Matt Busch
CS 4641

Project 3: Unsupervised Learning

Part 1: Clustering

In this report I will be looking at a variety of unsupervised learning techniques to learn about my datasets, without using their classifiers. In the first part, I will be demonstrating two different clustering algorithms to group the datasets into common attributes. First, I will be using K-Means clustering algorithm, and following up with Expectation Maximization. For this report, similarity / distance will be the Euclidean Distance formula when applicable. This seems to represent the distance the best for my dataset, as it tend to appear closer, and works well for the many continuous attributes.

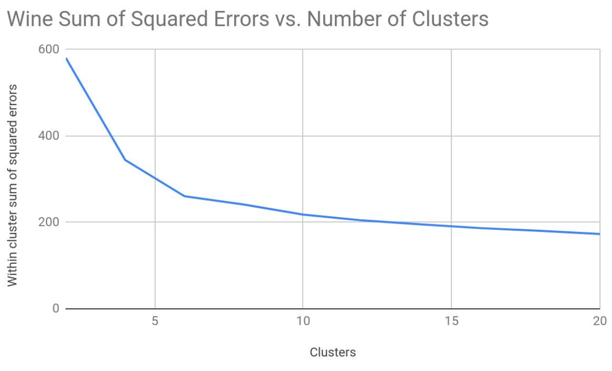
The datasets I am looking at in this report are the same as the first paper. The first dataset is a red wine ranking dataset, where many different red wines, have various features measured, and are ranked by wine experts on a scale of one to ten. This is however a classification over regression however, because the wines are ranked to a specific integer rank. A majority of the data falls within the five and six rank, and the minimum rank received is a three, and the maximum is an eight.

The second dataset I explored was a website database where a site was either classified as malicious or benign. This dataset looks at attributes of the site, and url, to determine if the site is a risk or not. Many of the attributes have zero values for a lot of examples, so clustering may prove to be especially easy or tricky depending on the remaining features. A majority of the websites are classified as benign, and much fewer are classified as malicious.

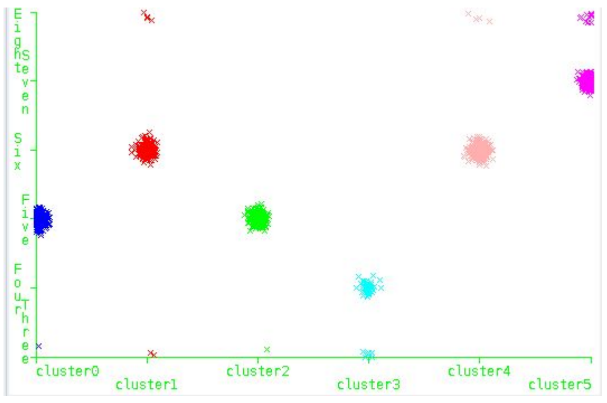
K-means Clustering

The first clustering algorithm I looked at is the k-means clustering algorithm. This algorithm groups the set into distinct clusters, based on their similarity through the Euclidean Distance Formula. It assigns a center point, and to compare closeness, it looks at the distance to the center point.

For the wine ranking dataset, I iterated through the number of clusters from 2 to 20 and compared the sum of squared errors. Based on the fact that more clusters cause there to be less number of wines per cluster, the sum should generally tend towards 0. To find the optimal number of clusters, I used the elbow method, which basically finds the largest change before the relationship slowly tends towards zero. In this case I found 6 was the optimal number of clusters, and I plotted the clusters versus the ranking below.

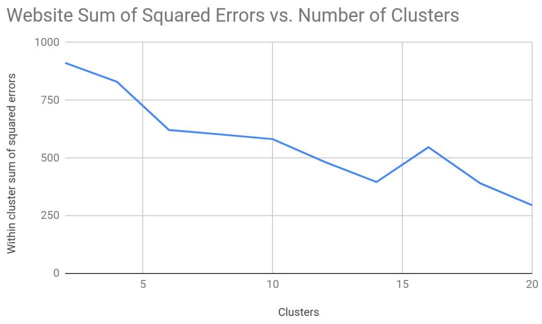


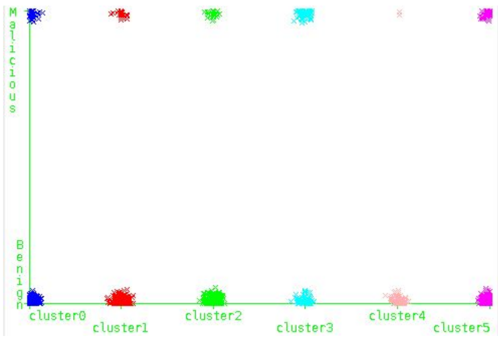
For 6

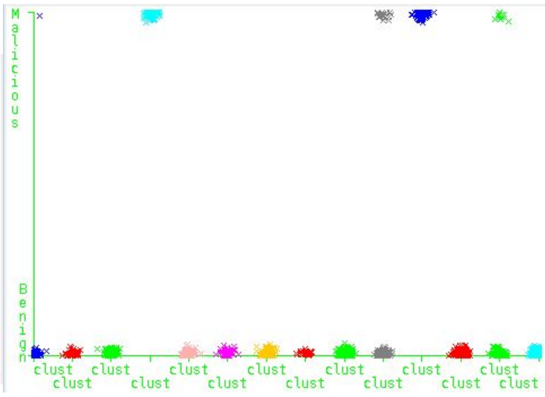


Clusters the algorithm does a pretty good job of clustering in regards to the ranking. There are distinct higher and lower clusters, for example cluster3 is made of fours and threes, where cluster5 is made of Sevens and Eights. There are some slight variations throughout the clusters, where some in cluster1 are Eight, some are Threes, and the majority are Sixes. This may be due to noise in the dataset, or perhaps an attribute can be very similar, but not be important for an overall ranking, causing these distinct rankings to be grouped similarly. Unfortunately due to the fewer numbers of Threes, and Eights, there would probably need to be a very large number of clusters before distinct Three and Eight Clusters are possible.

I followed a similar process for finding the best number of clusters for the Website Database. There is an unusual spike of square error at 16 clusters, which makes finding the optimal number more difficult, because the elbow is greatest at 14, but at 6, there is a more traditional elbow.





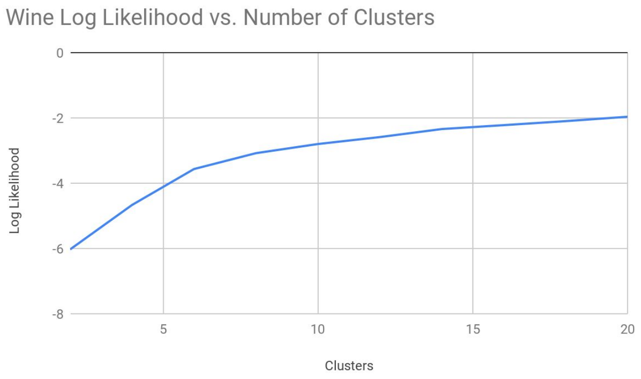


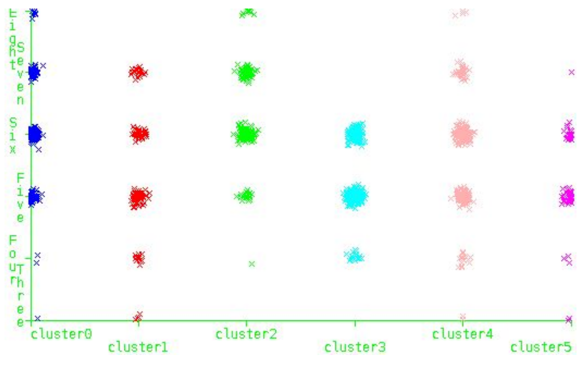
For 6 clusters, the algorithm doesn't cluster with regards to safety as well. The clusters all have Benign and some Malicious websites in the mix. When looking at 14 clusters, as seen below, it performs better, but even still, a few clusters contain both malicious and benign data. This indicates that the dataset may be tough to cluster with k-means. I will be looking at 6 clusters for a better comparison with the other dataset.

Expectation Maximization

The next algorithm we looked at for clustering is Expectation Maximization. This is similar to K-means as it groups the examples into clusters, but it uses a probability distributions for each example into each group, and maximizes the probabilities from there. To compare, we can look at the log likelihood that they are in proper cluster, the higher the likelihood, the better.

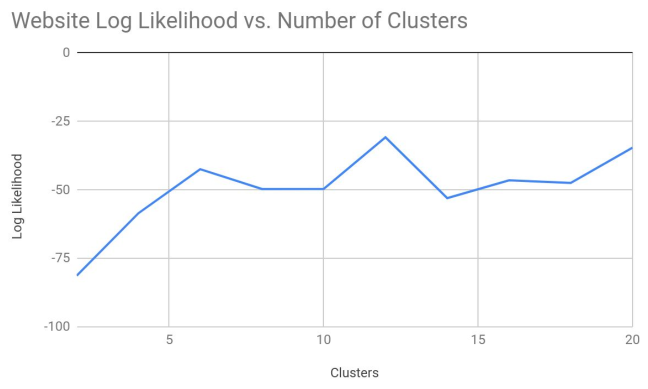
For the wine dataset I iterated from 2 to 20 increasing the number of clusters each time. I used a similar technique to find the optimal number of clusters and for this dataset around 6 clusters was the best.

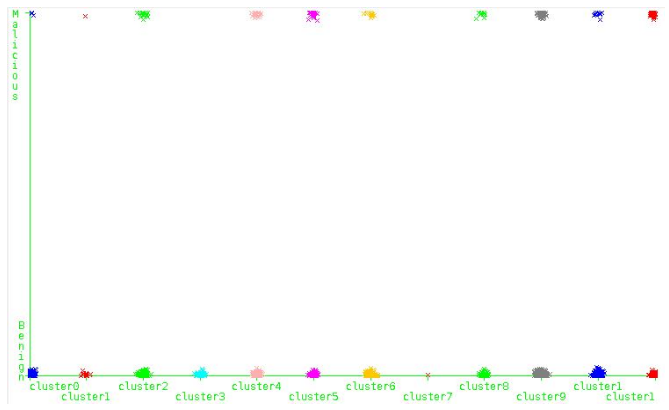




For the wine ranking dataset, the more clusters there are, the higher the likelihood is. At a cluster size of six, the clustering did not appear to cluster the examples well in regard to the ranking. Each cluster has rankings from all over the scale from Three to Eight. Only cluster3 had a cluster that was within only three ranks, so overall, these clusters do not very well represent the data.

Similar steps were used to find the best cluster size for the website dataset, and I found that the highest log likelihood was at 12 clusters. There was a maximum likelihood, and after 12 it held to less likely, and at 20 it came close to the same likelihood again.





Again Expectation Maximization did not perform well on this dataset either in regards to the safety of the site. There are very few clusters where the entire cluster is either benign or malicious. One cluster, cluster8, appears to have only one example in it, which indicates 12 may be slightly too many clusters. With only 11 clusters, that one lone example would probably fit into the next closest cluster.

Clustering Conclusion

The two clustering algorithms we tested behaved differently on our separate datasets. For the wine ranking dataset, there was a clear difference between K-means and Expectation Maximization algorithm. K-means was quicker and produced values that lined up with the classification much better than Expectation Maximization. Often times Expectation Maximization can get stuck at a local minimum and not produce the optimal results. With six main values that are being used for ranking, a higher number of clusters are needed as a minimum to ensure that the cluster breaks up the examples enough. The time complexity was much more for the Expectation Maximization due to the many attributes the algorithm had to consider. For K-means, this should be a simple task through Euclidian which is why it didn't have the same time penalty. 6 Clusters was optimal for both of these algorithms although did not match up with the classifications as well with Expectation Maximization.

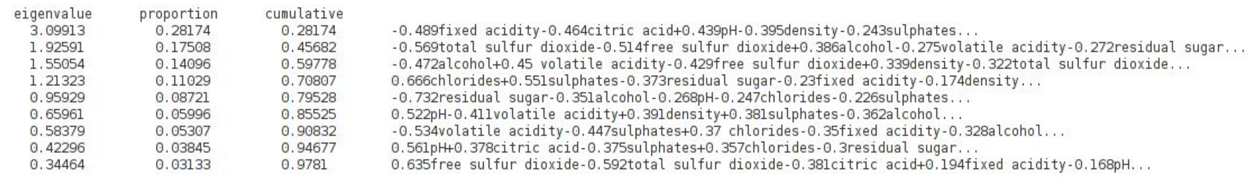
The website dataset was difficult to cluster with both algorithms. It appeared to perform better at the higher number of clusters for each, but the clusters did not match well to the safety classification with the Expectation Maximization. Like with the wine ranking dataset, there appeared to be an issue with matching up in EM, and even in K-means, there was not as great of a clustering as with the wine ranking. One issue could be with the number of common features that benign and malicious sites have. Where a supervised learning algorithm could learn that these features may not be important for classification, the unsupervised clustering may take that feature and have it dampen out and overpower the actual differences.

Part 2: Dimensionality Reduction Algorithms

In the first part we looked mostly at clustering algorithms, now we will look at Dimensionality Reduction Algorithms, and see how they work with the clustering algorithms.

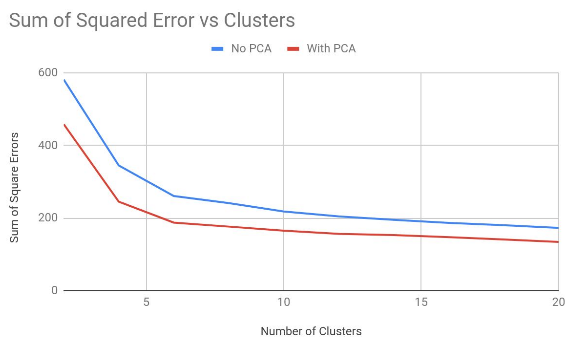
PCA

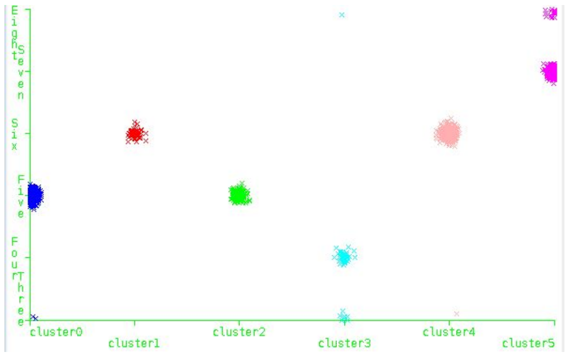
PCA converts the attributes to linearly separated variables using orthogonal projection. It should not be used with discrete values, but that is not an issue with the wine ranking dataset, but may cause some slight issues with the website dataset, as some attributes are discrete. I ran with 95% variance covered.



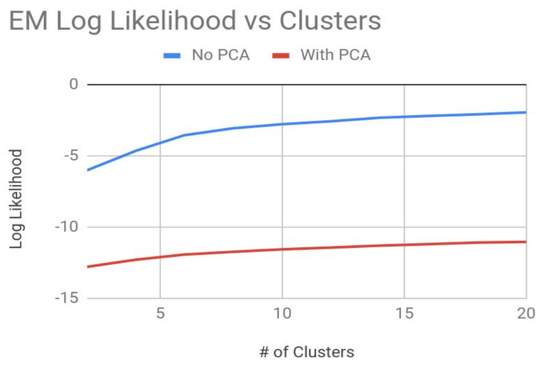
```
eigenvalue     proportion     cumulative
  3.09913        0.28174        0.28174      -0.489fixed acidity-0.464citric acid+0.439pH-0.395density-0.243sulphates...
  1.92591        0.17508        0.45682      -0.569total sulfur dioxide-0.514free sulfur dioxide+0.386alcohol-0.275volatile acidity-0.272residual sugar...
  1.55054        0.14096        0.59778      -0.472alcohol+0.45 volatile acidity-0.429free sulfur dioxide+0.339density-0.322total sulfur dioxide...
  1.21323        0.11029        0.70807      0.666chlorides+0.551sulphates-0.373residual sugar-0.23fixed acidity-0.174density...
  0.95929        0.08721        0.79528      -0.732residual sugar-0.351alcohol-0.268pH-0.247chlorides-0.226sulphates...
  0.65961        0.05996        0.85525      0.522pH-0.411volatile acidity+0.391density+0.381sulphates-0.362alcohol...
  0.58379        0.05307        0.90832      -0.534volatile acidity-0.447sulphates+0.37 chlorides-0.35fixed acidity-0.328alcohol...
  0.42296        0.03845        0.94677      0.561pH+0.378citric acid-0.375sulphates+0.357chlorides-0.3residual sugar...
  0.34464        0.03133        0.9781       0.635free sulfur dioxide-0.592total sulfur dioxide-0.381citric acid+0.194fixed acidity-0.168pH...
```

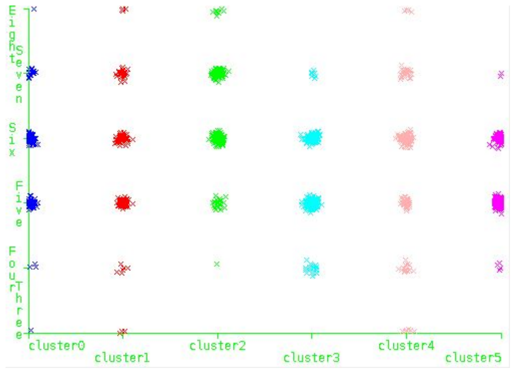
For the wine ranking dataset, we get the eigenvalues above. The eigenvalues vary enough which indicate that PCA ran effectively. PCA was able to lower the number of attributes from 12 to 9 with 95% variance covered. The next step is to see how it affects the clustering algorithm and learning through Neural Nets.





The plot of the squared error with respect to number of clusters is very similar between the graph with PCA, and without PCA. With PCA however always has a slightly less error for each cluster, and that may be because there are 3 less attributes. Those attributes probably were not the exact same, and because they overall did not affect the variance that much, still created a similar looking curve and similar looking clustering assignments. The clusters look very similar to those from Part 1, but slightly less errors in respect to the ranking, perhaps due to the less attributes.





And a similar

situation is shown with the Expectation Maximization. The likelihood is less each time, but follows a similar curve. Additionally, the graph looks very similar to the one from part 1, but the clusters seem to be a little more tight with respect to ranking.

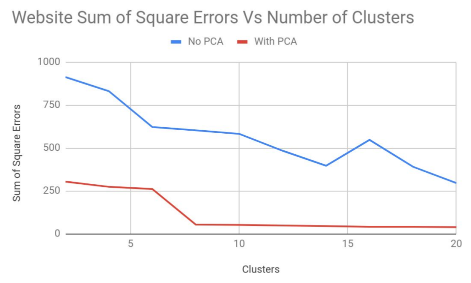
No PCA	PCA	PCA+K-Means	PCA + EM
60.73	60.037	68.21	65.52

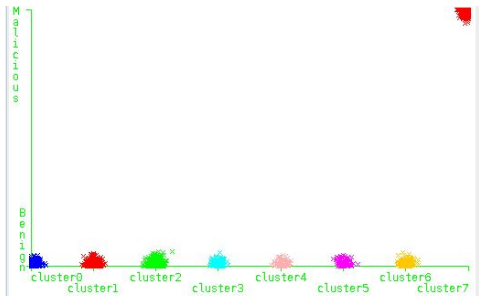
I then ran a neural net off of the new attributes created by PCA, PCA with K-means, and PCA with Expectation Maximization and looked at the accuracy. Each was ran on a neural net with the optimal hidden layers, and with a cross-validation of 10 folds. What we see is that with K-means Clustering, the neural net performs very strong. It has very high accuracy, probably due to the strength that the clustering algorithm clustering by ranking. Although the Expectation Maximization grouping doesn't appear to group the examples well in regards to ranking, when using grouping as an attribute for the neural network, the accuracy improves greatly. The PCA by itself, however does not show improvement over no PCA. The speed for PCA + K-means seemed to improve, perhaps because the strength of the clusters to define the ranking, the neural net can quickly find an optimal solution, and requires less change.

Website

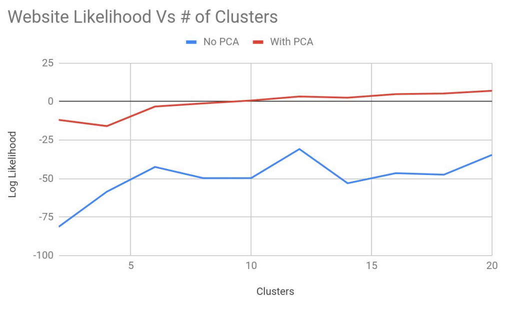
```
eigenvalue    proportion    cumulative
  6.08983       0.25374       0.25374       0.394SOURCE_APP_PACKETS+0.394APP_PACKETS+0.394REMOTE_APP_PACKETS+0.392TCP_CONVERSATION_EXCHANGE+0.313SOURCE_APP_BYTES...
  2.27346       0.09473       0.34847       0.461APP_BYTES+0.459REMOTE_APP_BYTES+0.412DNS_QUERY_TIMES=0.0-0.314REMOTE_IPS+0.307DIST_REMOTE_TCP_PORT...
  2.02321       0.0843        0.43277       -0.634NUMBER_SPECIAL_CHARACTERS-0.627URL_LENGTH-0.221CHARSET-0.173CONTENT_LENGTH+0.164APP_BYTES...
  1.48369       0.06182       0.49459       -0.435DNS_QUERY_TIMES=0.0+0.418REMOTE_IPS-0.397SOURCE_APP_BYTES+0.295DNS_QUERY_TIMES=6.0+0.234DNS_QUERY_TIMES=4.0...
  1.19511       0.0498        0.54439       -0.71DNS_QUERY_TIMES=4.0+0.598DNS_QUERY_TIMES=6.0-0.198CHARSET-0.166CONTENT_LENGTH+0.145DNS_QUERY_TIMES=12.0...
  1.13575       0.04732       0.59171       0.645DNS_QUERY_TIMES=2.0-0.532DNS_QUERY_TIMES=8.0-0.333CHARSET+0.25 CONTENT_LENGTH-0.139DNS_QUERY_TIMES=20.0...
  1.1003        0.04585       0.63756       -0.61DNS_QUERY_TIMES=8.0-0.54DNS_QUERY_TIMES=2.0+0.346DNS_QUERY_TIMES=6.0+0.318DNS_QUERY_TIMES=4.0-0.2CONTENT_LENGTH...
  1.05069       0.04378       0.68133       -0.739DNS_QUERY_TIMES=12.0+0.379DNS_QUERY_TIMES=6.0-0.311DNS_QUERY_TIMES=10.0+0.311CHARSET+0.173DNS_QUERY_TIMES=20.0...
  1.01589       0.04233       0.72366       0.869DNS_QUERY_TIMES=10.0-0.294DNS_QUERY_TIMES=12.0-0.249DNS_QUERY_TIMES=8.0+0.162CONTENT_LENGTH+0.157DNS_QUERY_TIMES=14.0...
  1.00274       0.04178       0.76544       -0.728DNS_QUERY_TIMES=14.0-0.513DNS_QUERY_TIMES=9.0-0.323DNS_QUERY_TIMES=NA+0.242DNS_QUERY_TIMES=10.0-0.161DNS_QUERY_TIMES=20.0...
  1.00142       0.04173       0.80717       -0.715DNS_QUERY_TIMES=9.0+0.622DNS_QUERY_TIMES=14.0-0.231DNS_QUERY_TIMES=NA-0.138DNS_QUERY_TIMES=12.0-0.108DNS_QUERY_TIMES=20.0...
  1.00023       0.04168       0.84885       -0.881DNS_QUERY_TIMES=NA+0.418DNS_QUERY_TIMES=9.0-0.141DNS_QUERY_TIMES=20.0+0.11 DNS_QUERY_TIMES=14.0+0.106CONTENT_LENGTH...
  0.97986       0.04083       0.88967       0.784DNS_QUERY_TIMES=20.0-0.402CONTENT_LENGTH-0.225DNS_QUERY_TIMES=NA+0.202DNS_QUERY_TIMES=2.0-0.168CHARSET...
  0.9703        0.04043       0.9301        0.601CONTENT_LENGTH+0.467DNS_QUERY_TIMES=20.0+0.447DNS_QUERY_TIMES=12.0+0.332CHARSET-0.127DNS_QUERY_TIMES=8.0...
  0.82852       0.03452       0.96462       -0.703CHARSET+0.491CONTENT_LENGTH+0.301DNS_QUERY_TIMES=8.0-0.298DNS_QUERY_TIMES=2.0-0.197DNS_QUERY_TIMES=12.0...
```

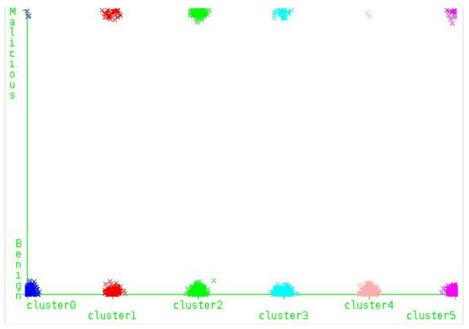
For the website dataset, the PCA did not remove any attributes, and still resulted in 15 attributes needed for the dataset. However the eigenvalues indicate that the algorithm successfully ran, and each attribute provided nearly equal proportion, besides the first. This indicates, all are important, and needed for high coverage.



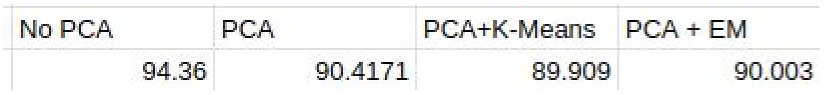


PCA + K-means had much lower squared error, and after 8 clusters, had a very low error. At eight clusters, the algorithm separated the examples into clusters that nearly perfectly matched the labeled data. All the Malicious data was in cluster7 while the other data was separated throughout the other clusters.





The Log Likelihood with PCA is much higher than that without using the algorithm, and actually breaks into the positive, which indicates EM is able to effectively cluster the examples. With respect to website danger, it still appears to be not clustered well, but with a positive likelihood, the examples must share many features.

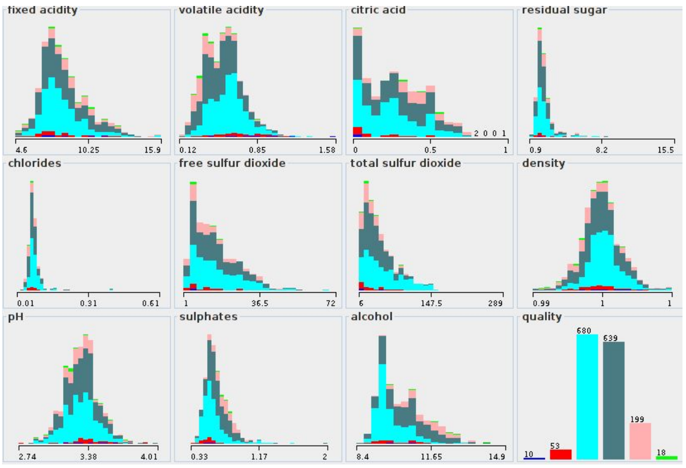


No PCA	PCA	PCA+K-Means	PCA + EM
94.36	90.4171	89.909	90.003

The addition of PCA performed poorly when used with a neural network. Although K-Means and PCA created a great clustering of data points with respect to labels, it does not translate to the neural network. It had the lowest accuracy of all the neural Networks, and without PCA performed the best. This shows that PCA did not perform overall very well on this dataset, which may be due to the discrete nature of some of the attributes

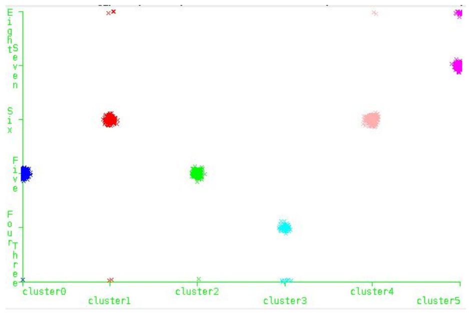
ICA

ICA is another dimensionality reduction algorithm that can be used to alter the data to learn on. Unlike with PCA, this tries to maximize independencies. With this algorithm, it assumes that all attributes are non gaussian and independent. It measures kurtosis in order to determine how normal the distribution is, and the closer to 3, the more normal the data is.

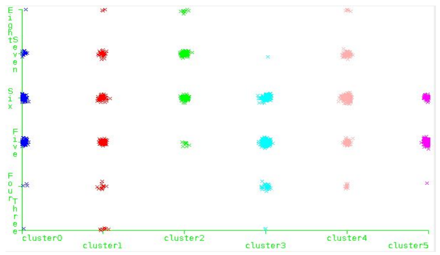


For the wine dataset, the kurtosis is: [4, 4, 2, 31, 44, 5, 6, 3, 3, 14, 3], which shows that some of the attributes are very normal, but a few are a bit off normal. If you look at the distribution of points, you can see this is true. Some look like normal curves, and some have large tails. This indicates the algorithm may not perform well, as it is so guasian in most attributes.

I filtered the data using this algorithm and test the best clusters using k-means to use with the least sum of squared errors. Once again I found that 6 was the best to use, so I looked at the clusters that came from a cluster size of 6. I found that this was almost identical to the original one from the first part. This indicates that the algorithm was unable to effect the data in a large way, or that this size is optimal for 6 clusters.

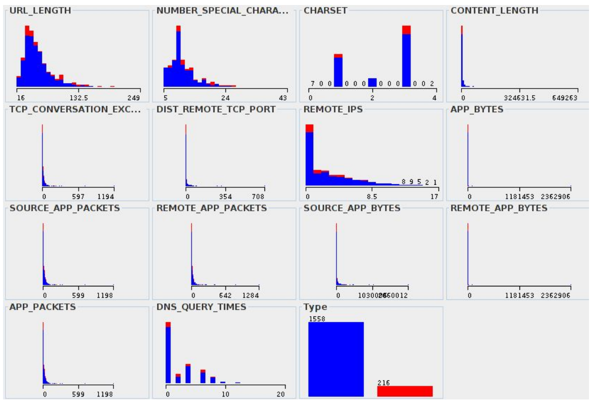


I then ran the algorithm and clustered with Expected Maximization and notice a very similar pattern to the original as well. I am unsure whether ICA is playing an important role in the clustering. I will take these results and run them through the neural network to determine if there was a positive effect of running ICA on the dataset.

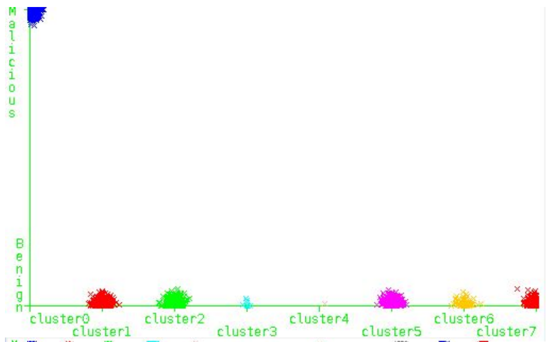


No ICA	ICA	ICA+K-Means	ICA + EM
60.73	56.285	55.22	56.097

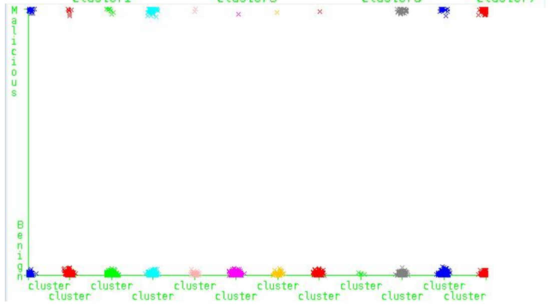
The results of the Neural Network using ICA are not positive. It seems that with the addition of ICA, the neural network performs worse, with the accuracy decreasing 4-5%. The K-means performs the worst with the addition of ICA, and Expectation Maximization doesn't improve much. Expectation Maximization took much longer than the other ones, and with all of the reduced accuracy, the default is probably the best without ICA. This may be because the graphs were too guasian, so the algorithm did not perform well.



For the website dataset, the ketosis is [7, 8, 1, 257, 454, 641, 3, 1759, 408, 374, 461, 1758, 408, 3], which indicates there are very non-gaussian attributes, but unfortunately I know that many of the attributes are dependent on each other, which also is a negative for this algorithm. As shown in the distribution of the dataset, many of the attributes have very large tails, and it is clear to see they should have large ketosis.



For K-means, the result of this clustering is similar to previous ones, but however this only reaches optimal results after 8 clusters. The interesting thing is that cluster3 and cluster4 seem like they aren't important, as they are so small, but having only 6 clusters give a very mixed result in regards to the safety of the site. Cluster4 seems to only have 1 example in it which probably indicates it is very different from the rest.



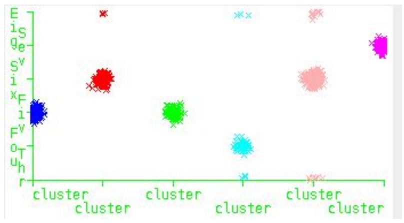
The next clustering shows the best result for Expectation Maximization, and it requires a higher number of clusters at 12 clusters in order to reach the best result. Nearly every cluster has at least one example of both malicious and benign data in it. Even at higher clusters, it isn't seemingly possible to create clusters of just malicious or benign. Using these clusters, I created neural nets to test how they perform.

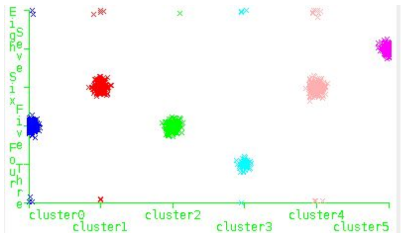
No ICA	ICA	ICA+K-Means	ICA + EM
94.36	94.12	94.402	94.233

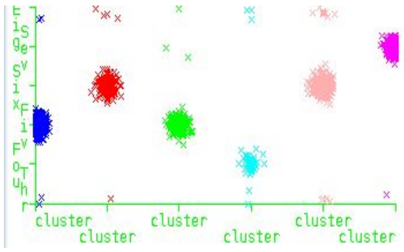
The result of the neural net are above. There was not a large negative decrease in accuracy using ICA, but there was one. Only ICA + K-means performed better than the default without ICA added. Running ICA and K-means added more time, but the amount was not a large concern. The dataset was better for ICA than the wine as there were less gaussian attributes, but it did not perform better, most likely due to the dependent nature of the attributes.

Randomized Projection

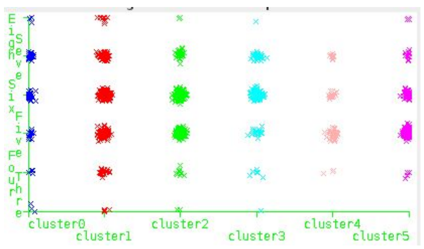
The next dimensionality reduction algorithm I used was randomized projection. This algorithm reduces the attributes down to a specific number randomly. It uses projections to reduce, but often are not optimal, as it is completely random. I experiment with reducing the algorithm drastically, about half and just a little from 12 to 3, 6 and 9 respectively. As this algorithm is random, using different random seeds will produce dramatically different results, so I will be experimenting with a few different seeds, and attempting to display the best result.

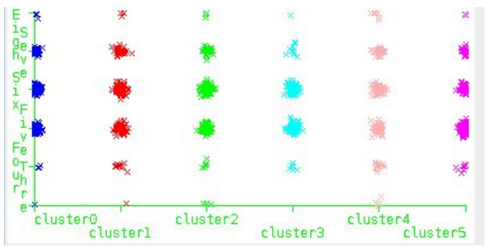


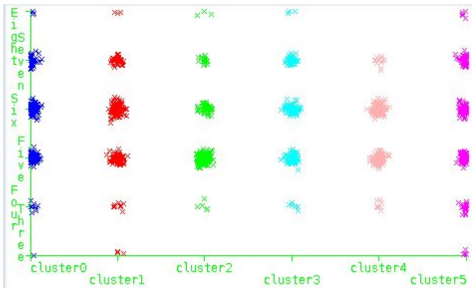




These graphs represent the best clusters, with the least square error I could find for 3, 6, 9 respectively. The lower the attributes get projected onto, ie. 3, the less the square error is, and the better the results. I will do the same thing with Expectation Maximization next.







With EM, the large number of attributes seemed to cluster the best. Projecting onto 3 attributes randomly appears better than onto 6, but in terms of ranking, projecting onto 9 attributes has a square error that is low for the amount of attributes, indicating they are clustered better. The website dataset performed similarly, with a projection of 4, 8, 12, and due to space concerns, and the lack of anything new, I chose not to show the results. Below is the results of the neural network on both datasets, and based on randomly projecting to the 3, 6, and 9, attributes.

Wine

No Random	Random 3	Random 6	Random 9	Random + K-Me	Random+EM
60.73	47.46	50.96	51.21	49.96	48.96

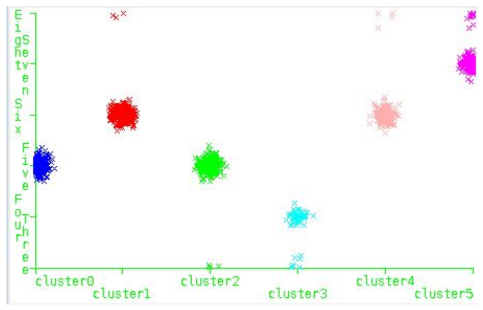
Website

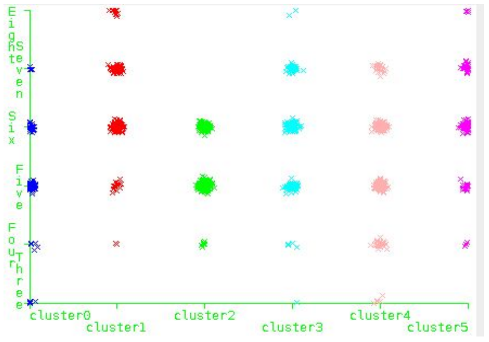
No Random	Random 4	Random 8	Random 12	Random + K-Means	Random+EM
94.36	87.82	88.38	89.17	88.89	83.2

Randomized Projection performed horribly for the neural networks. It neural network worked almost as poorly as randomly assigning a classification for the attributes. This indicates that randomly reducing the number of attributes, isn't a great idea. It did perform quicker, having less attributes to look at, but the lack of performance makes this not an effective method.

Information Gain

The final method I will look at for Dimensionality Reduction is Information Gain. This is different from the others where it is a supervised reducer, where it uses the classification of each example to rank the best attributes that provide the most information. I then remove the attributes with low information gain, and create the clusters with fewer attributes.

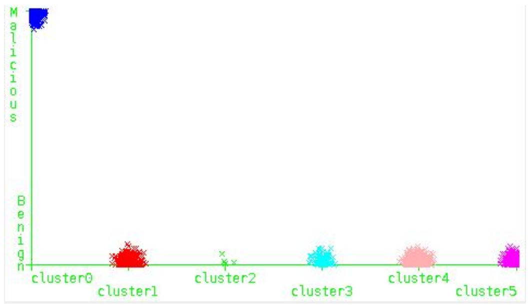


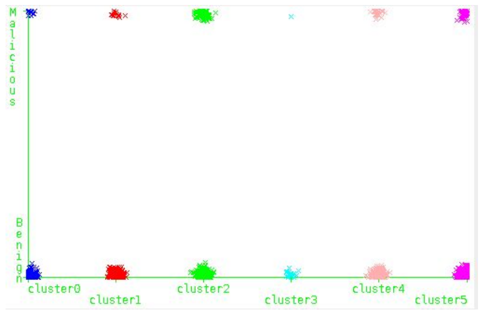


For the wine dataset above I determined removing the three least contributing attributes performed the best. The K-means cluster (left) performed very well in regards to the ranking. With fewer extraneous attributes, the clustering was able to use the attributes that mattered to cluster, making the clusters tighter, and more accurate. Additionally, the overall squared error decreased for similar reasons. The EM also performed better with a higher log likelihood with the fewer attributes.

No IG	IG	IG+K-Means	IG + EM
60.73	59.47	58.47	58.28

The Neural Network actually took longer to run after using Information Gain, even though there were less attributes. The accuracy also decreased. This could mean it took longer for the backpropagation to reach an equilibrium, so overall a decrease in performance. After each clustering, the algorithm again took even longer, so for the best results, I would not recommend using Information Gain for this dataset, unless file size is a concern, and you can reduce the unnecessary data.





For the website dataset, the results were similar to the wine, and the clustering seemed to be effective. I removed the 4 least contributing attributes and remained with 8. The left shows K-means, and looks similar to previous clusterings, but with the fewer attributes, the square error was down. For EM, on the right, when compared to similar looking cluster arrangements, it performed just as well with fewer attributes.

With Information Gain, the Neural Network ran significantly quicker, but the drop in accuracy is dramatic. As with almost all of the dimensionality reduction algorithms however, the accuracy decreased. This could be due to some attributes being dependent on each other, so although a specific attribute may not provide much information gain, together with others it provides a lot of information.

No IG	IG	ICA+K-Means	ICA + EM
94.36	90.13	90.41	83.7