Assignment 3: Unsupervised Learning

[Jainil Modi]
[jmodi30@gatech.edu]

I. Abstract

The following paper will demonstrate the results of running two clustering algorithms, K-means and Expectation Maximization (via the Gaussian Mixture Model). Additionally, the paper will demonstrate four dimensionality reduction algorithms: PCA, ICA, Randomized Projections, and a Manifold Projection (tSNE). The resulting datasets from said DR algorithms will then be retried on the clustering algorithms, and a neural network from Assignment 1. The datasets used are the canonical PIMA Indians Diabetes dataset, and the Red Wine Quality dataset. The goal of this work was again to optimize the f1-score, weighted. As a refresher, the f1-score (weighted) calculates the f1-score, and then it takes into account the data count for each respective category, thus being weighted.

II. Clustering Algorithms

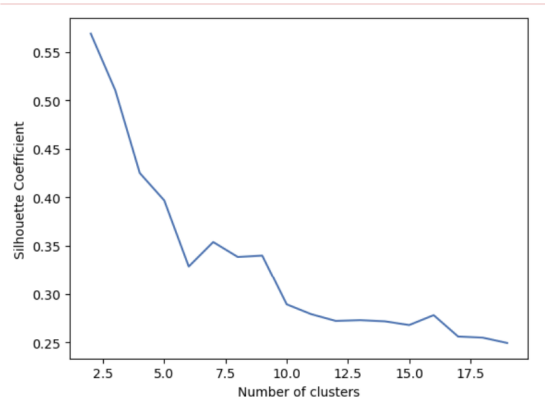
A. PIMA Indians Diabetes Dataset

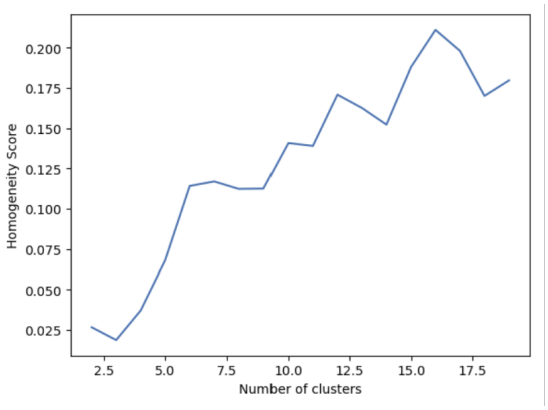
Seeing as to how this dataset is very simple, meaning there are not too many attributes, my expectations for clustering were high, but not too high, as the dataset does not have many records, and this may be poor for the training.

1) K-Means: In order to demonstrate the baseline for the clustering, I ran the classic elbow method on two different metrics: homogeneity score and silhouette score. Homogeneity score, as defined by scikit-learn, measures the whether or not ”all of [the] clusters contain only data points which are members of a single class”. Because we are doing k-means, which does not work like a ”soft” clustering algorithm, this is useful for measuring how accurately we clustered our points. Additionally, because we already have some sort of idea for what the labels should be (we have the y-values for each record), using this makes sense because there is some ”ground truth”, even if we are not using this directly. Another metric I tried using was silhouette score, which essentially measures intracluster distance. Silhouette score is measured from -1 to 1, where 1 is the strongest. A score above 0.5 is considered ”good”.

It almost seems like homogeneity and silhouette paint opposite pictures with respect to the number of clusters. Ideally, this is like the bias-variance trade-off that we are familiar with from supervised learning. We want to be in that ”sweet spot” so we avoid overfitting. Too many clusters, and we could ruin the picture by overfitting (high variance and little bias). Too few clusters, and we could be underfitting (low variance and high bias). As the picture is unclear here, we will try both 2 and 3 clusters.

After trying kmeans with n_clusters = 2 and n_clusters = 3, the results indicated that 2 clusters was the better choice:

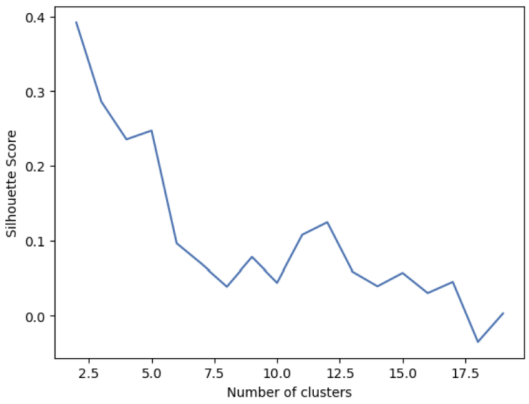


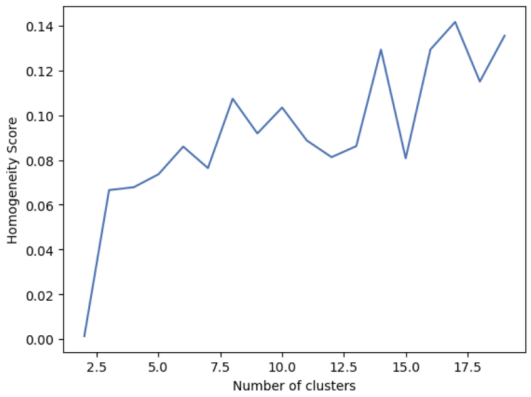


since we are trying to maximize our f1-score, we want to go with the number of clusters that does this. 3 clusters gave us an f1-score of 0.56878497, whereas 2 clusters resulted in 0.68470887. 2 clusters clearly outperforms 3. Further, the adjusted_mutual_info_score was used. This metric is on a scale from -1 to 1. A higher score is better. Although the score was very weak for both experiments, 2 clusters still had a higher score (0.00483582 for 2 clusters, 0.00364678 for 3 clusters).

2) EM via Gaussian Mixture Model: In the proceeding experiments, I anticipated very different results. This is due to the inherent nature of EM. It is the opposite of kmeans, for example, which is a hard clustering algorithm. A data point is labeled with exactly one value. It is not based on a probability, which is what EM does. The specific iteration that we are using is Gaussian Mixture Model. This iteration assumes that the data are generated from a mixture of several Gaussian distributions.

From above, I think there are 2 options for the numbers



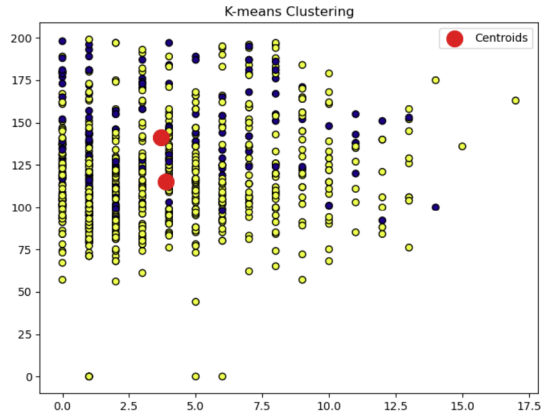


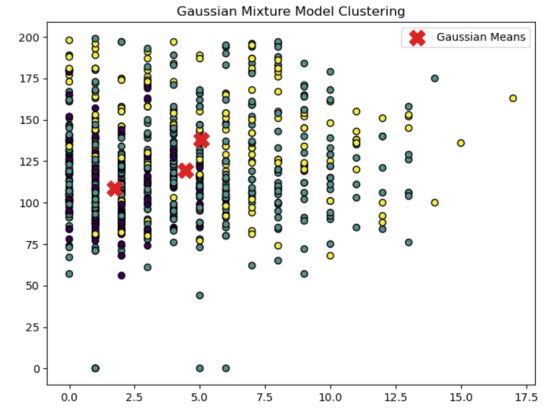
of clusters: the silhouette coefficient demonstrates that the number of clusters should be somewhere around 4, but the homogeneity score says 2. After trying both values, here are the results: the f1-score for 2 was 0.46447958, whereas for 4 clusters, it was significantly worse: 0.1436092.

I think this drastic, sharp decrease in score can be attributed to the weak nature of soft clustering algorithms: when datasets such as this one have labeled data (i.e. there is some ground truth), you cannot get away with using probabilities. Answers must be definitive, because they naturally are.

However, adjusted_mutual_info_score for 2 clusters was 0.000276845. The adjusted_mutual_info_score for 4 clusters was 0.04552632. I find it interesting that the mutual information score was higher for a greater number of clusters. Out of curiosity, I wanted to see what the adjusted_mutual_info_score would be for 3 clusters. My intuition was that 3 may potentially be a local/global maxima. I got 0.04937895, and I was correct. The reason I felt this might the case is that there was not a sudden drop in the silhouette score curve immediately after 2.

3) Comparison: Overall, kmeans is a much better suited algorithm for the type of data that we have: labeled data with definitive ground truths. I think one big difference that is worth pointing out here is the interpretability of KMeans versus GMMs: because KMeans is hard clustering algorithm, if we have ground truth labels for our data, we can use metrics like f1-score to evaluate the outcome. However, this is not the case whatsoever with GMMs, which are probabilistic by nature. Because there is not a definitive answer given by these algorithms, it is essentially impossible to run the same metrics on them, which is why you may have noticed that I did not use f1-score above when discussing EM. Lastly, the distribution of "centroids" was noticeably different for both algorithms:





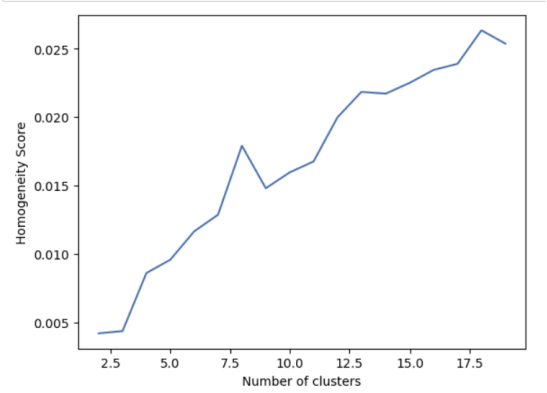
B. Red Wine Quality Dataset

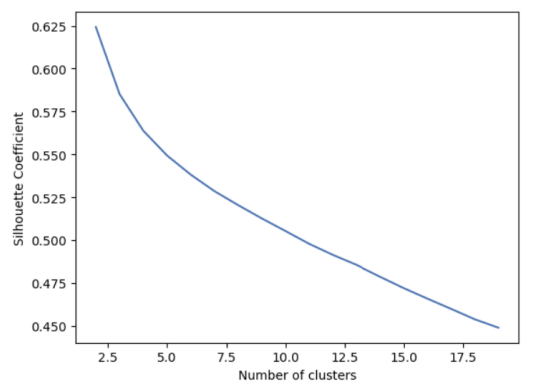
After going through the two algorithms above for the PIMA Indians diabetes dataset, my hypothesis is that kmeans again will outperform Gaussian Mixture Models.

1) K-Means: Like the other dataset, I wanted to draw an intuition about what amount of clusters may be the right choice. I started off with the silhouette score and homogeneity score graphs to get a rudimentary understanding of the data:

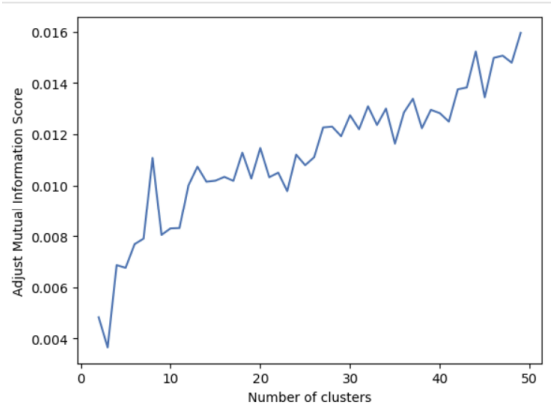
The interesting thing about this dataset that is different from the other is that both the silhouette score and homogeneity score seem to agree that 3 is the optimal number of clusters. However, it could be argued from the homogeneity score that 4 is better. Therefore, I will try both:

After attempting both three and four clusters for the f1-score, I received a score of 0.0 for both, even after rounding to 9 decimal places. This illustrates the idea that the data very homogeneous in nature and clustering may struggle to find distinct clusters. This can further be corroborated by an





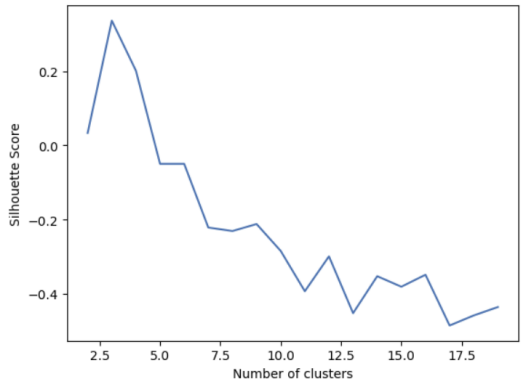
increasing adjusted mutual information score when plotted against an increasing number of clusters:



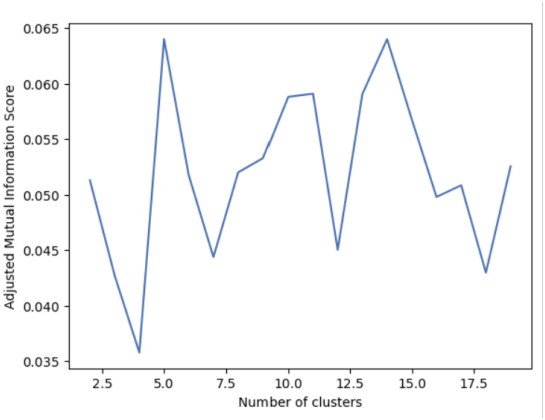
When the number of clusters has gotten substantially large, the algorithm is very likely to be overfitting.

2) EM via Gaussian Mixture Model: Like the dataset above, I anticipated very different clusters for this algorithm than KMeans above. I also think that because the results above gave poor values for f1-score, that EM will do ”worse”: a lower homogeneity score and silhouette score, as well as a better adjusted mutual information score.

In unison with my reasoning above, the homogeneity score was actually very poor: it remained stagnant at 0.0013. The logic for this, I think, is very intuitive: homogeneity score is measure of how well your assigned clusters actually map to the data. Because we know from the above examples that there are not very clear boundaries, it is hard to separate the data and get a clear picture from it.



The silhouette score is the only metric that sort of paints a clear picture: it seems like either 3 or 4 clusters would be the best in maximizing our adjusted mutual information score.



The graph above is very interesting: it tells a similar story to that of the homogeneity score: the data are not so easily separable, hence the ebbing and flowing of the score.

3) Comparison: Overall, KMeans again outshined EM. Although neither algorithm performed particularly well here, I would choose KMeans over EM. However, what I take away from these first few experiments is again the idea that not all algorithms will work well on all kinds of data. The datasets I have chosen do not seem to pair well with clustering algorithms, albeit soft clustering or hard clustering.

III. Dimensionality Reduction

Because both of datasets are relatively simple, meaning they do not have many records and many many attributes, I do not think that dimensionality reduction will benefit performance for either dataset all that much.

A. PIMA Indians Diabetes Dataset

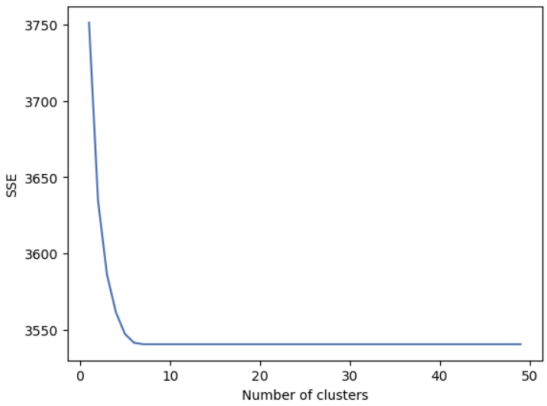
1) Principle Components Analysis: Two good metrics for PCA are explained variance ratio and reconstruction error.

Explained variance is exactly what it sounds like. In terms of a threshold, I was aiming for the top four attributes to keep.

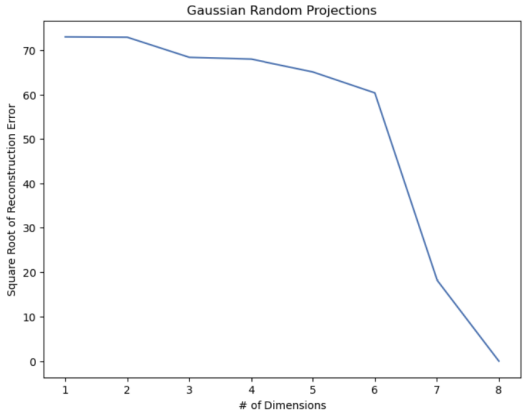
After running PCA on this dataset, the first four components had the following values: [0.88854663 0.06159078 0.02579012 0.01308614]. The first principle component explains 88 percent of the variation in the data.

The first four principle components amounted to 0.98901367 of the total explained variance. Lastly, the reconstruction error was not too high: 3561.3013249. This was measured using the mean squared error.

2) Independent Components Analysis: Unlike PCA, which attempts to project attributes orthogonally and find the highest explained variances, ICA attempts to separate multivariate signals into statistically independent, additive components. In order to assess the efficacy of ICA, I measured the reconstruction error to dictate how many attributes should be used:

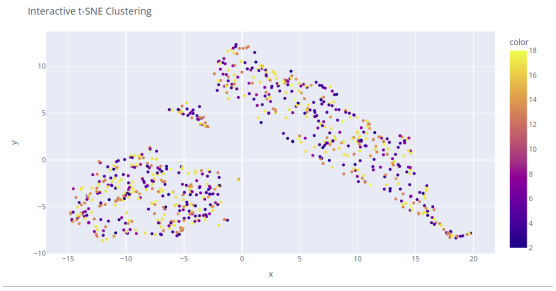


From the graph above, we can see that the best option would be either 6 or 7 features.



3) Random Components Analysis: After 7 features, we do not see a drastic drop in RMSE until all features are used. Therefore, I would choose 7 features based on GRP.

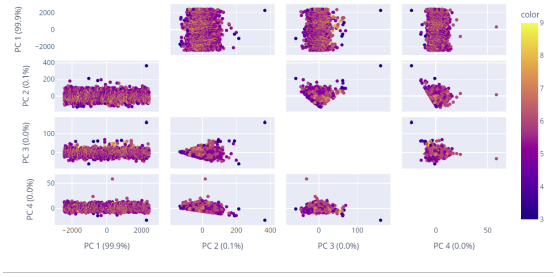
4) Manifold Analysis - tSNE: Looking at the graph above, t-sne did a fantastic job separating the data into 3 distinct clusters. However, something like a silhouette score would not make sense, as information from the original higher



dimensional space is required. Instead, I will use the Davis-Bouldin index, which measures the average 'similarity' between each cluster and its most similar one, while considering the dissimilarity between clusters. Here, a lower score is better. Since the score ranges from 0 to infinity, and the score we achieved was 4.42747, this is a fairly good score.

B. Red Wine Quality Dataset

1) Principle Components Analysis: In contrast to the other dataset, most the features here are extremely weak in terms of explained variance. I used the same 4 feature threshold here, and the attributes had the following values: [9.98963021e-01 9.42576295e-04 8.28146774e-05 1.07216436e-05]. These are extremely poor values.



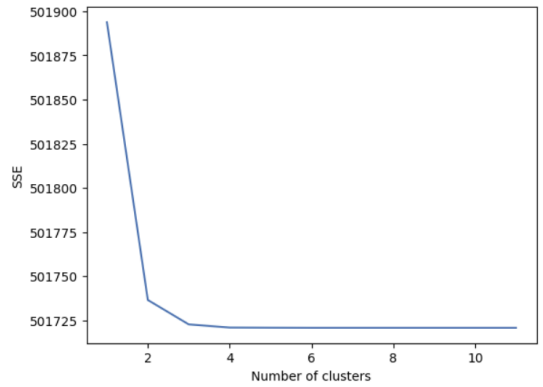
The above figure demonstrates the poor separation from PCA. There was an extremely high reconstruction error: 501721.02406, which is obviously attributable to poor explained variance.

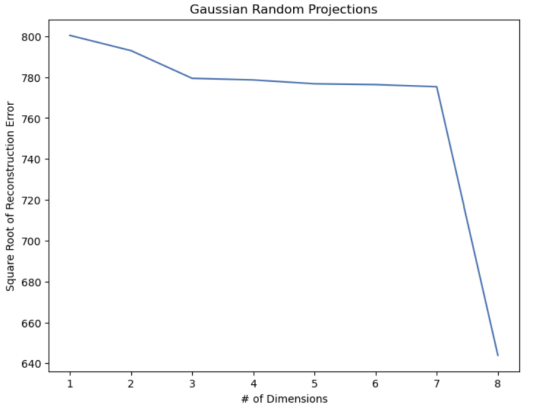
2) Independent Components Analysis: Similar to the ICA above, I used reconstruction error as a metric to determine the number of components:

From this, we should use 2 components. It should be noted, very interestingly that both PCA and ICA had the same reconstruction error value: 501721.0240. This is because the orthogonal rotation of the matrix does not technically affect the reconstruction error.

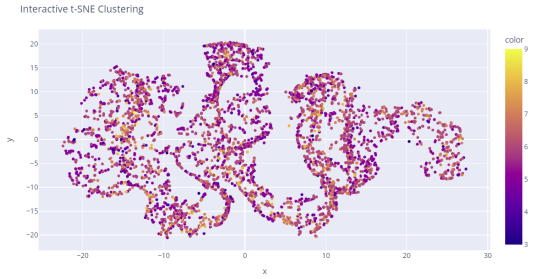
3) Random Components Analysis: Similar to above, the best way to measure the efficacy of the reduction and number of features is to evaluate the reconstruction error. I also decided to keep 4 features here, and the reconstruction error was 606319.30344.

From this picture though, the ideal value would have been 3, as the RMSE does not drop at all afterwards.





4) Manifold Analysis - tSNE: t-SNE by far was the most clear picture presented:



From this, although there are not clear separate boundaries, the data are much more separated than the above methods of dimensionality reduction. This very clearly indicates a non-linear relationship within the data.

IV. KMeans and EM Post Dimensionality Reduction

In this section, we will use the f1-score for KMeans and adjusted_mutual_info_score for GMM. A total of 16 iterations were run below.

A. PIMA Indians Diabetes Dataset: Post DR

1) Principle Components Analysis: After taking out half the features here, the KMeans algorithm performed under PCA, the algorithm not only ran extremely quickly, it performed very well: 0.6847088738408164, which was on par with the original score. This can be attributed to the fact that the first principle component constituted greater than 88 percent of the explained variance in the outcome variable. Keeping the top 4 attributes with highest explained variance did not really make much of a difference in terms of score because they were almost 96 percent of the total explained variance.

In contrast with KMeans, however, the Gaussian Mixture Model did not perform much better, if at all. Using the reduced dataset, we got a adjusted_mutual_info_score of 0.008439467020641737. In the grand scheme of things, this score is not that good. In the perspective of the work we have done, though, it is twice as good as the original score we got.

2) Independent Components Analysis: For the dataset that we "pruned" using ICA, we also saw a worsened f1-score: 0.4551834310665832. The reason for this, I think, is because ICA assumes statistical independence. If this were not in fact the case, then it would not help us. A tool we could use to determine multicollinearity would be something like variance inflation factor.

For the adjusted_mutual_info_score from the Gaussian Mixture Model, the value was 0.000459885025487152, which was 10 times worse than the original.

3) Random Components Analysis: Of all four dimensionality reduction algorithms run on this dataset, the Gaussian Random Projections performed the worst. The f1-score produced by KMeans when using this reduced dataset was 0.40122204751921736, almost 50 percent worse than the original. This signals that the data are not from a form that is Gaussian in structure.

Additionally, the adjusted_mutual_info_score from the Gaussian Mixture Model was 0.0037869421231026723, which was not as poor as the one for ICA, but still worse than the baseline nonetheless.

4) Manifold Analysis - tSNE: Of the four dimensionality reduction datasets, this one performed second to last on the KMeans evaluation: 0.47233711795241934 was the score. I anticipated this one being higher due to the efficacy above of t-SNE with the entire datasets, but I was surprised when this did not repeat.

Interestingly, the adjusted_mutual_info_score was negative here, the only time that that has happened thus far: -0.0015196303905651784. Although it was a weak negative value, the negative value implies that the clusters that were identified were unrelated to the true structure of the data.

B. Red Wine Quality Dataset: Post DR

1) Principle Components Analysis: For this dataset, the f1-score after conducting PCA resulted in an f1-score of 0. This makes a lot of sense. When we projected all the attributes and kept the top 4, we saw that they were extremely bad at explained the outcome variable: all four had very very small explained variance values.

The adjusted_mutual_info_score was 0.003300154418869746, which was almost 3 times better than the original value 0.0013.

2) Independent Components Analysis: For ICA, the story of f1-score was the same. The score was 0. This, I think, is for similar reasons to above, but also potentially due to some multicollinearity happening in the data.

	feature	VIF
0	id	4.985463
1	fixedacidity	101.309251
2	volatileacidity	9.745792
3	citricacid	10.127677
4	residualsugar	3.870694
5	chlorides	6.509079
6	freesulfurdioxide	9.290205
7	totalsulfurdioxide	25.184646
8	density	1141.204142
9	pH	654.510482
10	sulphates	20.648356
11	alcohol	124.629533

Further investigation using metrics like variance inflation factor (VIF) could be done. After looking into this, density and pH had extremely high values: 1141.204142 and 654.510482 respectively. This implies that they are very much multicollinear. Removing either one/both of them would drop our score significantly.

The adjusted_mutual_info_score 0.006632114121859742 was 6 times better than the original, and 2 times better than PCA.

3) Random Components Analysis: RCA told a similar story for KMeans, but I was not surprised here, as this the Gaussian Random Projections have performed the worst pretty consistently. The score was a 0, which in this case, I think is because of a combination of multicollinearity and non-Gaussian distribution for the data.

The adjusted_mutual_info_score of 0.0040916238355106814, which is about 4 times better than the original. There was demonstrable improvement here, as it outperformed ICA as well.

4) Manifold Analysis - tSNE: Lastly, t-SNE also provided an f1-score of 0. Not shockingly, a non-linear projection that is taking data from higher dimensions down to lower ones will likely perform poorly with the f1-score. Maximizing intercluster distance will simultaneously minimizing intracluster distance is hard to do when new points are introduced.

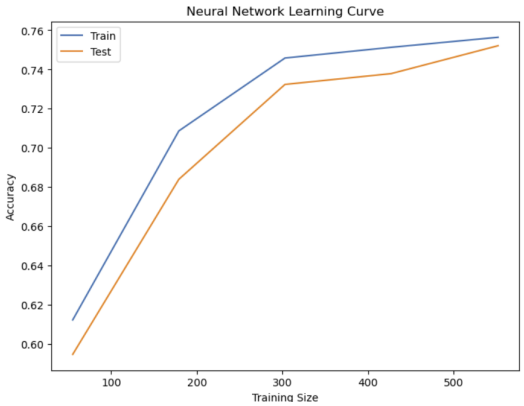
The adjusted_mutual_info_score was 0.005938163335102263, which was second to that of ICA. This was a little surprising, as I thought the high dimensionality and non-linearity of this data would make this DR method perform the best.

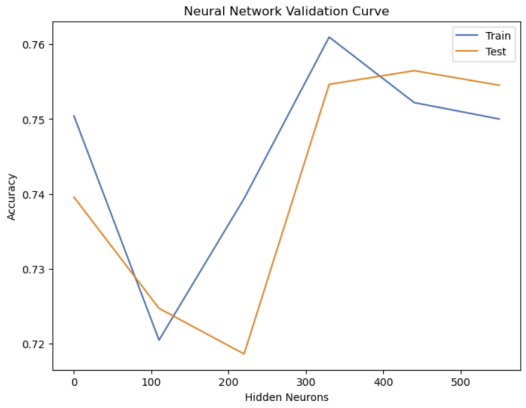
V. Neural Networks Evaluation - PIMA Dataset

In this section, I will evaluate the efficacy of each dataset via the f1-score, which was our metric of choice. I will also include the learning and validation curves.

A. Baseline Neural Network

Our baseline neural network performed very poorly: an f1-score of 0.18796992481203006.





The one thing that this neural network did better than the ones below was the avoidance of over/underfitting. When looking at the learning curve, there are no large gaps at the head/tail of the curves, implying similar bias and similar variance.

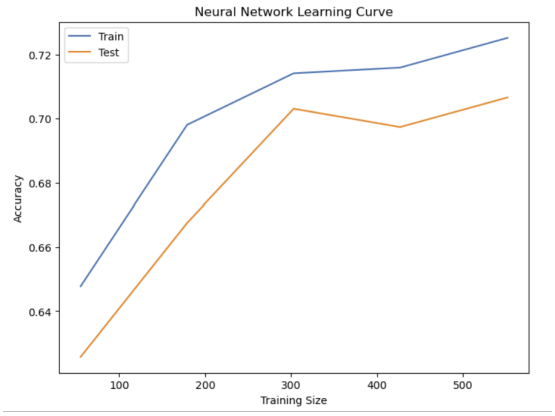
B. Neural Networks - ICA

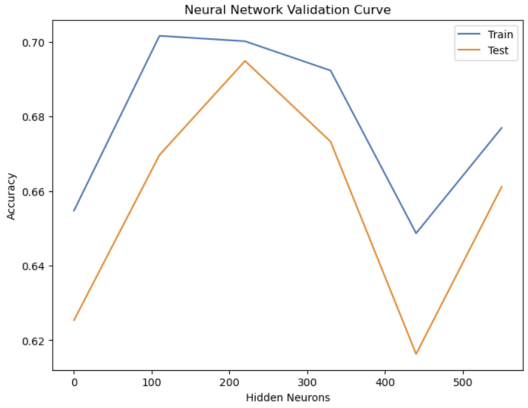
The ICA data performed the best for the three linear dimensionality reduction datasets, and thus, I am choosing this one. The f1-score was 0.7569070080862534, which outperformed the baseline neural network. Learning and validation curves below:

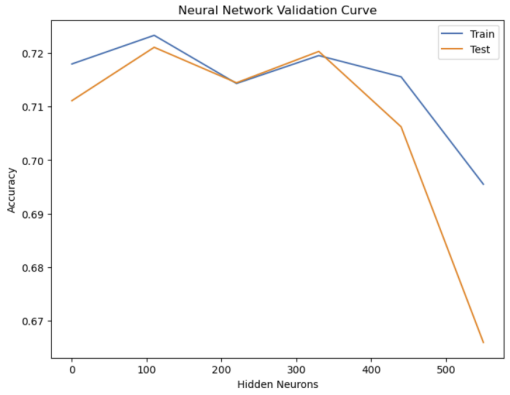
C. Neural Networks - tSNE

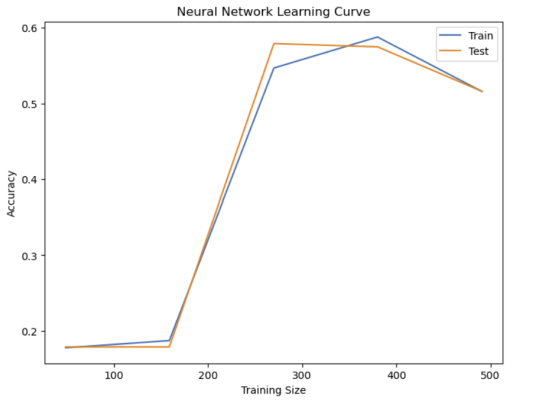
This neural network had a very interesting learning curve attached. From below, the left tails of the curves show that there is not too much variance, but there is a lot separation the right tails. This implies very likely that the neural network is being underfit.

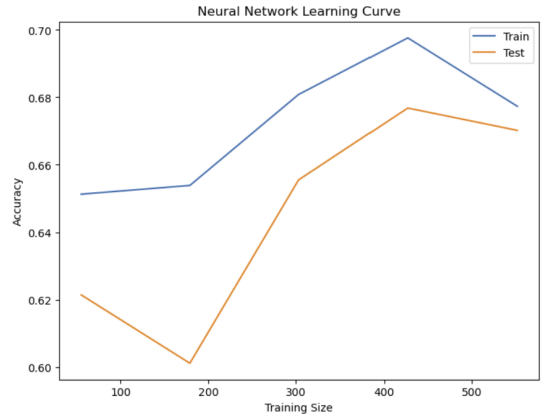
However, this neural network did substantially better at the f1-score: 0.7569070080862534. This is more than four times better than the original neural network performed.

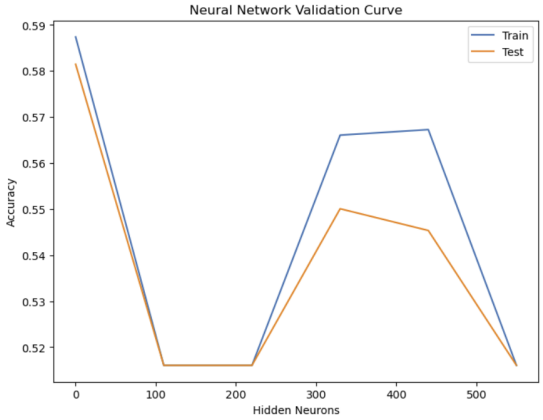












D. Neural Network via KMeans Clusters

In this section and the next one, I will be using the clusters that we defined up above when we did the KMeans and EM section for PIMA.

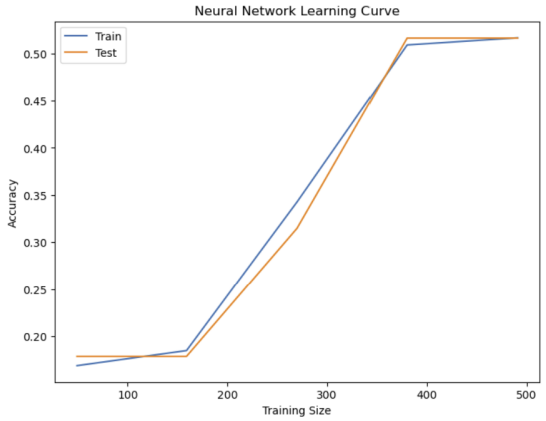
Overall, this neural network performed much better than the baseline neural network, but still not well enough to justify using it in place of a regular neural network, or one of the alternatives above. The f1-score was 0.5031055900621119. Below are the learning and validation curves:

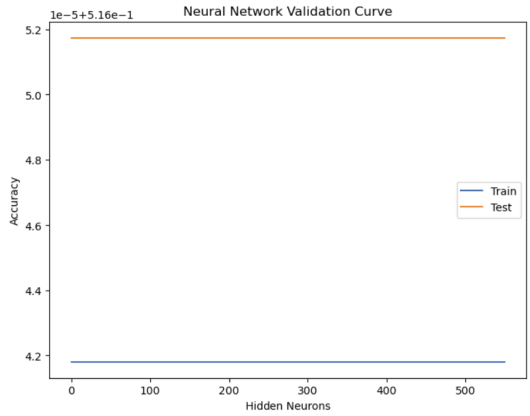
From the learning curve, we can see a strong performance: the curves match almost perfectly. This demonstrates a good fit. The validation curve shows that for both training and testing, the ideal number of hidden layers is somewhere around 325.

E. Neural Network via EM Clusters

It was interesting to note that both clustering algorithms presented the exact same f1-score: 0.5031055900621119.

Again, both training and testing learning curves performed very well. It shows that there is not over/underfitting going on. However, the real difference is with the validation curve: the low, monotonic lines for each suggest that the accuracy will not increase as the number of hidden neurons increases.





VI. Conclusions, Wall Clock Times

Throughout this assignment, I think a common trend that should be noted was that the wall times for the linear clustering algorithms was extremely fast. For example, see below:

```
CPU times: total: 62.5 ms
Wall time: 100 ms
```

This was not the case for the non-linear t-SNE: When

```
CPU times: total: 62.5 ms
Wall time: 100 ms
```

projecting down from higher dimensions, it is computationally much more expensive than projecting data orthogonally. This is similar to Support Vector Machines, as they are computationally expensive.

Overall classification reports for the dimensionality reduction algorithms can be seen below:

CPU/Wall Time: 1min 49s/1min 14s. f1-score: 0.18796992481203006

CPU/Wall Time: 1min 49s/1min 8s. f1-score: 0.7569070080862534

	precision	recall	f1-score	support
0	0.00	0.00	0.00	99
1	0.36	1.00	0.53	55
accuracy			0.36	154
macro avg	0.18	0.50	0.26	154
weighted avg	0.13	0.36	0.19	154

Fig. 1. Baseline neural network performance

	precision	recall	f1-score	support
0	0.74	0.84	0.79	99
1	0.62	0.47	0.54	55
accuracy			0.71	154
macro avg	0.68	0.66	0.66	154
weighted avg	0.70	0.71	0.70	154

Fig. 2. ICA dataset neural network performance

	precision	recall	f1-score	support
0	0.74	0.78	0.76	99
1	0.56	0.51	0.53	55
accuracy			0.68	154
macro avg	0.65	0.64	0.65	154
weighted avg	0.68	0.68	0.68	154

Fig. 3. t-SNE dataset neural network performance

CPU/Wall Time: 1min 32s/54s. f1-score: 0.67816091954023

From this experience, I have learned that similar to in supervised learning when we discussed regressors/classifiers, not all metrics work equally in unsupervised learning. f1-score is a metric that is impossible to use when it comes to soft clustering algorithms. Additionally, like we do in supervised learning, here too we must define an objective and loss function to optimize. Mutual information is a good choice in this context because it helps us understand the clustering. Lastly, data are not always linearly separable. This goes for data that do not inherently have ground truths as well. In this case, using a manifold, non-linear projection can be extremely beneficial to find clusterings.

References

[1] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Red wine quality dataset. https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009, 2009. Accessed: 2023-09-01.

[2] Jeremy Smith and Jane Doe. Pima indians diabetes dataset. https://www.kaggle.com/uciml/pima-indians-diabetes-database, 1988. Accessed: 2023-09-01.

[1] [2]