Shannon Ke
CS 4641
Unsupervised Learning Report

## Classification Problems

### The Nursery Dataset (from Project 1)

The first data set, the nursery database, was constructed originally to assist in ranking and ultimately giving explanations for accepting or denying applicants from nursing school and found through the UCI machine learning repositories. The data was derived from a hierarchical decision model that was used in the 1980s when applications to certain nursing schools was particularly high. The dataset has eight attribute values and five possible classifications with 12960 instances of data. The combinations of types of attribute values makes this an interesting learning problem since personally, there was no readily obvious connection between sets of attribute values and target values. I was curious to see how each learning model would attempt to classify each instance into its corresponding class.

### The Car Dataset (from Project 1)

The second data set, the car database, was also found through the UCI machine learning repositories. The attribute values, of which there are six and include buying price, maintenance price, number of doors, etc. , determine the acceptability of the car in question. There are four values for acceptability: unacceptable, acceptable, good, and very good. This dataset is interesting as it takes a very practical problem that would be useful in many cases for normal people looking to buy a car or looking for car recommendations and turns it into a machine learning problem that could be implemented as an aid for many.

## General Methodology

I pulled my datasets from the CSV files and encoded them using scikit's label encoder in order to transform string feature values into readable integers. Using scikit's train_test_split method, for graph analysis involving changing dataset sizes, I randomly split the data into training and test data. For graph analysis involving changing hyperparameters, I kept the data split (training data/test data) for the nursery dataset at (70/30) given that the dataset has a large number of samples, and the car dataset at (90/10) given that the dataset has a smaller number of samples.

I ran each dataset through GMM with expectation maximization, Kmeans clustering, and PCA before running them through GMM and Kmeans while applying PCA. When running the datasets through just PCA, I lowered dimensionality to two and three attributes for both datasets to make visualizations for both of them. However, when applying PCA to GMM, Kmeans, and later on, my neural network, I simplified my dimensions by removing only one or two attributes. I then ran only my cars dataset through my neural network again, then applied PCA, GMM, and Kmeans onto my data before rerunning the neural network, taking the average accuracy values across 15 trials.

## Kmeans Clustering

Kmeans clustering is a clustering method by which a number of *k* clusters is specified. The algorithm then attempts to take *k* number of centroids and fit the data to certain clusters by iteratively improving the centroids. In my implementation, I did not specify my centroids at the start of the algorithm, so they were chosen randomly for me. For both datasets, the optimal *k* value equaled the number of labels for each dataset, so I fixed my starting *k* value to four and five for the cars and nursery data respectively.

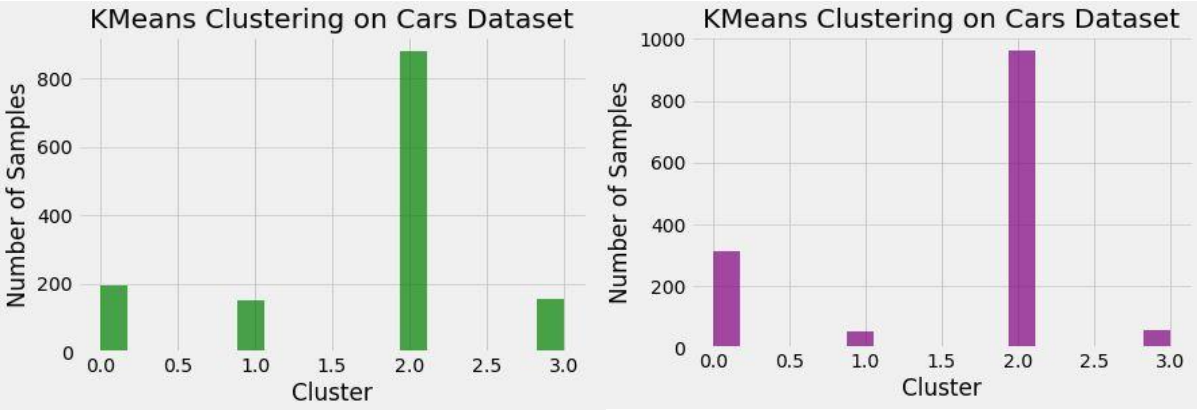


*Figure 1(a and b).* (a) The distribution of data points between different clusters for the cars dataset. The 0th cluster corresponds to the 'acc' label, the 1st to the 'good' label, the 2nd to the 'unacc' label, and the 3rd to the 'vgood' label. (b) The actual distribution of the cars dataset, with the cluster values the same as in graph (a).

We can see from the visualization of data distribution to different clusters (detailed in *Figure 1*'s description) that Kmeans performed well on the cars dataset even without dimensionality reduction by observing the general shape of number of samples in the graphs. Kmeans correctly clustered most of the data into cluster 2, which corresponded with the 'unacc' label. Clustering most likely did very well due to the highly skewed classification of the cars dataset, with most of the data points classifying to 'unacc.' The average accuracy over fifteen runs hovered around 40.06%, a good result for data with 4 classifications.

*Figure 2* displays graphical comparisons between how data was clustered for the nursery dataset without dimensionality reduction. There are similarities between the results in graph (a) and the actual values in graph (b). However, the nursery dataset did not perform quite as well the cars dataset did with Kmeans clustering. As stated before, with the nursery dataset having 5 unique labels, *k* was set to 5 and centroids were randomly produced at the start of the algorithm. The average accuracy over fifteen runs hovered around 24.78%, which is not much better than random guessing. Given the nature of Kmeans clustering and runs on the nursery dataset from previous projects, the poor performance of the algorithm can be attributed to the noisiness of the nursery dataset and irrelevance of the some of the attributes.

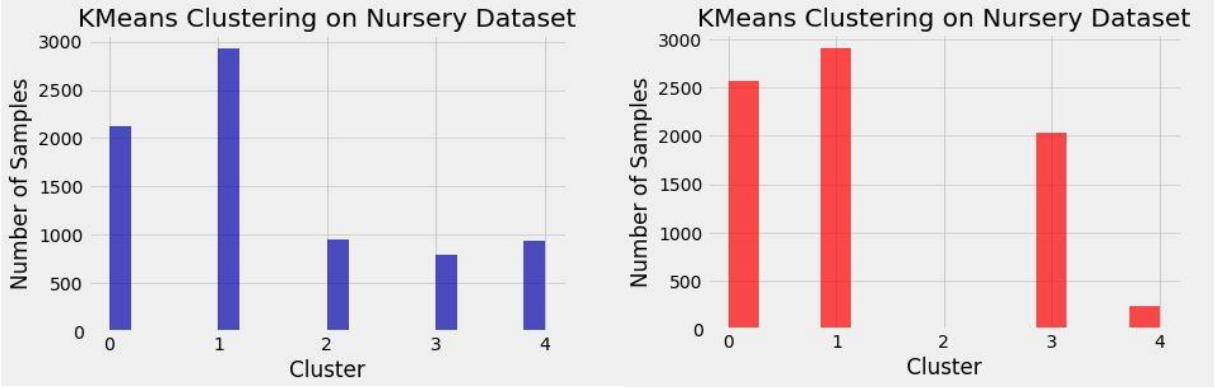


*Figure 2(a and b).*(a) The distribution of data points between different clusters for the nursery dataset. The 0th cluster corresponds to the 'not_recom' label, the 1st to the 'priority' label, the 2nd to the 'recommend' label, the 3rd to the 'spec_prior' label, and the 4th to the 'very_recom' label. (b) The actual distribution of the nursery dataset, with the cluster values the same as in graph (a). Notice that the nursery dataset is highly imbalanced, with only two samples mapping to the 2nd cluster, or the 'recommend' label. In hindsight, I most likely could have thrown out those two samples and instead trained the algorithm on four classifications without unnecessary noise.

## Gaussian Mixture Model with Expectation Maximization

The expectation maximization algorithm consists of two steps: the expectation step and the maximization step. The expectation step for the gaussian mixture model (GMM) involves calculating membership weights of each data point for all mixture components, effectively calculating the probability that each data point will belong to a certain gaussian. The maximization step involves using these weights and the data to calculate new values that update the gaussians themselves.

Since gaussian distribution is heavily influenced by its covariance matrix, I ran scikit's GMM with four covariance matrix types: full, tied, diagonal, and spherical, and plotted the eventual gaussian coverage for each.

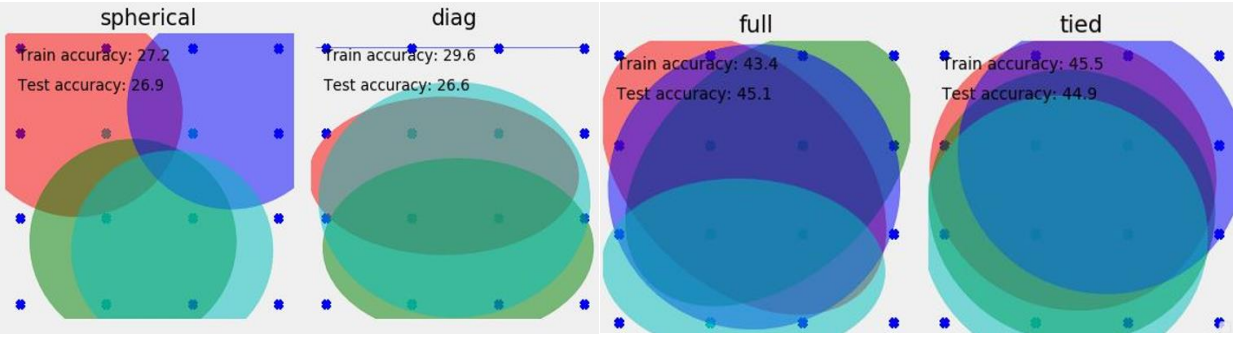


*Figure 3(a, b, c, and d).* The gaussian confidence ellipsoids for each covariance matrix type, including train and test accuracy on the car dataset. The highest accuracies correspond to many overlapping gaussians, indicating that there is a high amount of overlap between attributes.

Since full and tied covariance matrices are more forgiving in terms of component shape (with diagonal and spherical limiting shape by making sure that components are oriented along a coordinate axes), it makes sense that GMM performs better as depicted by graphs (c) and (d) in *Figure 3.* Performance is similar to and slightly better than the Kmeans clustering performance, again solidifying the cars dataset's aptitude for clustering algorithms.

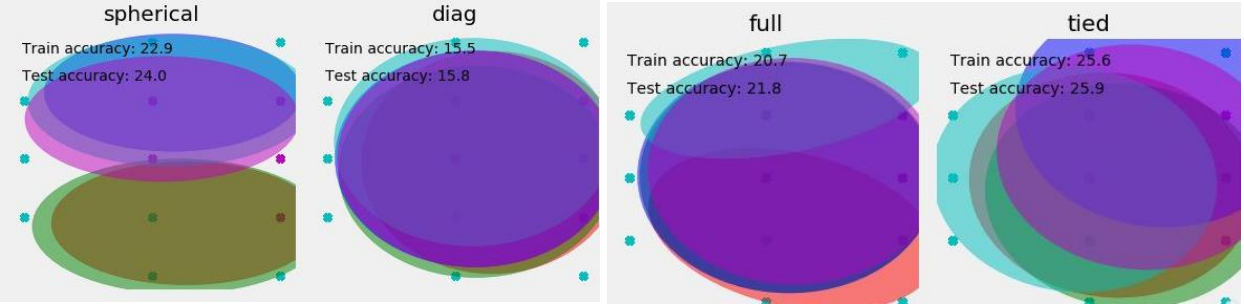


*Figure 4(a, b, c, and d).* The gaussian confidence ellipsoids for each covariance matrix type, including train and test accuracy on the nursery dataset. The highest accuracies correspond to less overlapping gaussians, indicating that there may not be a high amount of overlap between attributes.

Upon running GMM with EM on the nursery dataset, I noticed that the spherical and tied covariance matrices yielded higher accuracies. This implies that overlap between attributes may not be as predominant as it was in the cars dataset. Again, given the high noise profile of the nursery dataset, clustering algorithms are not particularly effective and yield low accuracies.

An important thing to note is that both the cars and nursery datasets are multidimensional, so the two-dimensional graphical representations in *Figure 3* and *Figure 4* do not accurately represent how the gaussian ellipsoids appear due to severe dimensionality limitations.

## Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction algorithm that converts features that appear to be correlated into principal component vectors that must be orthogonal to each other, with the first vector being the best explanation of the data, the second being the second best, and so on.

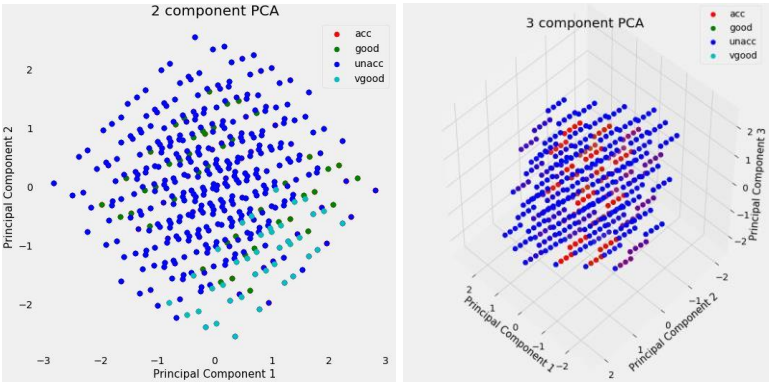


*Figure 5(a and b).* Scatter plots depicting new data clustering by lowering the cars dataset's dimensionality from 6 to 2 and 3. A high amount of overlap is clearly depicted by the graphs.

With the car dataset's already low dimensionality and high performance with basic clustering algorithms, PCA may not be very effective. Principal components are ordered by the % of variability they explain, and the explained variability (aka the eigenvalues) for each component with n_components set to 3 was very similar, hovering around 0.17. This implies that no few combination of components could overwhelmingly explain the variation between each data point, with each attribute describing an equal portion of its label. Overall, the three principal components describe 0.51 or about half the variability of the dataset.

The behavior of PCA on the nursery dataset is similar to that of the cars dataset. The explained variability (aka eigenvalues) of each principal component with n_components set to 3 hovered around 0.14, with each subsequent variability slightly lower than the previous. This again implies that no combination of components could overwhelmingly explain the variation between each data point. Overall, the three principal components describe 0.43 or almost half of the variability of the dataset. Another thing to consider is the noisiness of the nursery dataset, with many of the attributes not clearly correlating with each sample's label upon glancing over the attribute names. This makes the nursery dataset a particularly difficult one to learn, explaining classifiers' poor performances on attempting to explain the dataset.

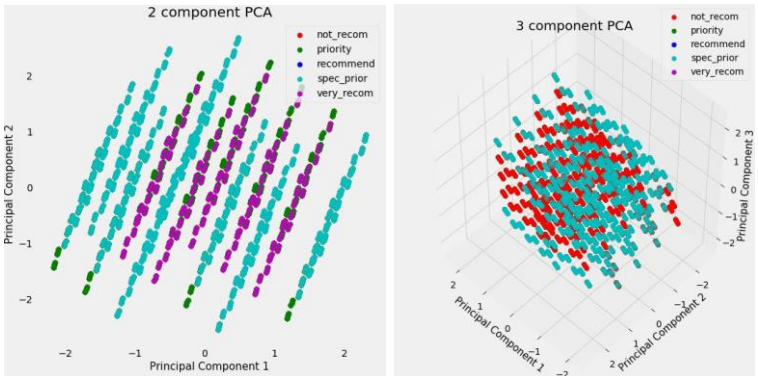


*Figure 6(a and b).* Scatter plots depicting new data clustering by lowering the nursery dataset's dimensionality from 8 to 2 and 3. Again, a high amount of overlap is clearly depicted by the graphs.

Based on these findings of PCA with the cars and nursery datasets, we can assume that applying PCA to the Kmeans and GMM clustering algorithms will not do much to improve performance since it does not effectively reduce dimensionality in a way that is beneficial to the datasets, most likely due to the already small number of attributes, meaning that each attribute is highly influential in classifying, or in the case of the nursery dataset, that each attribute explains each data point equally poorly.

## Running PCA on Kmeans

To see how dimensionality reduction effects the performance of clustering, I ran PCA on each dataset and used the newly reduced attribute spaces to run the algorithms.

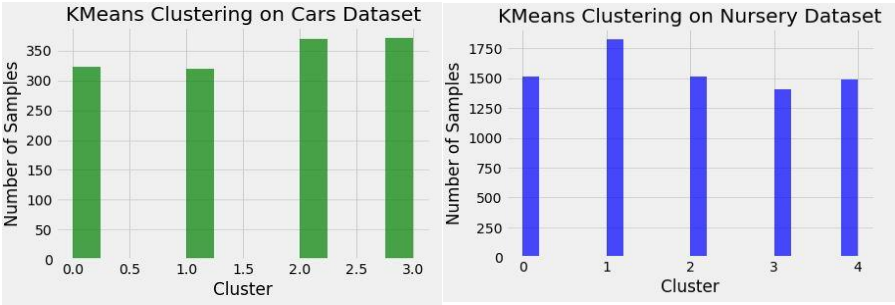


*Figure 7(a and b).* Histograms depicting the new clustering resulting from Kmeans with PCA for both datasets. Clustering is now much more uniform across different labels. Refer to *Figure 1b* and *Figure 2b* for the actual distribution of data points.

Performance on each dataset faired as expected given the equal weight that was applied to each principal classifier when running just PCA. The data has become distributed much more evenly across every cluster, implying that PCA smoothed over and combined attributes that ended up leading to classifying an equal number of samples to each cluster. Accuracies fell to 24.31% from 40.06% and 23.04% from 24.78% for the cars and nursery datasets respectively. While the severe drop in the cars dataset's accuracy is explained by the importance of all of the attributes (of which there were few to start with, so PCA reducing dimensionality actually hurt more), the accuracy in the nursery dataset stayed almost the same. This is again due to the noisiness of the data, and the low correlation between the

attribute features and the labels. As I continue to run tests on the nursery dataset, I am beginning to see more that the attributes of it were not designed to explain the sample classifications, with most transformations to the data doing little to improve classification accuracy past random guessing.

## Running PCA on GMM

To test how dimensionality reduction affects GMM clustering with EM, I ran PCA on both datasets, keeping my cars and nursery datasets' original variances at 95%.

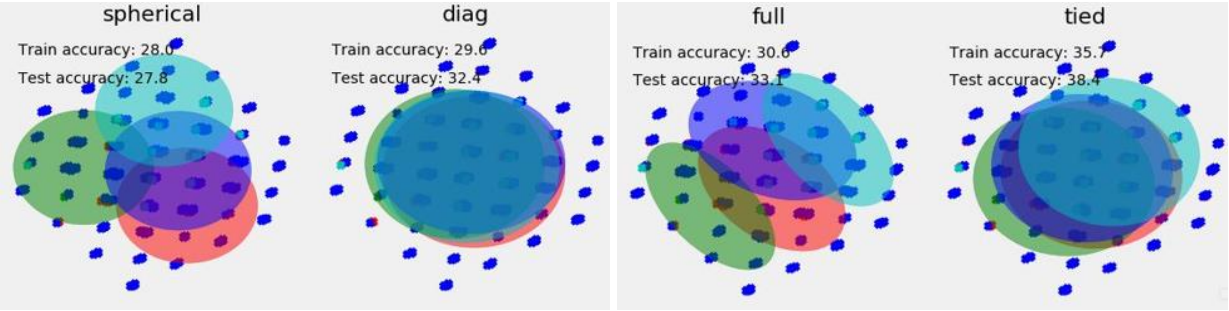


*Figure 8(a, b, c, and d).* The gaussian confidence ellipsoids for each covariance matrix type, including train and test accuracy on the cars dataset. Overall highest accuracy decreased with the addition of PCA fitting of the data. However, accuracies of spherical increased slightly and accuracies of diagonal covariance matrices increased.

Applying PCA to the cars dataset led to decreased accuracies for full and tied covariance matrices but increased accuracies in spherical and diagonal covariance matrices. This could be explained by how PCA creates a set of linearly uncorrelated variables (the principal components) from possibly correlated variables. Since spherical and diagonal covariance matrices makes the gaussian mixture contour axes oriented along the coordinate axes, making it a more limiting model, it should perform better with the addition of PCA since PCA will make each axes more associated with variance in the dataset. Again, the highest increase in accuracy was in *Figure 8b*, which features overlapping gaussian ellipsoids, portraying the overlap in importance of all of the attributes or the inability to express ellipsoid shape due to the two-dimensionality of the visualization.

We should expect similar behavior for the nursery dataset, with increase in spherical and diagonal accuracies and a decrease in full and tied accuracies.

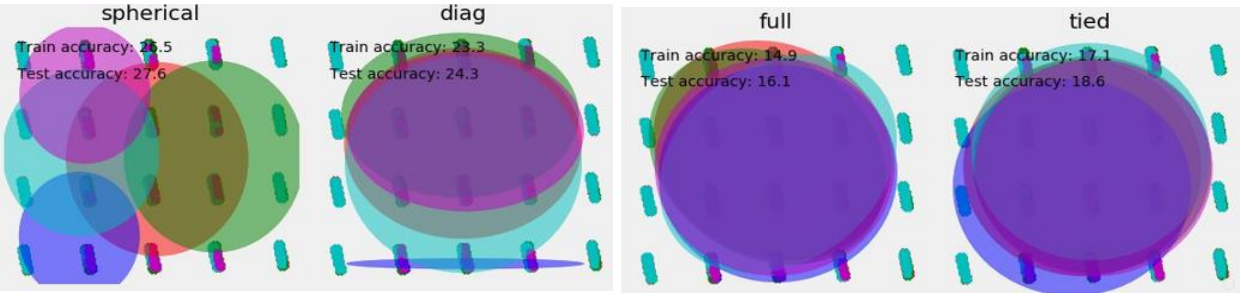


*Figure 9(a, b, c, and d).* The gaussian confidence ellipsoids for each covariance matrix type, including train and test accuracy on the nursery dataset. With the addition of PCA, accuracies of spherical and diagonal covariance matrices increased.

For both the cars and nursery datasets, the addition of PCA expanded and rounded out the ellipsoids, especially in the case of spherical covariance matrices. With spherical covariance, each component has its own single variance, making it close to diagonal covariance matrices but spherical in higher dimensions. It experiences less overlap due to its more restrictive nature over the gaussian contours. With PCA, each ellipsoid broadened in size, covering more data points, meaning that each ellipsoid is now a gaussian that can explain more of the data.

## Running PCA with a Neural Network

In order to test how PCA affects performance on a neural network, a projection was learned by fitting PCA to a my cars training dataset and projecting it onto both my training and testing datasets. To do this, I modified my neural network from project 1 by fitting my training data to a PCA that I first set to 70% of the original data's variance before incrementally adding 2% of the original data's variance to that value. My results can be seen in *Figure 10.* As observed from previous experiments, with the already low dimensionality of the cars dataset, eliminating variance with dimensionality reduction hurts more when too much is eliminated. However, I chose to operate the neural network on the cars dataset over the nursery dataset given that it has less noise and the more meaningful attributes.

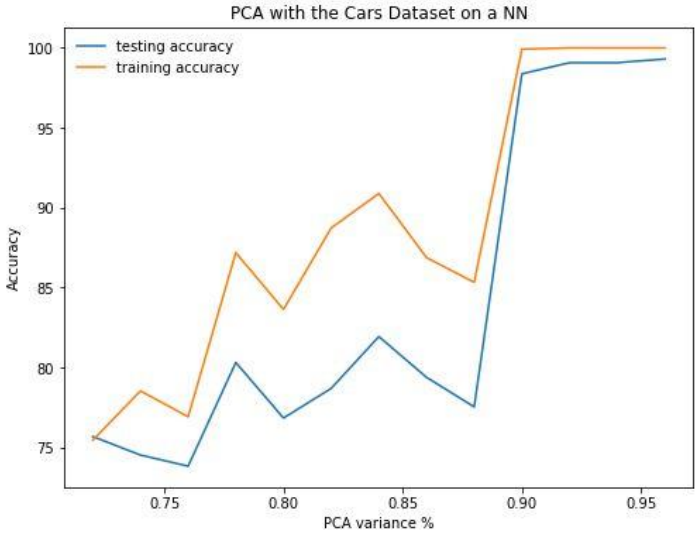


*Figure 10.* A multilayer perceptron neural network trained with the PCA fitted cars dataset. The percentage variance retained was input as a parameter to PCA, and performance with datasets with varying degrees of original variance was graphed.

While keeping a high percentage of the original data's variance was more desirable, I ended up choosing 95% original data variance for performance comparisons between neural networks and neural networks with PCA since having some amount of dimensionality reduction smoothed over a bit of noise, which improved performance on the neural network with PCA slightly.

As observed in *Figure 11*, the neural network's training accuracy with PCA implemented was close to perfect, whereas the network's training accuracy without it varied more between 95 and 100%. Testing accuracies for (b) never fell below 98%, while testing accuracies in (a) averaged more around 96% accuracy. The neural network performed better overall with the inclusion of some dimensionality

reduction, which was not enough to decrease the number of attributes but still created six principal components that correctly accounted for most of the variance in the dataset.

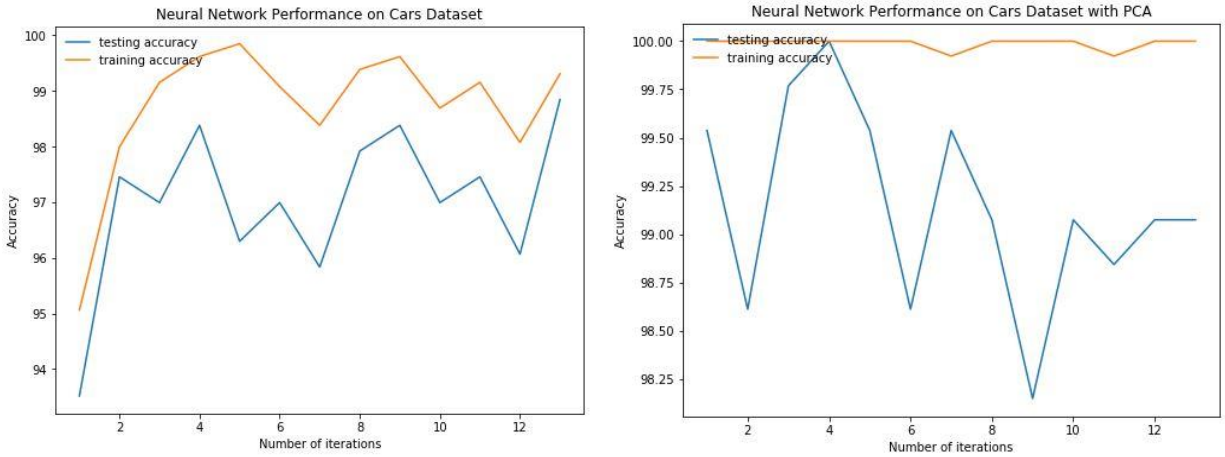


*Figure 11(a and b).* (a) The multilayer perceptron neural network was first run with no modifications over multiple iterations. The average training accuracy was 97.00%. (b) The neural network was then run with PCA at 95% of the original variance and yielded an average training accuracy of 99.14%.

## Running Kmeans with a Neural Network

To test how Kmeans clustering affects performance of a neural network, I ran the Kmeans clustering algorithm on my training dataset and predicted cluster placement for each data point in the cars dataset based on my training dataset. Since predicting Kmeans clustering on a dataset returns a list of clusters that a data point will be in, I could use these values as another attribute in the cars dataset. Thus, I created a new file named cars-data-kmeans.csv which includes a column with Kmeans predictions and retrained the neural network on this new data file.

Given the performance of Kmeans on the cars data from the first section of this paper, we can assume that adding its predictions as a feature will not cause accuracy to suffer.

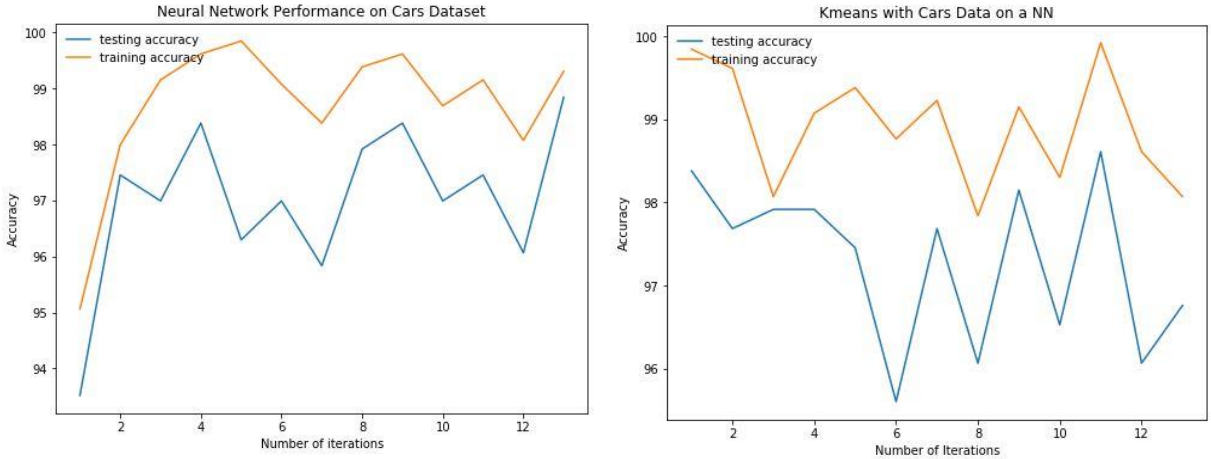


*Figure 12(a and b).* (a) A graph identical to *Figure 9a* depicting overall neural network performance for comparison. (b) A graph depicting the neural network trained on the cars dataset after an additional attribute from

the Kmeans predictions is added to each data sample. Overall average training accuracy sits at 97.68%, just a slight increase from the original neural network's 97%.

Given that the increase in training accuracies between *Figure12's* (a) and (b) is only 0.68%, we can see that the original dataset's attributes were able to classify the data well enough that adding an additional attribute from Kmeans did not increase the accuracy a notable amount. Outside of the outlier that is the first data point in *Figure 10a*, training and testing accuracies between the two neural network trails are similar as well. We can expect similar behavior using GMM with EM to run the neural network.

## Running GMM with a Neural Network

To test how GMM clustering with EM affects neural network performance on the cars dataset, I ran the GMM clustering algorithm on my training dataset and predicted probability values for each data point in the cars dataset based on my training dataset. Since predicting GMM clustering on a dataset returns a list of probabilities that a data point will be in a certain gaussian, I could use these probabilities as another attribute in the cars dataset. Thus, I created a new file named cars-data-gmm.csv which includes a column with GMM predictions and retrained the neural network on this new data file.

Since GMM performed similarly to Kmeans on the cars dataset, we can assume that behavior will be similar to running the previous section's experiment.

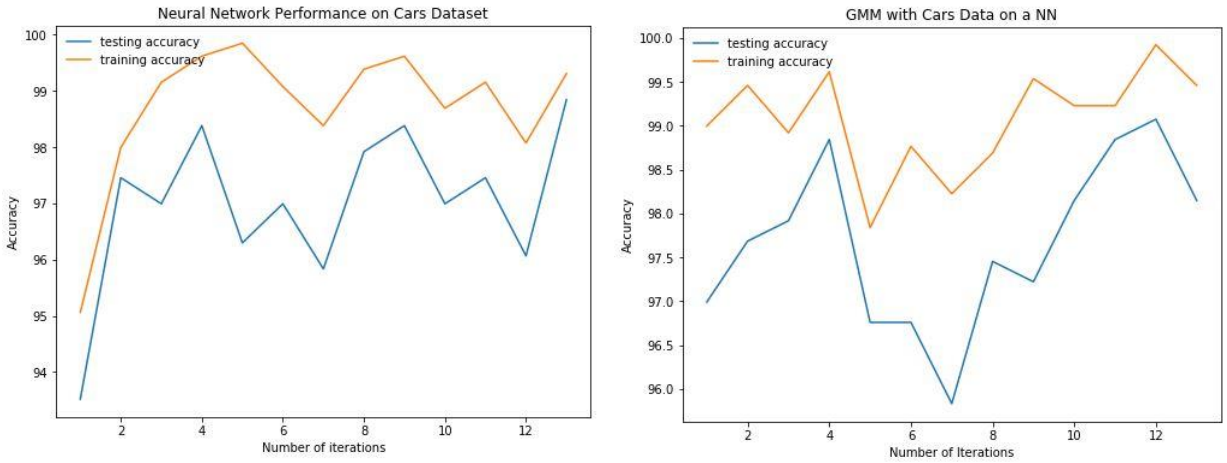


*Figure 13(a and b).* (a) A graph identical to *Figure 9a* depicting overall neural network performance for comparison. (b) A graph depicting the neural network trained on the cars dataset after an additional attribute from the GMM with EM predictions is added to each data sample. Overall average training accuracy sits at 97.67%, just a slight increase from the original neural network's 97%.

As expected, performance from one clustering algorithm to the next when applied to the neural network was similar. The original dataset's attributes were able to classify the data well enough that adding an additional attribute from GMM did not increase the accuracy a notable amount. Training and testing accuracies across both (a) and (b) are very similar.

## Conclusion

Based on results from these and previous experiments, several conclusions can be made about the natures of the chosen datasets. Clearly, the cars dataset had good attributes that equally affected classification. Thus, basic clustering with Kmeans performed fairly well in generalizing data, and GMM with EM without PCA performed twice as well as random guessing. The cars dataset also features a small number of attributes, making dimensionality reduction algorithms like PCA less effective. However, we still observed an increase in performance when applied to a neural network as long as dimensionality was conserved, meaning that the new principal components did well to describe the variance in the data.

Unfortunately, given the observations from the clustering algorithms, we were not able to observe much of a change in accuracy when applying Kmeans and GMM with EM to the dataset before running the cars dataset through the neural network due to the already optimal set of attributes. With the addition of the attributes created from predictions made by each clustering algorithm, the neural network could not find much use for the extra information since the accuracy was already so high. However, given the nature of the nursery dataset, the cars dataset was still the better dataset to use of the two to run analysis on.

This takes us to the nursery dataset, which over many experiments has proven to have attributes that very weakly classify each data point. Given that the dataset has labels relating to nursery school acceptance, the strongest attribute should be something more like application goodness or prior school GPA. These are not included as attributes, which are instead smaller criteria for school acceptance, such as social status and health status. These make sense when dealing with the acceptance of someone after their GPA and application have been considered, making them by themselves weak attributes to classify acceptance by. Thus, labels could not be strongly classified by the given attributes, and the nursery dataset performed similarly to the cars dataset for clustering but on the opposite end of the accuracy spectrum due to the uniformity of poor quality of the attributes.

In conclusion, after running multiple experiments, it is clear that clustering was not as noticeably effective on both the cars and the nursery datasets due to the previous reasons. In hindsight, I should have chosen better datasets, with a combination of both strong and weak attributes to portray the strengths and weaknesses of each clustering algorithm and PCA better, but nevertheless I am glad that I am able to come to these conclusions based off of my observations.