Alan Li
Project 2
CS 4641

Unsupervised Learning and Dimensionality Reduction Analysis

Datasets

The datasets I used were the ones I used for project 1. As a quick recap:

Bank: there are about 45,000 samples of data taken from a Portuguese bank. There are 16 attributes relating to the client's demographics, the nature of contact, and marketing campaign information, among other things. Each sample has a label—1 or 0—indicating whether that customer eventually subscribed to a term deposit with the bank. More details about the distribution of the data can be found in my project 1 analysis. This dataset is interesting with regards to clustering because the problem is business related. Revealing the underlying structure of the dataset could give insight on what kinds of people are likely to submit to a term deposit. Something else I mentioned in my analysis from project 1 was that there are lots of attributes, thus making the problem hard to perform well on. In project 1 I manually removed several attributes I thought were unnecessary to address the curse of dimensionality. This move made all my learners perform consistently worse. An interesting task for this project is to see whether I can successfully reduce the number of attributes in the dataset without sacrificing performance (or to improve performance).

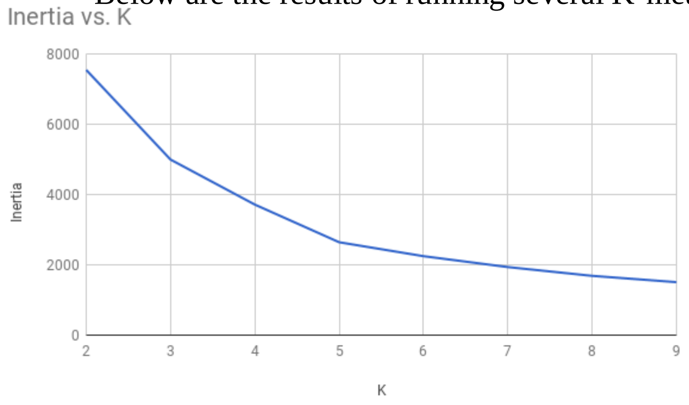
Wine: there are about 5000 samples of white wines scored from 0 to 10. Each sample has 12 attributes related to the chemical and physical properties of the wine. These attributes are quantitative. It is important to recall that while there are technically 11 different classification classes, the data only contains 6 of these classes. This dataset is interesting in the context of clustering because I believe that creating sensible clusters will be difficult because of the domain of wine tasting. My reason for this is because wines that are extremely similar chemically may have very different scores due to the intricacies of wine tasting.

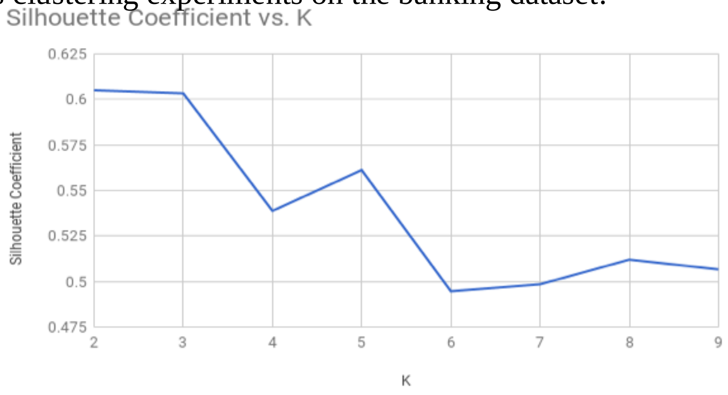
Step 1: Clustering

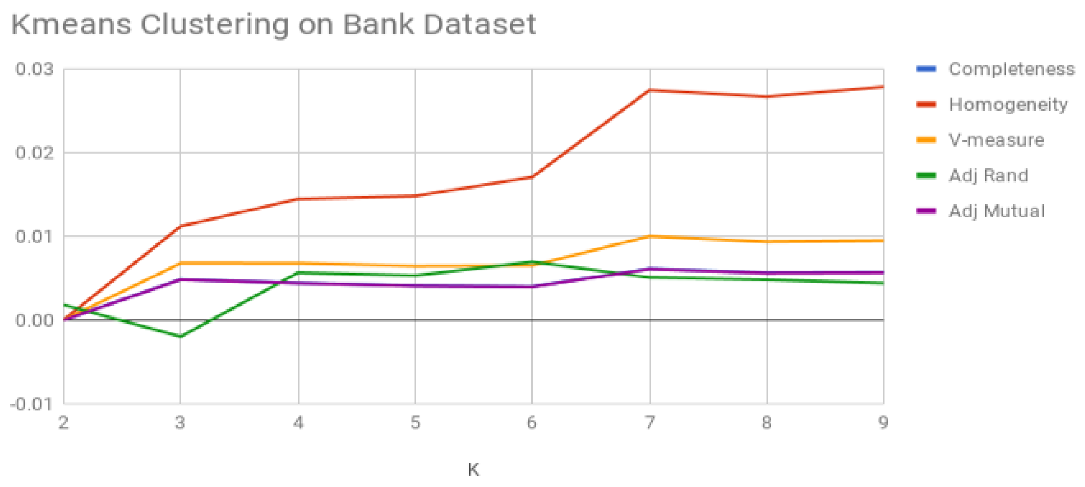
Method: To find the best value of K, I ran multiple clustering experiments and recorded various cluster evaluation metrics. I iterated from 2 to 9—2 being the lower bound because that is the minimum number of categories in either dataset and 9 being the max because I wanted to see what would happened if I forced a clustering where k > # categories.

K Means:

Below are the results of running several K-means clustering experiments on the banking dataset:





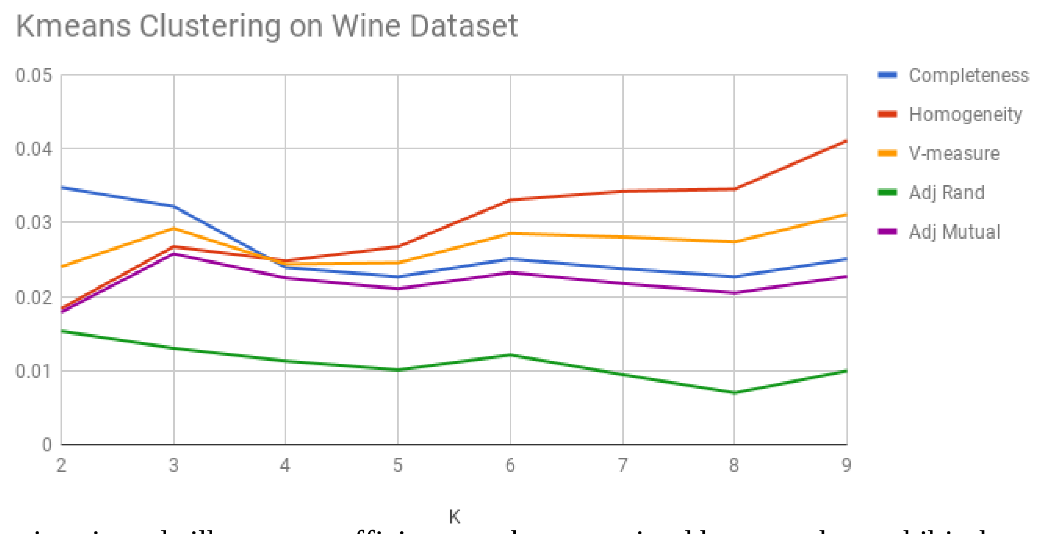


From the first graph, we see the inertia of the clusters decreasing as K increases. This makes perfect sense because inertia is the sum of squared distances of samples to their closest cluster center. As the number of centers increases, the number of samples each cluster must account for decreases. In fact, if we set K to the number of samples, we can expect the inertia to be 0. The silhouette coefficient is calculated as $(b - a)/\max(a, b)$, where b is the distance between a sample and the nearest cluster that the sample is not a part of, and a is the intra-cluster distance. Having a silhouette coefficient of 1.0 is desirable, because it suggests the clusters are far apart, and that each cluster has its samples close to the center. We see that the silhouette coefficient is the highest at K = 2, and slowly decreases as K increases. This suggests that in the banking dataset, having K=2 or 3 gives the clearest clusters, which makes sense because there are only two classification categories.

However, from the third graph, we see that for all the values of K used in the experiment, the clusters performed very poorly according to several metrics. The completeness line is impossible to see in this graph because all the values are very close to 0. Having a completeness of 1.0 indicates that all samples with the same label will be in the same cluster. The fact that the number of clusters does not seem to affect completeness suggests that the dataset of customers who subscribed to a term deposit is very diverse. In other words, it is impossible to use numerical distance to create the "perfect" cluster that captures all positive instances in this dataset. Homogeneity describes how similar the samples in each cluster are. As the number of clusters increases, homogeneity increases slightly. My guess is that the K-means algorithm is finding "neighborhoods" in which samples are the same. For example, maybe young widowed college dropouts are very likely to subscribe to a term deposit. As K increases, it is more likely that this specific "neighborhood" could receive its own cluster. Adjusted random score and adjusted mutual information score both relate to how similar the clusters are. The value of K does not seem to affect these scores much, which suggests that the clusters remain distinct (little overlapping).

Overall, the clusters did not seem to line up with the labelings, nor did they otherwise line up naturally. This is probably because of the way I handled my attributes, as well as the distance function I used, which is just Euclidean distance. To encode strings, I simply set each string category to 0, 1, 2, etc. Doing this means that the difference between "divorced" and "married" is 1, but the difference between "married" and "single" is 2, whatever that means. The Euclidean distance metric also does not make sense for some of the attributes in this problem. For example, the distance between the ages 20 and 25 is the same as the distance between 60 and 65; however, we know that age differences cannot be judged solely on the numerical difference, as there are certain stages of life that dictate your circumstances more accurately than just age. I also predicted that some attributes, such as duration of contact in seconds, would dominate the distance metric because the attribute would exhibit significantly higher values than the categorical attributes. To combat this problem, I normalized the data. The results were about the same. To improve the performance clustering, having domain knowledge would be helpful because it would allow us to use better distance metrics. In addition, reducing the number of attributes in the samples could possibly remove some noise.

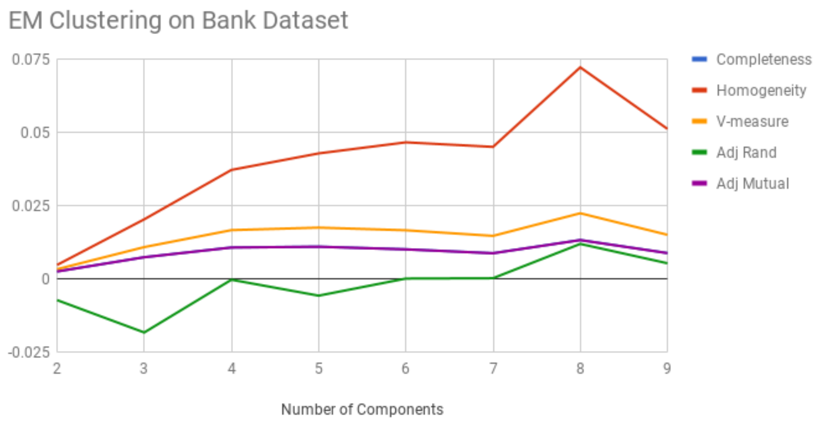
Below are the results for k-means on the wine dataset.



The inertia and silhouette coefficient graphs are omitted because they exhibit the same patterns as the banking dataset graphs. This makes sense because both inertia and the silhouette coefficient don’t depend on the number of classification classes or the predicted vs. actual labelling. This also makes sense because the K-means algorithm is designed to minimize the sum of squared distances to each cluster, which explains the lowering inertia. From the graph, we see the metric scores are still very low, but they seem to do better at K=6 and K=9. Since there are 6 classes represented in the dataset, it would make sense that the clustering performs better at K = 6. It’s important to note that the while both clustering algorithms performed poorly according to the metrics, the scores were slightly better for the wine dataset than the banking dataset. This may be because there are fewer attributes in the wine dataset than the banking dataset, and each attribute is already quantitative, meaning there is no loss of meaning by converting categorical attributes to numerical ones. However, even with numerical attributes, using Euclidean distance may not be the best metric to use. For example, one of the attributes the pH of the wine. Since pH uses a logarithmic scale, a substance with a pH of 8 is 10x more basic than a wine with a pH of 7, and 100x more basic than a wine with a pH of 6. By linearizing what is supposed to be a logarithmic metric, the clustering algorithm likely severely underestimated the differences in pH across different wines. The K-means algorithm also treats the distances of each attribute the same. For example, it treats a difference of 1 in pH the same as a difference of 1 in residual sugar Clearly, these differences do not carry the same meaning, but under a Euclidean distance metric, they are treated as such. Again, having domain knowledge would allow us to choose a better metric, or having fewer attributes to deal with would minimize the shortcomings of the K-means algorithm.

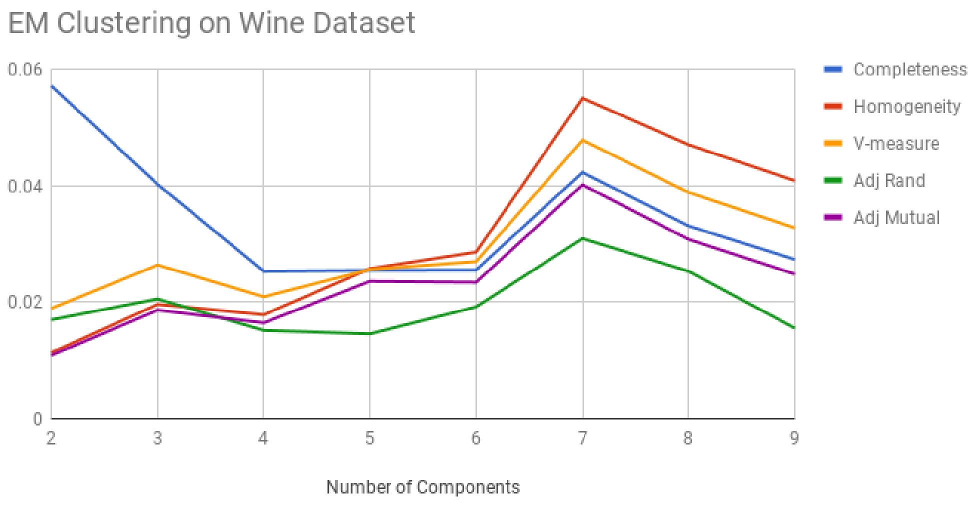
Expectation Maximization

Below are the results for EM on the banking dataset:



Expectation maximization does slightly better on the metrics than k-means, but still maintains a low score overall. This further supports the idea that the current data cannot be clustered in a way that separates the labels clearly. We also note that the adjusted random score for EM is negative for many values of K. This is not supposed to be common, and it suggests that the agreement between clusters is worse than random chance. There are two reasons this may be: 1) we got incredibly unlucky, and 2) the algorithm created clusters that have samples placed worse than chance. Getting the same results after running the algorithm multiple times leads me to think the second reason. An interesting thing to note is that the ARI scores are negative only after normalizing the data. My hypothesis for the negative ARI scores is that normalizing the data somehow mislead the Euclidean distance metric into creating bad clusters. In other words, it is possible that taking the values of the attributes at face value was more helpful than scaling them down. What that means in the context of Euclidean distance is that either attributes with large values (like contact duration) are very important, or attributes with very small values are irrelevant.

Below are the results for the wine dataset:



Here we can see that expectation maximization performs the best at K = 7, which is close to the number of classes in the dataset. We also note that overall, EM produces higher scores than K-means. One possible reason for the better performance overall is that K-means assumes a Gaussian distribution (centroids), whereas EM can work with other distributions as well.

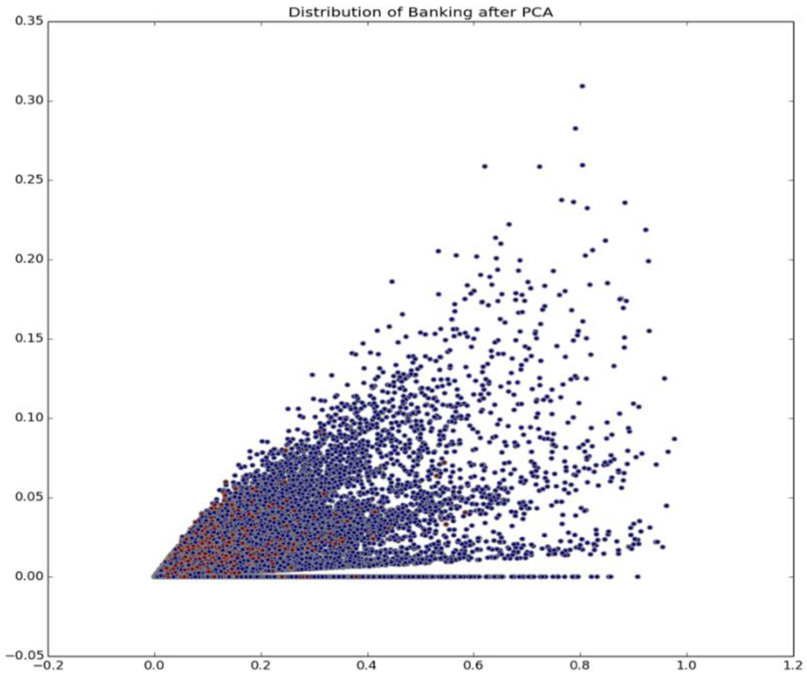
Both K-means and EM produced poor results according to the metrics for both datasets. Again, this is probably due to the nature of the data and distance metric; it is not possible to separate labels clearly based on Euclidean distance alone.

Step 2: Dimensionality Reduction

PCA:

I ran PCA on the banking dataset, before and after normalizing the data. Below are the tables for the sorted ratios of explained variance on the banking dataset:

Normalized	Raw
0.7459130922	0.9913380508
0.1474770441	0.007089561408
0.06943805648	0.001069662215
0.03187288908	1.19E-05
0.002802068088	7.31E-06
0.0008157653153	1.20E-06
0.0004217988941	1.04E-06
0.0003688261817	8.47E-07
2.71E-05	4.05E-07
1.62E-05	5.88E-08

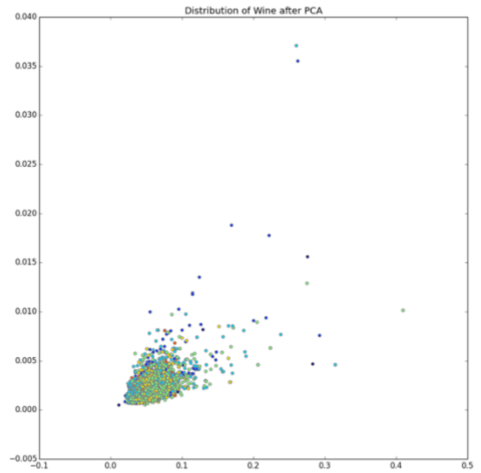


The values of the explained variance for the raw data suggests that there is one attribute that dominates the variance of the data, which suggests that the data is highly separable along one vector. After normalizing the data, we see that the variance can be mostly explained with 2 or 3 vectors.

Above is the plot of the data projected onto 2 dimensions. We can see that the 'yes' labels tend to be closer to the origin. However, we note that there are lots of 'no' labels near the origin as well, so it may be the case that projecting onto more dimensions could separate those samples.

Next I ran PCA on the wine dataset:

Normalized	Raw
0.6679916902	0.9090487394
0.2065112774	0.07928978822
0.1094867254	0.01014881713
0.007923926946	0.0005057144203



0.006792676958	0.0003230211935
0.0005429966604	8.60E-06
9.62E-05	6.67E-06

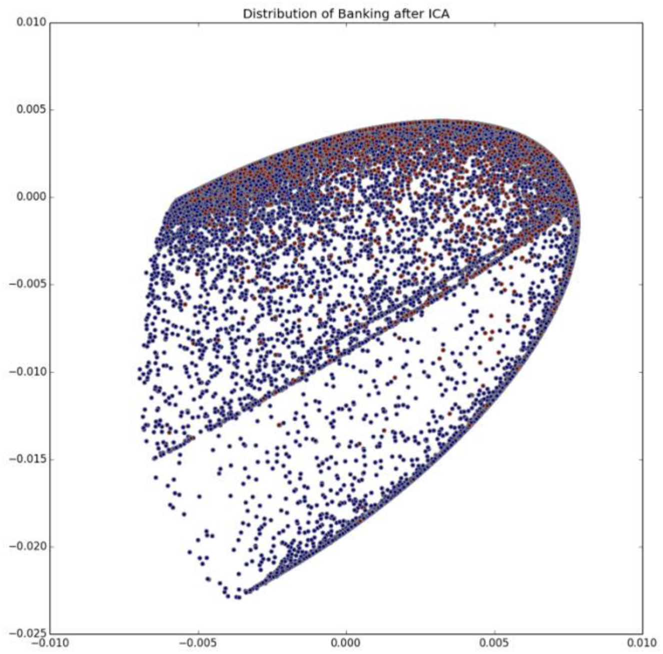
Again, we see that in the wine dataset, there is one attribute that explains much of the variance. Perhaps it is not so much that the attribute is important, but rather the fact that the attribute takes on large values such that its variance counts for more in a sample. After normalizing the data, we see that it takes about 2 or 3 vectors to explain most of the variance.

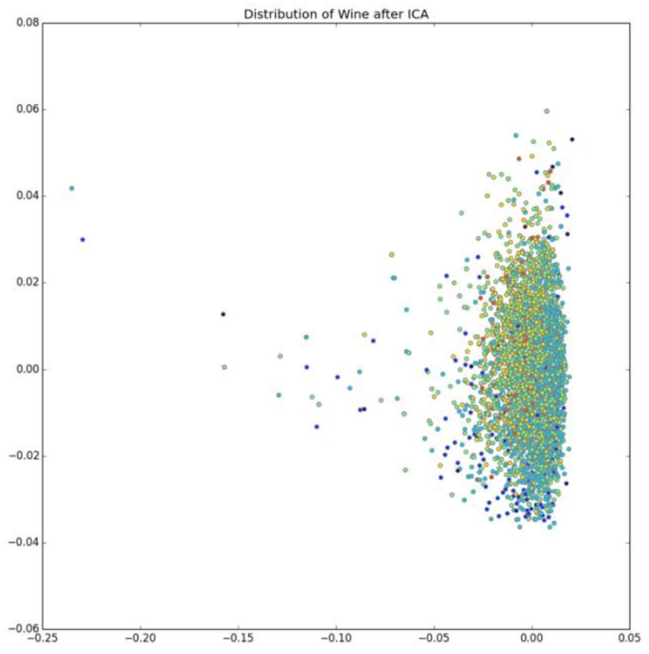
As expected, the complexities of wine tasting prevented PCA from creating a projection that distinguished labels. This is probably because you cannot distinguish labels effectively because even wines that are similar chemically may have drastically different quality.

ICA:

Below are the tables for the kurtosis of each attribute for both datasets after normalizing:

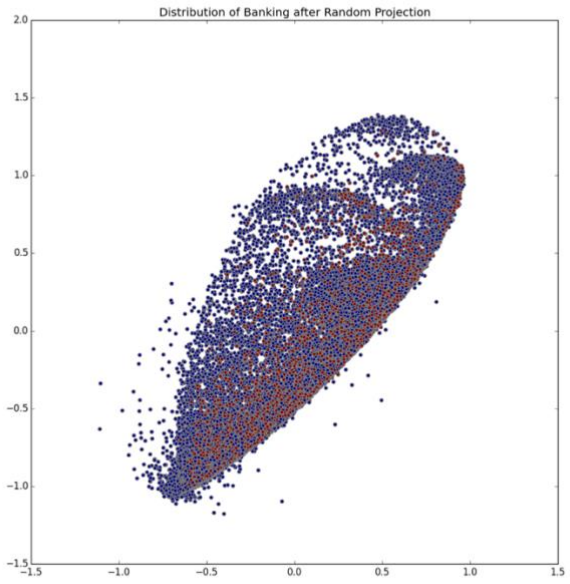
Bank	Wine
9.308749501	42.50490589
26.59821798	162.2504628
28.0557588	27.64856792
25.75143434	6.593253233
295.0727055	37.04577797
1.217285455	0.07354459338
21.26454272	38.6565078
48.35195611	38.21636047
31.68088199	39.12435199
18.15673787	16.92759962
24.37712343	43.36342898
-1.354739579	
193.2809481	
10.50390605	
1446.626819	
11.82832491	

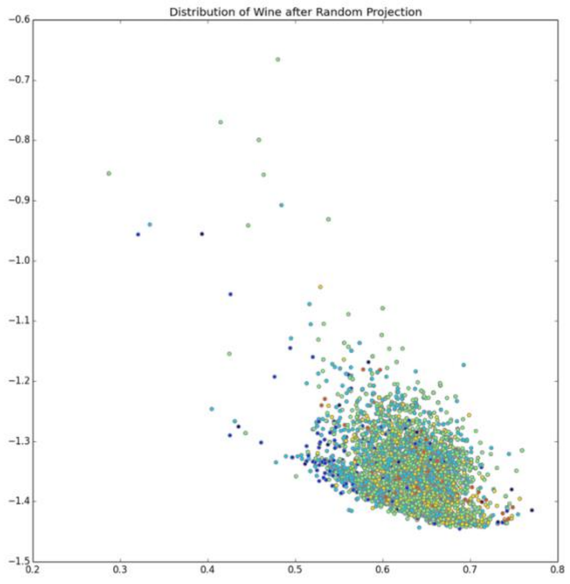




ICA makes two assumptions when trying to separate components: 1) the components are independent and 2) each component is non-Gaussian. From the table above, we see that almost all the attributes are have non-Gaussian kurtoses. As a result, ICA seems to do well in separating components for the banking dataset; we can see that there seems to be a clear line separating regions of red and blue. However, even though the wine attributes are non-Gaussian, ICA still cannot find meaningful projections. This further supports the idea that the wine problem is unique in that extremely similar wines have drastically different qualities, which makes it hard to separate wines based on the given attributes.

Randomized Projection

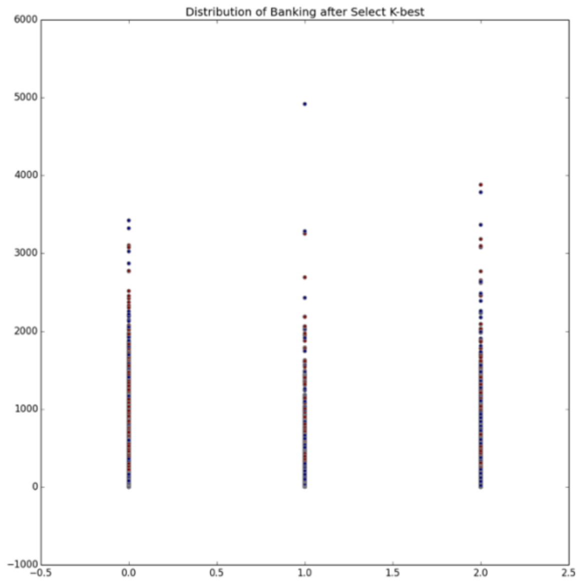


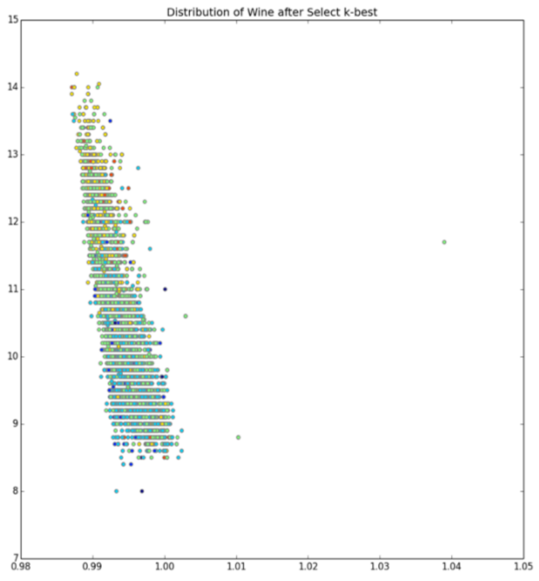


Randomized projection does not try to do anything special with its data. It just projects data onto different dimensions with randomly initialized matrices. It makes sense that both the scatters above do not seem to produce anything of value.

Select K-best:

The feature selection algorithm I chose was select k-best, which reduces the dimensionality of the data by simply scoring each attribute and taking the k best. I chose this algorithm because it seems like a naive approach that induces info loss. It should be a good baseline to test if the other algorithms can perform better than select k-best.

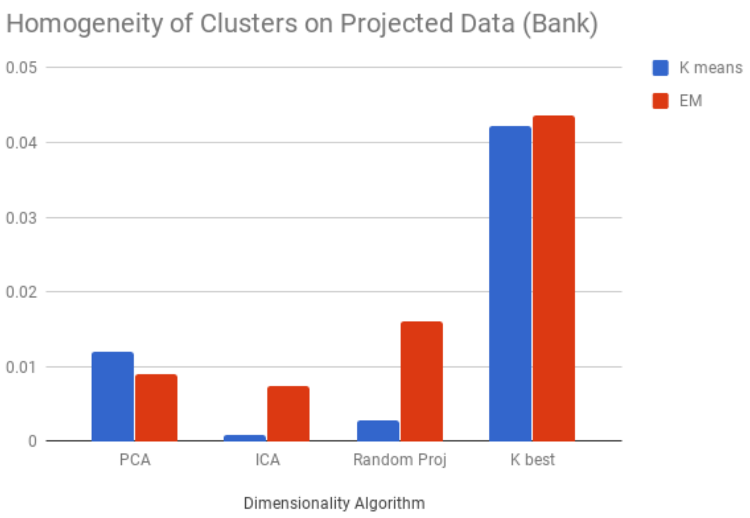


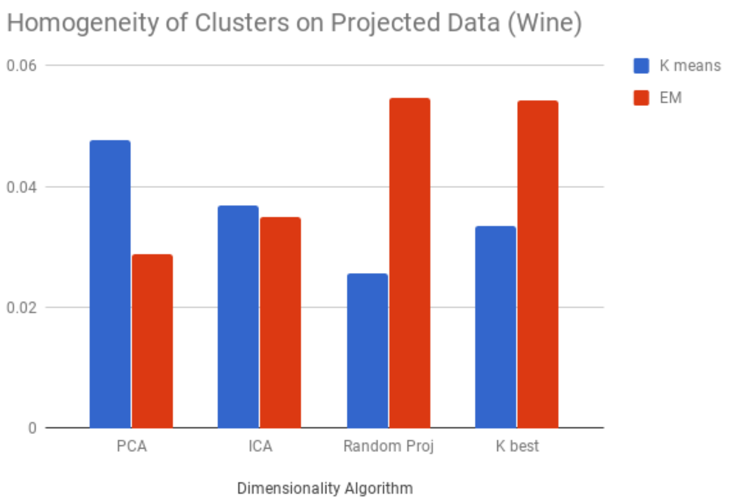


Above are the graphs for the projected data after choosing k-best. The two attributes chosen for the banking dataset were “type of contact” and “duration of contact”, with duration scoring 8 times higher than the second-best attribute. The duration of contact does seem to make sense, as a customer who talks for a long time is more likely interested in a term deposit and is asking for more details. However, the scatter does not give any useful information, which implies that the problem cannot be solved without most or all its attributes. This supports my project 1 conclusion. The two attributes chosen for wine were “density” and “alcohol.” While I’m not a wine expert, I would expect these two attributes make sense when determining the quality. We can see a general trend in the wine scatter, with yellow labels polluting the top and teal labels moving towards the bottom.

Step 3: Clustering Projected Data

Due to space limitations, I am repeating the clustering experiments only for k=(number of labels) + 1 on all data that has been projected to 3 dimensions. I chose 3 for the dimensions because it seemed that PCA only needed 3 vectors to explain almost all the variance, and (#labels) + 1 clusters because is close to the number of categories while allowing enough clusters to represent more complicated distributions.





Above are the homogeneity scores after running both clustering algorithms on all the projected data. Surprisingly, for the banking dataset, my K-best algorithm created the most homogeneous clusters with a score almost 4x the next best. I think this is because K-best chooses the attributes that predict labels the best, which is why the clusters seem to line up more with the labels. However, K-best did not perform the best on the wine dataset. My idea is that since there are 6 classification labels, only using 3 attributes is not sufficient to distinguish them. There doesn’t seem to be a clear winner between k-means and EM, which is contrary to the results from the non-projected data. Perhaps projecting the data into lower dimensions led to the datasets to become more Gaussian, which would help K-means perform better.

Clustering on the projected data did not seem to change the results on the clusters. One idea for why this is the case is because most feature selection algorithms try not to lose information when projecting. If I had more time, I would experiment with different values of K, perhaps looking at what happens if I set it to a drastic value.

Step 4: Neural Net

Raw Data

Algorithm	Train Time	Train Score	Test Time	Test Score
None	20.74347	0.88210	0.05364	0.88550
PCA	21.04673	0.88128	0.02331	0.88388
ICA	6.16358	0.89082	0.02368	0.88543
Random Proj	1.01466	0.83754	0.02430	0.88093
K Best	1.89864	0.88889	0.02300	0.890297

Normalized Data

Algorithm	Train Time	Train Score	Test Time	Test Score
None	11.47714	0.89063	0.02553	0.89015
PCA	5.64241	0.88460	0.02388	0.88668
ICA	5.56223	0.88314	0.02870	0.88270
Random Proj	10.0900	0.88039	0.01844	0.88137
K Best	9.31225	0.88889	0.02300	0.88911

Above are the tables of my neural net from project 1. As a reminder, the training scores are the average of 5-fold cross validation, and the neural net is composed of 3 layers. I ran the neural net after each feature selection algorithm projected the data onto 3 dimensions. I ran the experiment both before and after normalizing the data. The data suggests that normalizing the data does not significantly change the results of the test accuracy for any algorithm. In the unnormalized data, we see that the K best algorithm performed the best, whereas all the algorithms performed pretty much equal to the unnormalized data. My theory as to why K best performed better is because there may be noisy attributes after all. In project 1, I stated that I did not have the domain knowledge to prune irrelevant attributes, and as a result, my pruned dataset performed considerably worse. It seems that the K best algorithm could determine the most important attributes and remove some of the noise from the dataset. Even though the other feature selection algorithms did not improve accuracy, they were, except for PCA, much faster in terms of both training and testing time. This is expected because training a neural net with 3 attributes is going to be faster than training one with 16.

The normalized data results are not as impressive. None of the algorithms could surpass the default experiment, but again, the training times were much faster. It is also important to note that while K best did not beat the default experiment, it still surpassed the other feature selection algorithms. It seems that for this problem, projecting onto lower dimensions does not always improve performance, but it nevertheless improves training time with almost no cost to performance.

Step 5: Neural Net on Clustered and Projected Data

I interpreted step 5 as projecting the data into a lower dimension, clustering the results, and then appending the cluster prediction of each sample the sample's attributes. Due to the large number of experiments, I only ran the neural net on normalized data. Note that I normalized AFTER appending the cluster prediction.

K-means

Algorithm	Train Time	Train Score	Test Time	Test Score
PCA	7.08301	0.88927	0.02489	0.88801
ICA	2.66634	0.88283	0.02529	0.88535
Random Proj	9.59698	0.88321	0.02640	0.88152
K Best	6.19689	0.88311	0.02264	0.88277

EM

PCA	7.56283	0.88719	0.02484	0.88978
ICA	3.11389	0.88330	0.02471	0.88410
Random Proj	5.73676	0.88245	0.02477	0.88344
K Best	3.92300	0.88918	0.02345	0.88557

From the data above we cannot say anything conclusive about the performance of the neural net. Although EM does slightly better on most of the feature selection algorithms, the improvement is negligible. I did notice that the training times for the neural net were generally faster, which seems strange because I added another attribute to the samples. My idea is that having the cluster prediction in the sample allowed the neural net to converge faster. Other than the slightly faster training times, there is not much else to say; the performance is about the same as the non-clustered datasets.