CS7641 – Machine Learning

Assignment 3 – Unsupervised Learning and Dimensionality Reduction

Mason J. Kelchner
mason.kelchner@gatech.edu

1 INTRODUCTION

In this analysis, several techniques used in unsupervised learning are explored. Namely clustering and dimensionality reduction (DR) are used to determine likeness among data set samples to create more meaningful insight into the relationship between data features ultimately producing better prediction outcomes in a supervised learning algorithm. Several algorithms are implemented and applied to two data sets and their performance, likeness, advantages, and disadvantages are discussed on both data sets. Two clustering algorithms are implement: Expectation Maximization and K-Means. Four DR algorithms are implement, three of which are linear models including Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Randomized Projections (RP). The fourth DR algorithm is a nonlinear manifold learning algorithm which was selected as t-Distributed Stochastic Neighbor Embedding (t-SNE). Several experiments were run for each algorithm with the goal of disseminating how the clustering and DR techniques affect each data set. The two data sets used in this analysis were for classification problems. The first data set used was a binary classification of mushrooms being poisonous or edible based on 20 features with 61,069 instances. The second data set was a multi-classification of beans into 7 different beans based on 16 features, using 13,611 instances. Both are interesting and worthwhile problems for automatic identification systems. As the first data set is a binary classification using a mixture of continuous value and categorical features, its believed the clustering algorithms and DR techniques will be able to perform well in identifying partitions in the data for classification. Likewise, these clustered and feature reduced data sets will be able to be utilized in a neural network that will perform well for predicting the mushroom class on unseen test data. The second data set is multi-classification using all continuous value features; however, as the problem is for multiple classes, this will make the clustering and feature reduction techniques more susceptible to error. It isn't expected that the neural network trained on clustered or transformed data performs better than a normal multi-layer perceptron network using untransformed data.

2 CLUSTERING

Two clustering methods were implemented on the two data sets to explore their capabilities in partitioning the data set into the classes fundamentally represented using only the features and their mathematical variations. Unlike supervised learning, unsupervised learning does not use known labels or classes for training data; instead, the models are trying to determine which portions of data based only on the features differentiate each other into groups or clusters. The two clustering algorithms in this analysis, Expectation Maximization (EM) and KMeans (KM), were both implemented using *sklearn* in Python [1]. EM was implemented using a Gaussian Mixture Model (GMM). For the clustering algorithms, both data sets were preprocessed, most of the features in both datasets were continuous values; however, the categorical features were intrinsically related so a label encoder was used since their relative meaning was physically appropriate. The preprocessing for both datasets involved normalizing each feature using *sklearn.preprocessing* class *MinMaxScaler* to transform the input data into values between 0 and 1.0.

2.1 Expectation Maximization

The GMM assumes that the data are generated from a combination of several Gaussian "bell-shaped" distributions. As the number of clusters isn't known apriori, this parameter must be tuned and evaluated to determine which and how many splits "best" groups the data. There are several ways to evaluate the "goodness" of a GMM clustering and if the number of clusters set was appropriate, for this analysis, the main quantitative metric for evaluating the quality of the GMM clustering was calculating the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). AIC and BIC are both methods for model selection that penalize the inclusion of free parameters to prevent overfitting, but they do so differently with distinct behaviors. AIC focuses on finding the best approximating model but may over fit, while BIC aims to converge to the true model as sample size increases, but it might under fit due to its penalty on free parameters, making the choice between them dependent on specific goals and data characteristics. Quality of GMM was defined using the minimum BIC metric, but both are shown in this analysis. Both datasets had samples for each feature that were normally distributed; however, this does not imply that the prediction classes are normally distributed. For both data sets, a GMM model was fit for several cluster sizes ranging from 2 to 200, the AIC and BIC for both is shown in Figure 2-1.

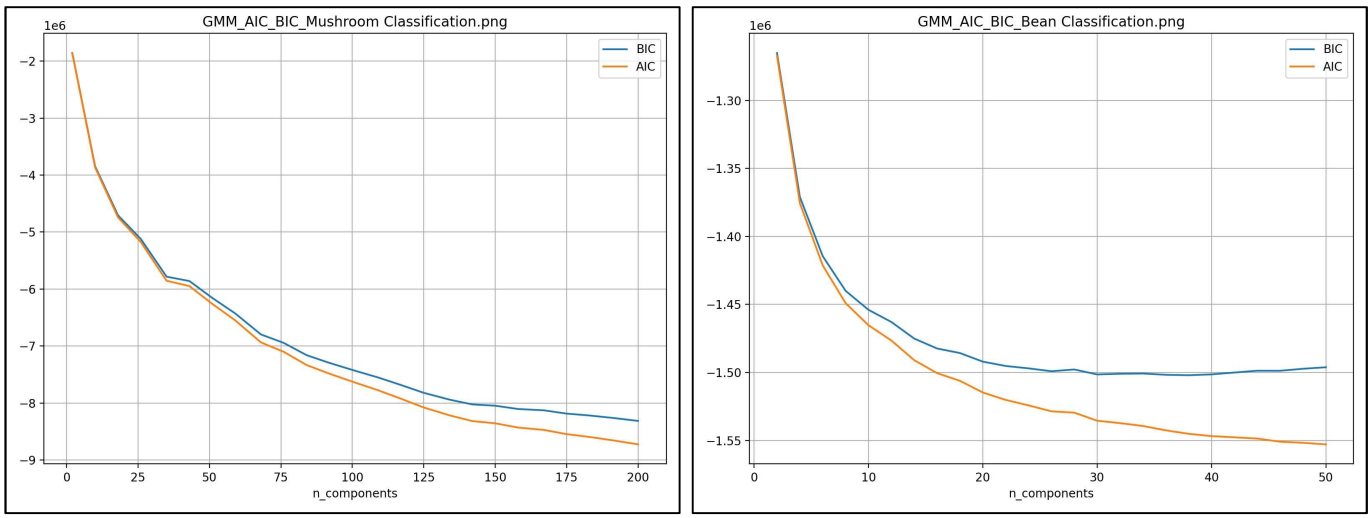

Figure 2-1 AIC and BIC vs. number of components in GMM clustering algorithm

Both data sets when fit with the GMM clustering algorithm showed optimal clusters above the number of classes in each data set, with optimal values over 200 for the mushroom classification and optimal values between 30 and 50 for the bean classification. Despite optimal AIC and BIC values for the GMM algorithm for both data sets, the GMM clusters of the training data when colored by the true classes were not well defined as shown in Figure 2-2.

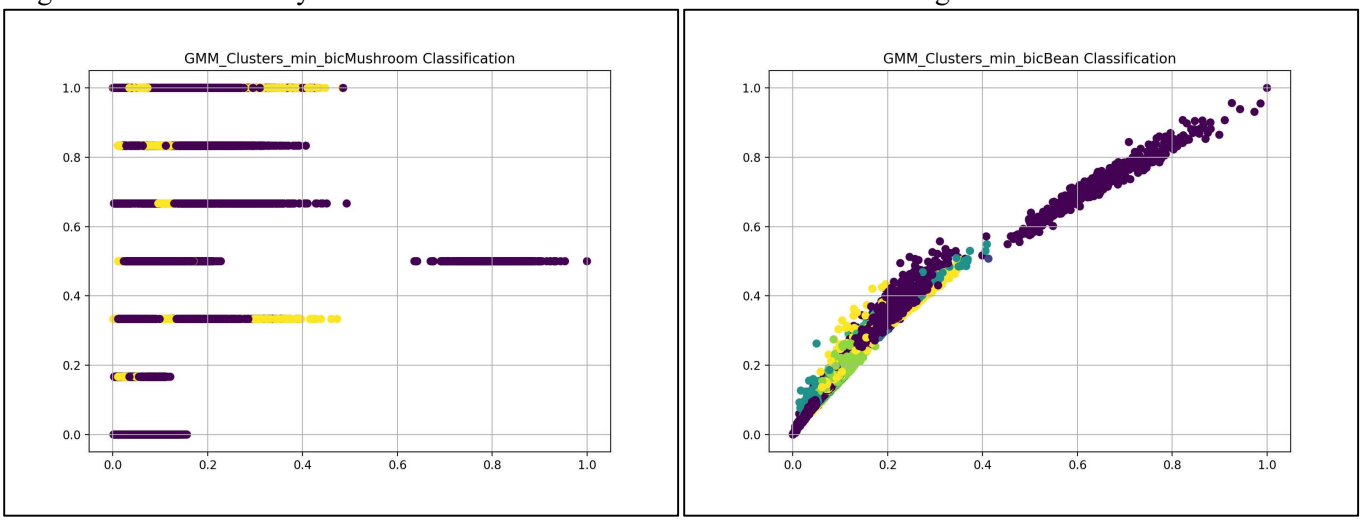

Figure 2-2 GMM Clusters with minimum BIC cluster colored by true data classification

This indicates that the GMM algorithm for clustering is not well suited for these classification problems and that the data sets are both not combined by linearly combining normally distributed features. Likewise, a neural network trained on the clusters defined by a GMM algorithm for these data sets would likely not benefit from this clustering algorithm and could potentially perform worse due to the introduction of misclassification.

2.2 K-Means

The second clustering algorithm chosen to analyze for both data sets was KMeans. KM is able to be used with combined continuous value and categorical features given that the data has been preprocessed as previously described. The distance metric used for both data sets was the Euclidean distance to the centroid centers. The quality metrics for the KM clusters was evaluating the silhouette score for a given cluster set. This score combines the intercluster and intracluster distances for points in clusters to given an over quantitative estimate of how well separated clusters are defined. The score ranges from 1 (perfect clustering) to -1 (very poor clustering). Clusters that separated from each other with no overlapping points would score well. For both data sets, the silhouette score for a range of cluster set sizes is shown in Figure 2-3.

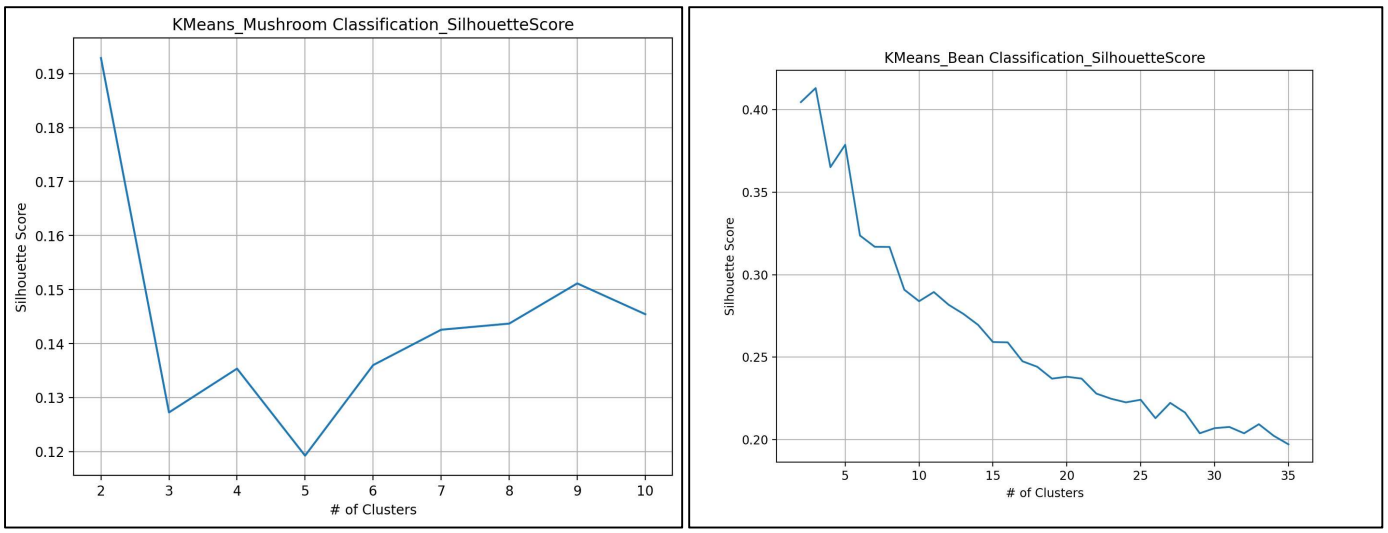

Figure 2-3 Silhouette Score for different number of clusters for KM algorithm applied to both data sets

As shown in Figure 2-3, the silhouette score for the bean classification drops precipitously even beyond the number of clusters equal to the true number of bean classes in the data. Lower clusters for the KM algorithm for this data set is better and the silhouette score is maximum with 3 clusters. The mushroom classification silhouette scores for the KM algorithm showed a non-monotonically decreasing silhouette score curve with a second local maximum at 9 clusters. However, the optimal number of clusters for KM for this data set based on silhouette score was 2 corresponding to the true number of classes in the data set. As unsupervised learning does not use known classes during training, the optimal number of clusters determined by the silhouette scores for both data sets were used in later analyses. The cluster prediction for these optimal values are shown in Figure 2-4 colored by predicted labels.

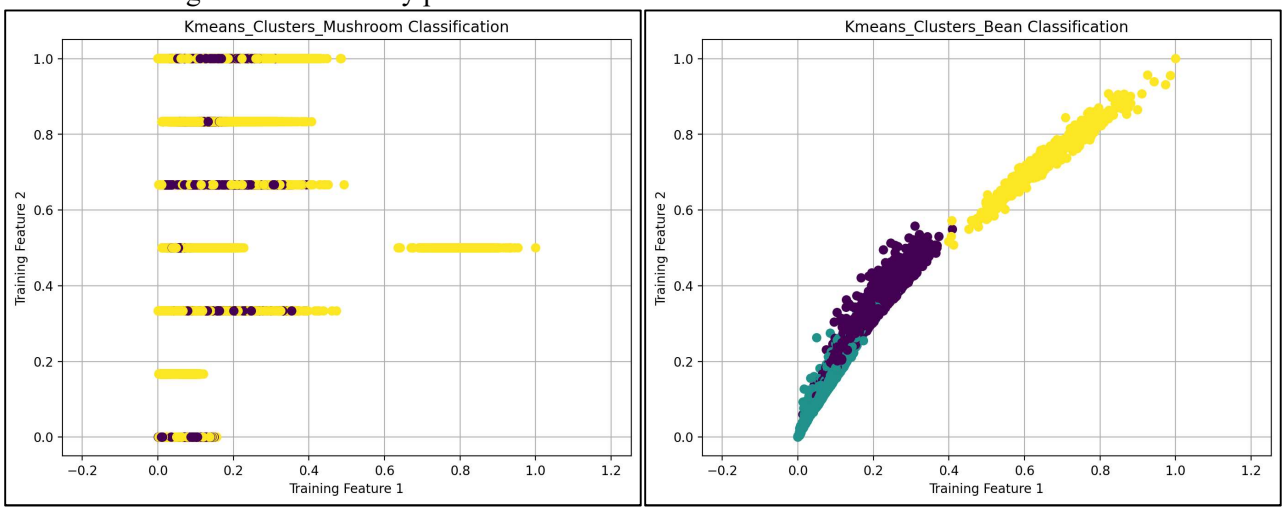

Figure 2-4 KM clusters using optimal number of clusters based on silhouette score analysis

Similarly to the GMM derived clusters for the data sets, the KM clusters for optimal silhouette score don't partition the mushroom data set into well-defined clusters while for the bean classification a modest distinction in classes is seen. This is reflected in the respective silhouette scores in Figure 2-3 for the optimal cluster values. Likewise, a neural network trained on the clusters defined by a KM algorithm for these data sets would likely not benefit from this clustering algorithm for the mushroom classification but could potentially benefit the bean classification.

3 DIMENSIONALITY REDUCTION

Four DR methods were implemented on the two data sets to explore their capabilities in transforming the original data sets into a reduced data set in terms of numbers of features. For machine learning algorithms, high dimensional or many feature data sets can prohibitively increase the computational time for training. These DR methods introduce mathematical techniques to elucidate more meaningful differences in samples for the purpose of training a model on. While both of these data sets don't contain a relatively large number of features, these DR techniques would provide a negligible improvement in training time for a neural network; however, they could potentially provide valuable transformed data which would improve the performance and convergence of the network.

3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in unsupervised learning to analyze and visualize data. While PCA is primarily used for dimensionality reduction, it can still be applied to classification data for feature reduction, data exploration, and visualization. PCA was analyzed for both data sets to determine an appropriate number of features to reduce to from the original data sets. PCA transforms the original data into new dimensions that are orthogonal to each and maximize variations along these dimensions. PCA was implemented using *sklearn.decomposition* class *PCA* [1]. The metrics used for determining the optimal number of components for PCA for each data set was the explained variance from the PCA transformed components as well as the mean-squared-error of the inverse transformed PCA data and the original data. These metrics for both data sets are shown in Figure 3-1. The optimal number of PCs for transformation capture a significant amount of the explained variance, for this analysis, the optimal number of PCs was set when the explained variance exceeded 0.9.

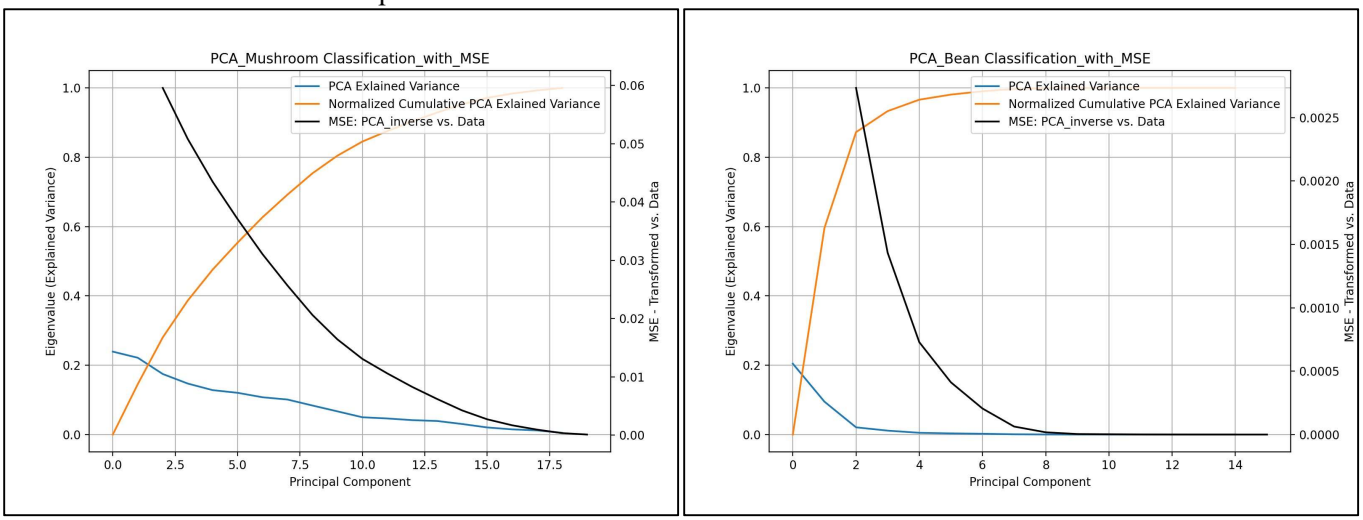

Figure 3-1 PCA explained variance and MSE analysis

For the mushroom classification data set the optimal number of PCs was 12 while for the bean classification data set the optimal number of PCs was 3. Figure 3-2 visualizes the comparison between the first and second principal component to show the performance of the method on transforming the data into a more descriptive dimension. The transformed data for the mushroom data set shows a poor dimension reduction performance for PCA as the first and second principal components do not improve classification as the plot color is the true data class. The transformed data for the bean classification shows a modest performance of the method to improve variation in data description by using the first two principal components. Again, as this method is being used for DR applied to unsupervised learning, the true data classes cannot be used to inform selection of optimal PCs. Thus, the optimal number of PCs for later are determined from the earlier analysis of explained variance.

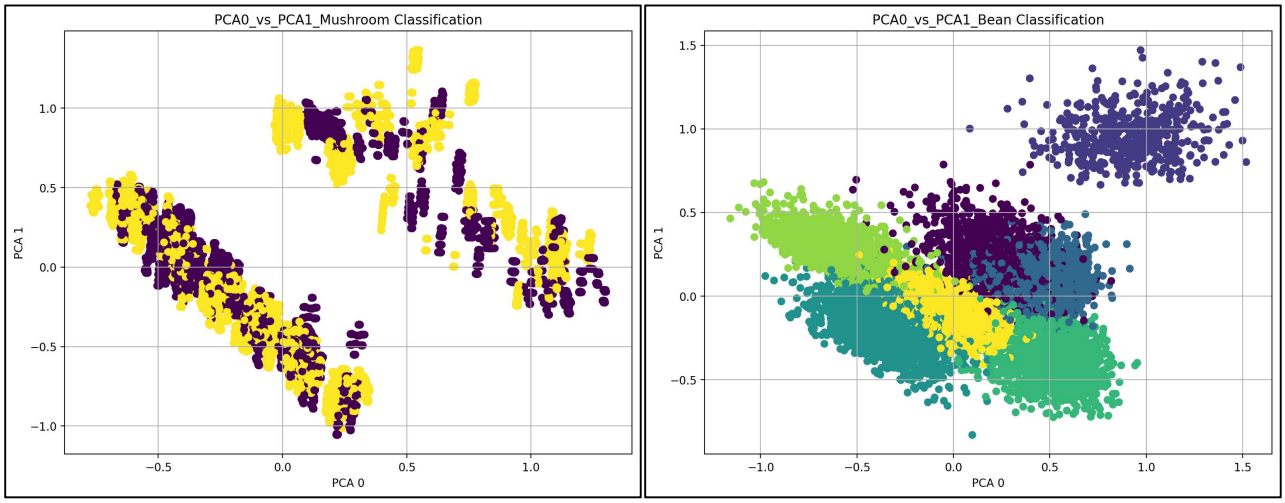

Figure 3-2 PCA Comparison of first and second principal components colored by true data class

3.2 *Independent Component*

Independent Component Analysis (ICA) is an unsupervised learning technique used to separate a multivariate signal into additive, statistically independent components. While ICA is more commonly applied in the field of signal processing and source separation, it can also have applications in classification data when determining underlying independent factors or features within the data. ICA was implemented using the *sklearn.decomposition* class *FastICA* [1]. ICA seeks to find a linear transformation of the original data that maximizes the statistical independence of the resulting components. Thus components have no relative rank as PCA does. The metric used to determine optimal ICA components for further analysis was using the average Kurtosis of the transformed data versus the number of components and an "elbow" analysis to determine where this data became significantly different from a Gaussian distribution where the average kurtosis for the transformed data exceeded 3.0. Highly kurtotic data has more frequent extreme deviations indicating highly statistically independent features in the transformed space. Figure 3-3 shows average kurtosis of the ICA transformed data.

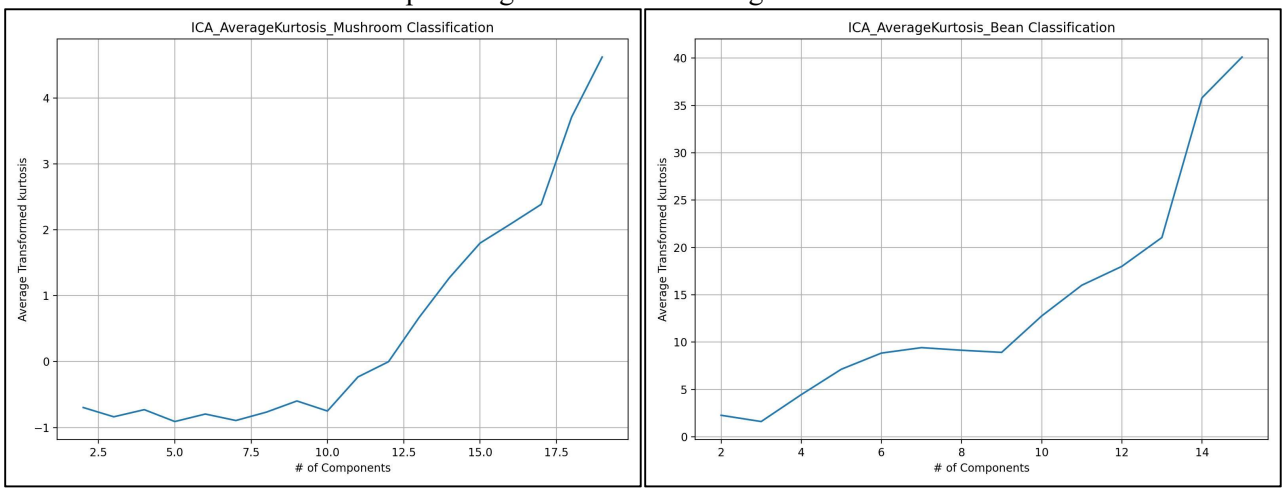

Figure 3-3 Average Kurtosis of ICA transformed data

For the mushroom data, the optimal number of components for ICA was determined to be 17 while for the bean classification data the optimal number of components for ICA was 4.

3.3 *Randomized Projections*

Randomized projection is a technique to reduce the dimensionality of the data while preserving its essential structure.
It involves creating a random matrix with values drawn from a Gaussian (normal) distribution. This random matrix is often referred to as the "projection matrix". The dimensionality reduction occurs by multiplying the original high-dimensional data by this projection matrix, which maps the data to a lower-dimensional space. The result of this random projection is a lower-dimensional representation of the data. The dimensionality of this reduced representation is determined by the size of the random projection matrix and is typically much smaller than the original dimensionality. For this analysis, Gaussian random projection GRP was implemented using *sklearn.random_projection* class *GaussianRandomProjection.* To determine an optimal number of projection components for GRP, the reconstruction error after inverse transformation of the projected data is determined by comparing to the original data showing that the data is maintained in the projected space. This error is shown in Figure 3-4.

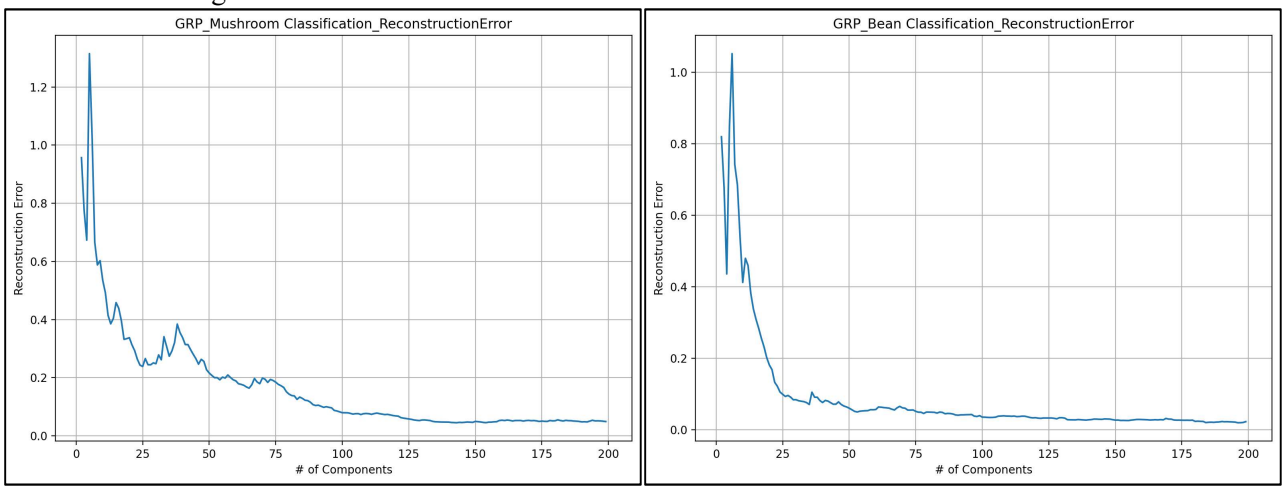

Figure 3-4 GRP Reconstruction Error

This shows that for both data sets that low reconstruction error is only achievable by increasing the number of components in the projected space far beyond the number of features in the original data set and even at high dimensions a small but non-negligible reconstruction error exists. This doesn't actually improve performance as significant error exists for projection spaces below the number of existing features in the data set. The optimal number of components was determined to be 13 for mushroom data set and 10 for bean classification.

3.4 T-Distributed Stochastic Neighbor Embedding

The last DR technique used is TSNE which aims to represent high-dimensional data in a lower-dimensional space (usually 2D or 3D) while preserving the pairwise similarities or dissimilarities between data points.

It does so by iteratively minimizing the difference between the pairwise distances in the original high-dimensional space and the lower-dimensional space. For this analysis, a projection into 2-dimension was used as both data sets already had a low number of features. This makes visualization of the classes in the projected data by plotting the transformed data in the new dimensional space as shown in Figure 3-5.

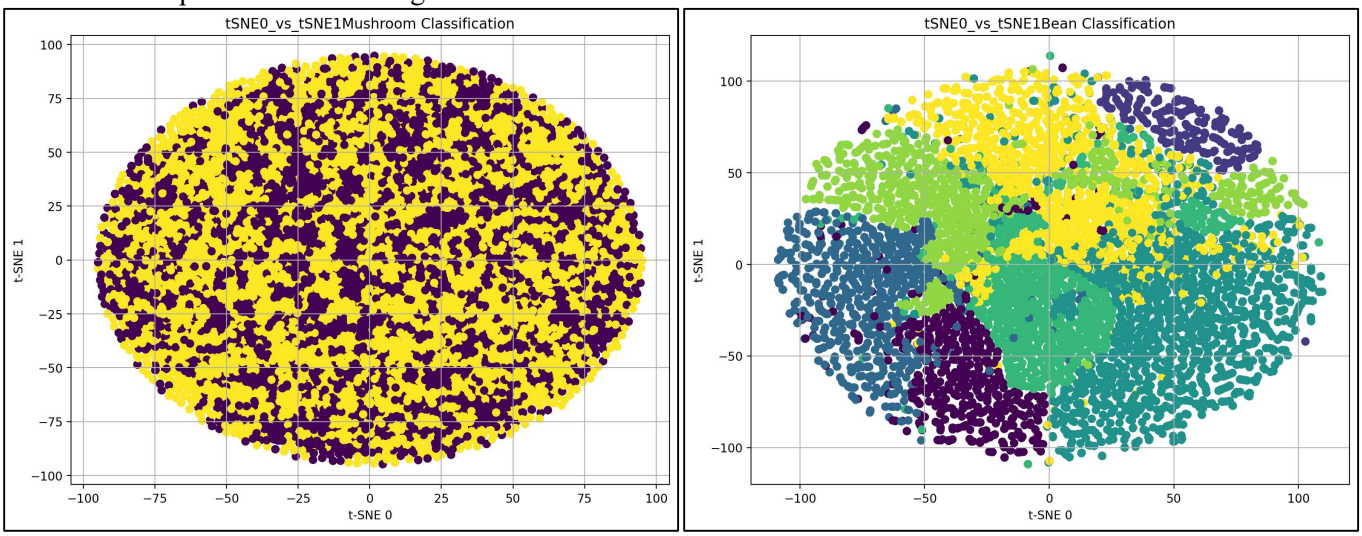

Figure 3-5 t-SNE transformed data colored by true class

Similar to the previous methods described, the mushroom classification data set did not perform well in t-SNE transformation while the bean classification data showed more consistent grouping in the transformed space. This indicates the a neural network for the mushroom data set would likely perform better on the original data than using any of the DR or clustering techniques here while the bean classification data set could potentially benefit from transformed data.

4 CLUSTERING APPLIED TO DIMENSIONALITY REDUCTION

For this analysis, the clustering algorithms discussed in section 2 were applied to each of the DR methods in section 3 for both data sets. The cluster and DR algorithm parameters were set as the optimally determined values from their independent analyses. These clusters are shown in Figure 4-1. The most well defined clusters from the combined analysis for the mushroom data set was using the GMM clustering algorithm on the PCA transformed and reduced data set using the optimally determined number of components from the PCA analysis. For the bean classification data set the most well defined clusters used the KM clustering algorithm along with PCA.

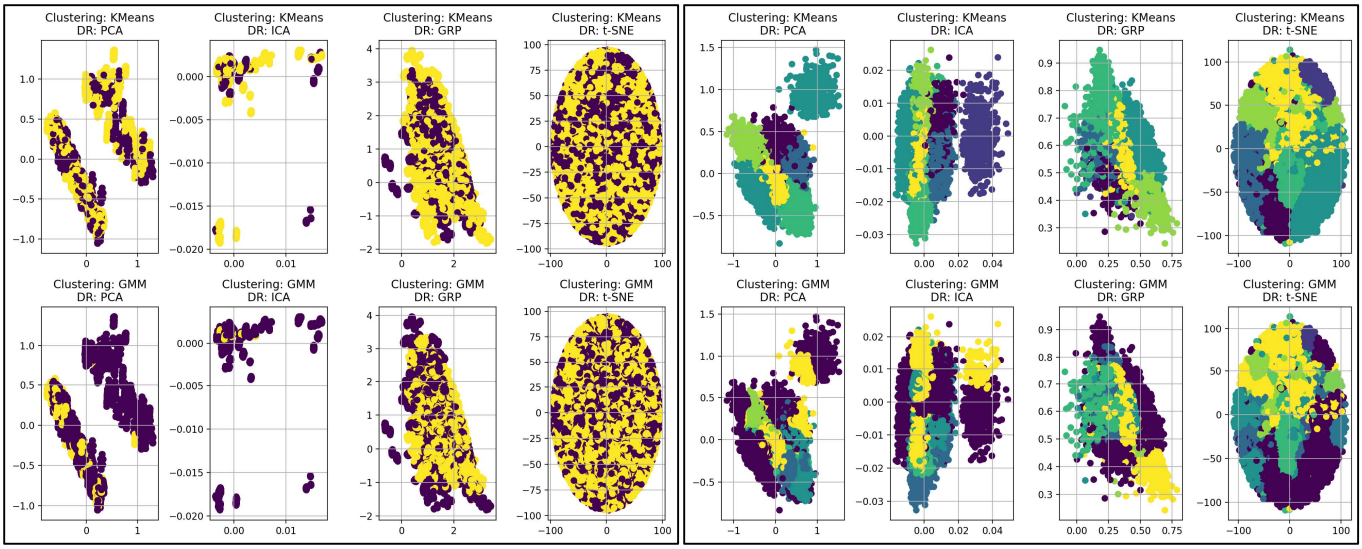

Figure 4-1 Clustering algorithms applied to DR transformed optimal methods from section 3. Mushroom data on left, bean data on right

5 Neural Network using Dimensionality Reduction

For this analysis, a multi-layer perceptron (MLP) neural network supervised learning algorithm was trained on the original data set using a training data set that contained 80% of the samples and used a test data set containing 20% of the samples for validation. Additionally, an MLP classifier was trained using each of the transformed data sets from the optimally defined DR methods described in section 3 to determine how these techniques would affect the performance of the neural network on each data set. A grid search on the optimal parameters for the MLP model was performed for the original model, these parameters were then applied to the MLP models trained on the transformed data, no additional grid searches were performed. Figure 5-1 shows a comparison of the performance of the neural networks trained on their respective data.

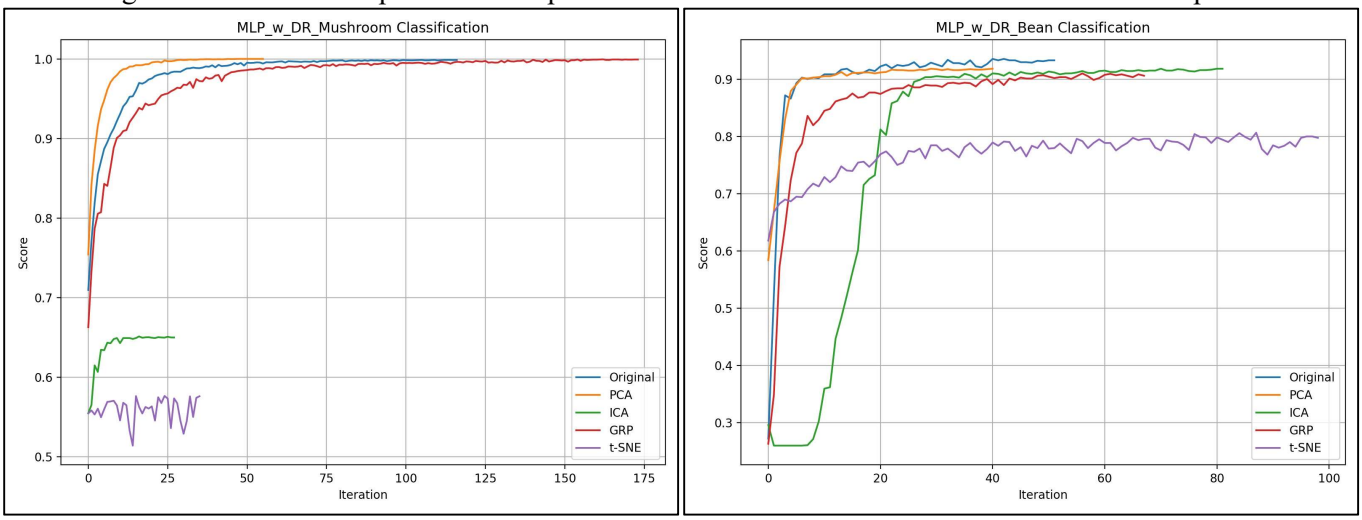

Figure 5-1 Validation curve comparison of MLP classifiers for DR

As indicated in the previous analysis for section 3, several of the DR techniques provided no benefit to the neural network for classification training. However, the PCA algorithm showed an improved convergence for both data sets compared to the original MLP classifier trained on the original, untransformed data. While the amount of data and features in these data sets were relatively small, this improved convergence could represent a significant improvement for large and complex data sets. The t-SNE algorithm for data reduction proved to be the least valuable method for DR for training the MLP, in fact, it significantly degraded the performance of the model for both data sets.

6 Neural Network using Clustering

For this analysis, the MLP was trained using both of the clustering methods as additional features for the data sets to determine how these techniques would affect the performance of the neural network. Figure 6-1shows a comparison of the performance of the neural networks trained on their respective data.

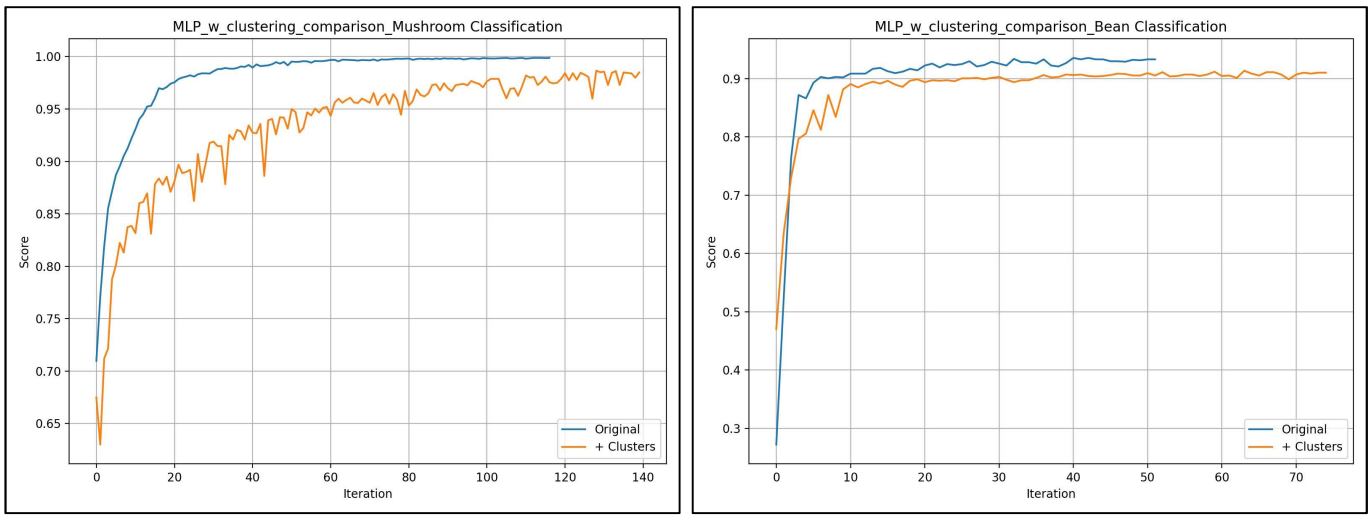

Figure 6-1 Validation curve comparison of MLP classifiers for clustering

As hypothesized, the clustering analysis actually degraded the convergence and performance of the MLP model relative to the model trained on the original data. This is due to the fact that both data sets likely were not well suited for clustering and the algorithms introduce more noise into the data causing the model to perform worse.

7 SUMMARY

This analysis explores various unsupervised learning methods, clustering and dimensionality reduction (DR), to gain insights into the relationships among data samples. These techniques aim to create more meaningful representations of data, with the ultimate goal of improving the performance of supervised learning algorithms. The analysis involves applying clustering algorithms (Expectation Maximization and K-Means) and DR methods (Principal Component Analysis, Independent Component Analysis, Randomized Projections, and t-Distributed Stochastic Neighbor Embedding) to two classification datasets—one for mushroom classification and another for bean classification. The findings indicate that some DR techniques, such as PCA, can improve convergence for an MLP neural network classifier, while others, like t-SNE, may degrade the performance of the neural network. Additionally, clustering analysis did not yield favorable results, likely due to the datasets not being well-suited for clustering.

8 REFERENCES

[1] F. Pedregosa and e. al, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.