Austin Prince

Homework 3

Data Mining

**P0:** In order to preprocess the data I used the python Pandas library. The original data was missing a number of data points. In order to impute these missing data points I selected a random point within the Gaussian distribution of gene expression levels for a particular gene. This process is detailed further in prior homework, and I used the same methods for this lab.

The further preprocessing that we wanted to do for this lab was to assume that all genes in a cell (or patient) are in the range of [-1, +1], where -1 and +1 represent the min and max values of this sample.

In order to accomplish this I used the MinMaxScaler tool from sklearn in python. In order to use this tool you give the min and the max value that you would like your data to be scaled to and it normalizes it to this given range with the following normalization formula.

Given the following code

MinMaxScaler(feature\_range(min, max)).fit(input\_matrix)

the MinMaxScalar will normalize your data according to the following formula

This will return the inputted matrix transformed such that each patient is scaled to be within [-1, +1], with -1 represented the gene with the lowest expression , and +1 representing the gene with the highest expression.

**P1:** For this problem we are attempting to find clusters of similar genes using different similarity metrics. Two larger problems that we are attempting to solve here are

1. What is the optimal number of clusters?
2. What is the best similarity measure to distinguish these clusters?

In order to tackle these problems I ran the KMeans algorithm for k value from 2-100 using both the Euclidean distance similarity metric and the Dot Product similarity metric.

I used sklearn again for this problem with some minor tweaks. The following two lines of code fits a KMeans with k=3 to the dataset and identifies the clusters.

kmeans = KMeans(n\_clusters=3)

kmeans.fit(df)

In order to change the similarity metric used by the KMeans algorithm I used the following code.

kmeans.euclidean\_distances = dot\_product

Where dot product is a function I defined that computes the dot product between two vectors.

This will return to me a list of labels describing which cluster each of the points in the dataset was placed in. Using this I can compute the following 3 metrics which are useful in determining if k is optimal and by extension how well the data was classified with this given k.

1. **S – average in cluster similarity (dot product as similarity measure)**

In order to compute the average in cluster similarity I used the following formula.

This will compute the in cluster similarity of one cluster, if I want to find the average over all the clusters I will use the following formula.

The equation for in cluster similarity will give us a metric of how similar all the elements in a given cluster are. Given that we are using the dot product as a similarity measure a S value of 0 indicates very low similarity and an S value of 1 indicates very high similarity.

1. **D – between cluster similarity (dot product as similarity measure)**

For this calculation I used the centroids of a given cluster to compute the similarity between two clusters. I used the following formula to compute this value.

This metric will tell us similar each cluster is to each other cluster. A D value of 0 indicates that each cluster is very unique (far apart) and a high D value indicates that the clusters are similar to each other (closer together).

1. **S/D – ratio of in cluster to between cluster similarity**

This metric will be instructive in telling us how the ratio of in cluster similarity to between cluster similarity. Given that we want in cluster similarity to be high (compact clusters) and between cluster similarity to be low (unique clusters) we want the value of this metric to be high.

Plotting the values of S, D and S/D from k=2 to k=100 using dot product as the similarity metric I obtained the following graph.

I have included another graph of just S/D because it is much smaller than either S or D.

In this example the values of S and D are increasing steadily as we increase K. If we keep in mind two rules we can reason how these rules make sense.

1. The dot product of a vector with itself is
2. The dot product of a vector with a vector that is orthogonal to it is 0

**S analysis**

* As we increase k the value of in cluster similarity (S) increases because the instances (patients) in this cluster will be more similar to each other.
* This is reasonable because with 2 clusters we are fitting all of our patients into 2 clusters whereas with 100 clusters, given that we have only 188 patients some clusters would be of size 1 (dot product of two similar vectors is higher, or vector with itself)

**D analysis**

* As we increase k the value of between cluster similarity (D) will go up as well. This is because we are filling the same geometric space with more clusters.
* The centroids are points in 8651 dimensional space that is bounded by the range of gene expression over all the patients. The centroids therefore have to fill this same geometric space with more points as we increase k (number of clusters), so it is sensible that the similarity of the centroids will increase as the number of clusters (k) increases.

**S/D analysis**

We want S (in cluster similarity) to be high which indicates that the all the patients in a given cluster are very similar, we also want D (between cluster similarity) to be low which indicates that the clusters are very distinct. This means that our optimal k for number of clusters would be a high value, indicating a high S, and a low D.

I ran the same experiments again, except this time I used Euclidean distance as a similarity measure to measure in cluster similarity, and between cluster similarity. Something to keep in mind for this analysis is that Euclidean distance is just that, a distance metric, it is not strictly a similarity metric. It is only informative of the inter and intra cluster distance.

Plotting the values of S, D and S/D using Euclidean distance as the metric I obtained following graph.

Because in this case again S/D is much smaller I have included a plot of this as well.

**S analysis**

* This is very similar to our inter cluster similarity that we obtained using the dot product, as k (number of clusters) increases the inter cluster similarity increases
* This is for the same reason, as we increase the number of clusters the number of instances (patients) per cluster decreases, meaning that they will be more similar

**D analysis**

* Using the Euclidean Distance as a similarity metric we see different patterns than we did using dot product, this is because Euclidean Distance does not take into account the angle between the vectors
* As we increase the number of clusters it makes sense that the intra cluster *distance* would increase, because the clusters would spread more uniformly across the space
  + For example if we have 2 clusters classifying the data they would be closer to the center of the data, but if we have 50 clusters on average they would be more specialized (further from each other), classifying smaller subsets of the data meaning the average distance between them would be higher

**S/D analysis**

We want S (in cluster similarity) to be high which indicates that the all the patients in a given cluster are very similar, we also want D (between cluster similarity) to be low which indicates that the clusters are very distinct. This means that our optimal k for number of clusters would be a high value, indicating a high S, and a low D. In this case it is very easy to see if our number of clusters increases then our S/D metric will also increase.

**Best Similarity Measure**

The question of what is the best metric can be estimated by comparing the three graphs that we obtained. The Silhouette Score is calculated using the Euclidean distance. I would say that the Euclidean distance metric is better for this problem because our search space is quite large. Using the dot product similarity score we are taking into account the angle between the two vectors. For this reason, the vector (0, 1) is more similar to (0, 10000) than it is to (1, 1). Using Euclidean distance, we obtain the distance between the clusters which is more informative than the angle between the clusters.

We want vectors (1, 1) and (0, 1) to be more similar than (0, 1) and (0, 10000) which is the case for the Euclidean distance, but is not the case for the dot product metric.

**Optimal Number of Clusters (k)**

As discussed above we want our S value (inter cluster similarity) to be high meaning that each individual cluster is very similar, and we want our D value (intra cluster similarity) to be low, meaning that each cluster is unique. If we analyze the Euclidean distance metric, we can see that as we increase the number of clusters, S/D is always increasing, but we want to find the *optimal* number for k (clusters), so we can search for a *point of diminishing returns*.

I used an additional method for finding the optimal k called the silhouette score. I used the Euclidean Distance to measure my silhouette score but you can use any metric.

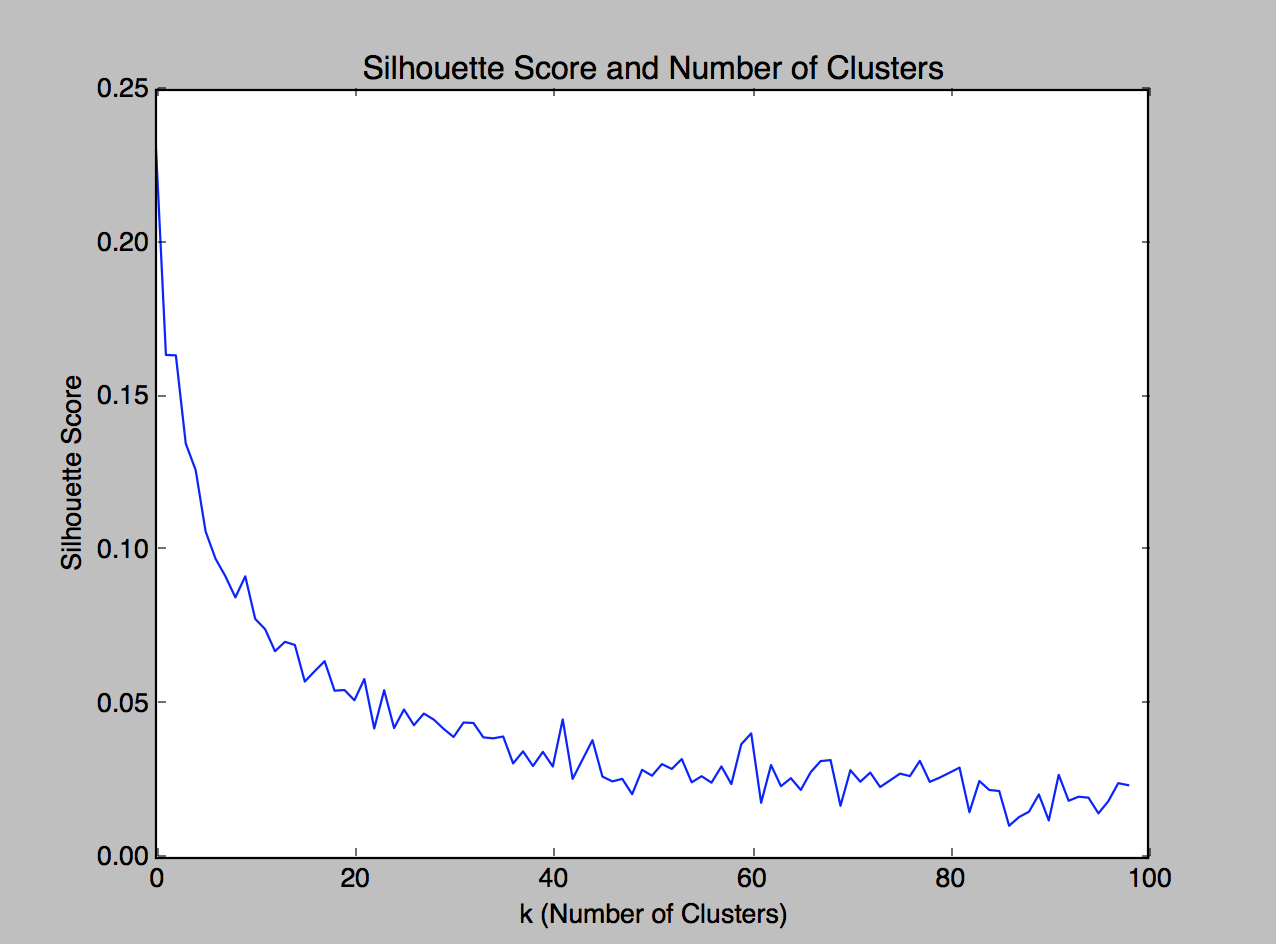
1. Silhouette Method: The Silhouette Method is similar to the metric that we used to determine the optimal number of clusters. It takes into account the intercluster and intracluster similarity. The metric can be mathematically defined as follows.

*For any data point (i) let a(i) be the average dissimilarity between point (i) and all other points in its assigned cluster. A smaller value of a(i) indicated a better assignment.*

*Let b(i) be the lowest average dissimilarity of (i) to any other cluster, that (i) is not a member of.*

*We want this value to be lower, indicated that incluster similarity is high and between cluster similarity is low.*

Using Pythons built in metric, silhouette\_score, I made a plot of the silhouette score of my classifications.



The silhouette score is similar to the trend of the metric (S/D) that we obtained. Thus we can say with confidence that fewer clusters is more optimal in this case. If we have fewer clusters we face a couple tradeoffs that turn out to be acceptable in the end.

* The inter cluster similarity is lower
* The intra cluster is similarity is lower (by a higher magnitude)

As we increase the number of clusters we can see that the silhouette score becomes better, however if we take into account another method, known as the elbow method then we can see at which point we reach the point of diminishing returns. This will be when the curve of our graph transitions from being an exponential curve to a linear curve. If we examine our graph above closely we can see that at between k=20 to 40 we have a levelling off of the silhouette score. So we will want to choose a k in this range for the optimal modelling of this dataset.

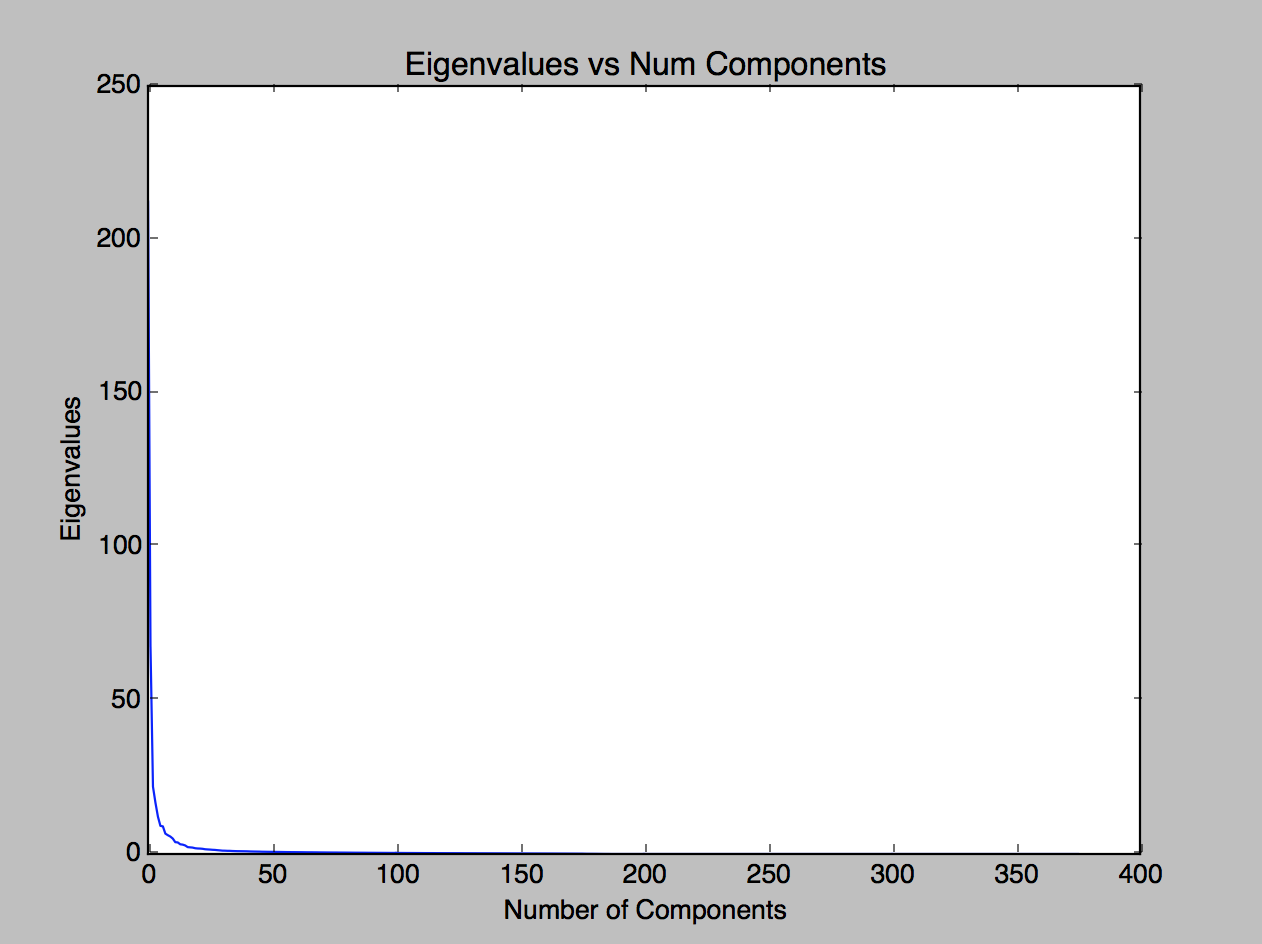
**P2:** For this problem we want to use PCA to reduce the dimensionality of our dataset.

I used the sklearn library in python which provides some simple tools to obtain eigenvectors and eigenvalues, as well as accumulative information. In order to create a PCA model I used the following code in python.

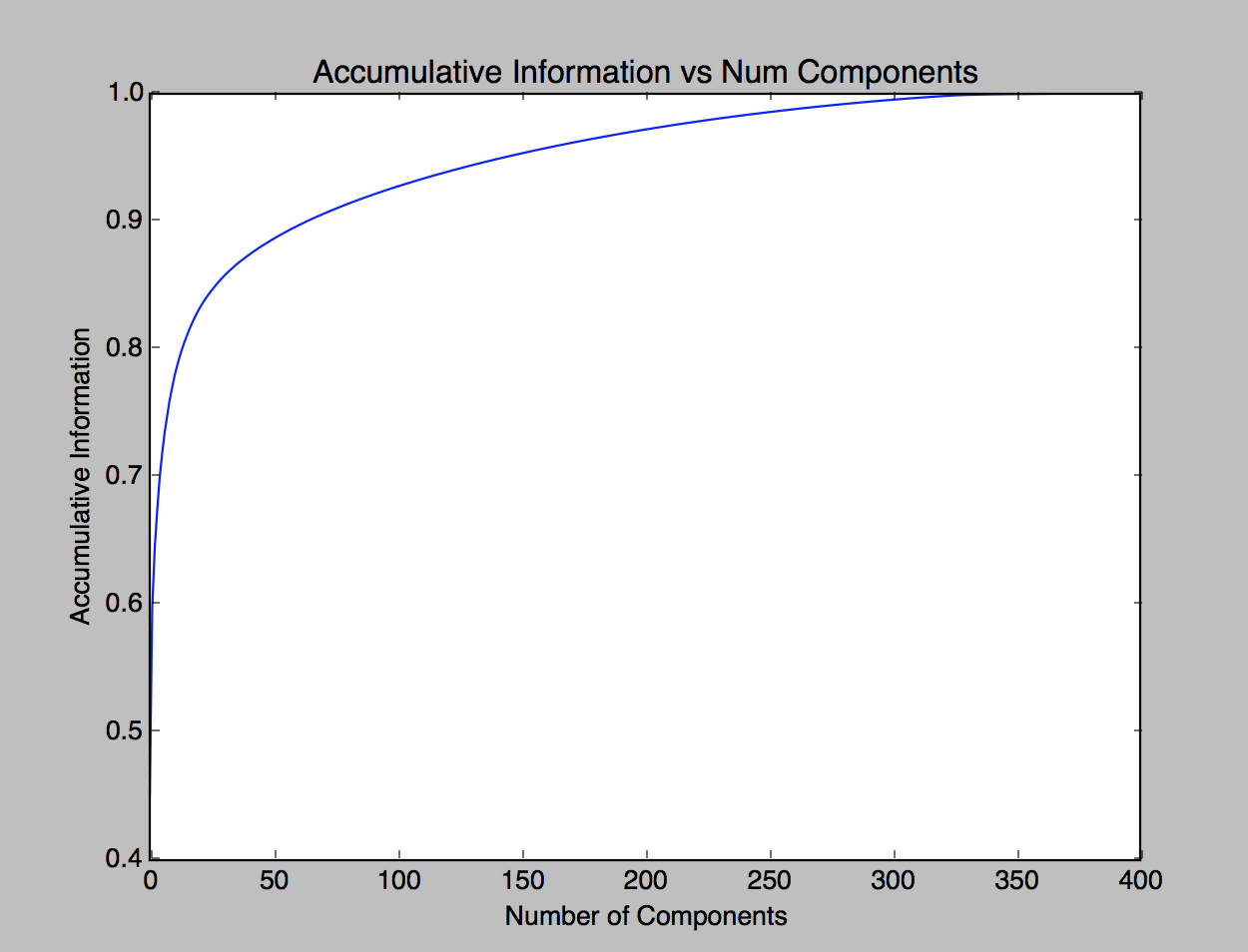
pca = decomposition.PCA()

This PCA model in python returns the eigenvectors as well as the eigenvalues associated with these association vectors. In addition to this it also the variance ratio, which is the same as information, so I can use this to obtain to the accumulative information.

1. Plotting the eigenvalues against the number of eigenvectors I obtained the following plot.

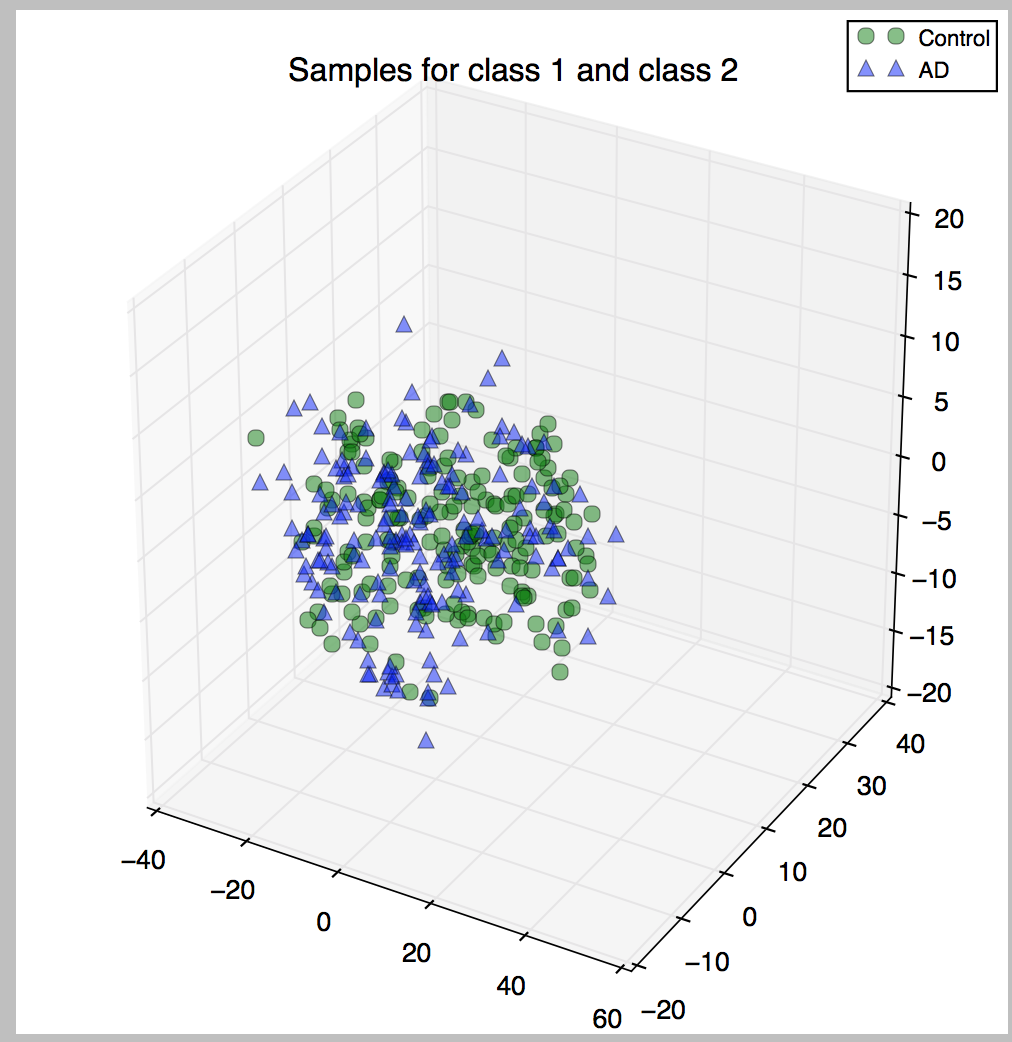


From this we can see that the top eigenvalues of the most important components are much higher than those of less important components. Using the accumulative information gain metric we can see just how much information we can retain in reducing the dimensionality of our input. Plotting the accumulative information against the number of components I obtained the following graph.

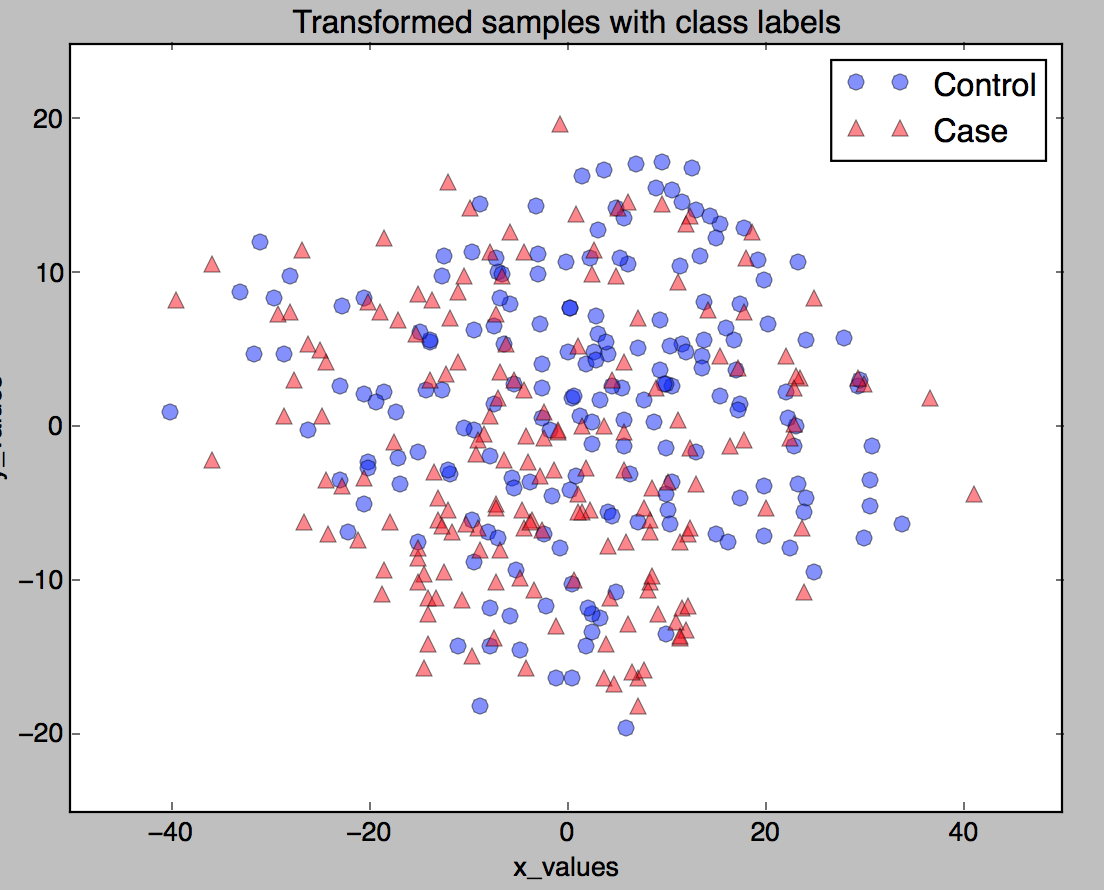


In analyzing this graph and the corresponding list that was used to create it, if we use a k value of 15 we will retain 80.5% of the information.

1. Plotting the AD cases in in blue and the control cases in green in a 3d plot, with the coordinates corresponding to the three largest eigenvectors I obtained the following plot. Looking at this plot it is quite difficult to ascertain a clear separation between the data. This is most likely because we are not capturing a sufficient amount of information. If we refer to the plot above about cumulative information, we can see that with the top 3 eigenvectors we are retaining about 65% of the information, which as evidenced by this graph is not enough information to separate the data.



Plotting the samples again in 2 dimensional space I obtained the following graph.



Yet again in this case we cannot see a clear separation of the data points. This is probably for the same reason as the above 3d case, we are not retaining enough information to create a clear separation of our data points (ie accumulative information <= 0.65).

Running 10 fold cross validation on these transformed datasets using a Support Vector classifier, I obtained about a 53% accuracy for the 2d case and 57% accuracy for the 3d case, which is in line with what I am seeing, in that it is very hard to distinguish between the classes with this transformation.