**Datasets**

My data sets are Bank and Eyes.

Each row in the Bank data set represents correspondence with a single bank client. The Bank “Y” (or “answer”) column is a boolean: “yes” or “no” for whether or not the client subscribed to a term deposit As the bank in question attempts to get clients to sign up for a term deposit, it records the frequency with which that client is contacted about the offer. As such, the features in the data set are a combination of the client’s personal information and contact history.

The Eyes data set is comprised of data collected during an EEG, which is a medical test in which doctors hook up a bunch of electrodes to a person’s brain and record the outputs of each electrode. In this data set, each row is a point in time, the features are the outputs of each electrode, and the Y is a boolean representing whether the patient’s eyes are open or closed.

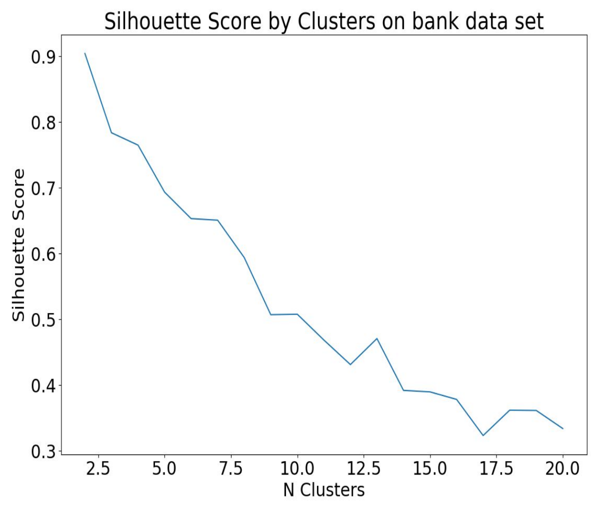
The Bank data set has almost exclusively class or boolean data as opposed to real numbers, whereas the Eyes data set has only real numbers. Additionally, the columns of the Bank data set are very different from each other, e.g. birthday and marital status, while each attribute of the Eyes data set is signal strength of a particular point in the same brain. The stark difference in the structure of these data sets makes them interesting to use together in the same algorithms.

**Clusters**

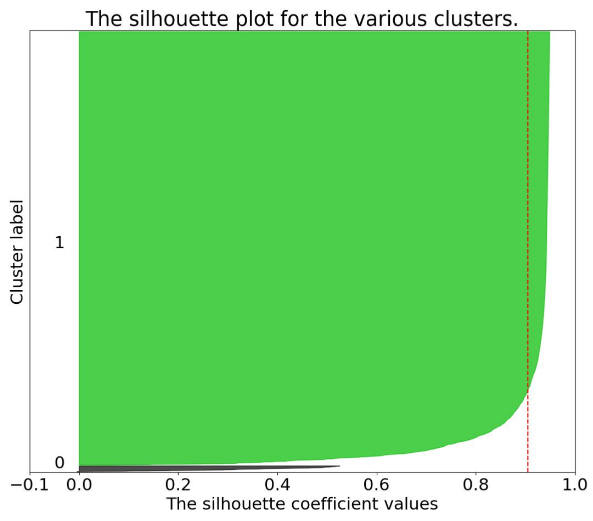
*KMeans*

KMeans clustering on the Bank data with no pre-processing gives a silhouette score of 0.9 for 2 clusters, which is far higher than the silhouette score for other number of clusters. However, upon reviewing the clustering, the vast majority of the points are in the same cluster with the other cluster representing a tiny fraction of the data. Since the Y values are evenly distributed (50% “yes”), such a lopsided cluster cannot be useful. Upon reviewing further, it is clear that the clustering was based on the “balance” attribute, which represents the client’s bank account balance. The bank account balance attribute is lopsided in that a tiny fraction of clients have a far higher balance than the other 99% of clients, it seems to have no correlation with the Y column, and all of the values are much higher in absolute terms than any of the other columns.

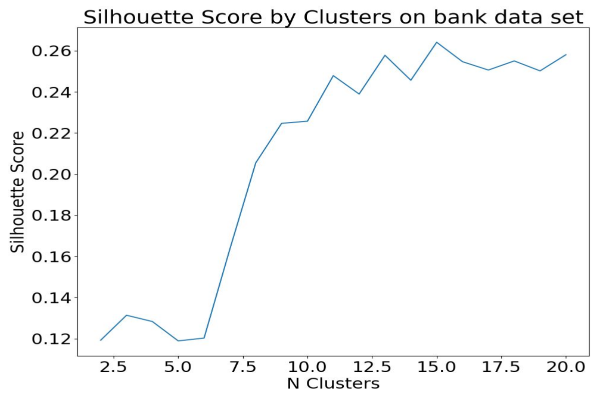
To fix this problem, the data is normalized before clustering. Additionally, to account for falsely large distances between different classes of the class attributes a One Hot Encoding is applied to the features, by which is class is split into separate boolean columns for each of its values, thus eliminating arbitrarily changing distances between difference values of the same class attribute. After preprocessing, the optimal silhouette score was significantly different but much more reasonable. Both the Silhouette Score maximum and the elbow point for Sum of Squared Error occur at 15 clusters. However, with 32 features and 15 clusters, deciphering how the attributes are mapped into clusters becomes seemingly impossible.



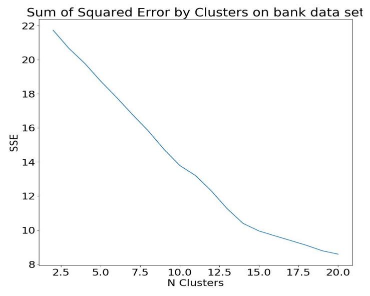
A



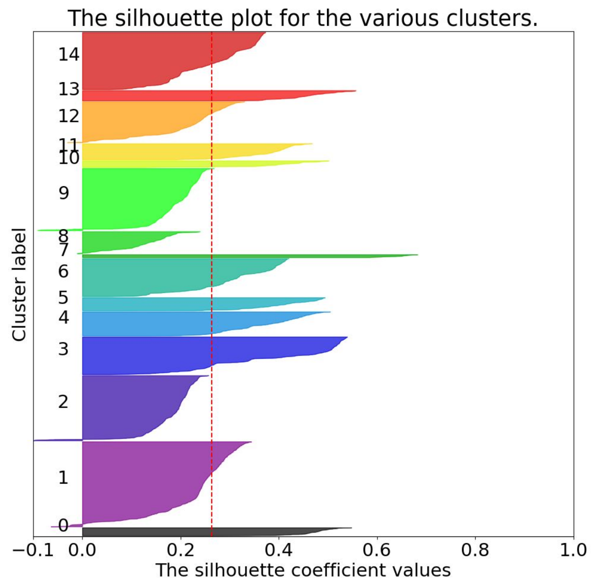
B



C



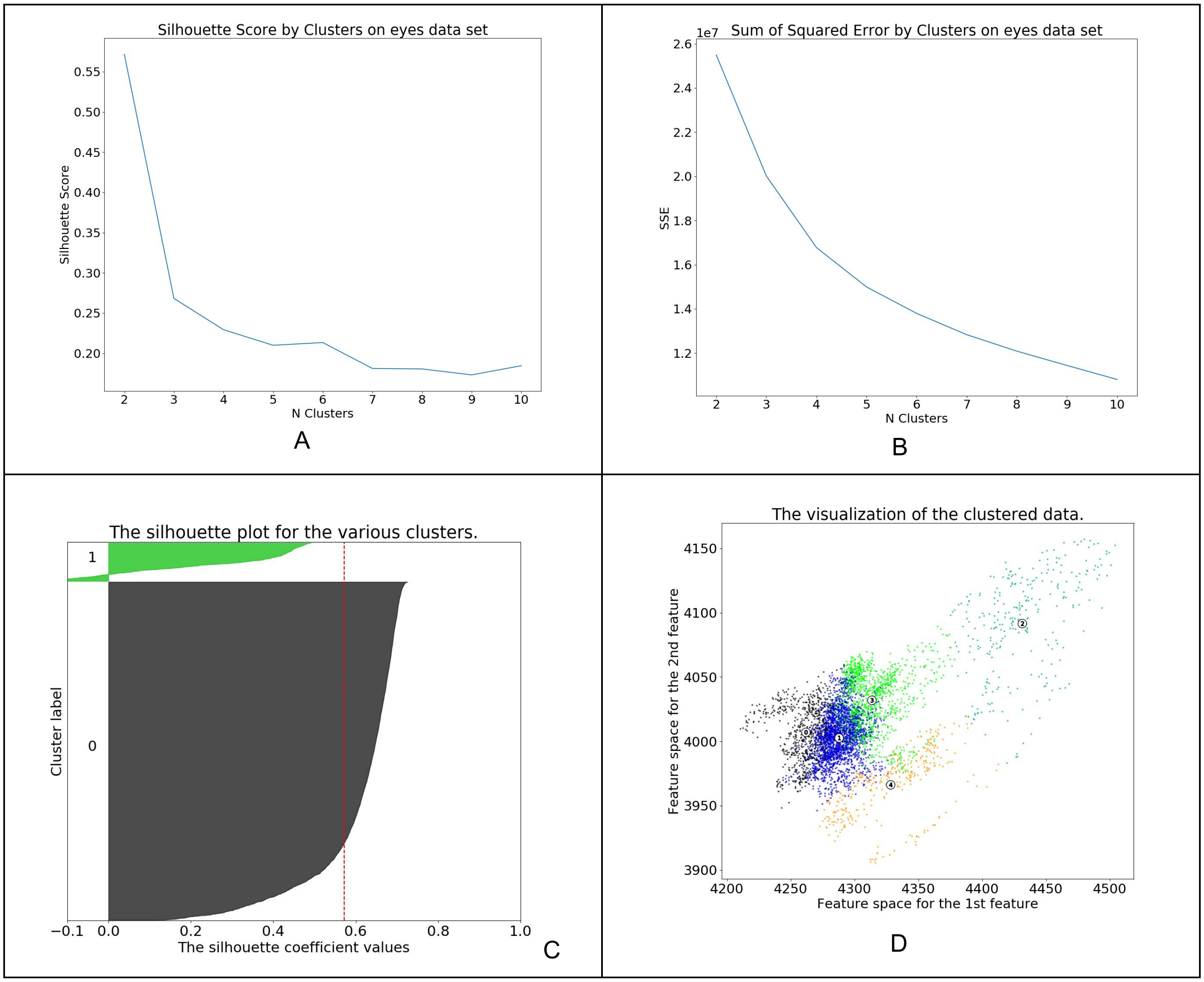
E



D

*(A) Average Silhouette Scores for each number of clusters WITHOUT preprocessing*
*(B) Individual Silhouette Scores for each data point WITHOUT preprocessing*
*(C) Average Silhouette Scores for each number of clusters WITH preprocessing*
*(D) Individual Silhouette Scores for each data point WITH preprocessing*
*(E) Elbow point at 15 clusters for data WITH preprocessing*

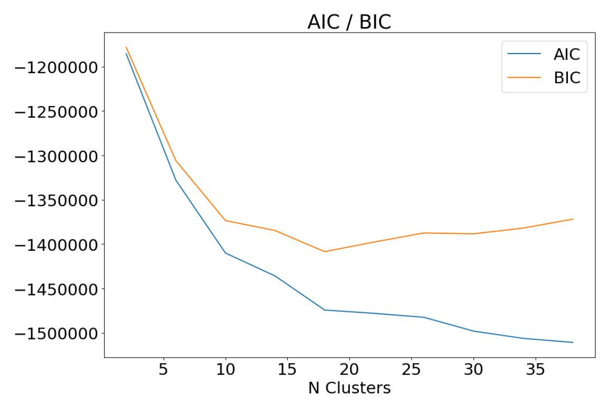
Unlike the Bank data set, the Eyes data set clustering changed not at all with preprocessing because the numbers are all real values on the same absolute scale. The Eyes data set has the highest silhouette score at 2 clusters, even though the clusters are lopsided. Upon reviewing the data, there was no clear attribute on which the data was being split. Looking at the Sum of Squared Error, there is no clear elbow point, although one might say there is an elbow point around 5 clusters. On separate note, the 2D visualization of the clustering on the eye data set is very cool and somewhat representative of what how clustering occurs at higher dimensions since all attributes are similar.



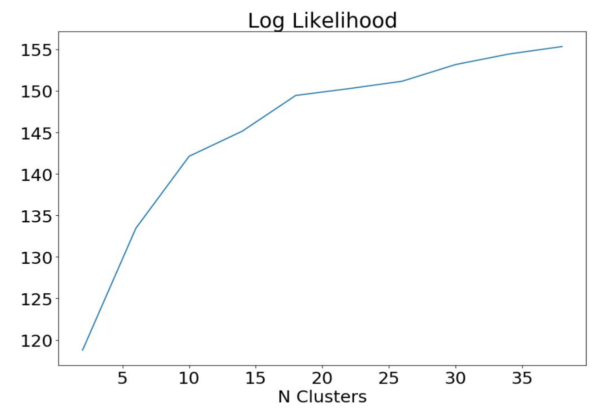
*(A) Silhouette Score by Number of Clusters - best for 2 clusters*
*(B) Sum of Squared Error by Number of Clusters - slight elbow at 5 clusters*
*(C) Individual Silhouette Scores of each data point for 2 clusters*
*(D) 2D visualization of 5 clusters*

*Expectation Maximization*

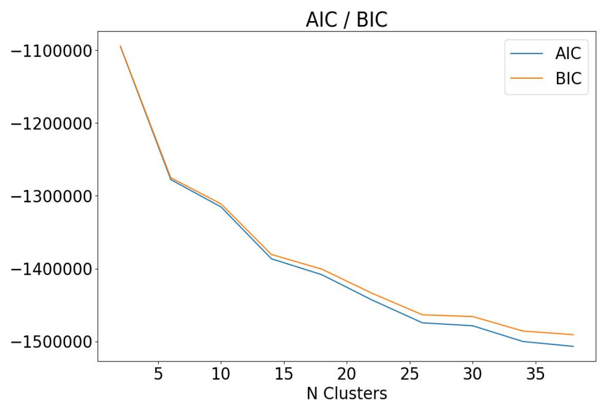
Instead of scoring clusters using Silhouette Score, which is based on distances, we use the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), which are based on likelihood. BIC and AIC both penalize for the number of Gaussian parameters, but BIC penalizes more and is therefore the best metric to acknowledge. Additionally, we look at Log Likelihood of the clustering, which represents the average likelihood that a given data point will belong to its cluster. For AIC and BIC, the lower the number the better. For Log Likelihood, the higher number the better, but the metric is always increasing with more clusters because increasing the number of clusters generally means that each point is closer to its center.



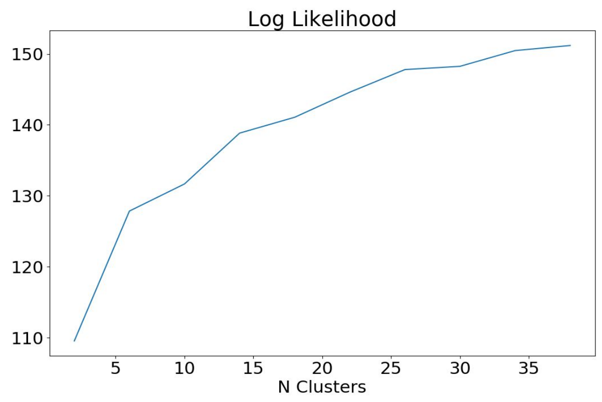
A



B

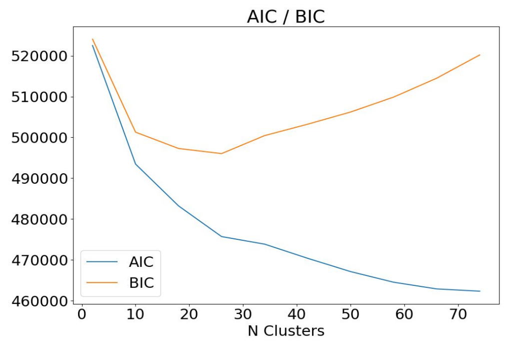


C

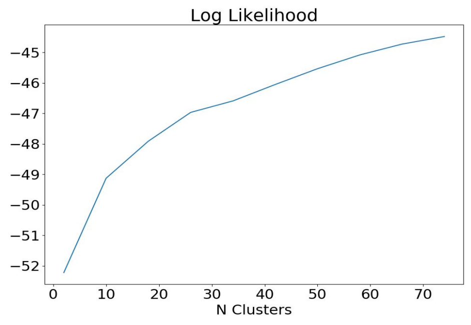


D

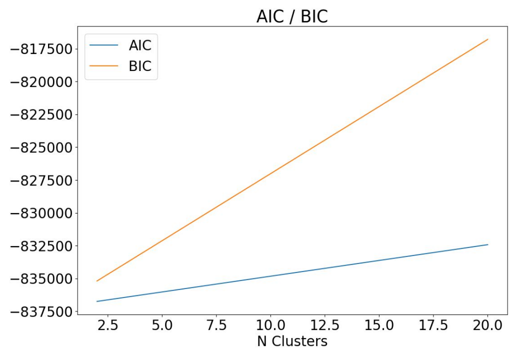
*(A) AIC and BIC by Number of Clusters for Bank data set using Full Covariance*
*(B) Log Likelihood by Number of Clusters for Bank data set using Full Covariance*
*(C) AIC and BIC by Number of Clusters for Bank data set using Diagonal Covariance*
*(D) Log Likelihood by Number of Clusters for Bank data set using Diagonal Covariance*



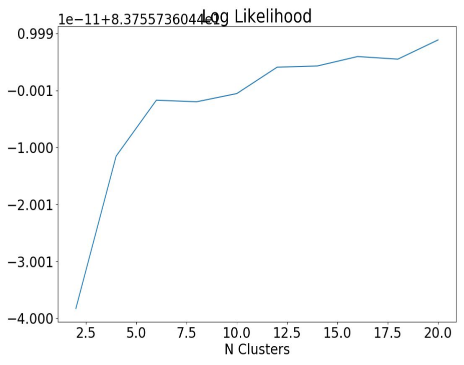
A



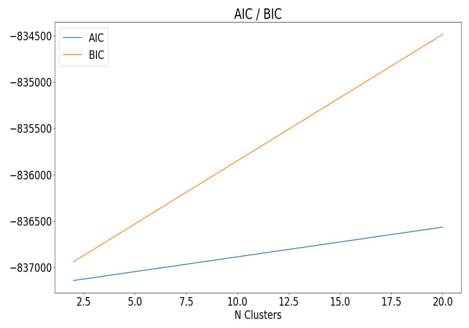
B



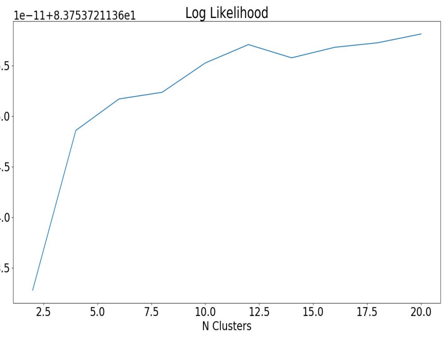
C



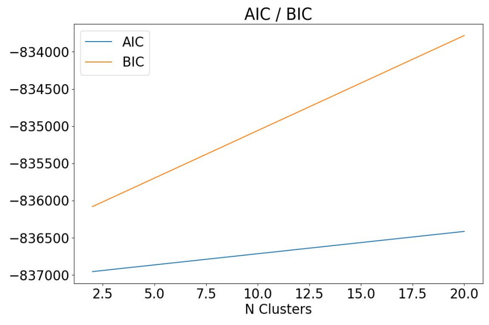
D



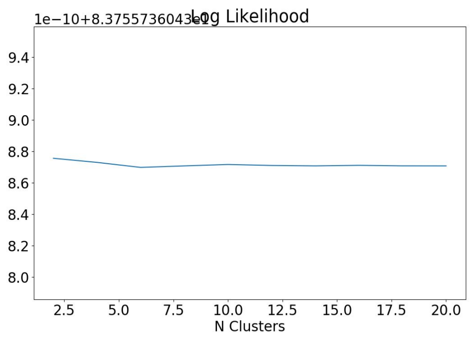
E



F



G



H

*(A) AIC & BIC for Eyes data WITHOUT preprocessing set using Full Covariance*
*(B) Log Likelihood for Eyes data WITHOUT preprocessing set using Full Covariance*
*(C) AIC & BIC for Eyes data WITH preprocessing set using Full Covariance*
*(D) Log Likelihood for Eyes data WITH preprocessing set using Full Covariance*
*(E) AIC & BIC for Eyes data WITH preprocessing set using Spherical Covariance*
*(F) Log Likelihood for Eyes data WITH preprocessing set using Spherical Covariance*
*(G) AIC & BIC for Eyes data WITH preprocessing set using Tied Covariance*
*(H) Log Likelihood for Eyes data WITH preprocessing set using Tied Covariance*

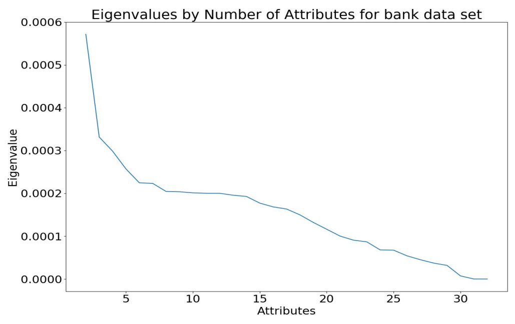
For an explanation of the various covariance types, see
https://stats.stackexchange.com/questions/326671/different-covariance-types-for-gaussian-mixture-models

For the preprocessed Bank data, using the Diagonal Covariance Type reduces the number of Gaussian parameters and therefore results in a matching BIC and AIC. The BIC for the Full Covariance Type offers an optimal number of clusters around 18.

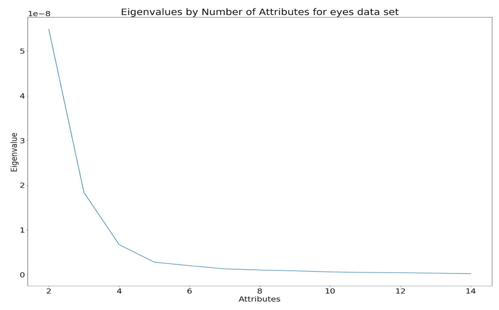
The Eyes data set scored very differently depending on whether or not the values were normalized, indicating that the clustering was not scale invariant. After normalizing the data, 2 clusters always has the best AIC and BIC scores, and different Full and Spherical Covariance types both show an elbow in the Log Likelihood curves around 5 clusters. These findings are reminiscent of the KMeans Silhouette Score and Sum of Squared Error, respectively. The Tied Covariance Type has a strange Log Likelihood curve, where the value is initially very high for 2 clusters and does not change with increasing numbers of clusters. Presumably the initial 2 clusters don't change much and the additional clusters don't contain many points.

## Feature Reduction

*PCA*



A

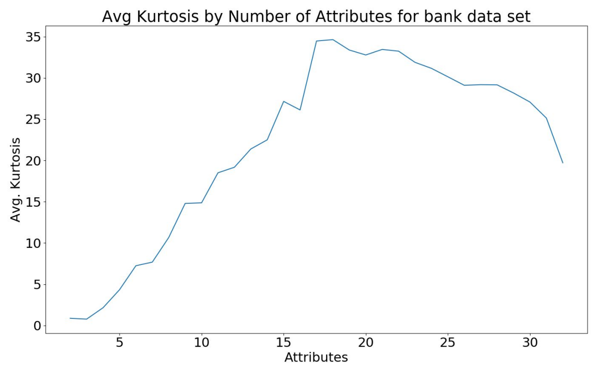


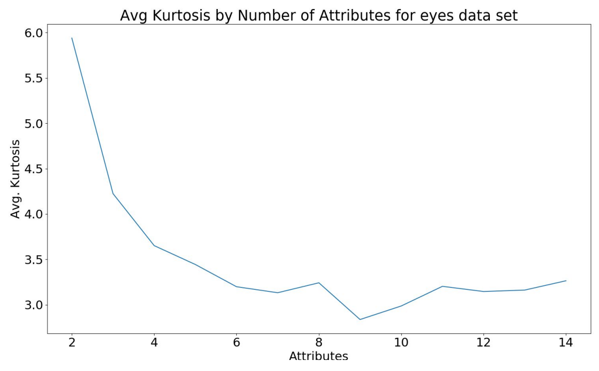
B

*(A) Eigenvalues by Attributes for Bank data*
*(B) Eigenvalues by Attributes for Eyes data*

For PCA, the vast majority of variance in Eyes data is captured in the first 5 axes, whereas every additional axis for the Bank data provides significantly more information. The reason for the must be independence of the Bank data attributes versus the relatedness of the Eyes data attributes. We see these results because the Bank data attributes are strongly relevant and the Eyes data attributes are weakly relevant.

*ICA*





*(A) Average Absolute Value of Kurtosis by Number of Attributes for Bank data*
*(B) Average Absolute Value of Kurtosis by Number of Attributes for Eyes data*

The goal with ICA is to have a high absolute value of Kurtosis for the resulting attributes. As such, ICA is most kurtotic with 17 features for the Bank data and 2 features for the Eyes data. In other words, ICA finds many independent attributes (or axes) for the Bank data and very few for the Eyes data. These findings are consistent with the results from PCA: both demonstrate lots of independence (or strong relevance) in the Bank data attributes and lots of similarity (or weak relevance) in the Eyes data attributes.

*Random*

In theory, random attribute filtering affects the Bank data much more than the Eyes data because, since the Eyes data attributes contain similar information, randomly picking a few won't harm the data integrity, whereas each attribute of the Bank data that is filtered is likely to contain information that isn't represented at all by any of the other attributes. For example, "Marital Status" and "Has Defaulted" are loosely related at best.

Repeating the experiments we have run so far on the randomly filtered data will give us insight into the information lost by the filter. For the Bank data, random filtering should decrease the optimal number of clusters, but random filtering on the Eyes data should affect the clustering minimally.

*Trees*

In the tree algorithm used, a given number of trees are used to represent each data row. The data is learned by a random forest with the given number of trees, and then every leaf in the forest is One Hot Encoded into an attribute such that each row has a 1 for every leaf that it belongs to. This mapping results in a high number of attributes, each with very few (2) possible values.

*Results*

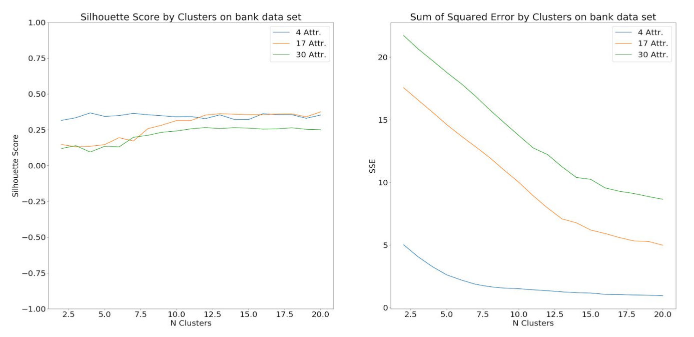
Below are results for each kind of Feature Transformation on each data set, then clustered by KMeans and Expectation Maximization algorithms. For each Transform -> Cluster combination, a few lines are shown for varying numbers of final attributes that are used for clustering.

In general, increasing numbers of clusters improve scores less for data sets with reduced features. This is because fewer clusters are needed to group less information. Additionally, feature transformation often improves the information density of features, making clustering more effective on smaller numbers of features.

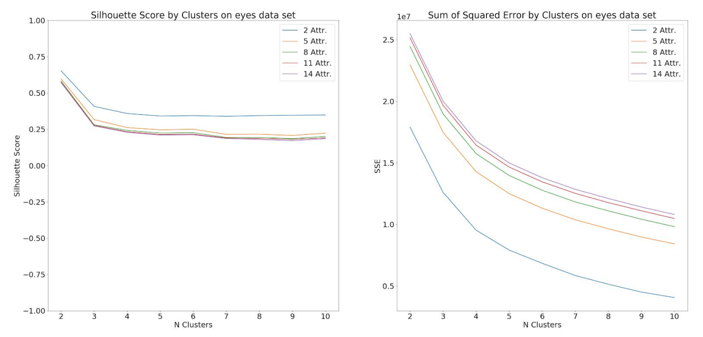
Still, the clustering metrics are largely similar to the clustering values obtained without using feature transformation (although in many place the graph scales are different). For similar numbers of features, the absolute values are close, and the trends are similar for fewer features, albeit with diminished curves.

Many of the graphs are stratified due to the difference in number of attributes. Sum of Squared Error, for instance, increased with more attributes simply because new distance dimensions are introduced.

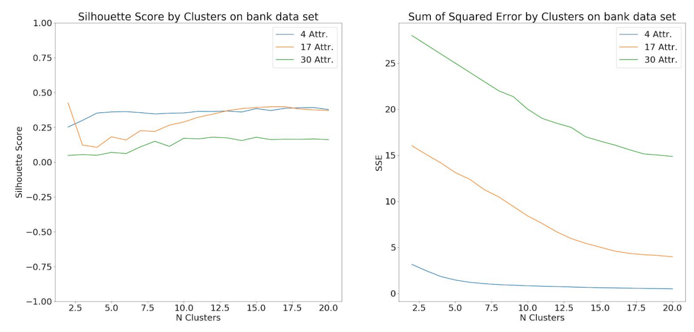
*KMeans*



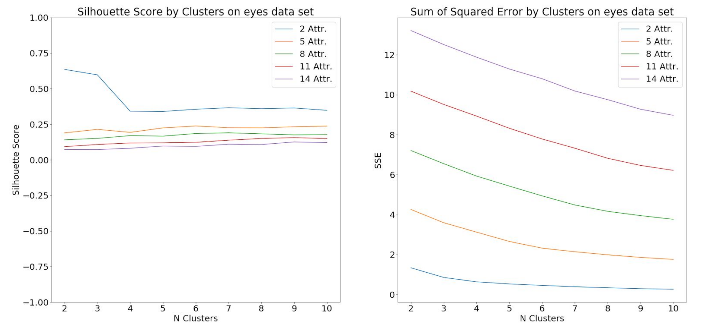
Bank, PCA



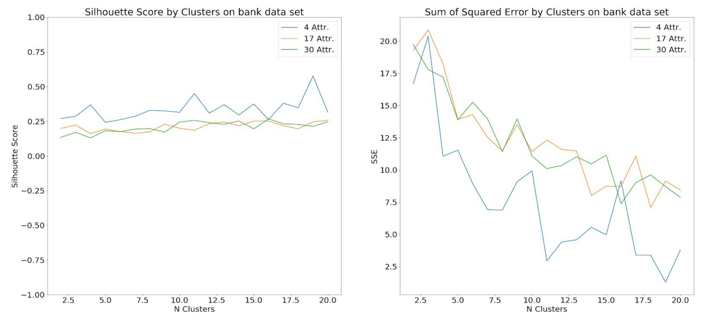
Eyes, PCA



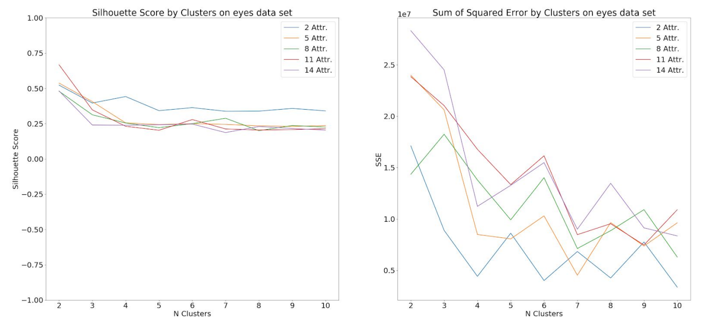
Bank, ICA



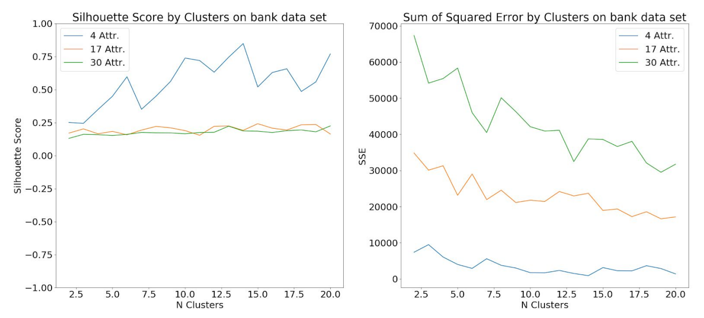
Eyes, ICA



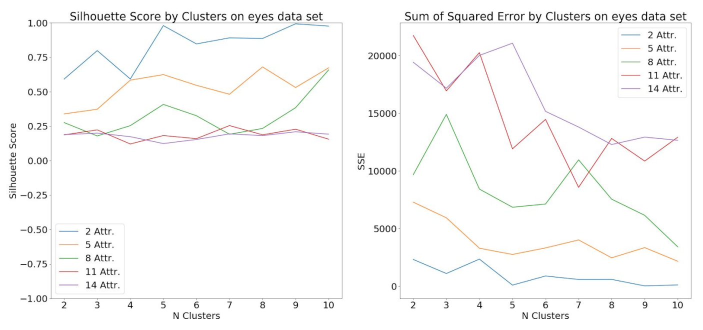
Bank, Random



Eyes, Random

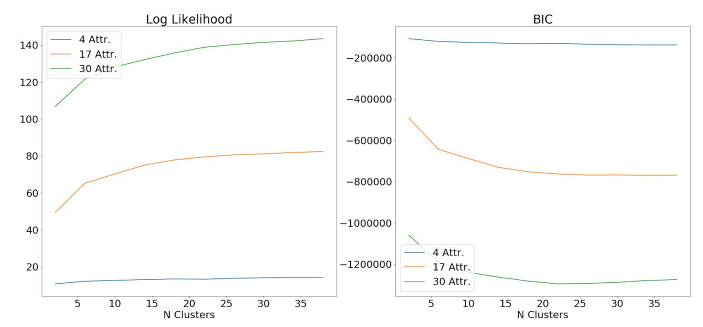


Bank, Tree

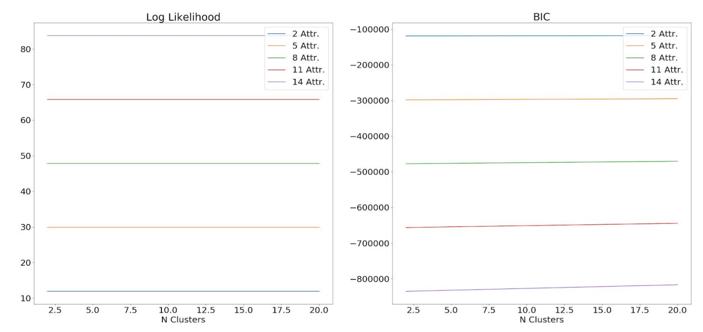


Eyes, Tree

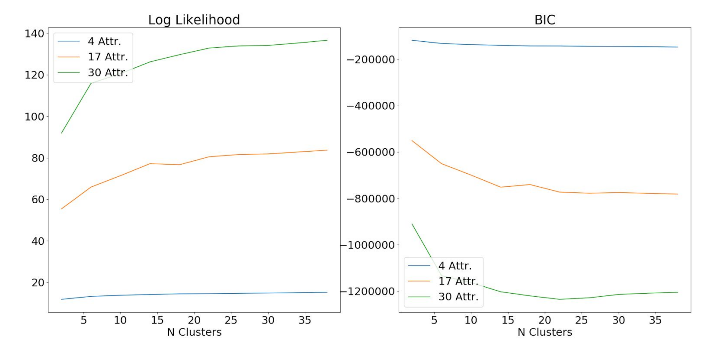
## *Expectation Maximization*



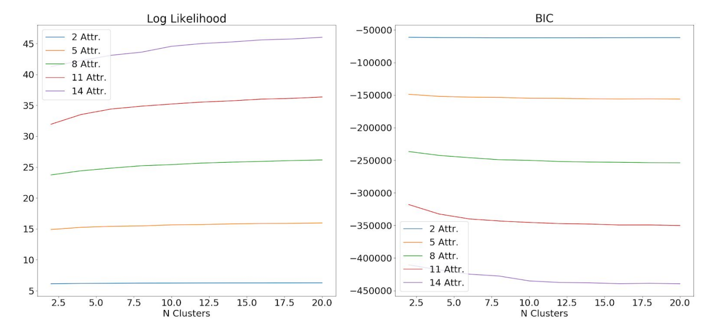
Bank, PCA



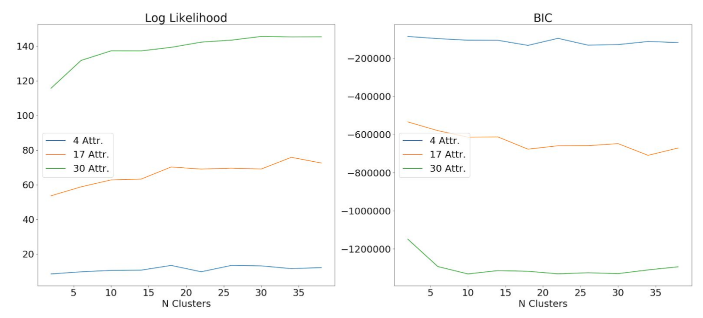
Eyes, PCA



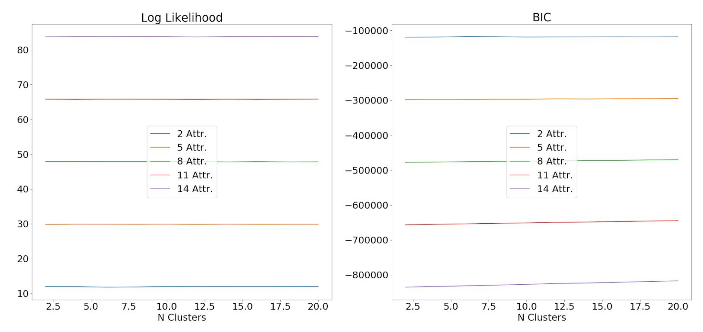
Bank, ICA



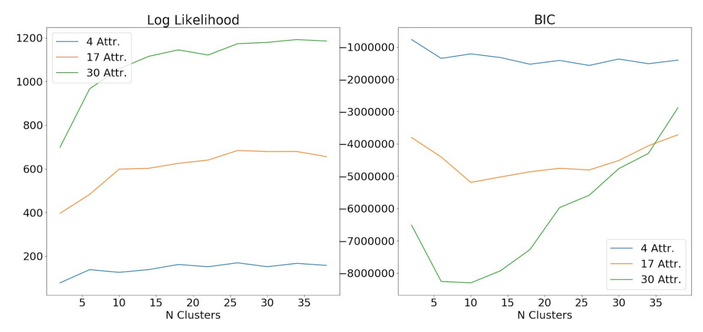
Eyes, ICA



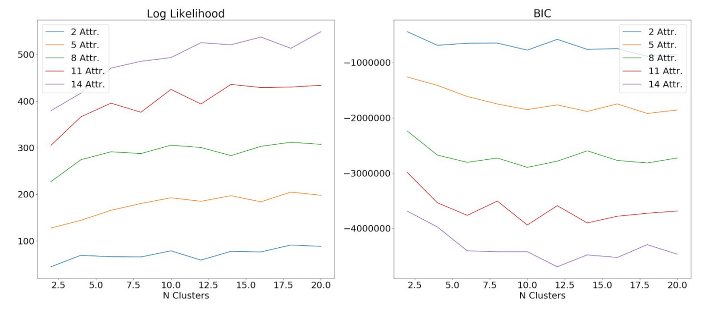
Bank, Random



Eyes, Random

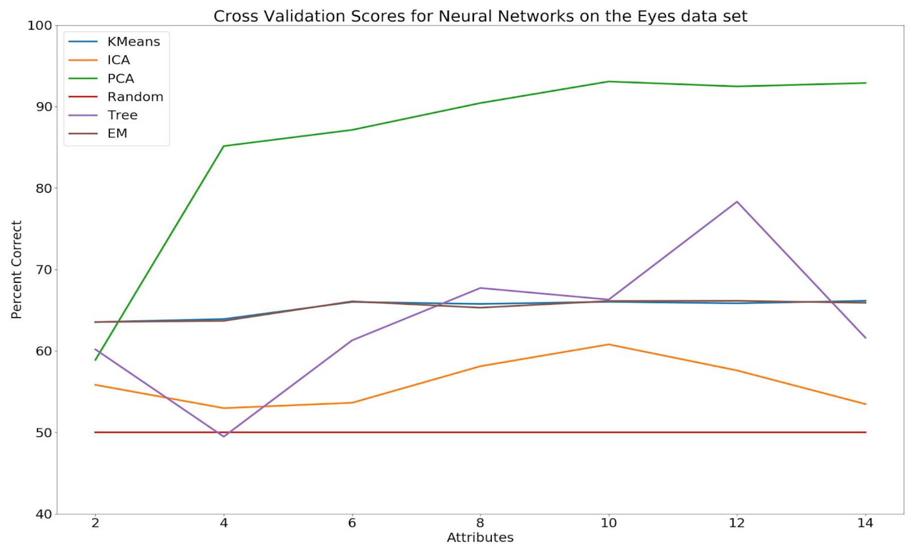


Bank, Tree



Eyes, Tree

## Neural Networks



In the above graph, PCA takes the cake; followed by the sporadic Tree Reducer; followed by the nearly identical KMeans and Estimation Maximization Cluster Reducer. followed by ICA; followed by the Random Reducer performing exactly as well as random guessing.

The Neural Network from Project 1 that trained on the same data set never performed better than random guessing (50%), and this was confirmed at the start of this experiment - classification without feature reduction scores 50% every time, regardless of the hyperparameters. As such, randomized feature selection doesn't perform the model performance at all.

PCA improves the model a staggering amount, taking it from 50% to 90%, implying that the Eyes data set has a few axes that capture a high percentage of the variance very well. This result is consistent with the PCA feature reduction test we performed earlier.

By contrast, ICA doesn't perform particularly well, although it does perform better than Random, showing that some independent features may be determined from the original data set.

The KMeans and EM Reducers correspond to an algorithm that transforms features by using clustering to create new features. For each, a new feature set is constructed following this procedure: for each feature in the set, remove it and score the clustering resulting from the remaining N-1 features; whichever clustering scored the best, create a new attribute where each row's value is the cluster it belongs to from the best clustering; without adding back the attribute that generated the best clustering, repeat this process until the specified number of attributes is attained. Using KMeans with Silhouette and Expectation Maximization with BIC yielded effectively identical results, which is most likely due to the fact that only removing a few attributes before performing clustering results in similar clusters. Still, the learners consistently performed right around 65%, indicating that the clusters did provide useful information to the classifier.

Finally, the Tree Reduction learner, which increases the number of attributes through a One Hot Encoding, performs inconsistently. This reduction algorithm simply provides a totally different way to represent the data, and its performance doesn't seem correlated with the number of attributes.