# Unsupervised Learning Report

## Datasets

*Adult Census Income Dataset*: The Census Income dataset has 48,841 instances of records containing census information about adults to predict whether their income will exceed $50,000 per year or not. There are 14 features which are included such as age, work class, education, occupation, race, sex, hours per week along with several other potentially relevant statistics. Education and education-num ended up representing the same data so I deleted the education column.

*Water Treatment Plant Dataset:* The water treatment plant dataset contains information about 527 water treatment plants with 38 features and is not labeled. I chose this dataset as it was stated to be an "ill-structured domain" and had just enough data points to produce clean visualizations in two dimensions. The types of attributes include chemical input and output and various important performance metrics to determine the operational state of the plant to predict potential faults at several stages in the water treatment process. This contrasts with the adult dataset in that there are significantly fewer instances and no labels for the data, so it is a pure unsupervised learning dataset.

## What makes the datasets interesting?

I found the adult dataset intriguing because I found it to be a practical application of machine learning to be able to make predictions based on U.S. census data. Considering that the census is performed every ten years and gathers a wealth of data, I thought it was an interesting usage to predict expected income, which might be useful in economics research for tracking predicting other information. I'm curious about the intersection of machine learning and finance so this might help gain insight into those types of problems.

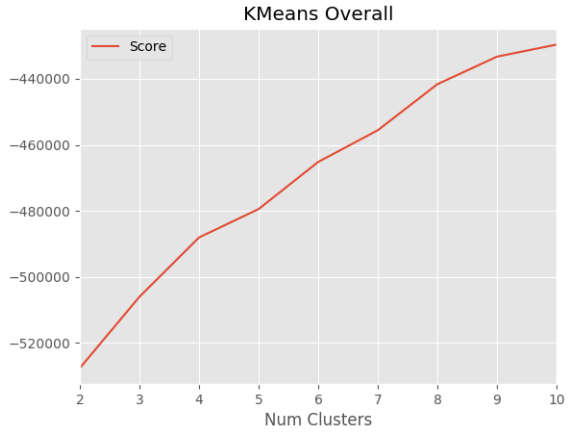
The water treatment plant is especially noteworthy to me because I'm interested in seeing the interaction of chemical processes and machine learning. Particularly, I'm curious to see whether unsupervised learning can help determine risks involved in chemical processes, and whether those ideas could be translated to other processes like nuclear power plants or mass manufacturing. This dataset is extremely small but I see that as an opportunity to witness

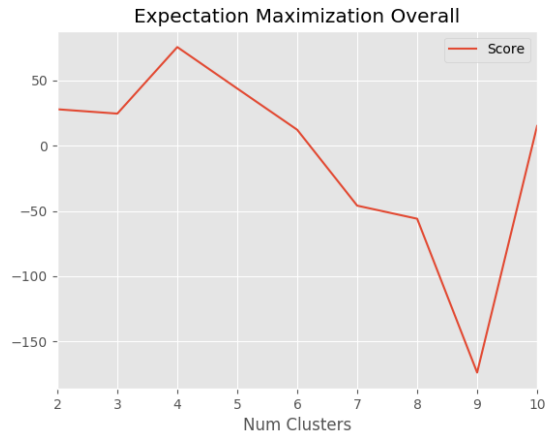
which features lead to various clusterings or how certain features are emphasized. Also, the small size enables great data visualizations.

## Clustering Algorithms and Results

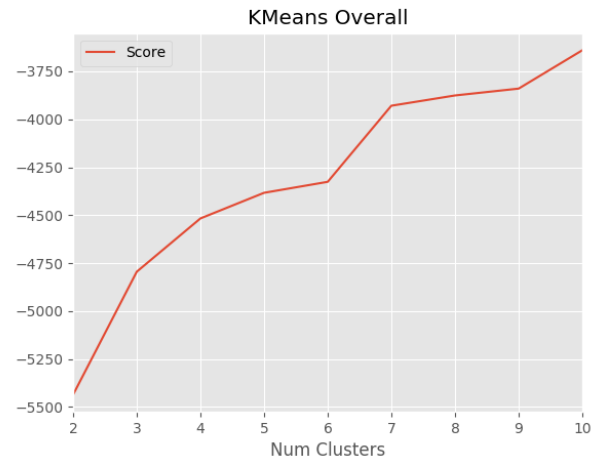
For both datasets, I used scikit-learn's KMeans and GaussianMixture (for expectation maximization) Python modules to cluster the various data sets. I used values of *k* from 2 to 10 for the adult dataset because the initial labeling was a binary classification problem and I wanted to see how increasing *k* would affect the performance. Given that humans are complex beings though, it seemed apt that that there might be several clustering of adults from the census according to various socioeconomic backgrounds. For the water treatment, I also used *k* from 2 to 10 to make that dataset comparable to the adult dataset. I used K-fold cross-validation for both datasets to average out several clustering algorithms. The value of *k* and

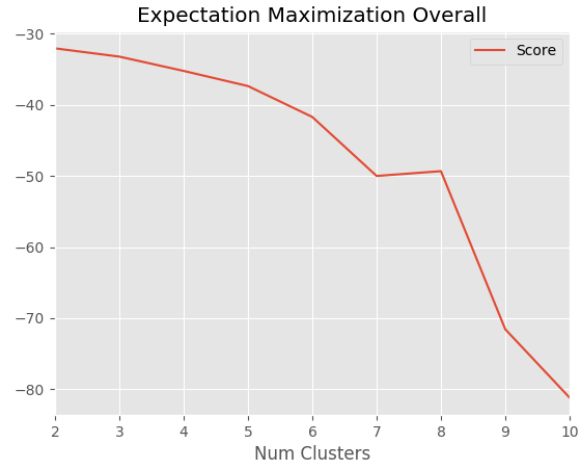
Adult Dataset




Water Treatment Dataset



distance metric used were two important hyperparameters that could reflect domain knowledge about the data. Unfortunately, scikit-learn only supports Euclidean distances for its clustering algorithms but a specifically tailored distance metric might have yielded better results.

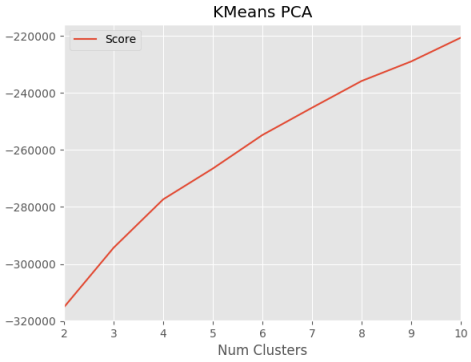
To score the various values of *k* I fitted each KMeans learner to the dataset to determine the centroids. Then, I transformed the test data to label them according to recently learned model and used the negative of the sum of the squared errors (so higher is better) between the points and centroids. Without scaling or normalizing the features, the adult dataset had features around 1e13 but with scaling the distances reduce to 1e6. Using the elbow method, we can estimate that the optimal number of clusters are eight clusters for the adult dataset and seven clusters for the water treatment dataset before we get diminishing marginal returns for increasing complexity.

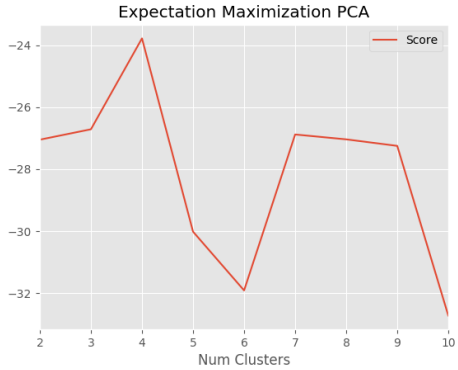
For the GaussianMixture or expectation maximization learner, I used the per-sample average log-likelihood of the data as defined by scikit-learn. For the adult dataset, the error would reach a peak at around four clusters but decrease consistently over the next few numbers before reaching a minimum around 8 or 9. Expectation maximization for the water dataset decreased consistently for greater numbers of clusters which would make sense as greater sharing between clusters means lower probabilities for each point in a cluster and higher log-likelihoods. This contrasts KMeans's score which would likely steadily increase as points aren't shared. From these results, clustering appears to be powerful in that automatically finds new features and information using a given distance metric but without any predetermined label.

**Dimensionality Reduction Algorithms**

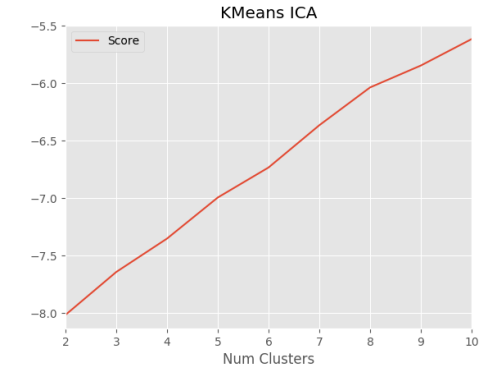
After testing the clustering algorithms, I applied the dimensionality reduction algorithms PCA, ICA, Randomized Projection, and LDA (Latent Dirichlet Allocation). I then applied the clustering algorithms to the reduced datasets so that I could compare their effects on clustering.

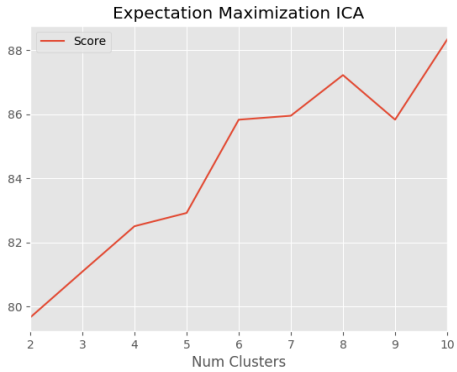
## Adult PCA



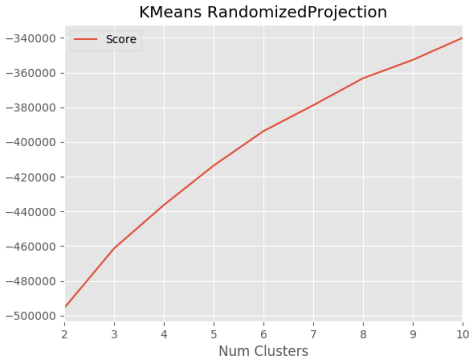



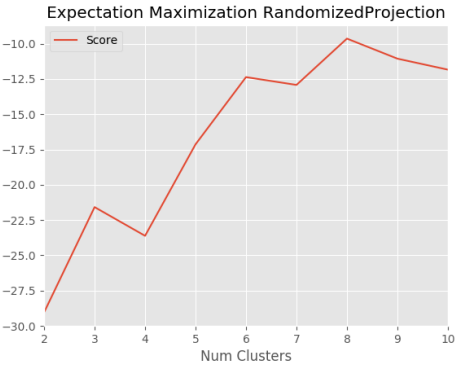

## Adult ICA



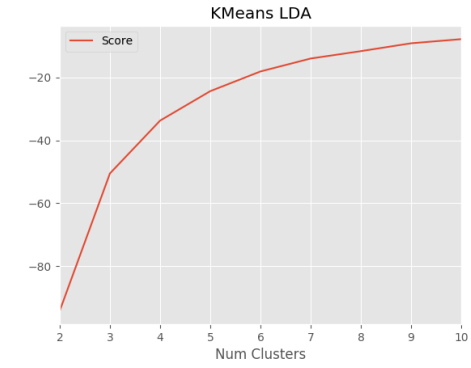



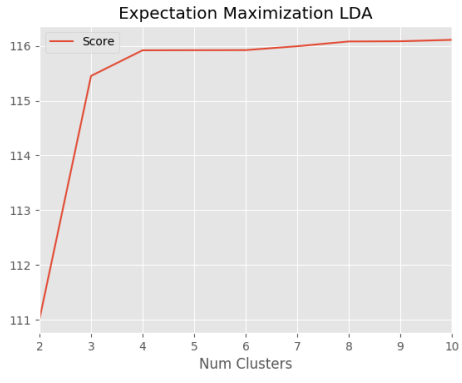

## Adult Randomized Projection

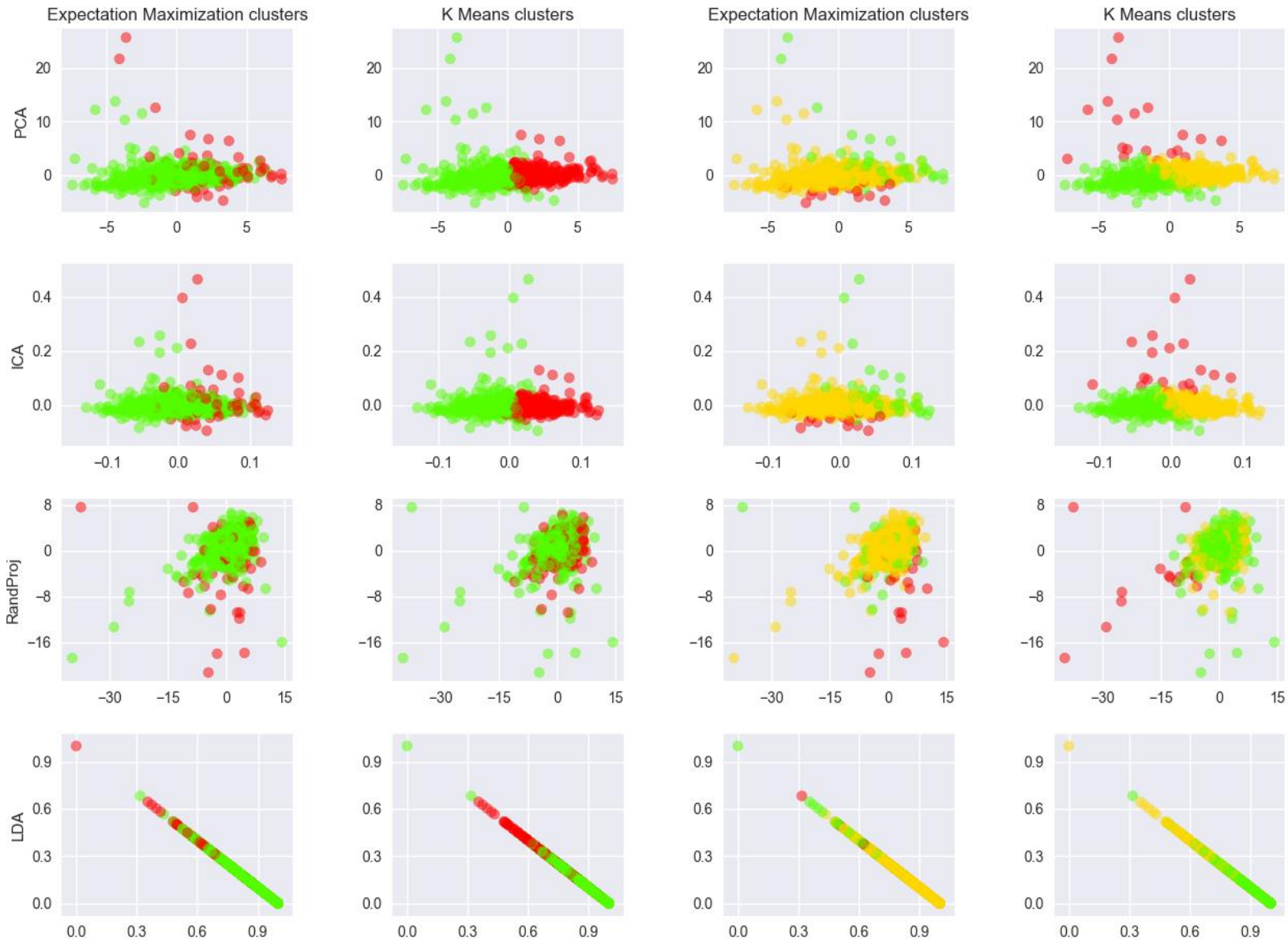



## Adult LDA




Expectation Maximization clusters
K Means clusters
Expectation Maximization clusters
K Means clusters
PCA
ICA
RandProj
LDA

A general assessment of the all the dimensionality reduction algorithms in general would find that, for the most part, all the algorithms improved the raw negative sum of squared errors for KMeans and raw average log-likelihood for expectation-maximization over the original. This increase would be justified by the reduction in the distances of all the instances in lower dimensional space while trying to retain as much information or variance as possible, or even randomly in the case of random projection. It should be noted that the original adult dataset had 14 features which grew to 46 after one-hot encoding all the categorical variables. I then decided to reduce the training set to twenty components because it's about half of the total features. I chose two components for water treatment dataset because I wanted to reduce the thirty-eight continuous variables to two-dimensions for graphical visualizations.

Principal component analysis only performed slightly better than overall for both KMeans and expectation maximization. This score comes in stark contrast to the performance of independent component analysis and its dominant scores over all the dimensionality reduction algorithms for KMeans. The difference underscores the computationally cheaper linear algebraic approach of PCA versus the more expensive but potentially more useful probabilistic and information theory guided approach of ICA. PCA focuses on mutually orthogonal components and maximum variance, which would result in preserving the structure of the initial training set. ICA, on the other hand, focuses less on preservation but rather more on relevance. By minimizing mutual information between constituent components and maximizing mutual information between the input data and the new components, ICA seems to be more relevant from the score charts of the adult dataset.

Juxtaposing ICA and PCA with Randomized Projection and Latent Dirichlet Allocation may yield further insight by exploring alternative approaches. Randomized Projection is fairly straightforward in that it projects the training set onto components chosen at random, in this case sampled from a Gaussian random projection. The main benefit is speed through reduced calculations for the components and working well in practice. However, it performed ever so slightly better than the original training set for KMeans and surprisingly beat PCA in expectation maximization. LDA performed the best of all the algorithms for expectation maximization and second best for KMeans. This performance is surprising as I couldn't use the normalized training

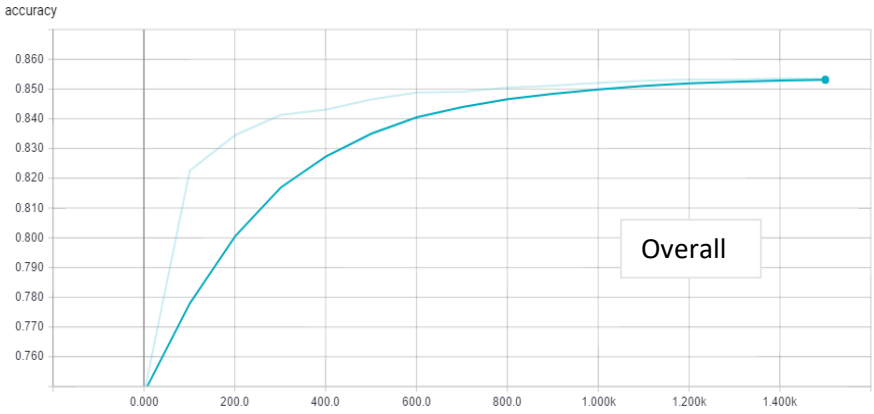
set I had used for the other algorithms because it only accepts positive numbers. By using a generative statistical model that uses unobserved groups to explain observations. This powerful approach yielded very strong performance compared to the other algorithms despite working on an unscaled training set, which is significant as neural networks on the scaled version gained almost 10% in accuracy.

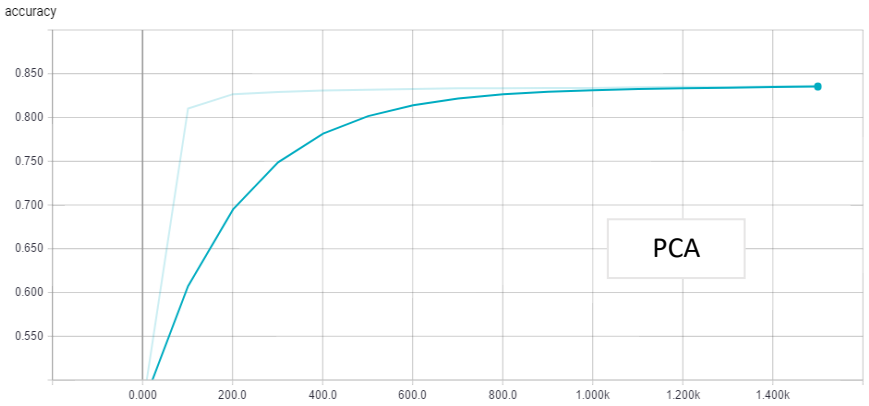
Looking at the cluster visualization for the water treatment dataset, we can immediately see a strong contrast between KMeans and expectation maximization. KMeans tends to favor clusters that are well defined in shape and composition while expectation maximization favors a mixture of the different clusters. However, this difference isn't as pronounced with randomized project compared to the other algorithms. Surprisingly, LDA created a straight line when reducing the dimensionality, possibly due to working on unscaled instances. Also quite unexpected, ICA and PCA produced graphs that look very similar. This might have been a chance occurrence or a result of projecting onto such low dimensional spaces. Between the two datasets, the water dataset had much higher scores, as would be expected because there are far fewer instances and features than in the adult dataset. The curves created were similar however.

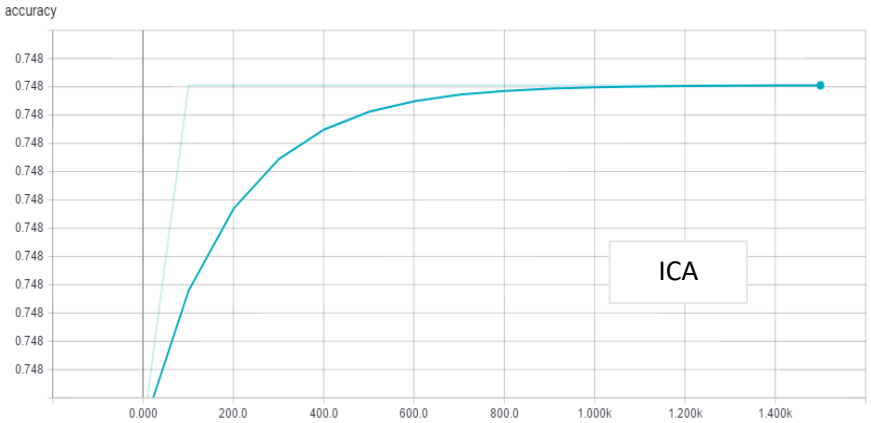
**Neural Networks on Adult Dataset**

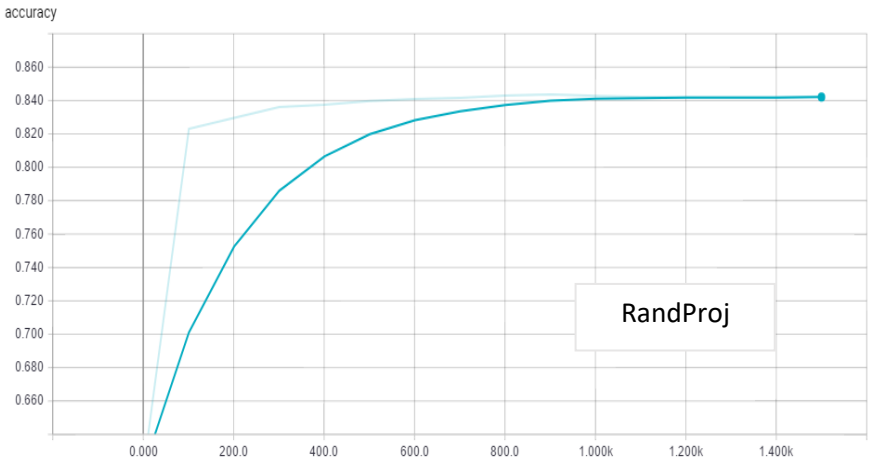
| | Overall | PCA | ICA | RandProj | LDA | Kmeans | EM | Double |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.8511 | 0.8384 | 0.7479 | 0.8344 | 0.7801 | 0.8477 | 0.847 | .8505 |
| Runtime | 10m 47s | 10m 27s | 10m 22s | 10m 32s | 17m 38s | 10m 18s | 10m 16s | 20m 26s |

Applying neural networks to the overall dataset, datasets transformed by dimensionality reduction algorithms, and datasets where clusters are a feature yielded the table above and the following charts. I changed the overall dataset by one-hot encoding the features and normalizing them, which boosted the accuracy by 10% to 85.11% accuracy. I tested each approach several times and averaged their accuracy and runtimes. I used Google's TensorFlow library with two layers of ten nodes each as in the previous assignment with the Adam optimizer for 1500 epochs.

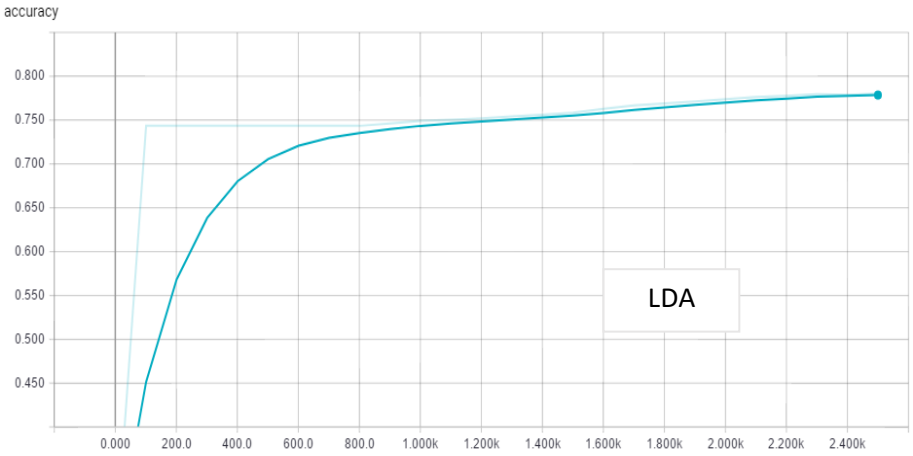


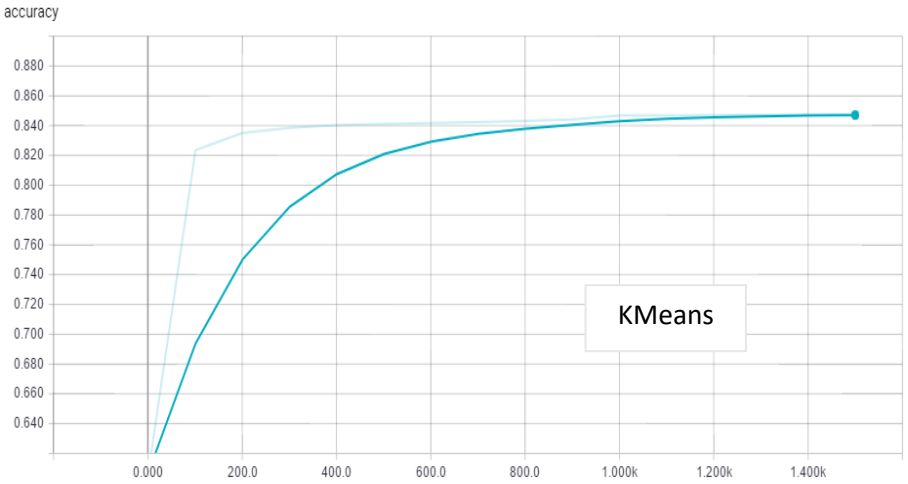



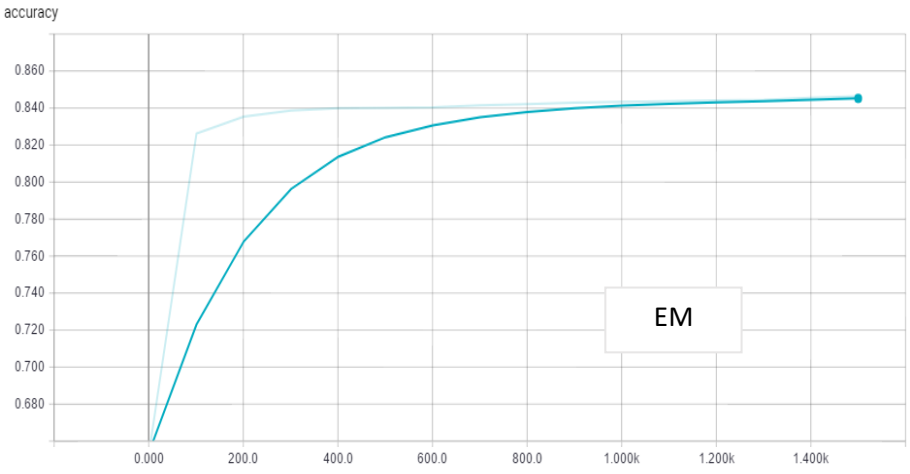




I'll begin by comparing the dimensionality reduction algorithms and the neural networks trained on them. I was honestly surprised the results. From the outcomes of the clustering algorithms, I thought ICA would outperform PCA with the neural networks. Instead, however, PCA significantly outperformed ICA and was 1.27% off from the original dataset. It makes sense that both would be lower as there is some loss of information between datasets. I believe that ICA performed so badly because it couldn't find components that were better than the initial features. PCA would be better suited as it tries to preserve the structure of the training data.




Another unexpected result was that randomized projection performed much better than ICA and LDA which I didn't expect given how simple randomized projection is. It probably preserved some structure of the instances better than ICA and LDA which is quite impressive considering the reduction time was significantly less than that of ICA and LDA. As mentioned before, LDA performed worse at 78.01% but that's better than a neural network on an unscaled one-hot encoded overall dataset at 75.12%. In terms of time, the neural networks took almost the same amount of time for each of the reduced datasets except LDA due to the aforementioned problem. Randomized projection was the quickest to transform the data, followed by PCA, then ICA, and then LDA which took a

significant amount of time and processing power. Doubling the amount of data didn't change the accuracy much but did double the training time.

Adding the clusters to the overall dataset produced slightly worse results than overall dataset by itself. This was also a surprise as I expected clustering to add more information than was already known. It appears that clustering would just be a condensation of previously known information and would just add unnecessary values for the neural network to train on. The accuracy wavered between 84.4-84.7% accuracy for 3, 6, and 8 clusters for both KMeans and expectation maximization over multiple runs. These outcomes might lead to the conclusion that clustering could lead to better insight and understanding of the data but not necessarily better accuracy for labeled data.