Assignment 3

Joe Kraemer

I. Datasets

A. Diabetic Retinopathy

This dataset contains features extracted from Messidor images and the goal is to predict whether an image contains signs of diabetic retinopathy [1]. The class labels are binary. There are 19 numeric features and about 1100 data points. This data set does not have any linearly dependent features. The data set is quite balanced with a class distribution of 46.9% and 53.1%.

Because this dataset is related to identifying key health issues, I would like to prioritize final classification accuracy within dimensionality reduction to hopefully lead to models with a similar accuracy scores as the original data with less variance and less bias.

B. Red Wine

The UCI Red Wine dataset has 11 continuous features and 1600 data points [2]. The features describe various chemical attributes of the wine such as ”citric acid”, ”residual sugar”, and ”chlorides”. The goal is to predict the quality of the wine with a score of 1 – 10. However, there are only 6 different scores represented in the dataset. This data set does not have any linearly dependent features. This data set is unbalanced and a majority of the samples land in the middle of the class distribution. Given the class labels: [3 4 5 6 7 8], the respective class distribution %: [0.6, 3.3, 42.6, 39.9, 12.4, 1.1].

Initially I checked to see if there were any strong correlations between the different features. I expected to see strong correlations between the different types of acid, and the overall pH level because this is a measure of acidity. I also expected a strong correlation between alcohol content and density because I know home brewers measure alcohol content of their beer using density measurements. As I predicted, alcohol content and density does have quite a high correlation of 0.5 but ”fixed acidity” was actually higher with .67. Also as predicted, ”citric acid” and ”fixed acidity” have high correlations with pH but ”volatile acidity” does not.

In general with this dataset, I would like to prioritize feature reduction within dimensionality reduction to hopefully lead to models with significantly faster train times.

II. Data Pre-Processing

We split the dataset into a test and train set. The training set will be used for hyper-parameter validation. The training set will be used for dimensionality reduction and clustering analysis to avoid biasing the modified datasets to the test data when applying the neural networks. The data is scaled based on the training data and then applied to the test data so that the neural networks can use the data properly. Because we are only scaling based on the training data, we are not biasing to the test data.

We use a weighted f1 score to reduce bias partially unbalanced fold. We will still use stratified shuffle folds cross validation methods. Stratified to ensure that there is an equal proportions of labels in the test and training folds. Shuffle so that we can create unlimited amounts of folds with larger samples per folds.

III. Clustering

A. Diabetic Dataset

First, we must select the optimal number of clusters. In Figure 1, we can see a plot of 2 different performance metrics that I used to select the optimal number of clusters for Expectation Maximization. We can see that for a cluster size of 2, the silhouette score is significantly higher than any other cluster. Looking at a silhouette plot, I noticed the clusters seem to be sized about the same and appear to have good separation. The Bayesian information criterion (BIC) disagrees with the silhouette score. BIC is a measure of the log likelihood of the model but includes a penalty for the complexity of the model. It is the likelihood of the model which is causing the BIC score to be so high. Looking at the log likelihood values shows that it is actually negative at this point. Because of the high BIC score, I decided against selecting a cluster size of 2. Instead I chose a cluster size of 11 because this point minimizes the BIC score and it also has a high silhouette score if we are ignoring a cluster size of 2.

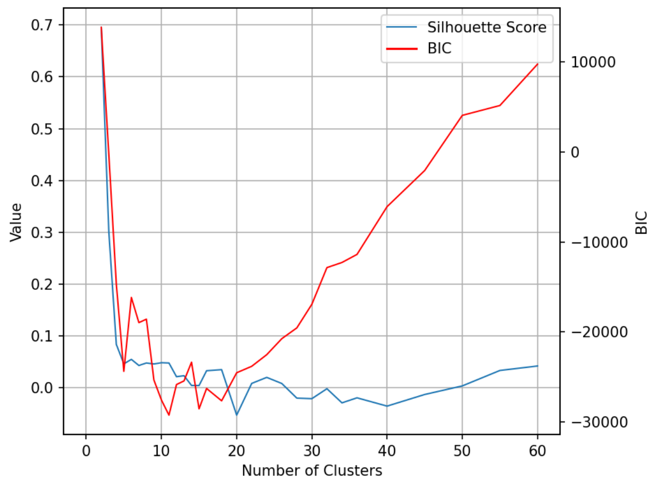


Fig. 1: Plot of Various Performance Metrics vs Number of clusters for Expectation Maximization on the Diabetic dataset.

When selecting number of clusters for K-Means, I looked at the silhouette score and the Sum of Square Error (SSE) where error is the distances of samples to their closest cluster center. When increasing the number of clusters, the silhouette score does not continue to decrease (Figure 2). This is different than with Expectation Maximization which continued to decrease and eventually became negative. Because of this, I think that it choosing a higher number of clusters could be better because it will reduce SSE (which results in tighter, more cohesive clusters) and has little penalty on the silhouette score. There is some risk with adding too many clusters, like the model could start to overfit. With too many clusters, there is no value in the clustering as we could just be representing noisy data and identifying relationships that actually don't exist. For this reason, I chose a cluster size of 7 as this maintains a relatively high silhouette score (83% of the max value), but still reduces the SSE to 52% of its max value. It also should still provide good generalization as the number of clusters is not too high.

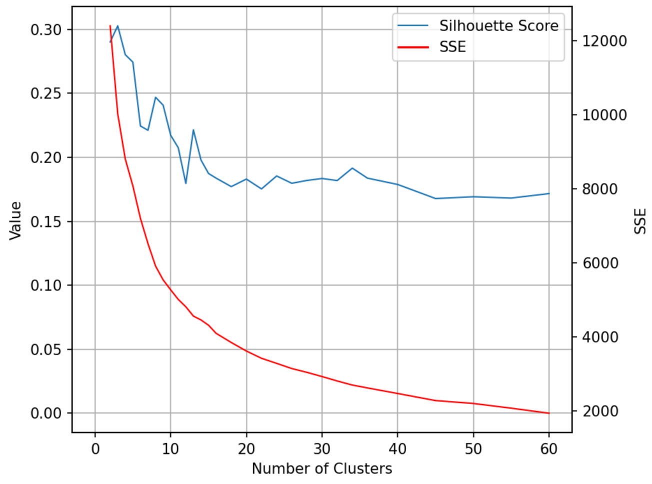


Fig. 2: Plot of Various Performance Metrics vs Number of clusters for K-Means on the Diabetic dataset.

After the number of clusters were selected for each algorithm and dataset, I evaluated the clusters using various supervised metrics (Table I). I used homogeneity which is a measure of how many data points within a cluster belong to a single class. I also used completeness which measures how many of the total data points of a class fall within a given cluster. Finally I looked at Rand Index and Adjusted Mutual Information (AMI). Rand Index is a measure of similarity between two different label sets, in this case, the true labels and the cluster labels. AMI is is a measurement of agreement between two clusterings that accounts for the chance of agreement based on random chance.

In the Diabetic dataset, the scores are quite similar, but EM seems to have a slight edge as it has higher scores in all categories except for homogeneity. Given that the diabetic dataset is binary, a homogeneity score of 0.5 is actually quite terrible because this means there are on average an equal amount of each class. This is not necessarily a bad thing though. This just shows that the clusters are exploring some other underlying structure in the data and not evaluating what features are relevant to label a patient as diabetic. This is okay and expected. The clustering algorithm is working in an unsupervised way, so expecting the outcome to match the labeled data is unrealistic. I think that the relationship between the labels and the features is complex. The unsupervised learning algorithms are finding more obvious relationships between the features.

I created pairwise plots of the different cluster labels, but the results were uninteresting and I found that there was little to no separation of the labels while looking at them in a two-dimensional spaces. Instead I created a t-SNE plot to visualize the labels of the two clustering algorithms. In the Diabetic dataset, the most interesting thing to me was that the EM algorithm was able to more completely label the singular cluster on the left, while K-Means has three different labels in that cluster.

Algorithm	AMI	Rand Index	Homogeneity	Complete
EM - RedWine	0.544	0.080	0.615	0.110
K-Means - RedWine	0.554	0.092	0.619	0.127
EM - Diabetic	0.619	0.048	0.505	0.096
K-Means - Diabetic	0.598	0.040	0.507	0.067

TABLE I: Table of Performance metrics of clusters produced by Kmeans and GMM for the Diabetic and Red Wine dataset.

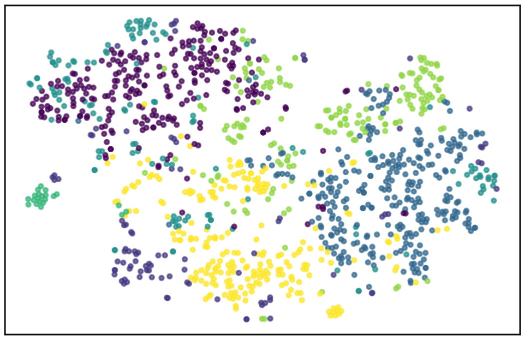


Fig. 3: Plot of Expectation Maximization Clustering labels on t-SNE reduced feature plane on the Red Wine dataset.

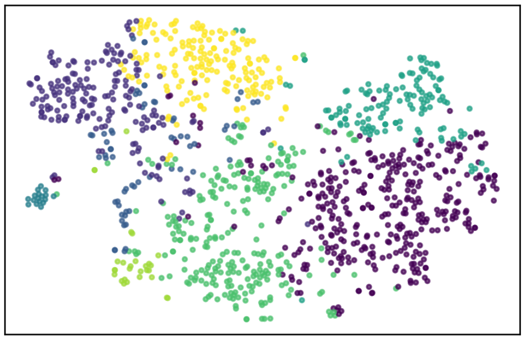


Fig. 4: Plot of K-Means Clustering labels on t-SNE reduced feature plane on the Red Wine dataset.

B. Red Wine Dataset

Looking at Figure 7, there seems to be a similar behavior with 2 clusters that was seen in Figure 1. The silhouette score

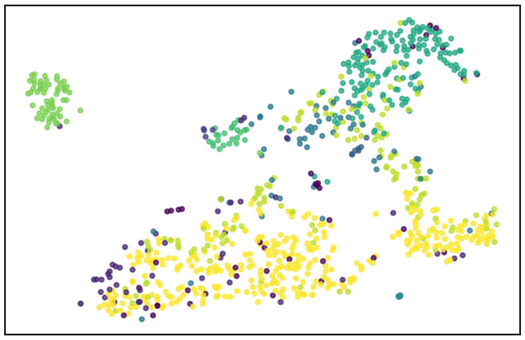


Fig. 5: Plot of Expectation Maximization Clustering labels on t-SNE reduced feature plane on the Diabetic dataset.

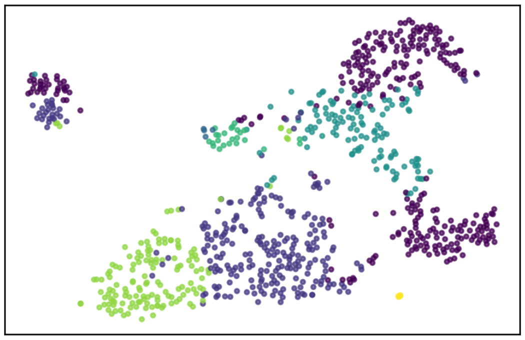


Fig. 6: Plot of K-Means Clustering labels on t-SNE reduced feature plane on the Diabetic dataset.

is originally very high but then drops dramatically, as well as the BIC score being high and then dropping. The BIC score behavior is not as dramatic here which means that the log likelihood of the model is likely not as negative as it was with the Diabetic dataset. I would like to balance the likelihood of the model with the silhouette score and so I chose 7 clusters. There is a nice bump in silhouettes score at this point with a low BIC score.

For K-Means and the Red Wine dataset, the curves behave similarly to the diabetic dataset. In Figure 8, the silhouette score plateaus as the SSE continues to drop. There is a sharp decline in silhouette score at 8 clusters. To keep a high silhouette score while reducing SSE as much as possible, I chose 8 clusters for K-Means and the Red Wine dataset.

In the Red Wine dataset, again the scores are quite similar, but Kmeans has a slight edge instead of EM (Table I). The homogeneity score is higher for both algorithms than that the Diabetic dataset which is significant because there are more true labels in the Red Wine dataset. It is interesting that the Rand Index is higher in Red Wine vs the Diabetic but the AMI is the opposite. This suggests that random chance of agreement is higher in the red wine dataset and is being accounted for negatively.

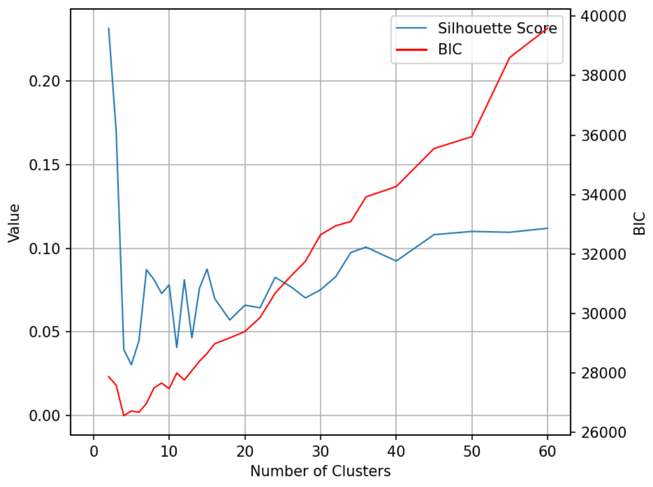


Fig. 7: Plot of Various Performance Metrics vs Number of clusters for Expectation Maximization on the Red Wine dataset.

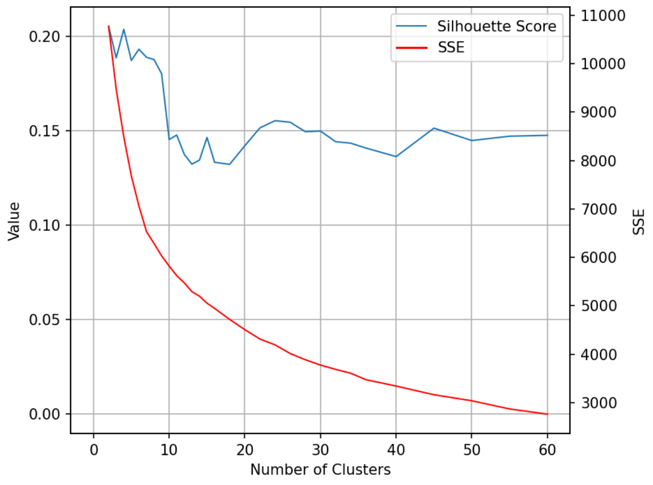


Fig. 8: Plot of Various Performance Metrics vs Number of clusters for K-Means on the Red Wine dataset.

IV. Dimensionality Reduction

A. Diabetic Dataset

In Figure 9, the knee value seems to be misleading as the curve is jagged and actually, with more than 3 components, we can continue to significantly increase kurtosis. In this case, we would like to maximize kurtosis so that the output contains more independent features. In this case, I would pick 15 components. This will still reduce the features space by 2 and I anticipate that this will still retain a majority of the original information and lead to similar accuracy scores.

In Figure 10, the explained variance curve drops significantly over the first 4 components and eventually drops to near zero at 15 components. This means that with just 4 linear combinations of the dataset, we are able to account for a large portion of the variation in the dataset. This could implies that the other features have some linear correlation with the other components. 4 components could be a plausible

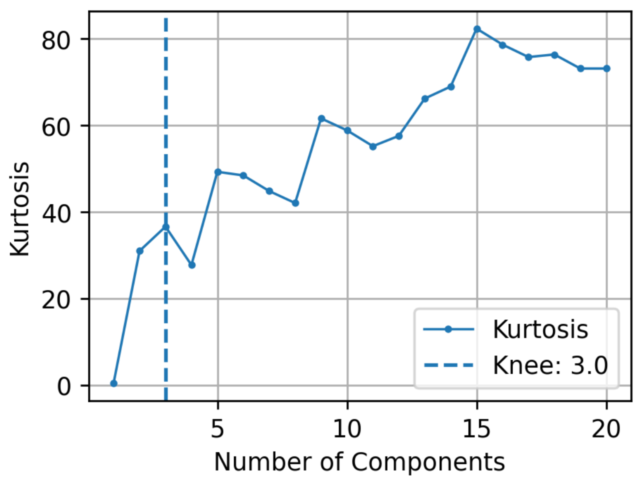


Fig. 9: Plot of Kurtosis vs Number of Components for ICA on the Diabetic dataset.

selection if the primary goal was to reduce the complexity of the dataset to reduce fit times. At 4 components, there is still significant variance which means that we are filtering relevant information from the original dataset and this may negatively affect prediction scores of classification models. Given that this dataset is medically related, I think that it make sense have more components to further reduce the explained variance even though the added complexity could negatively affect fit times, it should keep accuracy high. For this reason, I will pick 13 components as the variance is near zero.

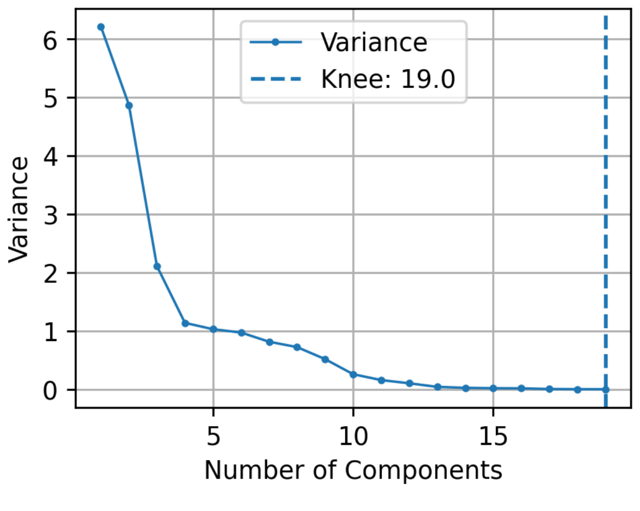


Fig. 10: Plot of Explained Variance vs Number of Components for PCA on the Diabetic dataset.

For selecting the number of features for Randomized Projections I looked to minimize the reconstruction error while increasing the pairwise distance correlation coefficient. .The reconstruction error continues to decrease roughly linearly, making it difficult to select a number of components based on some sort of knee (Figure 11). Looking at the pairwise distance correlations (Figure 12), we can see a more clear point of diminishing returns. In this case, I will use the selected knee point of 5 features.

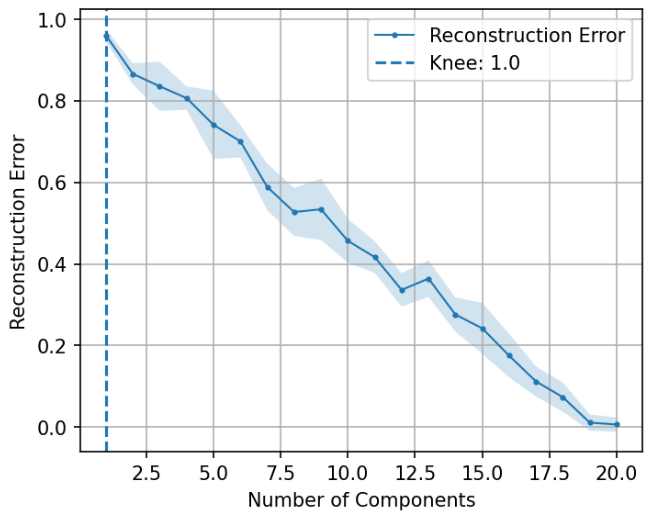


Fig. 11: Plot of Reconstruction Error vs Number of Components for RP on the Diabetic dataset.

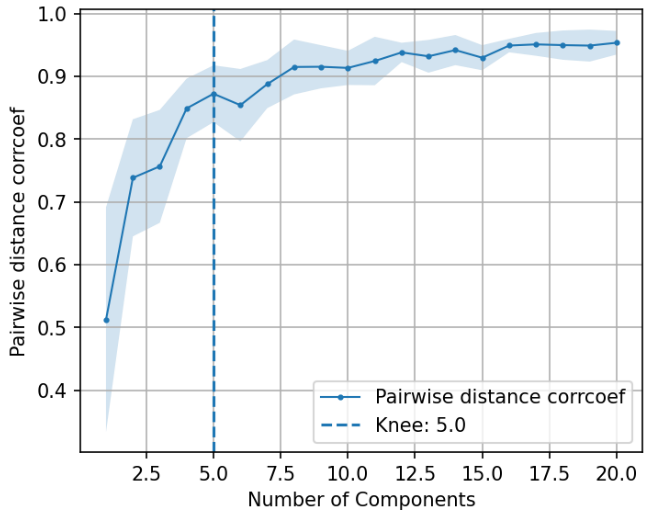


Fig. 12: Plot of Pairwise distance correlation coefficient vs Number of Components for RP on the Diabetic dataset.

For the last dimensionality reduction method, I chose to select features based on the permutation importance. I picked permutation importance over impurity based importance because it has less bias to from the cardinality of the features. I used an untuned random forest classifier to get accuracy scores. In Figure 13, we can see a box plot of the permutation importance of the features of the Diabetic dataset. We can see that after feature 11, the returns decrease significantly. If this dataset was not medically related, perhaps I would chose up to feature 11. However, I chose to include 10 features, from 2 to 1 (1,16,15,9,6,11,17,14,8,2). After feature 1 the accuracy decreases are quite close to zero and include noise which leads me to believe that these features are not statistically significant.

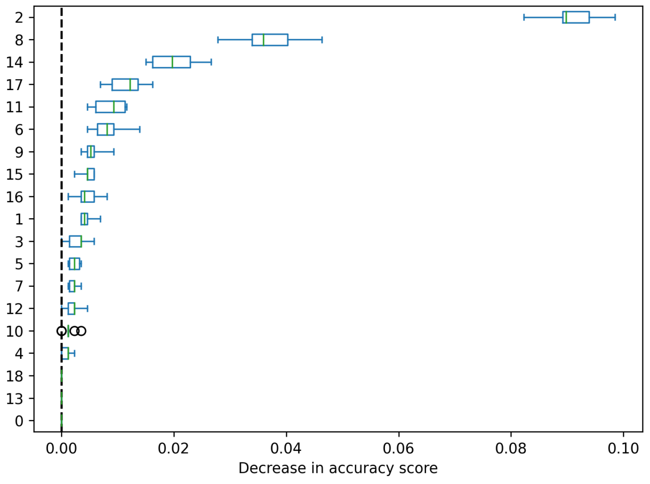


Fig. 13: Plot of permutation importance of the Diabetic dataset.

After selecting the number of components for each method, I applied a dummy classifier (in this case, an untuned DecisionTreeClassifier with all default values) to see if there were noticeable differences in fit times and accuracy (Table II). In all of the dimensionality reduction methods, we saw increases in test scores. I would expect different results if a fully tuned classifier was used, but this is promising as it doesn't seem like the reduction in features has negatively affected the scores. Additionally, all algorithms except for PCA resulted in a decrease in fit time. I expected feature importance to have the best scores and I am surprised that RP was able to beat it. I think that there could have been some luck with the test data where the relationships we expected to be important based on the cross validation in the training data were not present in the test split. At this point I would call the dimensionality reduction a success as we have helped alleviate the curse of dimensionality while maintaining high amounts of information in our transformed datasets.

Algorithm	Test Score	Fit Time (ms)
Original	0.530	10.617
ICA	0.594	7.605
PCA	0.611	11.598
RP	0.643	4.569
FI	0.588	5.023

TABLE II: Table of Dummy Classifier Performance of the original Diabetic dataset as well as modified datasets after running various dimensionality reduction methods.

B. Red Wine Dataset

In Figure 14, the knee value appears to be in the correct place on this curve and appears to be a good choice for number of components. The knee point does not coincide with the global maximum of the curve (n=8). It is tempting to pick 8 components, but given that this is almost double the components (from 5 to 8), it is not worth it for just 8% increase in kurtosis. So for ICA on the Red Wine dataset, I will choose 5 components.

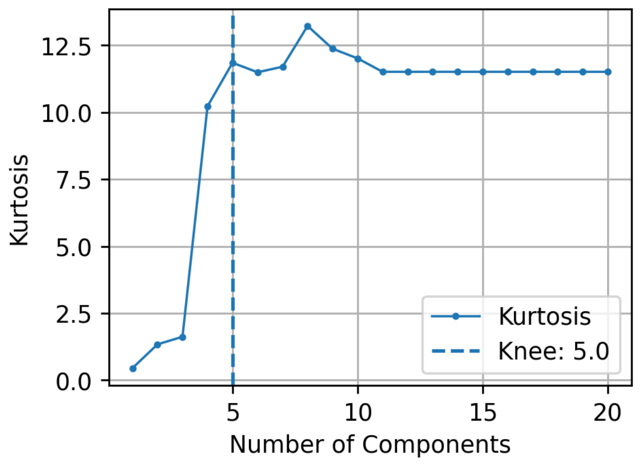


Fig. 14: Plot of Kurtosis vs Number of Components for ICA on the Red Wine dataset.

In Figure 15, we can see that there is no clear knee in this curve, the variance continues to decrease significantly with each additional component. With each additional principle component, it is difficult to find directions with high variance. This means that there is relatively equal variance in all dimensions. This means that the features of this dataset are not as linearly related as the features of the diabetic dataset. In this case I will have to select a threshold. Since I am prioritizing feature reduction, I want to pick a point that significantly reduces this. At 6 components, there appears to be a change in slope with how much variance is reduced with the subsequent components. Additionally, 6 would result in a significant feature reduction and thus accomplishing my goal.

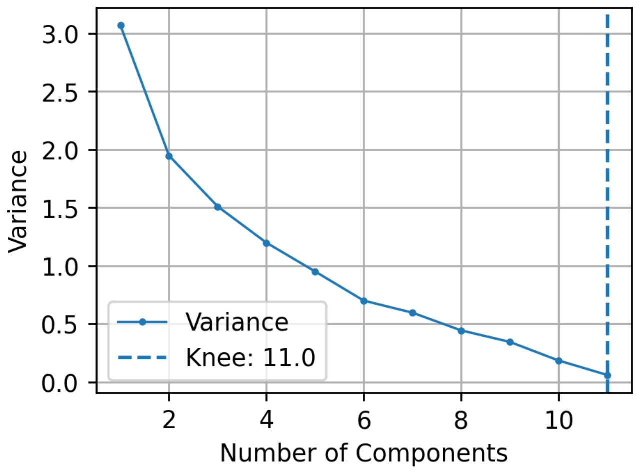


Fig. 15: Plot of Explained Variance vs Number of Components for PCA on the Red Wine dataset.

In Figure 16, we can see that the reconstruction error behaves in a similar way to the diabetic dataset. The error decreases linearly with the number of components. Because there is no clear knee in the reconstruction error, I decided to look at the pairwise distance correlation coefficient. In Figure 16, we can see a clear diminishing returns in the chart and I decided to take the calculated knee value of 4 components. I think that this aggressive reduction should yield interesting results as the number is so low (almost 70% reduction in components).

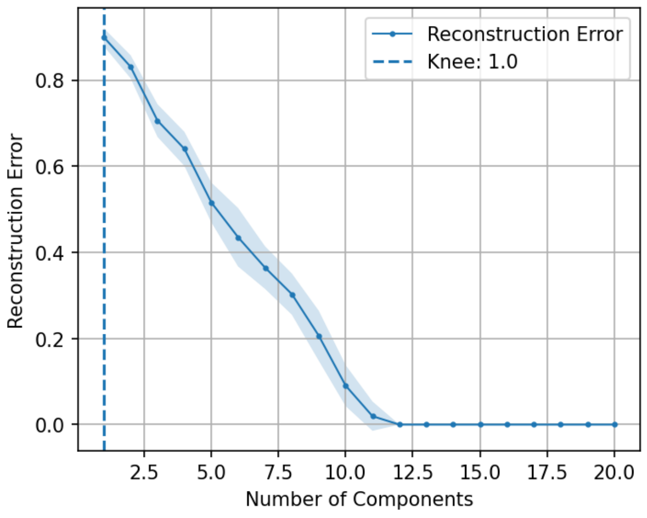


Fig. 16: Plot of Reconstruction Error vs Number of Components for RP on the Red Wine dataset.

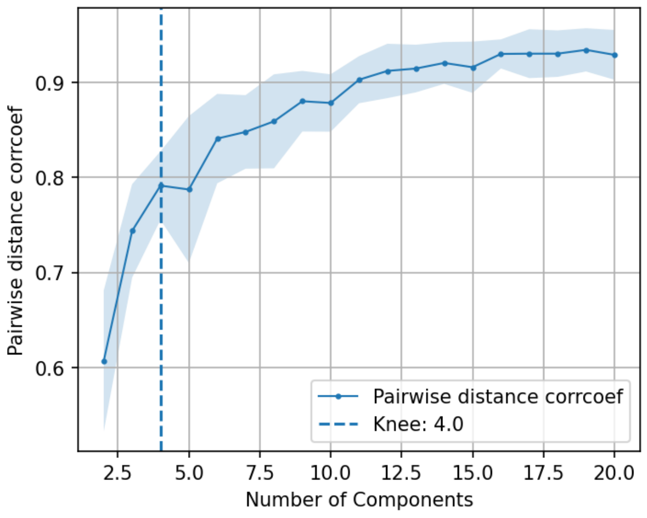


Fig. 17: Plot of Pairwise distance correlation coefficient vs Number of Components for RP on the Red Wine dataset.

In Figure 18, we can see that features 10, 9 and 1 are hugely important. We can see that the accuracy returns diminish greatly after feature 7. I selected from 8 total features, from 10 to 4 (4,7,2,6,1,9,10). I chose to include 4 because the standard deviation was low and so its inclusion seems like a move guaranteed increase in accuracy.

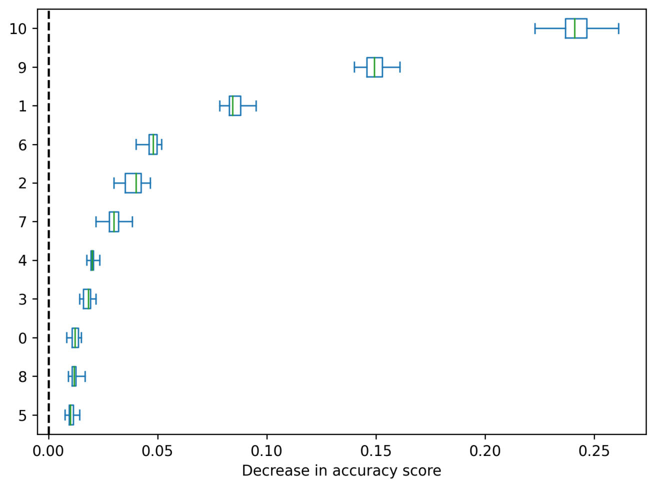


Fig. 18: Plot of permutation importance of the Red Wine dataset.

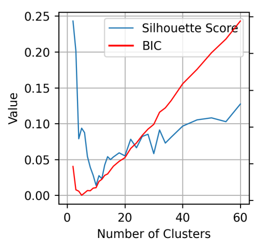
Just like with the diabetic dataset, after selecting the number of components for each dimensionality reduction method, I used a dummy classifier to get a heuristic for the changes in test scores and fit times (see Table III). We can see that all of the reductions seem to have reduced the fit time. All algorithms have not negatively affected the test scores except for RP. This is partially explained by the small number of components which is likely not capturing all of the relevant information with just 4 components. Unlike the diabetic dataset, we see that FI is better than RP in terms of test score and fit time. This is impressive as FI has more features than RP. This also reinforces my hypothesis that the test split of the diabetic dataset has some samples with feature relationships that are not well represented in the training data. As with the diabetic dataset, it appears that PCA and FI have increased the test scores. However, unlike the diabetic dataset, ICA was not able to increase the test scores.

Algorithm	Test Score	Fit Time (ms)
Original	0.585	10.268
ICA	0.583	8.364
PCA	0.604	7.253
RP	0.527	8.501
FI	0.601	4.775

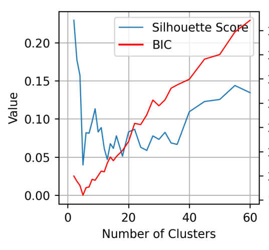
TABLE III: Table of Dummy Classifier Performance of the original Red Wine dataset as well as modified datasets after running various dimensionality reduction methods.

V. Clustering of Dimensionality Reduced Datasets

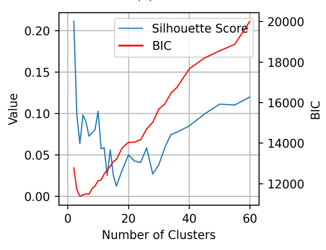
After performing dimensionality reduction with four algorithms, the two clustering algorithms were run again on the dimensionally reduced data from both datasets. With ICA and K-Means, we can see that in both the red wine dataset and the diabetic dataset, the initial peak in silhouette score is eliminated (Figure 20a, 22a) . This could mean that the original distinct clusters that existed in the original dataset were projected onto a new plane and were eliminated. We don't see a similar thing happen with ICA and Expectation Maximization



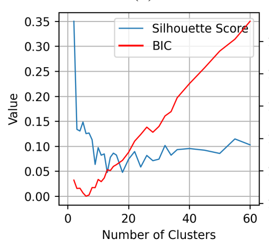
(a) ICA



(b) PCA

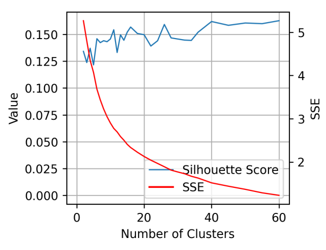


(c) RP

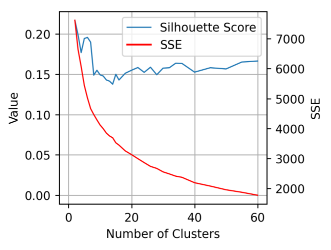


(d) FI

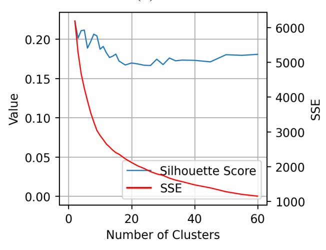
Fig. 19: Plot of Various Performance Metrics vs Number of clusters for Expectation Maximization on the Red Wine dataset after dimensionality reduction. Each subplot represents a different dimensionality reduction method.



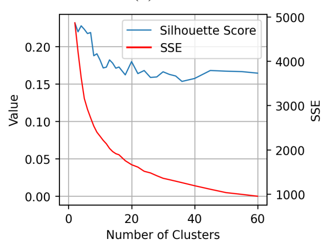
(a) ICA



(b) PCA



(c) RP



(d) FI

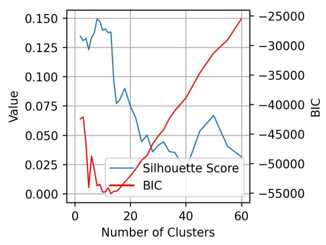
Fig. 20: Plot of Various Performance Metrics vs Number of clusters for Kmeans on the Red Wine dataset after dimensionality reduction. Each subplot represents a different dimensionality reduction method.

and this could be because expectation maximization doesn't assume the signal is isotropic like K-Means. However when ICA is applied to the diabetic dataset, we see a lot more variance in the silhouette score and the BIC score. This could hint that the clusters created by ICA are closer together and so when the clustering algorithm has to place another cluster, they en up close to one another and so the scores are lower. We would expect to see a similar thing happen when clustering is applied to a PCA reduced dataset. Since PCA is reducing variance, we would expect to see a reducing in log likelihood as with less variance the statistical chance of of a cluster should increase. In all plots, we see a huge reduction in that initial spike in log likelihood like expected.

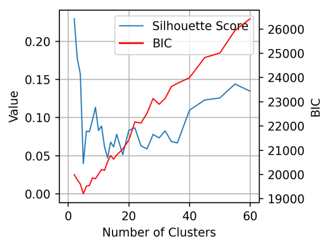
VI. Performance of Dimensionality Reduction

To analyze the benefits of dimensionality reduction, I will compare the performance of neural networks on the original Diabetic dataset and the dimensionality reduced dataset. Using the best number of dimensions that I previously identified in Section IV, I performed a grid search of various hyper parameters for the neural network using the processed training data so that the model was properly tuned for the dimensionally reduced dataset. The values for the hyper parameter sweep can be found in Table V. This step was repeated for all four of the dimensionality reduction methods.

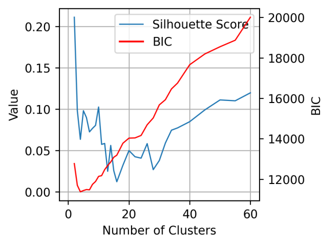
Then the optimized neural networks were trained and tested on the full dataset. The test scores and fit times (average of 100 runs) can be found in Table IV. I expected FI to perform better than PCA and ICA because FI is a supervised selection method. PCA is attempting to explain variance, but it could be attempting to explain variance in a feature that is totally irrelevant to the classification labels. In this way, I thought that FI would be more efficient at eliminated features unrelated to the labels. In terms of overall fit times, the clustering and



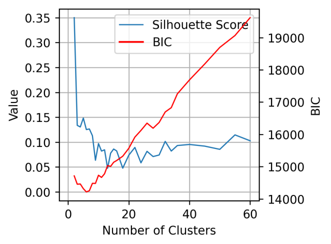
(a) ICA



(b) PCA



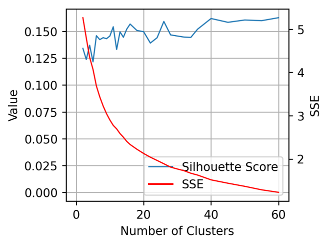
(c) RP



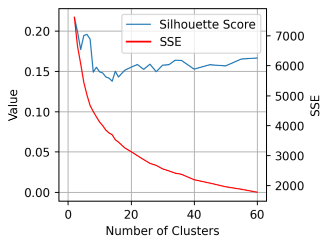
(d) Fi

Fig. 21: Plot of Various Performance Metrics vs Number of clusters for Expectation Maximization on the Diabetic dataset after dimensionality reduction. Each subplot represents a different dimensionality reduction method.

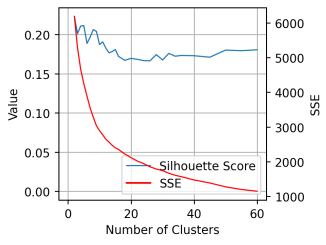
dimensionality reduction methods were a huge success. They all reduced the fit times by at least half, and ICA reduced by almost an order of magnitude. When I was selecting hyperparameters for the neural networks, I was worried when the hidden layer architectures were not as simplified as I originally thought the would be. The curse of dimensionality goes further than that though as the input layers are still larger and requires more back prop loss calculations. Additionally, some of the dimensionality reduction methods also alter how much variance is coming through each feature. This creates a more consistent signal and makes optimizing the model much faster as they can use higher learning rates. This was



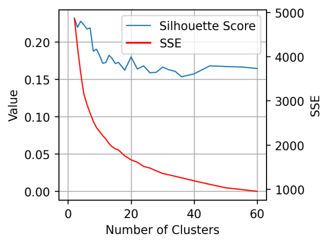
(a) ICA



(b) PCA



(c) RP



(d) FI

Fig. 22: Plot of Various Performance Metrics vs Number of clusters for Kmeans on the Diabetic dataset after dimensionality reduction. Each subplot represents a different dimensionality reduction method.

all done with keeping the test score relatively close to the benchmark score using the original dataset. This shows that more data isn't always better for creating accurate models. I'm especially impressed with FI as it was able to increase the score compared to the benchmark. I think that by removing not relevant features, the neural network is not affected by extraneous noisy signals which could cause the neural network to optimize to local minima.

Algorithm	Test Score	Fit Time (sec)
Original	0.736	9.55
ICA	0.695	1.75
PCA	0.688	4.32
RP	0.698	4.44
FI	0.747	4.05
EM	0.459	1.02
K-Means	0.604	0.81

TABLE IV: Table of Neural Network Performance of the original Diabetic dataset as well as modified datasets after running various dimensionality reduction and clustering methods.

When looking at the dimensionality reduction learning curves, we can see that that the FI (Figure 26) has a really low variance, but a rather high bias when compared to the other learning curves. This makes sense because the feature selection has biased the neural network to a select set of features. I was also impressed that ICA has such little variance. Of all of the dimensionality reduction methods, it has the lowest variance which make sense as the core of this algorithm is to reduce the amount of noise and extract true signals. RP is also quite interesting as it has a great reduction in the overall bias. While the other methods struggle with bias at low sample size, RP does not. This is also likely partially due to the simpler neural network architecture which will inherently be less prone to bias.

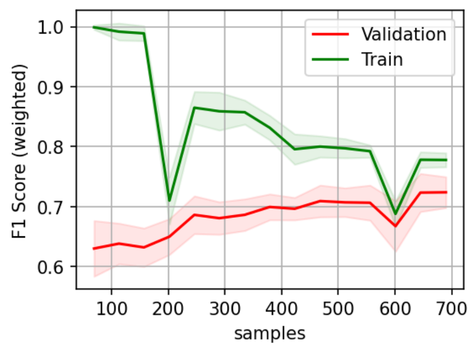


Fig. 23: Learning Curve of a neural network on the Diabetic dataset after being dimensionally reduced by ICA (n_components=15). Neural Network parameters: hidden_layer_architecture=[25], alpha=0.0001, learning_rate = 0.01

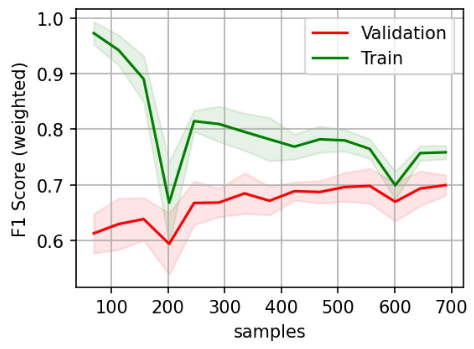


Fig. 24: Learning Curve of a neural network on the Diabetic dataset after being dimensionally reduced by PCA (n_components=14). Neural Network parameters: hidden_layer_architecture=[5, 5], alpha=0.0001, learning_rate = 0.001

VII. Performance of Clustering

To analyze the benefits of clustering, I will compare the performance of neural networks on the original dataset and a dataset with an additional feature which is the cluster labels. Similar to the performance analysis of dimensionality reduction in Section VI, I performed a grid search of the neural network hyper parameters for each of the clustering methods using the best number of clusters that I previously identified in Section III. The values for the hyper parameter sweep can be found in Table V.

Then the optimized neural networks were trained and tested on the full dataset. The test scores and fit times (average of 100 runs) can be found in Table IV. Because we are adding an additional feature here so I would think that the wall clock time would increase when comparing to the dimensionally reduced datasets.

I was surprised when I saw that the fit times were almost an order of magnitude smaller than the benchmark and faster than the other dimensionally reduced datasets. The two clustering algorithms which had a large negative effect on the score. This is because the additional feature clustering feature is not related to the label that the neural network is trying to classify and so it is just adding additional noise and making the

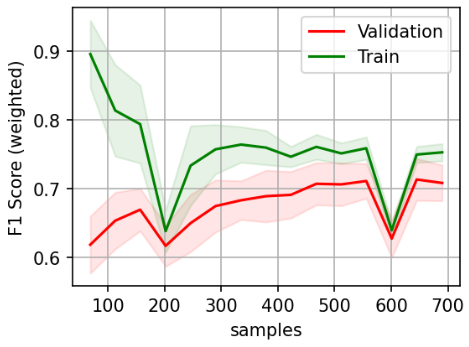


Fig. 25: Learning Curve of a neural network on the Diabetic dataset after being dimensionally reduced by RP (n_components=5). Neural Network parameters: hidden_layer_architecture=[5, 5], alpha=0.0001, learning_rate = 0.001

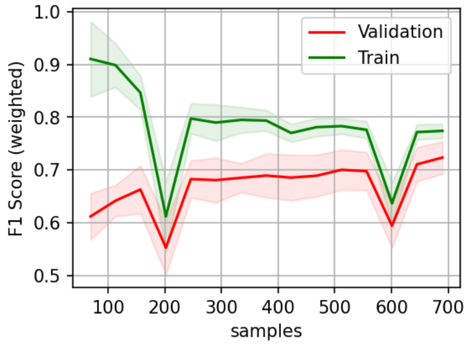


Fig. 26: Learning Curve of a neural network on the Diabetic dataset after being dimensionally reduced by FI (n_components=5). Neural Network parameters: hidden_layer_architecture=[5, 5], alpha=0.0001, learning_rate = 0.001

model worse. Looking at a learning curve of the two clustering methods, we can see that the EM clustering method has a much higher variance than K-means and the other methods. K-means does seem to have less variance than the EM model.

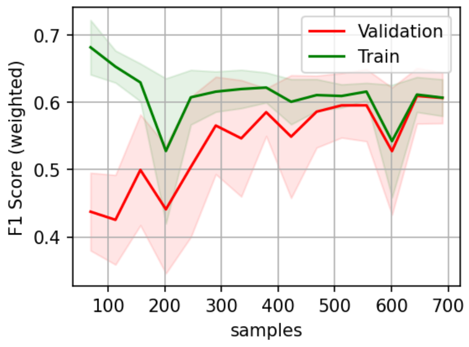


Fig. 27: Learning Curve of a neural network on the Diabetic dataset after clustering labels have been added as an additional feature using EM (n_clusters=11). Neural Network parameters: hidden_layer_architecture=[60, 60, 60], alpha=0.001, learning_rate = 0.01

VIII. Appendix

References

[1] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *CoRR*, vol. abs/1410.8576, 2014. [Online]. Available: http://arxiv.org/abs/1410.8576

[2] P. Cortez, A. L. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decis. Support Syst.*, vol. 47, pp. 547–553, 2009.

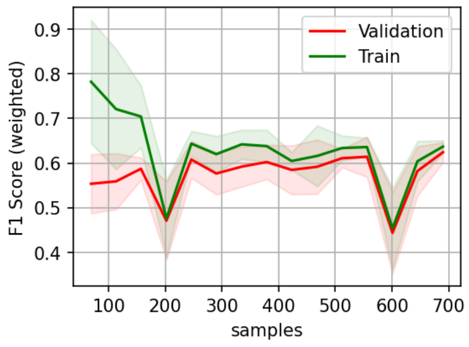


Fig. 28: Learning Curve of a neural network on the Diabetic dataset after clustering labels have been added as an additional feature using K-Means (n_clusters=7). Neural Network parameters: hidden_layer_architecture=[60, 60], alpha=0.0001, learning_rate = 0.01

Parameter	Tuning Range
initial learning rate	0.5 - 0.0001
number of nodes	5 - 60
number of layers	1 - 5
alpha	0.01, 0.0001

TABLE V: Table of hyperparameter default values as well as Grid Search tuning ranges.