrep = 1, verbose = TRUE, calc.se = TRUE)
 lca4 <- poLCA(f1, trinity _tf , nclass = 4, maxiter = 100, graphs = FALSE,
     tol = 1e-10,
                   nrep = 1, verbose = TRUE, calc.se = TRUE)
 lca5 <- poLCA(f1, trinity _tf , nclass = 5, maxiter = 100, graphs = FALSE,
     tol = 1e-10,
                   nrep = 1, verbose = TRUE, calc.se = TRUE)
```

```
lca6 <- poLCA(f1, trinity_tf, nclass = 6, maxiter = 100, graphs = FALSE,
    tol = 1e-10,
                nrep = 1, verbose = TRUE, calc.se = TRUE)
lca7 <- poLCA(f1, trinity_tf, nclass = 7, maxiter = 100, graphs = FALSE,
    tol = 1e-10,
                nrep = 1, verbose = TRUE, calc.se = TRUE)
lca8 <- poLCA(f1, trinity_tf, nclass = 8, maxiter = 100, graphs = FALSE,
    tol = 1e-10,
                nrep = 1, verbose = TRUE, calc.se = TRUE)


# find optimal number of clusters
# do not run this one
for(i in 2:10){
  max_ll <- -100000
  min_bic <- 100000
  for (j in 1:100) {
    rs <- poLCA(f, trinity_sel, nclass = i, maxiter = 100, tol = 1e-5)
    if(rs$bic<min_bic){
      min_bic <- rs$bic
      best_lca<-rs
    }
  }
}

# visualization for lca5
```

```
num <- c(3:8)
aic  <- c(lca3$aic, lca4$aic, lca5$aic, lca6$aic, lca7$aic, lca8$aic)
bic  <- c(lca3$bic, lca4$bic, lca5$bic, lca6$bic, lca7$bic, lca8$bic)
 llik  <- c(lca3$llik, lca4$llik,lca5$llik,lca6$llik,lca7$llik,lca8$llik)
ent  <- c(e3,e4,e5,e6,e7,e8)
df <- as.data.frame(cbind(num, aic, bic, llik, ent))

 library(ggplot2)
p1<- ggplot()+
   geom_line(data = df,aes(x = num,y = df$aic,colour = 'aic'),size=1)+
   geom_line(data = df,aes(x = num,y = df$bic,colour ='bic'),size=1) +
   ylim(390000,405000 )+
   scale_colour_manual('',values = c('aic' = 'brown2','bic' = 'darkcyan'))+
   xlab('Number␣of␣Class')+ylab('AIC/BIC')+
   theme(text=element_text(size=10, family="Comic␣Sans␣MS"))

p2 <- ggplot()+
   geom_line(data = df,aes(x = num,y = df$llik,colour ='llik'),size=1) +
   ylim(-210000, -190000)+
   scale_colour_manual('',values = c('llik'='darkgoldenrod3'))+
   xlab('Number␣of␣Class')+ylab('LogLikelihood')+
   theme(text=element_text(size=10, family="Comic␣Sans␣MS"))

p3 <- ggplot()+
   geom_line(data = df,aes(x = num,y = df$ent,colour ='ent'),size=1) +
   ylim (0.65,0.95) +
        scale_colour_manual('', values = c('ent'=' forestgreen '))+
        xlab('Number␣of␣Class')+ylab('Entropy')+
        theme(text=element_text(size=10, family="Comic␣Sans␣MS"))
      library (cowplot)
      plot_grid(p1, p2, p3, labels = "AUTO", ncol = 1)

      ########## PCA and OPTICS ##########
      # preparation
 install .packages(' factoextra ')
 library ( factoextra )
 install .packages('dbscan')
 library (dbscan)

## Data Summary
# summary statistics :
dim(data)
# sex
 levels ( factor (data$SEX))
sum(factor(data$SEX)== 'Female')
sum(factor(data$SEX)== 'Male')
# age
summary(data$AGE)
 hist (data$AGE)
sum(data$AGE < 20)
# race
 levels ( factor (data$RACE))
```

```
sum(na.omit( factor (data$RACE))== "Asian")
sum(na.omit( factor (data$RACE))== "Black")
sum(na.omit( factor (data$RACE))== "Hispanic")
sum(na.omit( factor (data$RACE))== "North American Native")
sum(na.omit( factor (data$RACE))== "Other")
sum(na.omit( factor (data$RACE))== "White")
# region
 levels ( factor (data$Census_Bureau_Region))
sum(na.omit( factor (data$Census_Bureau_Region))== 'MidWest')
sum(na.omit( factor (data$Census_Bureau_Region))== 'Other')
sum(na.omit( factor (data$Census_Bureau_Region))== 'South')
sum(na.omit( factor (data$Census_Bureau_Region))== 'Northeast')
sum(na.omit( factor (data$Census_Bureau_Region))== 'West')
# state
 nlevels ( factor (data$ State ) )
# obesity concerns
sum(data$Overweight == 1)
sum(data$Obesity == 1)
sum(data$Morbid Obesity == 1)
sum(data$Overweight == 1 & data$Obesity == 1)
sum(data$Overweight == 1 & data$Morbid Obesity == 1)
sum(data$Obesity == 1 & data$Morbid Obesity == 1)
sum(data$Overweight == 1 & data$Obesity == 1 & data$Morbid Obesity == 1)
# original reason for entitlement
 levels ( factor (data$ORIG_REASON_FOR_ENTITLEMENT))
sum(data$ORIG_REASON_FOR_ENTITLEMENT == "BOTH DIB AND ESRD")
```

```r

sum(data$ORIG_REASON_FOR_ENTITLEMENT == "DISABILITY INSURANCE")
sum(data$ORIG_REASON_FOR_ENTITLEMENT == "ESRD")
sum(data$ORIG_REASON_FOR_ENTITLEMENT == "OLD AGE AND 
    SURVIVORS INSURANCE")
# current reason for entitlement
 levels ( factor (data$CURR_REASON_FOR_ENTITLEMENT))
sum(data$CURR_REASON_FOR_ENTITLEMENT == "BOTH DIB AND ESRD")
sum(data$CURR_REASON_FOR_ENTITLEMENT == "DISABILITY INSURANCE"
    )
sum(data$CURR_REASON_FOR_ENTITLEMENT == "ESRD")
sum(data$CURR_REASON_FOR_ENTITLEMENT == "OLD AGE AND 
    SURVIVORS INSURANCE")

## PCA: Data Preparation
# convert all non-numeric categorical variables into numeric factors
datapca$SEX = as.numeric(as. factor (datapca$SEX))
datapca$RACE = as.numeric(as. factor (datapca$RACE))
datapca$Census_Bureau_Region = as.numeric(as. factor (datapca$Census_Bureau_
    Region))
datapca$State = as.numeric(as. factor (datapca$State ))
datapca$ORIG_REASON_FOR_ENTITLEMENT = as.numeric(as.factor(datapca$
    ORIG_REASON_FOR_ENTITLEMENT))
datapca$CURR_REASON_FOR_ENTITLEMENT = as.numeric(as.factor(datapca$
    CURR_REASON_FOR_ENTITLEMENT))
# remove columns with constant values
```

```
checkconstant = c()
for (i in 1:length(datapca)){
  checkconstant[i] = sum(datapca[,i], na.rm = T)
}
checkconstant
sum(checkconstant == 0)
coldelete = c()
for (i in 1:length(checkconstant)){
  if (checkconstant[i] == 0){
    coldelete = c(coldelete, i)
  }
}
coldelete
colnames(datapca[, coldelete])
datapca = datapca[, -coldelete]
# remove columns with too many NAs
nadelete = c()
for (i in 1:length(datapca)){
  if (mean(is.na(datapca[,i]))>0.1){
    nadelete = c(nadelete, i)
  }
}
nadelete
colnames(datapca[, nadelete])
datapca = datapca[, -nadelete]
# remove remaining data with NAs
```

```
datapca = na.omit(datapca)

## PCA: Process and Results
# conduct pca
pca = prcomp(datapca [,],  center = T, scale = T)
# 2-pc plot
plot(pca$x [,1],  pca$x [,2],  pch = 19, cex = 0.5,  xlab = "PC1", ylab = "PC2")
# scree plot
fviz_screeplot(pca, ncp = (dim(datapca)[2]-1))
# factor map
eigens = pca$eig
fviz_pca_var(pca, col.var = "cos2") +
  scale_color_gradient2(low = "white", mid = "blue", high = "purple", midpoint
       = 0.5) +
  theme_minimal()
# variance explanation
pca$sdev
sum(pca$sdev)
sum(pca$sdev[1:1])
sum(pca$sdev[1:45])
sum(pca$sdev[1:55])
sum(pca$sdev[1:66])
sum(pca$sdev[1:78])
sum(pca$sdev[1:91])

## OPTICS
```

```
# conduct optics
res = optics(datarf, eps = 200, minPts = 200)
res1 = extractXi(res, xi = 0.001)
res1
res1$clusters_xi
seqcolor = seq(1, 10705,1)
 colorlist = c()
for(i in 1:length(seqcolor)){
   if (seqcolor[i] >= 25 & seqcolor[i] <= 611){ colorlist [i] = 2}
   else if (seqcolor[i] >= 8 & seqcolor[i] <= 2916){ colorlist [i] = 3}
   else if (seqcolor[i] >= 3 & seqcolor[i] <= 4948){ colorlist [i] = 4}
   else if (seqcolor[i] >= 3 & seqcolor[i] <= 6324){ colorlist [i] = 5}

   else{ colorlist [i] = 1}
}
plot(res, col = colorlist )
```